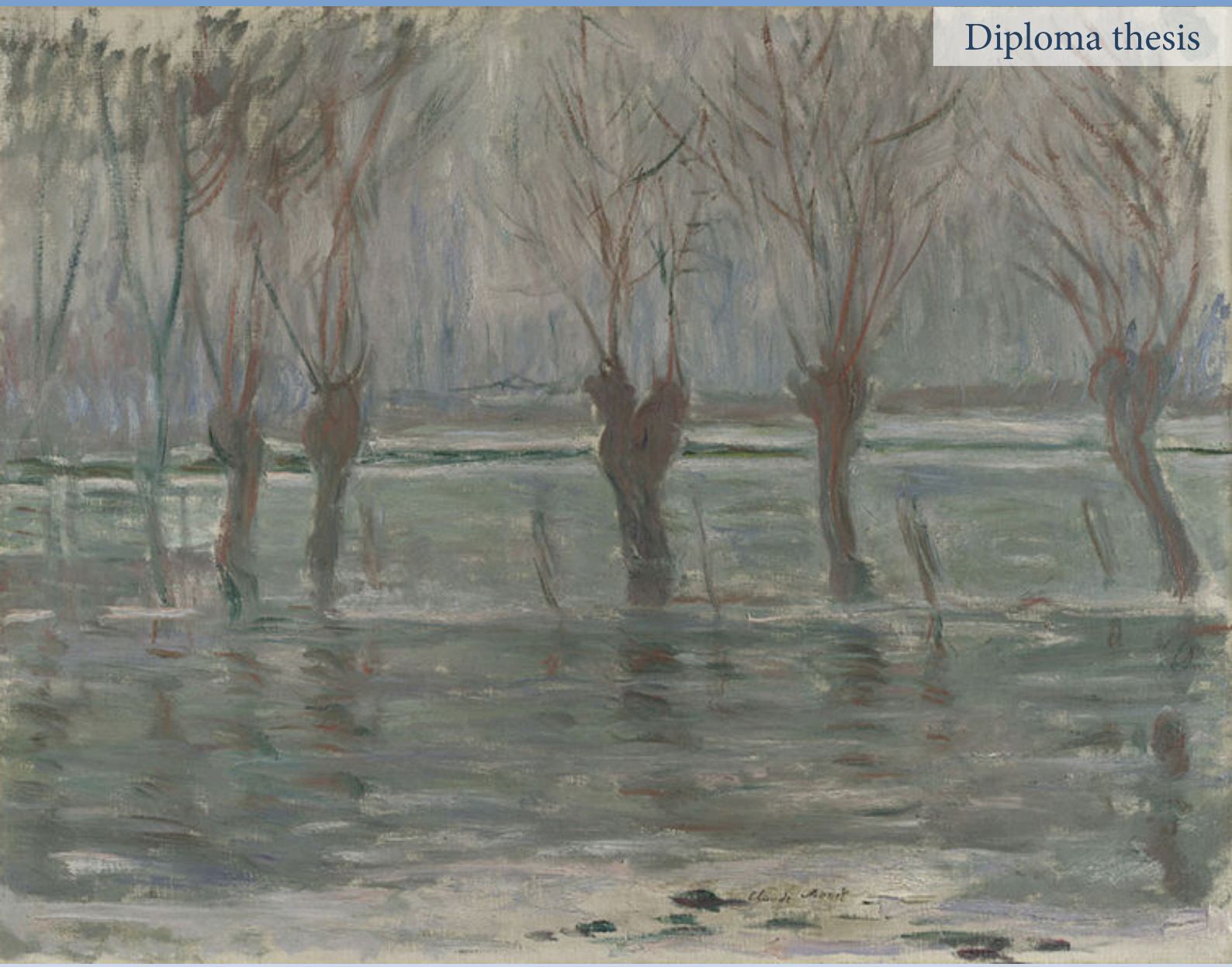
National Technical University of Athens
School of Civil Engineering
Department of Water Resources and Environmental Engineering

Spatiotemporal clustering of streamflow extremes and relevance for flood insurance

Konstantinos Papoulakos



Diploma thesis

Supervisor: Demetris Koutsoyiannis, Professor, NTUA
Co-supervisors: Theano Iliopoulou, PhD, NTUA
Panayiotis Dimitriadis, PhD, NTUA

Athens, November 2021

NATIONAL TECHNICAL UNIVERSITY OF ATHENS

SCHOOL OF CIVIL ENGINEERING

DEPARTMENT OF WATER RESOURCES AND ENVIRONMENTAL ENGINEERING

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΧΩΡΟΧΡΟΝΙΚΗ ΟΜΑΔΟΠΟΙΗΣΗ ΤΩΝ ΑΚΡΑΙΩΝ ΠΑΡΟΧΩΝ ΚΑΙ Η ΣΗΜΑΣΙΑ ΤΗΣ ΣΤΗΝ ΑΣΦΑΛΙΣΗ ΕΝΑΝΤΙ ΠΛΗΜΜΥΡΩΝ

DIPLOMA THESIS

SPATIOTEMPORAL CLUSTERING OF STREAMFLOW EXTREMES AND RELEVANCE FOR FLOOD INSURANCE

KONSTANTINOS PAPOULAKOS

SUPERVISOR:

DEMETRIS KOUTSOYIANNIS, PROFESSOR, NTUA

CO-SUPERVISORS:

THEANO ILIOPOULOU, PHD, NTUA

PANAYIOTIS DIMITRIADIS, PHD, NTUA

ATHENS, NOVEMBER 2021

Floods are 'acts of God', but flood losses are largely acts of man.

Gilbert F. White (1911-2006)

Ευχαριστίες / (Acknowledgments in Greek)

Με την εκπόνηση και την παρουσίαση της παρούσας διπλωματικής εργασίας κλείνει ο κύκλος των προπτυχιακών μου σπουδών στη Σχολή Πολιτικών Μηχανικών ΕΜΠ, ολοκληρώνοντας ένα σημαντικό κεφάλαιο της έως τώρα ζωής μου και σηματοδοτώντας την αρχή νέων προκλήσεων.

Παρά τις παθογένειες που χαρακτηρίζει διαχρονικά το «Ελληνικό Πανεπιστήμιο», νιώθω ιδιαίτερα χαρούμενος και ευγνώμων που υπήρξα μέλος αυτού του φημισμένου στην Ελλάδα και το εξωτερικό τεχνολογικού Ιδρύματος. Κάνοντας ανασκόπηση των εμπειριών που αποκόμισα και των ανθρώπων που γνώρισα κατά τη διάρκεια των σπουδών μου, συνειδητοποιώ ότι αυτό το ταξίδι με διαμόρφωσε ανεξίτηλα ως άνθρωπο και ως χαρακτήρα, προσφέροντάς μου επιστημονική γνώση, οξύνοντας την κριτική μου ικανότητα και διευρύνοντας τους ορίζοντές μου προς κάθε κατεύθυνση.

Πρωτίστως θα ήθελα να ευχαριστήσω από καρδιάς τον κ. Δημήτρη Κουτσογιάννη, Καθηγητή της Σχολής Πολιτικών Μηχανικών ΕΜΠ και επιβλέποντα της διπλωματικής μου εργασίας, για την ανάθεση ενός εξαιρετικά ενδιαφέροντος θέματος. Πέρα από τον δικαιολογημένο θαυμασμό που πάντα ένιωθα για αυτόν λόγω των επιστημονικών του επιτευγμάτων και της διδακτικής του δεινότητας, ο κος Κουτσογιάννης αποτελεί πηγή έμπνευσης για κάθε νέο μηχανικό και επιστήμονα. Παρά τη στοχαστικότητα του επιστημονικού πεδίου που μελετά, η καθαρότητα των ιδεών και των απόψεών του είναι βαθιά ντετερμινιστική χάρη στην αταλάντευτη πίστη του για ελεύθερη γνώση, προσβάσιμη έρευνα και τις αξίες του σχετικά με την επιστήμη και την τεχνολογία που βάζουν στο επίκεντρο τον άνθρωπο. Τον ευχαριστώ ειλικρινά για τη συνεργασία, τις γνώσεις και τις αρχές που μου μεταλαμπάδευσε.

Στη συνέχεια, θα ήθελα να ευχαριστήσω θερμά τη διδάκτορα Άνυ Ηλιοπούλου για την επιστημονική γνώση που μοιράστηκε μαζί μου απλόχερα, τον πολύτιμο χρόνο της και την εν γένει καθοριστική συμβολή της ώστε να φτάσει η εργασία στην τελική της μορφή. Επίσης, θέλω να ευχαριστήσω τον διδάκτορα Παναγιώτη Δημητριάδη για τις πολύτιμες και καίριες παρατηρήσεις του καθ' όλη τη διάρκεια εκπόνησης της εργασίας. Νιώθω ιδιαιτέρως τυχερός που συνεργάστηκα με τόσο αξιόλογους νέους ανθρώπους και επιστήμονες, οι οποίοι κοσμούν το ΕΜΠ με την παρουσία τους και δείχνουν ότι το μέλλον της έρευνας και της τεχνολογίας στη χώρα μας είναι λαμπρό.

Ακόμα, οφείλω ένα εγκάρδιο ευχαριστώ στον κ. Δημοσθένη Τσακνιά, Πολιτικό Μηχανικό με πολυετή επαγγελματική εμπειρία στον κλάδο των ασφαλειών έναντι πλημμύρας σε Μεγάλη Βρετανία, Ελβετία και Ελλάδα, ο οποίος μοιράστηκε μαζί μου με τρόπο γενναιόδωρο τις εμπειρίες και τις γνώσεις του στο επιστημονικό αυτό αντικείμενο, τόσο από ερευνητική όσο και πρακτική σκοπιά, προσφέροντάς μου την απαραίτητη θεμελιώδη γνώση και αντίληψη που απαιτούνταν για να προσεγγίσω το εν λόγω

πεδίο. Οι πολύωρες συζητήσεις μαζί του, παρά τον ιδιαιτέρως υψηλό φόρτο εργασίας του, αποτέλεσαν για εμένα πυξίδα προσανατολισμού κατά τη διαδικασία προσέγγισης του συγκεκριμένου θέματος, δίνοντάς μου ερμηνείες, διαφωτίζοντας τυφλά σημεία και παρέχοντάς μου υπολογιστικά και θεωρητικά εργαλεία αντιμετώπισης των πολυεπίπεδων προκλήσεων οι οποίες εμφανίστηκαν κατά την εκπόνηση της εργασίας.

Τέλος, το πιο μεγάλο ευχαριστώ το οφείλω στην οικογένειά μου, τους γονείς μου Γιώργο και Μαρία, τα αδέρφια μου Μιχάλη, Γιώργο και Πηνελόπη, και την Ειρήνη για την συνεχή υποστήριξή που μου παρέχουν σε οτιδήποτε με το οποίο καταπιάνομαι, την αγάπη, την υπομονή και τη σιγουριά που μου προσφέρουν, νιώθοντάς τους συνεχώς κοντά μου.

Κωνσταντίνος Παπουλάκος

Αθήνα, Νοέμβριος 2021

Abstract

Recent research has revealed the significance of Hurst-Kolmogorov dynamics (Koutsoyiannis, 2011), which is characterized by strong correlation and high uncertainty in large scales (Dimitriadis and Koutsoyiannis, 2015), in flood risk assessment as for example in inundated flood duration (Dimitriadis and Koutsoyiannis, 2020). However, classic risk estimation for flood insurance practices is formulated under the assumption of temporal independence of extreme flood events, which is unlikely to be tenable in real-world hydrometeorological processes exhibiting long range dependence (Iliopoulou and Koutsoyiannis, 2019). Additionally, insurable flood losses are considered as ideally independent and non-catastrophic in financial terms due to the widely spread perception of limited risk regarding catastrophically large flood losses.

As the accurate risk assessment is a fundamental part of flood insurance and reinsurance practices, this study investigates the effects of lack of fulfillment of these assumptions, paving the way for a deeper understanding of the underlying clustering mechanisms of streamflow extremes. For this purpose, a spatiotemporal analysis of the daily flow series from the US-CAMELS dataset (Newman et al., 2014) is applied, comprising the impacts of clustering mechanisms on return intervals, duration and severity of the over-threshold events which are treated as proxies for collective risk. Moreover, stochastic approaches are developed and an exploratory analysis is introduced regarding the stochastic aspects of the correlation between the properties of the extreme events and the actual claims records of the FEMA National Flood Insurance Program which are recently published.

Furthermore, regarding precipitation mechanisms, the presence of persistence in annual rainfall is expected to induce clustering in rainfall extremes (Iliopoulou et al., 2018), which should be manifested by clustering of floods. Therefore, in the framework of a case study, it is investigated whether the collective risk estimated using the former as a proxy, i.e. the magnitude of the rainfall peak over threshold events in a year, is correlated with the actual compensations given.

Eventually, the current insurance practices and actual compensations given in the agriculture domain in Greece are reviewed, while inspecting the underlying stochastic assumptions and evaluating changes in the estimated risk in the case that these assumptions are not valid.

Key words: Insurance; Hurst-Kolmogorov dynamics; Clustering mechanisms; Symmetric Moving Average; Climacogram; Monte-Carlo simulation; Risk assessment; Extreme value analysis; Collective risk; Return interval; Floods; Streamflow extremes; Precipitation; Rainfall extremes; FEMA; NFIP; Greece; Agriculture.

Εκτεταμένη περίληψη στα Ελληνικά / Extended abstract in Greek

Η επιστημονική έρευνα έχει αποκαλύψει τη σημασία της δυναμικής Hurst-Kolmogorov (Koutsoyiannis, 2011), η οποία χαρακτηρίζεται από ισχυρή συσχέτιση και υψηλή αβεβαιότητα στις μεγάλες κλίμακες (Dimitriadis and Koutsoyiannis, 2015), στην εκτίμηση πλημμυρικού κινδύνου, όπως για παράδειγμα στο πεδίο της διάρκειας πλημμυρικών γεγονότων (Dimitriadis and Koutsoyiannis, 2020). Παρ' όλα αυτά, οι κλασσικές μέθοδοι εκτίμησης ρίσκου στον τομέα των ασφαλειών έναντι πλημμύρας διατυπώνονται υπό την παραδοχή της ανεξαρτησίας των ακραίων πλημμυρικών γεγονότων στον χρόνο, η οποία όμως δεν συνάδει με τις πραγματικές υδρομετεωρολογικές διαδικασίες οι οποίες παρουσιάζουν μακροπρόθεσμη εμμονή (Iliopoulou and Koutsoyiannis, 2019). Επιπροσθέτως, οι ασφαλιστέες απώλειες λόγω πλημμύρας θεωρούνται ανεξάρτητες και μη καταστροφικές (σε οικονομικούς όρους) για τις ασφαλιστικές εταιρείες εξαιτίας της ευρεώς διαδεδομένης αντίληψης του περιορισμένου κινδύνου σχετικά με τις καταστροφικά μεγάλες απώλειες από πλημμυρικά γεγονότα.

Μιας και η ακριβής εκτίμηση του ρίσκου είναι θεμελιώδης διαδικασία που ακολουθείται στον κλάδο των ασφαλειών και αντασφαλειών έναντι πλημμύρας, η παρούσα εργασία διερευνά τις επιπτώσεις της μη ικανοποίησης των παραπάνω υποθέσεων, προλειαίνοντας τον δρόμο για μια βαθύτερη κατανόηση των υποκείμενων μηχανισμών ομαδοποίησης των ακραίων πλημμυρικών παροχών. Έχοντας αυτό το σκοπό, η εργασία εφαρμόζει μια χωροχρονική ανάλυση των χρονοσειρών παροχών της βάσης δεδομένων US-CAMELS (Newman et al., 2014) μέσω της μεθόδου των υπερβάσεων πάνω από ένα κατώφλι (peak-over-threshold method). Αναλυτικότερα, για κάθε έναν από τους εξεταζόμενους σταθμούς μέτρησης της US-CAMELS, υπολογίστηκε το αποκαλούμενο συλλογικό ρίσκο S (collective risk), το οποίο είναι μια συνήθης πρακτική που ακολουθείται στον κλάδο των ασφαλειών και αντασφαλειών. Σε οικονομικούς όρους, το συλλογικό ρίσκο S ορίζεται ως το άθροισμα (συσσώρευση) ποσών απαίτησης (claim amounts) σε ετήσια χρονική βάση. Στον τομέα της υδρολογίας, πρόσφατα προτάθηκε ότι οι υπερβάσεις των παροχών ποταμών πάνω από ένα κατώφλι μπορεί να θεωρηθεί ότι αντιπροσωπεύουν τα ποσά απαίτησης λόγω οικονομικών απωλειών που οφείλονται σε πλημμυρικά φαινόμενα (Serinaldi and Kilsby, 2016).

Επιπροσθέτως, πέρα από την επίδραση των μηχανισμών ομαδοποίησης στο συλλογικό ρίσκο S, εξετάστηκε η επιρροή που έχουν αυτοί οι μηχανισμοί σε συγκεκριμένα χαρακτηριστικά των πλημμυρικών γεγονότων που ξεπερνούσαν ένα επιλεχθέν κατώφλι, όπως στη διάρκεια, την ένταση και τα διαστήματα επαναφοράς τους. Επιπλέον, αναπτύχθηκαν στοχαστικές προσεγγίσεις που

αφορούσαν τη συσχέτιση ανάμεσα στο συλλογικό ρίσκο *S*, σε συγκεκριμένες ιδιότητες των ακραίων πλημμυρικών γεγονότων και στα πραγματικά ιστορικά στοιχεία αποζημιώσεων τα οποία δημοσίευσε πρόσφατα η Ομοσπονδιακή Υπηρεσία Διαχείρισης Έκτακτων Αναγκών των ΗΠΑ (FEMA, 2019) στο πλαίσιο του Εθνικού Προγράμματος Ασφάλισης Έναντι Πλημμυρών (NFIP). Όπως θα παρουσιαστεί και παρακάτω, αυτή η αντιπαραβολή των ευρημάτων της παρούσας εργασίας με τα ιστορικά στοιχεία αποζημιώσεων προσέφερε σημαντικά συμπεράσματα τα οποία πράγματι μας επιτρέπουν να αξιολογήσουμε χωροχρονικά στις ΗΠΑ τις παραδοχές που γίνονται στον κλάδο των ασφαλειών και οι οποίες αφορούν την ανεξαρτησία μεταξύ των ακραίων πλημμυρικών γεγονότων, όπως αναπτύχθηκαν στην πρώτη παράγραφο.

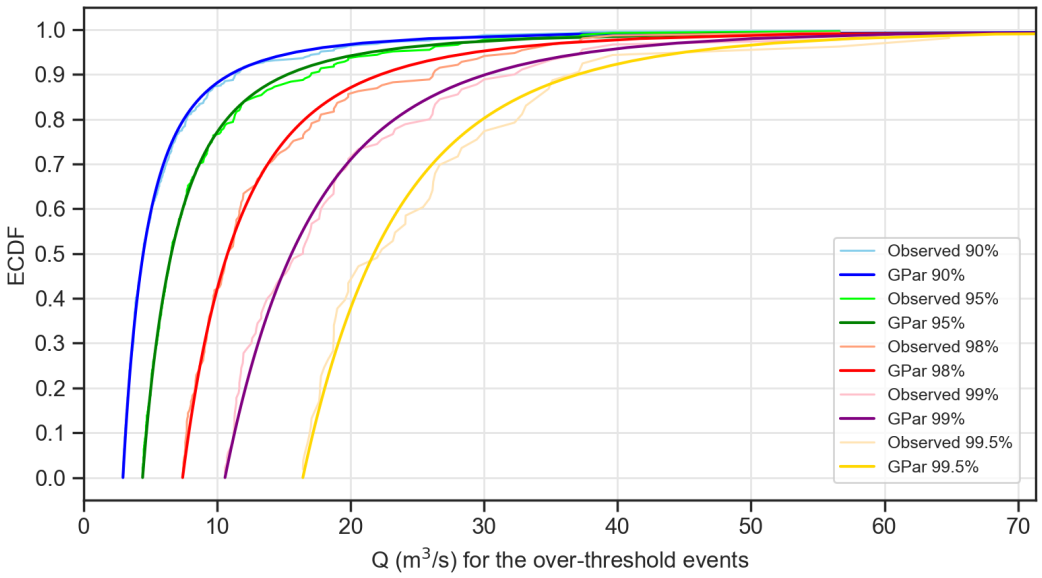
Ακόμα, σχετικά με τους μηχανισμούς κατακρήμνισης, η ύπαρξη εμμονής στα ετήσια μεγέθη βροχής δημιουργεί την προσδοκία ανάλογης συμπεριφοράς στα μέγιστα των βροχοπτώσεων (Iliopoulou et al., 2018), φαινόμενο το οποίο αναμένεται να εκδηλωθεί και στους μηχανισμούς ομαδοποίησης πλημμυρών. Για το λόγο αυτό, στο πλαίσιο μιας μελέτης περίπτωσης, διερευνάται ο βαθμός συσχέτισης του συλλογικού ρίσκου *S*, το οποίο πάλι θεωρούμε ότι μπορεί να εκτιμηθεί ότι αντιπροσωπεύεται από το άθροισμα των βροχοπτώσεων πάνω από ένα κατώφλι, με στοιχεία πραγματικών αποζημιώσεων λόγω πλημμύρας που έχουν δοθεί στον Ελλαδικό χώρο.

Τέλος, σε ένα γενικότερο πλαίσιο, εξετάστηκαν οι υφιστάμενες πρακτικές ασφαλίσεων έναντι πλημμύρας και τα πραγματικά στοιχεία αποζημιώσεων στον αγροτικό τομέα στην Ελλάδα, με παράλληλη παρουσίαση των υποκείμενων στοχαστικών παραδοχών που προκύπτουν από αυτή την ανάλυση, αξιολογώντας τις μεταβολές που προκύπτουν σε περίπτωση αναίρεσης τους.

Μεταξύ των παραδοσιακών στρατηγικών μοντελοποίησης που εφαρμόζονται στον ασφαλιστικό κλάδο, εξέχουσα θέση καταλαμβάνουν οι κατανομές ακραίων τιμών (extreme value analysis distributions), στις οποίες συμπεριλαμβάνονται οι κατανομές γενικευμένη ακραίων τιμών (generalized extreme value, GEV) και γενικευμένη Pareto (generalized Pareto, GPD), που χρησιμοποιούνται ως ένα εργαλείο για την στατιστική ανάλυση μεγίστων ή ελαχίστων και των υπερβάσεων πάνω από ένα κατώφλι.

Η εισαγωγή αυτών των κατανομών στην παρούσα εργασία, πέρα από την ποιοτική διερεύνηση της συμπεριφοράς τους, μας προσέφερε τη δυνατότητα να αξιολογήσουμε την επίδραση που διαδραματίζει το επιλεχθέν κατώφλι στη μέθοδο των υπερβάσεων. Αυτό συμβαίνει γιατί, από τη μια μεριά, επιδιώκουμε να επιλέξουμε ένα υψηλό κατώφλι ώστε να εξετάσουμε τις πραγματικά υψηλές τιμές παροχών που είναι ικανές να επιφέρουν σημαντικές οικονομικές απώλειες, από την άλλη όμως μεριά, είναι απαραίτητο να έχουμε στη διάθεσή μας πολλές τιμές πάνω από ένα

κατώφλι με σκοπό να υπάρχουν αρκετά δεδομένα ώστε να μην αλλοιώνεται η στατιστική τους συμπεριφορά, προκαλώντας πιθανά στρεβλά συμπεράσματα. Συνεπώς, ανάλογα με το μέγεθος της διατιθέμενης ιστορικής (παρατηρούμενης) χρονοσειράς, η απόφαση σχετικά με την τιμή που ορίζεται ως κατώφλι είναι εξαιρετικά σημαντική. Στην εργασία αυτή εξετάζονται οι εξής τιμές κατωφλίου: 90%, 95%, 98% και 99%.



Σχήμα 1: Διάγραμμα που παρουσιάζει την επίδραση του κατωφλίου στην εμπειρική αθροιστική συνάρτηση κατανομής (ECDF) των ακραίων παροχών πάνω από το εν λόγω κατώφλι σε έναν συγκεκριμένο σταθμό (ID: 01552500).

Η κατανομή των συνολικών ποσών απαίτησης, θεωρώντας ότι το χαρτοφυλάκιο μιας ασφαλιστικής εταιρείας είναι συλλογικό και ικανό να παράξει N αριθμό αιτημάτων προς αποζημίωση σε ένα πεπερασμένο χρονικό διάστημα, μπορεί να περιγραφεί από το μοντέλο συλλογικού ρίσκου (Kaas et al., 2008). Το συλλογικό ρίσκο S_x (collective risk) ορίζεται ως

$$S_x = X_1 + X_2 + \cdots + X_N, \tag{0.1}$$

όπου X_i είναι η i τιμή ποσού απαίτησης. Ο όρος S_x αντιστοιχεί στο πραγματικό ποσό απαίτησης. Προφανώς, $S_x = 0$ αν $N = 0$. Όμοια, στην περίπτωση ασφαλειών λόγω ενός ακραίου πλημμυρικού γεγονότος, το συλλογικό ρίσκο S είναι το συνολικό ποσό απαίτησης, θεωρώντας ξανά ένα χαρτοφυλάκιο ασφαλιστέων περιουσιών ως συλλογικό που παράγει έναν τυχαίο αριθμό N αιτημάτων προς αποζημίωση σε ένα πεπερασμένο χρονικό διάστημα (ενός έτους στην περίπτωσή μας). Δηλώνοντας ως y_t τα στοιχεία μιας χρονοσειράς, έχει προταθεί από τους Serinaldi and Kilsby (2016) ότι ως ένα μέγεθος το οποίο αντιπροσωπεύει χρονικά το συλλογικό ρίσκο S μπορεί να οριστεί το

$$S = \sum_{j=1}^{N} Y_j \qquad (0.2)$$

όπου Y_j είναι η j αντιπροσωπευτική τιμή του ποσού απαίτησης (τιμή παροχής πάνω από το κατώφλι). Ξανά, τα συνολικά ποσά απαίτησης $S = 0$ αν $N = 0$. Στην περίπτωση διερεύνησης ακραίων τιμών βροχής, θεωρούνται οι τιμές βροχής που είναι πάνω από το κατώφλι. Αν και ο ορισμός του συλλογικού ρίσκου που σχετίζεται με τον ασφαλιστικό κλάδο έναντι πλημμύρας είναι ουσιαστικά μια αντιπροσωπευτική τιμή του πραγματικού συλλογικού ρίσκου, μιας και βασίζεται σε υδρολογικές χρονοσειρές και όχι σε πραγματικά ποσά απαίτησης, στην παρούσα εργασία αποκαλείται και αυτό ως συλλογικό ρίσκο S.

Αυτός ο ορισμός ενέχει την παραδοχή, που εφαρμόζεται ευρέως στον κλάδο των ασφαλειών, ότι το συνολικό συλλογικό ρίσκο S προϋποθέτει ότι ο αριθμός αιτημάτων N και τα επιμέρους ποσά απαίτησης Y_j είναι ομοιόμορφα κατανεμημένα και ότι το N και τα συνολικά Y_j είναι ανεξάρτητα μεταξύ τους. Όπως έχει αναφερθεί και παραπάνω, αυτή η εργασία αξιολογεί την επίδραση της συσχέτισης και των μηχανισμών χωροχρονικής ομαδοποίησης στους υπολογισμούς, διερευνώντας την ορθότητα ή μη των παραπάνω υποθέσεων.

Αρχικά, για να χαρακτηρίσουμε την εξάρτηση και να αξιολογήσουμε τους μηχανισμούς ομαδοποίησης, είναι σημαντικό να ποσοτικοποιήσουμε τη διαφορά που παρουσιάζουν οι ιστορικές χρονοσειρές με μια ακολουθία ανεξάρτητων μεταβλητών. Μια μέθοδος που ακολουθείται συχνά για τη δημιουργία αυτών των ανεξάρτητων μεταβλητών έγκειται στην τυχαιοποίηση των στοιχείων μιας χρονοσειράς (ανακατεύοντας τυχαία τα στοιχεία της) με σκοπό τη παραγωγή μιας νέας η οποία θα έχει τα ίδια στατιστικά χαρακτηριστικά αλλά όχι χρονική συσχέτιση. Η ποσοτικοποίηση της διαφοράς μεταξύ της ιστορικής και της ανεξάρτητης χρονοσειράς γίνεται συγκρίνοντας συγκεκριμένα χαρακτηριστικά που προκύπτουν από αυτές, όπως το ετήσιο συλλογικό ρίσκο S, τη διάρκεια των γεγονότων που περνούν ένα συγκεκριμένο κατώφλι ή τη συχνότητα εμφάνισης ενός θεωρούμενου πλημμυρικού γεγονότος. Για το λόγο αυτό, για κάθε μία από τις ιστορικές χρονοσειρές που εξετάστηκαν, παρήχθησαν 100 νέες τυχαιοποιημένες (ανεξάρτητες) χρονοσειρές.

Στη συνέχεια, αναζητήθηκε η ύπαρξη της δυναμικής Hurst-Kolmogorov (HK), δηλαδή μακροπρόθεσμης εξάρτησης ή εμμονής, που, όπως έχει δείξει η επιστημονική έρευνα, κυριαρχεί σε πολλές φυσικές διεργασίες, όπως στους μηχανισμούς που σχετίζονται με τη παροχή ποταμών και τη βροχόπτωση. Με τον τρόπο αυτό διερευνήθηκαν οι υποκείμενες στοχαστικές διεργασίες που

επικρατούν με σκοπό τη βαθύτερη κατανόηση πιθανών κυρίαρχων μοτίβων. Η εμμονή ή το φαινόμενο Hurst μπορεί να ποσοτικοποιηθεί με τον συντελεστή Hurst *H*. Αναλυτικότερα, για:

- $0 \leq H < 0.5$, η διαδικασία χαρακτηρίζεται από αντι-εμμονή (αρνητική συσχέτιση),
- $H = 0.5$, η διαδικασία είναι ισοδύναμη με λευκό θόρυβο, που σημαίνει ότι δεν υπάρχει μακροπρόθεσμη εξάρτηση ή εμμονή στο δείγμα,
- $0.5 < H \leq 1$, η διαδικασία επιδεικνύει μακροπρόθεσμη εμμονή (ή θετικά συσχετιζόμενη), που είναι μια από τις πιο συνήθεις συμπεριφορές στις υδροκλιματικές διαδικασίες.

Για τον υπολογισμό του συντελεστή Hurst *H* και τον εντοπισμό της μακροπρόθεσμης εμμονής μιας διαδικασίας, η πιο ακριβής μέθοδος είναι μέσω της παραγωγής του *Κλιμακογράμματος* (Koutsoyiannis, 2010; Dimitriadis and Koutsoyiannis, 2015), το οποίο είναι ένα διδιάστατο γράφημα που αποτυπώνει σε λογαριθμικούς άξονες τη διακύμανση $\gamma(k)$ της μέσης συναθροισμένης σειράς της τυχαίας μεταβλητής *Z* στον κατακόρυφο άξονα και της συναθροισμένης κλίμακας *k* στο οριζόντιο άξονα:

$$Z_u^{(k)} = \frac{1}{k} \sum_{i=(u-1)k}^{uk} Z_i \tag{0.3}$$

όπου οι μεταβλητές *Z* και Z_u αναπαριστούν τη στοχαστική ανέλιξη και τη μέση στοχαστική ανέλιξη αντίστοιχα, ενώ u είναι το διάνυσμα-δείκτης του πεδίου, που δείχνει την υστέρηση, δηλαδή τη θέση στο πεδίο.

Σε μερικές περιπτώσεις, όπως συνέβη και σε αυτή την εργασία, η προσαρμογή της ευθείας γραμμής στο *Κλιμακόγραμμα* που προέρχεται από τα ιστορικά δεδομένα δεν μπορεί να αποτυπώσει την πλήρη συμπεριφορά της διακύμανσης της διαδικασίας σε όλο το εύρος των κλιμάκων. Για αυτό το λόγο, εφαρμόστηκε η γενικευμένη ανέλιξη HK (generalized-HK, GHK), η οποία επίσης παρουσιάζει συμπεριφορά HK στις μεγάλες κλίμακες αλλά είναι πιο ευέλικτη στις μικρές κλίμακες. Είναι μια μέθοδος που διατηρεί ρητά (για πλήρη αναλυτικό υπολογισμό) τις τέσσερις κλασσικές ροπές μιας διαδικασίας κάθε είδους για κάθε μια από τις δομές συσχέτισης 2ας τάξης (Dimitriadis and Koutsoyiannis, 2018). Το *Κλιμακόγραμμα* αυτού του μοντέλου είναι:

$$\gamma(k) = \frac{\lambda}{(1 + k/q)^{2-2H}} \tag{0.4}$$

όπου η παράμετρος Hurst *H* παίρνει τιμές από μηδέν έως ένα, το q είναι θετικό, ενώ το λ και το q έχουν διαστάσεις $[x^2]$ και T.

Επιπλέον, σε αυτή την εργασία εφαρμόστηκε η μέθοδος symmetric moving average (SMA) (Koutsoyiannis 2000 and 2016; Dimitriadis and Koutsoyiannis, 2018) ώστε να αναπτυχθούν και να αξιολογηθούν πιθανές στρατηγικές μοντελοποίησης. Η μέθοδος SMA είναι μια γενική μέθοδος για την παραγωγή συνθετικών χρονοσειρών μιας φυσικής ποσότητας διατηρώντας τις δομές συσχέτισης. Πιο συγκεκριμένα, το σύστημα παραγωγής του SMA για τη προσέγγιση της περιθώριας συνάρτησης πιθανότητας μπορεί να αναπαράξει τις φυσικές διαδικασίες διατηρώντας τις τέσσερις κεντρικές ροπές, το οποίο έχει βρεθεί ότι είναι ικανοποιητικό για πολλές κατανομές που κοινώς εφαρμόζονται στις γεωφυσικές διαδικασίες.

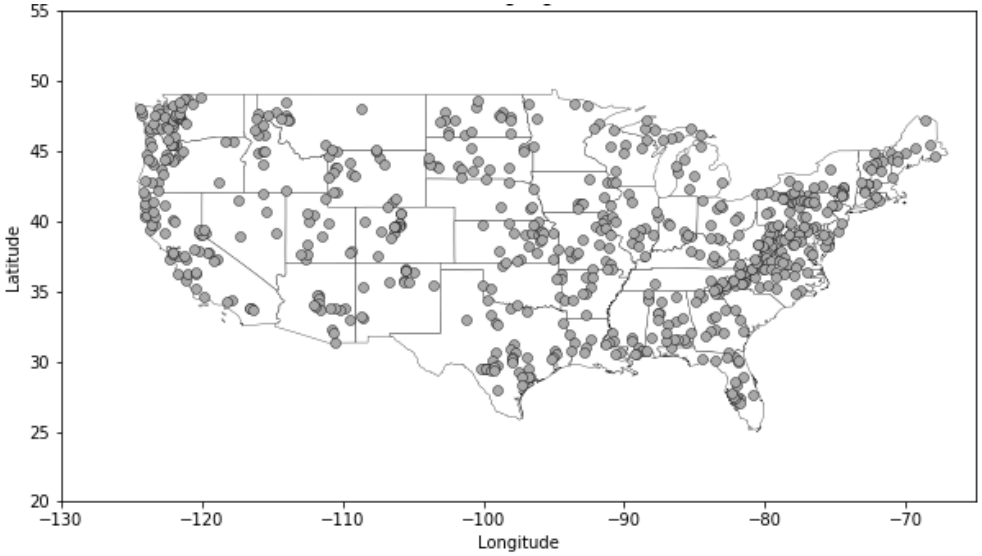
Η μέθοδος SMA μπορεί να περιγραφεί από την ακόλουθη εξίσωση:

$$\underline{x}_i = \sum_{j=-l}^{l} a_{|j|}\underline{v}_{i+j} \tag{0.5}$$

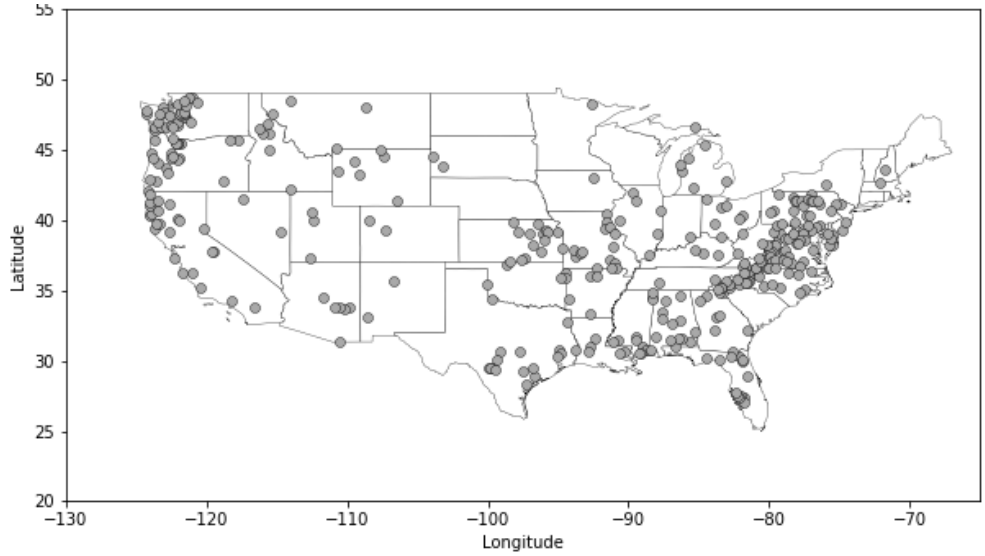
όπου $\underline{x}_i$ είναι οποιαδήποτε διαδικασία με κάποιας μορφής εξάρτηση, $a_{|j|}$ είναι συντελεστές που υπολογίζονται από τη συνάρτηση αυτοσυσχέτισης, $\underline{v}_{i+j}$ είναι ο λευκός θόρυβος σε διακριτό χρόνο και l θεωρητικά είναι ίσο με το άπειρο (αλλά ένας πεπερασμένος αριθμός μπορεί επίσης να χρησιμοποιηθεί διατηρώντας τη δομή συσχέτισης έως και καθυστέρηση μιας μονάδας).

Η μεθοδολογία εφαρμόστηκε στη βάση δεδομένων US-CAMELS, η οποία περιέχει 671 ημερήσιες χρονοσειρές παροχών (Newman et al., 2014). Από αυτές τις χρονοσειρές, επιλέχθηκαν προς περαιτέρω ανάλυση και επεξεργασία οι 360 που είχαν κάλυψη στο μέγιστο χρονικό μήκος (35 ετών, από το 1980 έως το 2014) και λιγότερα από 10% ελλείποντα στοιχεία.

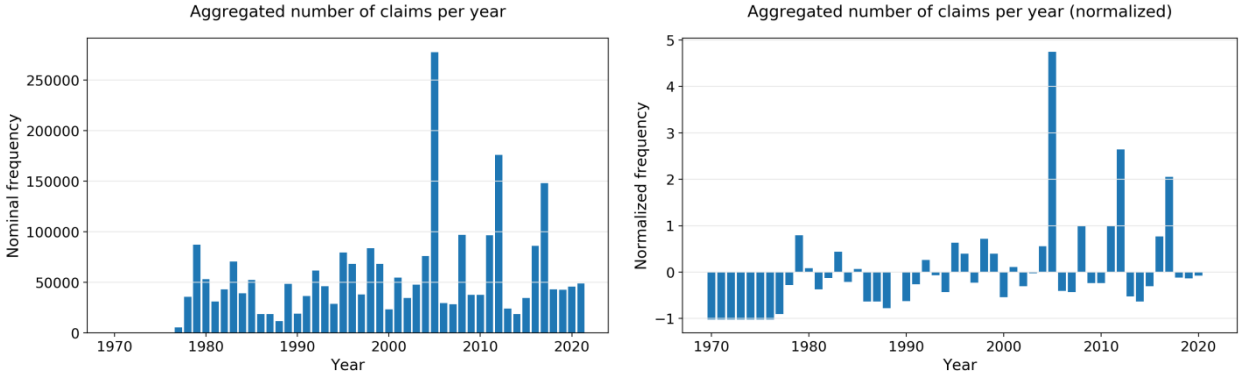
Ακόμα, στην εργασία αξιοποιήθηκαν τα πραγματικά ιστορικά στοιχεία αποζημιώσεων τα οποία δημοσίευσε πρόσφατα η Ομοσπονδιακή Υπηρεσία Διαχείρισης Έκτακτων Αναγκών των ΗΠΑ (FEMA, 2019) στο πλαίσιο του Εθνικού Προγράμματος Ασφάλισης Έναντι Πλημμυρών (NFIP) τα οποία περιλαμβάνουν περισσότερα από δύο εκατομμύρια αιτήματα προς αποζημίωση από το 1970 έως σήμερα, καθένα εκ των οποίων εμπεριέχει πληροφορίες που επιμερίζονται σε 39 διαφορετικές μεταβλητές. Τα στοιχεία αυτά παρουσιάζονται χωροχρονικά και γραφικά σε χάρτες με βάση τους οποίους προκύπτουν σημαντικά συμπεράσματα για τις περιοχές που κυρίως πλήττονται από πλημμύρες στις ΗΠΑ, καθώς και για την ανάγκη που καταδεικνύεται σε κάθε περιοχή για ασφαλιστική κάλυψη.



Σχήμα 2: Τα 671 σημεία μέτρησης της βάσης δεδομένων US-CAMELS.

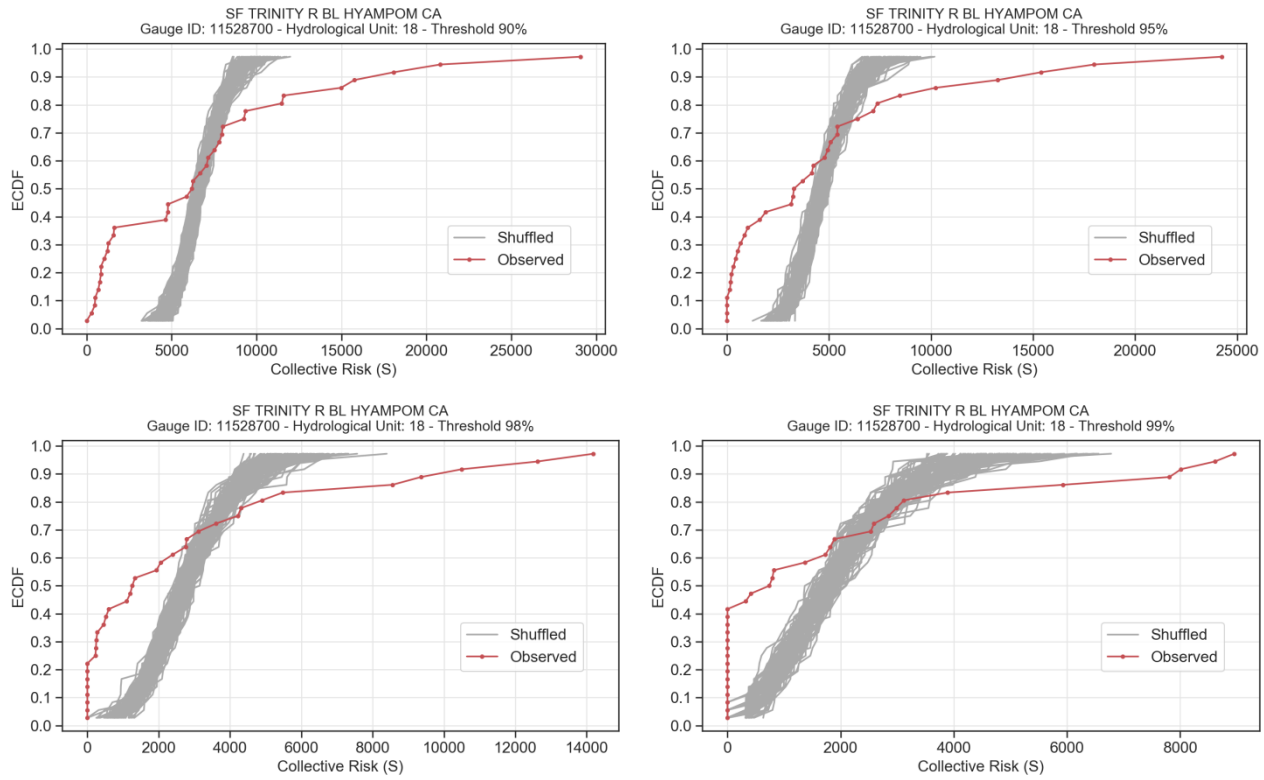


Σχήμα 3: Τα 360 τελικά σημεία μέτρησης της βάσης δεδομένων US-CAMELS που επιλέχθηκαν.



Σχήμα 4: Ο αθροιστικός αριθμός αιτημάτων προς αποζημίωση ανά έτος (1970-2021). Στο αριστερό διάγραμμα παρουσιάζονται οι ονομαστικές τιμές, ενώ στο δεξί οι κανονικοποιημένες, αφαιρώντας από την κάθε ετήσια τιμή τη μέση τιμή και διαιρώντας με την τυπική απόκλιση. Τελευταία ενημέρωση δεδομένων: 24-10-2021.

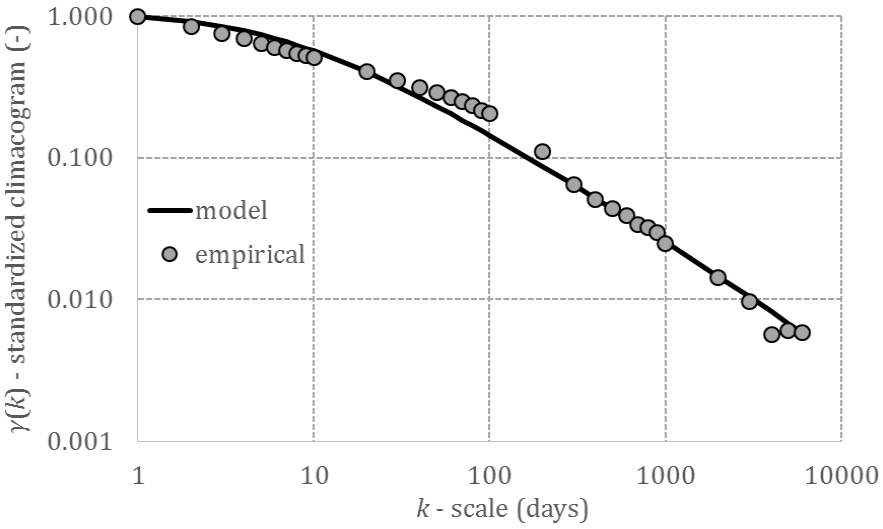
Για κάθε έναν από τους 360 σταθμούς μέτρησης που εξετάζονται και για κάθε μία από τις τέσσερις τιμές-κατώφλια (90%, 95%, 98%, 99%), υπολογίστηκε το ετήσιο συλλογικό ρίσκο *S*, η διάρκεια του θεωρούμενου πλημμυρικού γεγονότος καθώς και τα διαστήματα επαναφοράς τους, τόσο για τις ιστορικές (παρατηρούμενες) όσο και για τις 100 τυχαιοποιημένες (ανεξάρτητες) χρονοσειρές. Τα αποτελέσματα από αυτή τη διαδικασία είναι ιδιαιτέρως σημαντικά καθώς, σε πολλούς σταθμούς, η απόκλιση ανάμεσα στις πρώτες σε σχέση με τις δεύτερες στα διαγράμματα εμπειρικών αθροιστικών συναρτήσεων κατανομής (ECDF) ήταν μεγάλη. Τα παραπάνω αποτελέσματα είναι μια ξεκάθαρη ένδειξη μηχανισμών ομαδοποίησης ακραίων σε όρους συλλογικού ρίσκου *S*, διάρκειας γεγονότος αλλά και διαστημάτων επαναφοράς.



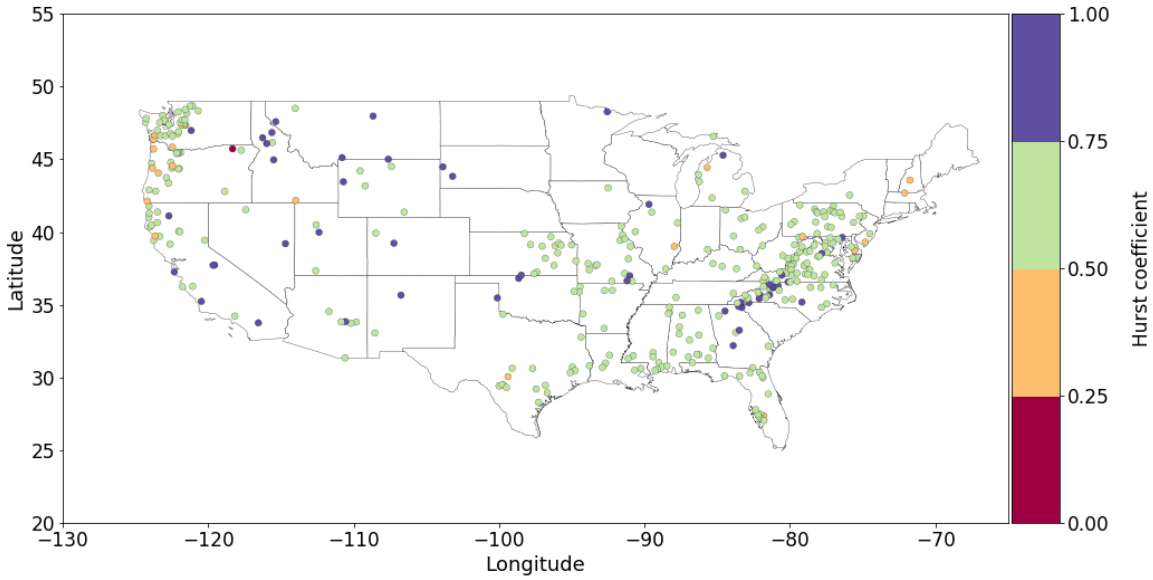
Σχήμα 5: Διάγραμμα εμπειρικών αθροιστικών συναρτήσεων κατανομής (ECDF) και συλλογικού ρίσκου *S*. Σταθμός μέτρησης: Sf Trinity River Below Hyampom, California (ID: 11528700).

Στη συνέχεια, υπολογίστηκε το μέσο *Κλιμακόγραμμα* από τη διαδικασία GHK (Dimitriadis and Koutsoyiannis, 2018) για όλες τις 360 ιστορικές χρονοσειρές παροχών της US-CAMELS, με την παράμετρο Hurst *H* να εκτιμάται ίση με 0.63, το οποίο είναι μια ένδειξη ισχυρής εμμονής. Η επίδραση της δομής συσχέτισης παρατηρείται στη συμπεριφορά των παροχών που ξεπερνούν το κατώφλι που τίθεται στην ετήσια κλίμακα όσο και στην εκτίμηση του συλλογικού ρίσκου. Συνεπώς, η συμπεριφορά των ημερήσιων παροχών της US-CAMELS βρέθηκε ότι είναι συνεπής με

τη δυναμική HK, η οποία χαρακτηρίζεται από τιμές H στο διάστημα 0.6-0.7, μέσω προσομοιώσεων Monte Carlo.



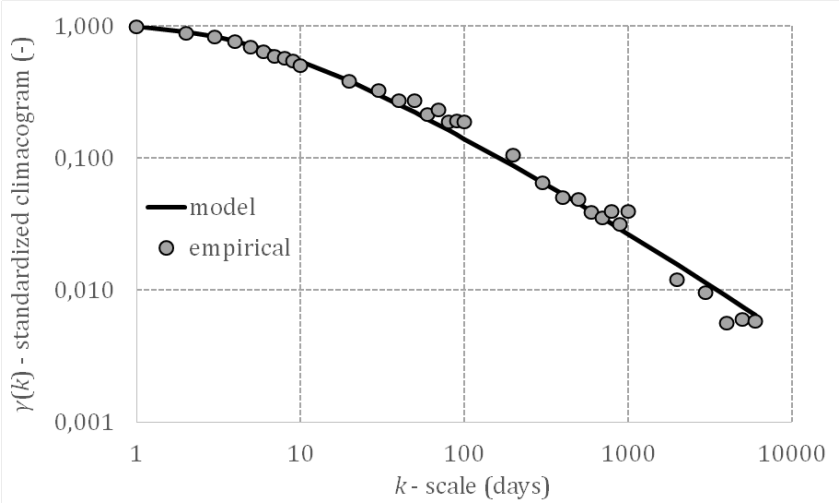
Σχήμα 6: Το μέσο *Κλιμακόγραμμα* των 360 επιλεγμένων χρονοσειρών της βάσης δεδομένων US-CAMELS.



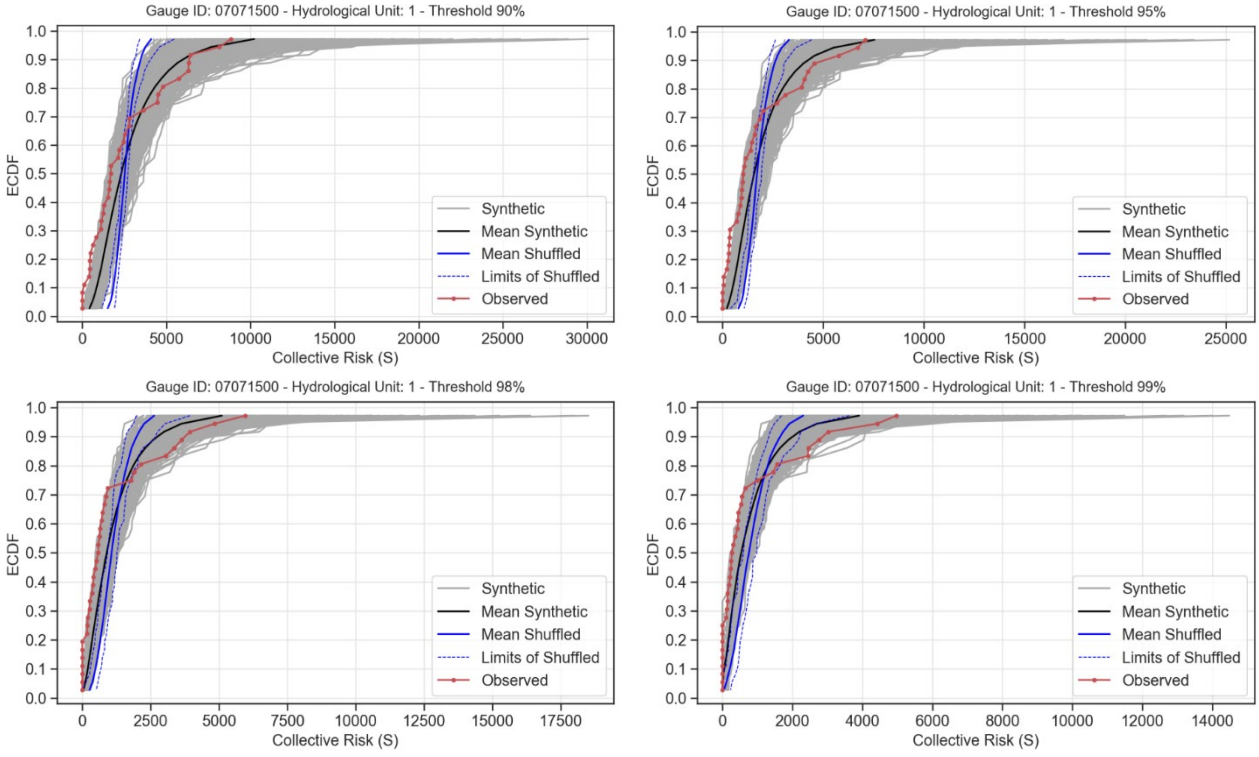
Σχήμα 7: Η παράμετρος Hurst H των 360 χρονοσειρών (σταθμών μέτρησης) της βάσης δεδομένων US-CAMELS.

Ακολούθως, για την αποτελεσματικότερη αξιοποίηση των ιστορικών χρονοσειρών έγιναν προσεγγίσεις μοντελοποίησης και προσομοίωσης με σκοπό τη βαθύτερη κατανόηση της σχέσης ανάμεσα στη στοχαστική δομή των πλημμυρικών γεγονότων και του συλλογικού ρίσκου S, μέσω της εφαρμογής του μοντέλου SMA-GHK (Dimitriadis and Koutsoyiannis, 2018). Ο αλγόριθμος για την παραγωγή των συνθετικών χρονοσειρών από τις ιστορικές απαιτεί ως είσοδο τη μέση τιμή (S_m), διακύμανση (S_v), συντελεστές ασυμμετρίας και κύρτωσης (S_s και S_k), παράμετρο Hurst του μοντέλου GHK (H), παράμετρο κλίμακας (q), μήκος της συνθετικής χρονοσειράς (N).

Μελετώντας τον σταθμό μέτρησης ID: 07071500, αναπτύχθηκαν 1000 συνθετικές χρονοσειρές μέσω προσομοιώσεων Monte Carlo από το μοντέλο GHK (H = 0.81, q = 1.00 d). Παρακάτω παρουσιάζεται το *Κλιμακόγραμμα*, καθώς και τα διαγράμματα εμπειρικής αθροιστικής συνάρτησης κατανομής (ECDF) του συλλογικού ρίσκου S για όλες τις τιμές-κατώφλια. Η καμπύλη του S εμφανίζεται εντός των ορίων των προσομοιώσεων Monte Carlo που παρήχθησαν από το μοντέλο GHK. Αντίθετα, οι καμπύλες των τυχαιοποιημένων χρονοσειρών παρουσιάζουν μια διαφορετική συμπεριφορά, ειδικά στις ουρές της κατανομής.



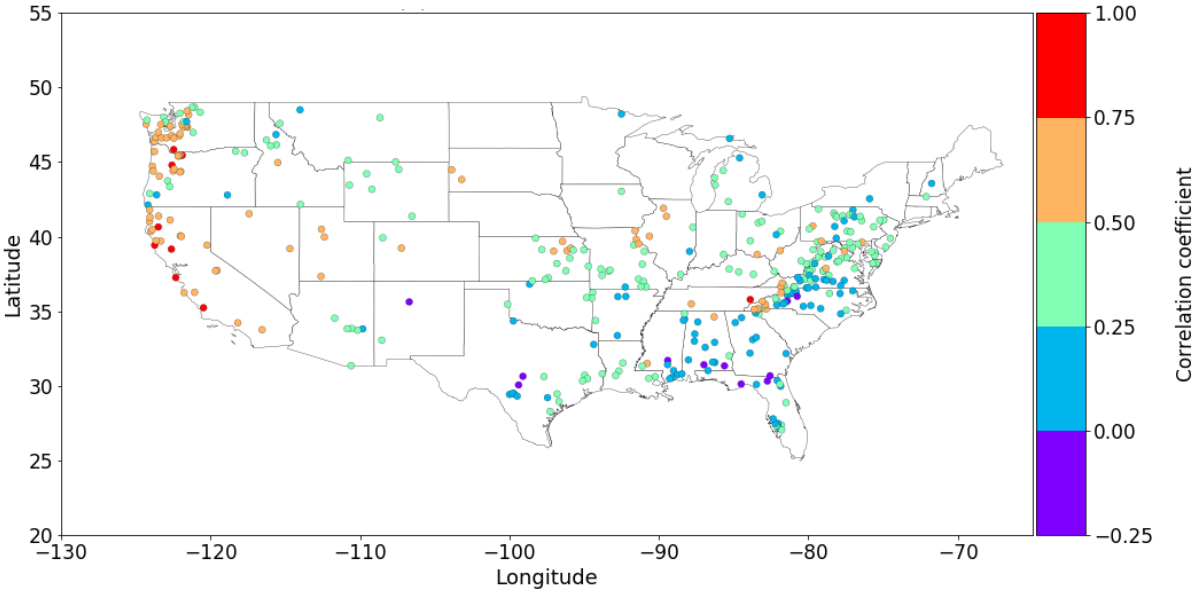
Σχήμα 8: *Κλιμακόγραμμα* του σταθμού μέτρησης Eleven Point river near Bardley, Missouri (ID: 07071500).



Σχήμα 9: Διάγραμμα ECDF - S των ιστορικών, τυχαιοποιημένων και συνθετικών χρονοσειρών (ID: 07071500).

Έχοντας υπολογίσει το ετήσιο συλλογικό ρίσκο *S* των επιλεγμένων 360 σταθμών μέτρησης της US-CAMELS για τα τέσσερα κατώφλια (90%, 95%, 98% και 99%) και έχοντας στη διάθεσή μας τα πραγματικά στοιχεία αποζημιώσεων της FEMA, μας δίνεται πλέον η δυνατότητα να διερευνήσουμε την εγκυρότητα της μεθοδολογίας μας σε χωροχρονική βάση, εξετάζοντας τον βαθμό συσχέτισης μεταξύ αυτών των μεγεθών. Αυτό το επιτυγχάνουμε με τον επιμερισμό των στοιχείων αποζημιώσεων σε κάθε μια από τις 21 υδρολογικές περιφέρειες των ΗΠΑ και ανά Πολιτεία. Σε αυτό το πλαίσιο, εκτιμάται ο συντελεστής συσχέτισης Spearman, ο οποίος ενδείκνυται στην ανάλυση ακραίων τιμών, συσχετίζοντας το ετήσιο συλλογικό ρίσκο *S* κάθε σταθμού μέτρησης με τα αθροιστικά αιτήματα προς αποζημίωση λόγω πλημμύρας που πραγματοποιήθηκαν στην υδρολογική περιφέρεια ή Πολιτεία που ανήκει ο σταθμός, για το μέγιστο διατιθέμενο χρονικό διάστημα (1980-2014).

Απεικονίζοντας στον χάρτη τους παραπάνω συντελεστές συσχέτισης κάθε σταθμού μέτρησης μπορούμε να σκιαγραφήσουμε τη χωρική κατανομή των περιοχών που ο συντελεστής συσχέτισης Spearman είναι μεγαλύτερος και αυτές που είναι σημαντικά μικρότερος. Είναι εμφανές ότι δημιουργείται ένα χωρικό μοτίβο με υψηλούς συντελεστής στις Δυτικές Ακτές και σημαντικά μικρότερους στις Ανατολικές.

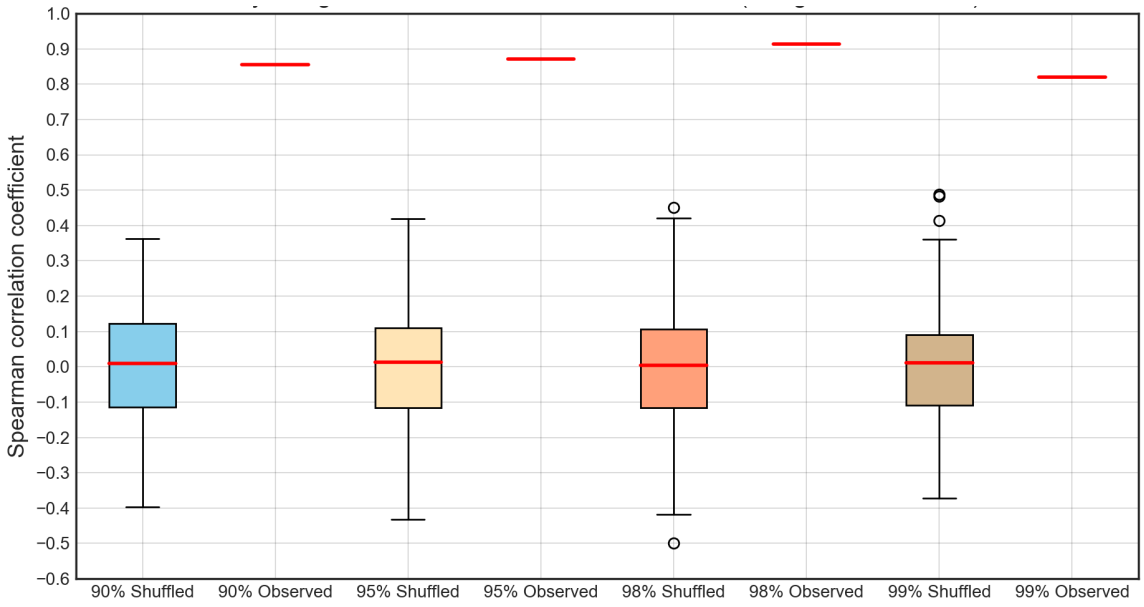


Σχήμα 10: Συντελεστής συσχέτισης Spearman για κάθε έναν από τους 360 επιλεγμένους συντελεστές συσχέτισης μεταξύ του ετήσιου συλλογικού ρίσκου και των συνολικών πραγματικών αιτημάτων προς αποζημίωση για την υδρολογική περιφέρεια στην οποία ανήκει ο κάθε σταθμός (κατώφλι 99%).

Μιας και το συλλογικό ρίσκο στην εργασία μας αναφέρεται κυρίως σε ποτάμια πλημμύρα, τα παραπάνω αποτελέσματα μας οδηγούν στο συμπέρασμα ότι αυτό το είδος πλημμύρας είναι

επικρατέστερο στις Δυτικές Ακτές των ΗΠΑ. Αντίθετα, αποκαλύπτεται ότι οι Ανατολικές Ακτές χαρακτηρίζονται από ένα πιο περίπλοκο χωρικό μοτίβο, η δυσκολία στην ερμηνεία του οποίου οφείλεται κυρίως στην ευαλωτότητα και στην υψηλή ευαισθησία αυτών των περιοχών σε πλημμυρικά γεγονότα που οφείλονται σε τυφώνες, στην επίδραση της ταλάντωσης του βορείου Ατλαντικού (North Atlantic Oscillation, NAO), στην άνοδο της στάθμης της θάλασσας (SLR) και σε φαινόμενα κύματος καταιγίδας (storm surge phenomena) τα οποία εμφανίζονται πολύ συχνότερα εκεί σε σχέση με τη Δυτική Ακτή (Ezer and Atkinson, 2014; Elsner et al., 2000).

Επιπροσθέτως, πέρα από τις ιστορικές χρονοσειρές της US-CAMELS, η ίδια διαδικασία υπολογισμού του συντελεστή συσχέτισης Spearman, ανάμεσα στο συλλογικό ρίσκο *S* και τα αθροιστικά αιτήματα προς αποζημίωση λόγω πλημμύρας που πραγματοποιήθηκαν στην υδρολογική περιφέρεια που ανήκει ο σταθμός, ακολουθήθηκε και για κάθε μία από τις 100 τυχαιοποιημένες χρονοσειρές που δημιουργήσαμε για κάθε έναν από τους 360 σταθμούς μέτρησης, με σκοπό να διερευνήσουμε για ακόμα μια φορά τους μηχανισμούς ομαδοποίησης που δημιουργούνται, απεικονίζοντάς τα σε ένα θηκόγραμμα (box plot) για τα τέσσερα κατώφλια.

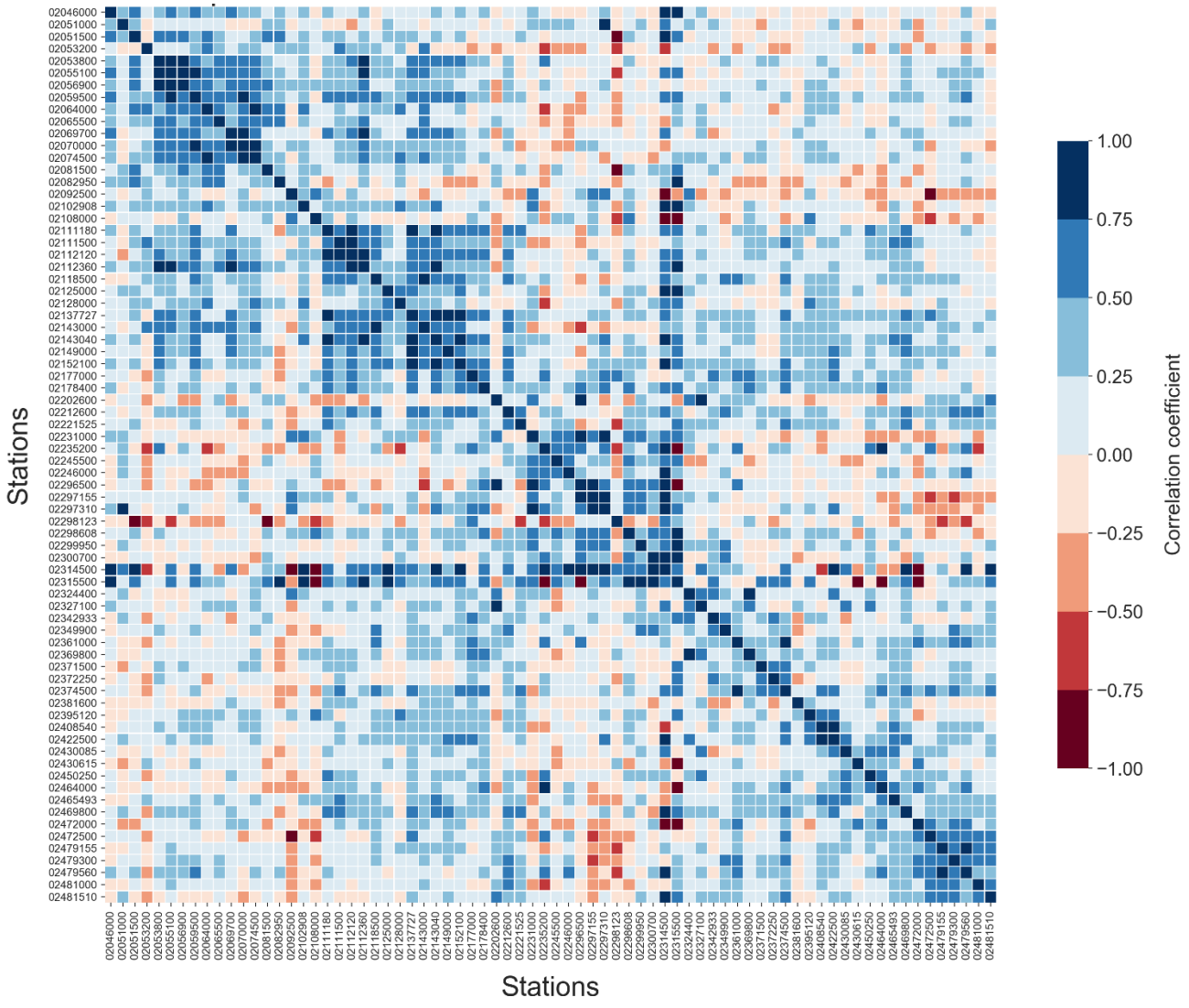


Σχήμα 11: Θηκόγραμμα (box plot) των συντελεστών συσχέτισης Spearman του σταθμού μέτρησης Lopez C NR Arroyo Grande, California (ID: 11141280) ανάμεσα στο ετήσιο *S* και τα συνολικά αιτήματα προς αποζημίωση της υδρολογικής περιφέρειας στην οποία ανήκει ο σταθμός (1980-2014) για τα τέσσερα επιλεγμένα κατώφλια μελέτης.

Τα αποτελέσματα αυτής της διερεύνησης είναι εντυπωσιακά καθώς σε πολλούς σταθμούς μέτρησης παρατηρούμε ότι η απόκλιση των συντελεστών συσχέτισης ανάμεσα στις ιστορικές και τις τυχαιοποιημένες (ανεξάρτητες) χρονοσειρές είναι σημαντική, μιας και ουσιαστικά ο συντελεστής συσχέτισης ανάμεσα στο ετήσιο συλλογικό ρίσκο *S* των τυχαιοποιημένων και στα ετήσια αιτήματα

προς αποζημίωση εμφανίζεται να είναι αμελητέος. Η υποεκτίμηση του παραπάνω συντελεστή συσχέτισης, των μηχανισμών ομαδοποίησης και του ρίσκου στο σύνολό του, βάσει αυτών των αποτελεσμάτων, φαίνεται ότι μπορεί να αποφέρει αξιοσημείωτες οικονομικές επιπτώσεις σε περίπτωση ενός ακραίου πλημμυρικού γεγονότος, αν δεν λαμβάνονται υπόψη αυτές οι παράμετροι στις στρατηγικές μοντελοποίησης.

Τέλος, με δεδομένο ότι συχνά οι ακραίες τιμές παροχών τείνουν να χαρακτηρίζονται από χωρική συσχέτιση (Quinn et al., 2019), διερευνήθηκαν πιθανοί χωρικοί μηχανισμοί ομαδοποίησης, μέσω του συντελεστή συσχέτισης Spearman, ανάμεσα στο ετήσιο συλλογικό ρίσκο *S* των σταθμών μέτρησης της US-CAMELS που ανήκουν στην ίδια υδρολογική περιφέρεια, για όλα τα επιλεγμένα κατώφλια μελέτης. Παρακάτω παρουσιάζεται ο βαθμός χωρικής συσχέτισης (συντελεστής Spearman) των σταθμών της υδρολογικής περιφέρειας 3 μέσω θερμοχάρτη (heatmap).



Σχήμα 12: Θερμοχάρτης (heatmap) του συντελεστή συσχέτισης Spearman ανάμεσα στο συλλογικό ρίσκο *S* όλων των σταθμών μέτρησης που ανήκουν στην υδρολογική περιφέρεια 3 μεταξύ τους (κατώφλι 99%).

Η ανάλυση έδειξε ότι επιλέγοντας χαμηλότερες τιμές-κατώφλια, ο συντελεστής συσχέτισης λαμβάνει σχεδόν ομοιόμορφα υψηλές τιμές σε όλη την έκταση της υδρολογικής περιφέρειας. Αντίθετα, αυξάνοντας τις τιμές των κατωφλίων που μελετώνται, δημιουργείται μια μεγαλύτερη ανομοιομορφία των συντελεστών συσχέτισης, ειδικά όταν αυτό φτάσει το 99%.

Αναλυτικότερα, όταν ο συντελεστής συσχέτισης του *S* ανάμεσα σε δύο σταθμούς μέτρησης βρίσκεται στο διάστημα:

- -0.25 έως 0.25, σημαίνει ότι η συσχέτιση είναι πρακτικά μηδενική, δηλαδή ότι δυο πιθανά πλημμυρικά γεγονότα σε κάθε έναν από τους εξεταζόμενους σταθμούς μέτρησης μπορούν να θεωρηθούν ανεξάρτητα.
- 0.25 and 1, σημαίνει ότι όταν συμβεί ένα πλημμυρικό γεγονός σε έναν από τους δύο εξεταζόμενους σταθμούς μέτρησης, είναι πολύ πιθανό να συμβεί ανάλογο γεγονός και στον άλλο σταθμό μέτρησης.
- -0.25 and -1, σημαίνει ότι όταν συμβεί ένα πλημμυρικό γεγονός σε έναν από τους δύο εξεταζόμενους σταθμούς μέτρησης, δεν είναι πολύ πιθανό να συμβεί ανάλογο γεγονός και στον άλλο σταθμό μέτρησης.

Οι ασφαλιστικές εταιρείες επιδιώκουν να δημιουργούν χαρτοφυλάκια τα στοιχεία των οποίων έχουν αρνητική χωροχρονική συσχέτιση, ή τουλάχιστον μηδενική, ώστε να συνδυάζονται και συναθροίζονται ρίσκα τα οποία δεν είναι πιθανό να αντιπροσωπεύουν πιθανά πλημμυρικά γεγονότα τα οποία θα συμβούν στον ίδιο χώρο ή χρόνο. Για τον λόγο αυτό ο παραπάνω θερμοχάρτης είναι σημαντικός, καθώς επιτρέπει στις εταιρείες να διαμορφώνουν ένα αθροιστικά μειωμένου ρίσκου χαρτοφυλάκιο σε όρους συλλογικού ρίσκου *S*.

Table of Contexts

List of Figures

List of Tables

1. Introduction

1.1. Research scope

During the last decades, the rising demand for crops for human consumption and industrial processes has led to a growth of investments and search for innovative solutions across the field of agriculture. However, one major risk that both investors and low-income farmers encounter worldwide is the impact of extreme weather events on their crop yield. The risk caused by extreme events is an inhibitor of growth of agriculture and, apparently, insurance is strategically important for dealing with that risk. In particular, crop-yield insurance is purchased by agricultural producers, and in many cases is subsidized by governments, to protect them against the loss of their crops due to natural disasters, such as extreme flood events.

In a wider point of view, population growth, economic development and risk-blind urbanization often increase exposure to risk, including that due to floods. While rural flooding may affect much larger areas of land, urban floods are more challenging to manage, since the higher population and asset density in the urban environment increase the environmental and social impacts of floods and make the potential flood damages more costly. Therefore, the need for integrated flood insurance policy and accurate flood risk assessment is pronounced in order to reduce the financial consequences of extreme flood events, which endanger in many cases the environmental, social and economic balance.

Regarding the central role of flood insurance for societies and individuals as a tool for hedging against the risk of financial loss due to natural hazards, this study investigates some key aspects that should be considered in modeling strategies in order to improve current risk assessment processes. In this regard, the modeling approach of the peak-over-threshold method, including the evaluation of the so-called collective risk S is a common procedure that is followed in insurance and reinsurance sector. In financial terms, collective risk S is defined as the accumulation of claim amounts over fixed one-year time windows. Yet in hydrology, it was recently suggested (Serinaldi and Kilsby, 2016) that streamflow exceedances over given thresholds may be considered as proxies for claim amounts from flood losses.

In this context, the aim of this research is to apply a stochastic approach for hydrological extremes informed by current insurance and reinsurance practices. To this end, it seeks to (a) provide insights into spatiotemporal clustering mechanisms of streamflow and rainfall extremes and (b) investigate

their impacts on flood insurance practices. In addition to the development of modeling approaches, the properties of collective risk, return intervals, duration and the number as well as the severity of the over-threshold events are evaluated in order to utilize effectively all the historical data available for a wider understanding of their footprint on flood claim amounts. Furthermore, a spatiotemporal exploratory analysis of streamflow in USA and rainfall extremes in Greece is introduced regarding their stochastic aspects and their correlation with actual insurance data derived from the recently published database of the National Flood Insurance Program (NFIP) by Federal Emergency Management Agency (FEMA, 2019) and the Hellenic Agricultural Insurance Organization (ELGA), respectively. The latter organization provided us directly the aforementioned data.

Preliminary outcomes of this research have been presented in the General Assembly of the European Geosciences Union (Papoulakos et al., 2020; Manolis et al., 2020; Goulianou et al., 2019).

1.2. Work structure

The thesis is structured into nine distinct chapters, all of whom are sorted in a way similar to the line of reasoning required for the understanding of the objective.

In the **first chapter** we present a preamble to the subject, the general context in which this study is introduced and its structure.

In the **second chapter** we overview the impacts of natural hazards and flood events on individuals and societies. Moreover, we define flood risk and describe the components of an integrated flood risk management system.

In the **third chapter** we introduce the principles of insurance and describe the partnership between societies, public and private sector for flood risk share and reduction.

In the **fourth chapter** we make an extensive presentation of the methodology and the stochastic tools used, as well as the rudimentary stochastic theory behind it, namely Extreme Value Theory and the Hurst-Kolmogorov dynamics.

In the **fifth chapter** we present the databases used, including the US-CAMELS dataset and the FEMA NFIP claims records. In addition, we explain the qualitative and quantitative criteria with which data were selected for processing, as well the preliminary data processing thereof.

In the **sixth chapter** we evaluate the spatiotemporal clustering mechanisms of streamflow extremes on flood insurance practices, including modelling approaches and Monte Carlo simulations. Moreover, we assess the strength of association between the studied proxy of collective risk and the FEMA's NFIP actual claims records. In addition, we perform a brief analysis of spatial dependence mechanisms of US-CAMELS dataset.

In the **seventh chapter** we investigate the interplay between precipitation clustering mechanisms, rainfall extremes and actual historic flood compensations considering a Greek case study in the Larissa region.

In the **eighth chapter** we present the conclusions of this study and we propose some suggestions for future research.

2. Theoretical analysis on the impacts of natural hazards and flood events

2.1. An overview of the impacts of climate-related and geophysical disasters

The Centre for Research on the Epidemiology of Disasters (CRED) defines a disaster as *"a situation or event which overwhelms local capacity, necessitating a request at national or international level for external assistance; an unforeseen and often sudden event that causes great damage, destruction and human suffering"*. The scientific research on the economic impacts of disasters caused by extreme weather events is a versatile issue that draws continuously the attention of governments, policy-makers and societies. These disasters could be separated into two main categories; the climate-related and the geophysical ones. Figures 2.1 and 2.2 present the most financially affected countries by disasters for the period 1998-2017 in terms of absolute losses and average annual percentage losses relative to GDP.

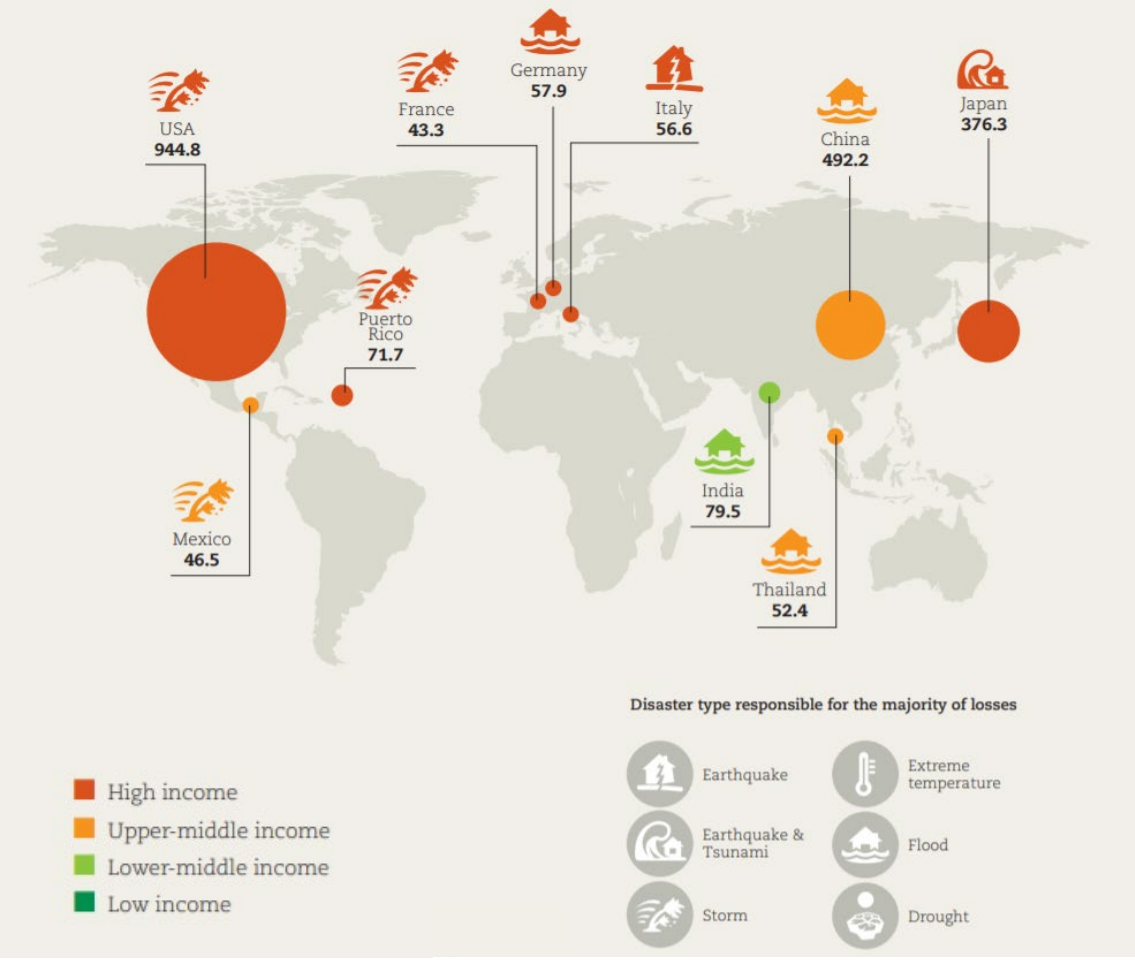


Figure 2.1 Top 10 countries/territories in terms of absolute losses (billion US$), 1998-2017 (CRED and UNISDR, 2018, modified).

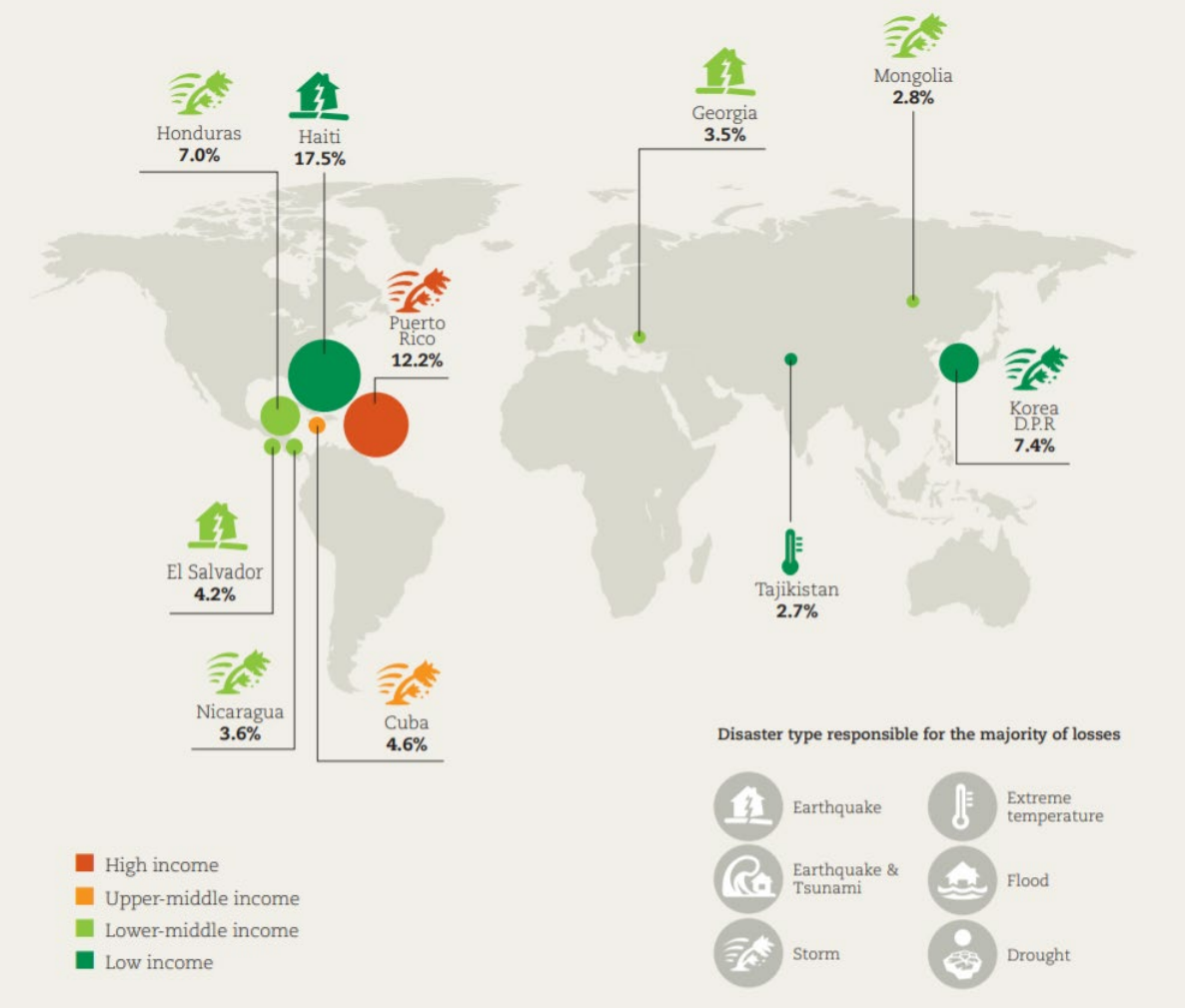


Figure 2.2 Top 10 countries/territories in terms of average annual percentage losses relative to GDP, 1998-2017 (CRED and UNISDR, modified).

According to the World Bank, the annual cost of disasters has been estimated to US$ 520 billion, reducing rapidly at the same time the standard of living of some 26 million people, pushing them into poverty every year (World Bank, 2017). During the period 1998-2017, the death toll from both types of disasters (climate-related and geophysical) worldwide is estimated to be approximately 1.3 million people and the reported cumulative financial losses of these disasters amount to US$ 2.908 billion (GDP and all economic data are adjusted at 2017 US$ value). In addition, tens of millions affected people found themselves in a state of emergency, requiring assistance in local, national or international level. Although these cumulative losses may seem large, they do not describe the full picture as financial reviews that are collected by the Emergency Events Database (EM-DAT) reveal that, unfortunately, the vast majority of reports (63%) contain no economic data (CRED AND UNISDR, 2018).

Figure 2.3 presents the proposed classification of the most common disasters (Integrated Research on Disaster Risk, 2014), excluding Biological and Extraterrestrial as their investigation is not part of the scope of this study. This introductory chapter focuses on events which are related to hydrological, meteorological and climatological parameters (which collectively are termed weather or climate-related) plus geophysical disasters.

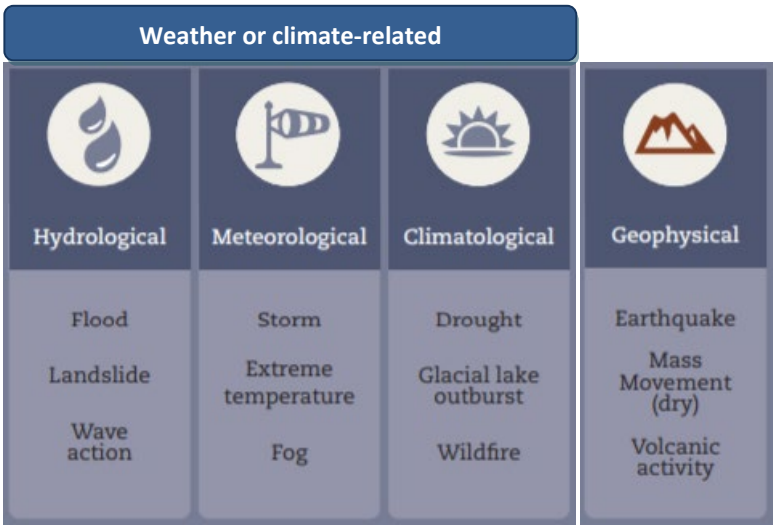


Figure 2.3 Classification of Natural Hazards (Integrated Research on Disaster Risk, 2014).

The crucial factor that determines whether an extreme weather event or a natural hazard can be described as a humanitarian or financial disaster is the vulnerability of the affected regions. For example, impacts of extreme flood events are sharpened in cases of increased exposure to risk which arise by parameters such as nonsensical urban development in risk-prone areas, inadequate construction of flood protection works, incautious land use changes and weak governance. Thus, a river flooding near a low density populated city can be deadly with incalculable financial impacts; however, a major storm surge in an inhabitable coastal area will not be a disaster if no people are affected or harmed.

Except anthropogenic interventions, it is under discussion and investigation whether climate change threatens the ability of policy-makers to manage risk. Nevertheless, there are scientific works which present evidence that, regarding extreme flood events, there is no detectable sign of human-induced climate change in the normalized flood losses. The steady increase in the original flood losses which is regularly observed in literature is mostly driven by societal factors (Barredo, 2008; Crompton and McAneney, 2019).

Regarding the occurrences of each type of disasters, statistics show (Figure 2.4) that the climate-related ones were the most frequent during the period 1998-2017. Within this subgroup, floods appear to be the most numerous in terms of aggregated disasters and approximate number of people affected (Figures 2.5 and 2.6). Furthermore, storms appear to be the costliest type of disaster, followed by earthquakes and floods (Figure 2.7).

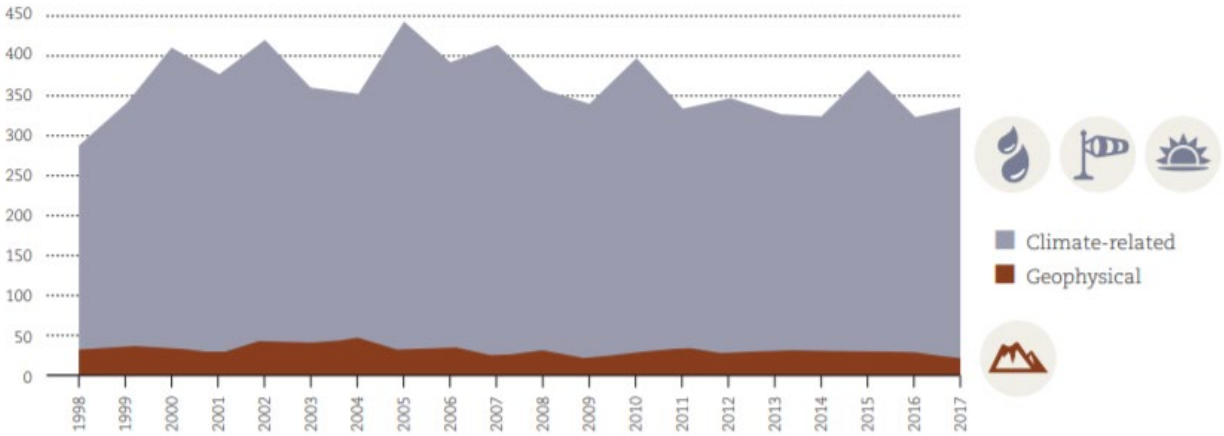


Figure 2.4 Number of disasters by major category per year 1998-2017 (CRED and UNISDR, 2018, modified).

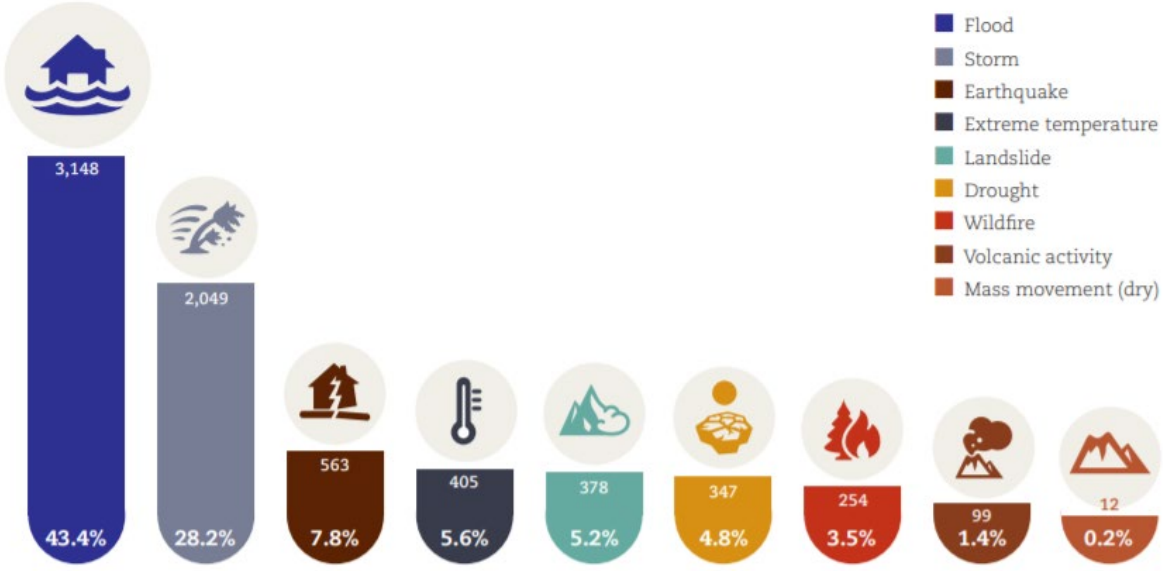


Figure 2.5 Numbers of disasters per type 1998-2017 (CRED and UNISDR, 2018, modified).

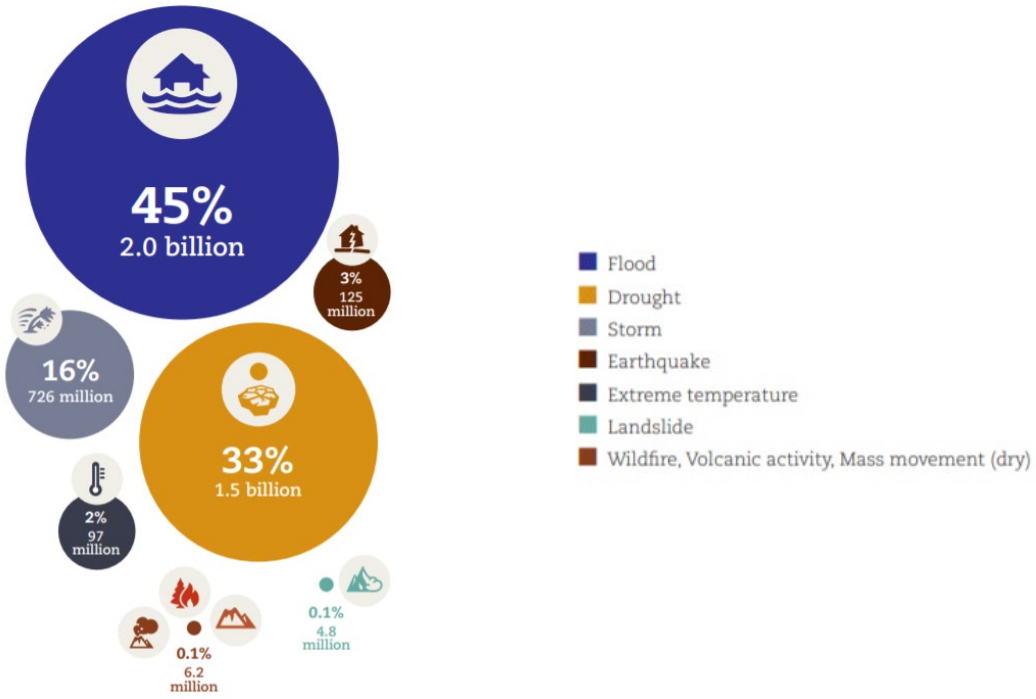


Figure 2.6 Number of people affected per disaster type 1998-2017 (CRED and UNISDR, 2018, modified).

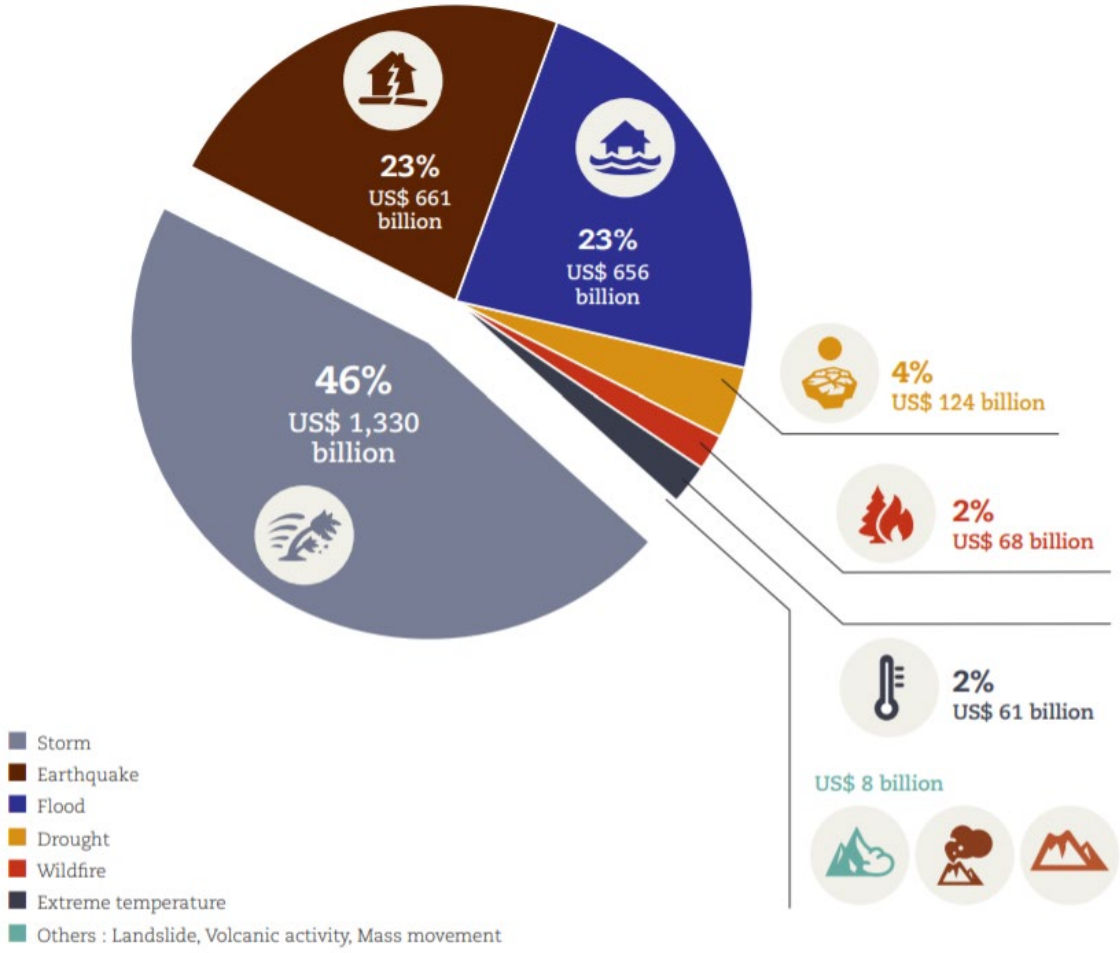


Figure 2.7 Breakdown of recorded economic losses (US$) per disaster type 1998-2017 (CRED and UNISDR, 2018, modified).

The quantification and the annual aggregation of the economic impacts of major disasters over the period 1998-2017 (Figure 2.8) unveil the significance of the structural and non-structural methods that should be developed in order to moderate the footprint of these events. Moreover, Figure 2.9 highlights the extensive contribution of storm events on the climate-related disasters mix.

Figure 2.8 Total reported economic losses per year, with major events highlighted, 1998-2017 (CRED and UNISDR, 2018, modified).

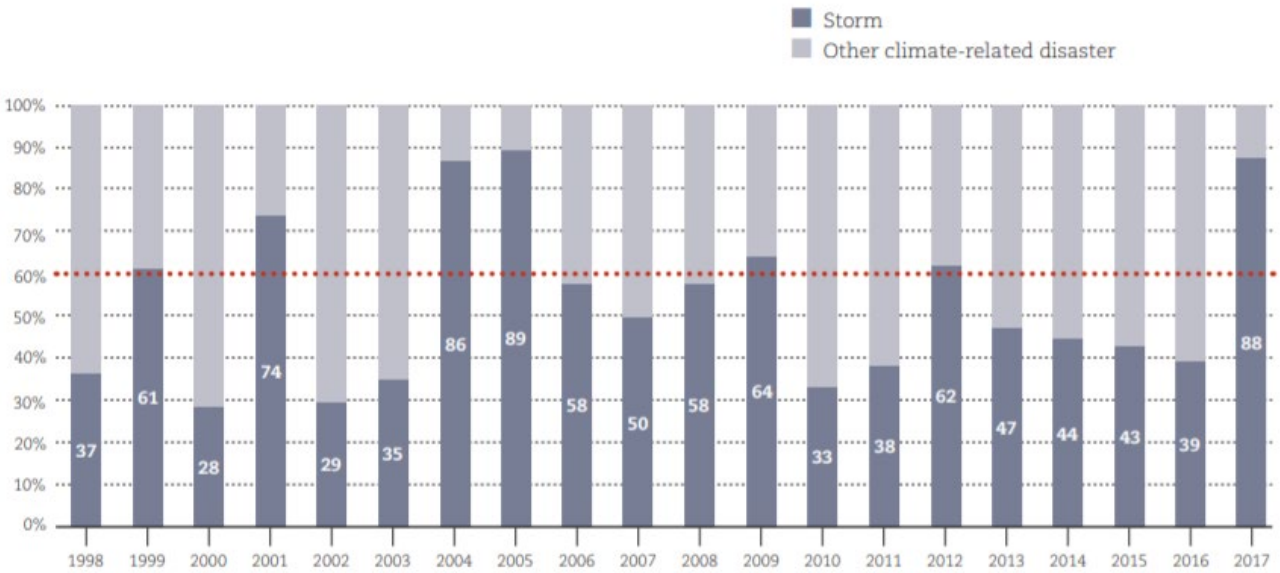


Figure 2.9 Total share of losses due to storms as a percentage of annual climate-related disaster losses, 1998-2017 (CRED and UNISDR, 2018, modified).

As mentioned previously, one parameter that we should take into consideration is that, consistently, a high percentage of disasters worldwide were inaccurately or ineffectively reported. This unfortunate factuality characterizes the data collected from both wealthier as well as poorer countries.

Tables 2.1, 2.2 and 2.3 present summaries of statistics regarding the report of economic losses (%) per income group, per continent and per disaster type. These statistics clearly highlight a gradual reduction of the percentage of the reported losses from the higher to the lower income groups. Searching for a continental pattern, Africa seems to be the continent with an extensive low report rate regarding disasters. In addition, storm is not only the costliest type of disaster, but also the one with the higher report rate. Eventually, efforts have been made in the direction of the improvement of disasters data collection process and the increase of their report rate by the United Nations and other organizations worldwide, such as the adoption of the Sendai Framework for Disaster Risk Reduction 2015-2030 (UNDRR, 2015).

	ALL	Climate-related	Geophysical
High income	53	52	61
Upper-middle income	40	40	37
Lower-middle income	31	30	31
Low income	13	13	20

Table 2.1 Reporting of economic losses (%) per income group (CRED and UNISDR, 2018, modified).

	ALL	Climate-related	Geophysical
Oceania	48	51	23
Americas	42	43	32
Asia	42	42	40
Europe	38	37	54
Africa	14	14	24

Table 2.2 Reporting of economic losses (%) per continent (CRED and UNISDR, 2018, modified).

	% reported		% reported
Storm	55	Earthquake	43
Wildfire	41	Volcanic activity	11
Flood	32	Mass movement (dry)	8
Drought	29		
Landslide	13		
Extreme temperature	11		

Table 2.3 Reporting of economic losses per disaster type for climate-related (left) and geophysical (right) disasters (CRED and UNISDR, 2018, modified).

Deepening on the impacts of extreme events per income group or continent, statistics and scientific studies (SOPAC, 2009; Kawasaki et al., 2020) have spotlighted the close relationship between floods, disasters and poverty. It has been highlighted that people who experience poverty tend to live in flood-prone areas where flood policy preparation mechanisms are inadequate, governance is weak and flood protection systems are not well maintained or not constructed at all. Furthermore, in such cases, decisions on precautionary measures prior to the outbreak of an extreme flood event ground on deterministic methods, as calculations on the poverty exposure bias does not change significantly under future climate scenarios, although the absolute number of people potentially exposed to floods can increase or decrease significantly, depending on the scenario and the region (Winsemius et al., 2015). The same logic is being followed on extreme drought phenomena. Studies have also indicated the impact of natural disasters and especially that of extreme floods and drought events on human development index and local poverty (Rodriquez-Oreggia et al., 2013). Accurate risk assessment and strong governance is the solution to such phenomena.

Although the report rate of economic losses is not ideal, some interesting conclusions could be extracted. Figure 2.10 present the recorded climate-related disaster losses per income group compared to GDP losses. Even though the absolute value (in US$ billions) is higher in high income countries, which makes sense as values in general are extremely higher in those countries, the parameter we should consider is the GDP loss (%), which is well above the International Monetary Fund's threshold for a major economic disaster of 0.5% (International Monetary Fund, 2020) in low and lower-middle income countries.

Figure 2.11 presents the classification of disasters based on income data. Although the disaster occurrences are almost equally divided between the income groups, deaths are higher in low and lower-middle income countries. Moreover, economic losses in high and upper-middle income group are extremely higher mainly due to the higher values that are dominant in those countries.

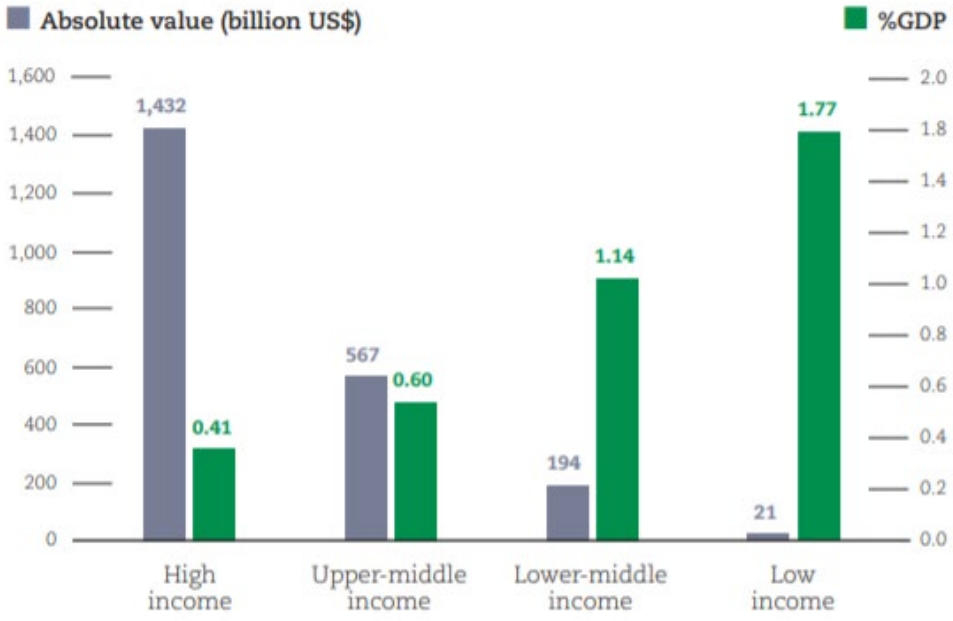


Figure 2.10 Recorded climate-related disaster losses per income group compared to GDP losses 1998-2017 (CRED and UNISDR, 2018, modified).

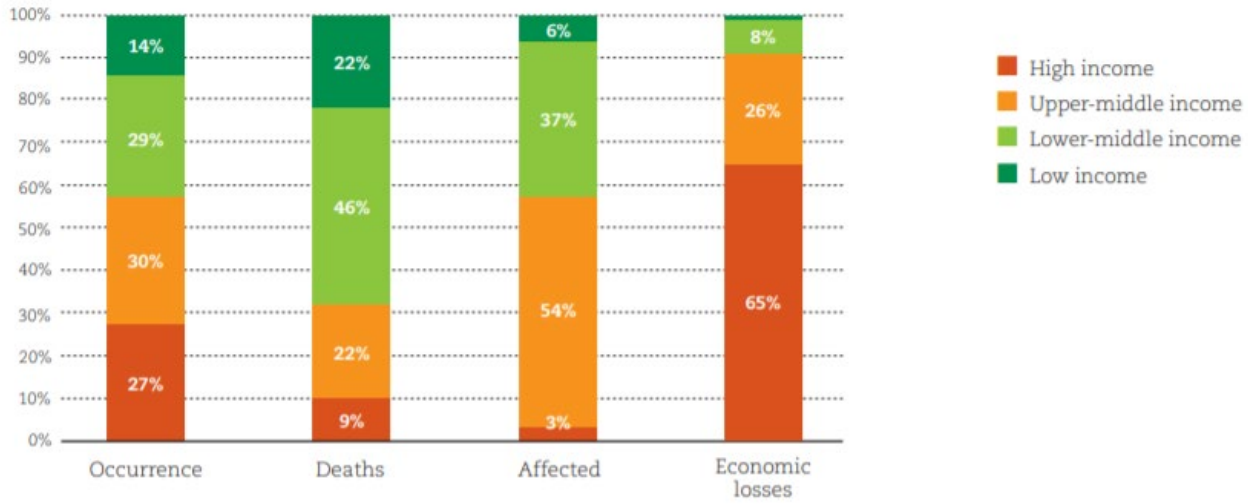


Figure 2.11 Classification of disasters based on income data 1998-2017 (CRED and UNISDR, 2018, modified).

Regarding the human cost of disasters, the EM-DAT measures it by two main parameters: the number of people killed, missing or presumed dead and the number of people affected by the

events, which refers to people requiring immediate assistance to provide basic survival needs (food, water, shelter, sanitation, medical assistance) during a period of emergency and includes people injured, homeless, displaced or evacuated during the emergency phase of a disaster. Figures 2.12 and 2.13 show that climate-related disasters impact mostly people who live in low and lower-middle income countries, as the percentage of deaths and affected people of per million population potentially exposed is higher.

Disasters can be considered as major contributors to entrenched poverty in low and middle income countries and, this is why, actions must be taken to empower the financially weak groups and countries. Reducing disaster risk is a cross-cutting issue for the reduction of poverty worldwide.

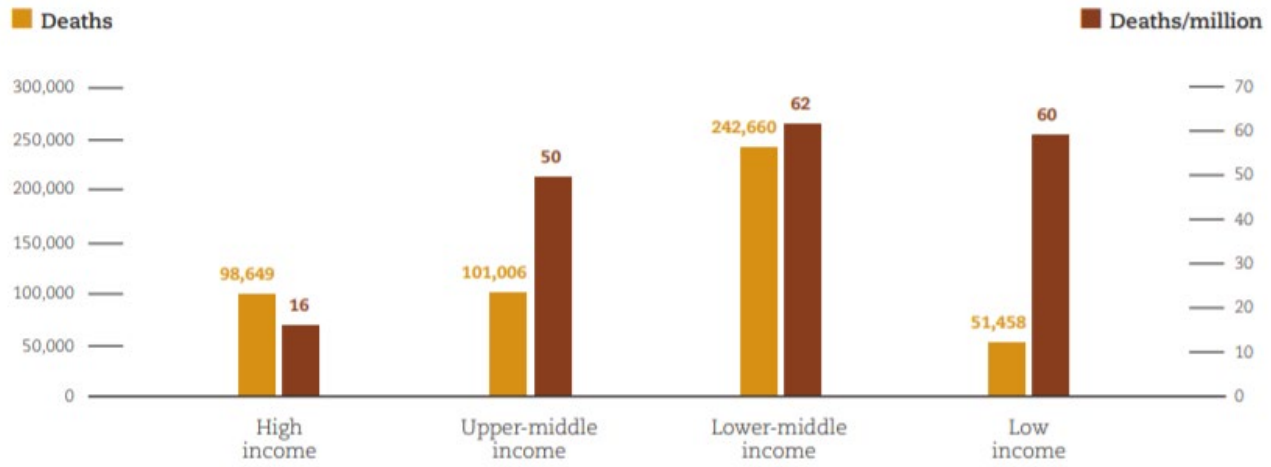


Figure 2.12 Climate-related disaster deaths in absolute numbers and percentage of per million population potentially exposed (PPE) 2000-2017 (CRED and UNISDR, 2018, modified).

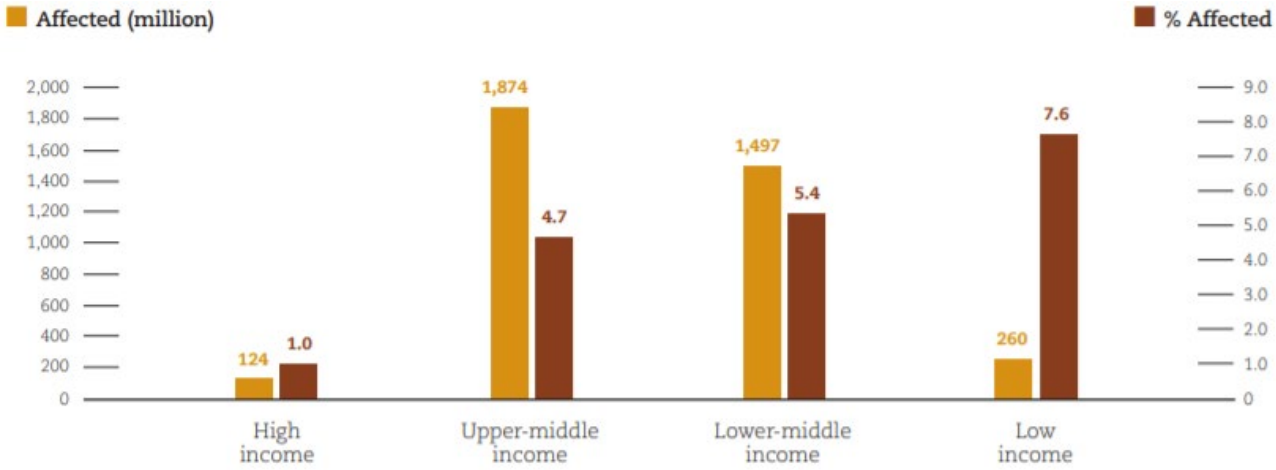


Figure 2.13 Climate-related disaster affected totals in absolute numbers and percentage of per million population potentially exposed (PPE) 2000-2017 (CRED and UNISDR, 2018, modified).

2.2. Components of flood vulnerability and types of flood

Extreme flood events impact on both individuals and communities, endangering in many cases the economic prosperity and social balance. These impacts depend on the topography of the flood area, the dominant hydrodynamic mechanisms and the vulnerability of the affected environments.

Societal and anthropogenic factors are frequently parts of the primary force that increase vulnerability in such environments, as economic development, risk-blind urbanization and population growth, place more people and assets in high-risk areas, such as flood plains, coastlines and dry lands, endangering their viability.

Three basic areas of flood vulnerability can be distinguished according to the principle of sustainability: social and cultural, economic and ecological vulnerability (Schanze et al., 2004). Social and cultural vulnerability refers to the impacts on people's physical and mental health, disintegration of social cohesion and inadequate protection of the cultural heritage. Economic vulnerabilities refers to the processes needed to be done in order to protect major infrastructures, assets and property, both public and private, reduce the direct and indirect financial losses and support of the reconstruction and relief efforts of the affected people and societies. Last but not least, ecological vulnerability increases the risk of potential environmental impacts on flora and fauna, biotic and abiotic components, biodiversity, water pollution, and ecological systems in general.

There are three main types of flood and a number of special cases that are introduced in the flood insurance practices (Munich Re, 1997; Kron, 2005; Lotsch, et al., 2010):

- river flooding and inundation
- flash floods
- storm surge and coastal flooding

2.2.1. River flooding and inundation

River flooding at any location can be caused by rainfall or snowmelt and may be occurred at long distances from the affected location. Distant rainfall from snowmelt or monsoons may be the main drivers of flood on major river systems, rather than localized rainfall. The actual extent of flood will be a combination of all contributing water, whether distant or localized, and is strongly affected by prior water logging of soils. By nature of the shallow slopes of a natural floodplain, river inundation

and flood duration can last for days, or weeks. Recession of the floodwaters is a function of floodplain drainage (natural or artificial), slope, permeability and constrictions to water flow.

In specific, river flooding occurs when the capacity of a river system is insufficient to contain the flow of water in the river, resulting in escape of water from the normal perimeter and submergence of surrounding low-lying land. Prolonged rainfall results in soil saturation and may occur at times of increased inflow from tributaries. As mentioned before, characteristics of the river flooding are determined by the capacity of river channel(s), slopes, soil permeability, land cover, land use, and control of water flows by any man-made engineering structures (training walls, dams, drainage, etc.). River floods are often slow moving and increased tributary flow can affect flood plains, as a result of land degradation in the catchment areas. Flood plains are formed from deposits made by earlier floods. In terms of flood mitigation, river systems range from heavily managed to unmanaged. In practice, although it is possible to take measures to manage floods arising from rivers, it is not possible to control such floods completely. Most river systems have been engineered, for purposes of urban flood protection, agricultural protection and irrigation management. Flood detention areas may be designated, which generally allow controlled flooding of agricultural areas in order to protect urban regions.

2.2.2. Flash floods

Flash floods arise from intense, localized rainfall, and can happen practically anywhere. Intense rainfall can be measured over any specific period, typically between one hour and a maximum of six hours in the case of flash flooding and, regarding the duration of rainfall, it is longest in slow-moving or stationary storms. A characteristic of flash floods is that flood water rises suddenly, may be fast flowing, may collect in lower lying areas, and normally runs off and ponds rapidly. Residual ponded areas of water (sometimes larger lakes) may be trapped, remaining for long after the flash flood event. The flood impact of intensive rainfall is more severe when the ground is already saturated, where soils are impermeable or unstable, and in heavily sloped areas. Sequential intense rainfall events can therefore have a cumulative impact. Where ground is sloping, water is channeled to gullies and temporary watercourses, leading to erosion or landslides, and washing out bridges, culverts, or roads. Flash floods may also impact areas downstream of an intense rainfall event. Within a valley, flash floods can affect foothills, and rivers flood the valley bottoms. Within a country, regions may be affected by flash flooding whereas river flooding is the main national flood exposure.

2.2.3. Storm surge and coastal flooding

Coastal zones are subject to flooding as a result of storm surge - increased sea levels driven by tropical storm systems (cyclones) or by strong windstorms arising from intense offshore low-pressure systems. Coastal areas most at risk are low lying, either river deltas or coastal plains. The extent of flooding caused by a coastal sea surge will depend on several factors, especially the topography of the low-lying inland areas, tidal conditions, wind and wave action, extent of inland river flow at the time of coastal surge, and occurrence of localized rainfall associated with the storm event. Torrential rains associated with monsoons and tropical cyclones are also important factors adding to the impact of storm surges. In Asia, severe floods recur during the monsoon and rainy seasons, often with disastrous consequences. The major cause of the most destructive phenomena is a storm surge - a rapid rise of sea level resulting from strong winds driving the water ashore and causing flooding in low-lying coastal areas.

2.2.4. Other types of floods

Furthermore, we have to mention some additional special cases of types of flood that, in many cases, are responsible for large numbers of human and financial losses; Tsunami, waterlogging and urban flooding. Tsunami is a series of waves in a water body caused by the displacement of a large volume of water, generally in an ocean or a large lake. Waterlogging is a form of natural flooding, especially in flat areas, when underground water rises to surface level as the result of over-irrigation. Urban flooding appears in cases of high rainfall's intensity and occurs when the city sewage system and draining canals do not have the necessary capacity to drain away the amounts of rain that are falling.

2.3. The components of flood risk

National Oceanic and Atmospheric Administration (NOAA) defines flooding as *an overflowing of water onto land that is normally dry*. Floods are, in many cases, a natural phenomenon, such as in natural floodplains, and they could happen in small and large river basins, in estuaries, at coasts and locally. Nevertheless, floods occurrences cause troubles in catchments with extensive anthropogenic interventions, disturbing the established balance in agricultural and urban land use and planning. Each flood event can be characterized by features such as water depth, flow velocity, sediment transport fluxes and other spatiotemporal dynamics.

Flood hazard maps are designed to indicate the probability of flooding over space and serve as a critical decision-making tool for a range of end users including building/infrastructure developers and disaster response planners (Sampson et al., 2015). Flood risk can be defined as *the combination of the probability of occurrence of floods and the potentially adverse effects on human health, the environment, cultural heritage and economic activity associated with the occurrence of a flood* (European Union, 2007). In more detail, flood risk is interpreted as harm to flood-prone elements with a specific vulnerability due to probable extreme events with their features. However, the term of risk should not be confused with the risk in the sense of reliability of structural projects which are used as a safety measure against flood events.

The conceptual Source-Pathway-Receptor-Consequence-Model (SPRC-Model) has been proposed in order to describe flood risk (ICE, 2001; Figure 2.14) and offers a diagrammatic depiction of a simple causal chain which introduces parameters regarding the meteorological and hydrological characteristics of the flood events, either in inland or at coasts (sources), through the discharge and inundation (pathways) and the physical impacts on the affected environments at risk (receptors) to the assessment of effects (consequences). The chain links 'source', 'pathway' and 'receptor' refer to the physical process, whereas the assessment of the '(negative) consequence' is a matter of societal values.

According to Schanze et al. (2004), 'source' and 'pathway' represent the flood hazard. In more detail:

- 'Source' is determined by the probability (p) of flood events with a certain magnitude and other features (m). Early warning (w) and the retention capacity of the source areas of inland floods (t) can be considered as two risk reduction factors.
- The 'pathway' can be described by the inland discharge or coastal overflow and inundation (i) with various attributes (a) and interventions for flood control (c).
- 'Receptor' and '(negative) consequence' state the vulnerability, whereas 'receptor' specifies the susceptibility (s) with interventions to strengthen resistance and resilience (r).
- 'Consequence' stands for the harm to values (v; damage) with interventions to decrease or to compensate them (d).

Accordingly, flood risk can be expressed by the following function:

$$\textbf{Flood risk} = \text{function}\left((p, m, w, t)_{\text{source}}, (i, a, c)_{\text{pathway}}, (s, r)_{\text{receptor}}, (v, d)_{\text{consequence}}\right)$$

Practice has shown that the causal chain of the SPRC-Model can be applied for each element at risk and each flood hazard. Policy makers should also take into consideration the complex interrelations that exist between the mentioned components. Eventually, these components form the so-called "flood risk system". The investigation of the mechanisms of this system leads to integrated solutions regarding the prevention and confrontation of extreme flood events. For inland floods, this system focuses on river catchments and for coastal floods, focuses on coastal cells, considering them as areas which are hydraulically connected. The overall risk assessment associated with a flood risk system can be described as the sum of risks of all individual elements.

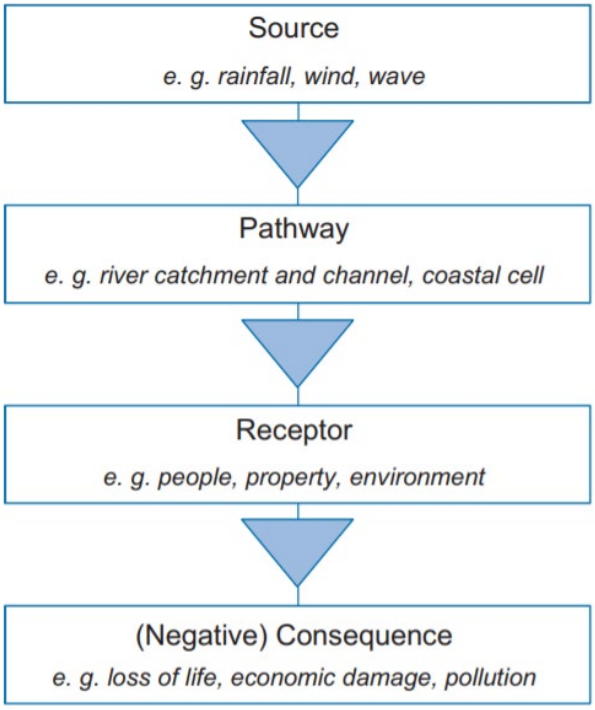


Figure 2.14 Source-Pathway-Receptor-Consequence-Model (ICE, 2001, modified).

In many of the following chapters, the vital role of flood risk assessment in the field of flood risk management context will be emerged, indicating its role on the reduction of the negative impacts of a potential flood event on individuals and societies, ensuring their sustainability.

2.4. Flood risk management

Flood risk management can be defined as the *continuous and holistic societal analysis, assessment and mitigation of flood risk* (Schanze et al., 2004). It includes tools for risk moderation regarding existing, under study or under construction systems in order to predict and control flood hazards and its negative consequences on individuals and societies with the assistance of instruments for risk management and reduction. The Source-Pathway-Receptor-Consequence-Model is inextricably linked with operational flood risk management, as presented on Figure 2.15.

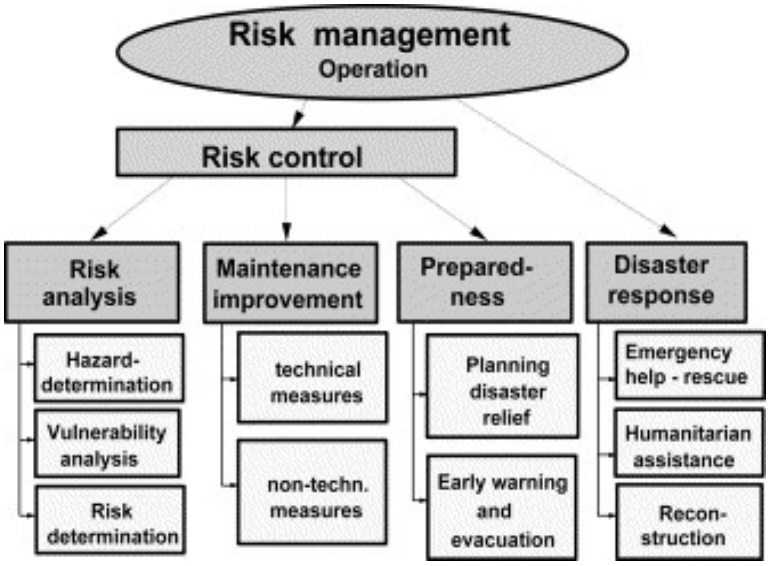


Figure 2.15 Stages of operational risk management (Eikenberg 1998, modified).

Risk management operation can be considered as the combination of actions that must be planed and implemented in order to control the evaluated risk and to response in case of a disaster outbreak.

Risk and vulnerability analysis are regularly performed in order to reassess hazard and risk according to new potential information and data available. Moreover, constant maintenance and improvement of the system is crucial in order to evaluate the existing risks, monitor changes and take technical and non-technical measures that decrease vulnerability. Furthermore, non-structural measures regarding preparedness planning should be considered, in order to provide the necessary decision support system for the unfortunate case that the flood protection system is being partially operated or failed. In addition, early warning systems and evacuation plans is a significant step towards the disaster mitigation, as an effective forecasting system permits the early identification and quantification of an imminent extreme flood event to which a population will be potentially

exposed to. Finally, designing integrated disaster response plans reduces losses of lives and accelerates humanitarian assistance (Plate, 2002).

The planning of disaster relief packages should be also introduced into the equation. Insuring assets and properties before the strike of a major flood event can provide sustainable financial solutions and the needed funding regarding the reconstruction process of the affected areas.

3. The partnership for flood risk reduction

3.1. General principles of insurance

Insurance is a means of protection from financial loss and is applied in the context of a risk management process, which aim is to hedge against the risk of a potential loss. Insurer or insurance company is called the entity that provides insurance services to other entities, such as companies or individuals, which are known as insured or policyholders. The insurance transaction involves the insured assuming a guaranteed and known relatively small loss in the form of payment to the insurer in exchange for the insurer's promise to compensate the insured in the event of a covered loss. The loss may or may not be financial, but it must be reducible to financial terms, and usually involves something in which the insured has an insurable interest established by ownership, possession, or pre-existing relationship (Vimala and Alamelu, 2018).

The insured receives a contract, called the insurance policy, which details the conditions and circumstances under which the insurer will compensate the insured. The amount of money charged by the insurer to the policyholder for the coverage set forth in the insurance policy is called the premium. If the insured experiences a loss which is potentially covered by the insurance policy, the insured submits a claim to the insurer for processing by a claims adjuster. The insurer may hedge its own risk by taking out reinsurance, whereby another insurance company agrees to carry some of the risk, especially if the primary insurer deems the risk too large for it to carry, such as aggregated risks regarding costly extreme flood events (Wikipedia, 2020).

3.1.1. Basic principle

Insurance involves pooling funds from many insured entities (known as exposures) to pay for the losses that some may incur. The insured entities are therefore protected from risk for a fee, with the fee being dependent upon the frequency and severity of the event occurring. In order to name a risk as insurable, the risk insured against must meet certain characteristics. Insurance as a financial intermediary is a commercial enterprise and a major part of the financial services industry, but individual entities can also self-insure through saving money for possible future losses (Gollier, 2003).

3.1.2. Insurability

The term insurability is used to indicate whether a particular client is insurable for by a particular company because of particular circumstance and the quality assigned by an insurance provider pertaining to the risk that a given client would have. Risks which can be insured by private companies typically share seven common characteristics (Mehr and Cammack, 1972):

- **Large number of similar exposure units**. Since insurance operates through pooling resources, the majority of insurance policies are provided for individual members of large classes, allowing insurers to benefit from the law of large numbers in which predicted losses are similar to the actual losses. However, all exposures will have particular differences, which may lead to different rates. For example, regarding flood insurance programs, rates may vary according to the flood zone the insured building belongs to.
- **Definite Loss**. The loss takes place at a known time, in a known place, and from a known cause. Flood events, fire, automobile accidents, and worker injuries may all easily meet this criterion. Other types of losses may only be definite in theory. Occupational disease, for instance, may involve prolonged exposure to injurious conditions where no specific time, place or cause is identifiable. Ideally, the time, place and cause of a loss should be clear enough that a reasonable person, with sufficient information, could objectively verify all three elements.
- **Accidental Loss**. The event that constitutes the trigger of a claim should be fortuitous, or at least outside the control of the beneficiary of the insurance. The loss should be 'pure,' in the sense that it results from an event for which there is only the opportunity for cost, such as the loss that is caused by a flood event on a building or on its contents. Events that contain speculative elements, such as ordinary business risks, are generally not considered insurable.
- **Large Loss**. The size of the loss must be meaningful from the perspective of the insured. Insurance premiums need to cover both the expected cost of losses, plus the cost of issuing and administering the policy, adjusting losses, and supplying the capital needed to reasonably assure that the insurer will be able to pay claims. For small losses these latter costs may be several times the size of the expected cost of losses. There is little point in paying such costs unless the protection offered has real value to a buyer. This is why, in many countries including Greece, the growth of the domestic insurance market for extreme flood events is weak.
- **Affordable Premium**. If the likelihood of an insured event is so high, or the cost of the event so large, that the resulting premium is large relative to the amount of protection

offered, it is not likely that anyone will buy insurance, even if on offer. Further, as the accounting profession formally recognizes in financial accounting standards, the premium cannot be so large that there is not a reasonable chance of a significant loss to the insurer. If there is no such chance of loss, the transaction may have the form of insurance, but not the substance.

- **Calculable Loss**. There are two elements that must be at least estimable, if not formally calculable: the probability of loss, and the attendant cost. Probability of loss is generally an empirical exercise, while cost has more to do with the ability of a reasonable person in possession of a copy of the insurance policy and a proof of loss associated with a claim presented under that policy to make a reasonably definite and objective evaluation of the amount of the loss recoverable as a result of the claim.
- **Limited risk of catastrophically large losses**. In most cases, insurable losses are considered as ideally independent and non-catastrophic, meaning that the losses do not happen all at once and individual losses are not severe enough to bankrupt the insurer. Nevertheless, taking into consideration the clustering mechanisms that characterize real-world hydroclimatic processes, this study investigates the effects of lack of fulfillment of this assumption on flood insurance practices.

3.1.3. Claims

Insurance claim can be defined as a formal request by a policyholder to an insurance company for coverage or compensation for a covered loss or policy event. Claims and loss handling is the materialized utility of insurance. In other words, it is the actual "product" paid for. In more detail, claims may be filed by insured directly with the insurer or through insurance broker. In order to deal with the large workflow, insurance company claims departments employ a large number of claims adjuster supported by a staff of records management and data entry clerk. In most cases, incoming claims are classified based on severity and are assigned to adjusters whose settlement authority varies with their knowledge and experience. The adjuster undertakes an investigation of each claim, which is a crucial process, usually in close cooperation with the insured, determines if coverage is available under the terms of the insurance contract, and if so, the reasonable monetary value of the claim, and authorizes payment.

The policyholder may hire their own public adjuster to negotiate the settlement with the insurance company on their behalf. For policies that are complicated, where claims may be complex, the

insured may take out a separate insurance policy add-on, called loss recovery insurance, which covers the cost of a public adjuster in the case of a claim. Managing the claims handling function can be a really challenging process, as insurers seek to balance the elements of customer satisfaction, administrative handling expenses, and claims overpayment leakages.

3.1.4. Indemnification

This term describes the process of reinstating the insured asset to the position that was is, in other words to make it whole again to the extent possible, prior to the happening of a specified event or peril. In general, we can categorize the types of insurance contracts that seek to indemnify an insured into three groups (Kulp and Hull, 1968):

- A "reimbursement" policy
- A "pay on behalf" or "on behalf of policy"
- An "indemnification" policy

In case of a “reimbursement” policy, the insured can be required to pay for a loss and then be reimbursed by the insurance carrier for the loss and out of pocket costs including, with the permission of the insurer, claim expenses. Regarding the "pay on behalf" policy, the insurance carrier would defend and pay a claim on behalf of the insured without any further action needed by the insured. Most modern liability insurance is written on the basis of "pay on behalf" language which enables the insurance carrier to manage and control the claim. Under an "indemnification" policy, the insurance carrier can generally either "reimburse" or "pay on behalf of" based on its decision which depends on the maximum potential benefit that comes out as a result from the claim handling process.

Concerning the insurance policy, an entity seeking to transfer risk (an individual, corporation, or association of any type, etc.) becomes the 'insured' party once risk is assumed by an 'insurer', the insuring party, by means of a contract. Generally, an insurance contract includes, at a minimum, the following elements: identification of participating parties (the insurer, the insured, the beneficiaries), the premium, the period of coverage, the particular loss event covered, the amount of coverage (i.e., the amount to be paid to the insured or beneficiary in the event of a loss), and exclusions (events not covered). An insured is thus said to be "Indemnity" against the loss covered in the policy.

When insured parties experience a loss for a specified peril, the coverage entitles the policyholder to make a claim against the insurer for the covered amount of loss as specified by the policy. Insurance premiums from many insured are used to fund accounts reserved for later payment of claims, in theory for a relatively few claimants, and for overhead (business) costs. So long as an insurer maintains adequate funds set aside for anticipated losses (called reserves), the remaining margin is an insurer's profit (Wikipedia, 2020).

3.1.5. Underwriting and investing

The business model of the insurance companies is to collect more in premium and investment income than is paid out in losses, and also to offer a competitive price which consumers will accept. This principle appears on flood insurance products, too. Profit can be reduced to a simple equation:

Profit = earned premium + investment income – incurred loss – underwriting expenses.

Insurers make money in two ways:

- Through underwriting, the process by which insurers select the risks to insure and decide how much in premiums to charge for accepting those risks
- By investing the premiums they collect from insured parties

It is a wide spread perception that the most complicated aspect of the insurance business is the actuarial science of ratemaking (price-setting) of policies, which uses statistics and probability to approximate the rate of future claims based on a given risk (Royal et al., 2014). After producing rates, the insurer will use discretion to reject or accept risks through the underwriting process. Regarding flood insurance products, this process can be really complex due to the multivariate factors that are introduced, such as the flood inundation modelling, several spatiotemporal dynamics elements under limited data availability and, in general, the uncertainties that characterize the calculation process.

Ratemaking process initially involves looking at the frequency and severity of insured perils and the expected average payout resulting from these perils. Thereafter an insurance company will collect historical loss data, bring the loss data to present value, and compare these prior losses to the premium collected in order to assess rate adequacy (Brown, 1993). Loss ratios and expense loads are also used. Rating for different risk characteristics involves at the most basic level comparing the

losses with "loss relativities" - a policy with twice as many losses would therefore be charged twice as much. More complex multivariate analyses are sometimes used when multiple characteristics are involved and a univariate analysis could produce confounded results. Other statistical methods may be used in assessing the probability of future losses.

Upon termination of a given policy, the amount of premium collected minus the amount paid out in claims is the insurer's underwriting profit on that policy (Feldstein and Fabozzi, 2008). Underwriting performance is measured by something called the "combined ratio", which is the ratio of expenses/losses to premiums. A combined ratio of less than 100% indicates an underwriting profit, while anything over 100 indicates an underwriting loss. A company with a combined ratio over 100% may nevertheless remain profitable due to investment earnings, as insurance companies earn investment profits on "float". Float, or available reserve, is the amount of money on hand at any given moment that an insurer has collected in insurance premiums but has not paid out in claims. Insurers start investing insurance premiums as soon as they are collected and continue to earn interest or other income on them until claims are paid out. Nevertheless, float method is difficult to carry out in an economically depressed periods.

3.2. Flood risk share and reduction; the role of insurance and reinsurance

Wolfgang Kron, the Head of Hydrological Risks in Munich Re's Geo Risks Research Department, states in his article "*Flood Risk = Hazard • Values • Vulnerability*" (2005) that the basic problem in flood insurance is the difference in the demand for coverage from potential clients who are exposed to flooding and the offer made by the insurance sector. It is evident that the fluctuation of demand, which usually depends on people's unfounded personal estimation on their potential exposure to flood risk, has a direct impact of many flood insurance parameters, such as the premium values and the amount of the total covered risk. This is why, in case that an insurance company wished to sell individual policies on a voluntary basis, the insurance premiums would have to be so high that policyholders would normally find them prohibitive. This phenomenon is called adverse selection or anti-selection. Furthermore, in other extreme flood phenomena, such as the storm surge hazard, the effect of adverse selection is even more severe.

Another problem that flood insurance policy-makers face is that premiums for flood insurance must, at least at some point, reflect the individual exposure. Nevertheless, an individual assessment of the risk and the calculation of an individual premium in most cases are impossible, so that the premium

must be fixed on the basis of a flat-rate assumption. This is why zones with a similar flood hazard must be identified and/or defined, within which the premiums are constant.

Regarding the viability of insurance and reinsurance companies, such companies must protect themselves against high losses in order to assure their survival by performing regular accumulation control processes, i.e. assess the probable maximum loss (PML) they may experience during an extreme event, evaluating their reserves and their reinsurance requirements as part of their portfolio analysis. Eventually, they estimate the aggregated losses in a regarded flood accumulation zone in order to obtain the probable maximum loss (accumulation).

Experience has shown that the most efficient way to cope with nature's destructive forces is on the basis of cooperation between the people and the government plus a third component, the insurance industry (Kron, 2005). This is why an integrated approach must be adopted, in which public authorities, affected individuals and insurance industry must be involved (Figure 3.1). In addition, an integral part of the process of designing an optimal strategy for risk reduction is also aggregated loss reduction and the implementation of disaster prevention mechanisms. The combined efforts resulting in structural and non-structural measures by the three groups allow minimization of the total costs and prevent and mitigate the impacts from floods and other natural events. Inevitably, such calculations become difficult when non-monetary losses and benefits are involved, such as the threat to human lives, environmental awareness and ecological protection.

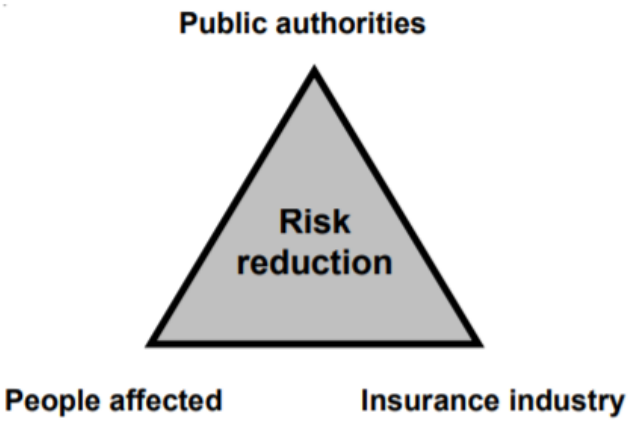


Figure 3.1The partnership for risk reduction (Kron, 2005, modified).

4. Methodology

4.1. Extreme value analysis (EVA) distributions

Extreme value analysis (EVA) is widely used and applied as a tool to analyze and study statistics on sample values that deviate extremely from the mean of the full sample, in order to develop a deeper understanding of the sample and precise modeling strategies. It generates significant applications across many scientific fields such as hydrology, insurance and finance and can be also used in order to predict the occurrence of rare events, such as extreme flooding, large insurance losses, crashing of the stock market and many others (Reis and Thomas, 2007).

One main application of extreme value analysis for the companies that operate in regional or global insurance market is to model the frequency of heavy damages caused by extreme flood events on urban or agricultural areas. It is important for them to balance the risk of damage with their potential consequences so as to make a profit or at least to minimize the financial losses over a fixed time-window. In this manner, generalized extreme value distribution (GEV) and generalized Pareto distribution (GPD) are introduced as a tool for the statistical analysis of maxima or minima and of exceedances over a given threshold.

4.1.1. Generalized Extreme Value distribution (GEV)

The development of stochastic methods for the characterization of flood peaks in drainage basins is one of the classical problems in extreme value statistics (Morrison and Smith, 2002). In particular, the generalized extreme value (GEV) distribution has been used for the modeling of flood peaks in at-site and regional settings. In addition to flood modeling, the GEV distribution is commonly used to model many other natural extreme events, such as the wind speed, air pollution concentration and precipitation maxima. The term “extreme events” often describes the maximum values of a quantity over a given period of time, such as the maximum annual discharge in a river.

The Generalized Extreme Value (GEV) distribution, introduced by Fisher and Tippett (1928) and developed furtherly by Jenkinson (1955), is a flexible three-parameter model that combines the Gumbel, Fréchet, and Weibull maximum extreme value distributions. Its probability density function is:

$$f(x) = \begin{cases} \frac{1}{\sigma}\exp\left(-(1+kz)^{-1/k}\right)(1+kz)^{-1-1/k} & for\ k \neq 0 \\ \frac{1}{\sigma}\exp(-z)\exp(-\exp(-z)) & for\ k = 0 \end{cases} \quad (4.1)$$

where $z = (x-\mu)/\sigma$, and k, σ, μ are the *shape*, *scale*, and *location* parameters respectively. The *scale* must be positive ($\sigma > 0$), the *shape* and *location* can take on any real value. The range of definition of the GEV distribution depends on k:

$$\begin{array}{ll} 1 + k\frac{(x-\mu)}{\sigma} > 0 & for\ k \neq 0 \\ -\infty < x < \infty & for\ k = 0 \end{array} \quad (4.2)$$

Its cumulative distribution function is:

$$F(x) = \begin{cases} \exp\left(-\left(1+\frac{k}{\sigma}(x-\mu)\right)^{1/k}\right) & for\ k \neq 0 \\ \exp\left(-\exp\left(-\frac{(x-\mu)}{\sigma}\right)\right) & for\ k = 0 \end{cases} \quad (4.3)$$

Various values of the *shape* parameter yield the extreme value type I, II, and III distributions. Specifically, the three cases $k = 0$, $k > 0$, and $k < 0$ correspond to the Gumbel, Fréchet, and "reversed" Weibull distributions. The reversed Weibull distribution is a quite rarely used model because it is bounded on the upper side and thus it is not appropriate for hydrological extremes. In more detail:

- $\mu + k/\sigma \leq x < +\infty$ when $k < 0$ (Fréchet),
- $-\infty < x < +\infty$ when $k = 0$ (Gumbel) and
- $-\infty < x \leq \mu + \sigma/\kappa$ when $k > 0$ (Weibull).

4.1.2. Generalized Pareto distribution (GPD)

The peak over threshold method (POT) has become one of the most preferable extreme value approaches in insurance. Largely, the reason is the obvious drawback of the traditional EVA by the block maxima method which discards potentially useful information, especially considering the typical short records available. Indeed, if there is more than one large loss in a given block, only the largest loss in the block is used in the subsequent analysis. Information loss of this kind is very likely to happen with insurance data considering extreme flooding events, due to the well-known stylized fact of volatility clustering, as large changes in prices tend to cluster together, resulting in

persistence of the amplitudes of price changes (Cont, 2007). This drawback is eliminated in the POT model of extreme losses by using all losses in a sample larger than some pre-specified threshold value. The probability density function for the generalized Pareto distribution, introduced by Pickands (1975), with *shape* parameter $k \neq 0$, *scale* parameter σ, and threshold parameter θ, is:

$$f(x) = \left(\frac{1}{\sigma}\right)\left(1 + k\frac{(x-\theta)}{\sigma}\right)^{-1-\frac{1}{k}} \tag{4.4}$$

for $\theta < x$, when $k > 0$, or for $\theta < x < \theta - \sigma/k$ when $k < 0$. For $k = 0$, the density is:

$$f(x) = \left(\frac{1}{\sigma}\right)e^{-\frac{(x-\theta)}{\sigma}} \tag{4.5}$$

for $\theta < x$. The probability distribution function for the generalized Pareto distribution is:

$$F(x) = \begin{cases} 1 - \left(1 - \dfrac{kx}{\sigma}\right)^{\frac{1}{k}} for\ k \neq 0 \text{ and } \sigma > 0 \\ 1 - \exp\left(-\dfrac{x}{\sigma}\right)\ for\ k = 0 \text{ and } \sigma > 0 \end{cases} \tag{4.6}$$

The range of x is $x > 0$ for $k \leq 0$ and $0 < x < \frac{\sigma}{k}$ for $k > 0$. The case $k = 0$, which is the exponential distribution, is the limiting distribution as $k \to 0$. The following values of the parameter k are of particular interest:

- When $k = 0$, the GPD reduces to the exponential distribution with mean σ.
- When $k = 1$, the GPD becomes a uniform $U[0, \sigma]$ distribution.
- When $k \leq -\frac{1}{2}$, $\text{var}(X) = \infty$. In fact, the rth central moment exists only if $k > -\frac{1}{r}$.
- When $k < 0$, the GPD reduces to the Pareto distribution.

4.2. Threshold selection process

The wide use of generalized Pareto model in insurance, which deals with exceedances over a threshold, offers us the opportunity to investigate the key-role of the threshold selection on this POT analysis.

Threshold selection is a challenge in insurance and especially in flood insurance practices. The threshold should be chosen such that all losses above the threshold are “extreme losses” in the sense

of the underlying extreme value analysis. This clearly leads to some arbitrariness in the choice of the threshold value and also to a non-trivial trade-off. On the one hand, for the underlying theory to go through, we want to choose a high threshold in order to investigate the behavior of the (really) extreme events. On the other hand, for the estimation of the parameters in the distribution of the extreme losses, we need many observations above the threshold (i.e., we want to choose a low threshold) in order to create a solid statistical foundation for our conclusions, based on a long sequence of values.

Figure 4.1 presents the observed streamflows as well as the ones that were developed by the process of fitting these observed data with the generalized Pareto distribution regarding one gauge station; it clearly shows that the selection of the threshold affects directly the cumulative probability of a specific streamflow value considering the over-threshold events. The dominant trend in insurance practices is to select high percentage thresholds (99% or greater) in order to analyze exclusively high-impact extreme flood events, which are mainly responsible for the large amounts of claims and compensations that insurance companies will have to pay to their clients. Although selecting a threshold of a significantly high percentage is a desirable option, it is not always a possible one. The main reason is that, in many cases, the length of the available observed time series is quite short and, as a result, selecting a high percentage threshold leads to inaccurate conclusions regarding their statistical behavior. In order to characterize the dynamics of extreme streamflow values, this study performed a POT analysis using four different percentage thresholds (90%, 95%, 98%, and 99%).

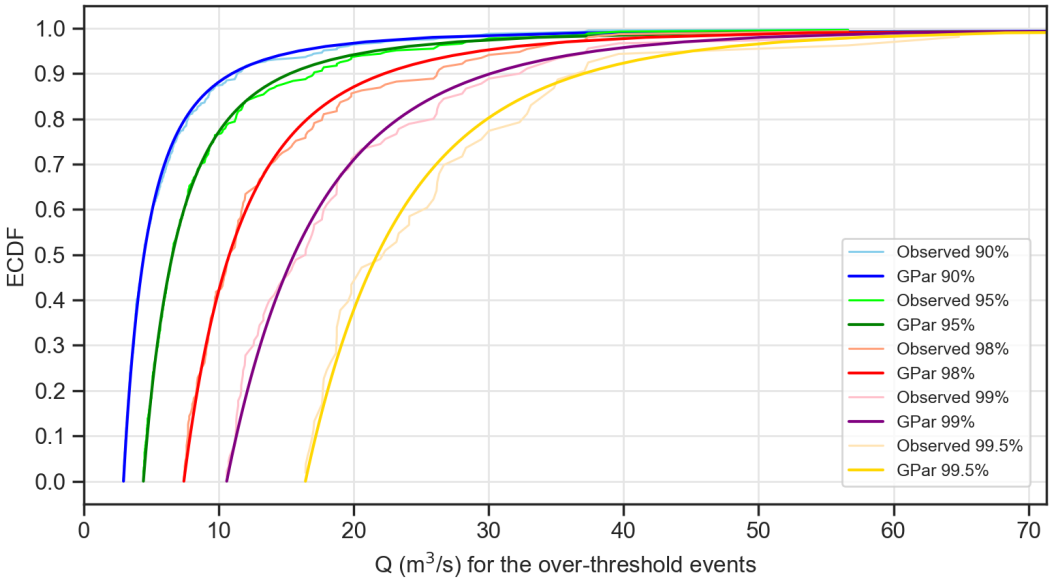


Figure 4.1 Diagram that shows the impact of threshold selection on ECDF of streamflow of the over-threshold events. Gauge ID: 01552500.

4.3. Collective risk model

The theoretical background on the need of existence of insurance sector can be explained by the expected utility model, in which an insured is a risk averse and rational decision maker, who by virtue of Jensen's inequality is ready to pay more than the expected value of his claims just to be in a secure financial position. In mathematical terms, the Theorem of Jensen's inequality is developed by Kaas et al. (2008). In more detail, if $v(x)$ is a convex function and Y is a random variable, then

$$\mathrm{E}[v(Y)] \geq v(\mathrm{E}[Y]) \tag{4.7}$$

with equality if and only if $v(\cdot)$ is linear on the support of Y (the set of its possible values), or $\mathrm{Var}[Y] = 0$. From this inequality, it follows that for a concave utility function

$$\mathrm{E}[u(w - X)] \leq u(\mathrm{E}[w - X]) = u(w - \mathrm{E}[X]) \tag{4.8}$$

Apparently, decision makers with such utility functions prefer to pay a fixed amount $\mathrm{E}[X]$ instead of a risky amount X, so they are indeed *risk averse*. Now, suppose that a risk averse insured with capital w has the utility function $u(\cdot)$. Assuming he is insured against a loss X for a premium P, his expected utility will increase if

$$\mathrm{E}[u(w - X)] \leq u(w - P) \tag{4.9}$$

Since $u(\cdot)$ is a non-decreasing continuous function, this is equivalent to $P \leq P^{+}$, where P^{+} denotes the maximum premium to be paid. This so-called zero utility premium is the solution to the following utility equilibrium equation

$$\mathrm{E}[u(w - X)] = u(w - P^{+}). \tag{4.10}$$

The insurer, say with utility function $U(\cdot)$ and capital W, will insure the loss X for a premium P if $\mathrm{E}[U(W + P - X)] \geq U(W)$, hence $P \geq P^{-}$ where P^{-} denotes the minimum premium to be asked. This premium follows from solving the utility equilibrium equation reflecting the insurer's position:

$$U(W) = \mathrm{E}[U(W + P^{-} - X)]. \tag{4.11}$$

A deal improving the expected utility for both sides will be possible if $P^+ \geq P^-$.

In this model, parameters such as the insured loss X, the capitals w of the insured risk averse, the insurer's capital W, the premium P, but also the maximum and minimum premium P^+ and P^- to be asked to be paid are introduced. The mechanism through which decisions are taken under uncertainty is not by direct comparison of the expected payoffs of decisions, but rather of the expected utilities associated with these payoffs. Additional parameters which have a significant role in this mechanism are the aggregated number of claims but also the claims amounts to be paid. Investigating the influence of uncertainty in the whole process, which characterizes and links many of these parameters, is a key element of this study in order to develop accurate risk assessment procedures and modelling strategies regarding flood insurance practices.

The distribution of total claim amounts, considering the insurance company's portfolio as a collective that produces a random number N of claims in a certain time period, can be described by the collective risk model (Kaas et al., 2008). Collective risk is defined as

$$S_x = X_1 + X_2 + \cdots + X_N, \tag{4.12}$$

where X_i is the ith claim amount. The terms of S_x correspond to actual claim amounts. Apparently, $S_x = 0$ if $N = 0$. Similarly, regarding flood insurance practices and in case of an extreme flood event, the collective risk S is the total claim amount, considering again the portfolio of (re)insured properties as a collective that produces a random number N of claims in a certain time period of one year in our case. Denoting the records y_t of a time series, a proxy of temporal collective risk S is defined by Serinaldi and Kilsby (2016) as

$$S = \sum_{j=1}^{N} Y_j \tag{4.13}$$

where Y_j is the jth claim amount proxy (over-threshold flow fluctuation severity). Again, the total claim amounts $S = 0$ if $N = 0$. In case of rainfall extremes investigation, over-threshold rainfall fluctuation severity is considered. Although the definition of collective risk regarding flood insurance practices is a proxy of the actual collective risk, as it involves hydrological series and not actual claim amounts, it will be called in this study collective risk as well.

The underlying assumption of this definition, which is applied abundantly in insurance practices, is that the aggregate amount S is to require that N and the individual claim amounts Y_j are independent and identically distributed, and also that N and all claim amounts Y_j are independent to each other. Additionally, it is usually assumed that the number of claims is a single Poisson process, or allows for some overdispersion by using the negative binomial distribution instead. This study evaluates the impact of dependence and spatiotemporal clustering mechanisms on the calculations, investigating the validity of the independence assumption.

In order to characterize the dependence and the clustering mechanisms, it is important to quantify how the time series differs from a sequence of independent variables. A widely used method to create a sequence of independent variables is to shuffle (randomize) the series in order to get a new series which has the same dimensional distribution but no correlation; the quantification of the distance between the independent and the observed variables is performed by comparing specific characteristics, i.e. the annual collective risk, the duration of the peak-over-threshold events and the occurrence frequency of return intervals in the original time series and in the shuffled one. Hence, in order to assess the clustering of extremes of the 360 observed time series, 100 new shuffled time series were reproduced for each one of the 360 original time series.

4.4. The Hurst-Kolmogorov dynamics

Long-term dependence and persistence is a theoretical property that was described mathematically by A. Kolmogorov (1941) on his work on turbulence characteristics and discovered by H.E. Hurst (1951) in the physical world during his investigation on long-term capacities of reservoirs. Nowadays, scientific research has revealed the existence of such behavior in many natural processes, including streamflow and rainfall dynamics, as a step towards the deeper understanding of the potential patterns of their underlying processes. The exhibited persistence is known as the Hurst phenomenon or Hurst-Kolmogorov (HK) dynamics and is quantified by the Hurst coefficient H. In more detail, for:

- $0 \leq H < 0.5$, the process is known as antipersistent (or anticorrelated)
- $H = 0.5$, the process is equivalent to white noise, meaning that there is no long term change (dependence) or persistence in the sample
- $0.5 < H \leq 1$, the process has enhanced long-term persistence (or positively correlated), which is the most common behavior on hydroclimatic processes

Regarding flood insurance practices, recent research has revealed the significance of Hurst-Kolmogorov dynamics, persistence and inherent uncertainties in real-world hydrometeorological processes (Koutsoyiannis, 2011; Dimitriadis, 2017), on flood inundation and flood mapping (Dimitriadis et al., 2016). These uncertainties affect directly the abovementioned parameters, such as the insured losses, the premiums to be asked to be paid by the clients, expected claim amounts and insurer's capital reserves. In other words, hydrological and hydraulic uncertainties are introduced in the financial calculations and are to be interpreted in financial terms though a risk estimation process. However, classic risk estimation for flood insurance practices is formulated under the assumption of temporal independence of extreme flood events, which is unlikely to be tenable in real-world hydrometeorological processes exhibiting long range dependence (Iliopoulou and Koutsoyiannis, 2019). Moreover, multiple analyses on observed and historical data worldwide note that floods and streamflow extremes have a tendency to exhibit a behavior that is closely related to short range and long range dependence. These clustering mechanisms have a prominent effect on spatiotemporal dynamics of streamflow and rainfall extremes. In this study, the effect of such clustering mechanisms on particular insurance parameters, such as the total claim amounts, is investigated.

4.5. Climacogram

In order to calculate the Hurst coefficient H and detect the potential long term dependence (or else persistence, clustering) of a process, the most accurate method is by formulating the *Climacogram* (Koutsoyiannis, 2010; Dimitriadis and Koutsoyiannis, 2015). *Climacogram* is a two dimensional plot, typically depicted on a double logarithmic plot, of the variance $\gamma(k)$ of the mean-aggregated series of the random variable Z on the vertical axis, and the aggregated scale k on the horizontal axis:

$$Z_u^{(k)} = \frac{1}{k} \sum_{i=(u-1)k}^{uk} Z_i \tag{4.14}$$

where Z and Z_u represent the random stochastic process and the mean aggregated stochastic process respectively, while u is the vector index of the field showing the lag; i.e. the location in the field.

The *Climacogram* (Dimitriadis and Koutsoyiannis, 2015; Table 4.1), depends on the nature of the stochastic process; there is a fundamental difference between continuous and discrete time processes:

Table 4.1 The *Climacogram*.

Type	***Climacogram***
Continuous	$\gamma(m) := \frac{\mathrm{Var}\left[\int_t^{t+m} \underline{x}(\xi)\mathrm{d}\xi\right]}{m^2} = \frac{\mathrm{Var}\left[\int_0^{m} \underline{x}(\xi)\mathrm{d}\xi\right]}{m^2}$ where m $\in \mathbb{R}^+$ and $\gamma(0) := \mathrm{Var}[\underline{x}(t)]$
Discrete	$\gamma_d^{(\Delta)} := \frac{\mathrm{Var}\left[\sum_{l=k(i-1)+1}^{ki} \underline{x}_l^{(\Delta)}\right]}{k^2} = \frac{\mathrm{Var}\left[\sum_{l=1}^{ki} \underline{x}_l^{(\Delta)}\right]}{k^2} = \gamma(k\Delta)$ where k $\in \mathbb{N}$ is the dimensionless scale for a discrete time process
Classical estimator	$\hat{\gamma}_d^{(\Delta)} = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{1}{k}\left(\sum_{l=k(i-1)+1}^{ki} \underline{x}_l^{(\Delta)}\right) - \frac{\sum_{l=1}^{n} \underline{x}_l^{(\Delta)}}{n}\right)$
Expectation of classical estimator	$\mathrm{E}\left[\hat{\gamma}_d^{(\Delta)}(k)\right] = \frac{1-\frac{\gamma_d^{(\Delta)}(n)}{\gamma_d^{(\Delta)}(k)}}{1-\frac{k}{n}}\gamma_d^{(\Delta)}(k)$

4.6. Generalized-HK (GHK) process

In some cases, such as in this study, fitting of straight line in the *Climacogram* derived from the observed data cannot capture the full variance behavior of the process at the whole range of scales. Thus, the generalized-HK (GHK) process is applied, which exhibits also an HK behavior in large scales but has more flexibility in smaller scales. It is a method that can preserve explicitly (i.e. fully analytical calculations) four marginal moments of a process for any type of second-order dependence structure (Dimitriadis and Koutsoyiannis, 2018). The *Climacogram* of the model is:

$$\gamma(k) = \frac{\lambda}{(1+k/q)^{2-2H}} \tag{4.15}$$

where the Hurst coefficient H is bounded between zero and one inclusive, q is positive, while λ and q have dimensions [x^2] and T, respectively.

4.7. Symmetric-moving average (SMA) method

In this study, the symmetric moving average (SMA) method (Koutsoyiannis 2000 and 2016; Dimitriadis and Koutsoyiannis, 2018) is applied in order to develop and evaluate potential modeling strategies. SMA is a general method for producing synthetic time series of a physical quantity by

preserving its dependence structure. In particular, SMA generation scheme for approximating the marginal probability function can replicate a natural process by exactly preserving its first four central moments, which has been found to be sufficient for various distributions commonly applied in geophysical processes.

The SMA method is described by the following equation:

$$\underline{x}_i = \sum_{j=-l}^{l} a_{|j|}\underline{v}_{i+j} \tag{4.16}$$

where $\underline{x}_i$ is any process with any type of dependence, $a_{|j|}$ are coefficients calculated from the autocovariance function, $\underline{v}_{i+j}$ is white noise averaged in discrete-time and l theoretically equals infinity but a finite number can also be used for preserving the dependence structure up to lag l.

The algorithm to produce time series with the SMA scheme, created by P. Dimitriadis (2018), required the first four central moments, the H coefficient of each physical quantity (average, maximum and minimum) and the length of the time series as well.

4.8. Pearson, Spearman and Kendall correlation

Pearson's correlation coefficient (Pearson, 1895) when applied to a sample is commonly represented by r_{xy} and may be referred to as the sample correlation coefficient or the sample Pearson correlation coefficient. We can obtain a formula for r_{xy} by substituting estimates of the covariances and variances based on a sample into the formula above. Given paired data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ consisting of n pairs, r_{xy} is defined as:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{4.17}$$

where:

- n is sample size
- x_i, y_i are the individual sample points indexed with i
- $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

The *Spearman correlation coefficient* (Spearman, 1904) is defined as the Pearson correlation coefficient between the rank variables. For a sample of size n, the n raw scores X_i, Y_i are converted to ranks $\mathrm{rg}X_i$, $\mathrm{rg}Y_i$, and r_s is computed as

$$r_s = \rho_{rg_X, rg_Y} = \frac{\mathrm{cov}(rg_X, rg_Y)}{\sigma_{rg_X}\sigma_{rg_Y}} \tag{4.18}$$

where:

- ρ denotes the usual Pearson correlation coefficient, but applied to the rank variables
- x_i, y_i are the individual sample points indexed with i
- $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ (the sample mean); and analogously for $\bar{y}$

Regarding the *Kendall correlation coefficient* (Kendall, 1938), let $(x_1,y_1),\ldots,(x_n,y_n)$ be a set of observations of the joint random variables X and Y, such that all the values of (x_i) and (y_i) are unique (ties are neglected for simplicity). Any pair of observations (x_i,y_i) and (x_j,y_j), where $i < j$, are said to be *concordant* if the sort order of (x_i,x_j) and (y_i,y_j) agrees: that is, if both $x_i > x_j$ and $y_i > y_j$ holds or both $x_i < x_j$ and $y_i < y_j$; otherwise they are said to be *discordant*.

The Kendall τ coefficient for n items is defined as:

$$\tau = \frac{(number\ of\ concordant\ pairs) - (number\ of\ discordant\ pairs)}{\binom{n}{2}} \tag{4.19}$$

5. Dataset

5.1. US-CAMELS dataset

This analysis is applied on the US-CAMELS dataset which comprises 671 daily streamflow time series from catchments in the contiguous United States (CONUS) minimally impacted by human activities (Newman et al., 2014). From this dataset, 360 streamflow time series with the maximum temporal overlap (namely, 35 years from 1980 to 2014) and less than 10% of missing values were selected. Figure 5.1 shows the study area and stream gauge locations for the full dataset and Figure 5.2 shows the selected 360 stream gauge locations. The list of the 360 selected gauge locations and their characteristics are provided in the appendix (Table A-1).

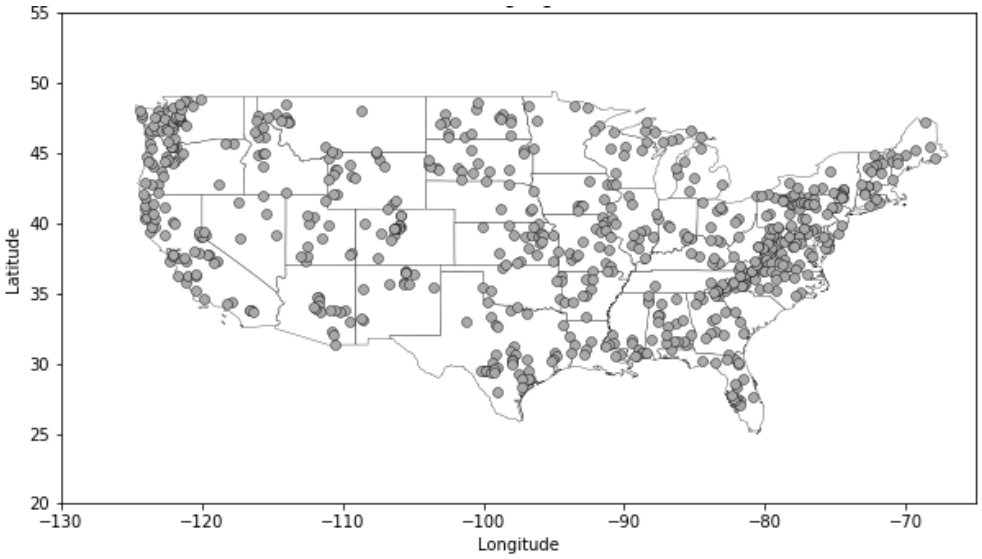


Figure 5.1 The 671 US-Camels stream gauge locations.

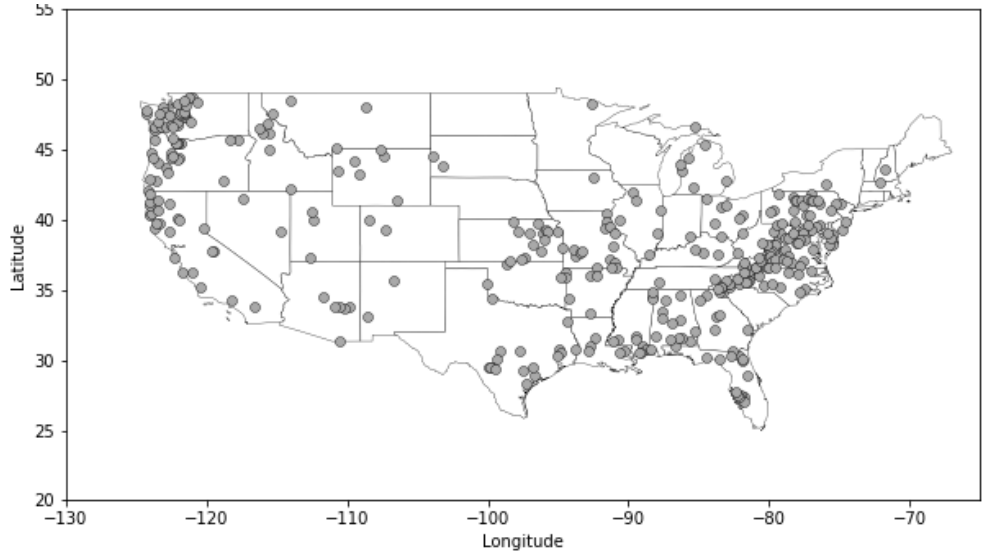


Figure 5.2 The selected 360 US-Camels stream gauge locations.

5.2. Hydrologic Units in USA

The United States Geological Survey (USGS) created a hierarchical system of hydrologic units originally called regions, sub-regions, accounting units, and cataloging units. Each unit was assigned a unique Hydrologic Unit Code (HUC). As first implemented the system had 21 regions, 221 subregions, 378 accounting units, and 2 264 cataloging units (Seaber, P.R., et al., 1987). The first level of classification divides the USA into 21 major geographic areas (Figures 5.3 and 5.4), or regions. These geographic areas contain either the drainage area of a major river, such as the Missouri region, or the combined drainage areas of a series of rivers, such as the Texas-Gulf region, which includes a number of rivers draining into the Gulf of Mexico.

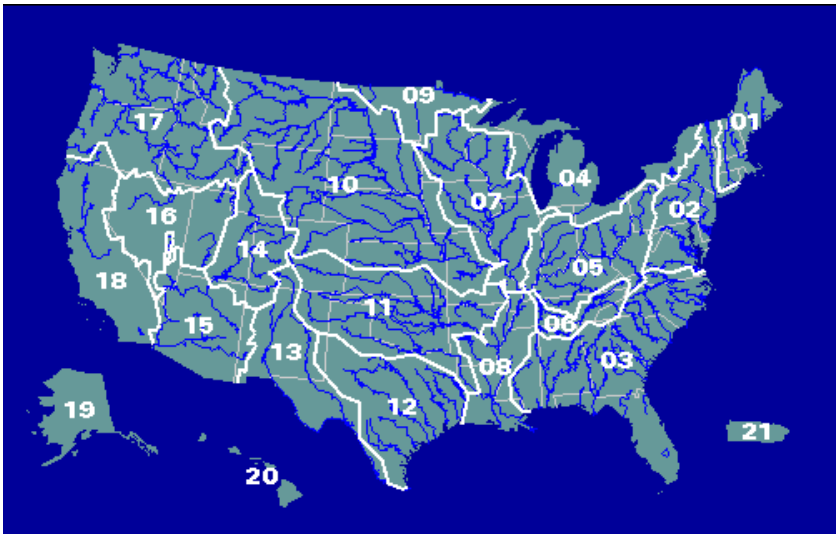


Figure 5.3 The 21 hydrological units of the USA. The gray lines are state lines, the blue lines are major rivers, and the white lines are water-resources region boundary lines (Seaber, P.R., et al., 1987).

Figure 5.4 The names of the 21 Hydrological Units in the USA (Wikipedia, 2020).

Eighteen of the regions occupy the land area of the conterminous United States. Alaska constitutes region 19, the Hawaii Islands are region 20, and Puerto Rico and other outlying Caribbean areas are region 21.

5.3. FEMA's NFIP claims records

The Federal Emergency Management Agency (FEMA) is an agency of the United States Department of Homeland Security and its primary purpose is to coordinate the response to a disaster that has occurred in the United States and that overwhelms the resources of local and state authorities.

The National Flood Insurance Program (NFIP) is a program administered by the FEMA and has a twofold purpose; to share the risk of flood losses through flood insurance and to reduce flood damages by restricting floodplain development. Furthermore, the program enables property owners in participating communities to purchase insurance protection, administered by the government, against losses from flooding. In addition, it requires flood insurance for all loans or lines of credit that are secured by existing buildings, manufactured homes, or buildings under construction, that are located in the Special Flood Hazard Area in a community that participates in the NFIP. Moreover, it is designed to provide an insurance alternative to disaster assistance to meet the escalating costs of repairing damage to buildings and their contents caused by flood (FEMA, 1986).

In more detail, according to the Federal Emergency Management Agency (FEMA): "*The National Flood Insurance Program (NFIP) aims to reduce the impact of flooding on private and public structures. It does so by providing insurance to property owners, renters and businesses and by encouraging communities to adopt and enforce floodplain management regulations. These efforts help mitigate the effects of flooding on new and improved structures. Overall, the program reduces the socio-economic impact of disasters by promoting the purchase and retention of general risk insurance, but also of flood insurance, specifically*." (FEMA NFIP, 2019).

On June 11 2019 the Federal Emergency Management Agency (FEMA) published National Flood Insurance Program (NFIP) data including more than two million claims records dating back to 1970 and more than 47 million policy records for transactions from the past ten years on its OpenFEMA website (FEMA, 2019). This data supplements existing NFIP data through OpenFEMA and provides additional data of interest. It is evident that this is a giant contribution for supporting

scientists and policy-makers on their research on how the National Flood Insurance Program (NFIP) works, where flood damage occurs, and what the costs are.

"This data demonstrates FEMA's commitment to build a culture of preparedness by providing insights to our stakeholders that can help close the nation's insurance gap. It gives the insurance industry, researchers, and the public the ability to analyze and evaluate this program. Insurance is the best tool to financially protect you and your family before a disaster." declared Dr. Daniel Kaniewski, FEMA's Deputy Administrator for Resilience, in a news release (FEMA, 2019).

"The proactive publication of this data will assist the private market to grow in the flood insurance space and help close the insurance gap. The private market will now be able to identify areas with prior flood claims and historical flood insurance policies," said David Maurstad, FEMA's Deputy Associate Administrator for Insurance and Mitigation (FEMA, 2019).

However, as useful as these data could be, the dataset does not include the exact addresses of affected buildings, to protect policyholders' privacy. Although it includes ZIP code-level data on where policyholders received payments, a home buyer might not be able to learn the full history of flood risk for a property.

The published data enables analysis of how coverage has changed in a geographic area, and where NFIP claims have been filed for more than 40 years. Information such as: state, census tract, ZIP code, year of loss, and amount paid on claims are included.

This dataset consists of around 2.4 million observations of 39 variables. The dataset's variables that are used in our study, accompanied by their data dictionary description, are the following:

- `yearofLoss:` Year in which the flood loss occurred (YYYY).
- `countyCode:` The Federal Information Processing Standard (FIPS) defined unique 5-digit code for the identification of counties and equivalent entities of the united States, its possessions, and insular areas. The NFIP relies on the geocoding service to assign county.
- `state:` The two-character alpha abbreviation of the state in which the insured property is located.

- `latitude`: Approximate latitude of the insured building (to 1 decimal place). This represents the approximate location of the insured property. The precision has been lessened to ensure individual privacy.

- `longitude:` Approximate longitude of the insured building (to 1 decimal place). This represents the approximate location of the insured property. The precision has been lessened to ensure individual privacy.

- `amountPaidOnBuildingClaim:` Dollar amount paid on the building claim. In some instances, a negative amount may appear which occurs when a check issued to a policy holder isn't cashed and has to be re-issued.

- `amountpaidoncontentsclaim:` Dollar amount paid on the contents claim. In some instances, a negative amount may appear which occurs when a check issued to a policy holder isn't cashed and has to be re-issued.

While there are supposed to be records from 1970, the records before 1978 seem to be pretty sparse. We also don't have a full year of records for 2021 yet. As such, records are filtered to comprise years between 1970 and 2021 (last data refresh: 10-24-2021).

5.4. A graphical and diagrammatic visualization of the FEMA's NFIP claims records

Figure 5.5 shows the number of claims per year. The years with the largest aggregated number of claims are (in descending order) 2005, 2012 and 2017. The common characteristic of these years is the occurrence of at least one of the five costliest Atlantic hurricanes in US history. The normalized damages reported in Table 5.1 give an estimation of the direct economic loss of these five Atlantic hurricanes if the same event was to occur under contemporary societal conditions, as described on Weinkle et al. (2018). The applied general formula for normalized losses for the adjustment at 2018 US$ value (D_{2018}) is:

$$D_{2018}=D_y \text{ x } I_y \text{ x } RWPC_y \text{ x } P_{2018/y} \qquad (5.1)$$

where D_y is reported damage in current-year US dollars, I_y is the GDP deflator for inflation adjustment, $RWPC_y$ is an estimate of current-cost net stock of fixed assets and consumer durable goods to capture changes in real wealth per-capita, and $P_{2018/y}$ is county population adjustment.

Table 5.1 The nominal and normalized direct economic losses of the five costliest Atlantic hurricanes in US history.

Name	Season	Nominal damage	Normalized damage
Katrina	2005	$125.0 Billion USD	$116.9 Billion USD
Harvey	2017	$125.0 Billion USD	$62.2 Billion USD
Maria	2017	$90 Billion USD	N/A
Irma	2017	$77.2 Billion USD	$31.0 Billion USD
Sandy	2012	$68.7 Billion USD	$73.5 Billion USD

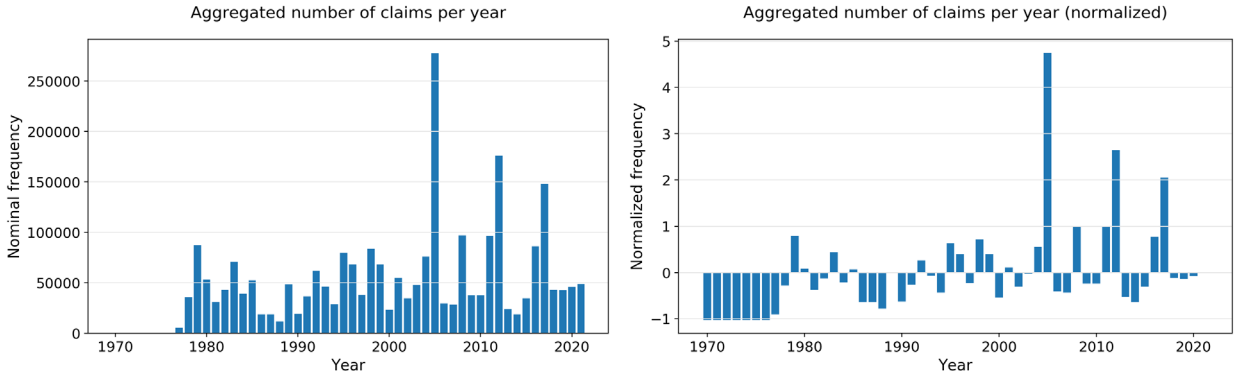


Figure 5.5 The aggregated number of claims per year (1970-2021). On the left diagram, the nominal values are presented, whereas, on the right diagram, the normalized values are presented by subtracting the mean from the individual values and dividing the difference by the standard deviation. Last data refresh: 10-24-2021.

Apart from the number of claims that occurred due to the hurricanes (storm surge and coastal flooding), numerous claims are caused by other types of floods, similar to the ones that are described in section 2.4, such as river flooding and flash floods. At this point, someone can conceive the difficulty that researchers and policy makers face during the really complex process of quantifying the mechanisms that generate every single flood event in order to investigate ways of eliminating their causes and impacts.

FEMA's NFIP claims dataset consists of aggregated claims provoked by all potential types of floods, and thus it is difficult to attribute claims records to a specific flood type, i.e. river flooding as in our study.

One significant and clear conclusion that is highlighted by the analysis of this dataset is the spatial distribution of the recorded claims. Figures 5.6-5.13, based on R scripts originally created by K. Tay (2019), present the number of claims across the United States and show us that East Coast and

Gulf states are by far the most flood-affected areas in terms of number of claims. These records indicate the need for further research on the field of flood risk management and the development of integrated financial solutions offered by the flood insurance sector in order to moderate the impacts of these extreme flood events.

Recently, FEMA designed and published some interactive data visualization instruments that offer a better understanding of the historical flood risk across the USA and potential flood-related costs (FEMA, 2020). Among others, it is mentioned that the percent of U.S. counties that are impacted by a flooding event is 99% (1996-2019), the average flood claim payout from the NFIP in 2019 was $52,000 and that the average annual flood insurance policy premium cost in 2019 was $700.

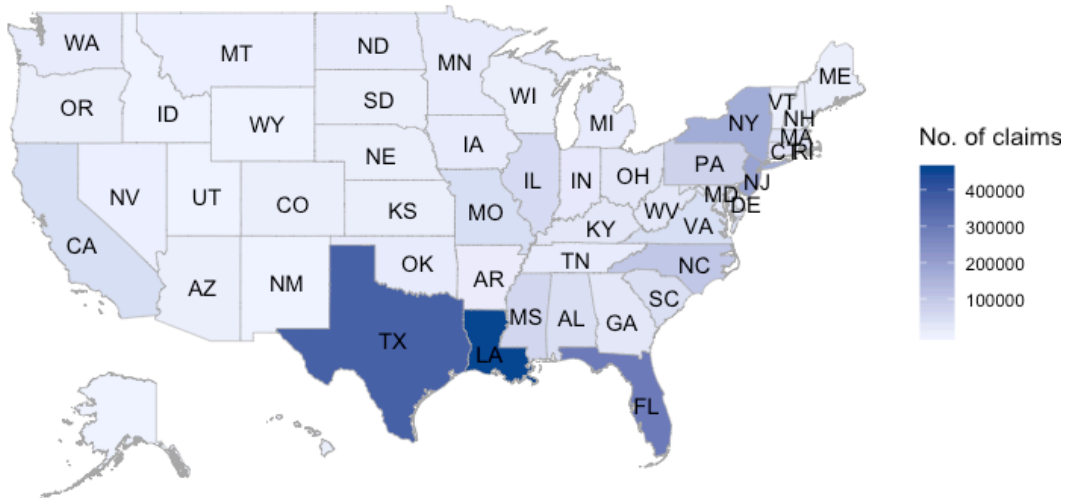


Figure 5.6 A per state depiction of the aggregated number of claims. It is clear that Louisiana experienced the majority of claims, followed by Texas and Florida (1970-2018).

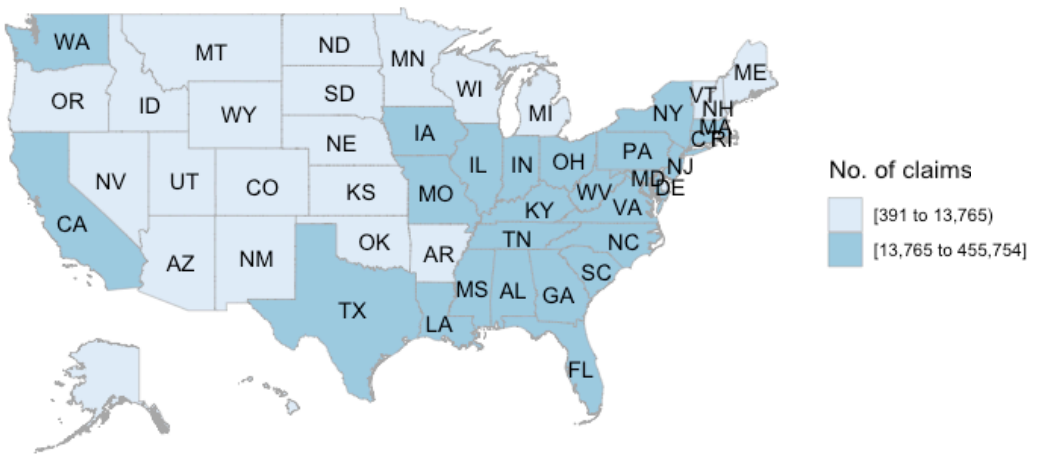


Figure 5.7 The claims values of all states split into 2 buckets (1970-2018).

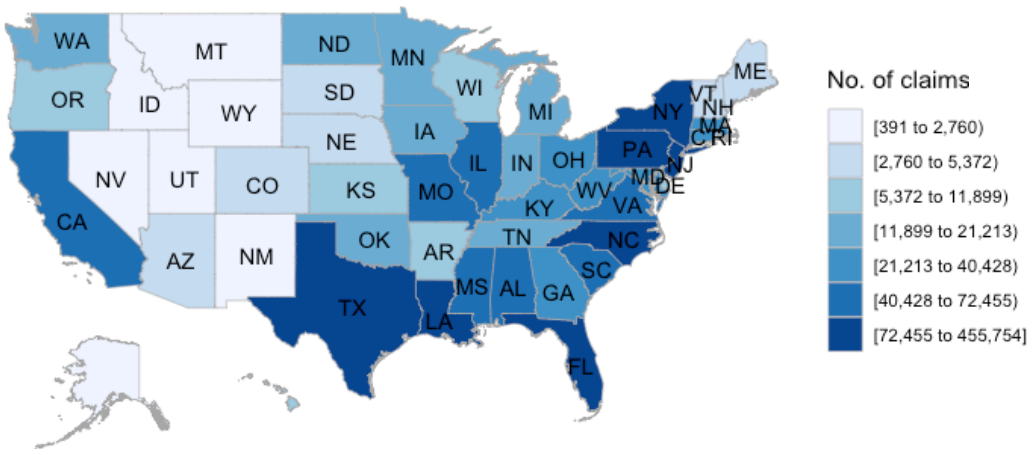


Figure 5.8 The claims values of all states split into 7 buckets (1970-2018).

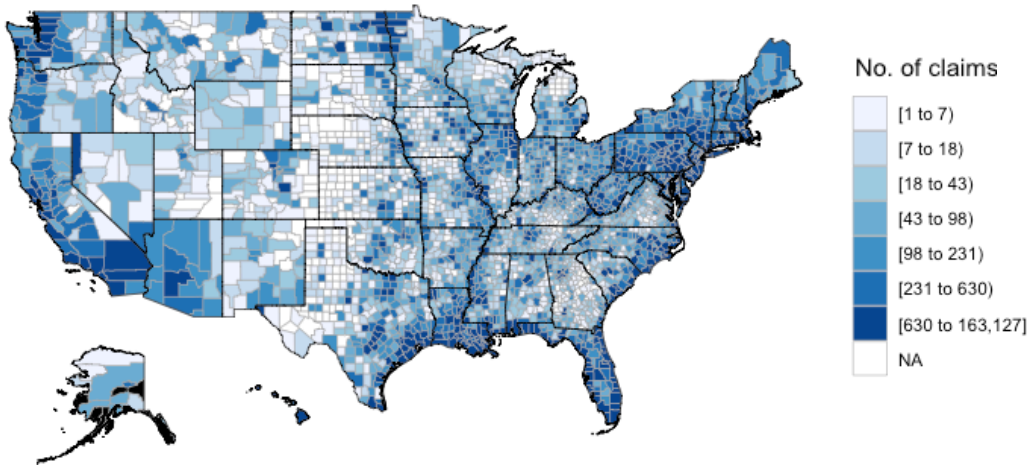


Figure 5.9 The aggregated number of claims per county (1970-2018).

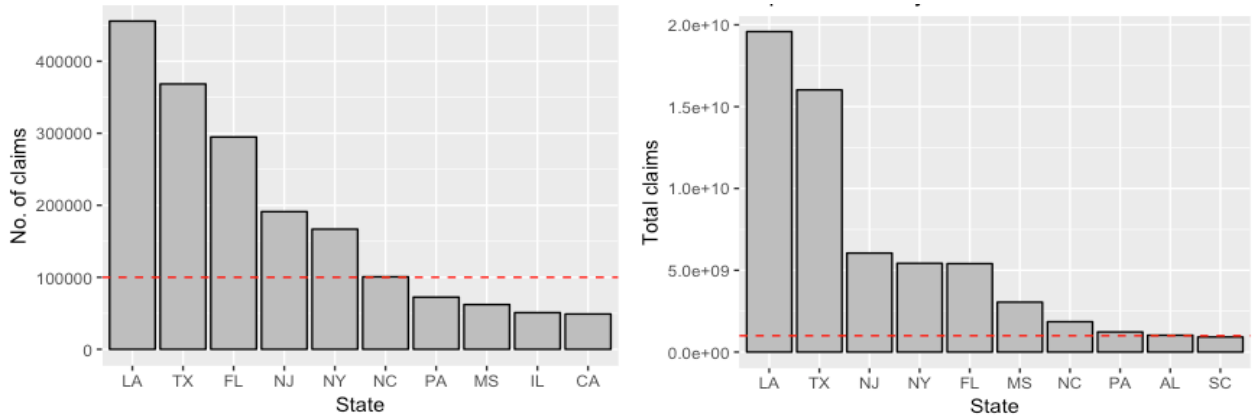


Figure 5.10 The above bar plots show the number of claims (left) and the claim amounts (right) of the most affected states. 6 states experienced more than 100,000 claims (red dashed line), with North Carolina barely hitting that number. It is not surprising that the top 5 states with the most number of claims are also the top 5 states with the largest claim amounts. What might be more surprising is how much more money was claimed in LA and TX compared to the other states (1970-2018).

Figure 5.11 The aggregated number of claims per state regarding the Gulf states (1970-2018).

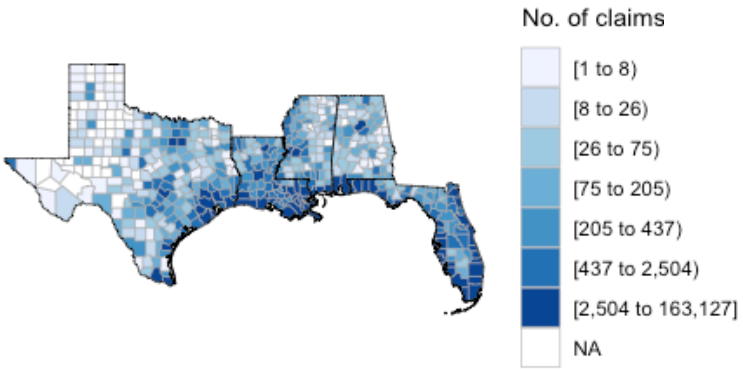


Figure 5.12 The aggregated number of claims per county regarding the Gulf states (1970-2018).

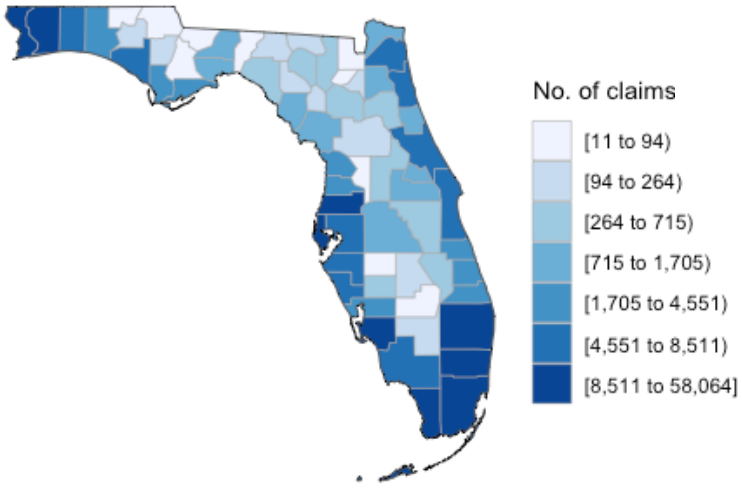


Figure 5.13 The aggregated number of claims per county regarding the Florida state (1970-2018).

6. Evaluating the clustering mechanisms of extremes on flood insurance practices with computational tools

6.1. Impacts of clustering mechanisms on collective risk

For each one of the selected 360 gauge locations and for each one of the 4 selected thresholds (90%, 95%, 98%, 99%), the annual collective risk is calculated for the observed as well as the shuffled time series. The results from this process are quite impressive as, in many gauge locations, the divergence between the observed and the shuffled (independent) time series on the diagram of the Empirical Cumulative Distribution Function (ECDF) of collective risk is evident in many gauge locations. These results are a clear indication of the existence of the clustering of extremes in terms of collective risk, which reflects directly on the claim amounts. The results regarding the gauge location at the Sf Trinity River Below Hyampom in California follow (Figures 6.1-6.3).

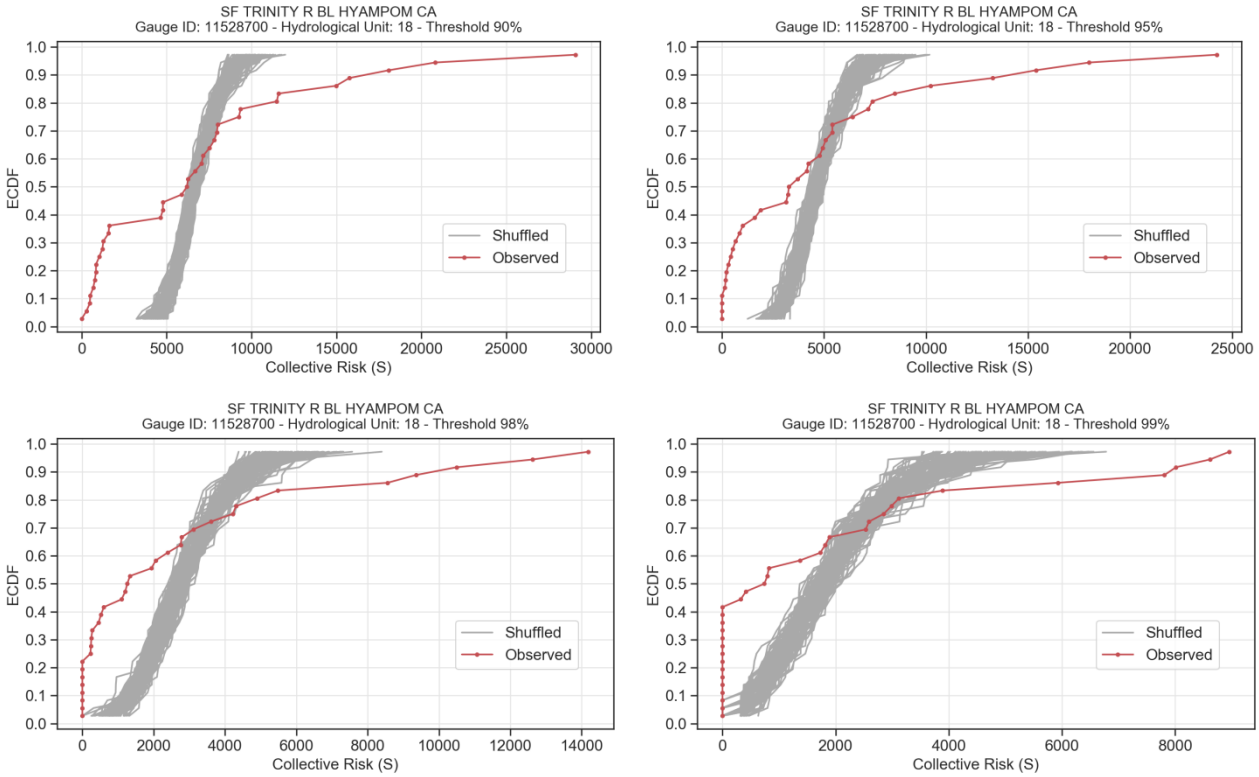


Figure 6.1 Collective risk's ECDF diagrams in linear scale (Gauge location ID: 11528700).

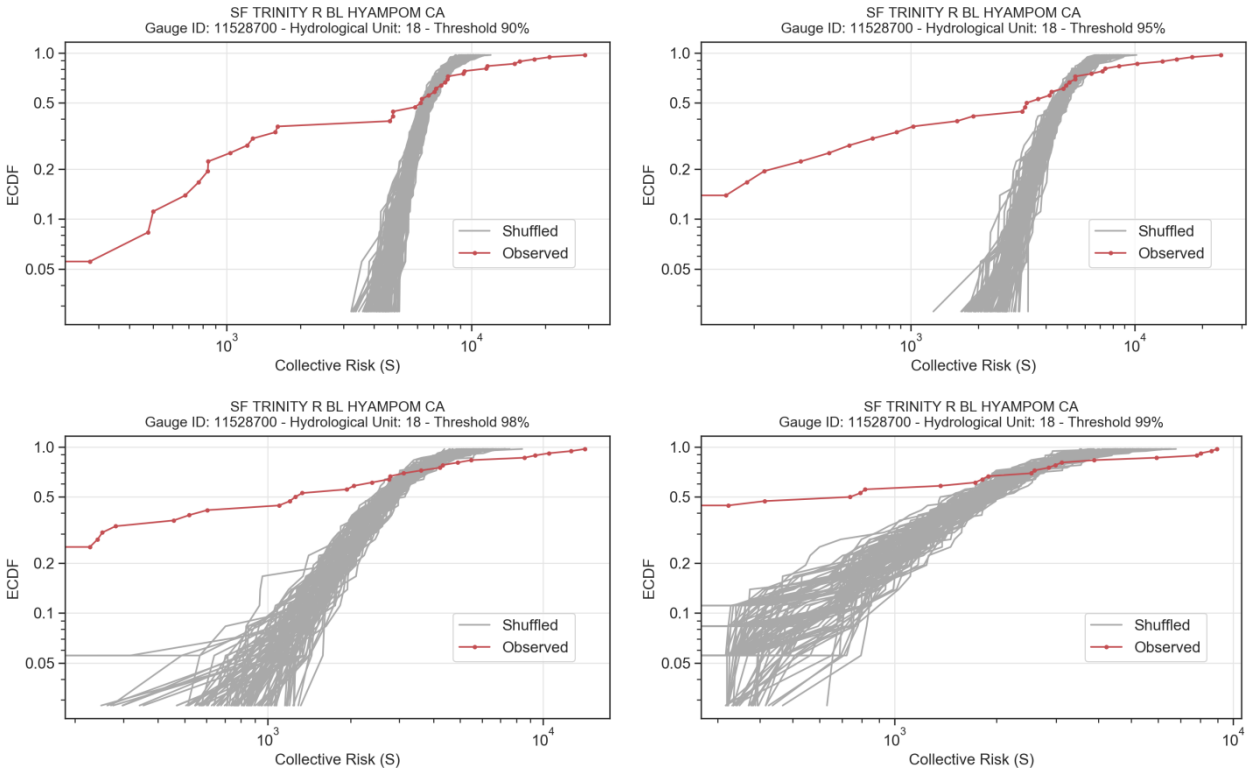


Figure 6.2 Collective risk's ECDF diagrams in logarithmic scale (Gauge location ID: 11528700).

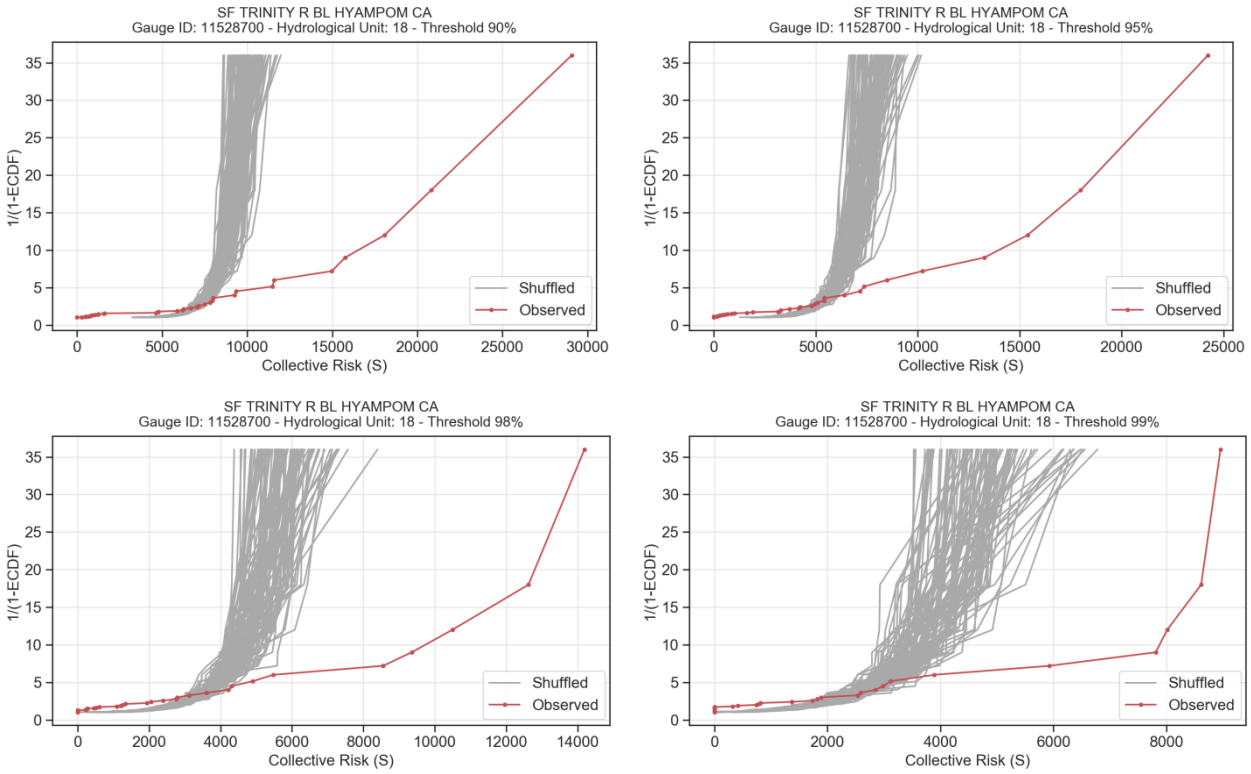


Figure 6.3 Collective risk's return period (1/(1-ECDF)) diagrams in linear scale (Gauge location ID: 11528700).

6.2. Impacts of clustering mechanisms on return intervals

For each one of the selected 360 gauge locations and for each one of the 4 thresholds, the return intervals of the over-threshold events are calculated for the observed as well as the shuffled time series. The return interval between two over-threshold events is a significant parameter for the insurance companies, as higher or lower values in short temporal periods can even affect their reserves, provoking potential financial instability. Once again, the divergence between the observed and the shuffled (independent) time series on the Empirical Cumulative Distribution Function (ECDF) for the return intervals of the over-threshold events diagram is evident in many gauge locations. The results regarding the Sf Trinity River Below Hyampom in California follow (Figures 6.4-6.5).

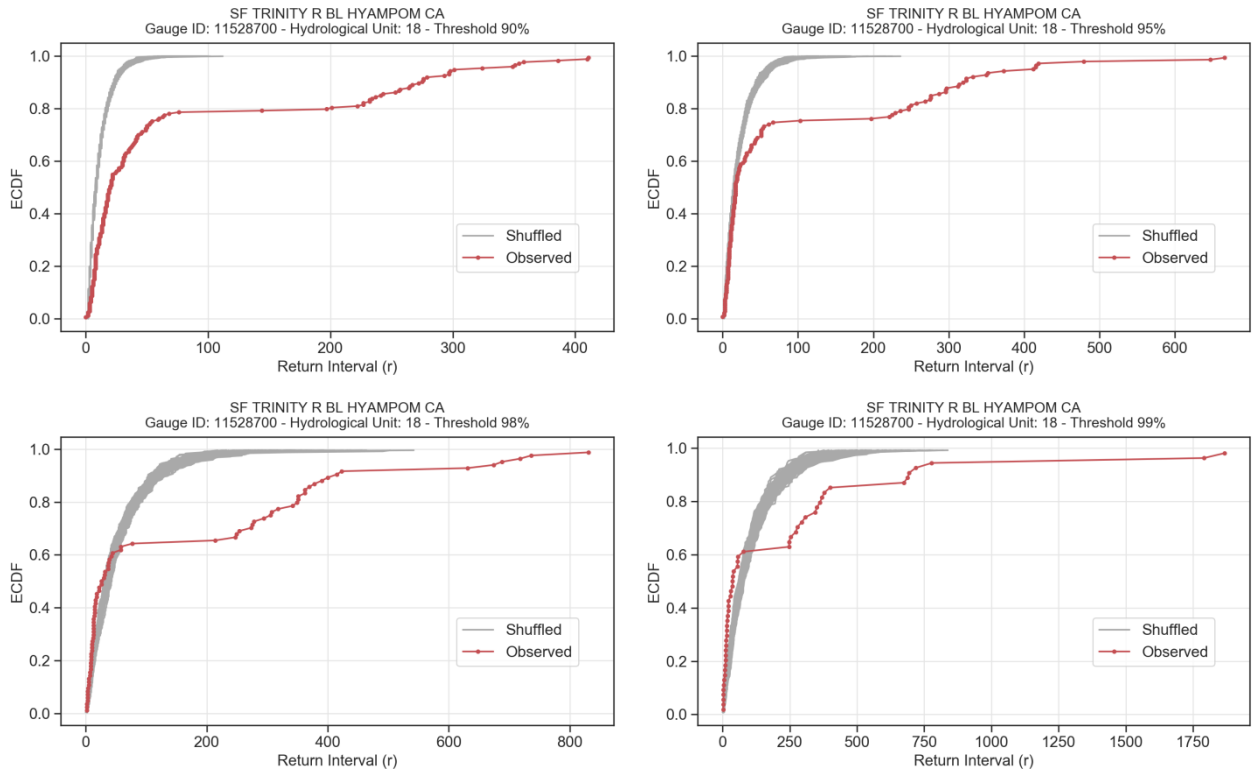


Figure 6.4 Return interval's ECDF diagrams in linear scale (Gauge location ID: 11528700).

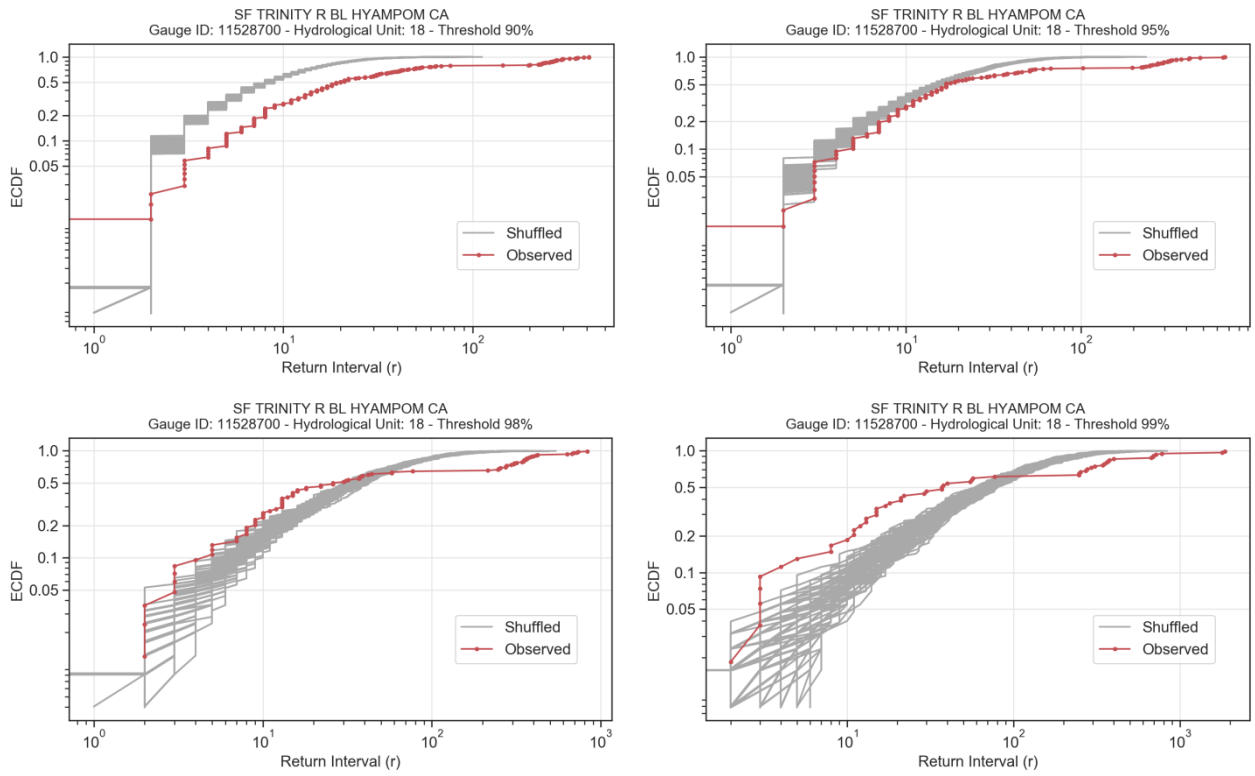


Figure 6.5 Return interval's ECDF diagrams in logarithmic scale (Gauge location ID: 11528700).

6.3. Impacts of clustering mechanisms on the duration of the over-threshold events

For each one of the selected 360 gauge locations and for each one of the 4 thresholds, the duration of the over-threshold events are calculated for the observed as well as the shuffled time series. The duration of the over-threshold events is a challenging parameter for the insurance companies, as higher or lower values can affect the severity of the event in terms of financial losses as well as the claim amounts. Once again, the divergence between the observed and the shuffled (independent) time series on the Empirical Cumulative Distribution Function (ECDF) for the duration of the over-threshold events diagram is evident in many gauge locations. The results regarding the Sf Trinity River Below Hyampom in California follow (Figures 6.6-6.7).

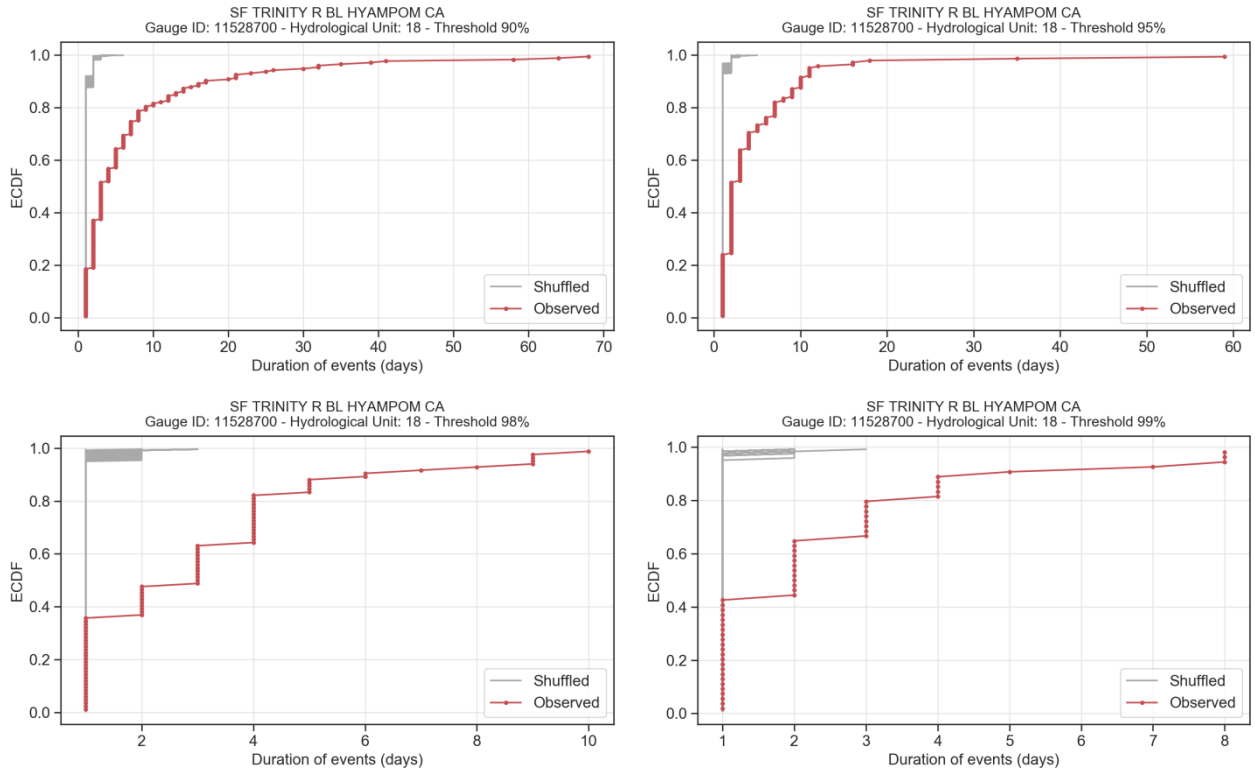


Figure 6.6 Events' duration ECDF diagrams in linear scale (Gauge location ID: 11528700).

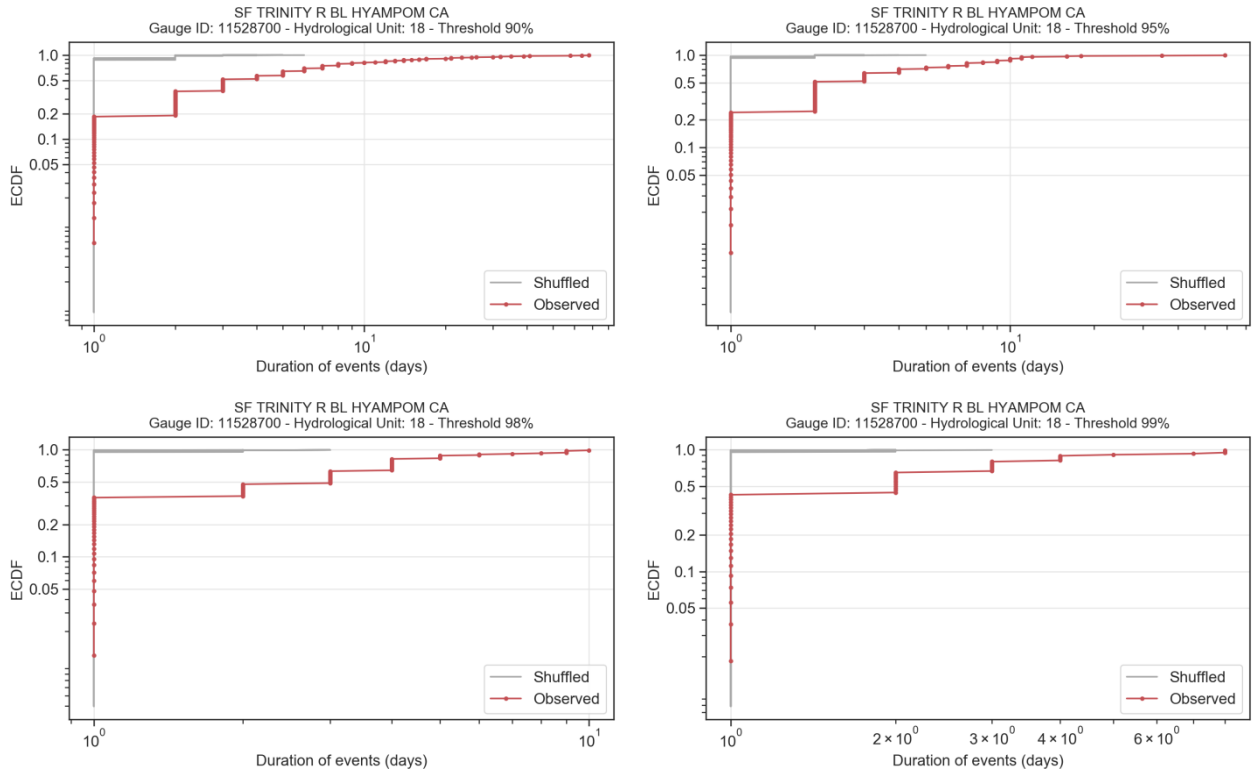


Figure 6.7 Events' duration ECDF diagrams in logarithmic scale (Gauge location ID: 11528700).

6.4. Mean Climacogram of US-CAMELS, Monte Carlo simulations and modeling approaches related with HK behavior

Based on the mean *Climacogram* of the GHK process (Dimitriadis and Koutsoyiannis, 2018) regarding the 360 empirical streamflow time series of the US-CAMELS dataset, the Hurst coefficient was estimated 0.63, which indicates clearly a persistent behavior (Figure 6.8-6.9). The effect of this dependence structure is tracked on the behaviors of POT flows at the annual scale and the estimation of the collective risk proxy. Subsequently, the behavior of daily streamflow in our dataset is found to be consistent with HK dynamics characterized by moderate *H* parameters (in the range 0.6-0.7), through Monte Carlo simulations.

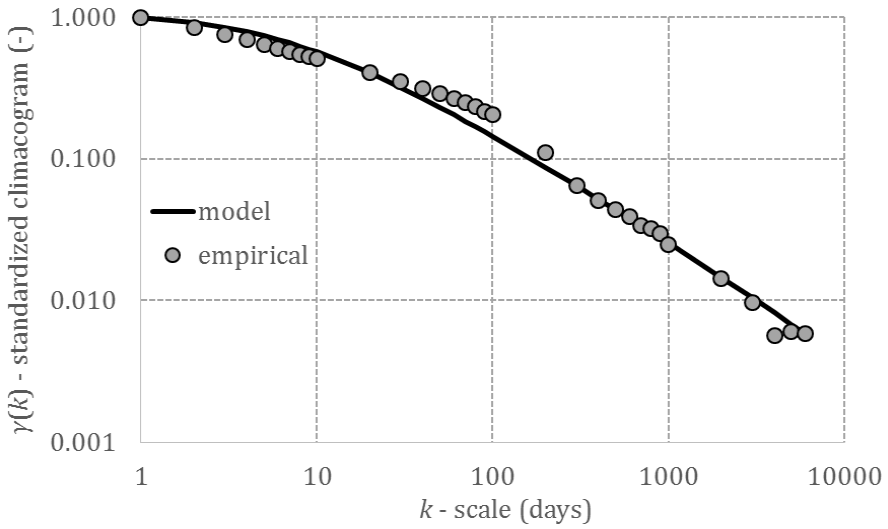


Figure 6.8 The mean *Climacogram* of the 360 selected gauge locations of the US-CAMELS dataset.

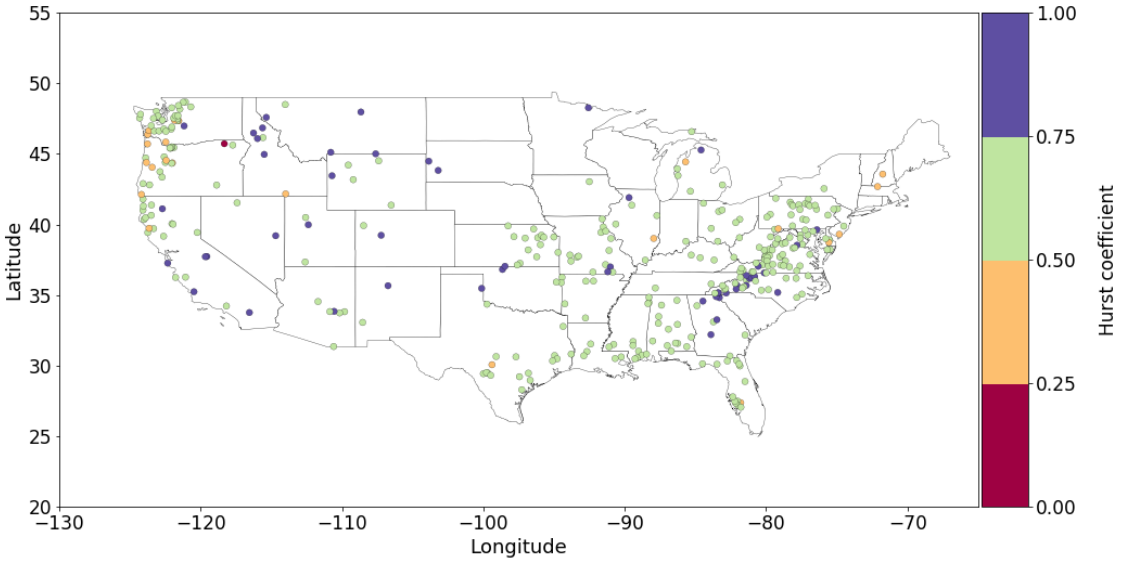


Figure 6.9 Hurst coefficient *H* of each one of the 360 selected gauge locations of the US-CAMELS dataset (see appendix, Table A-2).

Modelling and simulation approaches of extreme flows are developed as a step towards the deeper understanding of the relationship between the stochastic patterns of flood occurrence and proxies of insurance claims, paving the way for a more efficient use of the available streamflow records. In order to develop the stochastic simulation of a series with generalized long-range dependence, the SMA-GHK model is applied (Dimitriadis and Koutsoyiannis, 2018) by preserving explicitly the first four central moments of the sample series. The algorithm to produce synthetic time series from the data of the observed (empirical) one with the SMA scheme, created by P. Dimitriadis (2018), required as input the following: mean (S_m), variance (S_v), skewness and kurtosis coefficients (S_s and S_k), Hurst parameter of the GHK model (H), scale parameter (q), length of synthetic series (N).

The *Climacogram* (Figure 6.10) was formulated and the SMA-GHK modelling simulations were developed regarding the USGS 07071500 gauge location, Eleven Point river near Bardley, State of Missouri, USA, with parameters H = 0.81 and q = 1.00 days.

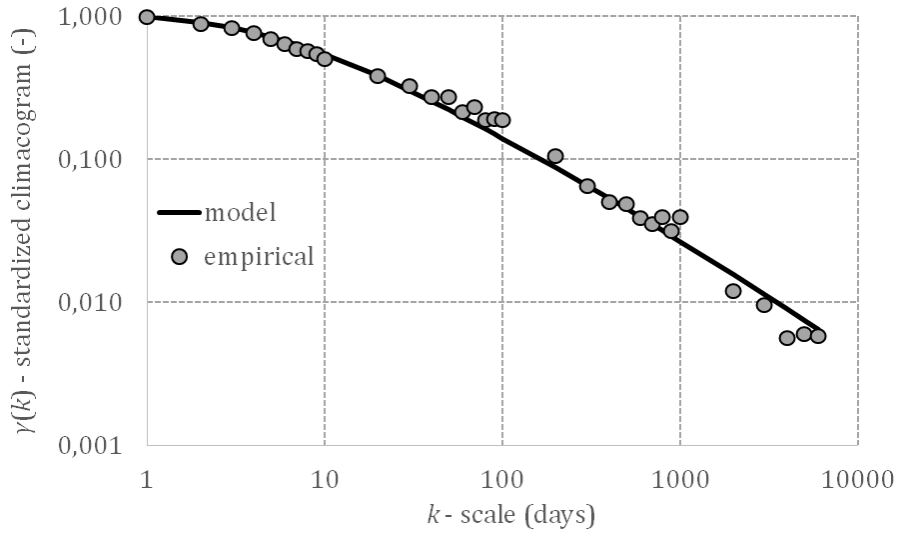


Figure 6.10 The *Climacogram* of the gauge location with ID: 07071500 (H = 0.81, q = 1.00 d).

Moreover, 1000 synthetic time series through Monte Carlo simulations of the USGS 07071500 gauge location were developed; the diagrams of the empirical cumulative distribution function (ECDF) of collective risk for the four thresholds are extracted (Figure 6.11). The ECDF curve of the observed collective risk proxy (Figure 6.11) is contained in the Monte Carlo prediction limits by the GHK model, preserving the HK dynamics and the 4 four moments. In contrary, shuffled (randomized) curves have a different behavior, especially in the tails of the distribution.

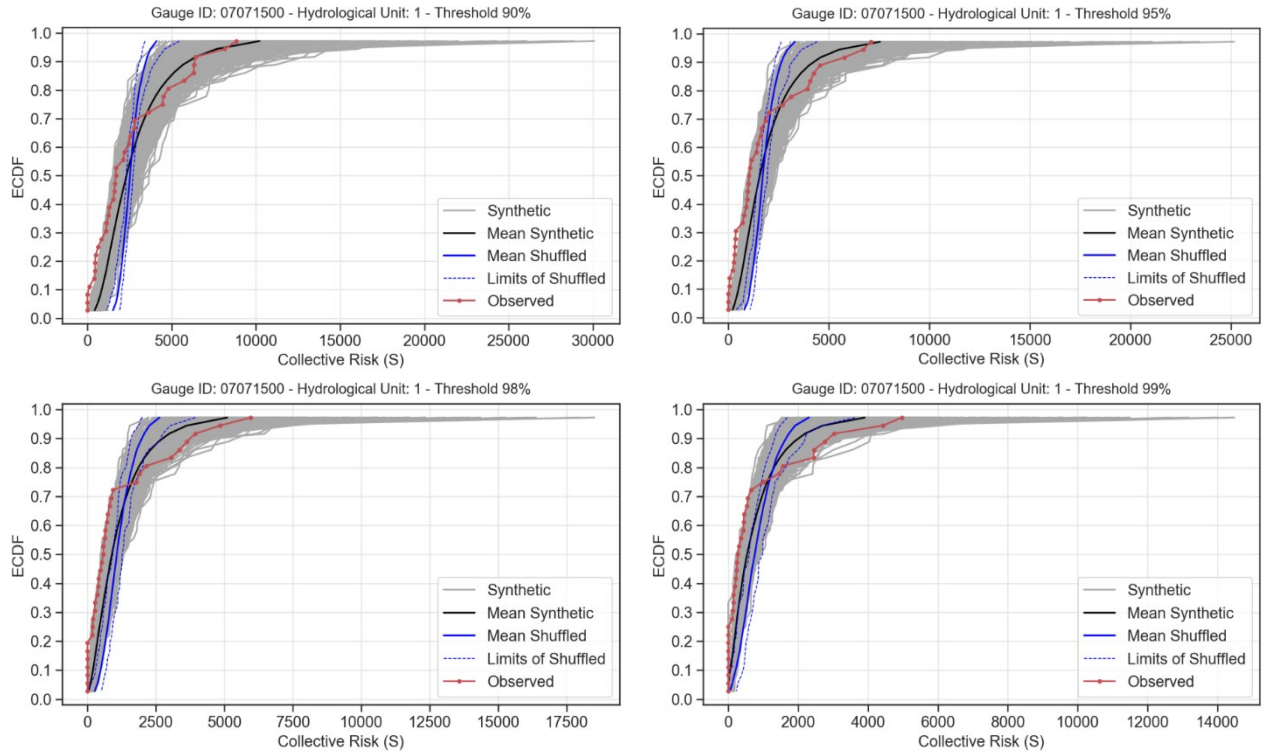


Figure 6.11 Collective risk's ECDF diagrams of observed, shuffled and synthetic time series (ID: 07071500).

6.5. Impacts of clustering mechanisms on correlation between *Average* Y_i and Number of over-threshold events *N*

A common assumption in the computation of collective risk is the independence between *Average* Y_i and Number of over-threshold events *N*. Here, losses *Yi* denote the flows exceeding the selected threshold and *N* is the number of such exceedances (number of events) over 365-day time windows. The relationship between *Average* Y_i and *N* is emerged by the Spearman, Pearson and Kendall correlation coefficient between *N* and the average value of the over-threshold flow intensities

$$\frac{1}{N}\sum Y_i = \frac{S}{N} = Average\ Y_i \qquad (6.1)$$

Insurance companies' concern about this correlation factor is noteworthy, due to the fact that they try to investigate the dependence between the annual number of extreme events and the provoked *Average* Y_i, which is a proxy of the average claim amounts per over-threshold event on a specific region. Introducing this parameter, Pearson correlation coefficient, Spearman correlation coefficient and Kendall correlation coefficient for the 4 selected thresholds are investigated.

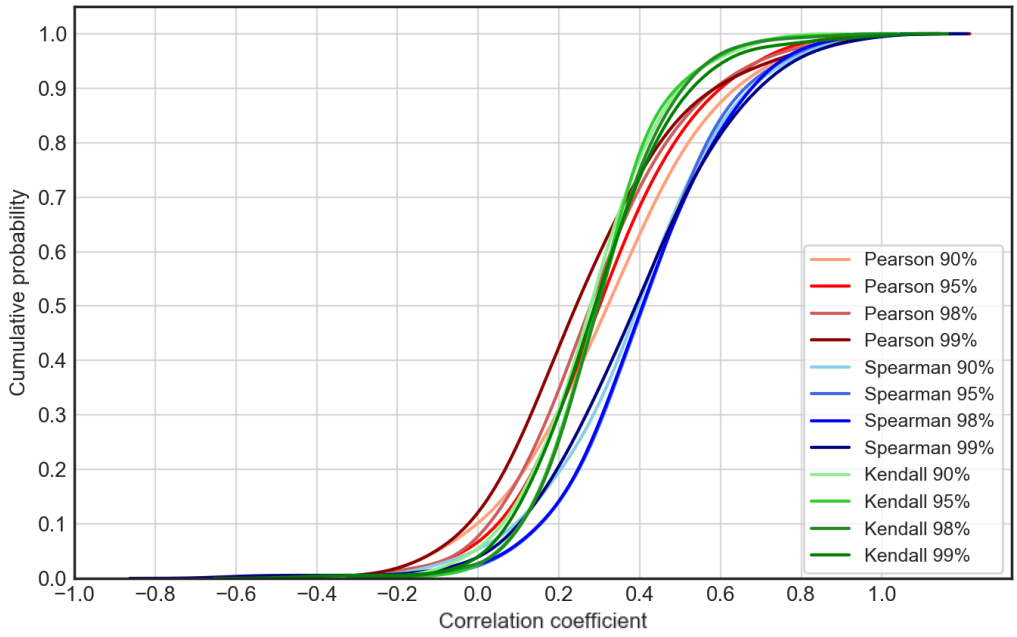


Figure 6.12 Cumulative histogram curves of the Pearson, Spearman and Kendall correlation coefficient between *Average* Y_i and number of over-threshold events N for the 360 selected gauge locations and for all thresholds.

Figure 6.12 shows cumulative histogram curves of the Pearson, Spearman and Kendall correlation coefficient between *Average* Y_i and number of over-threshold events N for the 360 selected gauge locations and for all thresholds. This study evaluates the Spearman correlation coefficient, as it is considered as the most suitable tool for the analysis of extremes. Instinctively, someone would expect that years that are more active in terms of Number of Events N tend to exhibit extreme events also in terms of *Average* Y_i magnitude. Indeed, our study shows that this assumption holds true in most cases, yet there are exceptions shown in the maps in Figures 6.13-6.16 suggesting that it may not be universally applicable. The following Figures (6.13-6.16) present the Spearman correlation coefficient between *Average* Y_i and Number of over-threshold events N for all the gauge locations for the selected thresholds.

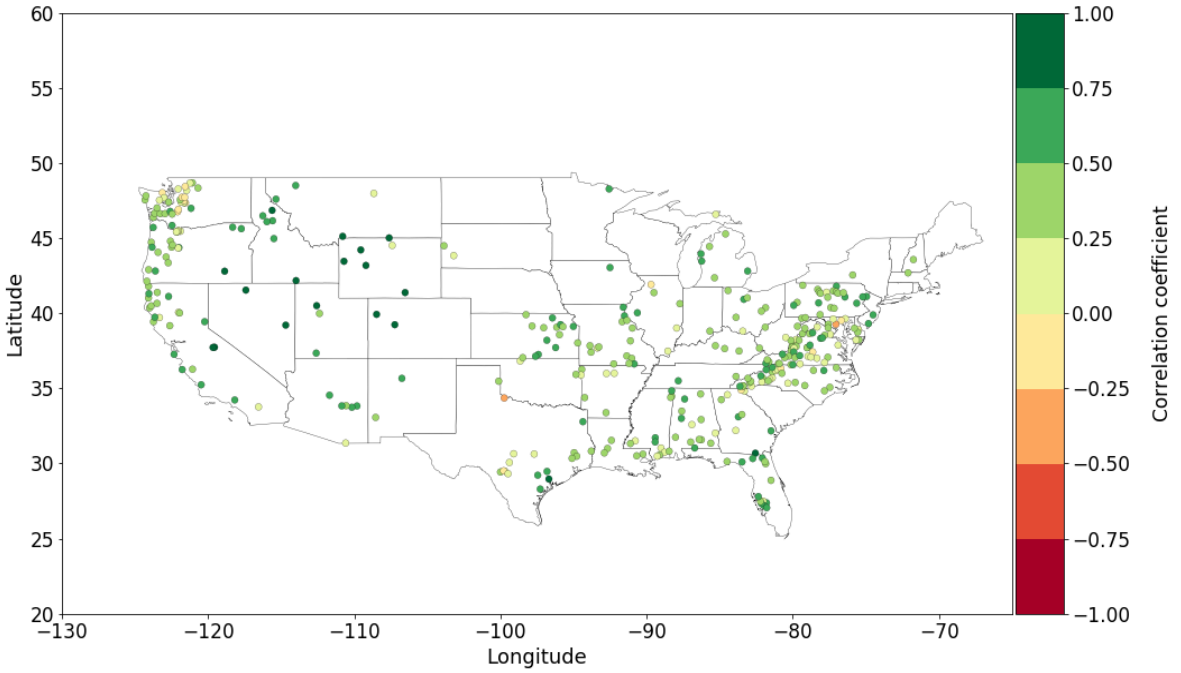


Figure 6.13 Spearman correlation coefficient between *Average* Y_i and Number of over-threshold events N for all gauge locations (threshold 90%).

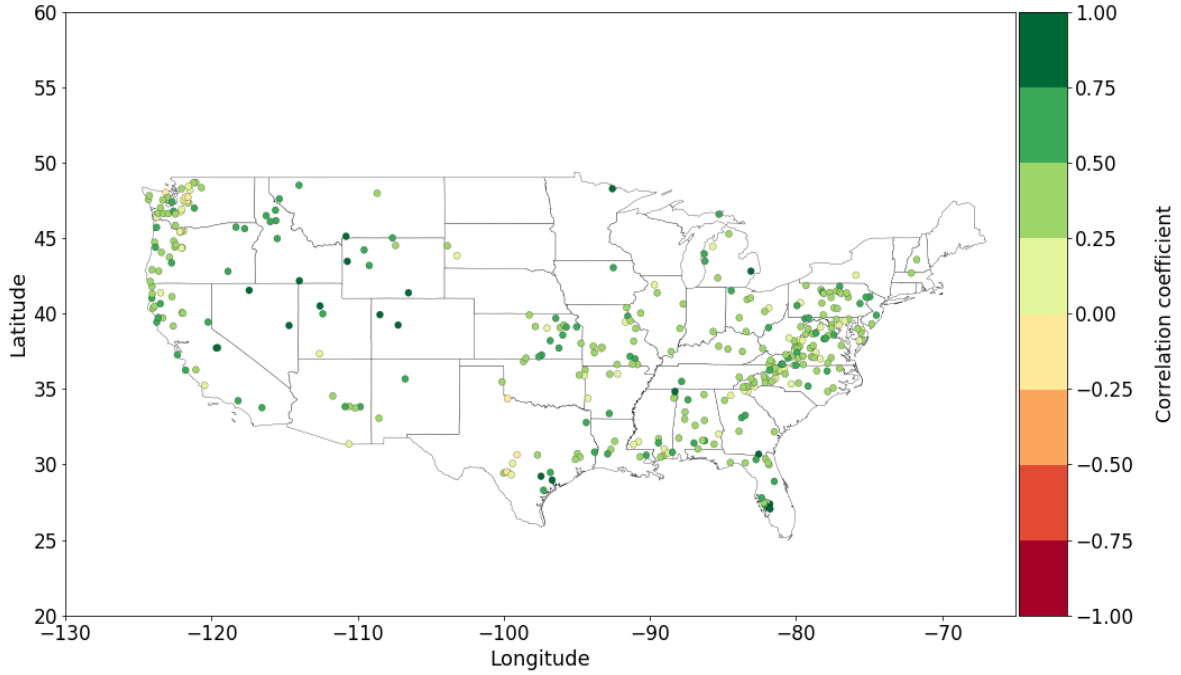


Figure 6.14 Spearman correlation coefficient between *Average* Y_i and Number of over-threshold events N for all gauge locations (threshold 95%).

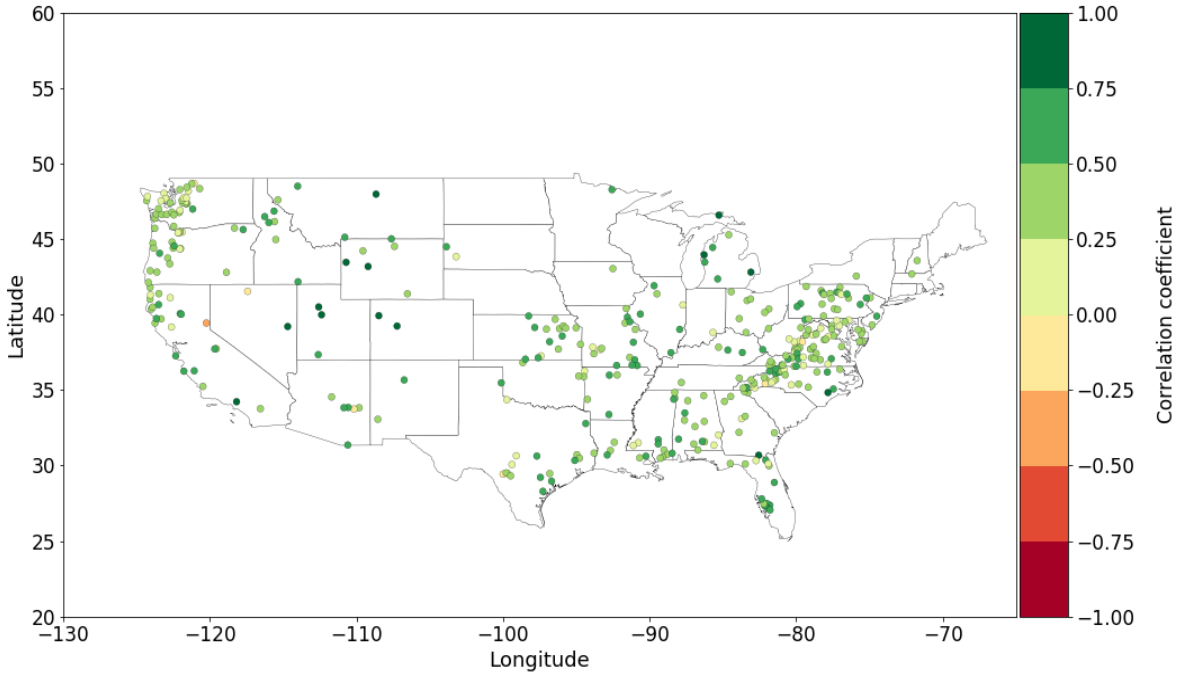


Figure 6.15 Spearman correlation coefficient between *Average* Y_i and Number of over-threshold events N for all gauge locations (threshold 98%).

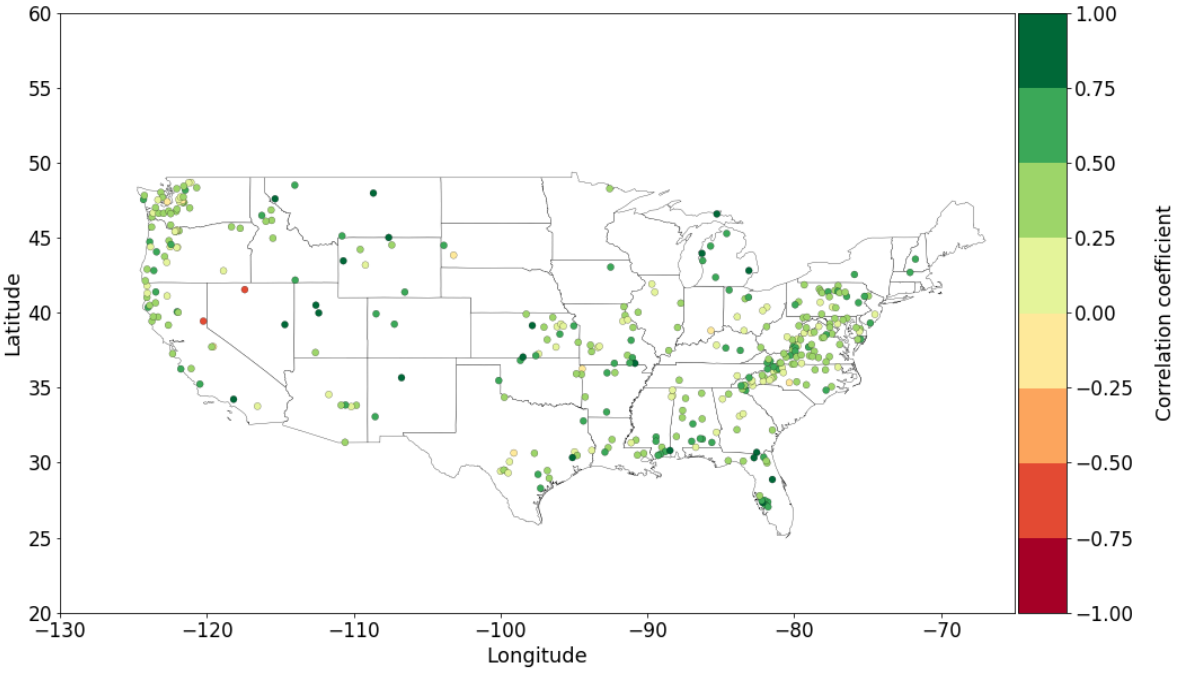


Figure 6.16 Spearman correlation coefficient between *Average* Y_i and Number of over-threshold events N for all gauge locations (threshold 99%).

This depiction offers a spatial categorization of areas with high Spearman correlation coefficient between the *Average* Y_i and the Number of over-threshold events N, in contrast with the ones where the correlation coefficient is noticeably lower. In other words, this spatial categorization indicates the regions that are subjected to numerous claim amounts in case of a year that an extreme number of over-threshold events occur. Moreover, it is shown that threshold selection influence slightly the Spearman, Pearson and Kendall correlation coefficient.

In addition, for each one of the 360 gauge locations and for all selected thresholds, the Spearman correlation coefficient between the *Average* Y_i and the Number of over-threshold events N was calculated for the observed as well as the shuffled (independent) time series in order to evaluate the clustering mechanisms on this correlation parameter. The following box plots show that clustering mechanisms that are prevailing over the observed data introduce significant correlation between the N and *Average* Y_i in many gauge locations.

The conclusions of this investigation are quite impressive once again, as the divergence of the correlation coefficient between the observed and the shuffled ones in many gauge locations is profound. In more detail, the shuffled series tend to underestimate the correlation coefficient in comparison with the observed ones, which apparently introduce the impacts of clustering mechanisms. Ignorance of this behavior could lead insurance policy-makers on inaccurate conclusions which could potentially provoke financial impacts.

The results that are shown on the following Figures (6.17-6.20) present the above mentioned conclusions. The gauge locations of these figures are:

- Suwannee River AT US 441 AT Fargo, GA (ID: 02314500)
- Arroyo Seco NR Pasadena, CA (ID: 11098000)
- SF Trinity R BL Hyampom, CA (ID: 11528700)
- Cache Creek Near Jackson, WY (ID: 13018300)

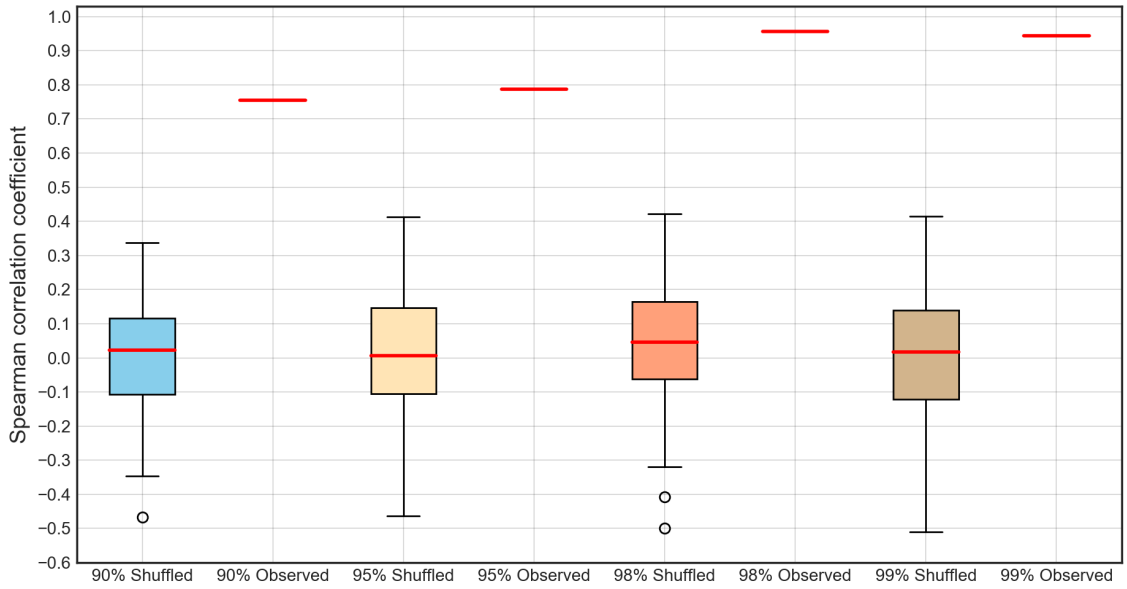


Figure 6.17 Box plot of Spearman correlation coefficient between *Average* Y_i and Number of over-threshold events N for the shuffled as well as the observed time series for all thresholds (Gauge ID: 02314500).

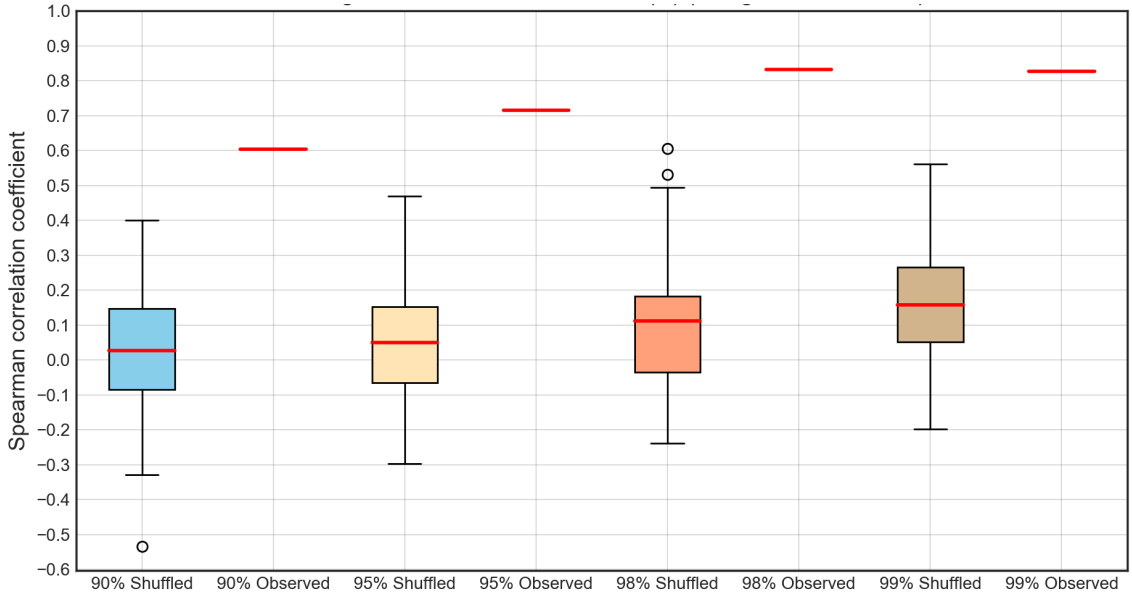


Figure 6.18 Box plot of Spearman correlation coefficient between *Average* Y_i and Number of over-threshold events N for the shuffled as well as the observed time series for all thresholds (Gauge ID: 11098000).

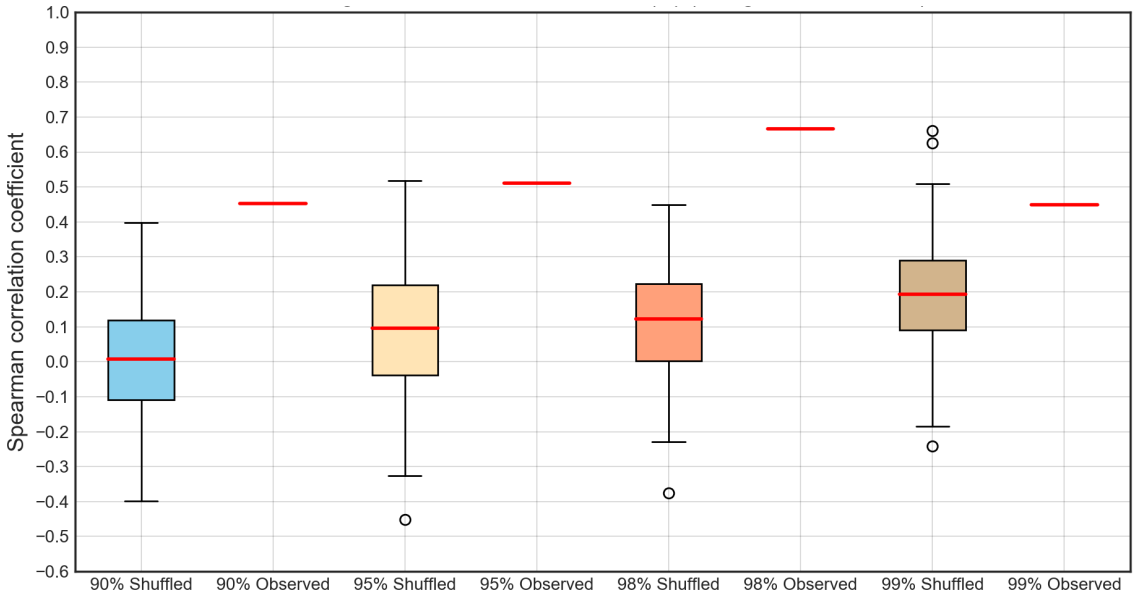


Figure 6.19 Box plot of Spearman correlation coefficient between *Average* Y_i and Number of over-threshold events N for the shuffled as well as the observed time series for all thresholds (Gauge ID: 11528700).

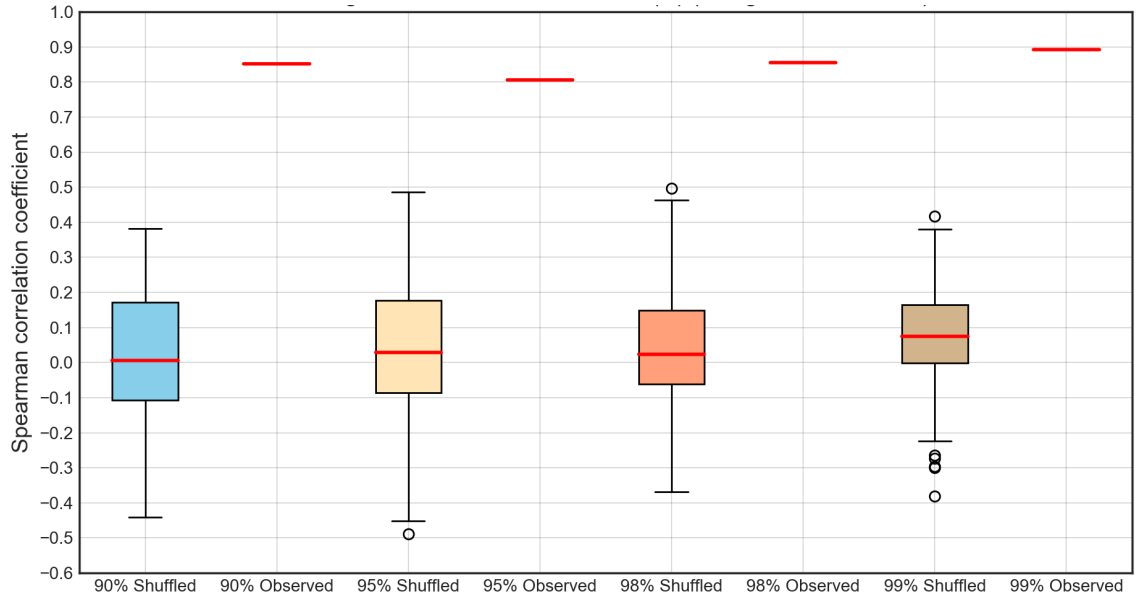


Figure 6.20 Box plot of Spearman correlation coefficient between *Average* Y_i and Number of over-threshold events N for the shuffled as well as the observed time series for all thresholds (Gauge ID: 13018300).

6.6. Validating the streamflow-based collective risk proxy method by FEMA's NFIP actual claims records

The annual collective risk proxy of the 360 selected gauge locations of the US-CAMELS dataset has already been computed, considering the 4 selected thresholds (90%, 95%, 98% and 99%) for the years 1980-2014. Moreover, the published FEMA claims records offers us the opportunity to investigate the validity of the developed method on a spatial basis by assessing the correlation between these claims records with the results of our study on streamflow POT events. The FEMA claims records were distributed spatially on the 21 Hydrological Units and the 50 States. In this respect, the Spearman correlation coefficient is evaluated between the annual collective risk for each one of the gauge locations and the aggregated claims records of the Hydrological Unit and the State that a specific gauge location belongs to. The cumulative histogram (Figure 6.21) curves of these Spearman correlation coefficients for the 360 gauge locations and for all the selected thresholds follow, showing that, in general, considering the aggregated claims of States tends to underestimate the correlation coefficient in contrast to the aggregated claims of the Hydrological Units.

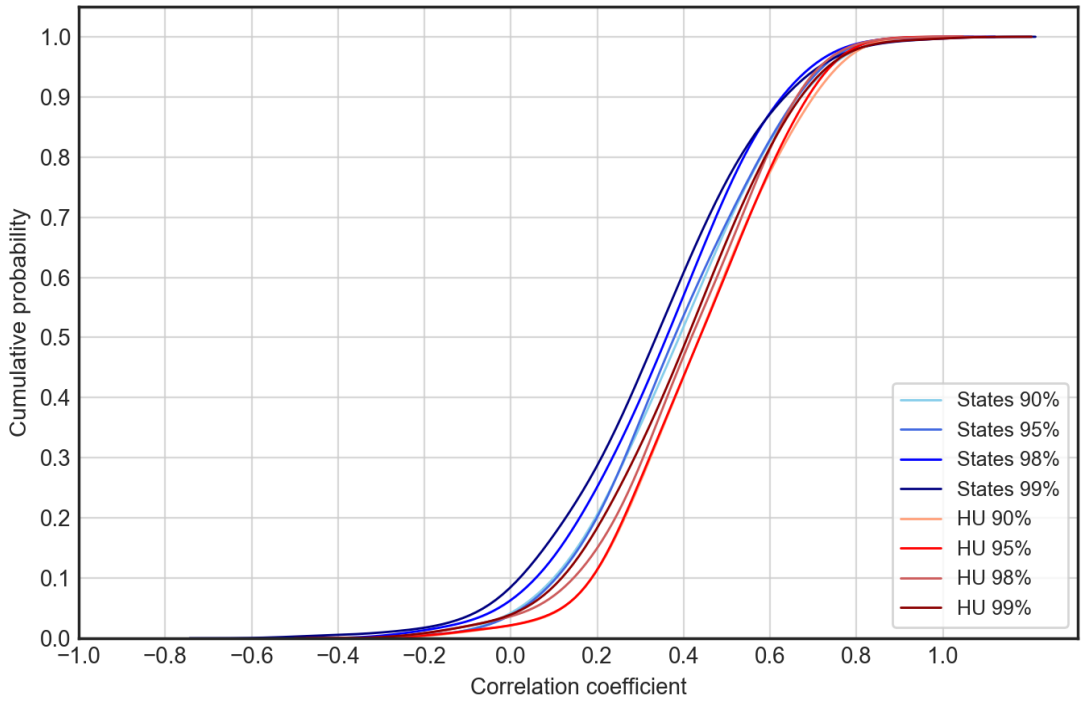


Figure 6.21 The cumulative histogram curves of the Spearman correlation coefficient between the annual collective risk of the 360 gauge locations and the States/Hydrological Units claims records that a specific gauge location belongs to.

Subsequently, the USA maps that show the Spearman correlation coefficient between the collective risk and the Hydrological Units' claims records for all the gauge locations and the selected

thresholds follow (Figures 6.22-6.25), highlighting the spatial distribution of the correlation coefficient and indicating the areas the latter is higher or lower. A spatial pattern is evident, showing that higher values of Spearman correlation coefficient emerge in West Coast, in comparison with the ones in East Coast, which are significantly lower.

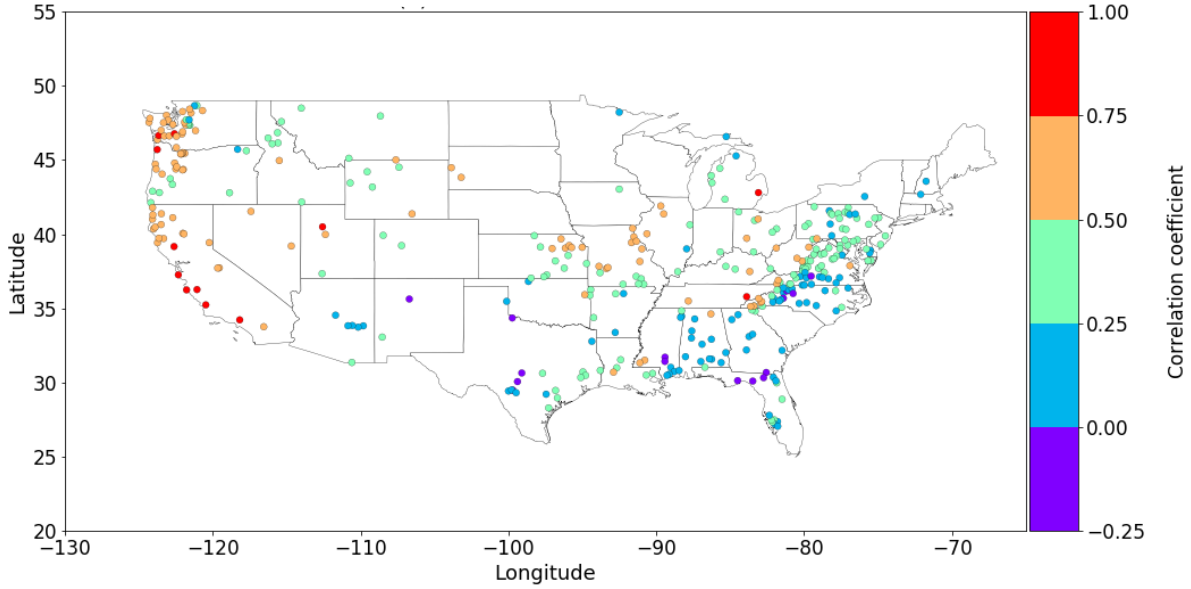


Figure 6.22 Spearman correlation coefficient for each one of the selected 360 gauge locations between annual collective risk and the claims records of the Hydrological Units that a specific gauge location belongs to (threshold 90%).

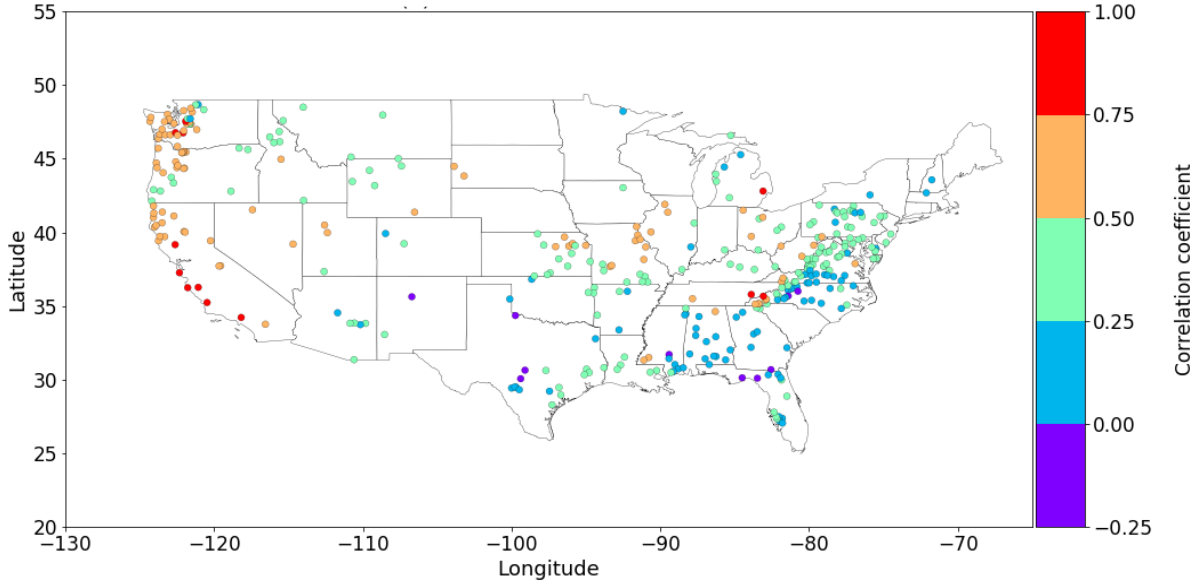


Figure 6.23 Spearman correlation coefficient for each one of the selected 360 gauge locations between annual collective risk and the claims records of the Hydrological Units that a specific gauge location belongs to (threshold 95%).

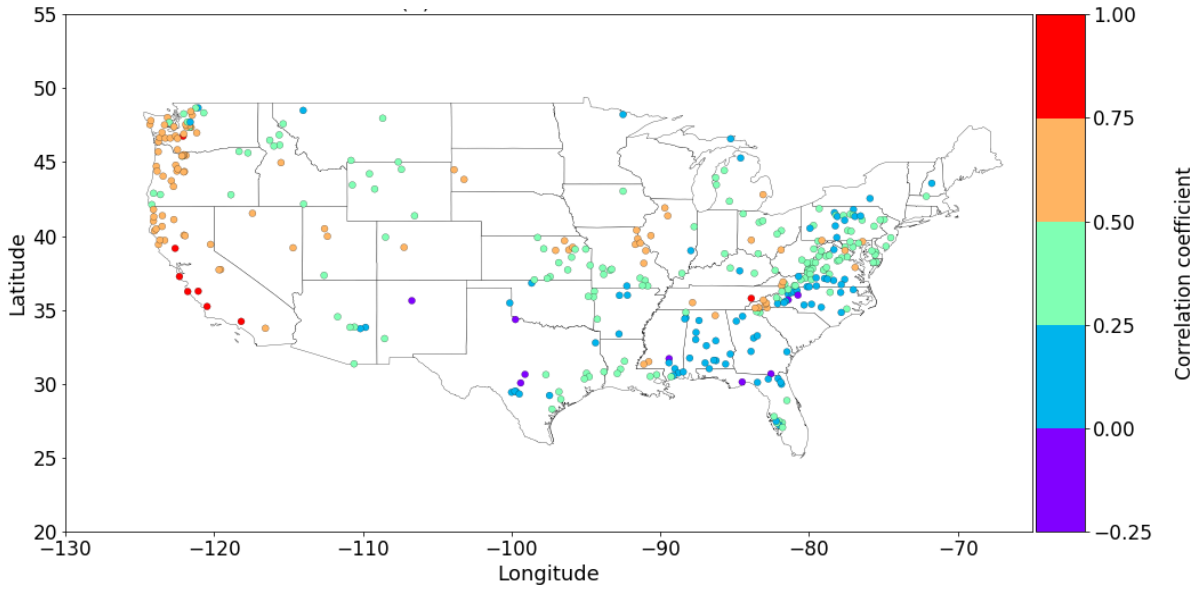


Figure 6.24 Spearman correlation coefficient for each one of the selected 360 gauge locations between annual collective risk and the claims records of the Hydrological Units that a specific gauge location belongs to (threshold 98%).

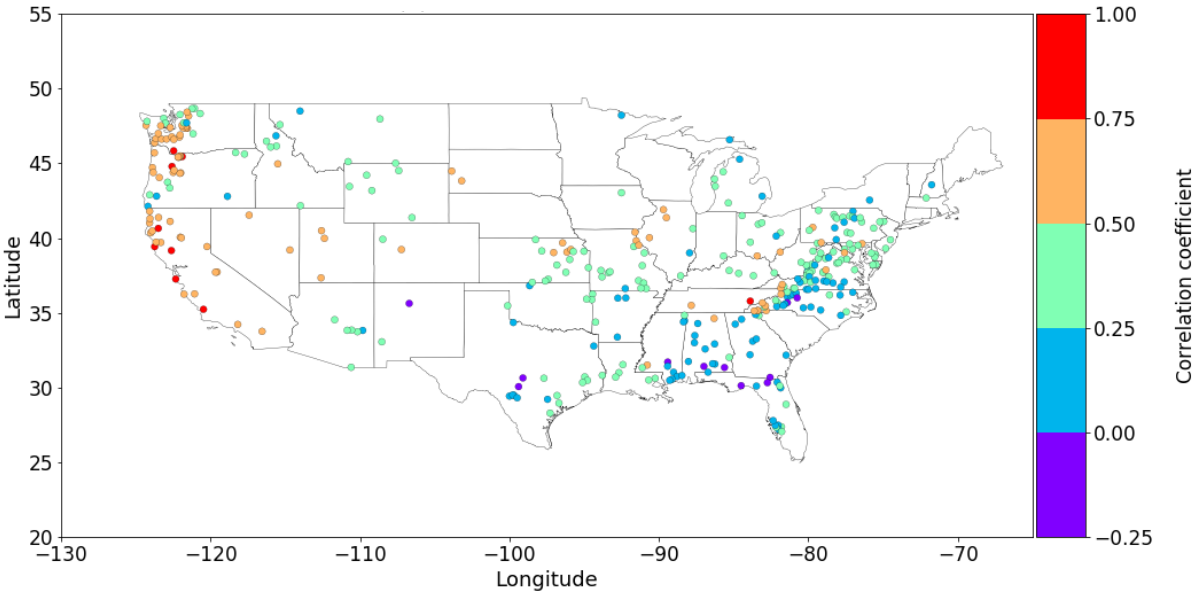


Figure 6.25 Spearman correlation coefficient for each one of the selected 360 gauge locations between annual collective risk and the claims records of the Hydrological Units that a specific gauge location belongs to (threshold 99%).

As collective risk is formulated in order to refer to river flooding in our study, these results show that this type of flooding is dominant in West Coast. In contrast, it is revealed that the source of flooding events that provoke claims in East Coast present a different and more complicated pattern, mainly due to the significant vulnerability of these areas on accelerated flooding events due to

hurricane hits, sea-level rise (SLR) and storm surge phenomena (Ezer and Atkinson, 2014). Moreover, these results are also explained by the fact that during relaxed (excited) North Atlantic Oscillation (NAO), Gulf (East) Coast is more susceptible to major hurricane strikes (Elsner et al., 2000).

Furthermore, for each one of the 360 gauge locations and for all the selected thresholds, the Spearman correlation coefficient between the collective risk and claims records of the Hydrological Unit that the gauge location belongs to was calculated for the observed as well as the shuffled (independent) time series in order to evaluate, once again, the effect of the clustering mechanisms on this correlation parameter. The following box plots in Figures 6.25-6.32 show that the dominant clustering mechanisms introduce significant correlation.

The conclusions of this investigation are quite impressive, as the divergence of the correlation coefficient between the observed and the shuffled series (independent) is evident in many gauge locations. In more detail, the shuffled series show no correlation between collective risk and actual claims records. Thus, the apparent existence of clustering mechanisms, which are indicated by this divergence between the observed and the shuffled, are considered to have significant financial impacts for the insurance companies, caused by inaccurate risk assessment. Underestimation of collective risk in cases where clustering mechanisms are not included in the calculations, could provoke unpredictable large values of collective risk (claim amounts) in case of an extreme flood event, stressing a company's reserves. Figures (6.26-6.33) present the gauge locations:

- Nehalem River Near Forr, OR (ID: 14301000)
- Sauk River Near Sauk, WA (ID: 12189500)
- Satsop River Near Satsop, WA (ID: 12035000)
- Lopez C NR Arroyo Grande, CA (ID: 11141280)
- Big Bureau Creek at Princeton, IL (ID: 05556500)
- Paint Rock River Near Woodville, AL(ID: 03574500)
- Shade River near Chester, OH (ID: 03159540)
- Piscataway Creek Near Tappahannock, VA (ID: 01669000)

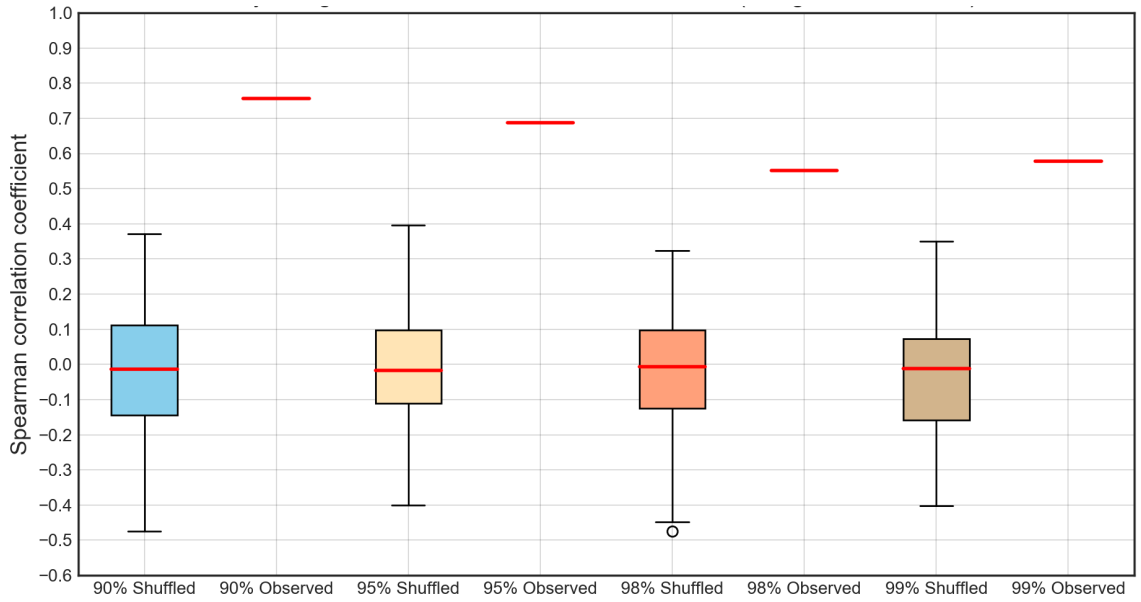


Figure 6.26 Box plot of Spearman correlation coefficient between collective risk and the Hydrological Unit's aggregated claims that the gauge location (ID: 14301000) belongs to, for all thresholds.

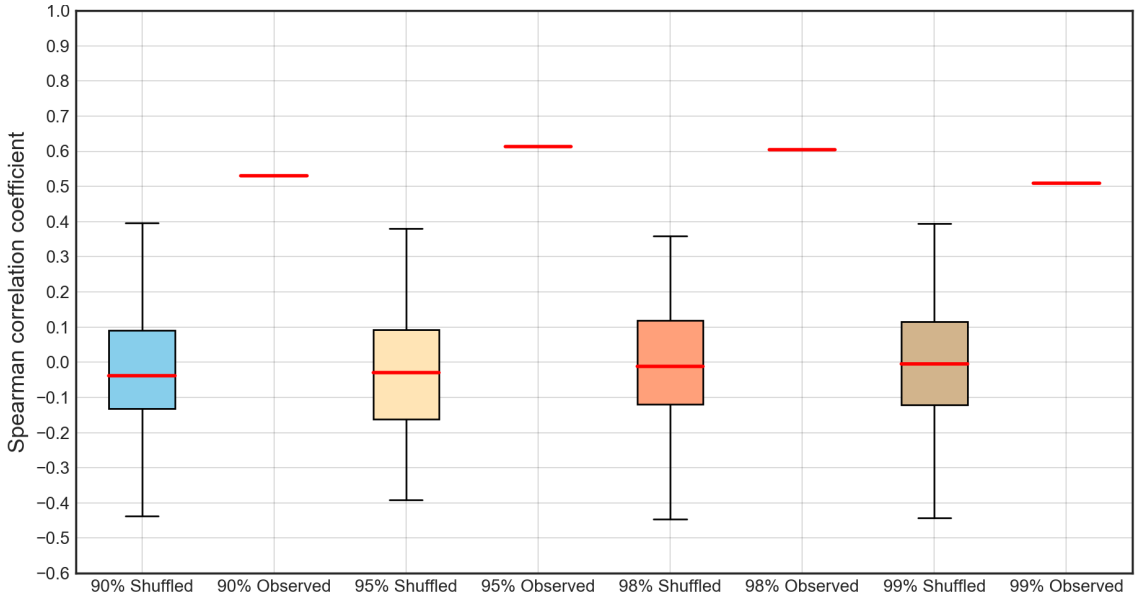


Figure 6.27 Box plot of Spearman correlation coefficient between collective risk and the Hydrological Unit's aggregated claims that the gauge location (ID: 12189500) belongs to, for all thresholds.

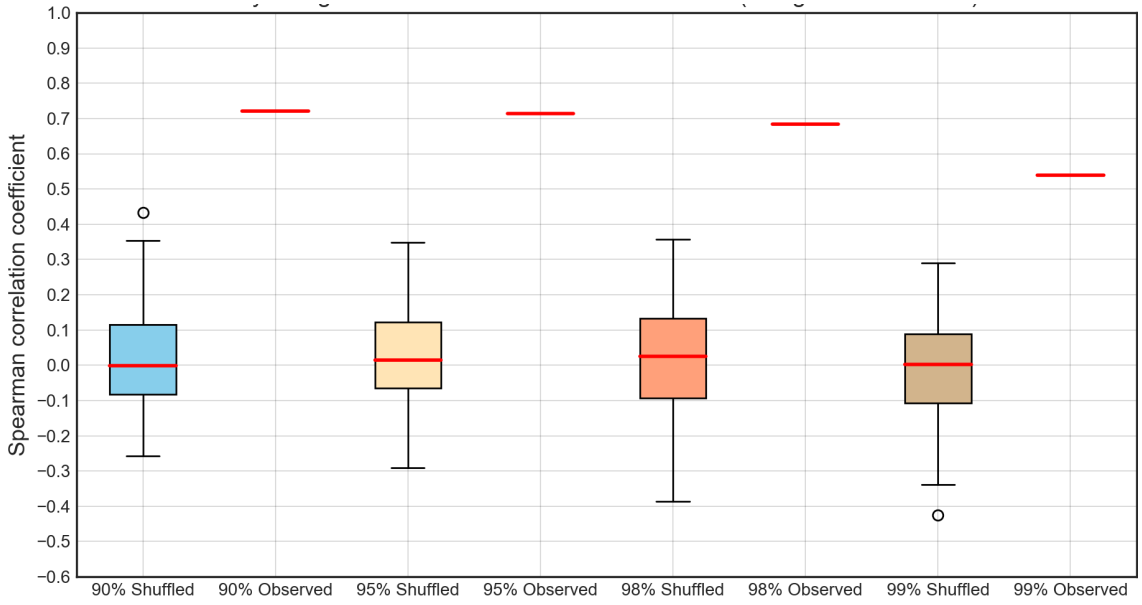


Figure 6.28 Box plot of Spearman correlation coefficient between collective risk and the Hydrological Unit's aggregated claims that the gauge location (ID: 12035000) belongs to, for all thresholds.

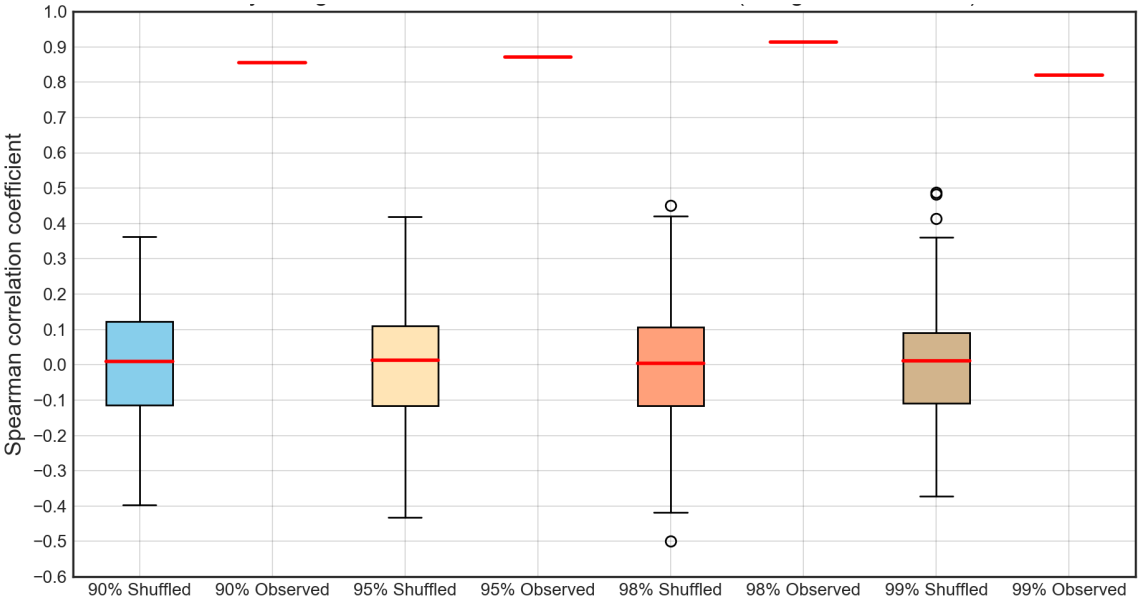


Figure 6.29 Box plot of Spearman correlation coefficient between collective risk and the Hydrological Unit's aggregated claims that the gauge location (ID: 11141280) belongs to, for all thresholds.

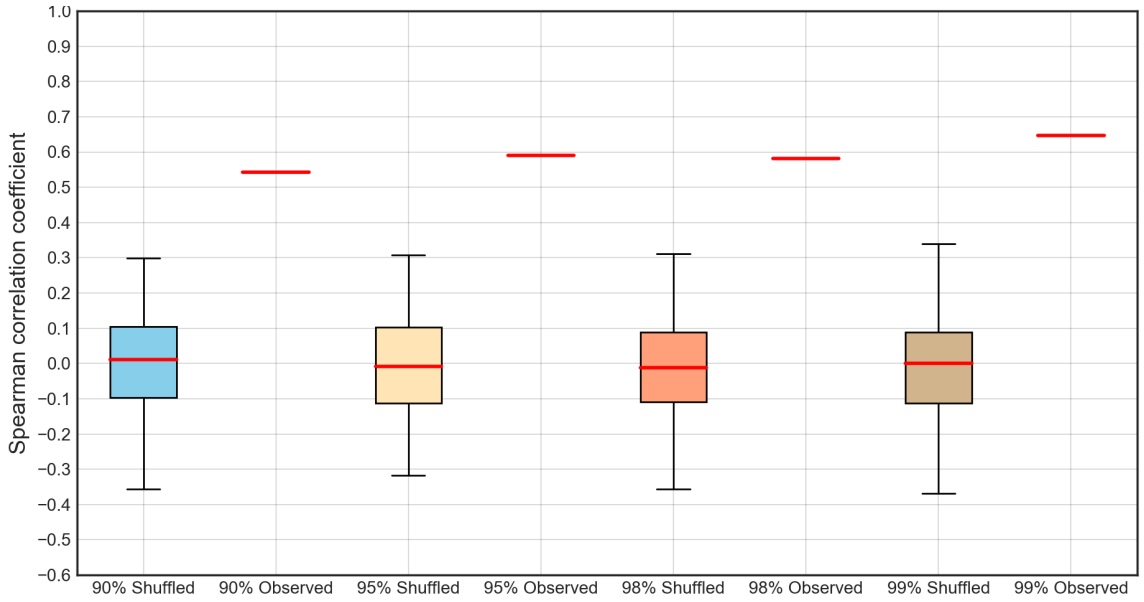


Figure 6.30 Box plot of Spearman correlation coefficient between collective risk and the Hydrological Unit's aggregated claims that the gauge location (ID: 05556500) belongs to, for all thresholds.

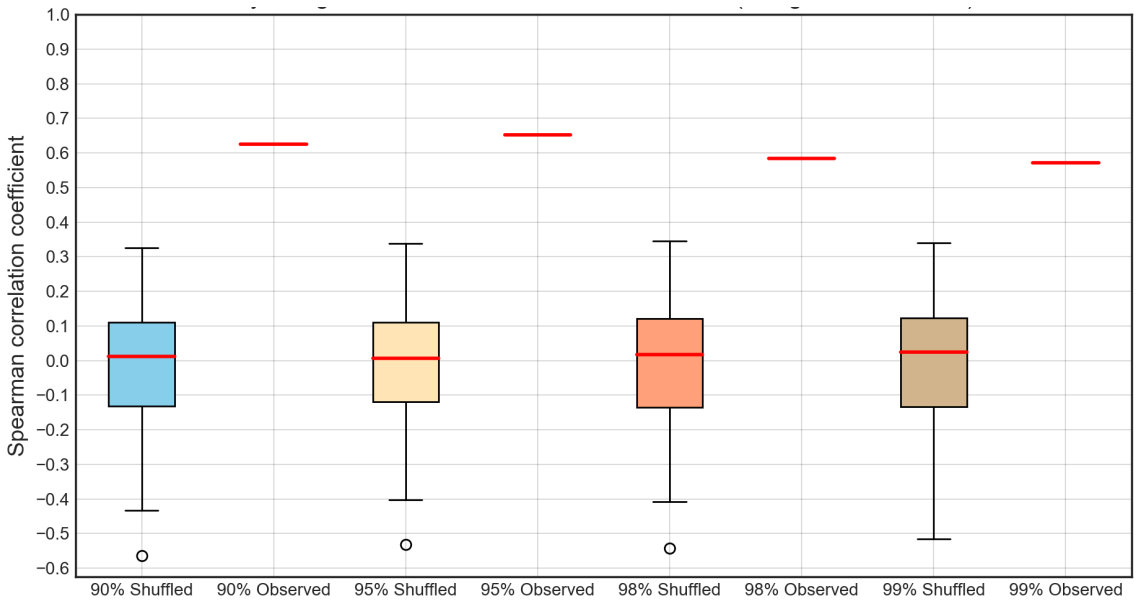


Figure 6.31 Box plot of Spearman correlation coefficient between collective risk and the Hydrological Unit's aggregated claims that the gauge location (ID: 03574500) belongs to, for all thresholds.

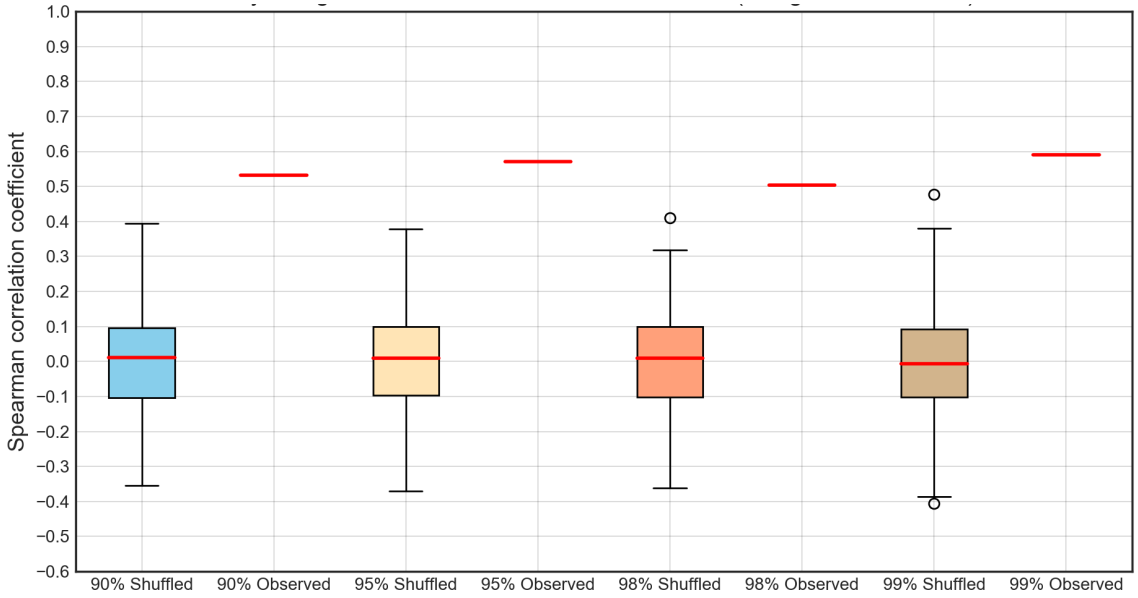


Figure 6.32 Box plot of Spearman correlation coefficient between collective risk and the Hydrological Unit's aggregated claims that the gauge location (ID: 03159540) belongs to, for all thresholds.

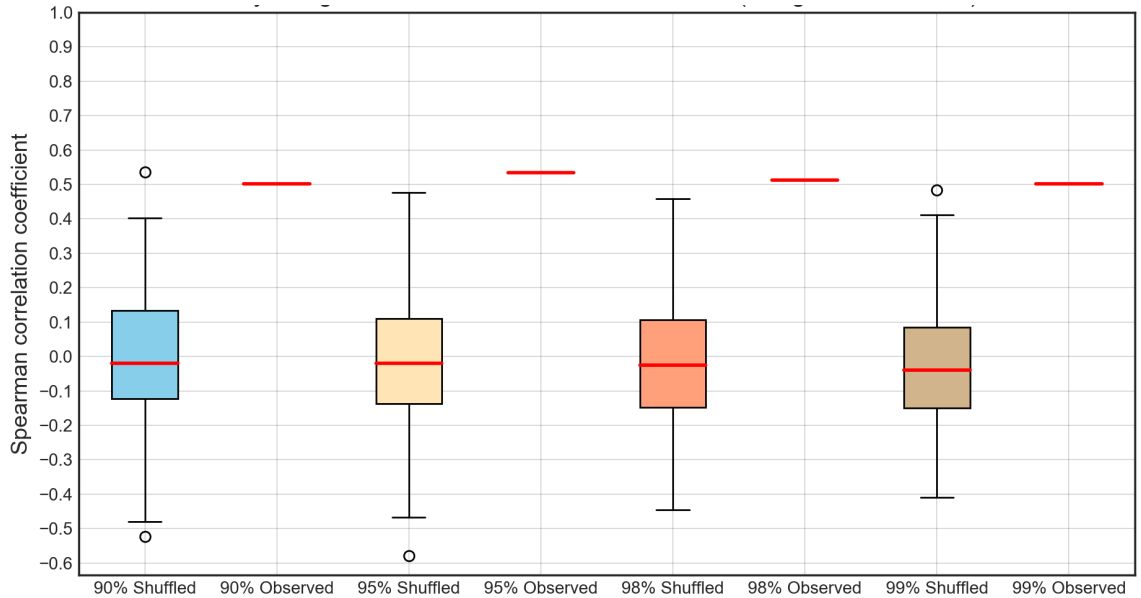


Figure 6.33 Box plot of Spearman correlation coefficient between collective risk and the Hydrological Unit's aggregated claims that the gauge location (ID: 01669000) belongs to, for all thresholds.

6.7. Clustering mechanisms related with GEV distribution simulation

For each one of the 360 gauge locations and for each one of the 4 thresholds, the annual maximum peak was calculated for the observed as well as the shuffled time series. Subsequently, the scale, location and shape parameters were estimated for the observed and the shuffled annual data. The next step was to fit generalized extreme value distribution (GEV) to these block maxima data. The results are shown on the following figures. Once again, the divergence between the observed with the shuffled (independent) time series on the Empirical Cumulative Distribution Function (ECDF) is evident as a clear indication of the existence of the clustering mechanisms. The results regarding the Little Bighorn River at State Line Near Wyola, MT follow (Figure 6.34).

Although, as it was mentioned in the section 4.1, peak-over-threshold methods are preferable in flood insurance practices in contrast to the traditional block maxima methods, which are likely to throw away information, this section highlights the indications of the impacts of the clustering mechanisms on annual maxima.

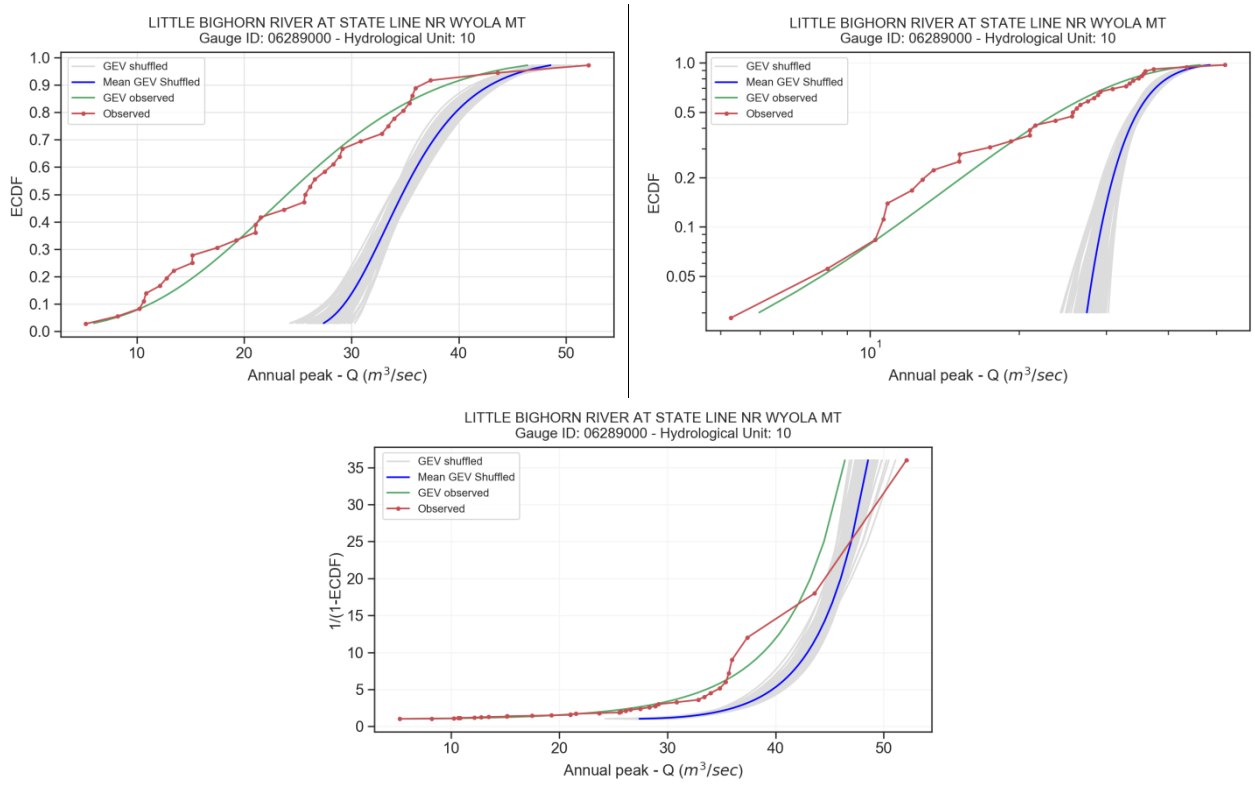


Figure 6.34 Annual peak ECDF diagrams related with GEV simulations in linear, logarithmic and return period (1/(1-ECDF)) scale (Gauge location ID: 11528700).

6.8. A brief case study on spatial dependence mechanisms of US-CAMELS dataset

Extreme river flows often appear to have spatial patterns, as a result, among others, of the complicated hydrological processes, weather systems' interplay and catchment structure. As Quinn et al. (2019) mention, "*the spatial pattern of maximum flood return periods caused by a single weather event at a set of gauge sites across a region is usually characterized by spatial dependence"*; therefore, in such cases, extreme flows in different sites are considered as spatially auto-correlated. In hydrological terms, this highlights the need of risk assessment procedures that account for the impacts of spatial dependence on flood frequency.

On a global scale, firms that operate in the flood insurance and reinsurance sector utilize the theory and the applications of the spatial dependence in order to assess the probability distributions of annual losses to which their portfolio is exposed (i.e., the so-called loss-exceedance curve) and to reassure market regulators. The significance of this process is vital as regulators require evidence that firms can absorb the 0.5% annual probability (1 in 200-year return period) loss in order to ensure that companies can cope with major financial losses caused by extreme flood events. This is why treating the flood frequency at a point as if it were independent of all other points will misestimate the potential losses (Timonina et al., 2015). Moreover, governments, policy-makers and local authorities are interested in spatial dependence mechanisms, as they evaluate economic scenarios for investments and payments regarding flood defenses structures as well as compensations as a result of extreme flood events. In many countries, these compensations are subsidized as part of hybrid insurance programs that involve both the public and the private sector.

In all these cases, univariate extreme value analysis conducted at a series of sites independently cannot generate the year-to-year variation in flood losses that insurers experience (Pielke et al., 2008), and a multivariate analysis considering spatial dependence is required (Quinn et al., 2019). Furthermore, although scientific research (Schumann et al., 2016) has shown that large-scale modeling studies forced by observed gauged flows can capture elements of the historic spatial dependence, the length of these records is a crucial parameter of the process, as the 40- to 50-year length of typical gauged records is not sufficient to simulate all possible patterns of flooding (Serinaldi and Kilsby, 2017).

The available methods that are used to investigate the spatial dependence in streamflow extremes during flood events can be categorized to (Quinn et al., 2019):

- empirical analyses of spatial dependence in gauged records (e.g., Villarini et al., 2011) or model output (e.g., Falter et al., 2015; Lu et al., 2017);
- studies which attempt to fit statistical models of the dependence to large sets of gauge records containing multiple events (Heffernan and Tawn, 2004; Keef et al., 2009).

In the following figures (6.35-6.38), regarding spatial correlation mechanisms in Hydrological Unit 3 of the US-CAMELS dataset, the Spearman correlation coefficients between annual collective risk (which in our study is in fact a streamflow-based proxy for flood claims amounts) of the gauge locations that belong to this Hydrological Unit are presented for all the selected thresholds. It is clear that in lower thresholds, the correlation coefficient is higher across the unit. In contrast, increasing the threshold has a strong impact on the results, as the range of the correlation coefficient across the unit seems to vary greatly, especially when the threshold is set to 99%.

In more detail, when the correlation coefficient between two gauge locations ranges between:

- -0.25 and 0.25, it means that the correlation is practically zero and, as a result, flood events in these gauge locations can be considered as uncorrelated.
- 0.25 and 1, it means that when a flood event occurs in one of these gauge locations, it is highly probable that another event will occur on the other gauge locations, too.
- -0.25 and -1, it means that when a flood event occurs in one of these gauge locations, it is highly improbable that another potential event will occur on the other gauge locations.

Insurance companies aim at having in their portfolios risks which have negative spatial or temporal correlation or, at least, zero correlation, in order to combine and aggregate risks which represent extreme flood events that are unlikely to happen at the same space or time (see chapter 2 and 3). Accordingly, the following figures are extremely significant from an insurance viewpoint, as they reveal which combination of insured properties on the mentioned gauge locations could compose a profitable portfolio in terms of zero or negative dependence.

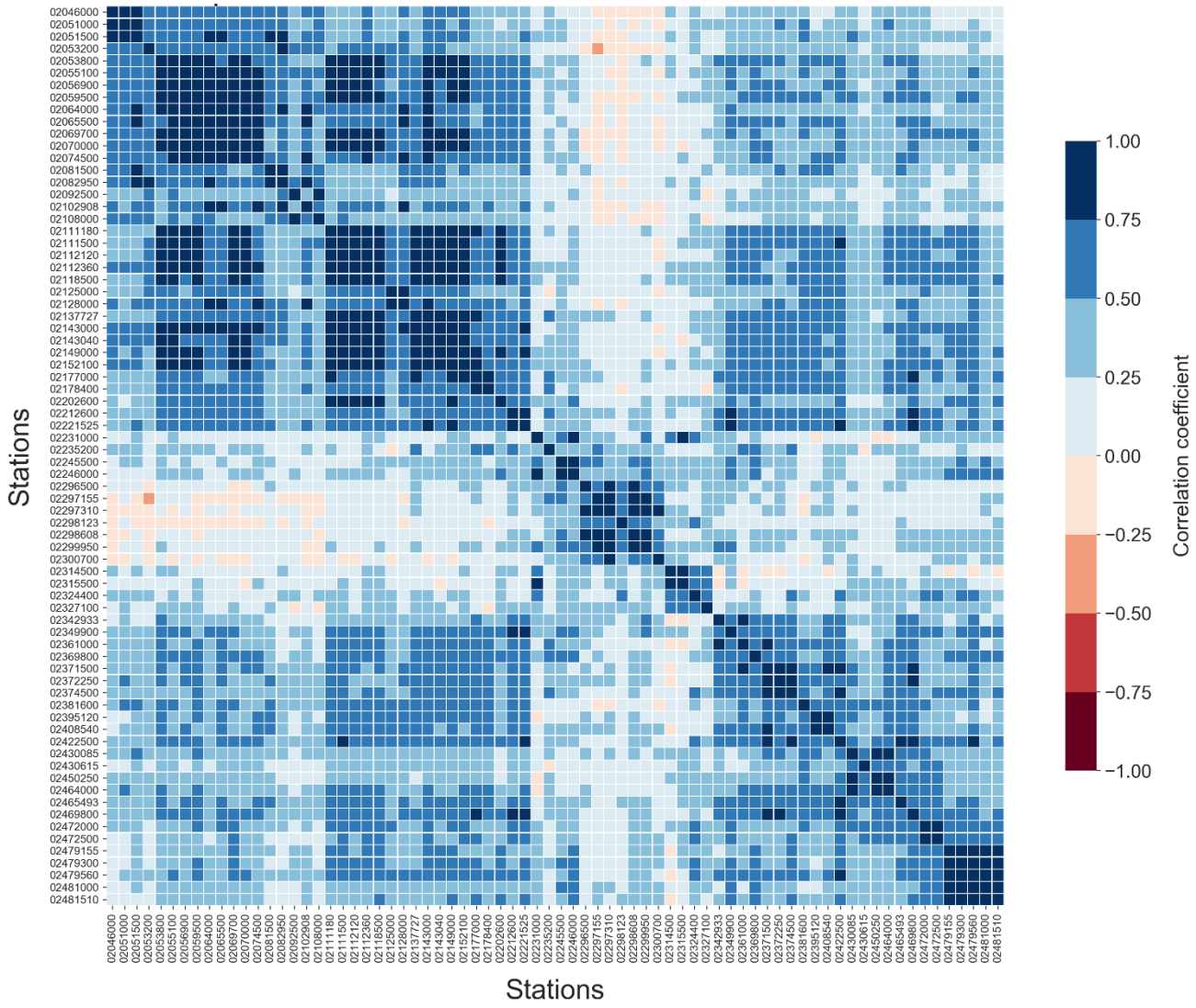


Figure 6.35 Spearman correlation between annual collective risk of gauge locations in Hydrological Unit 3; Threshold 90%.

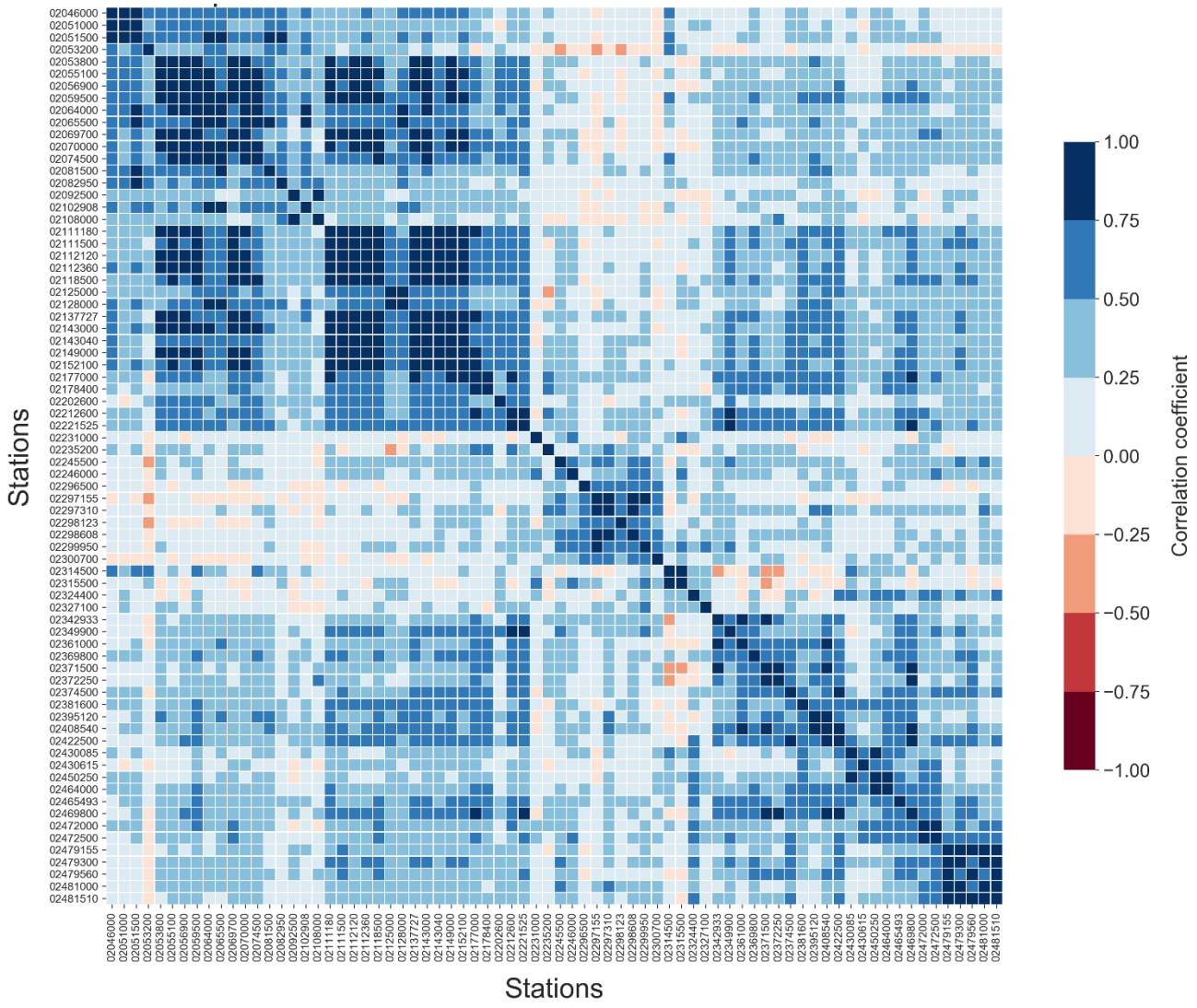


Figure 6.36 Spearman correlation between annual collective risk of gauge locations in Hydrological Unit 3; Threshold 95%.

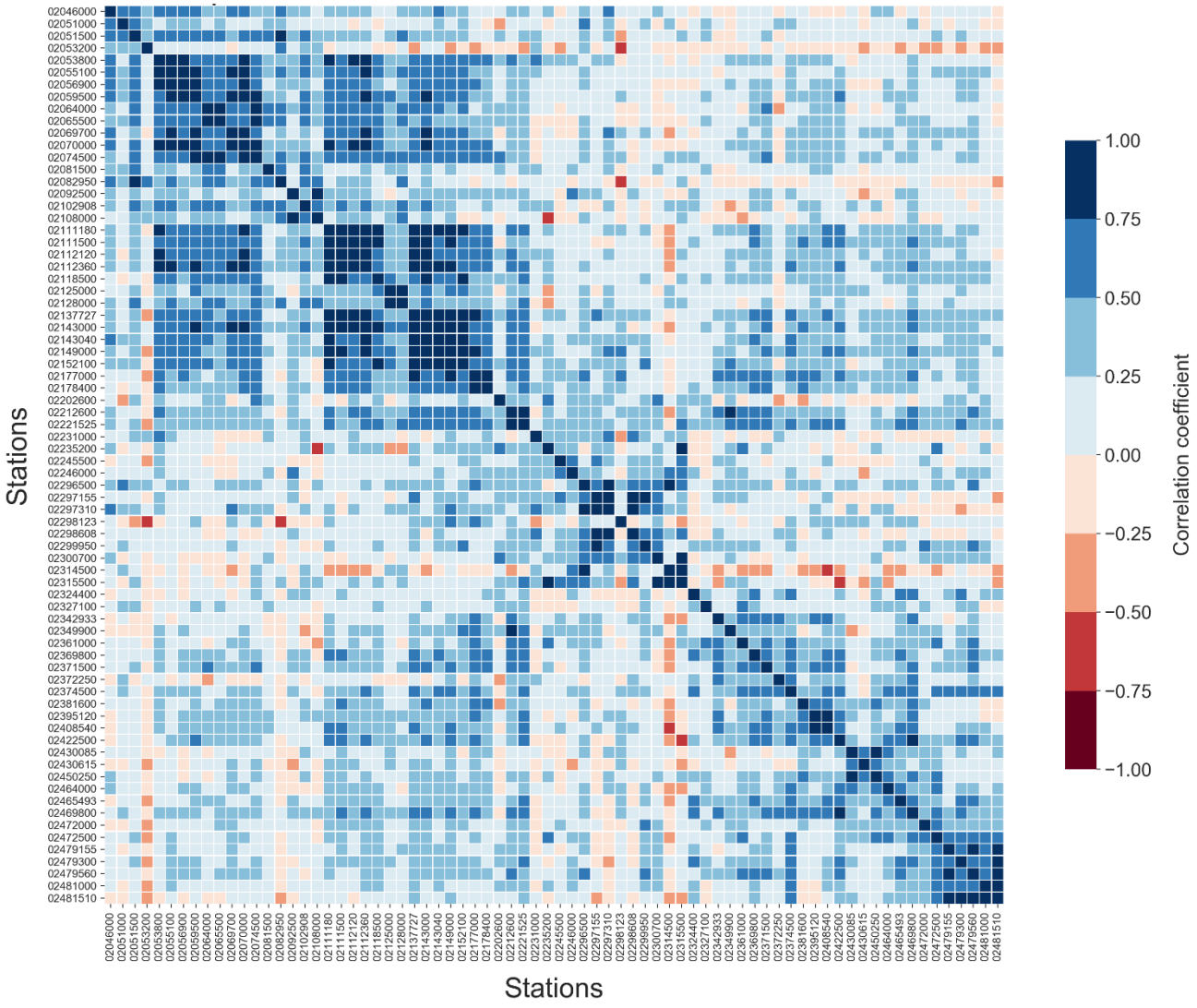


Figure 6.37 Spearman correlation between annual collective risk of gauge locations in Hydrological Unit 3; Threshold 98%.

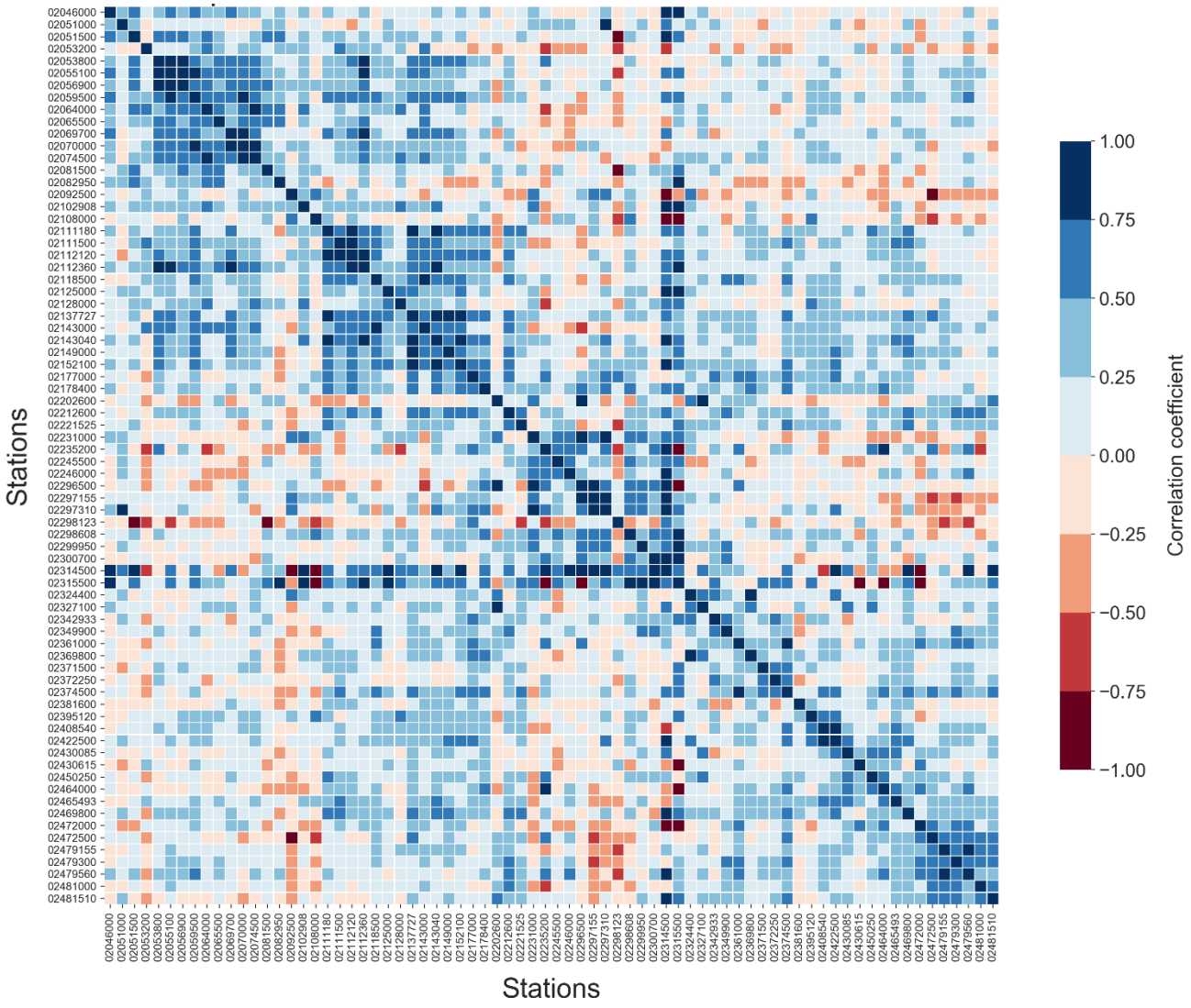


Figure 6.38 Spearman correlation between annual collective risk of gauge locations in Hydrological Unit 3; Threshold 99%.

7. The interplay between precipitation clustering mechanisms, rainfall extremes and actual flood compensations; a Greek case study

7.1. A review of the agricultural insurance practices and market in Greece

The agricultural insurance market in Greece is mainly supported by the Hellenic Agricultural Insurance Organization (ELGA), which is supervised by the Ministry of Agriculture and is wholly owned by the State. ELGA covers all crop damages caused by both natural (such as extreme flood events) and non-natural disasters. The annual maximum amount of compensation that ELGA may pay to its beneficiaries is defined as:

- per beneficiary of compensation: the maximum amount of 250,000€ per year,
- per parcel: 80% of the insurable value of the production of the damaged parcel.

Compensation shall be paid by ELGA when the damage is greater than 20% of the production and it is equal to 88% of the over 15% of the loss. The 15% of loss that is not covered by ELGA can be covered by private insurance companies, such as ERGO Hellas. These additional private insurance products operate in addition to ELGA's coverage and cover the quantitative loss of the insured product. Private providers ensure that, in cooperation with ELGA, the total amount of compensation will be 85% of the total loss (up to 10,000€), based on ELGA's experts reports (see appendix, Table A-2). However, these insurance products are being estimated with simplified statistical methods, which systematically misestimate the risk.

ERGO Hellas insurance company provided us statistics and data, in order to review the current insurance practices in Greece. The company offers crop insurance products which address to farmers, agricultural cooperatives, producer groups and food and agricultural companies. The insurance risks that are insured for crop production are mentioned below.

- Natural hazards (damaging causes) covered by ELGA such as:
  - Hail & frost
  - Windstorm
  - Flood
  - Heat & Solar radiation
  - Rainfall
  - Snow

- Surge storms

- Wildlife Damage covered by ELGA such as:
  - Bear
  - Wild boar
  - Wild rabbits on Lemnos Island

In addition to the above risks, fire incidents and accidents at work are covered by this insurance package.

The Crop Production Insurance Period starts on April 1st of each year and ends on December 31st, while insured can be included in the Program until August 31st. Loss is considered to be the loss of crop production that is expected to be harvested per plot or its qualitative deterioration due to the impact of the covered loss-making causes, which results in a reduction in the objectively expected income of the producers. Insurance coverage is valid after expiry (a) 7 days for "ELGA insurance" and (b) 24 hours for fire from the beginning of insurance.

ELGA provided us statistics and data about the compensations (claim amounts) per crop and per prefecture which have been given to agricultural producers due to damages that occurred on their crops and caused exclusively by flood events during the period 1999-2017 (see appendix, Table A-3). Figure 7.1 shows the aggregated compensations related with flood events that are given per year.

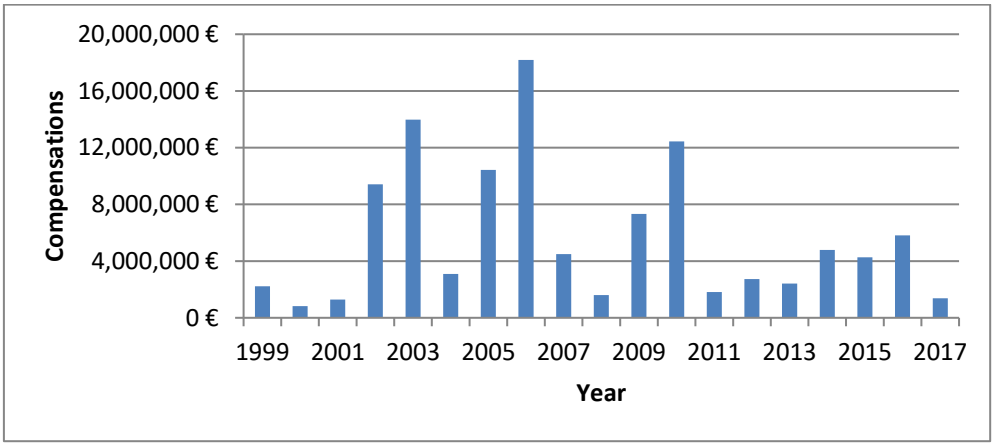


Figure 7.1 The annual aggregated amounts of compensations related with flood events.

7.2. Case study: Larissa (Thessaly), Greece

Regarding precipitation mechanisms, the presence of persistence in annual rainfall is expected to induce clustering in rainfall extremes (Iliopoulou et al., 2018), which should be manifested by clustering of floods. Therefore, in the framework of a case study, it is investigated whether the collective risk estimated using the former as a proxy, i.e. the magnitude of the aggregated rainfall peak over threshold events in a year, is correlated with the actual compensations given.

Following the methodology that was developed in chapters 3 and 4 regarding the annual collective risk computation on streamflow extremes, we investigate at this time, in the frame work of a case study, the clustering mechanisms of rainfall extremes using the available data on the NOAA dataset regarding the station with ID: GHCND:GR000016648 in Larissa (Lat/Lon: 39.65, 22.45), Thessaly, Greece (NOAA, 2019). The data consists of daily precipitation magnitudes for the years 1955-2018, respectively.

Figure 7.2 The location of the selected NOAA station in Larissa, Thessaly, Greece, ID: GHCND:GR000016648.

Once again, in order to characterize the dependence and the clustering mechanisms of the observed time series, we produced 100 new shuffled (randomized) time series, which have the same marginal distribution as the observed (historical) time series but no correlation, considering them as a sequence of independent variables. Selecting 4 thresholds (90%, 95%, 98%, 99%), the annual collective risk,the return intervals and the duration of the over-threshold events are calculated for the observed as well as the shuffled time series.

The results from this procedure follow (Figures 7.3-7.8) and show that the divergence between the observed and the shuffled on the diagram of the Empirical Cumulative Distribution Function (ECDF) of collective risk and return intervals is slight and not as manifested as in the streamflow case studies.

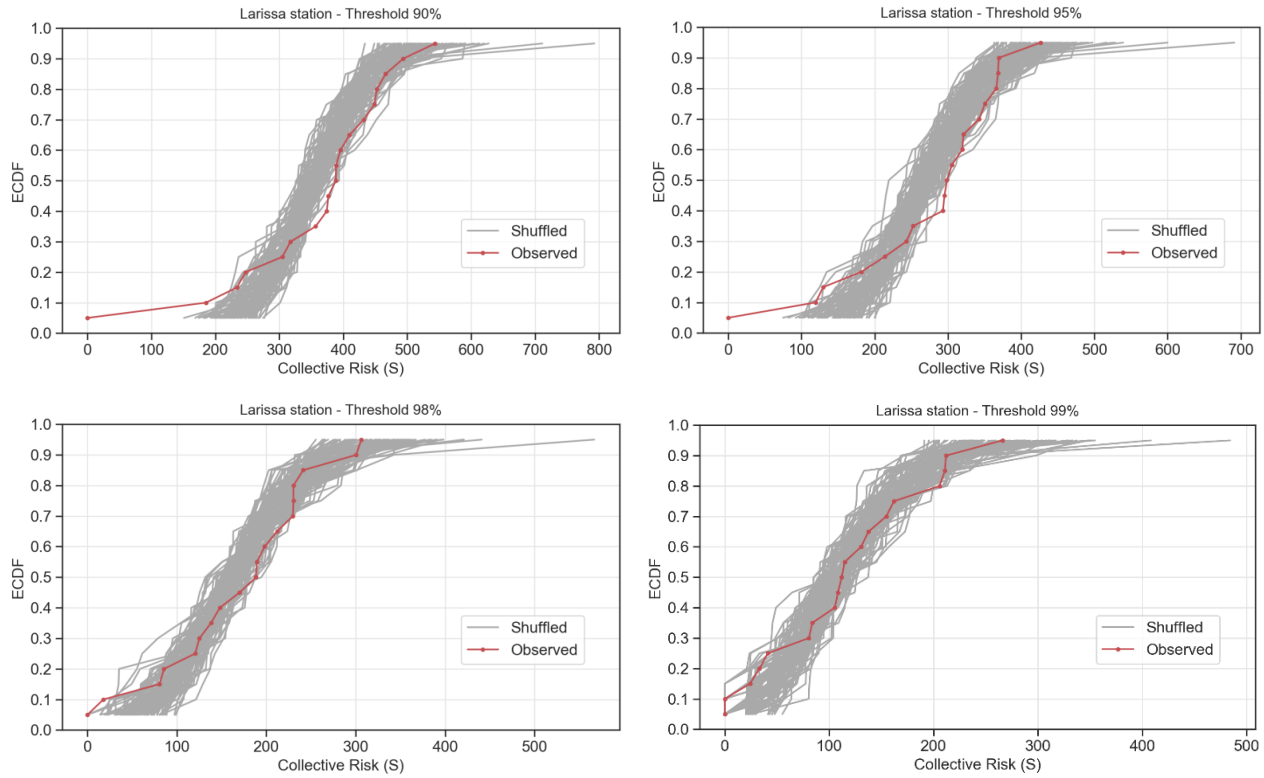


Figure 7.3 Collective risk's ECDF diagrams in linear scale of Larissa station.

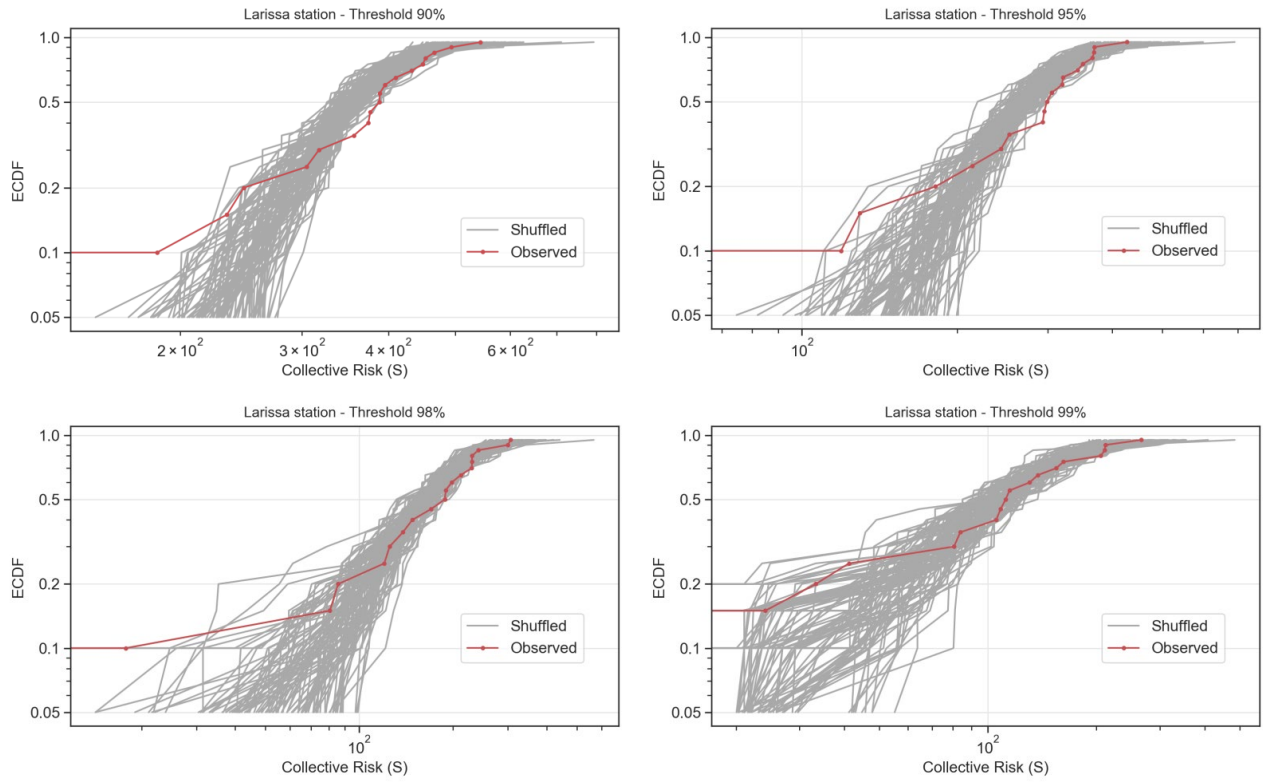


Figure 7.4 Collective risk's ECDF diagrams in logarithmic scale of Larissa station.

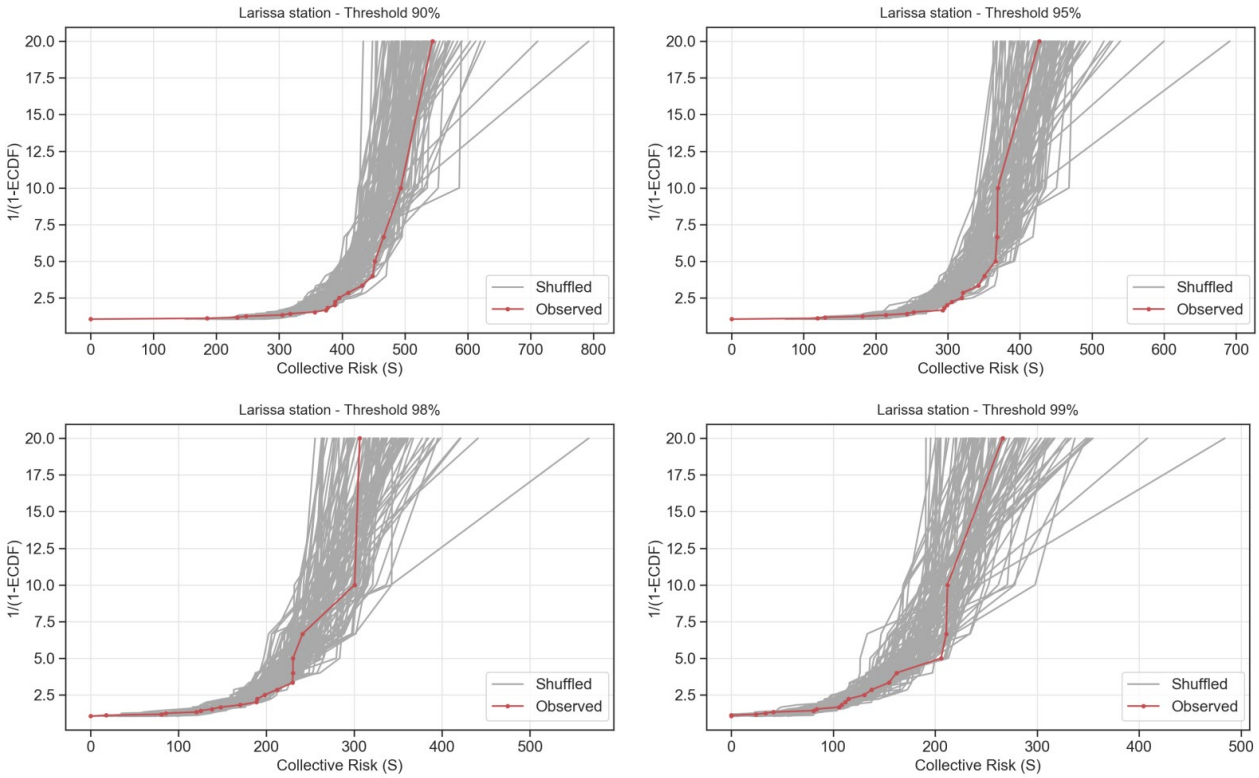


Figure 7.5 Collective risk's return period (1/(1-ECDF)) diagrams in linear scale of Larissa station.

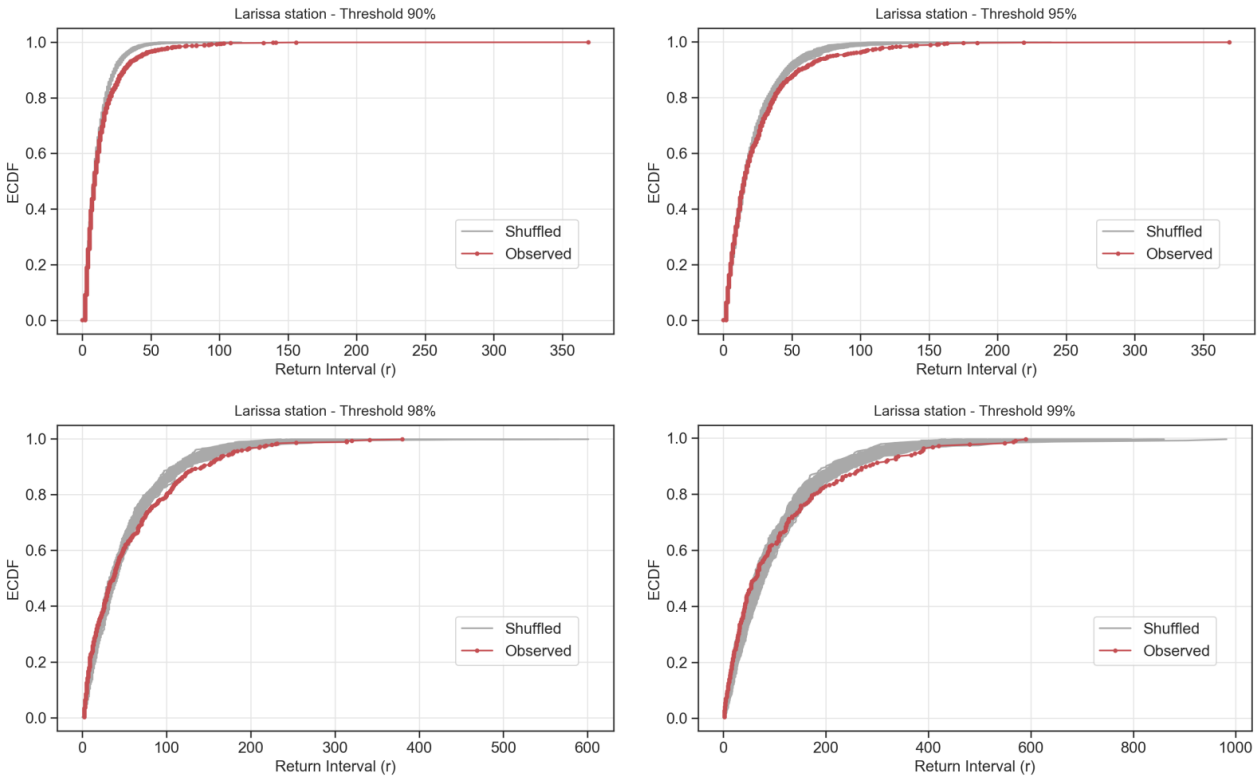


Figure 7.6 Return intervals' ECDF diagrams in linear scale of Larissa station.

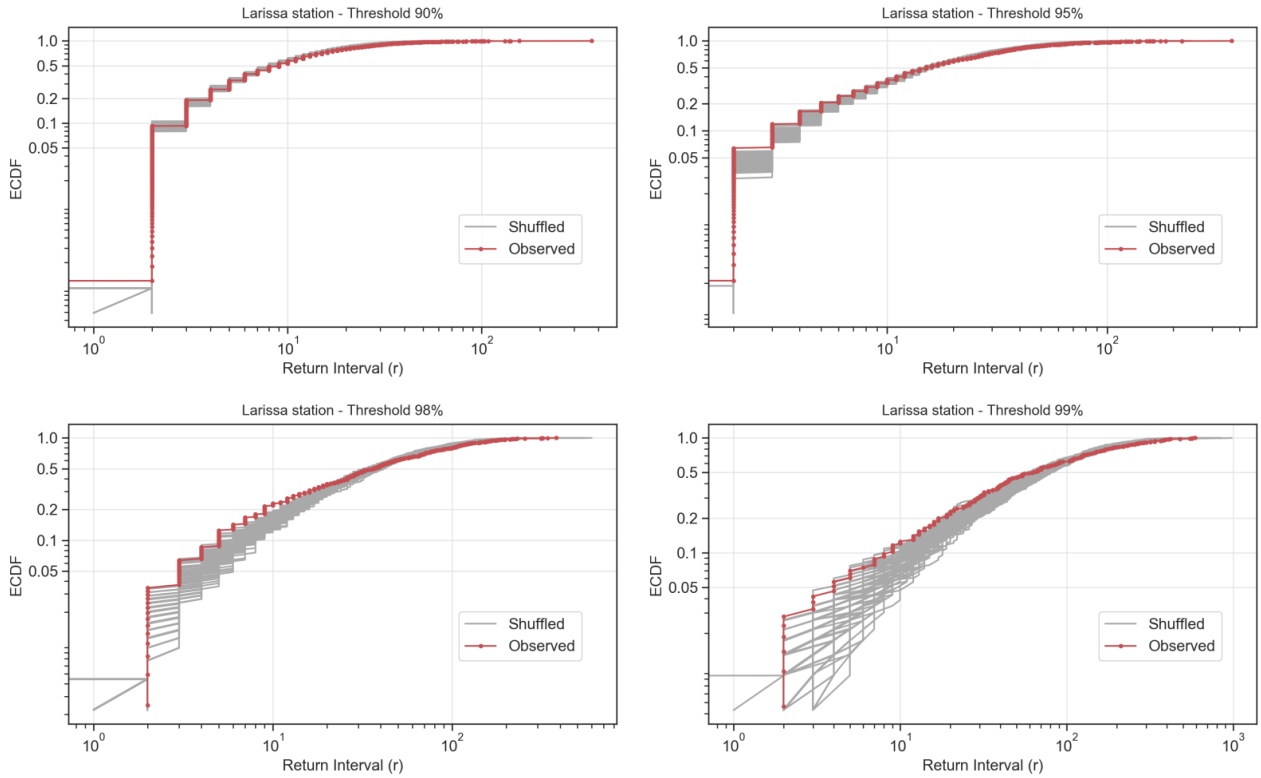


Figure 7.7 Return intervals' ECDF diagrams in logarithmic scale of Larissa station.

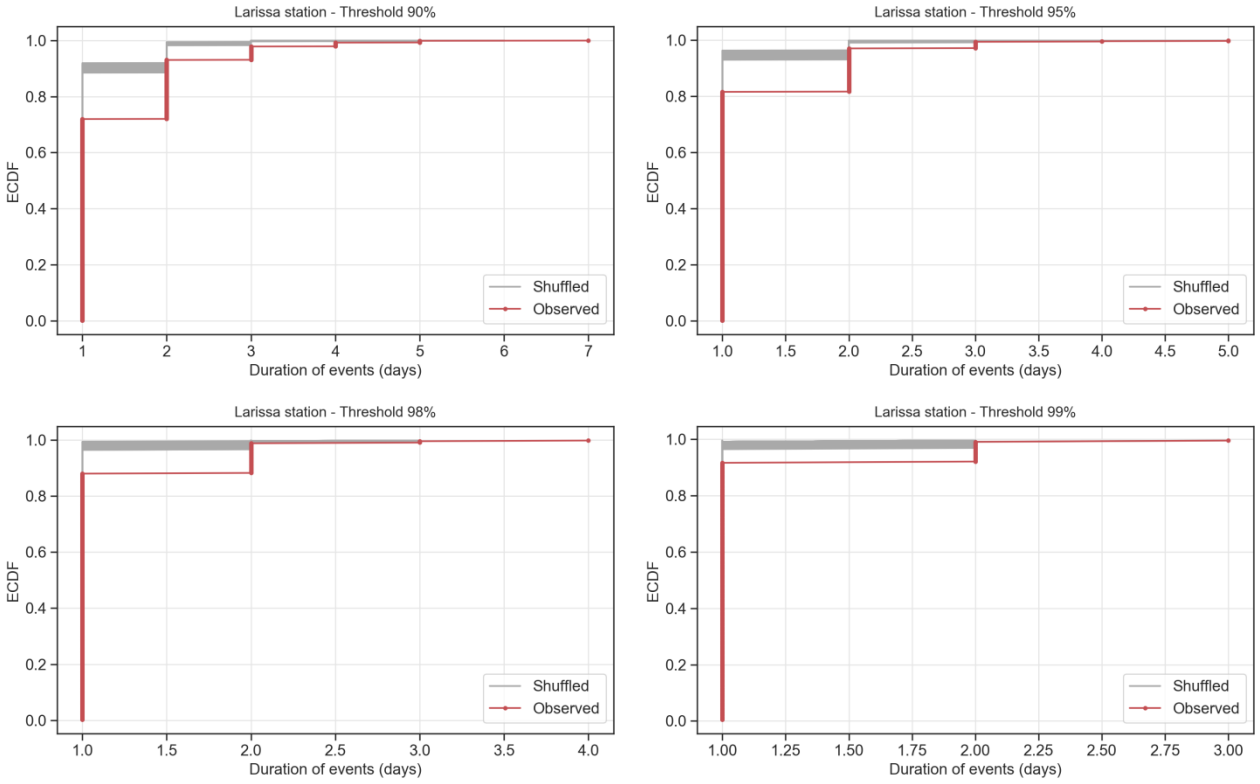


Figure 7.8 Events' duration ECDF diagrams in linear scale of Larissa station.

7.3. Correlating Collective Risk with actual compensations

For all the selected thresholds, the Spearman correlation coefficient between the *Average* Y_i and the Number of the over-threshold events N is calculated for the observed as well as the shuffled time series in order to evaluate the clustering mechanisms on this correlation parameter. Figure 7.9 shows that the divergence of the Spearman correlation coefficient between the observed and the shuffled ones (considered as independent) is negligible.

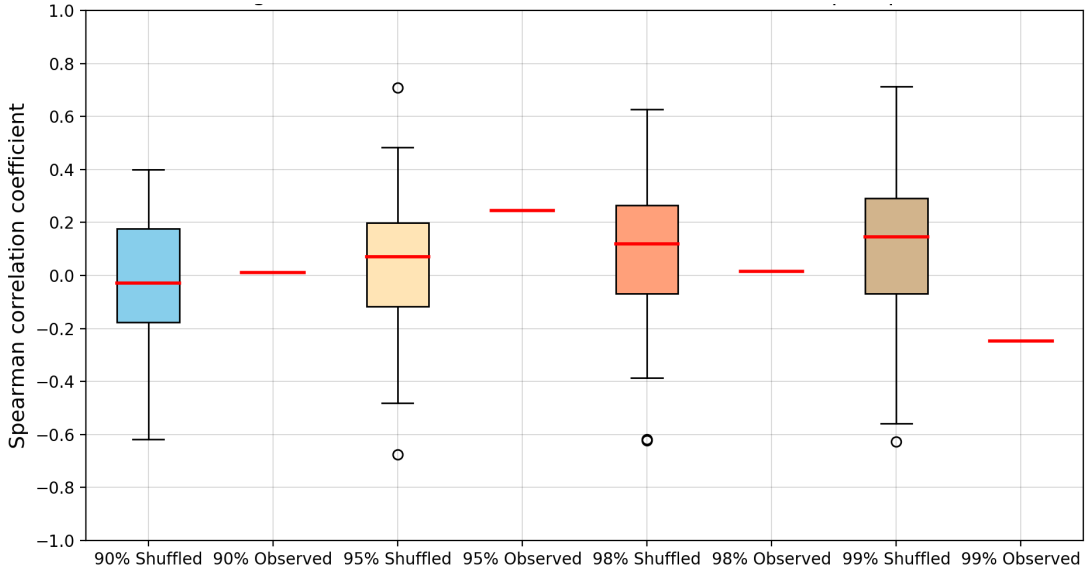


Figure 7.9 Box plot of Spearman correlation coefficient between *Average* Y_i and Number of over-threshold events N for the shuffled as well as the observed time series for all thresholds regarding the 1999-2017 period (Larissa station).

Exploiting the actual compensation amounts data regarding the Larissa prefecture (see appendix, Table A-3), Figure 7.10 offers a depiction of the Spearman correlation coefficients between these amounts and the stations' annual collective risk for the years 1999-2017, respectively.

Regarding precipitation clustering mechanisms, the case study in Larissa region highlights a fair correlation between the annual collective risk of the observed time series and the actual compensations, in contrast to correlation regarding the shuffled ones. In more detail, as for low thresholds (90% and 95%), the correlation coefficient of the observed is out of the boundaries of the shuffled ones. Moreover, increasing threshold (98% and 99%), provokes a decrease in the correlation coefficient of the observed, even though it is still close enough to the boundaries of the shuffled ones.

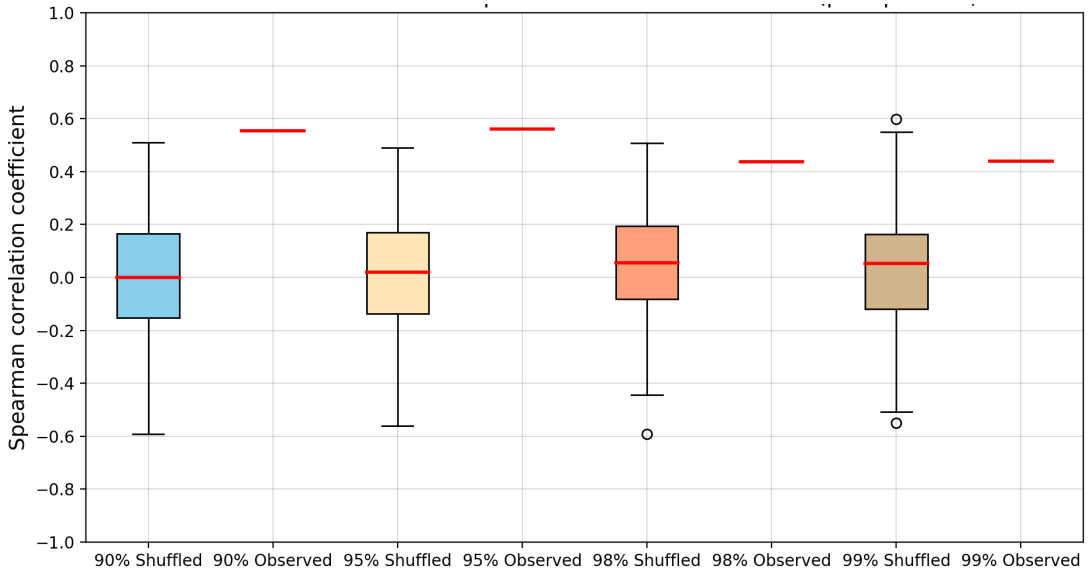


Figure 7.10 Box plot of Spearman correlation coefficient between annual collective risk and actual compensations for the shuffled as well as the observed time series for all thresholds regarding the 1999-2017 period (Larissa station).

Initially, the ECDF diagrams of collective risk and return intervals, as well as the low correlation of the observed and the shuffled between *Average* Y_i and Number of over-threshold events N indicated that the impact of the precipitation clustering mechanisms on this case study is slight. On the contrary, the correlation between the annual collective risk (which is produced by precipitation data) and actual compensation amounts is remarkable. This is of outmost importance for the insurance companies, as it creates a relationship of dependence between the primary index of precipitation (in terms of magnitude) and the actual amounts of compensations that these companies would pay in cases of extreme precipitation events. Nevertheless, the accuracy of the data regarding the actual compensations given that ELGA provided us should be furtherly investigated in order to validate whether the claimed and given amounts are real and not manipulatively raised by the beneficiaries.

8. Conclusions

From the very beginning, the aim of this study was to provide new insights into the spatiotemporal clustering mechanisms of streamflow and rainfall extremes, in order to pave the way for a more accurate risk assessment on flood insurance and reinsurance practices, as a non-structural measure against the impacts of extreme flood events on individuals and societies. Moreover, the classic assumption of independence between the frequency and the severity of extreme flood events as well as the assumption that insurable flood losses are independent were meant to be under investigation. Results and conclusions can be summarized in the following sections.

8.1. Collective risk, return intervals and duration of flood events

Regarding the impacts of clustering mechanisms on collective risk, return intervals and the duration of the over-threshold events of US-CAMELS dataset, our analysis showed that, for all thresholds, the divergence between the observed and the shuffled (randomized, considered as independent) time series on the diagram of the Empirical Cumulative Distribution Function (ECDF) is evident in many gauge locations. These results are a clear indication of the existence of clustering of streamflow extremes on the observed time series and its impacts on increasing claim amounts (in terms of collective risk) and the severity of the over-threshold events (in terms of return intervals and duration of over-threshold events). In other words, this persistent behavior reflects directly on financial parameters.

8.2. HK dynamics and Monte Carlo simulation

Based on the mean *Climacogram* of the GHK process regarding the 360 empirical streamflow time series of the US-CAMELS dataset, the Hurst parameter was estimated 0.63, which indicates clearly a persistent behavior. The effect of this dependence structure is tracked on the behaviors of POT flows at the annual scale and the estimation of the collective risk proxy. Subsequently, the behavior of daily streamflow in our dataset is found to be consistent with HK dynamics characterized by moderate H parameters (in the range 0.6-0.7), through Monte Carlo simulations. Regarding the Monte Carlo simulations based on the SMA-GHK model, the diagrams of the empirical cumulative distribution function (ECDF) curve of the observed collective risk proxy is contained in the Monte Carlo prediction limits by the model, preserving the HK dynamics and the 4 four moments. In contrary, shuffled (randomized) curves have a different behavior, especially in the tails of the distribution.

8.3. Correlation between *Average* Y_i and Number of over-threshold events *N*

A common assumption in the computation of collective risk is the independence between *Average* Y_i and Number of over-threshold events *N*. Insurance companies' concern about this correlation factor is noteworthy, due to the fact that they try to investigate the dependence between the annual number of extreme events and the provoked *Average* Y_i, which is a proxy of the average claim amounts per over-threshold event on a specific region. Initially, we showed that the assumption that years which are more active in terms of Number of events *N* tend to exhibit extreme events also in terms of *Average Yi* magnitude generally holds true yet it cannot be universally applied. We categorized spatially the areas with high Spearman correlation coefficient between the *Average* Y_i and the Number of over-threshold events *N*, and the others (areas) where the correlation coefficient is noticeably lower. In other words, this spatial categorization indicates the regions that are subjected to numerous claim amounts in case of a year that an extreme number of over-threshold events occur.

Subsequently, we highlighted through a box plot analysis the existence of clustering mechanisms that are prevailing over the observed data, as they introduce significant correlation between the *N* and *Average* Y_i in many gauge locations, in contrast to the shuffled ones, in which the correlation is zero.

8.4. The association between streamflow-based collective risk estimation and FEMA's NFIP actual claims records

Regarding the association between the streamflow-based collective risk proxy used herein and the FEMA's NFIP actual claims records, a spatial pattern is evident, showing that higher values of Spearman correlation coefficient between the two emerge in West Coast, in contrast to the ones in East Coast, which are significantly lower.

As collective risk refers to river flooding, these results show that this type of flooding is dominant in West Coast. In contrast, it is revealed that the source of flooding events that provoke claims in East Coast present a different and more complicated pattern, mainly due to the significant vulnerability of these areas on accelerated flooding events due to hurricane hits, sea-level rise (SLR) and storm surge phenomena (Ezer and Atkinson, 2014). Moreover, these results are also explained by the fact that during relaxed North Atlantic Oscillation (NAO), Gulf Coast is more

susceptible to major hurricane strikes; on the contrary, during excited NAO, such phenomena are strongly observable in the East Coast (Elsner et al., 2000).

Furthermore, we performed a box plot analysis of the Spearman correlation coefficient between the collective risk and the claims records of the Hydrological Unit that the gauge location belongs to, which was calculated for the observed as well as the shuffled (independent) time series. This analysis showed that the dominant clustering mechanisms introduce significant correlation, as the divergence of the correlation coefficient between the observed and the shuffled ones (independent) is evident in many gauge locations. In more detail, the shuffled ones tend to underestimate the correlation coefficient in comparison with the observed ones. The apparent existence of clustering mechanisms, which are indicated by this divergence between the observed and the shuffled, are considered to have significant financial impacts for the insurance companies, due to inaccurate risk assessment processes. Underestimation of collective risk, in cases where clustering mechanisms are not included in the calculations, could provoke unpredictable large values of collective risk (claim amounts) in a potential extreme flood event, stressing their reserves.

8.5. Spatial dependence on US-CAMELS dataset

Regarding the spatial dependence mechanisms of US-CAMELS dataset, these mechanisms were investigated in Hydrological Unit 3, as part of a case study. The analysis showed that in lower thresholds, the correlation coefficient is higher across the unit. In contrast, increasing the threshold has a strong impact on the results, as in that case the range of the correlation coefficient across the unit seems to vary greatly, especially when the threshold is set to 99%. Moreover, the extracted heat maps revealed which combination of insured properties on the mentioned gauge locations could compose a profitable portfolio in terms of zero or negative spatial dependence, as insurance companies' aim is to have in their portfolio risks which have negative spatial or temporal correlation or, at least, zero correlation, in order to combine and aggregate risks which represent potential extreme flood events that are unlikely to happen at the same space or time.

8.6. Precipitation clustering mechanisms and correlation with actual claim amounts

Eventually, we reviewed the agricultural insurance practices in the public as well as in the private sector of Greece and we presented the compensations' collected data that ELGA provided us, caused exclusively by flood events. We applied again the collective risk model on precipitation time

series, as part of a case study in Larissa region, for the period 1955-2018. We showed that the impact of clustering mechanisms on collective risk, return intervals and the Spearman correlation coefficient between *Average* Y_i and Number of over-threshold events N, for all thresholds, is negligible. Nevertheless, this case study highlighted a fair correlation between the annual collective risk of the observed time series and the actual compensations (period 1999-2017), in contrast to correlation regarding the shuffled ones. In more detail, as for low thresholds (90% and 95%), the correlation coefficient of the observed is out of the boundaries of the shuffled ones. Moreover, increasing threshold (98% and 99%), provokes a decrease in the correlation coefficient of the observed, even though it is still close enough to the boundaries of the shuffled ones. This is of outmost importance for the insurance companies, as it creates a relationship of dependence between the primary index of precipitation (in terms of magnitude) and the actual amounts of compensations in cases of extreme precipitation events.

8.7. Suggestions for future research

The driving force of this study was to investigate the validity of the assumption of temporal independence of extreme flood events for insurance purposes. From the experience gained so far regarding the spatiotemporal stochastic nature of flood insurance variables, several issues have been detected for future research. In more detail:

- Further study must follow regarding the deeper understanding why the Spearman correlation coefficients between the streamflow-based collective risk proxy (used herein) and the FEMA's NFIP actual claims records found to be higher in West Coast in contrast to the ones in East Coast, which appeared to be significantly lower.
- The development of hydraulic and hydrologic simulations must be encouraged in order to validate the spatiotemporal stochastic patterns of floods in US-CAMELS dataset, evaluating the flood susceptibility for the vulnerable locations.
- More research is essential in order to improve the effectiveness and the robustness of current computational techniques regarding flood insurance modelling strategies, as they are characterized nowadays by great complexity. Although the stochastic structure of extremes of flood insurance variables is evident, computational complexity is a principle reason why the gap between research in hydrological dependence and practical applications in insurance, engineering and finance is still widened.
- Following the FEMA's NFIP claims and policy data publication, scientists and policy-makers have the opportunity to study more extensively and to investigate potential trends in

the interplay between flood risk, climate variability and their impact on flood insurance claims and losses. The role of the three basic ocean-atmosphere oscillation modes (ENSO, IPO and AMO) in the economic outcomes must also be thoroughly investigated.

9. References

Barredo, J. I. (2009). Normalised flood losses in Europe: 1970–2006. *Natural Hazards and Earth System Sciences*, 9, 97–104.

Brown, R.L. (1993). *Introduction to Ratemaking and Loss Reserving for Property and Casualty Insurance*. ACTEX Publications.

Cont, R. (2007). Volatility Clustering in Financial Markets: Empirical Facts and Agent-Based Models. *Teyssière G., Kirman A.P. (eds) Long Memory in Economics*, Springer, Berlin, Heidelberg, 289-309.

CRED, and UNISDR (2018). *Economic losses, poverty & disasters: 1998-2017*. [Online] Available at: https://www.undrr.org/publication/economic-losses-poverty-disasters-1998-2017 [Accessed July 2020]

Crompton, R. P., and McAneney K.J. (2008). Normalised Australian insured losses from meteorological hazards: 1967–2006. *Environmental Science & Policy*, 11 (5), 371-378.

Dimitriadis, P. (2017). *Hurst-Kolmogorov dynamics in hydrometeorological processes and in the microscale of turbulence*. PhD thesis, Department of Water Resources and Environmental Engineering, National Technical University of Athens, Greece.

Dimitriadis, P., and Koutsoyiannis, D. (2015). Climacogram versus autocovariance and power spectrum in stochastic modelling for Markovian and Hurst–Kolmogorov processes. *Stochastic Environmental Research & Risk Assessment*, 29 (6), 1649–1669.

Dimitriadis, P., and Koutsoyiannis, D. (2018). Stochastic synthesis approximating any process dependence and distribution. *Stochastic Environmental Research & Risk Assessment*, 32 (6), 1493-1515.

Dimitriadis, P., and Koutsoyiannis, D. (2020). The mode of the climacogram estimator for a Gaussian Hurst-Kolmogorov process. *Journal of Hydroinformatics*, 22 (1), 160-169.

Dimitriadis, P., Tegos, A., Oikonomou, A., Pagana, V., Koukouvinos, A., Mamassis, N., Koutsoyiannis, D., and Efstratiadis, A. (2016). Comparative evaluation of 1D and quasi-2D hydraulic models based on benchmark and real-world applications for uncertainty assessment in flood mapping. *Journal of Hydrology*, 534, 478-492.

Eikenberg, C. (1998). *Journalistenhandbuch zum Katastrophenman-agement*, fifth edition, German IDNDR-Committee, Bonn.

Elsner, J.B., Liu, K., and Kocher, B. (2000). Spatial Variations in Major U.S. Hurricane Activity: Statistics and a Physical Mechanism. Journal of Climate, 13(13), 2293-2305.

European Union (2007). Directive 2007/60/EC of the European Parliament and of the Council of 23 October 2007 on the assessment and management of flood risks. *Official Journal of the European Union*, L 288/27.

Ezer, T., and Atkinson, L.P. (2014). Accelerated flooding along the U.S. East Coast: On the impact of sea-level rise, tides, storms, the Gulf Stream, and the North Atlantic Oscillations. Earth's Future, 2 (8), 362-382.

Falter, D., Schröter, K., Dung, N. V., Vorogushyn, S., Kreibich, H., Hundecha, Y., Apel, H., and Merz, B. (2015). Spatially coherent flood risk assessment based on long-term continuous simulation with a coupled model chain. *Journal of Hydrology*, 524, 182- 193.

Feldstein, S.G., and Fabozzi, F.J. (2008). *The Handbook of Municipal Bonds*. Wiley.

FEMA (1986). *A Unified National Program for Floodplain Management*. Washington, D.C.: FEMA.

FEMA (2019). *FEMA publishes NFIP claims and policy data*. [Online]
Available at: https://www.fema.gov/openfema-data-page/fima-nfip-redacted-claims-v1
Last Data Refresh: 10-24-2021
[Accessed: October 2021]

FEMA (2020). *Historical Flood Risk and Costs*. [Online]
Available at: https://www.fema.gov/data-visualization/historical-flood-risk-and-costs
[Accessed: October 2021]

FEMA NFIP (2019). *The National Flood Insurance Program*. [Online]
Available at: https://www.fema.gov/national-flood-insurance-program
[Accessed: October 2021]

Fisher, R.A., and Tippett, L.H.C. (1928). Limiting forms of the frequency distribution of the largest and smallest member of a sample. *Proceedings of the Cambridge Philosophical Society*, 24 (2), 180-190.

Gollier, C. (2003). To Insure or Not to Insure?: An Insurance Puzzle. *The Geneva Papers on Risk and Insurance Theory*, 28 (1), 5-24.

Goulianou, T., Papoulakos, K., Iliopoulou, T., Dimitriadis, P., and Koutsoyiannis, D. (2019). Stochastic characteristics of flood impacts for agricultural insurance practices. *European Geosciences Union General Assembly 2019*, Geophysical Research Abstracts, Vol. 21, European Geosciences Union, Vienna, EGU2019-5891.

Heffernan, J. E., and Tawn, J. A. (2004). A conditional approach for multivariate extreme values (with discussion). *Journal of the Royal Statistical Society, Series B: Statistical Methodology*, 66 (3), 497-546.

Hurst, H.E. (1951). Long Term Storage Capacities of Reservoirs. *Transactions of the American Society of Civil Engineering*, 116, 776-808.

ICE (2001). *Learning to live with rivers,* Institution of Civil Engineers, London, UK.

Iliopoulou, T., and Koutsoyiannis, D. (2019). Revealing hidden persistence in maximum rainfall records, *Hydrological Sciences Journal*, 64 (14), 1673-1689.

Iliopoulou, T., Papalexiou, S.M., Markonis, Y., and Koutsoyiannis, D. (2018). Revisiting long-range dependence in annual precipitation. *Journal of Hydrology*, 556, 891-900.

Integrated Research on Disaster Risk (2014). *Peril Classification and Hazard Glossary (IRDR DATA Publication No. 1).* Beijing: Integrated Research on Disaster Risk.

International Monetary Fund (2020). *Global Financial Stability Report: Markets in the Time of COVID-19*, Washington, DC, April.

Jenkinson, A.F. (1955). The frequency distribution of the annual maximum (or minimum) values of meteorological elements. *Quarterly Journal of the Royal Meteorological Society*, 81 (348) 158-171.

Kaas, R., Goovaerts, M., Dhaene, J., and Denuit, M. (2008). *Modern Actuarial Risk Theory Using R*. Springer.

Kawasaki, A., Kawamura, G., and Zin, W. W. (2020). A local level relationship between floods and poverty: A case in Myanmar. *International Journal of Disaster Risk Reduction*, 42.

Keef, C., Svensson, C., and Tawn, J. A. (2009). Spatial dependence in extreme river flows and precipitation for Great Britain. *Journal of Hydrology*, 378 (3-4), 240-252.

Kendall, M. (1938). A New Measure of Rank Correlation. *Biometrika*, 30 (1-2), 81-89.

Kolmogorov, A.N. (1940). Wienersche Spiralen und Einige Andere Interessante Kurven in Hilbertschen Raum. *Doklady Akademii nauk URSS*, 26, 115- 118.

Koutsoyiannis, D. (2000). A generalized mathematical framework for stochastic simulation and forecast of hydrologic time series. *Water Resources Research*, 36 (6), 1519-1533.

Koutsoyiannis, D. (2010). A random walk on water. *Hydrology and Earth System Sciences*, 14, 585-601.

Koutsoyiannis, D. (2011). Hurst-Kolmogorov dynamics and uncertainty. *Journal of the American Water Resources Association*, 47 (3), 481-495.

Koutsoyiannis, D. (2016). Generic and parsimonious stochastic modelling for hydrology and beyond. *Hydrological Sciences Journal*, 61 (2), 225-244.

Kron, W. (2005). Flood Risk = Hazard • Values • Vulnerability. *Water International*, 30 (1), 58-68.

Kulp, C., and Hall, J. (1968). *Casualty Insurance*, The Ronald Press Company, fourth edition.

Lotsch, A., Dick, W., and Manuamorn, O.P. (2010). *Assessment of innovative approaches for flood risk management and financing in agriculture*, Agriculture and rural development discussion paper, no. 46, Washington, D.C. : World Bank Group.

Lu, P., Smith, J. A., and Lin, N. (2017). Spatial characterization of flood magnitudes over the drainage network of the Delaware River basin. *Journal of Hydrometeorology*, 18 (4), 957- 976.

Manolis, G. T., Papoulakos, K., Iliopoulou, T., Dimitriadis, P., Tsaknias, D., and Koutsoyiannis, D. (2020). Clustering mechanisms of flood occurrence; modelling and relevance to insurance practices. *EGU General Assembly 2020, Online, 4-8 May 2020*, EGU2020-9357.

Mehr, R., and Camack, R. (1976). *Principles of Insurance*, Homewood, Ill.: R.D. Irwin, sixth edition.

Morrison, J.E., and Smith, J.A. (2002). Stochastic modeling of flood peaks using the generalized extreme value distribution. *Water Resources Research*, 38 (12), 41-1-41-12.

Munich Reinsurance Company (1997). *Flooding and Insurance*.

Newman, A., Sampson, K., Clark, M. P., Bock, A., Viger, R. J., and Blodgett, D. (2014). *A large-sample watershed-scale hydrometeorological dataset for the contiguous USA*. Boulder, CO: UCAR/NCAR.

NOAA (2019). *Data Tools: Find a Station*. [Online]
Available at: https://www.ncdc.noaa.gov/cdo-web/datatools/findstation
[Accessed: October 2021]

Papoulakos, K., Iliopoulou, T., Dimitriadis, P., Tsaknias, D., and Koutsoyiannis, D. (2020). Investigating the impacts of clustering of floods on insurance practices; a spatiotemporal analysis in the USA. *EGU General Assembly 2020, Online, 4-8 May 2020*, EGU2020-866.

Pearson, K. (1895). Notes on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58, 240-242.

Pickands, J. (1975). Statistical inference using extreme order statistics. *Annals of Statistics*, 3, 119-131.

Pielke, R. A., Gratz, J., Landsea, C. W., Collins, D., Saunders, M. A., and Musulin, R. (2008). Normalized hurricane damage in the United States: 1900-2005. *Natural Hazards Review*, 9 (1), 29- 42.

Plate, E. (2002). Flood risk and flood management. *Journal of Hydrology*, 267 (1-2), 2-11.

Quinn, N., Bates, P.D., Neal, J., Smith, A., Wing, O., Sampson, C., Smith, J., and Heffernan, J. (2019). The Spatial Dependence of Flood Hazard and Risk in the United States. *Water Resources Research*, 55 (3), 1890-1911.

Ramanathan, K.A., Bose, V. (2018). Insurance Penetration and Insurance Density in India - An Analysis. *International Journal of Research and Analytical Reviews*, 5 (4), 229-232.

Reiss, R.D., and Thomas, M. (2007). *Statistical Analysis of Extreme Values with Applications to Insurance, Finance, Hydrology and Other Fields*. Birkhäuser.

Rodriguez-Oreggia, E., De La Fuente, A., De La Torre, R., and Moreno. H.A. (2013). Natural Disasters, Human Development and Poverty at the Municipal Level in Mexico. *The Journal of Development Studies*, 49 (3), 442-455.

Royal, C., Evans, J., and Windsor, S.S. (2014). The missing strategic link - human capital knowledge, and risk in the finance industry - two mini case studies. *Venture Capital. An International Journal of Entrepreneurial Finance.*, 16 (3), 189-206.

Sampson, C. C., Smith, A. M., Bates, P. D., Neal, J. C., Alfieri, L., and Freer, J. E. (2015). A high-resolution global flood hazard model, Water Resources Research, 51 (9), 7358-7381

Schanze, J., Zeman, E., and Marsalek, J. (2016). Flood Risk Management: Hazards, Vulnerability and Mitigation Measures. *NATO Science Series*, 67,. Springer, Dordrecht.

Schumann, G. J. -P., Stampoulis, D., Smith, A. M., Sampson, C. C., Andreadis, K. M., Neal, J. C., and Bates, P. D. (2016). Rethinking flood hazard at the global scale. *Geophysical Research Letters*, 43, 10,249-10,256.

Seaber, P.R., Kapinos, F.P., and Knapp, G.L. (1987). *Hydrologic Unit Maps*. United States Geological Water-Supply Paper 2294.

Serinaldi, F. and Kilsby, C.G. (2016). Understanding Persistence to Avoid Underestimation of Collective Flood Risk. *Water*, 8 (4), 152

Serinaldi, F., and Kilsby, C. G. (2017). A blueprint for full collective flood risk estimation: Demonstration for European River flooding. *Risk Analysis*, 37 (10), 1958- 1976.

SOPAC (2009). Relationship between natural disasters and poverty: a Fiji case study. *SOPAC Miscellaneous Report 678*.

Spearman, C. (1904). The proof and measurement of association between two things. *American Journal of Psychology*, 15 (1), 72-101.

Tay, K. (2019). *Looking at flood insurance claims with choroplethr*. [Online]
Available at: https://github.com/kjytay/misc/blob/master/blog/2019-07-14_flood_data.R
[Accessed: July 2020]

Timonina, A., Hochrainer-Stigler, S., Pflug, G., Jongman, B., and Rojas, R. (2015). Structured Coupling of Probability Loss Distributions: Assessing Joint Flood Risk in Multiple River Basins. *Risk Analysis : an Official Publication of the Society for Risk Analysis*, 35 (11), 2102-2119.

United Nations (2015). *Sendai Framework for Disaster Risk Reduction 2015-2030*. [Online] Available at: https://www.undrr.org/publication/sendai-framework-disaster-risk-reduction-2015-2030 [Accessed: July 2020]

Villarini, G., Smith, J. A., Baeck, M. L., Marchok, T., and Vecchi, G. A. (2011). Characterization of rainfall distribution and flooding associated with U.S. landfalling tropical cyclones: Analyses of Hurricanes Frances, Ivan, and Jeanne (2004). *Journal of Geophysical Research*, 116, D23116.

Weinkle, J., Landsea, C., Collins, D., Musulin, R., Crompton, R.P., Klotzbach, P.J., and Pielke Jr, R. (2018). Normalized hurricane damage in the continental United States 1900-2017. *Nature Sustainability*, 1, 808-813.

Wikipedia (2020). *Indemnity*. [Online] Available at: https://en.wikipedia.org/wiki/Indemnity [Accessed: July 2020]

Wikipedia (2020). *Insurance*. [Online] Available at: https://en.wikipedia.org/wiki/Insurance [Accessed: July 2020]

Wikipedia (2020). *The 21 first-level 2-digit region hydrologic unit boundaries*. [Online] Available at: https://en.wikipedia.org/wiki/Water_resource_region [Accessed: July 2020]

Winsemius, H. C., Jongman, B., Veldkamp, T.I.E. Hallegatte, S., Bangalore, M., and Ward, P.J. (2018). Disaster risk, climate change, and poverty: assessing the global exposure of poor people to floods and droughts. *Environment and Development Economics*, 23 (3), 328-348.

World Bank (2017). *World Bank Annual Report 2017*. Washington, DC: World Bank.

10. Appendix

10.1. Table A-1. The list of the 360 selected US-CAMELS gauge locations and related information.

HU: Region (2-digit USGS hydrologic unit code).
GAUGE ID: Catchment identifier (8-digit USGS hydrologic unit code).
GAUGE NAME: Gauge name, followed by the state.
LATITUDE: The latitude of the gauge location.
LONGITUDE: The longitude of the gauge location.
DRAINAGE AREA (km^2): The drainage (catchment) area.
GHK MODEL PARAMETERS: For more information regarding the applied GHK modeling strategies, see 4.4, 4.5 and 4.6.

	HU	GAUGE ID	GAUGE NAME	LATITUDE	LONGTITUDE	DRAINAGE AREA (km^2)	GHK MODEL PARAMETERS		
							H	q	λ
1	1	01078000	SMITH RIVER NEAR BRISTOL, NH	43,56646	-71,74786	222,46	0,46	17,42	1,06
2	1	01162500	PRIEST BROOK NEAR WINCHENDON, MA	42,68259	-72,11508	49,71	0,49	17,51	1,06
3	2	01411300	TUCKAHOE RIVER AT HEAD OF RIVER NJ	39,30694	-74,82056	79,32	0,43	52,23	1,03
4	2	01440000	FLAT BROOK NEAR FLATBROOKVILLE NJ	41,10667	-74,95222	167,72	0,60	5,57	1,15
5	2	01440400	BRODHEAD CREEK NEAR ANALOMINK, PA	41,08481	-75,21462	175,21	0,60	5,71	1,14
6	2	01451800	JORDAN CREEK NEAR SCHNECKSVILLE, PA	40,66176	-75,62685	135,84	0,55	5,30	1,17
7	2	01466500	MCDONALDS BRANCH IN LEBANON STATE FOREST NJ	39,885	-74,50528	5,38	0,53	46,34	1,03
8	2	01484100	BEAVERDAM BRANCH AT HOUSTON, DE	38,90578	-75,51275	9,00	0,70	2,57	1,23
9	2	01485500	NASSAWANGO CREEK NEAR SNOW HILL, MD	38,22892	-75,47144	141,54	0,56	10,83	1,08
10	2	01486000	MANOKIN BRANCH NEAR PRINCESS ANNE, MD	38,21389	-75,67139	11,18	0,64	3,24	1,22
11	2	01487000	NANTICOKE RIVER NEAR BRIDGEVILLE, DE	38,72833	-75,56186	187,41	0,40	76,04	1,02
12	2	01491000	CHOPTANK RIVER NEAR GREENSBORO, MD	38,99719	-75,78581	292,04	0,67	3,12	1,21
13	2	01510000	OTSELIC RIVER AT CINCINNATUS NY	42,54118	-75,89964	382,99	0,54	8,85	1,11
14	2	01516500	COREY CREEK NEAR MAINESBURG, PA	41,79091	-77,01469	31,48	0,65	1,00	1,63
15	2	01539000	FISHING CREEK NEAR BLOOMSBURG, PA	41,07814	-76,43106	701,78	0,57	5,34	1,16
16	2	01542810	WALDY RUN NEAR EMPORIUM, PA	41,57895	-78,29251	13,62	0,57	4,10	1,21
17	2	01543000	DRIFTWOOD BR SINNEMAHONING CR AT STERLING RUN, PA	41,4134	-78,19695	705,50	0,55	7,24	1,13
18	2	01543500	SINNEMAHONING CREEK AT SINNEMAHONING, PA	41,31728	-78,10306	1778,26	0,53	9,79	1,10
19	2	01545600	YOUNG WOMANS CREEK NEAR RENOVO, PA	41,38951	-77,69082	120,00	0,53	12,92	1,08
20	2	01547700	MARSH CREEK AT BLANCHARD, PA	41,05951	-77,60583	113,54	0,61	4,36	1,18
21	2	01549500	BLOCKHOUSE CREEK NEAR ENGLISH CENTER, PA	41,47368	-77,23081	97,49	0,59	5,44	1,15
22	2	01550000	LYCOMING CREEK NEAR TROUT RUN, PA	41,41841	-77,03275	452,68	0,58	5,44	1,16
23	2	01552000	LOYALSOCK CREEK AT LOYALSOCKVILLE, PA	41,32508	-76,91246	1129,49	0,57	4,80	1,18
24	2	01552500	MUNCY CREEK NEAR SONESTOWN, PA	41,35702	-76,53467	60,64	0,59	3,05	1,26
25	2	01557500	BALD EAGLE CREEK AT TYRONE, PA	40,68367	-78,23362	115,30	0,52	10,93	1,09
26	2	01567500	BIXLER RUN NEAR LOYSVILLE, PA	40,37092	-77,40221	38,77	0,66	2,28	1,29
27	2	01568000	SHERMAN CREEK AT SHERMANS DALE, PA	40,32342	-77,16887	534,39	0,61	3,93	1,20
28	2	01580000	DEER CREEK AT ROCKS, MD	39,62997	-76,40331	244,37	0,76	1,00	1,43
29	2	01583500	WESTERN RUN AT WESTERN RUN, MD	39,51078	-76,6765	155,85	0,73	1,00	1,48
30	2	01591400	CATTAIL CREEK NEAR GLENWOOD, MD	39,25597	-77,05106	59,09	0,70	1,00	1,52
31	2	01605500	SOUTH BRANCH POTOMAC RIVER AT FRANKLIN, WV	38,63567	-79,33782	463,85	0,61	3,12	1,25
32	2	01606500	SO. BRANCH POTOMAC RIVER NR PETERSBURG, WV	38,99122	-79,17587	1684,55	0,64	3,56	1,20
33	2	01613050	TONOLOWAY CREEK NEAR NEEDMORE, PA	39,89842	-78,13223	27,90	0,61	5,04	1,16
34	2	01632000	N F SHENANDOAH RIVER AT COOTES STORE, VA	38,63706	-78,8528	543,36	0,57	3,19	1,27
35	2	01632900	SMITH CREEK NEAR NEW MARKET, VA	38,69345	-78,64279	245,05	0,75	0,83	1,51
36	2	01634500	CEDAR CREEK NEAR WINCHESTER, VA	39,08122	-78,32945	263,98	0,71	1,00	1,50
37	2	01638480	CATOCTIN CREEK AT TAYLORSTOWN, VA	39,25455	-77,57638	232,00	0,67	1,00	1,59
38	2	01639500	BIG PIPE CREEK AT BRUCEVILLE, MD	39,61236	-77,23744	267,18	0,70	1,00	1,52
39	2	01644000	GOOSE CREEK NEAR LEESBURG, VA	39,01955	-77,57749	859,17	0,72	1,40	1,37
40	2	01658500	S F QUANTICO CREEK NEAR INDEPENDENT HILL, VA	38,58734	-77,4286	19,34	0,61	1,00	1,72
41	2	01664000	RAPPAHANNOCK RIVER AT REMINGTON, VA	38,53068	-77,8136	1605,10	0,76	1,00	1,42
42	2	01666500	ROBINSON RIVER NEAR LOCUST DALE, VA	38,32513	-78,09556	463,15	0,68	1,00	1,56
43	2	01667500	RAPIDAN RIVER NEAR CULPEPER, VA	38,35041	-77,975	1209,75	0,73	1,00	1,48
44	2	01669000	PISCATAWAY CREEK NEAR TAPPAHANNOCK, VA	37,87708	-76,90052	71,84	0,71	3,08	1,19
45	2	02011400	JACKSON RIVER NEAR BACOVA, VA	38,04235	-79,88144	407,72	0,59	4,86	1,17
46	2	02011460	BACK CREEK NEAR SUNRISE, VA	38,2454	-79,76866	156,34	0,59	2,37	1,34
47	2	02013000	DUNLAP CREEK NEAR COVINGTON, VA	37,8029	-80,047	424,78	0,67	1,00	1,58
48	2	02014000	POTTS CREEK NEAR COVINGTON, VA	37,72901	-80,04228	396,78	0,59	4,59	1,18
49	2	02015700	BULLPASTURE RIVER AT WILLIAMSVILLE, VA	38,1954	-79,57032	285,41	0,54	3,59	1,25
50	2	02016000	COWPASTURE RIVER NEAR CLIFTON FORGE, VA	37,7918	-79,75949	1194,55	0,58	3,43	1,24

	HU	GAUGE ID	GAUGE NAME	LATITUDE	LONGTITUDE	DRAINAGE AREA (km²)	GHK MODEL PARAMETERS		
							H	q	λ
51	2	02017500	JOHNS CREEK AT NEW CASTLE, VA	37,50624	-80,10671	276,15	0,57	8,06	1,11
52	2	02018000	CRAIG CREEK AT PARR, VA	37,66596	-79,91144	852,34	0,59	4,82	1,17
53	2	02027000	TYE RIVER NEAR LOVINGSTON, VA	37,71542	-78,98169	240,77	0,66	3,68	1,19
54	2	02027500	PINEY RIVER AT PINEY RIVER, VA	37,70236	-79,02752	123,34	0,60	3,86	1,20
55	2	02028500	ROCKFISH RIVER NEAR GREENFIELD, VA	37,86959	-78,82335	245,45	0,68	3,45	1,19
56	2	02038850	HOLIDAY CREEK NEAR ANDERSONVILLE, VA	37,41542	-78,63584	22,00	0,67	1,00	1,59
57	3	02046000	STONY CREEK NEAR DINWIDDIE, VA	37,06709	-77,60249	288,52	0,74	1,33	1,36
58	3	02051000	NORTH MEHERRIN RIVER NEAR LUNENBURG, VA	36,99737	-78,34972	144,82	0,63	1,00	1,68
59	3	02051500	MEHERRIN RIVER NEAR LAWRENCEVILLE, VA	36,71681	-77,83166	1428,71	0,73	1,00	1,47
60	3	02053200	POTECASI CREEK NEAR UNION, NC	36,37083	-77,02556	583,66	0,60	13,73	1,07
61	3	02053800	S F ROANOKE RIVER NEAR SHAWSVILLE, VA	37,14013	-80,26643	280,72	0,74	1,00	1,46
62	3	02055100	TINKER CREEK NEAR DALEVILLE, VA	37,41763	-79,93532	30,54	0,64	1,00	1,64
63	3	02056900	BLACKWATER RIVER NEAR ROCKY MOUNT, VA	37,04514	-79,8442	298,11	0,70	1,00	1,52
64	3	02059500	GOOSE CREEK NEAR HUDDLESTON, VA	37,1732	-79,52031	485,41	0,73	0,50	1,84
65	3	02064000	FALLING RIVER NEAR NARUNA, VA	37,12681	-78,95974	427,77	0,67	1,00	1,59
66	3	02065500	CUB CREEK AT PHENIX, VA	37,07931	-78,76362	252,58	0,70	1,00	1,52
67	3	02069700	SOUTH MAYO RIVER NEAR NETTLERIDGE, VA	36,57097	-80,12949	221,13	0,76	1,00	1,42
68	3	02070000	NORTH MAYO RIVER NEAR SPENCER, VA	36,56819	-79,98726	270,61	0,71	1,00	1,51
69	3	02074500	SANDY RIVER NEAR DANVILLE, VA	36,61958	-79,50419	288,53	0,64	1,00	1,65
70	3	02081500	TAR RIVER NEAR TAR RIVER, NC	36,19417	-78,58306	428,37	0,66	1,00	1,61
71	3	02082950	LITTLE FISHING CREEK NEAR WHITE OAK, NC	36,18333	-77,87611	460,85	0,73	1,00	1,46
72	3	02092500	TRENT RIVER NEAR TRENTON, NC	35,06417	-77,46139	447,58	0,54	14,88	1,07
73	3	02102908	FLAT CREEK NEAR INVERNESS, NC	35,18278	-79,1775	19,73	0,77	1,00	1,41
74	3	02108000	NORTHEAST CAPE FEAR RIVER NEAR CHINQUAPIN, NC	34,82889	-77,83222	1569,48	0,54	22,23	1,05
75	3	02111180	ELK CREEK AT ELKVILLE, NC	36,07139	-81,40306	131,47	0,75	1,00	1,44
76	3	02111500	REDDIES RIVER AT NORTH WILKESBORO, NC	36,175	-81,16889	233,66	0,80	1,00	1,37
77	3	02112120	ROARING RIVER NEAR ROARING RIVER, NC	36,25028	-81,04444	322,27	0,77	1,00	1,40
78	3	02112360	MITCHELL RIVER NEAR STATE ROAD, NC	36,31139	-80,80722	205,30	0,79	1,00	1,39
79	3	02118500	HUNTING CREEK NEAR HARMONY, NC	36,00056	-80,74556	400,53	0,74	1,00	1,46
80	3	02125000	BIG BEAR CR NR RICHFIELD, NC	35,33472	-80,33556	144,55	0,70	1,00	1,52
81	3	02128000	LITTLE RIVER NEAR STAR, NC	35,38722	-79,83139	273,48	0,64	1,00	1,65
82	3	02137727	CATAWBA R NR PLEASANT GARDENS, NC	35,68583	-82,06028	329,54	0,76	1,00	1,43
83	3	02143000	HENRY FORK NEAR HENRY RIVER, NC	35,68444	-81,40333	216,67	0,75	1,00	1,44
84	3	02143040	JACOB FORK AT RAMSEY, NC	35,59069	-81,56704	66,48	0,73	1,00	1,47
85	3	02149000	COVE CREEK NEAR LAKE LURE, NC	35,42345	-82,1115	203,85	0,81	1,00	1,36
86	3	02152100	FIRST BROAD RIVER NEAR CASAR, NC	35,49306	-81,68222	154,99	0,73	1,00	1,47
87	3	02177000	CHATTOOGA RIVER NEAR CLAYTON, GA	34,81398	-83,30599	526,77	0,84	1,00	1,35
88	3	02178400	TALLULAH RIVER NEAR CLAYTON, GA	34,89037	-83,53044	151,28	0,84	1,00	1,35
89	3	02202600	BLACK CREEK NEAR BLITCHTON, GA	32,16798	-81,48817	590,47	0,73	6,38	1,11
90	3	02212600	FALLING CREEK NEAR JULIETTE, GA	33,09985	-83,72351	187,96	0,67	1,00	1,59
91	3	02221525	MURDER CREEK BELOW EATONTON, GA	33,25236	-83,48128	490,80	0,77	1,00	1,41
92	3	02231000	ST. MARYS RIVER NR MACCLENNY, FLA.	30,35885	-82,0815	1748,37	0,53	23,56	1,04
93	3	02235200	BLACKWATER CREEK NEAR CASSIA, FL	28,87453	-81,48938	297,70	0,55	75,89	1,03
94	3	02245500	SOUTH FORK BLACK CREEK NR PENNEY FARMS, FLA.	29,97941	-81,85204	348,43	0,72	1,00	1,50
95	3	02246000	NORTH FORK BLACK CREEK NR MIDDLEBURG, FLA.	30,1133	-81,90649	451,06	0,58	4,98	1,17
96	3	02296500	CHARLIE CREEK NEAR GARDNER FL	27,37504	-81,79647	886,40	0,49	52,71	1,03
97	3	02297155	HORSE CREEK NEAR MYAKKA HEAD FL	27,48726	-82,02342	93,60	0,59	8,33	1,10
98	3	02297310	HORSE CREEK NEAR ARCADIA FL	27,19949	-81,98842	528,38	0,53	27,18	1,04
99	3	02298123	PRAIRIE CREEK NEAR FORT OGDEN FL	27,052	-81,78453	540,65	0,50	50,32	1,03
100	3	02298608	MYAKKA RIVER AT MYAKKA CITY FL	27,34366	-82,15676	312,34	0,58	11,38	1,08
101	3	02299950	MANATEE RIVER NEAR MYAKKA HEAD FL	27,47365	-82,2112	174,27	0,61	3,87	1,20
102	3	02300700	BULLFROG CREEK NEAR WIMAUMA FL	27,79197	-82,35204	73,84	0,58	3,01	1,28
103	3	02314500	SUWANNEE RIVER AT US 441, AT FARGO, GA	30,68056	-82,56056	3322,15	0,60	62,07	1,04
104	3	02315500	SUWANNEE RIVER AT WHITE SPRINGS, FLA.	30,32578	-82,73818	6136,26	0,60	58,83	1,04
105	3	02324400	FENHOLLOWAY RIVER NEAR FOLEY, FLA.	30,09827	-83,47181	176,27	0,63	38,97	1,04
106	3	02327100	SOPCHOPPY RIVER NR SOPCHOPPY, FLA.	30,12937	-84,49435	271,12	0,55	12,31	1,08
107	3	02342933	SOUTH FORK COWIKEE CREEK NEAR BATESVILLE AL	32,01766	-85,29577	290,10	0,66	1,00	1,60
108	3	02349900	TURKEY CREEK AT BYROMVILLE, GA	32,19556	-83,90222	122,87	0,75	1,00	1,43
109	3	02361000	CHOCTAWHATCHEE RIVER NEAR NEWTON, AL.	31,34295	-85,61049	1781,61	0,66	4,52	1,15
110	3	02369800	BLACKWATER RIVER NEAR BRADLEY AL	31,02768	-86,70996	227,51	0,75	1,00	1,44
111	3	02371500	CONECUH RIVER AT BRANTLEY AL	31,5735	-86,25162	1292,79	0,53	21,37	1,05
112	3	02372250	PATSALIGA CREEK NEAR BRANTLEY AL	31,59627	-86,40552	1145,22	0,55	15,65	1,06
113	3	02374500	MURDER CREEK NEAR EVERGREEN AL	31,4185	-86,98664	445,72	0,75	1,73	1,28
114	3	02381600	FAUSETT CREEK NEAR TALKING ROCK, GA	34,57037	-84,46882	10,32	0,76	1,00	1,43
115	3	02395120	TWO RUN CREEK NEAR KINGSTON, GA	34,24278	-84,88972	84,01	0,72	1,00	1,48
116	3	02408540	HATCHET CREEK BELOW ROCKFORD AL	32,91679	-86,27025	682,41	0,73	1,00	1,48
117	3	02422500	MULBERRY CREEK AT JONES AL	32,58291	-86,9036	527,55	0,72	1,50	1,35
118	3	02430085	RED BUD CREEK NR MOORES MILL, MS	34,46667	-88,28361	40,56	0,71	0,79	1,63
119	3	02430615	MUD CREEK NR FAIRVIEW, MS	34,3925	-88,355	28,71	0,70	1,00	1,53
120	3	02450250	SIPSEY FORK NEAR GRAYSON AL	34,28538	-87,39891	231,83	0,62	1,71	1,42
121	3	02464000	NORTH RIVER NEAR SAMANTHA AL	33,47928	-87,59723	578,94	0,63	2,46	1,29

	HU	GAUGE ID	GAUGE NAME	LATITUDE	LONGTITUDE	DRAINAGE AREA (km²)	GHK MODEL PARAMETERS		
							H	q	λ
122	3	02465493	ELLIOTTS CREEK AT MOUNDVILLE AL	32,99735	-87,62223	83,42	0,73	1,08	1,44
123	3	02469800	SATILPA CREEK NEAR COFFEEVILLE AL	31,74432	-88,02251	422,68	0,65	2,48	1,28
124	3	02472000	LEAF RIVER NR COLLINS, MS	31,70694	-89,40694	1927,13	0,55	9,61	1,10
125	3	02472500	BOUIE CREEK NR HATTIESBURG, MS	31,42583	-89,41472	789,94	0,69	1,00	1,56
126	3	02479155	CYPRESS CREEK NR JANICE, MS	31,02528	-89,01667	137,28	0,63	0,97	1,68
127	3	02479300	RED CREEK AT VESTRY, MS	30,73611	-88,78111	1144,20	0,59	7,93	1,11
128	3	02479560	ESCATAWPA RIVER NEAR AGRICOLA MS	30,80908	-88,45863	1454,33	0,53	15,19	1,06
129	3	02481000	BILOXI RIVER AT WORTHAM, MS	30,55861	-89,12194	249,23	0,57	3,00	1,28
130	3	02481510	WOLF RIVER NR LANDON, MS	30,48361	-89,27444	800,66	0,60	4,32	1,19
131	5	03015500	BROKENSTRAW CREEK AT YOUNGSVILLE, PA	41,85256	-79,31727	784,85	0,50	7,88	1,13
132	5	03049000	BUFFALO CREEK NEAR FREEPORT, PA	40,7159	-79,69949	357,15	0,60	4,79	1,17
133	5	03049800	LITTLE PINE CREEK NEAR ETNA, PA	40,52035	-79,93811	14,89	0,63	2,04	1,35
134	5	03069500	CHEAT RIVER NEAR PARSONS, WV	39,12288	-79,68117	1856,85	0,58	3,98	1,21
135	5	03076600	BEAR CREEK AT FRIENDSVILLE, MD	39,65614	-79,39411	127,40	0,55	8,58	1,11
136	5	03078000	CASSELMAN RIVER AT GRANTSVILLE, MD	39,70219	-79,13639	161,48	0,43	10,86	1,11
137	5	03140000	MILL CREEK NEAR COSHOCTON OH	40,36285	-81,86236	70,35	0,66	2,16	1,30
138	5	03144000	WAKATOMIKA CREEK NEAR FRAZEYSBURG OH	40,13257	-82,14792	363,41	0,65	1,00	1,62
139	5	03159540	SHADE RIVER NEAR CHESTER OH	39,06369	-81,8818	400,85	0,70	1,00	1,52
140	5	03161000	SOUTH FORK NEW RIVER NEAR JEFFERSON, NC	36,39333	-81,40694	527,86	0,77	1,00	1,40
141	5	03164000	NEW RIVER NEAR GALAX, VA	36,64735	-80,97897	2952,74	0,74	2,47	1,22
142	5	03165000	CHESTNUT CREEK AT GALAX, VA	36,64596	-80,91924	102,00	0,71	1,00	1,51
143	5	03170000	LITTLE RIVER AT GRAYSONTOWN, VA	37,03763	-80,55672	794,64	0,76	1,00	1,43
144	5	03173000	WALKER CREEK AT BANE, VA	37,26818	-80,70951	773,32	0,64	3,81	1,19
145	5	03182500	GREENBRIER RIVER AT BUCKEYE, WV	38,18595	-80,13062	1364,97	0,60	3,44	1,23
146	5	03186500	WILLIAMS RIVER AT DYER, WV	38,379	-80,48397	329,68	0,54	4,92	1,19
147	5	03213700	TUG FORK AT WILLIAMSON, WV	37,67316	-82,28014	2425,06	0,62	5,32	1,14
148	5	03237500	OHIO BRUSH CREEK NEAR WEST UNION OH	38,80368	-83,42102	1003,21	0,58	1,55	1,53
149	5	03241500	MASSIES CREEK AT WILBERFORCE OH	39,72284	-83,88271	170,91	0,61	3,99	1,20
150	5	03281500	SOUTH FORK KENTUCKY RIVER AT BOONEVILLE, KY	37,47981	-83,67519	1838,22	0,63	3,34	1,22
151	5	03285000	DIX RIVER NEAR DANVILLE, KY	37,64202	-84,66078	822,34	0,69	1,00	1,54
152	5	03300400	BEECH FORK AT MAUD, KY	37,83284	-85,29607	1128,58	0,68	1,00	1,56
153	5	03346000	NORTH FORK EMBARRAS RIVER NEAR OBLONG, IL	39,01004	-87,94559	814,69	0,48	4,82	1,22
154	5	03366500	MUSCATATUCK RIVER NEAR DEPUTY, IN	38,80422	-85,67386	755,22	0,64	1,00	1,66
155	5	03384450	LUSK CREEK NEAR EDDYVILLE, IL	37,47227	-88,54727	111,13	0,61	0,81	1,87
156	6	03439000	FRENCH BROAD RIVER AT ROSMAN, NC	35,14333	-82,82472	178,67	0,80	1,00	1,37
157	6	03455500	W F PIGEON R ABOVE LAKE LOGAN NR HAZELWOOD, NC	35,39611	-82,9375	72,95	0,73	1,00	1,48
158	6	03456500	EAST FORK PIGEON RIVER NEAR CANTON, NC	35,46167	-82,86972	133,43	0,73	1,54	1,32
159	6	03460000	CATALOOCHEE CREEK NEAR CATALOOCHEE, NC	35,6675	-83,07361	127,02	0,71	3,72	1,17
160	6	03463300	SOUTH TOE RIVER NEAR CELO, NC	35,83151	-82,18429	112,16	0,72	1,00	1,49
161	6	03471500	S F HOLSTON RIVER AT RIVERSIDE NR CHILHOWIE, VA	36,76039	-81,63123	198,16	0,63	4,88	1,15
162	6	03473000	S F HOLSTON RIVER NEAR DAMASCUS, VA	36,65178	-81,84401	784,81	0,67	4,56	1,15
163	6	03479000	WATAUGA RIVER NEAR SUGAR GROVE, NC	36,23917	-81,82222	235,60	0,67	1,18	1,51
164	6	03488000	N F HOLSTON RIVER NEAR SALTVILLE, VA	36,89678	-81,74623	578,21	0,67	2,81	1,23
165	6	03498500	LITTLE RIVER NEAR MARYVILLE, TN	35,78562	-83,8846	696,76	0,73	1,00	1,47
166	6	03500000	LITTLE TENNESSEE RIVER NEAR PRENTISS, NC	35,15	-83,37972	361,14	0,79	3,08	1,18
167	6	03500240	CARTOOGECHAYE CREEK NEAR FRANKLIN, NC	35,15889	-83,39417	145,86	0,79	1,91	1,24
168	6	03504000	NANTAHALA RIVER NEAR RAINBOW SPRINGS, NC	35,1275	-83,61861	134,52	0,71	7,35	1,10
169	6	03574500	PAINT ROCK RIVER NEAR WOODVILLE AL	34,62425	-86,30637	813,80	0,53	4,82	1,19
170	6	03592718	LITTLE YELLOW CREEK EAST NR BURNSVILLE, MS	34,83556	-88,28806	66,99	0,52	1,00	1,93
171	6	03604000	BUFFALO RIVER NEAR FLAT WOODS, TN	35,49591	-87,8328	1163,03	0,67	1,00	1,58
172	4	04045500	TAHQUAMENON RIVER NEAR PARADISE, MI	46,57502	-85,26955	1909,59	0,68	1,00	1,56
173	4	04105700	AUGUSTA CREEK NEAR AUGUSTA, MI	42,35337	-85,35389	98,04	0,72	9,94	1,08
174	4	04122200	WHITE RIVER NEAR WHITEHALL, MI	43,46418	-86,23257	1048,28	0,61	29,25	1,04
175	4	04122500	PERE MARQUETTE RIVER AT SCOTTVILLE, MI	43,94501	-86,27869	1769,19	0,54	42,16	1,03
176	4	04124000	MANISTEE RIVER NEAR SHERMAN, MI	44,43639	-85,69868	2243,63	0,48	42,40	1,03
177	4	04127997	STURGEON RIVER AT WOLVERINE, MI	45,27446	-84,60003	487,16	0,79	4,32	1,15
178	4	04161580	STONY CREEK NEAR ROMEO, MI	42,80086	-83,09021	64,31	0,59	1,00	1,77
179	4	04185000	TIFFIN RIVER AT STRYKER OH	41,5045	-84,42967	1064,00	0,68	1,00	1,56
180	4	04196800	TYMOCHTEE CREEK AT CRAWFORD OH	40,92283	-83,34881	608,07	0,60	5,49	1,15
181	4	04197100	HONEY CREEK AT MELMORE OH	41,02228	-83,10964	388,21	0,56	5,65	1,16
182	9	05129115	VERMILION RIVER NR CRANE LAKE, MN	48,26463	-92,56599	2357,59	0,79	1,00	1,38
183	7	05444000	ELKHORN CREEK NEAR PENROSE, IL	41,90281	-89,69622	376,35	0,77	1,00	1,41
184	7	05458000	LITTLE CEDAR RIVER NEAR IONIA, IA	43,03328	-92,50354	776,76	0,67	5,57	1,13
185	7	05495000	FOX RIVER AT WAYLAND, MO	40,39241	-91,59788	1027,55	0,69	1,94	1,30
186	7	05501000	NORTH RIVER AT PALMYRA, MO	39,81799	-91,5177	922,83	0,65	1,21	1,52
187	7	05507600	LICK CREEK AT PERRY, MO	39,43143	-91,67627	275,38	0,60	1,00	1,75
188	7	05508805	SPENCER CR. BL PLUM CR. NR FRANKFORD, MO	39,52046	-91,34275	536,40	0,63	0,93	1,73
189	7	05514500	CUIVRE RIVER NEAR TROY, MO	39,0088	-90,97754	2407,41	0,66	1,00	1,61
190	7	05525500	SUGAR CREEK AT MILFORD, IL	40,63004	-87,72392	1158,76	0,58	6,86	1,12
191	7	05556500	BIG BUREAU CREEK AT PRINCETON, IL	41,36587	-89,49843	505,09	0,69	2,02	1,29
192	7	05585000	LA MOINE RIVER AT RIPLEY, IL	40,02477	-90,63179	3354,61	0,62	12,44	1,07

	HU	GAUGE ID	GAUGE NAME	LATITUDE	LONGTITUDE	DRAINAGE AREA (km²)	GHK MODEL PARAMETERS		
							H	q	λ
193	10	06154410	LITTLE PEOPLES CREEK NEAR HAYS MT	47,9661	-108,66071	33,41	0,97	15,68	3,58
194	10	06191500	YELLOWSTONE RIVER AT CORWIN SPRINGS MT	45,11188	-110,79438	6783,59	0,75	1,00	1,43
195	10	06224000	BULL LAKE CREEK ABOVE BULL LAKE, WY	43,1769	-109,20291	484,86	0,70	1,00	1,52
196	10	06278300	SHELL CREEK ABOVE SHELL CREEK RESERVOIR, WY	44,50802	-107,40369	58,85	0,63	1,00	1,68
197	10	06280300	SOUTH FORK SHOSHONE RIVER NEAR VALLEY, WY	44,20828	-109,55489	793,95	0,72	1,00	1,49
198	10	06289000	LITTLE BIGHORN RIVER AT STATE LINE NR WYOLA MT	45,00691	-107,61509	471,27	0,77	1,00	1,40
199	10	06406000	BATTLE CR AT HERMOSA SD	43,82805	-103,19602	436,34	0,90	1,00	1,44
200	10	06431500	SPEARFISH CR AT SPEARFISH SD	44,48248	-103,86159	427,10	0,80	9,21	1,12
201	10	06622700	NORTH BRUSH CREEK NEAR SARATOGA, WY	41,37024	-106,52058	98,70	0,72	1,00	1,50
202	10	06853800	WHITE ROCK C NR BURR OAK, KS	39,89925	-98,25043	589,43	0,70	1,00	1,52
203	10	06876700	SALT C NR ADA, KS	39,13907	-97,83696	1056,44	0,65	9,61	1,08
204	10	06878000	CHAPMAN C NR CHAPMAN, KS	39,03111	-97,04029	776,42	0,71	0,87	1,57
205	10	06885500	BLACK VERMILLION R NR FRANKFORT, KS	39,68192	-96,44289	1062,87	0,69	1,00	1,55
206	10	06888500	MILL C NR PAXICO, KS	39,06265	-96,15025	842,35	0,68	1,00	1,56
207	10	06889200	SOLDIER C NR DELIA, KS	39,23833	-95,8886	385,66	0,66	1,00	1,61
208	10	06889500	SOLDIER C NR TOPEKA, KS	39,09938	-95,72495	748,58	0,64	1,00	1,65
209	10	06892000	STRANGER C NR TONGANOXIE, KS	39,11633	-95,01088	1092,72	0,67	2,49	1,25
210	10	06910800	MARAIS DES CYGNES R NR READING, KS	38,56701	-95,96163	444,60	0,63	1,00	1,67
211	10	06917000	L OSAGE R AT FULTON, KS	38,01897	-94,71367	765,78	0,60	2,07	1,38
212	10	06918460	TURNBACK CREEK ABOVE GREENFIELD, MO	37,40237	-93,80203	650,69	0,70	1,70	1,33
213	10	06919500	CEDAR CREEK NEAR PLEASANT VIEW, MO	37,8342	-93,87549	1069,42	0,59	3,89	1,21
214	10	06921070	POMME DE TERRE RIVER NEAR POLK, MO	37,68265	-93,37033	712,59	0,66	1,00	1,61
215	10	06921200	LINDLEY CREEK NEAR POLK, MO	37,75047	-93,26618	323,62	0,62	1,00	1,71
216	7	07014500	MERAMEC RIVER NEAR SULLIVAN, MO	38,15853	-91,10846	3845,50	0,69	2,69	1,23
217	11	07056000	BUFFALO RIVER NEAR ST. JOE, AR	35,98318	-92,74755	2149,36	0,57	2,71	1,31
218	11	07057500	NORTH FORK RIVER NEAR TECUMSEH, MO	36,62303	-92,24813	1456,44	0,73	1,00	1,47
219	11	07060710	NORTH SYLAMORE CREEK NEAR FIFTY SIX, AR	35,99535	-92,21265	150,03	0,55	1,00	1,87
220	11	07066000	JACKS FORK AT EMINENCE, MO	37,15408	-91,35816	1053,50	0,69	1,60	1,36
221	11	07067000	CURRENT RIVER AT VAN BUREN, MO	36,99138	-91,01351	4349,01	0,79	1,00	1,38
222	11	07068000	CURRENT RIVER AT DONIPHAN, MO	36,622	-90,84762	5318,59	0,68	4,79	1,14
223	11	07071500	ELEVEN POINT RIVER NEAR BARDLEY, MO	36,64868	-91,20082	2023,77	0,81	1,00	1,36
224	11	07145700	SLATE C AT WELLINGTON, KS	37,24955	-97,4039	399,81	0,63	1,00	1,67
225	11	07148400	SALT FORK ARKANSAS RIVER NR ALVA, OK	36,81503	-98,64814	2544,50	0,78	1,00	1,40
226	11	07149000	MEDICINE LODGE R NR KIOWA, KS	37,0392	-98,47021	2291,20	0,81	0,49	1,59
227	11	07151500	CHIKASKIA R NR CORBIN, KS	37,12891	-97,60144	2109,00	0,68	1,00	1,57
228	11	07167500	OTTER C AT CLIMAX, KS	37,70821	-96,2236	319,62	0,62	1,00	1,70
229	11	07180500	CEDAR C NR CEDAR POINT, KS	38,19645	-96,82458	275,51	0,63	1,00	1,68
230	11	07195800	FLINT CREEK AT SPRINGTOWN, AR	36,25548	-94,43394	38,23	0,68	1,57	1,38
231	11	07196900	BARON FORK AT DUTCH MILLS, AR	35,88008	-94,48661	106,35	0,63	1,00	1,67
232	11	07197000	BARON FORK AT ELDON, OK	35,9212	-94,83856	808,45	0,69	1,00	1,55
233	8	07291000	HOMOCHITTO RIVER AT EDDICETON, MS	31,50306	-90,7775	479,30	0,59	1,00	1,76
234	8	07292500	HOMOCHITTO RIVER AT ROSETTA, MS	31,32472	-91,10944	2073,09	0,64	1,16	1,58
235	11	07299670	GROESBECK CK AT SH 6 NR QUANAH, TX	34,35452	-99,74037	828,36	0,53	1,00	1,92
236	11	07301410	SWEETWATER CK NR KELTON, TX	35,4731	-100,12095	769,67	0,78	1,00	1,39
237	11	07340300	COSSATOT RIVER NEAR VANDERVOORT, AR	34,37955	-94,23576	230,43	0,61	1,00	1,72
238	11	07346045	BLACK CYPRESS BAYOU AT JEFFERSON, TX	32,77792	-94,35742	959,89	0,60	18,99	1,05
239	8	07362100	SMACKOVER CREEK NEAR SMACKOVER, ARK.	33,37596	-92,77711	996,08	0,66	3,60	1,19
240	8	07373000	BIG CREEK AT POLLOCK, LA	31,53629	-92,40847	131,18	0,61	1,00	1,73
241	8	07375000	TCHEFUNCTA RIVER NEAR FOLSOM, LA	30,61602	-90,24869	249,35	0,66	1,30	1,47
242	8	07376000	TICKFAW RIVER AT HOLDEN, LA	30,5038	-90,67732	651,61	0,61	6,73	1,12
243	8	08013000	CALCASIEU RIVER NR GLENMORA, LA	30,99602	-92,67375	1293,69	0,54	11,84	1,08
244	8	08014500	WHISKEY CHITTO CREEK NEAR OBERLIN, LA	30,69875	-92,89319	1305,15	0,60	6,37	1,13
245	12	08029500	BIG COW CK NR NEWTON, TX	30,81889	-93,78556	333,40	0,66	0,86	1,69
246	12	08066200	LONG KING CK AT LIVINGSTON, TX	30,71631	-94,95882	364,12	0,56	1,00	1,84
247	12	08066300	MENARD CK NR RYE, TX	30,48139	-94,77972	383,57	0,63	4,14	1,18
248	12	08070000	E FK SAN JACINTO RV NR CLEVELAND, TX	30,3366	-95,1041	841,06	0,63	2,63	1,27
249	12	08104900	S FK SAN GABRIEL RV AT GEORGETOWN, TX	30,62575	-97,69112	342,62	0,64	1,00	1,66
250	12	08150800	BEAVER CK NR MASON, TX	30,64352	-99,09588	557,52	0,55	1,00	1,87
251	12	08164000	LAVACA RV NR EDNA, TX	28,95998	-96,68637	2124,00	0,64	1,81	1,38
252	12	08164300	NAVIDAD RV NR HALLETTSVILLE, TX	29,46691	-96,81276	861,54	0,61	1,00	1,71
253	12	08165300	N FK GUADALUPE RV NR HUNT, TX	30,0641	-99,38699	435,99	0,47	1,00	2,10
254	12	08175000	SANDIES CK NR WESTHOFF, TX	29,21525	-97,44943	1421,66	0,61	1,35	1,55
255	12	08189500	MISSION RV AT REFUGIO, TX	28,29195	-97,27916	1808,29	0,56	3,26	1,27
256	12	08190000	NUECES RV AT LAGUNA, TX	29,42857	-99,99729	1961,43	0,66	1,00	1,60
257	12	08195000	FRIO RV AT CONCAN, TX	29,48857	-99,70478	1028,29	0,69	1,00	1,55
258	12	08196000	DRY FRIO RV NR REAGAN WELLS, TX	29,50468	-99,78145	326,98	0,63	1,00	1,67
259	12	08198500	SABINAL RV AT SABINAL, TX	29,31431	-99,48047	624,35	0,65	1,00	1,63
260	13	08324000	JEMEZ RIVER NEAR JEMEZ, NM	35,66169	-106,74337	1207,95	0,82	1,00	1,35
261	14	09081600	CRYSTAL RIVER AB AVALANCHE C, NEAR REDSTONE, CO.	39,23221	-107,22727	432,89	0,76	1,00	1,42
262	14	09306242	CORRAL GULCH NEAR RANGELY, CO.	39,92025	-108,47287	81,99	0,74	24,10	1,08
263	15	09404450	EAST FORK VIRGIN RIVER NEAR GLENDALE, UT	37,33943	-112,60438	192,96	0,63	69,19	1,04

	HU	GAUGE ID	GAUGE NAME	LATITUDE	LONGTITUDE	DRAINAGE AREA (km²)	GHK MODEL PARAMETERS		
							H	q	λ
264	15	09430500	GILA RIVER NEAR GILA, NM	33,06118	-108,53727	4804,93	0,74	1,00	1,45
265	15	09480000	SANTA CRUZ RIVER NEAR LOCHIEL, AZ.	31,35538	-110,58953	213,38	0,60	1,00	1,73
266	15	09492400	EAST FORK WHITE RIVER NEAR FORT APACHE, AZ.	33,82227	-109,81454	128,97	0,60	43,35	1,04
267	15	09494000	WHITE RIVER NEAR FORT APACHE, AZ.	33,73644	-110,16677	1628,23	0,60	49,34	1,03
268	15	09497800	CIBECUE CREEK NEAR CHYSOTILE, AZ.	33,84311	-110,55761	750,86	0,76	1,00	1,42
269	15	09497980	CHERRY CREEK NEAR GLOBE, AZ.	33,82783	-110,85623	516,80	0,72	1,30	1,39
270	15	09505800	WEST CLEAR CREEK NEAR CAMP VERDE, AZ.	34,53864	-111,69404	614,94	0,68	1,46	1,40
271	16	10172700	VERNON CREEK NEAR VERNON, UT	39,97939	-112,38023	69,93	0,95	24,40	2,32
272	16	10172800	SOUTH WILLOW CREEK NEAR GRANTSVILLE, UT	40,49633	-112,5744	11,07	0,60	73,71	1,04
273	16	10244950	STEPTOE C NR ELY, NV	39,20154	-114,68916	28,20	0,83	49,21	1,23
274	18	10259000	ANDREAS C NR PALM SPRINGS CA	33,76002	-116,55002	22,46	0,80	4,89	1,14
275	16	10329500	MARTIN C NR PARADISE VALLEY, NV	41,53462	-117,41791	454,52	0,51	49,66	1,03
276	16	10343500	SAGEHEN C NR TRUCKEE CA	39,43157	-120,23798	27,60	0,60	49,71	1,04
277	17	10396000	DONNER UND BLITZEN RIVER NR FRENCHGLEN OR	42,79083	-118,8675	528,89	0,60	31,75	1,04
278	18	11098000	ARROYO SECO NR PASADENA CA	34,22223	-118,17757	41,57	0,69	3,46	1,18
279	18	11141280	LOPEZ C NR ARROYO GRANDE CA	35,23553	-120,47239	54,01	0,76	2,85	1,19
280	18	11143000	BIG SUR R NR BIG SUR CA	36,2458	-121,77329	120,61	0,55	30,93	1,04
281	18	11151300	SAN LORENZO C BL BITTERWATER C NR KING CITY CA	36,26802	-121,0663	608,23	0,68	1,00	1,57
282	18	11162500	PESCADERO C NR PESCADERO CA	37,26078	-122,32886	118,87	0,77	1,00	1,41
283	18	11264500	MERCED R A HAPPY ISLES BRIDGE NR YOSEMITE CA	37,73159	-119,55877	467,98	0,80	1,00	1,37
284	18	11266500	MERCED R A POHONO BRIDGE NR YOSEMITE CA	37,71687	-119,66628	833,08	0,82	1,00	1,36
285	18	11381500	MILL C NR LOS MOLINOS CA	40,0546	-122,02415	337,63	0,73	3,72	1,16
286	18	11383500	DEER C NR VINA CA	40,01405	-121,94832	539,78	0,71	5,03	1,13
287	18	11451100	NF CACHE C A HOUGH SPRING NR CLEARLAKE OAKS CA	39,16545	-122,61999	155,09	0,66	6,11	1,12
288	18	11468500	NOYO R NR FORT BRAGG CA	39,42822	-123,73779	273,51	0,61	6,81	1,12
289	18	11473900	MF EEL R NR DOS RIOS CA	39,70627	-123,32529	1925,01	0,68	6,54	1,11
290	18	11475560	ELDER C NR BRANSCOMB CA	39,7296	-123,64391	16,79	0,48	27,99	1,04
291	18	11476600	BULL C NR WEOTT CA	40,35125	-124,00393	71,85	0,62	11,82	1,07
292	18	11478500	VAN DUZEN R NR BRIDGEVILLE CA	40,48042	-123,89088	572,38	0,59	7,49	1,11
293	18	11481200	LITTLE R NR TRINIDAD CA	41,01096	-124,08173	105,40	0,64	5,54	1,13
294	18	11482500	REDWOOD C A ORICK CA	41,29929	-124,05118	718,04	0,52	22,55	1,05
295	18	11522500	SALMON R A SOMES BAR CA	41,37763	-123,47756	1943,12	0,67	19,54	1,05
296	18	11523200	TRINITY R AB COFFEE C NR TRINITY CTR CA	41,11126	-122,70558	382,94	0,79	1,00	1,38
297	18	11528700	SF TRINITY R BL HYAMPOM CA	40,64986	-123,4942	1980,08	0,62	16,98	1,06
298	18	11532500	SMITH R NR CRESCENT CITY CA	41,7915	-124,07619	1577,96	0,60	8,36	1,10
299	17	12010000	NASELLE RIVER NEAR NASELLE, WA	46,37399	-123,74348	142,18	0,48	12,18	1,09
300	17	12013500	WILLAPA RIVER NEAR WILLAPA, WA	46,65093	-123,65266	337,69	0,46	17,81	1,06
301	17	12020000	CHEHALIS RIVER NEAR DOTY, WA	46,61732	-123,27764	294,22	0,53	9,14	1,10
302	17	12025000	NEWAUKUM RIVER NEAR CHEHALIS, WA	46,6201	-122,94513	404,90	0,59	10,50	1,08
303	17	12025700	SKOOKUMCHUCK RIVER NEAR VAIL, WA	46,7726	-122,59401	103,30	0,53	10,57	1,09
304	17	12035000	SATSOP RIVER NEAR SATSOP, WA	47,00065	-123,49488	769,90	0,53	11,60	1,08
305	17	12040500	QUEETS RIVER NEAR CLEARWATER, WA	47,53786	-124,31575	1153,44	0,53	6,74	1,14
306	17	12041200	HOH RIVER AT US HIGHWAY 101 NEAR FORKS, WA	47,80675	-124,25103	655,80	0,58	5,65	1,15
307	17	12048000	DUNGENESS RIVER NEAR SEQUIM, WA	48,01426	-123,13268	405,01	0,68	7,96	1,09
308	17	12054000	DUCKABUSH RIVER NEAR BRINNON, WA	47,68398	-123,01155	171,68	0,63	6,01	1,13
309	17	12056500	NF SKOKOMISH R BL STAIRCASE RPDS NR HOODSPORT, WA	47,51426	-123,32989	146,97	0,57	6,63	1,13
310	17	12073500	HUGE CREEK NEAR WAUNA, WA	47,38926	-122,69902	15,67	0,69	4,40	1,15
311	17	12082500	NISQUALLY RIVER NEAR NATIONAL, WA	46,75261	-122,08372	350,00	0,60	6,64	1,12
312	17	12092000	PUYALLUP RIVER NEAR ELECTRON, WA	46,90372	-122,03511	240,88	0,56	6,79	1,13
313	17	12114500	CEDAR RIVER BELOW BEAR CREEK NEAR CEDAR FALLS, WA	47,34205	-121,54899	66,56	0,44	20,06	1,06
314	17	12115000	CEDAR RIVER NEAR CEDAR FALLS, WA	47,37011	-121,6251	102,80	0,54	9,16	1,10
315	17	12115500	REX RIVER NEAR CEDAR FALLS, WA	47,35066	-121,66316	34,72	0,52	6,56	1,15
316	17	12117000	TAYLOR CREEK NEAR SELLECK, WA	47,38649	-121,84622	44,85	0,27	59,60	1,03
317	17	12141300	MIDDLE FORK SNOQUALMIE RIVER NEAR TANNER, WA	47,48594	-121,64788	401,51	0,72	1,00	1,48
318	17	12145500	RAGING RIVER NEAR FALL CITY, WA	47,53982	-121,909	78,98	0,54	8,44	1,11
319	17	12147500	NORTH FORK TOLT RIVER NEAR CARNATION, WA	47,71232	-121,78873	102,89	0,63	3,58	1,20
320	17	12147600	SOUTH FORK TOLT RIVER NEAR INDEX, WA	47,70677	-121,60011	14,13	0,58	2,60	1,32
321	17	12167000	NF STILLAGUAMISH RIVER NEAR ARLINGTON, WA	48,26149	-122,04764	683,75	0,58	5,29	1,16
322	17	12175500	THUNDER CREEK NEAR NEWHALEM, WA	48,67263	-121,0729	273,82	0,72	1,00	1,49
323	17	12178100	NEWHALEM CREEK NEAR NEWHALEM, WA	48,65596	-121,23846	69,68	0,61	7,53	1,11
324	17	12186000	SAUK RIVER AB WHITECHUCK RIVER NEAR DARRINGTON, WA	48,16872	-121,47067	398,44	0,60	7,27	1,11
325	17	12189500	SAUK RIVER NEAR SAUK, WA	48,42456	-121,56846	1855,32	0,62	7,70	1,10
326	17	12358500	MIDDLE FORK FLATHEAD RIVER NR WEST GLACIER MT	48,49524	-114,01012	2939,19	0,71	1,00	1,50
327	17	12390700	PROSPECT CREEK AT THOMPSON FALLS MT	47,58605	-115,35515	470,23	0,78	1,00	1,39
328	17	12451000	STEHEKIN RIVER AT STEHEKIN, WA	48,32958	-120,69177	830,60	0,71	1,00	1,52
329	17	12488500	AMERICAN RIVER NEAR NILE, WA	46,97762	-121,1687	205,17	0,76	1,00	1,42
330	17	13018300	CACHE CREEK NEAR JACKSON, WY	43,45215	-110,70409	27,90	0,80	1,00	1,37
331	17	13083000	TRAPPER CREEK NR OAKLEY ID	42,16583	-113,98361	133,18	0,44	140,05	1,02
332	17	13313000	JOHNSON CREEK AT YELLOW PINE ID	44,96167	-115,5	561,94	0,75	1,00	1,43
333	17	13331500	MINAM RIVER NEAR MINAM, OR	45,61987	-117,72659	618,93	0,74	1,00	1,46
334	17	13337000	LOCHSA RIVER NR LOWELL ID	46,15083	-115,58722	3053,42	0,73	1,00	1,46

	HU	GAUGE ID	GAUGE NAME	LATITUDE	LONGTITUDE	DRAINAGE AREA (km²)	GHK MODEL PARAMETERS		
							H	q	λ
335	17	13338500	SF CLEARWATER RIVER AT STITES ID	46,08639	-115,97667	3027,00	0,79	1,00	1,38
336	17	13340000	CLEARWATER RIVER AT OROFINO ID	46,47833	-116,2575	14268,92	0,75	1,00	1,44
337	17	13340600	NF CLEARWATER RIVER NR CANYON RANGER STATION ID	46,84047	-115,6207	3354,62	0,76	1,00	1,42
338	17	14020000	UMATILLA RIVER ABOVE MEACHAM CREEK, NR GIBBON, OR	45,71958	-118,32329	341,43	0,24	67,96	1,02
339	17	14137000	SANDY RIVER NEAR MARMOT, OR	45,39956	-122,13731	674,24	0,69	5,48	1,12
340	17	14138800	BLAZED ALDER CREEK NEAR RHODODENDRON, OREG.	45,45262	-121,89147	21,29	0,51	6,91	1,14
341	17	14138870	FIR CREEK NEAR BRIGHTWOOD, OR	45,48012	-122,02564	14,02	0,51	8,58	1,12
342	17	14138900	NORTH FORK BULL RUN RIVER NEAR MULTNOMAH FALLS, OR	45,49429	-122,03592	21,68	0,53	7,22	1,13
343	17	14139800	SOUTH FORK BULL RUN RIVER NEAR BULL RUN, OR	45,44456	-122,10953	40,70	0,58	6,61	1,13
344	17	14141500	LITTLE SANDY RIVER NEAR BULL RUN, OREG.	45,4154	-122,17148	59,87	0,58	7,48	1,11
345	17	14154500	ROW RIVER ABOVE PITCHER CREEK NEAR, DORENA, OREG	43,73596	-122,8734	546,81	0,56	8,53	1,11
346	17	14158500	MCKENZIE RIVER AT OUTLET OF CLEAR LAKE, OR	44,36095	-121,99562	237,08	0,42	129,58	1,02
347	17	14158790	SMITH R AB SMITH R RES NR BELKNAP SPRGS,OREG.	44,33457	-122,04701	40,57	0,51	13,22	1,08
348	17	14166500	LONG TOM RIVER NEAR NOTI, OREG.	44,04984	-123,42621	226,52	0,39	38,80	1,03
349	17	14182500	LITTLE NORTH SANTIAM RIVER NEAR MEHAMA, OR	44,79151	-122,57897	286,85	0,53	7,82	1,12
350	17	14185000	SOUTH SANTIAM RIVER BELOW CASCADIA, OR	44,39179	-122,49758	458,18	0,52	11,04	1,09
351	17	14185900	QUARTZVILLE CREEK NEAR CASCADIA, OREG.	44,54012	-122,43591	258,20	0,49	10,36	1,10
352	17	14222500	EAST FORK LEWIS RIVER NEAR HEISSON, WA	45,83678	-122,46621	323,89	0,45	19,37	1,06
353	17	14236200	TILTON RIVER AB BEAR CANYON CREEK NEAR CINEBAR, WA	46,59538	-122,45956	360,99	0,50	13,02	1,08
354	17	14301000	NEHALEM RIVER NEAR FOSS, OR	45,704	-123,7554	1743,54	0,30	50,54	1,03
355	17	14305500	SILETZ RIVER AT SILETZ, OR	44,71512	-123,88733	526,33	0,51	12,49	1,08
356	17	14306500	ALSEA RIVER NEAR TIDEWATER, OR	44,38595	-123,83178	857,16	0,38	38,59	1,03
357	17	14309500	WEST FORK COW CREEK NEAR GLENDALE, OR	42,804	-123,61091	224,92	0,54	10,77	1,09
358	17	14316700	STEAMBOAT CREEK NEAR GLIDE, OR	43,34984	-122,72894	587,90	0,57	8,90	1,10
359	17	14325000	SOUTH FORK COQUILLE RIVER AT POWERS, OR	42,8915	-124,07065	443,07	0,56	9,71	1,09
360	17	14400000	CHETCO RIVER NEAR BROOKINGS, OR	42,12344	-124,18731	702,63	0,44	26,19	1,04

10.2. Table A-2. The contribution of ERGO insurance company on the agricultural insurance loss, as an additional insurance coverage, in combination to ELGA's insurance coverage.

% OF LOSS	LOSS (€)	LIMIT OF 85% (€)	ELGA'S COMPENSATION (€)	COMPENSATION FROM ERGO (€)	% OF LOSS	LOSS (€)	LIMIT OF 85% (€)	ELGA'S COMPENSATION (€)	COMPENSATION FROM ERGO (€)
21	2100	1785	528	1257	61	6100	5185	4048	1137
22	2200	1870	616	1254	62	6200	5270	4136	1134
23	2300	1955	704	1251	63	6300	5355	4224	1131
24	2400	2040	792	1248	64	6400	5440	4312	1128
25	2500	2125	880	1245	65	6500	5525	4400	1125
26	2600	2210	968	1242	66	6600	5610	4488	1122
27	2700	2295	1056	1239	67	6700	5695	4576	1119
28	2800	2380	1144	1236	68	6800	5780	4664	1116
29	2900	2465	1232	1233	69	6900	5865	4752	1113
30	3000	2550	1320	1230	70	7000	5950	4840	1110
31	3100	2635	1408	1227	71	7100	6035	4928	1107
32	3200	2720	1496	1224	72	7200	6120	5016	1104
33	3300	2805	1584	1221	73	7300	6205	5104	1101
34	3400	2890	1672	1218	74	7400	6290	5192	1098
35	3500	2975	1760	1215	75	7500	6375	5280	1095
36	3600	3060	1848	1212	76	7600	6460	5368	1092
37	3700	3145	1936	1209	77	7700	6545	5456	1089
38	3800	3230	2024	1206	78	7800	6630	5544	1086
39	3900	3315	2112	1203	79	7900	6715	5632	1083
40	4000	3400	2200	1200	80	8000	6800	5720	1080
41	4100	3485	2288	1197	81	8100	6885	5808	1077
42	4200	3570	2376	1194	82	8200	6970	5896	1074
43	4300	3655	2464	1191	83	8300	7055	5984	1071
44	4400	3740	2552	1188	84	8400	7140	6072	1068
45	4500	3825	2640	1185	85	8500	7225	6160	1065
46	4600	3910	2728	1182	86	8600	7310	6248	1062
47	4700	3995	2816	1179	87	8700	7395	6336	1059
48	4800	4080	2904	1176	88	8800	7480	6424	1056
49	4900	4165	2992	1173	89	8900	7565	6512	1053
50	5000	4250	3080	1170	90	9000	7650	6600	1050
51	5100	4335	3168	1167	91	9100	7735	6688	1047
52	5200	4420	3256	1164	92	9200	7820	6776	1044
53	5300	4505	3344	1161	93	9300	7905	6864	1041
54	5400	4590	3432	1158	94	9400	7990	6952	1038
55	5500	4675	3520	1155	95	9500	8075	7040	1035
56	5600	4760	3608	1152	96	9600	8160	7128	1032
57	5700	4845	3696	1149	97	9700	8245	7216	1029
58	5800	4930	3784	1146	98	9800	8330	7304	1026
59	5900	5015	3872	1143	99	9900	8415	7392	1023
60	6000	5100	3960	1140	100	10000	8500	7480	1020

10.3. Table A-3. The actual compensations given by ELGA caused by flood events per Prefecture for the period 1999-2017.

PREFECTURE	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	Total
AITOLOAKARNANIA	89.610 €	5.166 €	1.049 €	59.079 €	76.650 €	5.859 €	789.726 €	72.628 €	291.590 €	3.371 €	13.786 €	10.674 €	7.914 €		120 €	7.871 €	874 €	2.982 €	2.048 €	1.440.996 €
ARGOLIDA	205.895 €	289 €		1.189 €	424.389 €		8.016 €	13.279 €		0 €	2.393 €	12.334 €	990 €	4.333 €	112.603 €					785.710 €
ARKADIA	12.837 €	1.412 €	33.626 €	54.361 €	771.283 €	13.479 €	44.822 €	53.818 €	24.928 €	12.239 €	130.877 €	127.018 €	42.680 €	51.571 €	70.319 €	1.329 €	6.305 €	1.396 €		1.454.300 €
ARTA				7.259 €	82.346 €	475 €	273.429 €	4.312 €	33.696 €	26.539 €	188.410 €	120.991 €	4.148 €	118.256 €	27.923 €	13.227 €	49.496 €		7.023 €	957.529 €
ATTICA	737.140 €	9.540 €	148.513 €	302.792 €	68.602 €	189.245 €	88.927 €	222.232 €	48.872 €	0 €			2.276 €	5.010 €		6.765 €	18.617 €	2.278 €	274.752 €	2.125.560 €
ACHAIA	4.693 €		19.999 €	407.441 €	43.876 €	1.293 €	1.145.114 €	6.978 €	17.941 €		1.347.327 €	13.518 €	458.400 €	21.057 €	8.532 €	639 €	4.442 €	11.084 €	6.927 €	3.519.260 €
VOIOTIA	1.745 €		230.931 €	453.289 €	308.052 €	227.958 €	109.349 €	93.063 €	6.977 €	767 €	8.234 €	18.641 €	84.873 €	12.582 €	43.186 €	23.521 €	170.478 €	5.424 €	8.414 €	1.807.483 €
GREVENA	155 €			37 €	287 €			264 €			1.760 €						353 €	453 €		3.309 €
DRAMA	4.025 €			24.899 €	10.256 €	53.309 €	416.028 €	8.889 €	126.187 €	5.179 €	8.194 €	19.557 €		686 €	1.144 €	41.931 €	1.649 €	1.510 €		723.442 €
DODEKANISA	1.844 €			11.977 €	3.174 €			5.462 €	1.946 €		9.590 €	1.741 €	1.028 €		2.055 €		755 €			39.572 €
EVROS	187.788 €	262.493 €	8.127 €	231.502 €	1.158.791 €	23.358 €	4.019.643 €	13.706.735 €	951.841 €	184.951 €	884.710 €	8.669.030 €	26.201 €	531.412 €	722.163 €	999.524 €	2.103.138 €	562.828 €	55.554 €	35.289.790 €
EVIA	19.125 €		61.330 €	492.367 €	55.897 €	33.711 €	66.163 €	216.300 €		0 €	489.124 €						4.590 €		22.901 €	1.461.507 €
EVRITANIA					351 €							67 €								418 €
ZAKYNTHOS				8.987 €									6.643 €					14.318 €		29.948 €
ILEIA	10.660 €	162 €	102.445 €	174.078 €	249.183 €	104 €	234.922 €	154.804 €	48.874 €	0 €	866.953 €	285.377 €	273.977 €	234.831 €	270.588 €	96.192 €	21.000 €	28.362 €	14.890 €	3.067.401 €
IMATHIA	3.978 €	8.786 €	128.966 €	263.452 €	1.777.499 €	15.110 €	0 €	21.600 €	6.785 €	5.965 €	113.147 €	140.929 €		358.252 €	212.575 €	39.516 €	28.056 €	345.979 €	19.248 €	3.489.844 €
HERACLION	23.305 €	692 €	633 €	239.320 €	13.690 €	87.909 €		173.079 €	570 €	267.960 €	199.087 €	9.262 €	7.747 €		297 €		5.043 €			1.028.594 €
THESPROTIA	1.837 €	821 €		117.755 €	101.439 €		16.842 €	77.848 €	2.477 €	2.331 €	110.566 €	396.205 €		61.436 €	4.085 €	111.175 €	6.681 €	55.703 €	120.185 €	1.187.385 €
THESSALONIKI	6.033 €	79.524 €	6.703 €	281.354 €	45.288 €	115.916 €	43.802 €	663.073 €	84.831 €	1.939 €	60.243 €	38.528 €	7.439 €	1.334 €	8.110 €	53.167 €	24.696 €	2.242 €		1.524.221 €
IOANNINA				104.183 €	26.597 €	1.393 €	7.968 €	313.888 €	2.787 €	6.295 €	2.551 €	16.916 €		27.511 €			203 €	11.224 €	462 €	521.978 €
KAVALA	1.027 €			96.204 €	16.822 €	8.480 €	95.354 €	92.069 €	33.936 €		22.398 €	1.720 €		24.806 €	809 €	117.716 €	8.129 €			519.469 €
KARDITSA	3.483 €	634 €	57.779 €	126.350 €	79.692 €	108.708 €	3.119 €	7.562 €			80.910 €	309.280 €	144.345 €	120.158 €	16.942 €	1.356 €	42.031 €	613.134 €	19.286 €	1.734.769 €
KASTORIA			1.719 €	5.813 €	336 €	1.147 €	4.244 €	1.323 €			2.972 €	1.279 €					2.723 €	60.027 €	12.543 €	94.126 €
KERKIRA		2.498 €		17.552 €		2.741 €		1.483 €			286 €	10.586 €								35.145 €
KEFALLINIA		2.485 €			102 €															2.587 €
KILKIS	14.362 €	32.879 €	3.837 €	278.001 €	341.525 €	329.827 €		112.857 €	5.509 €	3.256 €	144.200 €	180.285 €	1.572 €		33.291 €	278.730 €	22.017 €	130.444 €	13.262 €	1.925.853 €
KOZANI	1.248 €			4.991 €	227 €	10.211 €		43.295 €			11.657 €	694 €			3.976 €		80.891 €	5.883 €	84.228 €	247.301 €
KORINTHIA	16.064 €		3.894 €	6.071 €	74.953 €		4.363 €	63.601 €		8.039 €	20.582 €	18.000 €	3.777 €	5.305 €	10.612 €	8.195 €	5.175 €			248.631 €
CYCLADES	47.094 €		7.481 €	60.008 €	647.037 €	70.435 €		43.664 €	0 €	7.580 €		834 €	688.867 €	2.185 €						1.575.184 €
LACONIA	425.341 €	109.995 €	483 €		139.240 €	3.055 €	219.309 €	0 €	28.502 €					1.358 €	693 €	787 €		286.639 €		1.215.403 €
LARISSA	13.494 €	64.195 €	293 €	569.451 €	138.473 €	46.884 €	1.879 €	12.410 €	30.718 €	643 €	261.853 €	29.462 €		217.043 €	321 €	6.831 €	6.149 €	230.394 €	311.304 €	1.941.796 €
LASITHIOU		40.799 €	67.150 €	131.317 €	242.050 €	98.769 €	9.200 €	127.851 €	845 €	374.831 €	537.916 €	20.206 €		2.077 €			25.879 €		34.478 €	1.713.367 €
LESVOS	15.983 €	549 €	37.854 €	41.515 €	15.562 €	54.353 €	112.616 €				1.120 €	317 €	6.870 €	34.713 €			124.536 €	69.318 €	3.876 €	519.183 €
MAGNISIA	37.206 €		19.747 €	81.788 €	104.573 €	14.030 €		167.930 €	13.467 €	6.314 €	148.098 €			12.411 €					2.455 €	608.019 €
MESSINIA	1.100 €		703 €	12.210 €	165.646 €	97 €	93.455 €	10.706 €	2.261 €		154.567 €				36.305 €	360.213 €	35.172 €	651.120 €		1.523.554 €
XANTHI	8.763 €			1.150.278 €	145.015 €	9.764 €	53.375 €	182.387 €	934.251 €	585.231 €	61.391 €	411.343 €		330.593 €	152.177 €	726.755 €	409.264 €		4.230 €	5.164.817 €
PEIRAIOS					6.986 €							48 €								7.034 €
PELLA		4.850 €	69.183 €	115.767 €	6.048.783 €	7.568 €	0 €	19.232 €		3.497 €	174.108 €	88.990 €	3.231 €	8.452 €	358.149 €	372.862 €	401.211 €	425.522 €	23.871 €	8.125.276 €
PIERIA	144.732 €	160 €	6.286 €	306.796 €	27.669 €	668.946 €	1.105.144 €	282.976 €	209.554 €	40.688 €	183.586 €	104.009 €	675 €	81.210 €	4.085 €	3.299 €	1.197 €	27.369 €	79.225 €	3.277.603 €
PREVEZA				542.667 €	8.204 €	6.776 €	83.393 €	34.587 €	60.280 €	23.796 €	677.529 €	398.320 €		193.927 €	95.474 €	27.463 €	36.538 €	215.538 €	59.057 €	2.463.546 €
RETHYMNO		6.298 €	0 €	21.130 €	93 €	240 €	8.014 €	4.417 €	3.504 €			2.798 €								46.493 €
RODOPI	7.196 €	4.378 €	638 €	640.511 €	64.748 €	263.409 €	302.404 €	205.533 €	1.231.814 €	4.324 €	195.217 €	618.094 €		146.027 €	102.137 €	1.407.559 €	21.334 €	939.489 €	13.347 €	6.168.159 €
SAMOS	4.256 €		121.973 €		11.527 €				231 €		21.306 €	11.505 €	1.734 €		269 €					172.800 €
SERRES	63.453 €		12.835 €	1.239.141 €	166.440 €	307.713 €	243.546 €	502.798 €	213.232 €	9.005 €	53.358 €	61.893 €		67.612 €	81.356 €	41.310 €	459.506 €	31.247 €		3.554.444 €
TRIKALA	10.825 €	0 €	0 €	24.433 €	24.482 €	6.755 €	7.951 €	41.750 €	9.034 €		25.850 €	218.160 €	11.850 €	42.784 €	2.010 €	776 €	27.599 €	860.322 €	7.592 €	1.322.174 €
FTHIOTIDA	76.348 €	0 €	107.717 €	568.699 €	210.970 €	257.545 €	51.869 €	281.489 €	9.610 €		89.704 €	30.418 €	31.715 €	9.742 €	26.175 €	18.248 €	52.399 €	179.933 €	178.818 €	2.181.397 €
FLORINA	3.222 €					0 €		6.108 €	832 €		1.588 €	1.597 €	406 €		2.985 €	18.803 €	48.680 €	43.458 €		127.678 €
FOKIDA	222 €																			222 €
CHALKIDIKI	6.013 €	34.201 €	1.097 €	28.082 €	27.009 €	31.791 €	16.394 €	117.948 €	51.193 €	12.607 €	22.595 €	19.559 €		727 €		288 €	5.432 €			374.935 €
CHANIA	795 €	152.793 €	20.384 €	0 €	24.180 €	10.181 €	233 €	11.954 €	1.494 €		2.040 €									224.055 €
CHIOS	6.178 €			112.362 €	30.887 €		747.332 €					21.227 €					7.553 €			925.540 €
TOTAL	**2.219.075 €**	**825.601 €**	**1.283.376 €**	**9.416.462 €**	**13.980.881 €**	**3.088.540 €**	**10.427.941 €**	**18.184.177 €**	**4.490.565 €**	**1.597.345 €**	**7.331.781 €**	**12.441.409 €**	**1.819.357 €**	**2.729.401 €**	**2.411.465 €**	**4.786.047 €**	**4.269.789 €**	**5.815.621 €**	**1.379.974 €**	108.498.806 €

11. Python scripts

11.1. Selection of US-CAMELS timeseries with the maximum temporal overlap (namely, 35 years from 1980 to 2014) and less than 10% of missing values. Distribution of these gauge locations to their Hydrological Units (regions). Get their four first moments.

```
import numpy as np
import pandas as pd
import scipy
import os
from scipy.io import loadmat
from pandas import DataFrame
import seaborn as sns
from matplotlib import pyplot as plt

def missing_values_percentage(df_TS):
    df_withoutNaN=df_TS.dropna(axis=0,how='any', thresh=None, subset=None, inplace=False)
    Num_cols1=len(df_withoutNaN)
    Num_cols2=len(df_TS)
    missing_values_number=1-Num_cols1/Num_cols2
    return missing_values_number

def Get_Matlab_file_TS(filename):
    mat=scipy.io.loadmat("C:\\Users\\Konstantinos\\Documents\\THESIS\\Timeseries\\"+filename)
    a= mat['TS']
    df_TS=pd.DataFrame(a)
    return df_TS

#Selection of US-CAMELS time series with 35 years overlap (1980-2014) and max 10% missing values
acceptance_mv_rate=0.1

mat=scipy.io.loadmat('C:\\Users\\Konstantinos\\Documents\\THESIS\\Timeseries\\07067000.mat')
a= mat['TS']
original_df=pd.DataFrame(a)
Num_cols_file = len(original_df)

filenames=os.listdir('C:\\Users\\Konstantinos\\Documents\\THESIS\\Timeseries')
df_filenames=pd.DataFrame(filenames)
df_filenames.columns = ['Names']
Num_cols_filenames = len(df_filenames)

i=0
df_filenames_=df_filenames.copy()
while i<Num_cols_filenames:
    name=df_filenames_.iloc[i,0]
    name1=name.replace(".mat","")
    mat=scipy.io.loadmat('C:\\Users\\Konstantinos\\Documents\\THESIS\\Timeseries\\'+name)
    a= mat['TS']
    file=pd.DataFrame(a)
    missing_values_per=missing_values_percentage(file)
#    print(missing_values_per)
    if missing_values_per<=acceptance_mv_rate:
        i+=1
    else:
        df_filenames_=df_filenames_.drop([i], axis=0)
        df_filenames_=df_filenames_.reset_index(drop=True)
        Num_cols_filenames=Num_cols_filenames-1
#print(df_filenames_)
i=0
Num=len(df_filenames_)
while i<Num:
    name=df_filenames_.iloc[i,0]
    name1=name.replace(".mat","")
    mat=scipy.io.loadmat('C:\\Users\\Konstantinos\\Documents\\THESIS\\Timeseries\\'+name)
    a= mat['TS']
    file=pd.DataFrame(a)
    if file.iloc[0,0]==1980:
        i+=1
```

```
    else:
        df_filenames_=df_filenames_.drop([i], axis=0)
        df_filenames_=df_filenames_.reset_index(drop=True)
        Num=Num-1
#print(df_filenames_)
df_filenames_.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Timeseries 1980-2014 and 10% limit for missing values')

#Dataframe with the Gauge info of all US-CAMELS time series (671 gauge locations)
df=pd.read_excel('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\GAUGE_INFO.xlsx')
Num_cols=len(df)
i=0
while i<Num_cols:
    str_i=str(df.iloc[i,1])
    x=df.iloc[i,1]
    y=x/10000000
    y=int(y)
    if y==0:
        df.iloc[i,1]='0'+str_i
    str_=str(df.iloc[i,1])
    df.iloc[i,1]=str_
    i+=1
#print(df)
df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Gauge_info - 671 stations')

#Dataframe creation in order to distribute each station to its region (all stations without criteria)
i=1
while i<=18:
    net_i=str(i)
    stations=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Gauge_info - 671 stations')
    Num_cols_stations = len(stations)
    j=0
    while j<Num_cols_stations:
        if stations.iloc[j,0]!=i:
            stations=stations.drop([j], axis=0)
            stations=stations.reset_index(drop=True)
            Num_cols_stations=Num_cols_stations-1
        else:
            j+=1
    Num_cols=len(stations)
    k=0
    while k<Num_cols:
        filename=stations.iloc[k,1]
        filename1=filename.replace(".mat","")
        stations.iloc[k,1]=filename1
        k+=1
    stations.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Region '+net_i+' - All stations without criteria')
#    print(stations)
    i+=1

#Distribute each one of the 360 selected gauge locations to their Hydrologic Unit (regions)
df_360_stations=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Timeseries 1980-2014 and 10% limit for missing values')
Num_cols_360_stations=len(df_360_stations)
i=0
while i<Num_cols_360_stations:
    filename=df_360_stations.iloc[i,0]
    filename1=filename.replace(".mat","")
    df_360_stations.iloc[i,0]=filename1
    i+=1
#print(df_360_stations)
i=1
sum_=0
while i<=18:
    net_i=str(i)
    df_region=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Region '+net_i+' - All stations without criteria')
    Num_cols_df_region=len(df_region)
    index_region=0
    while index_region<Num_cols_df_region:
        identification_index=0
        index_360=0
```

```
        while index_360<Num_cols_360_stations:
            if df_region.iloc[index_region,1]==df_360_stations.iloc[index_360,0]:
                identification_index=1
            index_360+=1
        if identification_index==1:
            index_region+=1
        else:
            df_region=df_region.drop([index_region], axis=0)
            df_region=df_region.reset_index(drop=True)
            Num_cols_df_region=Num_cols_df_region-1
#    print(df_region)
    sum_=sum_+len(df_region)
    df_region.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Region '+net_i+' - All stations with criteria (1980-2014 and 10% missing values')
    i+=1
#print(sum_) #check if all 360 stations are distributed to regions


#Dataframe with the Gauge info of the selected US-CAMELS time series (360 gauge locations)
i=2
file=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Region 1 - All stations with criteria (1980-2014 and 10% missing values')
while i<19:
    file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Region '+str(i)+' - All stations with criteria (1980-2014 and 10% missing values')
    file=file.append(file1, ignore_index = True)
    i+=1
#print(file)
file['GAUGE_NAME'] = file['GAUGE_NAME'].str.upper()
file = file.sort_values(by ='GAUGE_ID')
file=file.reset_index(drop=True)
file.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Gauge_info - 360 stations')
file.to_excel("C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\GAUGE_INFO_360.xlsx")


#Get the Moments of the 360 selected timeseries
df_filenames=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Timeseries 1980-2014 and 10% limit for missing values')
Num_cols_filenames = len (df_filenames)
moments_df=df_filenames.copy()
moments_df['Mean']=df_filenames['Names']
moments_df['Variance']=df_filenames['Names']
moments_df['Skewness']=df_filenames['Names']
moments_df['Kurtosis']=df_filenames['Names']
i=0
while i<Num_cols_filenames:
    filename=df_filenames.iloc[i,0]
    filename1=filename.replace(".mat","")
    df_TS=Get_Matlab_file_TS(filename)
    df_TS.columns = ['Year', 'Month', 'Day', 'Q']
    df_TS=df_TS.drop(['Year', 'Month', 'Day'], axis=1)
    df_TS=df_TS.dropna(axis=0,how='any', thresh=None, subset=None, inplace=False)
    x=df_TS['Q']
    moments_df.iloc[i,1]=x.mean()
    moments_df.iloc[i,2]=x.var()
    moments_df.iloc[i,3]=x.skew()
    moments_df.iloc[i,4]=x.kurt()
    i+=1
i=0
while i<Num_cols_filenames:
    filename=df_filenames.iloc[i,0]
    filename1=filename.replace(".mat","")
    moments_df.iloc[i,0]=filename1
    i+=1
moments_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Timeseries 1980-2014 and 10% limit for missing values MOMENTS')
#print(moments_df)
```

11.2. Plot of all US-CAMELS gauge locations and the selected ones in different USA maps.

```
import geopandas as gpd
import matplotlib.pyplot as plt
```

```
import pandas as pd

shapefile='C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles\\States\\states.shp'
stations1=gpd.read_file(shapefile)
ax=stations1.plot(color='white', edgecolor='black',linewidth=0.3, figsize=(10,10))
ax.set_xlim(xmin=-130, xmax=-65)
ax.set_ylim(ymin=20, ymax=55)

Stations=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Gauge_info -
 671 stations')
x=Stations.iloc[:,4]
y=Stations.iloc[:,3]
plt.scatter(x,y,c='darkgrey', marker='o', edgecolor='black', linewidth=0.4)
plt.title('All CAMELS gauge stations')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Map with all the CAMELS gauge statio
ns.png')

region_index=1
while region_index<19:
    region_index_str=str(region_index)
    stations=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Region '+region_inde
x_str+' - All stations with criteria (1980-2014 and 10% missing values')
    x=stations.iloc[:,4]
    y=stations.iloc[:,3]
    plt.scatter(x,y, c='darkgrey', marker='o', edgecolor='black', linewidth=0.4)
    region_index+=1
plt.title('The selected 360 CAMELS gauge stations')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Map with the selected 360 CAMELS gau
ge stations.png')
```

11.3. Shuffle timeseries. Calculation of annual collective risk, annual peak, *Average* Y_i, the duration and the number of the over-threshold events. Calculation of the ECDF diagrams of collective risk and return intervals for all US-CAMELS timeseries and for all selected thresholds. Applied on the observed as well as the shuffled time series, and all thresholds.

```
import pandas as pd
import scipy
import os

def Get_Matlab_file_TS(filename):
    mat=scipy.io.loadmat("C:\\Users\\Konstantinos\\Documents\\THESIS\\Timeseries\\"+filename)
    a= mat['TS']
    df_TS=pd.DataFrame(a)
    return df_TS

def Get_Matlab_file_Info(filename):
    mat=scipy.io.loadmat("C:\\Users\\Konstantinos\\Documents\\THESIS\\Timeseries\\"+filename)
    b= mat['Info']
    df_Info=pd.DataFrame(b)
    return df_Info

def missing_values_percentage(df_TS):
    df_withoutNaN=df_TS.dropna(axis=0,how='any', thresh=None, subset=None, inplace=False)
    Num_cols1=len(df_withoutNaN)
    Num_cols2=len(df_TS)
    missing_values_percentage=1-Num_cols1/Num_cols2
    return missing_values_percentage

def Get_events_df(df_TS, threshold):
    #Drop NaN (missing values)
    df_withoutNaN=df_TS.dropna(axis=0,how='any', thresh=None, subset=None, inplace=False)
```

```
    df_sorted_noNaN=df_withoutNaN.sort_values(3, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last')
    Num_cols = len (df_withoutNaN)
    position=int(threshold*Num_cols)
    threshold_value=df_sorted_noNaN.iloc[position,3]
    events_df=df_withoutNaN
    events_df.columns = ['Year', 'Month', 'Day', 'Q']
    events_df.drop(events_df.loc[events_df['Q'] < threshold_value].index, inplace=True)
    return events_df

def Get_events_df_without_ones(events_df):
    Num_cols = len (events_df)
    events_df_without_ones=events_df.take([0])
    i=1
    while i<=Num_cols-1:
        if events_df.index[i]-1==events_df.index[i-1]:
            i+=1
        else:
            append_row=events_df.iloc[i].copy()
            events_df_without_ones=events_df_without_ones.append(append_row)
            i+=1
    return events_df_without_ones

def Get_return_interval_df(events_df_without_ones):
    Num_cols = len (events_df_without_ones)
    return_interval_df = {'Return interval': [events_df_without_ones.index[0]]}
    return_interval_df = pd.DataFrame(data=return_interval_df)
    i=1
    while i<=(Num_cols-1):
        j=events_df_without_ones.index[i]-events_df_without_ones.index[i-1]
        append_row = {'Return interval': [j]}
        append_row1 = pd.DataFrame(data=append_row)
        return_interval_df=return_interval_df.append(append_row1)
        i+=1
    return_interval_df=return_interval_df.reset_index(drop=True)
    return return_interval_df

def collective_risk_function(df_TS, events_df):
    first_year=df_TS.iloc[0,0]
    Num_cols = len (df_TS)-1
    last_year=df_TS.iloc[Num_cols,0]
    collective_risk_S_df = {'Year': [0], 'Collective Risk S': [0.0]}
    collective_risk_S_df = pd.DataFrame(data=collective_risk_S_df)
    a=first_year
    while a<=last_year-1:
        m={'Year': [0], 'Collective Risk S': [0.0]}
        m = pd.DataFrame(data=m)
        append_row=m.iloc[0].copy()
        collective_risk_S_df=collective_risk_S_df.append(append_row)
        a+=1
    collective_risk_S_df=collective_risk_S_df.reset_index(drop=True)
    a=first_year
    i=0
    while a<=last_year:
        collective_risk_S_df.xs(i)['Year'] = a
        a+=1
        i+=1
    a=first_year
    i=0
    j=0
    Num_cols = len (events_df)
    while a<=last_year:
        while j<=(Num_cols-1):
            if events_df.iloc[j,0]==a:
                collective_risk_S_df.xs(i)['Collective Risk S'] = collective_risk_S_df.iloc[i,1] + events_df.iloc[j,3]
                j+=1
            else:
                j+=1
        j=0
        i+=1
        a+=1
    return collective_risk_S_df

```

```
def number_of_events_function(df_TS, events_df):
    first_year=df_TS.iloc[0,0]
    Num_cols = len (df_TS)-1
    last_year=df_TS.iloc[Num_cols,0]
    number_of_events_df = {'Year': [0], 'Number of events': [0.0]}
    number_of_events_df = pd.DataFrame(data=number_of_events_df)
    a=first_year
    while a<=last_year-1:
        m={'Year': [0], 'Number of events': [0.0]}
        m = pd.DataFrame(data=m)
        append_row=m.iloc[0].copy()
        number_of_events_df=number_of_events_df.append(append_row)
        a+=1
    number_of_events_df=number_of_events_df.reset_index(drop=True)
    a=first_year
    i=0
    while a<=last_year:
        number_of_events_df.xs(i)['Year'] = a
        a+=1
        i+=1
    a=first_year
    i=0
    j=0
    Num_cols = len (events_df)
    while a<=last_year:
        while j<=(Num_cols-1):
            if events_df.iloc[j,0]==a:
                number_of_events_df.xs(i)['Number of events'] = number_of_events_df.iloc[i,1] + 1
                j+=1
            else:
                j+=1
        j=0
        i+=1
        a+=1
    return number_of_events_df

def annual_peak_only_for_events_function(df_TS, events_df):
    first_year=df_TS.iloc[0,0]
    Num_cols = len (df_TS)-1
    last_year=df_TS.iloc[Num_cols,0]
    annual_peak_only_for_events_df = {'Year': [0], 'Annual peak (Block maxima) only for events': [0.0
]}
    annual_peak_only_for_events_df = pd.DataFrame(data=annual_peak_only_for_events_df)
    a=first_year
    while a<=last_year-1:
        m={'Year': [0], 'Annual peak (Block maxima) only for events': [0.0]}
        m = pd.DataFrame(data=m)
        append_row=m.iloc[0].copy()
        annual_peak_only_for_events_df=annual_peak_only_for_events_df.append(append_row)
        a+=1
    annual_peak_only_for_events_df=annual_peak_only_for_events_df.reset_index(drop=True)
    a=first_year
    i=0
    while a<=last_year:
        annual_peak_only_for_events_df.xs(i)['Year'] = a
        a+=1
        i+=1
    a=first_year
    i=0
    j=0
    Num_cols = len (events_df)
    max=0
    while a<=last_year:
        while j<=(Num_cols-1):
            if events_df.iloc[j,0]==a:
                if events_df.iloc[j,3]>max:
                    annual_peak_only_for_events_df.xs(i)['Annual peak (Block maxima) only for events'
] = events_df.iloc[j,3]
                    max=events_df.iloc[j,3]
                    j+=1
                else:
                    j+=1
            else:
                j+=1
```

```
        max=0
        j=0
        i+=1
        a+=1
    return annual_peak_only_for_events_df

#Estimation of the annual_peak_for_ALL_df
def annual_peak_for_all_function(df_TS, events_df):
    first_year=df_TS.iloc[0,0]
    Num_cols = len (df_TS)-1
    last_year=df_TS.iloc[Num_cols,0]
    annual_peak_for_all_df = {'Year': [0], 'Annual peak (Block maxima) for all': [0.0]}
    annual_peak_for_all_df = pd.DataFrame(data=annual_peak_for_all_df)
    a=first_year
    while a<=last_year-1:
        m={'Year': [0.0], 'Annual peak (Block maxima) for all': [0.0]}
        m = pd.DataFrame(data=m)
        append_row=m.iloc[0].copy()
        annual_peak_for_all_df=annual_peak_for_all_df.append(append_row)
        a+=1
    annual_peak_for_all_df=annual_peak_for_all_df.reset_index(drop=True)
    a=first_year
    i=0
    while a<=last_year:
        annual_peak_for_all_df.xs(i)['Year'] = a
        a+=1
        i+=1
    a=first_year
    i=0
    j=0
    Num_cols = len (df_TS)
    max=0
    while a<=last_year:
        while j<=(Num_cols-1):
            if df_TS.iloc[j,0]==a:
                if df_TS.iloc[j,3]>max:
                    annual_peak_for_all_df.xs(i)['Annual peak (Block maxima) for all'] = df_TS.iloc[j
,3]
                    max=df_TS.iloc[j,3]
                    j+=1
                else:
                    j+=1
            else:
                j+=1
        max=0
        j=0
        i+=1
        a+=1
    return annual_peak_for_all_df

def average_Yi_function(df_TS, events_df):
    first_year=df_TS.iloc[0,0]
    Num_cols = len (df_TS)-1
    last_year=df_TS.iloc[Num_cols,0]
    average_Yi_df = {'Year': [0], 'Average Yi': [0.0]}
    average_Yi_df = pd.DataFrame(data=average_Yi_df)
    a=first_year
    while a<=last_year-1:
        m={'Year': [0], 'Average Yi': [0.0]}
        m = pd.DataFrame(data=m)
        append_row=m.iloc[0].copy()
        average_Yi_df=average_Yi_df.append(append_row)
        a+=1
    average_Yi_df=average_Yi_df.reset_index(drop=True)
    a=first_year
    i=0
    while a<=last_year:
        average_Yi_df.xs(i)['Year'] = a
        a+=1
        i+=1
    a=first_year
    i=0
    j=0
    Num_cols = len (events_df)
```

```
    while a<=last_year:
        r=0
        u=0
        while j<=(Num_cols-1):
            if events_df.iloc[j,0]==a:
                r=r + events_df.iloc[j,3]
                u+=1
                j+=1
            else:
                j+=1
        if u!=0:
            average_Yi_df.xs(i)['Average Yi'] =r/u
        j=0
        i+=1
        a+=1
    return average_Yi_df

def ecdf_S(collective_risk_S_df):
    ecdf_S_df=collective_risk_S_df.drop(['Year'], axis=1)
    ecdf_S_df.columns = ['Large sorted S']
    ecdf_S_df=ecdf_S_df.sort_values(by=['Large sorted S'],ascending=False)
    ecdf_S_df=ecdf_S_df.reset_index(drop=True)
    Num_cols=len(ecdf_S_df)
    x=ecdf_S_df.copy()
    ecdf_S_df['ECDF']=x['Large sorted S']
    i=0
    while i<Num_cols:
        ecdf_S_df.iloc[i,1]=1-((ecdf_S_df.index[i]+1)/(Num_cols+1))
        i+=1
    return ecdf_S_df

def ecdf_r(return_interval_df):
    ecdf_r_df=return_interval_df
    ecdf_r_df.columns = ['Large sorted r']
    ecdf_r_df=ecdf_r_df.sort_values(by=['Large sorted r'],ascending=False)
    ecdf_r_df=ecdf_r_df.reset_index(drop=True)
    Num_cols=len(ecdf_r_df)
    x=ecdf_r_df.copy()
    ecdf_r_df['ECDF']=x['Large sorted r']
    i=0
    while i<Num_cols:
        ecdf_r_df.iloc[i,1]=1-((ecdf_r_df.index[i]+1)/(Num_cols+1))
        i+=1
    return ecdf_r_df

def Get_duration_df(return_interval_df):
    ret_int_index=0
    while ret_int_index<len(return_interval_df):
        if return_interval_df.iloc[ret_int_index,0]!=1:
            return_interval_df.iloc[ret_int_index,0]=0.0
        ret_int_index+=1
#    print(return_interval_df)
    ret_int_index=1
    Num_return_interval_df=len(return_interval_df)
    while ret_int_index<Num_return_interval_df:
        if return_interval_df.iloc[ret_int_index,0]==1.0:
            return_interval_df.iloc[ret_int_index-1,0]=return_interval_df.iloc[ret_int_index-1,0]+1
            return_interval_df=return_interval_df.drop(return_interval_df.index[ret_int_index])
            return_interval_df=return_interval_df.reset_index(drop=True)
            Num_return_interval_df=Num_return_interval_df-1
        else:
            ret_int_index+=1
    ret_int_index=0
    duration_df=return_interval_df.copy()
    duration_df.columns=['Duration of event']
    while ret_int_index<len(duration_df):
        duration_df.iloc[ret_int_index,0]=duration_df.iloc[ret_int_index,0]+1
        ret_int_index+=1
    return duration_df

def ecdf_duration(duration_df):
    duration_df.columns = ['Large sorted duration']
    duration_df=duration_df.sort_values(by=['Large sorted duration'],ascending=False)
    duration_df=duration_df.reset_index(drop=True)
```

```
    Num_cols=len(duration_df)
    x=duration_df.copy()
    duration_df['ECDF']=x['Large sorted duration']
    i=0
    while i<Num_cols:
        duration_df.iloc[i,1]=1-((duration_df.index[i]+1)/(Num_cols+1))
        i+=1
    return duration_df

#Create 100 shuffled timeseries for observed timeseries
def Shuffle_100_timeseries(df_TS):
    df_TS.columns = ['Year', 'Month', 'Day', 'Q']
    shuffled_df=df_TS
    shuffled_df=shuffled_df.drop(['Year', 'Month', 'Day'], axis=1)
    x=shuffled_df
    i=1
    while i<101:
        x = x.sample(frac=1, axis=1).sample(frac=1).reset_index(drop=True)
        shuffled_df[i]=x['Q']
        i+=1
    shuffled_df.insert(loc=0, column='Day', value=df_TS['Day'])
    shuffled_df.insert(loc=0, column='Month', value=df_TS['Month'])
    shuffled_df.insert(loc=0, column='Year', value=df_TS['Year'])
    return shuffled_df

filenames=os.listdir('C:\\Users\\Konstantinos\\Documents\\THESIS\\Timeseries')
df_filenames=pd.DataFrame(filenames)
df_filenames.columns = ['Matlab']
Num_cols_filenames = len(df_filenames)
d = {'Year': [1980,1981,1982,1983,1984,1985,1986,1987,1988,1989,1990,1991,1992,1993,1994,1995,1996,19
97,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010,2011,2012,2013,2014,0]}
final_prototype_df = pd.DataFrame(data=d)
how_many_shuffled_timeseries=100
Collective_Risk_S_FOR_ALL_df=final_prototype_df.copy()
number_of_events_FOR_ALL_df=final_prototype_df.copy()
annual_peak_for_all_FOR_ALL_df=final_prototype_df.copy()
average_Yi_FOR_ALL_df=final_prototype_df.copy()

threshold_df_data = {'Text': ['90%','95%','98%','99%'],'Numerical': [0.9, 0.95, 0.98, 0.99]}
threshold_df = pd.DataFrame(data=threshold_df_data)
Num_threshold_df=len(threshold_df)
#print(threshold_df)

#Calculations for the observed time series
threshold_loop=0
while threshold_loop<Num_threshold_df:
    threshold=threshold_df.iloc[threshold_loop,1]
    threshold_text=threshold_df.iloc[threshold_loop,0]
    i=0
    while i<Num_cols_filenames:
        filename=df_filenames.iloc[i,0]
        filename1=filename.replace(".mat","")
        df_TS=Get_Matlab_file_TS(filename)
        df_Info=Get_Matlab_file_Info(filename)
    #    print(df_TS)
    #    print(df_Info)
        events_df=Get_events_df(df_TS, threshold)
    #    print(events_df)
        events_df_without_ones=Get_events_df_without_ones(events_df)
    #    print(events_df_without_ones)
        return_interval_df=Get_return_interval_df(events_df_without_ones)
    #    print (return_interval_df)
        collective_risk_S_df=collective_risk_function(df_TS, events_df)
    #    print(collective_risk_S_df)
        number_of_events_df=number_of_events_function(df_TS, events_df)
    #    print(number_of_events_df)
        annual_peak_only_for_events_df=annual_peak_only_for_events_function(df_TS, events_df)
    #    print(annual_peak_only_for_events_df)
        annual_peak_for_all_df=annual_peak_for_all_function(df_TS, events_df)
    #    print(annual_peak_for_all_df)
        average_Yi_df=average_Yi_function(df_TS, events_df)
    #    print(average_Yi_df)
        missing_values_per=missing_values_percentage(df_TS)
    #    print(missing_values_percentage)
```

```
391.         ecdf_S_df=ecdf_S(collective_risk_S_df)
392.     #    print(ecdf_S_df)
393.         ecdf_S_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\1. Results for the O
    bserved\\'+threshold_text+'\\ecdf_S\\ecdf_S '+filename1)
394.         ecdf_r_df=ecdf_r(return_interval_df)
395.     #    print(ecdf_r_df)
396.         ecdf_r_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\1. Results for the O
    bserved\\'+threshold_text+'\\ecdf_r\\ecdf_r '+filename1)
397.         shuffled_100_df=Shuffle_100_timeseries(df_TS)
398.     #    print(shuffled_100_df)
399.         shuffled_100_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\2. The shuffle
    d of all 671\\Shuffled_100 '+filename1)
400.
401.         Collective_Risk_S_FOR_ALL_df[filename1]=collective_risk_S_df['Collective Risk S']
402.         Collective_Risk_S_FOR_ALL_df.iloc[35,i+1]=missing_values_per
403.         number_of_events_FOR_ALL_df[filename1]=number_of_events_df['Number of events']
404.         number_of_events_FOR_ALL_df.iloc[35,i+1]=missing_values_per
405.         annual_peak_for_all_FOR_ALL_df[filename1]=annual_peak_for_all_df['Annual peak (Block maxima)
    for all']
406.         annual_peak_for_all_FOR_ALL_df.iloc[35,i+1]=missing_values_per
407.         average_Yi_FOR_ALL_df[filename1]=average_Yi_df['Average Yi']
408.         average_Yi_FOR_ALL_df.iloc[35,i+1]=missing_values_per
409.
410.         i+=1
411.
412.     Sorted_Collective_Risk_S_FOR_ALL_df = Collective_Risk_S_FOR_ALL_df.sort_values(by=35, axis=1, asc
    ending=True)
413.     Sorted_number_of_events_FOR_ALL_df = number_of_events_FOR_ALL_df.sort_values(by=35, axis=1, ascen
    ding=True)
414.     Sorted_annual_peak_for_all_FOR_ALL_df = annual_peak_for_all_FOR_ALL_df.sort_values(by=35, axis=1,
     ascending=True)
415.     Sorted_average_Yi_FOR_ALL_df = average_Yi_FOR_ALL_df.sort_values(by=35, axis=1, ascending=True)
416.     #print(Sorted_Collective_Risk_S_FOR_ALL_df)
417.     #print(Sorted_number_of_events_FOR_ALL_df)
418.     #print(Sorted_annual_peak_for_all_FOR_ALL_df)
419.     #print(Sorted_average_Yi_FOR_ALL_df)
420.
421.     Sorted_Collective_Risk_S_FOR_ALL_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python
    \\1. Results for the Observed\\'+threshold_text+'\\Collective Risk S CAMELS 1980-2014')
422.     Sorted_number_of_events_FOR_ALL_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\
    \1. Results for the Observed\\'+threshold_text+'\\Number of events N CAMELS 1980-2014')
423.     Sorted_annual_peak_for_all_FOR_ALL_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Pyth
    on\\1. Results for the Observed\\'+threshold_text+'\\Annual Peak CAMELS 1980-2014')
424.     Sorted_average_Yi_FOR_ALL_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\1. Re
    sults for the Observed\\'+threshold_text+'\\Annual Average Yi CAMELS 1980-2014')
425.     threshold_loop+=1
426.
427.#Calculations for the shuffled time series
428.threshold_loop=0
429.while threshold_loop<Num_threshold_df:
430.     threshold=threshold_df.iloc[threshold_loop,1]
431.     threshold_text=threshold_df.iloc[threshold_loop,0]
432.     i=0
433.     l=0
434.     while i<Num_cols_filenames:
435.         filename=df_filenames.iloc[i,0]
436.         filename1=filename.replace(".mat","")
437.         pickle_file = pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\2. The shuf
    fled of all 671\\Shuffled_100 '+filename1)
438.         j=0
439.         while j+3<how_many_shuffled_timeseries+3:
440.             df_TS=pickle_file[['Year', 'Month', 'Day',j+1]].copy()
441.             df_withoutNaN=df_TS.dropna(axis=0,how='any', thresh=None, subset=None, inplace=False)
442.     #        print(df_withoutNaN)
443.             df_sorted_noNaN=df_withoutNaN.sort_values(j+1, axis=0, ascending=True, inplace=False, kin
    d='quicksort', na_position='last')
444.             Num_cols = len (df_withoutNaN)
445.             position=int(threshold*Num_cols)
446.             threshold_value=df_sorted_noNaN.iloc[position,3]
447.             events_df=df_withoutNaN
448.             events_df.columns = ['Year', 'Month', 'Day', 'Q']
449.             events_df.drop(events_df.loc[events_df['Q'] < threshold_value].index, inplace=True)
450.     #        print(events_df)
451.             events_df_without_ones=Get_events_df_without_ones(events_df)
```

```
452.    #        print(events_df_without_ones)
453.             return_interval_df=Get_return_interval_df(events_df_without_ones)
454.    #        print (return_interval_df)
455.             collective_risk_S_df=collective_risk_function(df_TS, events_df)
456.    #        print(collective_risk_S_df)
457.             number_of_events_df=number_of_events_function(df_TS, events_df)
458.    ##        print(number_of_events_df)
459.             annual_peak_only_for_events_df=annual_peak_only_for_events_function(df_TS, events_df)
460.    ##        print(annual_peak_only_for_events_df)
461.             annual_peak_for_all_df=annual_peak_for_all_function(df_TS, events_df)
462.    ##        print(annual_peak_for_all_df)
463.             average_Yi_df=average_Yi_function(df_TS, events_df)
464.    ##        print(average_Yi_df)
465.             if j==0:
466.                 ecdf_S_df=ecdf_S(collective_risk_S_df)
467.    #             print(ecdf_S_df)
468.                 l+=2
469.             else:
470.                 u=ecdf_S(collective_risk_S_df)
471.                 u.columns = ['Large sorted S', 'ECDF']
472.                 ecdf_S_df[l]=u['Large sorted S']
473.                 ecdf_S_df[l+1]=u['ECDF']
474.    #             print(ecdf_S_df)
475.                 l+=2
476.             ecdf_r_df=ecdf_r(return_interval_df)
477.#             print(ecdf_r_df)
478.             Collective_Risk_S_FOR_ALL_df[j]=collective_risk_S_df['Collective Risk S']
479.             number_of_events_FOR_ALL_df[j]=number_of_events_df['Number of events']
480.             annual_peak_for_all_FOR_ALL_df[j]=annual_peak_for_all_df['Annual peak (Block maxima) for
    all']
481.             average_Yi_FOR_ALL_df[j]=average_Yi_df['Average Yi']
482.             ecdf_r_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\4. Results for t
    he Shuffled of the timeseries with criteria\\'+threshold_text+'\\'+filename1+' 100_ecdf_r '+str(j))
483.#             print(filename+' shuffle:'+str(j))
484.#             print(threshold_text, i, j)
485.             j+=1
486.         ecdf_S_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\4. Results for the S
    huffled of the timeseries with criteria\\'+threshold_text+'\\'+filename1+' 100 ecdf_S')
487.         Collective_Risk_S_FOR_ALL_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\4
    . Results for the Shuffled of the timeseries with criteria\\'+threshold_text+'\\'+filename1+' Collect
    ive Risk for 100 Shuffled')
488.         number_of_events_FOR_ALL_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\4.
     Results for the Shuffled of the timeseries with criteria\\'+threshold_text+'\\'+filename1+' Number o
    f events N for 100 Shuffled')
489.         annual_peak_for_all_FOR_ALL_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\
    \4. Results for the Shuffled of the timeseries with criteria\\'+threshold_text+'\\'+filename1+' Annua
    l Peak  for 100 Shuffled')
490.         average_Yi_FOR_ALL_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\4. Resul
    ts for the Shuffled of the timeseries with criteria\\'+threshold_text+'\\'+filename1+' Annual Average
     Yi  for 100 Shuffled')
491.         Collective_Risk_S_FOR_ALL_df=final_prototype_df.copy()
492.         number_of_events_FOR_ALL_df=final_prototype_df.copy()
493.         annual_peak_for_all_FOR_ALL_df=final_prototype_df.copy()
494.         average_Yi_FOR_ALL_df=final_prototype_df.copy()
495.#         print(i)
496.         i+=1
497.         threshold_loop+=1
498.
499.#Calculations of the duration of the over threshold events for the observed and the shuffled series
500.threshold_loop=1
501.while threshold_loop<Num_threshold_df:
502.     threshold=threshold_df.iloc[threshold_loop,1]
503.     threshold_text=threshold_df.iloc[threshold_loop,0]
504.     i=0
505.     while i<Num_cols_filenames:
506.         filename=df_filenames.iloc[i,0]
507.         filename1=filename.replace(".mat","")
508.         df_TS=Get_Matlab_file_TS(filename)
509.         df_Info=Get_Matlab_file_Info(filename)
510.     #     print(df_TS)
511.     #     print(df_Info)
512.         events_df=Get_events_df(df_TS, threshold)
513.#         print(events_df)
514.         return_interval_df=Get_return_interval_df(events_df)
```

```
#        print(return_interval_df)
        duration_df_obs=Get_duration_df(return_interval_df)
        duration_df_obs.columns=['Observed Duration']
        duration_df_obs.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\15. Duration of events\\duration_df\\'+threshold_text+'\\'+filename1+' duration_df OBSERVED '+threshold_text)
#        print(duration_df_obs)
        ecdf_duration_df=ecdf_duration(duration_df_obs)
#        print(ecdf_duration_df)
        ecdf_duration_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\15. Duration of events\\ECDF files\\'+threshold_text+'\\'+filename1+' ECDF OBSERVED '+threshold_text)
        pickle_file = pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\2. The shuffled of all 671\\Shuffled_100 '+filename1)
        j=0
        while j+3<how_many_shuffled_timeseries+3:
            df_TS=pickle_file[['Year', 'Month', 'Day',j+1]].copy()
#            print(df_TS)
            df_withoutNaN=df_TS.dropna(axis=0,how='any', thresh=None, subset=None, inplace=False)
    #        print(df_withoutNaN)
            df_sorted_noNaN=df_withoutNaN.sort_values(j+1, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last')
            Num_cols = len (df_withoutNaN)
            position=int(threshold*Num_cols)
            threshold_value=df_sorted_noNaN.iloc[position,3]
            events_df=df_withoutNaN
#            print(events_df)
            events_df.columns = ['Year', 'Month', 'Day', 'Q']
            events_df.drop(events_df.loc[events_df['Q'] < threshold_value].index, inplace=True)
    #        print(events_df)
            return_interval_df=Get_return_interval_df(events_df)
#            print(return_interval_df)
            duration_df_shuffled=Get_duration_df(return_interval_df)
            duration_df_shuffled.columns=['Shuffled Duration']
            duration_df_shuffled.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\15. Duration of events\\duration_df\\'+threshold_text+'\\'+filename1+' duration_df SHUFFLED '+str(j+1)+' '+threshold_text)
#            print(duration_df_shuffled)
            ecdf_duration_df=ecdf_duration(duration_df_shuffled)
#            print(ecdf_duration_df)
            ecdf_duration_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\15. Duration of events\\ECDF files\\'+threshold_text+'\\'+filename1+' ECDF SHUFFLED '+str(j+1)+' '+threshold_text)
            j+=1
        i+=1
    threshold_loop+=1
```

11.4. Plot of the ECDF diagrams of collective risk, return intervals and the duration of the over-threshold events of the selected gauge locations and for all thresholds.

```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt

threshold_df_data = {'Text': ['90%','95%','98%','99%'],'Numerical': [0.9, 0.95, 0.98, 0.99]}
threshold_df = pd.DataFrame(data=threshold_df_data)
Num_threshold_df=len(threshold_df)
#print(threshold_df)

df_filenames=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Timeseries 1980-2014 and 10% limit for missing values')
df_filenames.columns = ['Names']
Num_cols_filenames = len (df_filenames)

HU_df=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Gauge_info - 360 stations')
HU_df=HU_df.drop("DRAINAGE AREA (KM^2)", axis=1)
HU_df=HU_df.drop("LONG", axis=1)
HU_df=HU_df.drop("LAT", axis=1)
Num=len(HU_df)
#print(HU_df)
i=0
```

```
while i<Num:
    filename=df_filenames.iloc[i,0]
    filename1=filename.replace(".mat","")
    if HU_df.iloc[i,1]==filename1:
        i+=1
    else:
        HU_df_df=HU_df.drop(HU_df.index[i])
        Num=Num-1
        HU_df=HU_df.reset_index(drop=True)
#print(HU_df)

how_many_shuffled_times=100
threshold_loop=0
while threshold_loop<Num_threshold_df:
    threshold=threshold_df.iloc[threshold_loop,0]
#    Export ECDF-S diagram
    j=1
    while j<=Num_cols_filenames:
        filename=df_filenames.iloc[j-1,0]
        filename1=filename.replace(".mat","")
        HU=str(HU_df.iloc[j-1,0])
        Gauge_name=str.upper(HU_df.iloc[j-1,2])
        file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\4. Results for the Shuffled of the timeseries with criteria\\'+threshold+'\\'+filename1+' 100 ecdf_S')
        file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\1. Results for the Observed\\'+threshold+'\\ecdf_S\\ecdf_S '+filename1)
        sns.set()
        sns.set_style("ticks")
        plt.figure(figsize=(8,5),dpi=200)
        for i in  np.arange(1,200,2):
            ecdf_s=file1.iloc[:,i]
            colrisk=file1.iloc[:,i-1]
            plt.plot(colrisk,ecdf_s,label='',color='darkgrey')
        plt.plot(colrisk,ecdf_s,label='Shuffled',color='darkgrey')
        plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='Observed', marker='o', markersize=3, color='r')
        plt.title(Gauge_name+'\nGauge ID: '+filename1+' - Hydrological Unit: '+HU+' - Threshold '+threshold,fontsize=13, y=1.01)
        plt.ylabel("ECDF",fontsize=14)
        plt.xlabel("Collective Risk (S)",fontsize=14)
        plt.yticks(np.arange(0, 1.1, 0.1))
        plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
        plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(0.95, 0.4),loc=1,ncol=1,fontsize=14)
        plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
        plt.tight_layout()
        plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\5. ECDF diagrams for S and r\\Linear\\ECDF_S\\Linear ECDF_S '+filename1+' '+threshold+'.png',facecolour='white')
        plt.close()
        j+=1

    #Export ECDF-r diagram
    j=1
    while j<=Num_cols_filenames:
        filename=df_filenames.iloc[j-1,0]
        filename1=filename.replace(".mat","")
        HU=str(HU_df.iloc[j-1,0])
        Gauge_name=str.upper(HU_df.iloc[j-1,2])
        file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\1. Results for the Observed\\'+threshold+'\\ecdf_r\\ecdf_r '+filename1)
        sns.set()
        sns.set_style("ticks")
        plt.figure(figsize=(8,5),dpi=200)
        i=0
        while i<100:
            file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\4. Results for the Shuffled of the timeseries with criteria\\'+threshold+'\\'+filename1+' 100_ecdf_r '+str(i))
            ecdf_r=file1.iloc[:,1]
            retint=file1.iloc[:,0]
            plt.plot(retint,ecdf_r,label='',color='darkgrey')
            i+=1
        plt.plot(retint,ecdf_r,label='Shuffled',color='darkgrey')
        plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='Observed', marker='o', markersize=3, color='r')
```

```
        plt.title(Gauge_name+'\nGauge ID: '+filename1+' - Hydrological Unit: '+HU+' - Threshold '+threshold,fontsize=13, y=1.01)
        plt.ylabel("ECDF",fontsize=14)
        plt.xlabel("Return Interval (r)",fontsize=14)
        plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
        plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(0.95, 0.4),loc=1,ncol=1,fontsize=14)
        plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
        plt.tight_layout()
        plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\5. ECDF diagrams for S and r \\Linear\\ECDF_r\\Linear ECDF_r '+filename1+' '+threshold+'.png',facecolour='white')
        plt.close()
        j+=1

#    Export LOG-LOG ECDF-S diagram
    j=1
    while j<=Num_cols_filenames:
        filename=df_filenames.iloc[j-1,0]
        filename1=filename.replace(".mat","")
        HU=str(HU_df.iloc[j-1,0])
        Gauge_name=str.upper(HU_df.iloc[j-1,2])
        file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\4. Results for the Shuffled of the timeseries with criteria\\'+threshold+'\\'+filename1+' 100 ecdf_S')
        file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\1. Results for the Observed\\'+threshold+'\\ecdf_S\\ecdf_S '+filename1)
        sns.set()
        sns.set_style("ticks")
        plt.figure(figsize=(8,5),dpi=200)
        for i in  np.arange(1,200,2):
            ecdf_s=file1.iloc[:,i]
            colrisk=file1.iloc[:,i-1]
            plt.plot(colrisk,ecdf_s,label='',color='darkgrey')
        plt.plot(colrisk,ecdf_s,label='Shuffled',color='darkgrey')
        plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='Observed', marker='o', markersize=3, color='r')
        plt.title(Gauge_name+'\nGauge ID: '+filename1+' - Hydrological Unit: '+HU+' - Threshold '+threshold,fontsize=13, y=1.01)
        plt.ylabel("ECDF",fontsize=14)
        plt.xlabel("Collective Risk (S)",fontsize=14)
        plt.xscale('log')
        plt.yscale('log')
        labels = ['0.05', '0.1', '0.2', '0.5', '1.0']
        nthreads = [0.05, 0.1, 0.2, 0.5, 1.0]
        plt.yticks(nthreads, labels)
        plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
        plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(0.95, 0.4),loc=1,ncol=1,fontsize=14)
        plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
        plt.tight_layout()
        plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\5. ECDF diagrams for S and r \\LOG\\ECDF_S\\Log ECDF_S '+filename1+' '+threshold+'.png',facecolour='white')
        plt.close()
        j+=1

    #Export LOG-LOG ECDF-r diagram
    j=1
    while j<=Num_cols_filenames:
        filename=df_filenames.iloc[j-1,0]
        filename1=filename.replace(".mat","")
        HU=str(HU_df.iloc[j-1,0])
        Gauge_name=str.upper(HU_df.iloc[j-1,2])
        file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\1. Results for the Observed\\'+threshold+'\\ecdf_r\\ecdf_r '+filename1)
        sns.set()
        sns.set_style("ticks")
        plt.figure(figsize=(8,5),dpi=200)
        i=0
        while i<100:
            file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\4. Results for the Shuffled of the timeseries with criteria\\'+threshold+'\\'+filename1+' 100_ecdf_r '+str(i))
            ecdf_r=file1.iloc[:,1]
            retint=file1.iloc[:,0]
            plt.plot(retint,ecdf_r,label='',color='darkgrey')
            i+=1
            plt.plot(retint,ecdf_r,label='',color='darkgrey')
        plt.plot(retint,ecdf_r,label='Shuffled',color='darkgrey')
```

```
        plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='Observed', marker='o', markersize=3, color='r')
        plt.title(Gauge_name+'\nGauge ID: '+filename1+' - Hydrological Unit: '+HU+' - Threshold '+threshold,fontsize=13, y=1.01)
        plt.ylabel("ECDF",fontsize=14)
        plt.xlabel("Return Interval (r)",fontsize=14)
        plt.xscale('log')
        plt.yscale('log')
        labels = ['0.05', '0.1', '0.2', '0.5', '1.0']
        nthreads = [0.05, 0.1, 0.2, 0.5, 1.0]
        plt.yticks(nthreads, labels)
        plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
        plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(0.95, 0.4),loc=1,ncol=1,fontsize=14)
        plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
        plt.tight_layout()
        plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\5. ECDF diagrams for S and r\\LOG\\ECDF_r\\Log ECDF_r '+filename1+' '+threshold+'.png',facecolour='white')
        plt.close()
        j+=1

#    Export 1/(1-f) ECDF-S diagram
    j=1
    while j<=Num_cols_filenames:
        filename=df_filenames.iloc[j-1,0]
        filename1=filename.replace(".mat","")
        HU=str(HU_df.iloc[j-1,0])
        Gauge_name=str.upper(HU_df.iloc[j-1,2])
        file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\4. Results for the Shuffled of the timeseries with criteria\\'+threshold+'\\'+filename1+' 100 ecdf_S')
        file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\1. Results for the Observed\\'+threshold+'\\ecdf_S\\ecdf_S '+filename1)
        sns.set()
        sns.set_style("ticks")
        plt.figure(figsize=(8,5),dpi=200)
        for i in  np.arange(1,200,2):
            ecdf_s=1/(1-file1.iloc[:,i])
            colrisk=file1.iloc[:,i-1]
            plt.plot(colrisk,ecdf_s,label='',color='darkgrey')
        plt.plot(colrisk,ecdf_s,label='Shuffled',color='darkgrey')
        plt.plot(file2.iloc[:,0],1/(1-file2.iloc[:,1]),label='Observed', marker='o', markersize=3, color='r')
        plt.title(Gauge_name+'\nGauge ID: '+filename1+' - Hydrological Unit: '+HU+' - Threshold '+threshold,fontsize=13, y=1.01)
        plt.ylabel("1/(1-ECDF)",fontsize=14)
        plt.xlabel("Collective Risk (S)",fontsize=14)
#        plt.yticks(np.arange(0, 1.1, 0.1))
        plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
        plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(1.005, 0.215),loc=1,ncol=1,fontsize=14)
        plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
        plt.tight_layout()
        plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\5. ECDF diagrams for S and r\\1--(1-ECDF)\\ECDF_S\\1--(1-ECDF) ECDF_S '+filename1+' '+threshold+'.png',facecolour='white')
        plt.close()
        j+=1

    #Export 1/(1-f) ECDF-r diagram
    j=1
    while j<=Num_cols_filenames:
        filename=df_filenames.iloc[j-1,0]
        filename1=filename.replace(".mat","")
        HU=str(HU_df.iloc[j-1,0])
        Gauge_name=str.upper(HU_df.iloc[j-1,2])
        file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\1. Results for the Observed\\'+threshold+'\\ecdf_r\\ecdf_r '+filename1)
        sns.set()
        sns.set_style("ticks")
        plt.figure(figsize=(8,5),dpi=200)
        i=0
        while i<100:
            file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\4. Results for the Shuffled of the timeseries with criteria\\'+threshold+'\\'+filename1+' 100_ecdf_r '+str(i))
            ecdf_r=1/(1-file1.iloc[:,1])
            retint=file1.iloc[:,0]
            plt.plot(retint,ecdf_r,label='',color='darkgrey')
```

```
            i+=1
        plt.plot(retint,ecdf_r,label='Shuffled',color='darkgrey')
        plt.plot(file2.iloc[:,0],1/(1-
    file2.iloc[:,1]),label='Observed', marker='o', markersize=3, color='r')
        plt.title(Gauge_name+'\nGauge ID: '+filename1+' - Hydrological Unit: '+HU+' -
     Threshold '+threshold,fontsize=13, y=1.01)
        plt.ylabel("1/(1-ECDF)",fontsize=14)
        plt.xlabel("Return Interval (r)",fontsize=14)
        plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
        plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(1, 1),loc=1,ncol=1,fontsize=14)
        plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
        plt.tight_layout()
        plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\5. ECDF diagrams for S and r
    \\1--(1-ECDF)\\ECDF_r\\1--(1-ECDF) ECDF_r '+filename1+' '+threshold+'.png',facecolour='white')
        plt.close()
        j+=1

    #Export ECDF-Duration diagram
    j=1
    while j<=Num_cols_filenames:
        filename=df_filenames.iloc[j-1,0]
        filename1=filename.replace(".mat","")
        HU=str(HU_df.iloc[j-1,0])
        Gauge_name=str.upper(HU_df.iloc[j-1,2])
        file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\15. Duration of eve
    nts\\ECDF files\\'+threshold+'\\'+filename1+' ECDF OBSERVED '+threshold)
        sns.set()
        sns.set_style("ticks")
        plt.figure(figsize=(8,5),dpi=200)
        shuffle_index=1
        while shuffle_index<=how_many_shuffled_times:
            file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\15. Duration of
     events\\ECDF files\\'+threshold+'\\'+filename1+' ECDF SHUFFLED '+str(shuffle_index)+' '+threshold)
            plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='',color='darkgrey')
            shuffle_index+=1
        plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='Shuffled',color='darkgrey')
        plt.plot(file1.iloc[:,0],file1.iloc[:,1],label='Observed', marker='o', markersize=3, color='r
    ')
        plt.title(Gauge_name+'\nGauge ID: '+filename1+' - Hydrological Unit: '+HU+' -
     Threshold '+threshold,fontsize=13, y=1.01)
        plt.ylabel("ECDF",fontsize=14)
        plt.xlabel("Duration of events (days)",fontsize=14)
        plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
        plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(1, 0.22),loc=1,ncol=1,fontsize=14)
        plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
        plt.tight_layout()
        plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\15. Duration of events\\Plot
    s\\Linear\\Linear ECDF Duration '+filename1+' '+threshold+'.png',facecolour='white')
        plt.close()
        j+=1

    #Export LOG-LOG ECDF-Duration diagram
    j=1
    while j<=Num_cols_filenames:
        filename=df_filenames.iloc[j-1,0]
        filename1=filename.replace(".mat","")
        HU=str(HU_df.iloc[j-1,0])
        Gauge_name=str.upper(HU_df.iloc[j-1,2])
        file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\15. Duration of eve
    nts\\ECDF files\\'+threshold+'\\'+filename1+' ECDF OBSERVED '+threshold)
        sns.set()
        sns.set_style("ticks")
        plt.figure(figsize=(8,5),dpi=200)
        shuffle_index=1
        while shuffle_index<=how_many_shuffled_times:
            file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\15. Duration of
     events\\ECDF files\\'+threshold+'\\'+filename1+' ECDF SHUFFLED '+str(shuffle_index)+' '+threshold)
            plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='',color='darkgrey')
            shuffle_index+=1
        plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='Shuffled',color='darkgrey')
        plt.plot(file1.iloc[:,0],file1.iloc[:,1],label='Observed', marker='o', markersize=3, color='r
    ')
        plt.title(Gauge_name+'\nGauge ID: '+filename1+' - Hydrological Unit: '+HU+' -
     Threshold '+threshold,fontsize=13, y=1.01)
```

```
        plt.ylabel("ECDF",fontsize=14)
        plt.xlabel("Duration of events (days)",fontsize=14)
        plt.xscale('log')
        plt.yscale('log')
        labels = ['0.05', '0.1', '0.2', '0.5', '1.0']
        nthreads = [0.05, 0.1, 0.2, 0.5, 1.0]
        plt.yticks(nthreads, labels)
        plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
        plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(1, 0.22),loc=1,ncol=1,fontsize=14)
        plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
        plt.tight_layout()
        plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\15. Duration of events\\Plots\\LOG\\LOG ECDF Duration '+filename1+' '+threshold+'.png',facecolour='white')
        plt.close()
        j+=1

    #Export 1/(1-f) ECDF-Duration diagram
    j=1
    while j<=Num_cols_filenames:
        filename=df_filenames.iloc[j-1,0]
        filename1=filename.replace(".mat","")
        HU=str(HU_df.iloc[j-1,0])
        Gauge_name=str.upper(HU_df.iloc[j-1,2])
        file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\15. Duration of events\\ECDF files\\'+threshold+'\\'+filename1+' ECDF OBSERVED '+threshold)
        sns.set()
        sns.set_style("ticks")
        plt.figure(figsize=(8,5),dpi=200)
        shuffle_index=1
        while shuffle_index<=how_many_shuffled_times:
            file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\15. Duration of events\\ECDF files\\'+threshold+'\\'+filename1+' ECDF SHUFFLED '+str(shuffle_index)+' '+threshold)
            plt.plot(file2.iloc[:,0],1/(1-file2.iloc[:,1]),label='',color='darkgrey')
            shuffle_index+=1
        plt.plot(file2.iloc[:,0],1/(1-file2.iloc[:,1]),label='Shuffled',color='darkgrey')
        plt.plot(file1.iloc[:,0],1/(1-file1.iloc[:,1]),label='Observed', marker='o', markersize=3, color='r')
        plt.title(Gauge_name+'\nGauge ID: '+filename1+' - Hydrological Unit: '+HU+' - Threshold '+threshold,fontsize=13, y=1.01)
        plt.ylabel("1/(1-ECDF)",fontsize=14)
        plt.xlabel("Duration of events (days)",fontsize=14)
        plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
        plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(1.005, 1.008),loc=1,ncol=1,fontsize=14)
        plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
        plt.tight_layout()
        plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\15. Duration of events\\Plots\\1--(1-ECDF)\\1--(1-ECDF) ECDF Duration '+filename1+' '+threshold+'.png',facecolour='white')
        plt.close()
        j+=1

    threshold_loop+=1
```

11.5. Heatmaps of the Spearman and Pearson correlation of collective risk between the gauge locations of every Hydrological Unit (region) for all thresholds.

```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from scipy.stats import pearsonr
from scipy.stats import spearmanr

def CorrMtx(df, region_index_str, threshold_text, correlation_label, dropDuplicates = True):
    if dropDuplicates:
        mask = np.zeros_like(df, dtype=np.bool)
        mask[np.triu_indices_from(mask)] = True
    sns.set_style(style = 'white')
    sns.set_style("ticks")
    fig, ax = plt.subplots(figsize=(20,15),dpi=200)
    sns.heatmap(df, vmin=-1, vmax=1, cmap=plt.cm.get_cmap('RdBu', 8),
                square=True,
```

```
                linewidth=.5, cbar_kws={"shrink": .7,"ticks":[-1,-0.75,-0.5,-0.25,0,0.25,0.5,0.75,1]}, ax=ax)

    plt.title(correlation_label+' correlation for basin '+region_index_str+' - Threshold '+threshold_text, fontdict={'fontsize': 30,'verticalalignment':'top'})
    plt.xlabel("Stations", fontsize=25,labelpad=15)
    plt.ylabel("Stations", fontsize=25,labelpad=15)
    plt.tick_params(axis='both',which='both',labelsize=15,zorder=20)
    if len(df)>50:
        plt.tick_params(axis='both',which='both',labelsize=10,zorder=20)
    plt.xticks(rotation=90)
    plt.yticks(rotation=0)
    cbar = ax.collections[0].colorbar
    cbar.ax.tick_params(labelsize=17)
    cbar.set_label('Correlation coefficient', fontsize=19, labelpad=20)
    plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\\S Correlation between 360 stations\\Basins\\Heatmap ('+correlation_label+') in Basin '+region_index_str+' ('+threshold_text+') - no zeros')
    plt.close()

threshold_df_data = {'Text': ['90%','95%','98%','99%'],'Numerical': [0.9, 0.95, 0.98, 0.99]}
threshold_df = pd.DataFrame(data=threshold_df_data)
Num_threshold_df=len(threshold_df)
#print(threshold_df)

threshold_loop=0
while threshold_loop<Num_threshold_df:
    threshold=threshold_df.iloc[threshold_loop,1]
    threshold_text=threshold_df.iloc[threshold_loop,0]

    file=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\1. Results for the Observed\\'+threshold_text+'\\Collective Risk S CAMELS 1980-2014')
    file=file.drop(file.index[35])
    file=file.drop('Year', axis=1)

    #print(df)
    pearson_df=file.corr(method ='pearson')
    spearman_df=file.corr(method ='spearman')
    #print(pearson_df)

    df_filenames=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Timeseries 1980-2014 and 10% limit for missing values')
    Num_cols=len(df_filenames)
    i=0
    while i<Num_cols:
        filename=df_filenames.iloc[i,0]
        filename1=filename.replace(".mat","")
        df_filenames.iloc[i,0]=filename1
        i+=1
    length=len(pearson_df)

    i=0
    while i<length:
        y=0
        k=0
        r=pearson_df.index[i]
        while y<Num_cols:
            filename1=df_filenames.iloc[y,0]
            if filename1==r:
                k=1
                y=Num_cols
            y+=1
        if k==1:
            i+=1
        else:
            pearson_df=pearson_df.drop([r], axis=1)
            pearson_df=pearson_df.drop([r], axis=0)
            spearman_df=spearman_df.drop([r], axis=1)
            spearman_df=spearman_df.drop([r], axis=0)
            length=length-1
    pearson_df=pearson_df.sort_index(axis=0)
    pearson_df=pearson_df.sort_index(axis=1)
    spearman_df=spearman_df.sort_index(axis=0)
    spearman_df=spearman_df.sort_index(axis=1)
```

```
85.     pearson_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\\S Correlation between 360 stations\\Pearson for observed S between 360 stations ('+threshold_text+') - with zeros')
86.     spearman_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\\S Correlation between 360 stations\\Spearman for observed S between 360 stations ('+threshold_text+') - with zeros')
87. #    print(pearson_df)
88. #    print(spearman_df)
89.
90.     ii=0
91.     while ii<Num_cols:
92.         jj=0
93.         while jj<Num_cols:
94.             pearson_df.iloc[ii,jj]=0.0
95.             spearman_df.iloc[ii,jj]=0.0
96.             jj+=1
97.         ii+=1
98.
99.     i=0
100.    while i<Num_cols:
101.        filename1=df_filenames.iloc[i,0]
102.        j=0
103.        while j<Num_cols:
104.            filename2=df_filenames.iloc[j,0]
105.            s1_df=file[filename1]
106.            s2_df=file[filename2]
107.            Num_s=len(s2_df)
108.            k=0
109.            while k<Num_s:
110.                x1=s1_df.iloc[k]
111.                x2=s2_df.iloc[k]
112.                if (x1==0) or (x2==0):
113.                    s1_df=s1_df.drop(s1_df.index[k])
114.                    s2_df=s2_df.drop(s2_df.index[k])
115.                    s1_df=s1_df.reset_index(drop=True)
116.                    s2_df=s2_df.reset_index(drop=True)
117.                    Num_s=Num_s-1
118.                else:
119.                    k+=1
120.            corr1,p_value1=spearmanr(s1_df, s2_df)
121.            spearman_df.iloc[i,j]=corr1
122.            corr2,p_value1=pearsonr(s1_df, s2_df)
123.            pearson_df.iloc[i,j]=corr2
124.            j+=1
125. #        print(i)
126.        i+=1
127.    spearman_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\\S Correlation between 360 stations\\Spearman for observed S between 360 stations ('+threshold_text+') - no zeros')
128.    pearson_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\\S Correlation between 360 stations\\Pearson for observed S between 360 stations ('+threshold_text+') - no zeros')
129. #    print(spearman_df)
130. #    print(pearson_df)
131.
132.    #Print correlation heat-map for every basin's stations
133.    spearman_df=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\\S Correlation between 360 stations\\Spearman for observed S between 360 stations ('+threshold_text+') - no zeros')
134.    pearson_df=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\\S Correlation between 360 stations\\Pearson for observed S between 360 stations ('+threshold_text+') - no zeros')
135.    region_index=1
136.    while region_index<19:
137.        region_index_str=str(region_index)
138.        stations_in_region=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Region '+region_index_str+' - All stations with criteria (1980-2014 and 10% missing values')
139.        Num_stations_in_region=len(stations_in_region)
140.        pearson_df1=pearson_df.copy()
141.        spearman_df1=spearman_df.copy()
142.        i=0
143.        Num=len(pearson_df)
144.        while i<Num:
145.            j=0
```

```
            k=0
            r=pearson_df1.index[i]
            while j<Num_stations_in_region:
                station_name=stations_in_region.iloc[j,1]
                if station_name==r:
                    k=1
                    j=Num_stations_in_region
                j+=1
            if k==1:
                i+=1
            else:
                pearson_df1=pearson_df1.drop(r, axis=0)
                pearson_df1=pearson_df1.drop(r, axis=1)
                spearman_df1=spearman_df1.drop(r, axis=0)
                spearman_df1=spearman_df1.drop(r, axis=1)
                Num=Num-1
        spearman_df1.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\\S Correlation between 360 stations\\Basins\\Dfs\\Spearman for observed S in basin '+region_index_str+' ('+threshold_text+') - no zeros')
        pearson_df1.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\\S Correlation between 360 stations\\Basins\\Dfs\\Pearson for observed S in basin '+region_index_str+' ('+threshold_text+') - no zeros')
        correlation_label='Spearman'
        CorrMtx(spearman_df1, region_index_str, threshold_text, correlation_label, dropDuplicates = True)
        correlation_label='Pearson'
        CorrMtx(pearson_df1, region_index_str, threshold_text, correlation_label, dropDuplicates = True)
        region_index+=1
    threshold_loop+=1
```

11.6. USA map with Spearman, Pearson and Kendall correlation coefficient between *Average Yi* and the Number of the over-threshold events *N* of the selected gauge locations and for all thresholds

```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import geopandas as gpd
from scipy.stats import pearsonr
from scipy.stats import spearmanr
from scipy.stats import kendalltau
from mpl_toolkits.axes_grid1 import make_axes_locatable

def Map_plot(corr_coef, threshold_text, correlation_label, zeros_label):
    shapefile='C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles\\States\\states.shp'
    stations1=gpd.read_file(shapefile)
    ax=stations1.plot(color='white', edgecolor='black',linewidth=0.3, figsize=(16,16))
    ax.set_xlim(xmin=-130, xmax=-65)
    ax.set_ylim(ymin=20, ymax=60)
    Stations=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Gauge_info - 671 stations')
    Num_stations=len(Stations)
    lat_lon=corr_coef.copy()
    lat_lon['column']=corr_coef['Corr. coef.']
    lat_lon.columns=['Name','Lat','Lon']
    Num_lat_lon=len(lat_lon)
    #print(lat_lon)
    i=0
    j=0
    while i<Num_lat_lon:
        j=0
        while j<Num_stations:
            if lat_lon.iloc[i,0]==Stations.iloc[j,1]:
                lat_lon.iloc[i,1]=Stations.iloc[j,3]
                lat_lon.iloc[i,2]=Stations.iloc[j,4]
            j+=1
```

```
        i+=1
    lat_lon['Corr. coef.']=corr_coef['Corr. coef.']
    #print(lat_lon)
    x=lat_lon.iloc[:,2]
    y=lat_lon.iloc[:,1]
    plt.scatter(x,y, c=lat_lon.iloc[:,3], marker='o', edgecolor='black', linewidth=0.2, alpha=1, cmap=plt.cm.get_cmap('RdYlGn', 8))
    plt.clim(-1, 1);
    plt.title(correlation_label+' correlation coefficient between Average Yi and Number of events N - Threshold '+threshold_text,fontsize=19, y=1.05)
    plt.xlabel('Longitude',fontsize=18)
    plt.ylabel('Latitude',fontsize=18)
    plt.tick_params(axis='both',which='both',labelsize=17,zorder=20)

    divider = make_axes_locatable(ax)
    cax = divider.append_axes("right", size="5%", pad=0.05)

    v1 = np.linspace(-1, 1, 9)
    cbar = plt.colorbar(ticks=v1, cax=cax)
    cbar.ax.tick_params(labelsize=17)
    cbar.set_label('Correlation coefficient', fontsize=17, labelpad=20)

    plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\\Correlation for Average Yi and N\\Observed Average Yi-N '+correlation_label+' correlation ('+threshold_text+') - '+zeros_label+'.png')


threshold_df_data = {'Text': ['90%','95%','98%','99%'],'Numerical': [0.9, 0.95, 0.98, 0.99]}
threshold_df = pd.DataFrame(data=threshold_df_data)
Num_threshold_df=len(threshold_df)
#print(threshold_df)

threshold_loop=0
while threshold_loop<Num_threshold_df:
    threshold=threshold_df.iloc[threshold_loop,1]
    threshold_text=threshold_df.iloc[threshold_loop,0]

    #File for Correlation coefficient of Average Yi and Number of Events per year N
    file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\1. Results for the Observed\\'+threshold_text+'\\Annual Average Yi CAMELS 1980-2014')
    Yi=file1.drop(file1.index[35])
    Yi=Yi.drop(['Year'], axis=1)
    file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\1. Results for the Observed\\'+threshold_text+'\\Number of events N CAMELS 1980-2014')
    N=file2.drop(file2.index[35])
    N=N.drop(['Year'], axis=1)
    #print(Yi)
    #print(N)

    corr_coef=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Timeseries 1980-2014 and 10% limit for missing values')
    pearson_corr_coef=corr_coef.copy()
    spearman_corr_coef=corr_coef.copy()
    kendall_corr_coef=corr_coef.copy()
    Num_cols=len(corr_coef)
    i=0
    while i<Num_cols:
        filename=corr_coef.iloc[i,0]
        filename1=filename.replace(".mat","")
        pearson_corr_coef.iloc[i,0]=filename1
        spearman_corr_coef.iloc[i,0]=filename1
        kendall_corr_coef.iloc[i,0]=filename1
        i+=1

    #Excluding the zero values on the correlation
    zeros_label='without zeros'
    i=0
    while i<Num_cols:
        filename=pearson_corr_coef.iloc[i,0]
        N_column=N[filename]
        Yi_column=Yi[filename]
        j=0
        k=34
        while j<=k:
```

```
            if N_column.iloc[j]==0:
                N_column=N_column.drop(N_column.index[j])
                Yi_column=Yi_column.drop(Yi_column.index[j])
                k=k-1
            else:
                j+=1

        corr1,p_value1=pearsonr(N_column, Yi_column)
        pearson_corr_coef.iloc[i,1]=corr1
        corr2,p_value1=spearmanr(N_column, Yi_column)
        spearman_corr_coef.iloc[i,1]=corr2
        corr3,p_value1=kendalltau(N_column, Yi_column)
        kendall_corr_coef.iloc[i,1]=corr3
        i+=1
#    print(pearson_corr_coef)
#    print(spearman_corr_coef)
#    print(kendall_corr_coef)
    pearson_corr_coef.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\
\Correlation for Average Yi and N\\Observed Average Yi-N Pearson correlation ('+threshold_text+') -
 without zeros')
    spearman_corr_coef.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations
\\Correlation for Average Yi and N\\Observed Average Yi-N Spearman correlation ('+threshold_text+') -
 without zeros')
    kendall_corr_coef.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\
\Correlation for Average Yi and N\\Observed Average Yi-N Kendall correlation ('+threshold_text+') -
 without zeros')

    correlation_label='Pearson'
    Map_plot(pearson_corr_coef, threshold_text, correlation_label, zeros_label)
    correlation_label='Spearman'
    Map_plot(spearman_corr_coef, threshold_text, correlation_label, zeros_label)
    correlation_label='Kendall'
    Map_plot(kendall_corr_coef, threshold_text, correlation_label, zeros_label)

    threshold_loop+=1

#Cumulative histograms for Pearson's and Spearman's correlation coefficients
pearson_corr_coef1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlation
s\\Correlation for Average Yi and N\\Observed Average Yi-N Pearson correlation (90%) -
 without zeros')
spearman_corr_coef1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlatio
ns\\Correlation for Average Yi and N\\Observed Average Yi-N Spearman correlation (90%) -
 without zeros')
kendall_corr_coef1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlation
s\\Correlation for Average Yi and N\\Observed Average Yi-N Kendall correlation (90%) -
 without zeros')
pearson_corr_coef2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlation
s\\Correlation for Average Yi and N\\Observed Average Yi-N Pearson correlation (95%) -
 without zeros')
spearman_corr_coef2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlatio
ns\\Correlation for Average Yi and N\\Observed Average Yi-N Spearman correlation (95%) -
 without zeros')
kendall_corr_coef2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlation
s\\Correlation for Average Yi and N\\Observed Average Yi-N Kendall correlation (95%) -
 without zeros')
pearson_corr_coef3=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlation
s\\Correlation for Average Yi and N\\Observed Average Yi-N Pearson correlation (98%) -
 without zeros')
spearman_corr_coef3=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlatio
ns\\Correlation for Average Yi and N\\Observed Average Yi-N Spearman correlation (98%) -
 without zeros')
kendall_corr_coef3=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlation
s\\Correlation for Average Yi and N\\Observed Average Yi-N Kendall correlation (98%) -
 without zeros')
pearson_corr_coef4=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlation
s\\Correlation for Average Yi and N\\Observed Average Yi-N Pearson correlation (99%) -
 without zeros')
spearman_corr_coef4=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlatio
ns\\Correlation for Average Yi and N\\Observed Average Yi-N Spearman correlation (99%) -
 without zeros')
kendall_corr_coef4=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlation
s\\Correlation for Average Yi and N\\Observed Average Yi-N Kendall correlation (99%) -
 without zeros')

```

```
plt.figure(figsize=(8,5),dpi=200)
plt.style.use('seaborn-white')
plt.xlabel('Correlation coefficient')
plt.ylabel('Cumulative probability')
plt.title('Cumulative histogram curves of the correlation coefficient \nof Average Yi and Number of Events N for the 360 stations')
sns.distplot(pearson_corr_coef1.iloc[:,1], bins=100,hist_kws=dict(cumulative=True),
                 kde_kws=dict(cumulative=True),hist = False, color="lightsalmon", label="Pearson 90%")
sns.distplot(pearson_corr_coef2.iloc[:,1], bins=100,hist_kws=dict(cumulative=True),
                 kde_kws=dict(cumulative=True),hist = False, color="red", label="Pearson 95%")
sns.distplot(pearson_corr_coef3.iloc[:,1], bins=100,hist_kws=dict(cumulative=True),
                 kde_kws=dict(cumulative=True),hist = False, color="indianred", label="Pearson 98%")

sns.distplot(pearson_corr_coef4.iloc[:,1], bins=100,hist_kws=dict(cumulative=True),
                 kde_kws=dict(cumulative=True),hist = False, color="darkred", label="Pearson 99%")

sns.distplot(spearman_corr_coef1.iloc[:,1], bins=100,hist_kws=dict(cumulative=True),
                 kde_kws=dict(cumulative=True),hist = False, color="skyblue", label="Spearman 90%")
sns.distplot(spearman_corr_coef2.iloc[:,1], bins=100,hist_kws=dict(cumulative=True),
                 kde_kws=dict(cumulative=True),hist = False, color="royalblue", label="Spearman 95%")

sns.distplot(spearman_corr_coef3.iloc[:,1], bins=100,hist_kws=dict(cumulative=True),
                 kde_kws=dict(cumulative=True),hist = False, color="blue", label="Spearman 98%")
sns.distplot(spearman_corr_coef4.iloc[:,1], bins=100,hist_kws=dict(cumulative=True),
                 kde_kws=dict(cumulative=True),hist = False, color="navy", label="Spearman 99%")

sns.distplot(kendall_corr_coef1.iloc[:,1], bins=100,hist_kws=dict(cumulative=True),
                 kde_kws=dict(cumulative=True),hist = False, color="lightgreen", label="Kendall 90%")

sns.distplot(kendall_corr_coef2.iloc[:,1], bins=100,hist_kws=dict(cumulative=True),
                 kde_kws=dict(cumulative=True),hist = False, color="limegreen", label="Kendall 95%")

sns.distplot(kendall_corr_coef3.iloc[:,1], bins=100,hist_kws=dict(cumulative=True),
                 kde_kws=dict(cumulative=True),hist = False, color="forestgreen", label="Kendall 98%")
sns.distplot(kendall_corr_coef4.iloc[:,1], bins=100,hist_kws=dict(cumulative=True),
                 kde_kws=dict(cumulative=True),hist = False, color="green", label="Kendall 99%")
#plt.xlim(-1, 1, 0.1)
plt.xticks(np.arange(-1, 1.01, 0.2))
plt.yticks(np.arange(0, 1.01, 0.1))
plt.grid(axis='both', alpha=0.75)
plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(1, 0.6),loc=1,ncol=1,fontsize=9)
plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\\Correlation for Average Yi and N\\Cumulative histogram curves of Pearson, Spearman and Kendall (all thresholds).png')
```

11.7. Box plot of the correlation coefficient (Spearman, Pearson and Kendall) between the *Average Yi* and the Number of the over-threshold events *N* of the selected gauge locations, considering observed and shuffled data, and for all thresholds.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from scipy.stats import pearsonr
from scipy.stats import spearmanr
from scipy.stats import kendalltau

def boxplot(correlation_label, x, filename):
    file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\\Correlation for Average Yi and N\\Shuffled\\Shuffled Average Yi-N '+correlation_label+' correlation (90%) - without zeros')
    file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\\Correlation for Average Yi and N\\Shuffled\\Shuffled Average Yi-N '+correlation_label+' correlation (95%) - without zeros')
    file3=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\\Correlation for Average Yi and N\\Shuffled\\Shuffled Average Yi-N '+correlation_label+' correlation (98%) - without zeros')
```

```
    file4=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\\Correlation for Average Yi and N\\Shuffled\\Shuffled Average Yi-N '+correlation_label+' correlation (99%) - without zeros')

    # Create data
    collectn_1 = file1.iloc[x,1:100]
    collectn_11 = file1.iloc[x,101]
    collectn_2 = file2.iloc[x,1:100]
    collectn_22 = file2.iloc[x,101]
    collectn_3 = file3.iloc[x,1:100]
    collectn_33 = file3.iloc[x,101]
    collectn_4 = file4.iloc[x,1:100]
    collectn_44 = file4.iloc[x,101]

    # combine these different collections into a list
    data_to_plot = [collectn_1, collectn_11, collectn_2, collectn_22, collectn_3, collectn_33, collectn_4, collectn_44]

    # Create a figure instance
    fig=plt.figure(figsize=(11,6),dpi=200)
    ax = fig.add_subplot(111)
    medianprops = dict(linestyle='-', linewidth=2, color='red')
    ax.set_axisbelow(True)
    ax.set_title('Boxplot of the '+correlation_label+' correlation coefficient between \nAverage Yi and Number of Events (N) (Gauge ID: '+filename+')',fontsize=14)
    ax.set_ylabel(correlation_label+' correlation coefficient',fontsize=13)
    ax.tick_params(axis='x', labelsize=10,zorder=20,rotation=0)
    bp = ax.boxplot(data_to_plot,medianprops=medianprops,patch_artist=True,labels=['90% Shuffled','90% Observed','95% Shuffled','95% Observed','98% Shuffled','98% Observed','99% Shuffled','99% Observed'])
    ax.grid(color='black', linestyle='-', linewidth=0.1)
    colors = ['skyblue','blue', 'moccasin','green', 'lightsalmon','tan', 'tan','pink']
    for patch, color in zip(bp['boxes'], colors):
        patch.set_facecolor(color)
    plt.yticks(np.arange(-0.6, 1.01, 0.1))
    plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\\Correlation for Average Yi and N\\Boxplots\\'+correlation_label+' Boxplot '+filename+'.png')
    plt.close()

threshold_df_data = {'Text': ['90%','95%','98%','99%'],'Numerical': [0.9, 0.95, 0.98, 0.99]}
threshold_df = pd.DataFrame(data=threshold_df_data)
Num_threshold_df=len(threshold_df)
#print(threshold_df)

df_filenames=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Timeseries 1980-2014 and 10% limit for missing values')
Num_cols_filenames = len (df_filenames)
i=0
while i<Num_cols_filenames:
    filename=df_filenames.iloc[i,0]
    filename1=filename.replace(".mat","")
    df_filenames.iloc[i,0]=filename1
    i+=1
df_filenames1=df_filenames.copy()
df1=df_filenames.copy()
i=1
while i<101:
    i_str=str(i)
    df_filenames1.columns = [i_str]
    df1[i_str]=df_filenames1[i_str]
    i+=1
#print(df1)
#print(df_filenames)

pearson_corr_coef=df1.copy()
spearman_corr_coef=df1.copy()
kendall_corr_coef=df1.copy()


threshold_loop=0
while threshold_loop<Num_threshold_df:
    threshold=threshold_df.iloc[threshold_loop,1]
    threshold_text=threshold_df.iloc[threshold_loop,0]

```

```
    #Excluding the zero values on the correlation
    i=0
    while i<Num_cols_filenames:
        filename=df_filenames.iloc[i,0]
        Yi=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\4. Results for the Shu
ffled of the timeseries with criteria\\'+threshold_text+'\\'+filename+' Annual Average Yi  for 100 Sh
uffled')
        N=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\4. Results for the Shuf
fled of the timeseries with criteria\\'+threshold_text+'\\'+filename+' Number of events N for 100 Shu
ffled')
        Yi=Yi.drop(['Year'], axis=1)
        N=N.drop(['Year'], axis=1)
        j=0
        while j<100:
            Yi_column=Yi[j]
            N_column=N[j]
            z=0
            k=34
            while z<=k:
                if N_column.iloc[z]==0:
                    N_column=N_column.drop(N_column.index[z])
                    Yi_column=Yi_column.drop(Yi_column.index[z])
                    N_column=N_column.reset_index(drop=True)
                    Yi_column=Yi_column.reset_index(drop=True)
                    k=k-1
                else:
                    z+=1
            corr1,p_value1=pearsonr(N_column, Yi_column)
            pearson_corr_coef.iloc[i,j+1]=corr1
            corr2,p_value1=spearmanr(N_column, Yi_column)
            spearman_corr_coef.iloc[i,j+1]=corr2
            corr3,p_value1=kendalltau(N_column, Yi_column)
            kendall_corr_coef.iloc[i,j+1]=corr3
            j+=1
        i+=1
#    print(pearson_corr_coef)
#    print(spearman_corr_coef)
#    print(kendall_corr_coef)
    observed_pearson=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlati
ons\\Correlation for Average Yi and N\\Observed Average Yi-
N Pearson correlation ('+threshold_text+') - without zeros')
    pearson_corr_coef['Observed']=observed_pearson['Corr. coef.']
    observed_spearman=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlat
ions\\Correlation for Average Yi and N\\Observed Average Yi-
N Spearman correlation ('+threshold_text+') - without zeros')
    spearman_corr_coef['Observed']=observed_spearman['Corr. coef.']
    observed_kendall=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlati
ons\\Correlation for Average Yi and N\\Observed Average Yi-
N Kendall correlation ('+threshold_text+') - without zeros')
    kendall_corr_coef['Observed']=observed_kendall['Corr. coef.']
    pearson_corr_coef.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\
\Correlation for Average Yi and N\\Shuffled\\Shuffled Average Yi-
N Pearson correlation ('+threshold_text+') - without zeros')
    spearman_corr_coef.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations
\\Correlation for Average Yi and N\\Shuffled\\Shuffled Average Yi-
N Spearman correlation ('+threshold_text+') - without zeros')
    kendall_corr_coef.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\6. Correlations\
\Correlation for Average Yi and N\\Shuffled\\Shuffled Average Yi-
N Kendall correlation ('+threshold_text+') - without zeros')

    threshold_loop+=1

i=0
while i<Num_cols_filenames:
    filename=df_filenames.iloc[i,0]
    correlation_label='Spearman'
    boxplot(correlation_label, i, filename)
    correlation_label='Pearson'
    boxplot(correlation_label, i, filename)
    correlation_label='Kendall'
    boxplot(correlation_label, i, filename)
    i+=1
```

11.8. Plot of the ECDF of annual peak of observed and shuffled, as well as the GEV distribution of observed and shuffled for all gauge locations and all thresholds.

```
import numpy as np
import pandas as pd
from scipy.stats import genextreme as gev
import seaborn as sns
from matplotlib import pyplot as plt

def ecdf_annual_maxima(file,filename):
    ecdf_annual_maxima=file[filename]
    ecdf_annual_maxima_df = pd.DataFrame(ecdf_annual_maxima)
    if len(ecdf_annual_maxima)==36:
        ecdf_annual_maxima_df=ecdf_annual_maxima_df.drop(ecdf_annual_maxima_df.index[35])
    ecdf_annual_maxima_df.columns = ['Large sorted Maxima']
    ecdf_annual_maxima_df=ecdf_annual_maxima_df.sort_values(by=['Large sorted Maxima'],ascending=Fals
e)
    ecdf_annual_maxima_df=ecdf_annual_maxima_df.reset_index(drop=True)
    Num_cols=len(ecdf_annual_maxima_df)
    x=ecdf_annual_maxima_df.copy()
    ecdf_annual_maxima_df['ECDF']=x['Large sorted Maxima']
    j=0
    while j<Num_cols:
        ecdf_annual_maxima_df.iloc[j,1]=1-((ecdf_annual_maxima_df.index[j]+1)/(Num_cols+1))
        j+=1
    return ecdf_annual_maxima_df

def ecdf_annual_maxima_shuffled(file,i):
    ecdf_annual_maxima=file.iloc[:,i]
    ecdf_annual_maxima_df = pd.DataFrame(ecdf_annual_maxima)
    if len(ecdf_annual_maxima)==36:
        ecdf_annual_maxima_df=ecdf_annual_maxima_df.drop(ecdf_annual_maxima_df.index[35])
    ecdf_annual_maxima_df.columns = ['Large sorted Maxima']
    ecdf_annual_maxima_df=ecdf_annual_maxima_df.sort_values(by=['Large sorted Maxima'],ascending=Fals
e)
    ecdf_annual_maxima_df=ecdf_annual_maxima_df.reset_index(drop=True)
    Num_cols=len(ecdf_annual_maxima_df)
    x=ecdf_annual_maxima_df.copy()
    ecdf_annual_maxima_df['ECDF']=x['Large sorted Maxima']
    j=0
    while j<Num_cols:
        ecdf_annual_maxima_df.iloc[j,1]=1-((ecdf_annual_maxima_df.index[j]+1)/(Num_cols+1))
        j+=1
    return ecdf_annual_maxima_df


# Creation of ECDF dataframe files for annual peak for the observed and shuffled time series
df_filenames=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Timeseries 1980-
2014 and 10% limit for missing values')
Num_cols = len (df_filenames)
i=0
while i<Num_cols:
    filename=df_filenames.iloc[i,0]
    filename1=filename.replace(".mat","")
    df_filenames.iloc[i,0]=filename1
    i+=1
file=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\1. Results for the Observed\
\95%\\Annual Peak CAMELS 1980-2014')

i=1
while i<Num_cols+1:
    filename=df_filenames.iloc[i-1,0]
    #For observed
    ecdf_annual_maxima_df=ecdf_annual_maxima(file,filename)
#    print(ecdf_annual_maxima_df)
    ecdf_annual_maxima_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREM
E\\Observed\\'+filename+' ECDF annual maxima OBSERVED')
    #For shuffled
    file_shuffled=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\4. Results for
the Shuffled of the timeseries with criteria\\95%\\'+filename+' Annual Peak  for 100 Shuffled')
    j=1 #1st shuffled
```

```
    o=2
    while j<101:
        if j==1:
            ecdf_annual_maxima_shuffled_df=ecdf_annual_maxima_shuffled(file_shuffled,j)
        else:
            ecdf_annual_maxima_shuffled_df_1=ecdf_annual_maxima_shuffled(file_shuffled,j)
            ecdf_annual_maxima_shuffled_df[o]=ecdf_annual_maxima_shuffled_df_1['Large sorted Maxima']

            ecdf_annual_maxima_shuffled_df[o+1]=ecdf_annual_maxima_shuffled_df_1['ECDF']
        j+=1
        o+=2
#    print(ecdf_annual_maxima_shuffled_df)
    ecdf_annual_maxima_shuffled_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\Shuffled\\'+filename+' ECDF annual maxima SHUFFLED')
    i+=1

#Get GEV parameters of the observed and the shuffled time series of the selected gauge locations and their ECDF dataframes files
#In GEV Annual Peak, threshold does not matter, only in Generalised Pareto
threshold='95%' #freezed value

file=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\1. Results for the Observed\'+threshold+'\\Annual Peak CAMELS 1980-2014')
file=file.drop(file.index[35])
file=file.drop(columns=['Year'])
df_filenames=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Timeseries 1980-2014 and 10% limit for missing values')
Num_cols = len (df_filenames)
i=0
while i<Num_cols:
    filename=df_filenames.iloc[i,0]
    filename1=filename.replace(".mat","")
    df_filenames.iloc[i,0]=filename1
    i+=1
#print(df_filenames)

i=1
while i<Num_cols+1:
    filename=df_filenames.iloc[i-1,0]
    x=file[filename]
#    print(x)
    shape, loc, scale = gev.fit(x)
#    print(shape)
#    print(loc)
#    print(scale)
    d = {'Shape': [shape], 'Location': [loc], 'Scale': [scale]}
    parameters_df = pd.DataFrame(data=d)
    parameters_df.rename(index={0:'Observed'}, inplace=True)
#    print(parameters_df)
    file_shuffled=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\4. Results for the Shuffled of the timeseries with criteria\\'+threshold+'\\'+filename+' Annual Peak  for 100 Shuffled')
    file_shuffled=file_shuffled.drop(columns=['Year'])
    j=0
    while j<100:
        y=file_shuffled.iloc[:,j]
        shape, loc, scale = gev.fit(y)
        d = {'Shape': [shape], 'Location': [loc], 'Scale': [scale]}
        parameters_df1 = pd.DataFrame(data=d)
        k=j+1
        k=str(k)
        k='Shuffled '+k
        parameters_df1.rename(index={0:k}, inplace=True)
        parameters_df=parameters_df.append(parameters_df1)
        j+=1
#    print(parameters_df)
#    print(shape, loc, scale)
    parameters_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\GEV Parameters\\'+filename+' GEV Parameters')
    i+=1

#ECDF tables for GEV distribution
i=1
while i<=Num_cols:
```

```
    GEV_ECDF={'abc': [0.0],'def': [0.0]}
    GEV_ECDF = pd.DataFrame(data=GEV_ECDF)
    GEV_ECDF.index=['0.0']
    filename=df_filenames.iloc[i-1,0]
    file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\GEV Parameters\\'+filename+' GEV Parameters')
    Num_cols1 = len (file1)
    j=0
    while j<Num_cols1:
        if j==0:
            GEV_ECDF1=GEV_ECDF
            c=file1.iloc[j,0]
            l=file1.iloc[j,1]
            s=file1.iloc[j,2]
            k=0
            for ecdf_value in np.arange(0.03,0.97,0.01):
                annual_peak_value=gev.ppf(ecdf_value, c, l, s)
                GEV_ECDF1.loc[k,0]=annual_peak_value
                GEV_ECDF1.loc[k,1]=ecdf_value
                k+=1
            annual_peak_value=gev.ppf(0.972222, c, l, s)
            GEV_ECDF1.loc[k,0]=annual_peak_value
            GEV_ECDF1.loc[k,1]=0.972222
            GEV_ECDF1=GEV_ECDF1.drop(['abc'], axis=1)
            GEV_ECDF1=GEV_ECDF1.drop(['def'], axis=1)
            GEV_ECDF1=GEV_ECDF1.drop(['0.0'], axis=0)
            GEV_ECDF1.columns = ['Large annual peak OBS', 'ECDF OBS']
        else:
            GEV_ECDF2=GEV_ECDF
            c=file1.iloc[j,0]
            l=file1.iloc[j,1]
            s=file1.iloc[j,2]
            k=0
            for ecdf_value in np.arange(0.03,0.97,0.01):
                annual_peak_value=gev.ppf(ecdf_value, c, l, s)
                GEV_ECDF.loc[k,0]=annual_peak_value
                GEV_ECDF.loc[k,1]=ecdf_value
                k+=1
            annual_peak_value=gev.ppf(0.972222, c, l, s)
            GEV_ECDF2.loc[k,0]=annual_peak_value
            GEV_ECDF2.loc[k,1]=0.972222
            GEV_ECDF2=GEV_ECDF2.drop(['abc'], axis=1)
            GEV_ECDF2=GEV_ECDF2.drop(['def'], axis=1)
            GEV_ECDF2=GEV_ECDF2.drop(['0.0'], axis=0)
            GEV_ECDF2.columns = ['Large annual peak SHU', 'ECDF SHU']
            z=str(j)
            GEV_ECDF1['LARGE '+z]=GEV_ECDF2['Large annual peak SHU']
            GEV_ECDF1['ECDF '+z]=GEV_ECDF2['ECDF SHU']
        j+=1

    GEV_ECDF1.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\ECDF files for GEV\\'+filename+' GEV ECDF')
#    print(GEV_ECDF1)
    i+=1

#Mean Shape, Mean Location and Mean Scale Parameters
i=1
while i<=Num_cols:
    filename=df_filenames.iloc[i-1,0]
    file_GEV_Parameters=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\GEV Parameters\\'+filename+' GEV Parameters')
    Mean_shape=file_GEV_Parameters.iloc[1:,0].mean()
    Mean_location=file_GEV_Parameters.iloc[1:,1].mean()
    Mean_scale=file_GEV_Parameters.iloc[1:,2].mean()
    d = {'Mean Shuffled Shape': [Mean_shape], 'Mean Shuffled Location': [Mean_location], 'Mean Shuffled Scale': [Mean_scale]}
    parameters_df1 = pd.DataFrame(data=d)
    parameters_df1.rename(index={0:filename}, inplace=True)
#    print(parameters_df1)
    parameters_df1.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\Mean Shuffled GEV Parameters\\'+filename+' Mean Shuffled GEV Parameters')
    i+=1

#ECDF tables for Mean Shape, Mean Location and Mean Scale
```

```
i=1
while i<=Num_cols:
    GEV_ECDF={'abc': [0.0],'def': [0.0]}
    GEV_ECDF = pd.DataFrame(data=GEV_ECDF)
    GEV_ECDF.index=['0.0']
    filename=df_filenames.iloc[i-1,0]
    file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\Mean Shuffled GEV Parameters\\'+filename+' Mean Shuffled GEV Parameters')
    GEV_ECDF1=GEV_ECDF
    c=file1.iloc[0,0]
    l=file1.iloc[0,1]
    s=file1.iloc[0,2]
    k=0
    for ecdf_value in np.arange(0.03,0.97,0.01):
        annual_peak_value=gev.ppf(ecdf_value, c, l, s)
        GEV_ECDF1.loc[k,0]=annual_peak_value
        GEV_ECDF1.loc[k,1]=ecdf_value
        k+=1
    annual_peak_value=gev.ppf(0.972222, c, l, s)
    GEV_ECDF1.loc[k,0]=annual_peak_value
    GEV_ECDF1.loc[k,1]=0.972222
    GEV_ECDF1=GEV_ECDF1.drop(['abc'], axis=1)
    GEV_ECDF1=GEV_ECDF1.drop(['def'], axis=1)
    GEV_ECDF1=GEV_ECDF1.drop(['0.0'], axis=0)
    GEV_ECDF1.columns = ['Large annual peak', 'ECDF']
    GEV_ECDF1.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\ECDF files for Mean Shuffled GEV Parameters\\'+filename+' Mean Shuffled GEV ECDF')
#    print(GEV_ECDF1)
    i+=1

# Plot of ECDF of GEV distribution of annual peak and the ones of observed and shuffled timeseries
df_filenames=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Timeseries 1980-2014 and 10% limit for missing values')
Num_cols_filenames = len (df_filenames)
i=0
while i<Num_cols_filenames:
    filename=df_filenames.iloc[i,0]
    filename1=filename.replace(".mat","")
    df_filenames.iloc[i,0]=filename1
    i+=1

HU_df=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Gauge_info - 360 stations')
HU_df=HU_df.drop("DRAINAGE AREA (KM^2)", axis=1)
HU_df=HU_df.drop("LONG", axis=1)
HU_df=HU_df.drop("LAT", axis=1)
Num=len(HU_df)
#print(HU_df)
i=0
while i<Num:
    filename=df_filenames.iloc[i,0]
    filename1=filename.replace(".mat","")
    if HU_df.iloc[i,1]==filename1:
        i+=1
    else:
        HU_df_df=HU_df.drop(HU_df.index[i])
        Num=Num-1
        HU_df=HU_df.reset_index(drop=True)
#print(HU_df)


#Export ECDF-Annual peak diagram
j=1
while j<=Num_cols_filenames:
    filename=df_filenames.iloc[j-1,0]
    HU=str(HU_df.iloc[j-1,0])
    Gauge_name=str.upper(HU_df.iloc[j-1,2])
#    print(filename)
    file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\Shuffled\\'+filename+' ECDF annual maxima SHUFFLED')
    file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\Observed\\'+filename+' ECDF annual maxima OBSERVED')
    file3=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\ECDF files for GEV\\'+filename+' GEV ECDF')
```

```
    file4=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\ECDF files for Mean Shuffled GEV Parameters\\'+filename+' Mean Shuffled GEV ECDF')
    sns.set()
    sns.set_style("ticks")
    plt.figure(figsize=(8,5),dpi=200)
    for z in  np.arange(3,202,2):
        ecdf=file3.iloc[:,z]
        annual_peak=file3.iloc[:,z-1]
        plt.plot(annual_peak,ecdf,label='',color='gainsboro')
    plt.plot(annual_peak,ecdf,label='GEV shuffled',color='gainsboro')
    plt.plot(file4.iloc[:,0],file4.iloc[:,1],label='Mean GEV Shuffled', color='blue')
    plt.plot(file3.iloc[:,0],file3.iloc[:,1],label='GEV observed',color='g')
    plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='Observed', marker='o', markersize=3, color='r')

    plt.title(Gauge_name+'\nGauge ID: '+filename+' - Hydrological Unit: '+HU,fontsize=13, y=1.01)
    plt.ylabel("ECDF",fontsize=14)
    plt.xlabel("Annual peak - Q ($m^3/sec$)",fontsize=14)
    plt.yticks(np.arange(0, 1.1, 0.1))
    plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
    plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(0.27, 1),loc=1,ncol=1,fontsize=10)
    plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
    plt.tight_layout()
    plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\Plots\\ECDF diagrams Linear\\ECDF_Linear_annual_peak_with_GEV '+filename+'.png',facecolour='white')
    plt.close()
    j+=1


#Export LOG-LOG ECDF-Annual peak diagram
j=1
while j<=Num_cols_filenames:
    filename=df_filenames.iloc[j-1,0]
    HU=str(HU_df.iloc[j-1,0])
    Gauge_name=str.upper(HU_df.iloc[j-1,2])
#    print(filename)
    file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\Shuffled\\'+filename+' ECDF annual maxima SHUFFLED')
    file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\Observed\\'+filename+' ECDF annual maxima OBSERVED')
    file3=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\ECDF files for GEV\\'+filename+' GEV ECDF')
    file4=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\ECDF files for Mean Shuffled GEV Parameters\\'+filename+' Mean Shuffled GEV ECDF')
    sns.set()
    sns.set_style("ticks")
    plt.figure(figsize=(8,5),dpi=200)
    for z in  np.arange(3,202,2):
        ecdf=file3.iloc[:,z]
        annual_peak=file3.iloc[:,z-1]
        plt.plot(annual_peak,ecdf,label='',color='gainsboro')
    plt.plot(annual_peak,ecdf,label='GEV shuffled',color='gainsboro')
    plt.plot(file4.iloc[:,0],file4.iloc[:,1],label='Mean GEV Shuffled', color='blue')
    plt.plot(file3.iloc[:,0],file3.iloc[:,1],label='GEV observed',color='g')
    plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='Observed', marker='o', markersize=3, color='r')

    plt.xscale('log')
    plt.yscale('log')
    labels = ['0.05', '0.1', '0.2', '0.5', '1.0']
    nthreads = [0.05, 0.1, 0.2, 0.5, 1.0]
    plt.yticks(nthreads, labels)
    plt.title(Gauge_name+'\nGauge ID: '+filename+' - Hydrological Unit: '+HU,fontsize=13, y=1.01)
    plt.ylabel("ECDF",fontsize=14)
    plt.xlabel("Annual peak - Q ($m^3/sec$)",fontsize=14)
    plt.grid(color='whitesmoke',b=True, which='major',axis='both', linestyle='-',zorder=2)
    plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(0.27, 1),loc=1,ncol=1,fontsize=10)
    plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
    plt.tight_layout()
    plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\Plots\\ECDF diagrams LOG\\ECDF_LOG_annual_peak_with_GEV '+filename+'.png',facecolour='white')
    plt.close()
    j+=1


#Export 1/(1-f) ECDF diagram
```

```
j=1
while j<=Num_cols_filenames:
    filename=df_filenames.iloc[j-1,0]
    HU=str(HU_df.iloc[j-1,0])
    Gauge_name=str.upper(HU_df.iloc[j-1,2])
#    print(filename)
    file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\Shuffled \\'+filename+' ECDF annual maxima SHUFFLED')
    file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\Observed \\'+filename+' ECDF annual maxima OBSERVED')
    file3=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\ECDF fil es for GEV\\'+filename+' GEV ECDF')
    file4=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\ECDF fil es for Mean Shuffled GEV Parameters\\'+filename+' Mean Shuffled GEV ECDF')
    sns.set()
    sns.set_style("ticks")
    plt.figure(figsize=(8,5),dpi=200)
    for z in  np.arange(3,202,2):
        ecdf=1/(1-file3.iloc[:,z])
        annual_peak=file3.iloc[:,z-1]
        plt.plot(annual_peak,ecdf,label='',color='gainsboro')
    plt.plot(annual_peak,ecdf,label='GEV shuffled',color='gainsboro')
    plt.plot(file4.iloc[:,0],1/(1-file4.iloc[:,1]),label='Mean GEV Shuffled', color='blue')
    plt.plot(file3.iloc[:,0],1/(1-file3).iloc[:,1],label='GEV observed',color='g')
    plt.plot(file2.iloc[:,0],1/(1-
file2).iloc[:,1],label='Observed', marker='o', markersize=3, color='r')
    plt.title(Gauge_name+'\nGauge ID: '+filename+' - Hydrological Unit: '+HU,fontsize=13, y=1.01)
    plt.ylabel("1/(1-ECDF)",fontsize=14)
    plt.xlabel("Annual peak - Q ($m^3/sec$)",fontsize=14)
    plt.grid(color='whitesmoke',b=True, which='major',axis='both', linestyle='-',zorder=2)
    plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(0.27, 1),loc=1,ncol=1,fontsize=10)
    plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
    plt.tight_layout()
    plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\7. GENEXTREME\\Plots\\ECDF diagr ams 1--(1-ECDF)\\ECDF_1--(1-ECDF)_annual_peak_with_GEV '+filename+'.png',facecolour='white')
    plt.close()
    j+=1
```

11.9. ECDF plot of observed time series and Generalized Pareto distribution for all gauge locations and all thresholds.

```
import numpy as np
import pandas as pd
import scipy
import seaborn as sns
from matplotlib import pyplot as plt
from scipy.stats import genpareto

def Get_Matlab_file_TS(filename):
    mat=scipy.io.loadmat("C:\\Users\\Konstantinos\\Documents\\THESIS\\Timeseries\\"+filename+".mat")

    a= mat['TS']
    df_TS=pd.DataFrame(a)
    return df_TS

def Get_events_df(df_TS, threshold):
    df_withoutNaN=df_TS.dropna(axis=0,how='any', thresh=None, subset=None, inplace=False)
    df_sorted_noNaN=df_withoutNaN.sort_values(3, axis=0, ascending=True, inplace=False, kind='quicksо rt', na_position='last')
    Num_cols = len (df_withoutNaN)
    position=int(threshold*Num_cols)
    threshold_value=df_sorted_noNaN.iloc[position,3]
    events_df=df_withoutNaN
    events_df.columns = ['Year', 'Month', 'Day', 'Q']
    events_df.drop(events_df.loc[events_df['Q'] < threshold_value].index, inplace=True)
    return events_df

def Get_events_df_without_ones(events_df):
    Num_cols = len (events_df)
    events_df_without_ones=events_df.take([0])
    i=1
```

```
    while i<=Num_cols-1:
        if events_df.index[i]-1==events_df.index[i-1]:
            i+=1
        else:
            append_row=events_df.iloc[i].copy()
            events_df_without_ones=events_df_without_ones.append(append_row)
            i+=1
    return events_df_without_ones

def ecdf(file):
    ecdf_df=file
    ecdf_df = pd.DataFrame(ecdf_df)
    ecdf_df.columns = ['Large sorted']
    ecdf_df=ecdf_df.sort_values(by=['Large sorted'],ascending=False)
    ecdf_df=ecdf_df.reset_index(drop=True)
    Num_cols=len(ecdf_df)
    x=ecdf_df.copy()
    ecdf_df['ECDF']=x['Large sorted']
    j=0
    while j<Num_cols:
        ecdf_df.iloc[j,1]=1-((ecdf_df.index[j]+1)/(Num_cols+1))
        j+=1
    return ecdf_df

df_filenames=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Timeseries 1980-2014 and 10% limit for missing values')
Num_cols_filenames = len (df_filenames)
i=0
while i<Num_cols_filenames:
    filename=df_filenames.iloc[i,0]
    filename1=filename.replace(".mat","")
    df_filenames.iloc[i,0]=filename1
    i+=1

HU_df=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Gauge_info - 360 stations')
HU_df=HU_df.drop("DRAINAGE AREA (KM^2)", axis=1)
HU_df=HU_df.drop("LONG", axis=1)
HU_df=HU_df.drop("LAT", axis=1)
Num=len(HU_df)
#print(HU_df)
i=0
while i<Num:
    filename=df_filenames.iloc[i,0]
    filename1=filename.replace(".mat","")
    if HU_df.iloc[i,1]==filename1:
        i+=1
    else:
        HU_df_df=HU_df.drop(HU_df.index[i])
        Num=Num-1
        HU_df=HU_df.reset_index(drop=True)
#print(HU_df)

threshold_df_data = {'Text': ['90%','95%','98%','99%','99.5%'],'Numerical': [0.9, 0.95, 0.98, 0.99, 0.995]}
threshold_df = pd.DataFrame(data=threshold_df_data)
Num_threshold_df=len(threshold_df)
#print(threshold_df)

threshold_loop=0
while threshold_loop<Num_threshold_df-1:
    threshold=threshold_df.iloc[threshold_loop,1]
    threshold_text=threshold_df.iloc[threshold_loop,0]

    i=0
    while i<Num_cols_filenames:
        filename=df_filenames.iloc[i,0]
        df_TS=Get_Matlab_file_TS(filename)
        events_df=Get_events_df(df_TS, threshold)
        events_df1=events_df
        events_df_without_ones=Get_events_df_without_ones(events_df1)
        events_df_without_ones=events_df_without_ones.drop(['Year'], axis=1)
        events_df_without_ones=events_df_without_ones.drop(['Month'], axis=1)
        events_df_without_ones=events_df_without_ones.drop(['Day'], axis=1)
```

```
        events_df_without_ones=events_df_without_ones.reset_index(drop=True)
        events_df_without_ones.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\'+threshold_text+'\\Events dataframes\\'+filename+' Events without ones')
    #    print(events_df_without_ones)
        i+=1

    i=0
    while i<Num_cols_filenames:
        filename=df_filenames.iloc[i,0]
        file = pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\'+threshold_text+'\\Events dataframes\\'+filename+' Events without ones')
        ecdf1=ecdf(file)
        ecdf1.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\'+threshold_text+'\\ECDF files for observed\\'+filename+' ECDF without ones OBSERVED')
    #    print(ecdf1)
        i+=1

    i=1
    while i<=Num_cols_filenames:
        filename=df_filenames.iloc[i-1,0]
        x=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\'+threshold_text+'\\Events dataframes\\'+filename+' Events without ones')
        shape, loc, scale = genpareto.fit(x)
        d = {'Shape': [shape], 'Location': [loc], 'Scale': [scale]}
        parameters_df = pd.DataFrame(data=d)
        parameters_df.rename(index={0:'Observed'}, inplace=True)
        parameters_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\'+threshold_text+'\\GPar Parameters\\'+filename+' GPar Parameters')
        i+=1

    i=1
    while i<=Num_cols_filenames:
        GPar_ECDF={'abc': [0.0],'def': [0.0]}
        GPar_ECDF = pd.DataFrame(data=GPar_ECDF)
        GPar_ECDF.index=['0.0']
        filename=df_filenames.iloc[i-1,0]
        file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\'+threshold_text+'\\GPar Parameters\\'+filename+' GPar Parameters')
        c=file1.iloc[0,0]
        l=file1.iloc[0,1]
        s=file1.iloc[0,2]
        k=0
        for ecdf_value in np.arange(0.00,1,0.01):
            annual_peak_value=genpareto.ppf(ecdf_value, c, l, s)
            GPar_ECDF.loc[k,0]=annual_peak_value
            GPar_ECDF.loc[k,1]=ecdf_value
            k+=1
        annual_peak_value=genpareto.ppf(0.999, c, l, s)
        GPar_ECDF.loc[k,0]=annual_peak_value
        GPar_ECDF.loc[k,1]=0.999
        GPar_ECDF=GPar_ECDF.drop(['abc'], axis=1)
        GPar_ECDF=GPar_ECDF.drop(['def'], axis=1)
        GPar_ECDF=GPar_ECDF.drop(['0.0'], axis=0)
        GPar_ECDF.columns = ['Large GPar', 'ECDF GPar']
        GPar_ECDF.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\'+threshold_text+'\\ECDF files for GPar\\'+filename+' GPar ECDF')
    #    print(GPar_ECDF)
        i+=1

#Export Linear ECDF-Observed and GPar for ones and without ones
i=1
while i<=Num_cols_filenames:
    filename=df_filenames.iloc[i-1,0]
    HU=str(HU_df.iloc[i-1,0])
    Gauge_name=str.upper(HU_df.iloc[i-1,2])
    file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\90%\\ECDF files for observed\\'+filename+' ECDF without ones OBSERVED')
    file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\90%\\ECDF files for GPar\\'+filename+' GPar ECDF')
    file3=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\95%\\ECDF files for observed\\'+filename+' ECDF without ones OBSERVED')
```

```
    file4=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\95%\\ECDF files for GPar\\'+filename+' GPar ECDF')
    file5=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\98%\\ECDF files for observed\\'+filename+' ECDF without ones OBSERVED')
    file6=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\98%\\ECDF files for GPar\\'+filename+' GPar ECDF')
    file7=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\99%\\ECDF files for observed\\'+filename+' ECDF without ones OBSERVED')
    file8=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\99%\\ECDF files for GPar\\'+filename+' GPar ECDF')

    sns.set()
    sns.set_style("ticks")
    plt.figure(figsize=(8,5),dpi=200)
    plt.plot(file1.iloc[:,0],file1.iloc[:,1],label='Observed 90%', linewidth=1.1, color='skyblue')
    plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='GPar 90%', color='blue')
    plt.plot(file3.iloc[:,0],file3.iloc[:,1],label='Observed 95%', linewidth=1.1, color='lime')
    plt.plot(file4.iloc[:,0],file4.iloc[:,1],label='GPar 95%', color='green')
    plt.plot(file5.iloc[:,0],file5.iloc[:,1],label='Observed 98%', linewidth=1.1, color='lightsalmon')
    plt.plot(file6.iloc[:,0],file6.iloc[:,1],label='GPar 98%', color='red')
    plt.plot(file7.iloc[:,0],file7.iloc[:,1],label='Observed 99%', linewidth=1.1, color='pink')
    plt.plot(file8.iloc[:,0],file8.iloc[:,1],label='GPar 99%', color='purple')

    axes = plt.gca()
    max_value=max(file1.iloc[:,0].max(),file3.iloc[:,0].max(),file5.iloc[:,0].max(),file7.iloc[:,0].max())
    axes.set_xlim([0,max_value*1.1])
    plt.title(Gauge_name+'\nGauge ID: '+filename+' - Hydrological Unit: '+HU,fontsize=13, y=1.01)
    plt.ylabel("ECDF",fontsize=12);
    plt.xlabel("Q ($m^3/sec$) for the over-threshold events",fontsize=12);
    plt.yticks(np.arange(0, 1.1, 0.1))
    plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
    plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(0.992, 0.55),loc=1,ncol=1,fontsize=9)
    plt.tick_params(axis='both',which='both',labelsize=12,zorder=20)
    plt.tight_layout()
    plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\Plots for without ones for all thresholds\\ECDF diagrams Linear\\'+filename+' ECDF GPar Linear for all thresholds')
    plt.close()
    i+=1

#Export LOG ECDF-Observed and GPar for ones and without ones
i=1
while i<=Num_cols_filenames:
    filename=df_filenames.iloc[i-1,0]
    HU=str(HU_df.iloc[i-1,0])
    Gauge_name=str.upper(HU_df.iloc[i-1,2])
    file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\90%\\ECDF files for observed\\'+filename+' ECDF without ones OBSERVED')
    file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\90%\\ECDF files for GPar\\'+filename+' GPar ECDF')
    file3=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\95%\\ECDF files for observed\\'+filename+' ECDF without ones OBSERVED')
    file4=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\95%\\ECDF files for GPar\\'+filename+' GPar ECDF')
    file5=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\98%\\ECDF files for observed\\'+filename+' ECDF without ones OBSERVED')
    file6=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\98%\\ECDF files for GPar\\'+filename+' GPar ECDF')
    file7=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\99%\\ECDF files for observed\\'+filename+' ECDF without ones OBSERVED')
    file8=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\events without ones\\99%\\ECDF files for GPar\\'+filename+' GPar ECDF')

    sns.set()
    sns.set_style("ticks")
    plt.figure(figsize=(8,5),dpi=200)
    plt.plot(file1.iloc[:,0],file1.iloc[:,1],label='Observed 90%', linewidth=1.1, color='skyblue')
    plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='GPar 90%', linewidth=1.1, color='blue')
    plt.plot(file3.iloc[:,0],file3.iloc[:,1],label='Observed 95%', linewidth=1.1, color='lime')
    plt.plot(file4.iloc[:,0],file4.iloc[:,1],label='GPar 95%', linewidth=1.1, color='green')
```

```
    plt.plot(file5.iloc[:,0],file5.iloc[:,1],label='Observed 98%', linewidth=1.1, color='lightsalmon'
)
    plt.plot(file6.iloc[:,0],file6.iloc[:,1],label='GPar 98%', linewidth=1.1, color='red')
    plt.plot(file7.iloc[:,0],file7.iloc[:,1],label='Observed 99%', linewidth=1.1, color='pink')
    plt.plot(file8.iloc[:,0],file8.iloc[:,1],label='GPar 99%', linewidth=1.1, color='purple')

    plt.xscale('log')
    plt.yscale('log')
    labels = ['0.05', '0.1', '0.2', '0.5', '1.0']
    nthreads = [0.05, 0.1, 0.2, 0.5, 1.0]
    plt.yticks(nthreads, labels)
    axes = plt.gca()
    axes.set_ylim([0.025,1.2])
    plt.title(Gauge_name+'\nGauge ID: '+filename+' -
 Hydrological Unit: '+HU,fontsize=13, y=1.01)
    plt.ylabel("ECDF",fontsize=12);
    plt.xlabel("Q ($m^3/sec$) for the over-threshold events",fontsize=12);
    plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
    plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(0.992, 0.55),loc=1,ncol=1,fontsize=9)
    plt.tick_params(axis='both',which='both',labelsize=12,zorder=20)
    plt.tight_layout()
    plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\Plots for
 without ones for all thresholds\\ECDF diagrams LOG\\'+filename+' ECDF GPar LOG for all thresholds')

    plt.close()
    i+=1

#Export 1/(1-f) ECDF-Observed and GPar for ones and without ones
i=1
while i<=Num_cols_filenames:
    filename=df_filenames.iloc[i-1,0]
    HU=str(HU_df.iloc[i-1,0])
    Gauge_name=str.upper(HU_df.iloc[i-1,2])
    file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\
events without ones\\90%\\ECDF files for observed\\'+filename+' ECDF without ones OBSERVED')
    file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\
events without ones\\90%\\ECDF files for GPar\\'+filename+' GPar ECDF')
    file3=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\
events without ones\\95%\\ECDF files for observed\\'+filename+' ECDF without ones OBSERVED')
    file4=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\
events without ones\\95%\\ECDF files for GPar\\'+filename+' GPar ECDF')
    file5=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\
events without ones\\98%\\ECDF files for observed\\'+filename+' ECDF without ones OBSERVED')
    file6=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\
events without ones\\98%\\ECDF files for GPar\\'+filename+' GPar ECDF')
    file7=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\
events without ones\\99%\\ECDF files for observed\\'+filename+' ECDF without ones OBSERVED')
    file8=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\
events without ones\\99%\\ECDF files for GPar\\'+filename+' GPar ECDF')

    sns.set()
    sns.set_style("ticks")
    plt.figure(figsize=(8,5),dpi=200)
    plt.plot(file1.iloc[:,0],1/(1-
file1).iloc[:,1],label='Observed 90%', linewidth=1.1, color='skyblue')
    plt.plot(file2.iloc[:,0],1/(1-file2).iloc[:,1],label='GPar 90%', linewidth=1.1, color='blue')
    plt.plot(file3.iloc[:,0],1/(1-
file3).iloc[:,1],label='Observed 95%', linewidth=1.1, color='lime')
    plt.plot(file4.iloc[:,0],1/(1-file4).iloc[:,1],label='GPar 95%', linewidth=1.1, color='green')
    plt.plot(file5.iloc[:,0],1/(1-
file5).iloc[:,1],label='Observed 98%', linewidth=1.1, color='lightsalmon')
    plt.plot(file6.iloc[:,0],1/(1-file6).iloc[:,1],label='GPar 98%', linewidth=1.1, color='red')
    plt.plot(file7.iloc[:,0],1/(1-
file7).iloc[:,1],label='Observed 99%', linewidth=1.1, color='pink')
    plt.plot(file8.iloc[:,0],1/(1-file8).iloc[:,1],label='GPar 99%', linewidth=1.1, color='purple')

    plt.title(Gauge_name+'\nGauge ID: '+filename+' -
 Hydrological Unit: '+HU,fontsize=13, y=1.01)
    plt.ylabel("1/(1-ECDF)",fontsize=12);
    plt.xlabel("Q ($m^3/sec$) for the over-threshold events",fontsize=12);
    plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
    plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(0.992, 0.55),loc=1,ncol=1,fontsize=9)
    plt.tick_params(axis='both',which='both',labelsize=12,zorder=20)
    plt.tight_layout()
```

```python
    plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\8. Generalized Pareto\\Plots for without ones for all thresholds\\ECDF diagrams 1--(1-ECDF)\\'+filename+' ECDF GPar ECDF diagrams 1--(1-ECDF) for all thresholds')
    plt.close()
    i+=1
```

11.10. Distribution of FEMA NFIP claims records per County.

```python
import pandas as pd

#Read FEMA records dataframe
FEMA_file=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\9. FEMA records\\FEMA records dataframe')
FEMA_file=FEMA_file[['countycode','state','yearofloss']]
#print(FEMA_file)

#Number of NaN values per column
print('countycode column missing values: ',FEMA_file['countycode'].isnull().sum().sum())
print('state column missing values: ',FEMA_file['state'].isnull().sum().sum())
print('yearofloss column missing values: ',FEMA_file['yearofloss'].isnull().sum().sum())

FEMA_file=FEMA_file.dropna() #Drop rows with NaN values
#print(FEMA_file)

#print(FEMA_file['countycode'])
FEMA_file['countycode']=FEMA_file['countycode'].astype(int, errors='ignore') #Convert floats to integers
#print(FEMA_file['countycode'])

FEMA_dataframe_Num=len(FEMA_file) #Final number of rows of modified FEMA records' file

d = {'Year': [1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1988,1989,1990,1991,1992,1993,1994,1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010,2011,2012,2013,2014,2015,2016,2017,2018,2019], 'Claims': [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]}
draft_dataframe = pd.DataFrame(data=d)
Num_draft_dataframe=len(draft_dataframe)
#print(draft_dataframe)

#Claims per county (1970-2019)
x=FEMA_file.groupby('countycode').size().reset_index()
#print(x)

county_names_df=x.iloc[:,0]

#Aggregate claims per county per year (1970-2019)
x=FEMA_file.groupby(['countycode','yearofloss']).size().reset_index()
#print(x)

county_claims_matrix=draft_dataframe.copy()
county_claims_matrix=county_claims_matrix.drop(['Claims'], axis=1)
i=0
while i<len(county_names_df):
    county_name=county_names_df.iloc[i]
    county_claims_matrix[county_name]=draft_dataframe['Claims']
    j=0
    while j<len(x):
        if x.iloc[j,0]==county_name:
            year=x.iloc[j,1]
            z=0
            while z<len(county_claims_matrix):
                if year==county_claims_matrix.iloc[z,0]:
                    county_claims_matrix.iloc[z,i+1]=x.iloc[j,2]
                z+=1
        j+=1
    i+=1

county_claims_matrix=county_claims_matrix.drop(county_claims_matrix.index[49])

county_claims_matrix.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\FEMA claims for COUNTIES\\Claims per county df')
```

11.11. Distribution of FEMA NFIP claims records per State.

```
import pandas as pd

#Read FEMA records dataframe
FEMA_file=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\9. FEMA records\\FEMA records dataframe')
FEMA_file=FEMA_file[['countycode','state','yearofloss']]
#print(FEMA_file)

#Number of NaN values per column
print('countycode column missing values: ',FEMA_file['countycode'].isnull().sum().sum())
print('state column missing values: ',FEMA_file['state'].isnull().sum().sum())
print('yearofloss column missing values: ',FEMA_file['yearofloss'].isnull().sum().sum())

FEMA_file=FEMA_file.dropna() #Drop rows with NaN values
#print(FEMA_file)

#print(FEMA_file['countycode'])
FEMA_file['countycode']=FEMA_file['countycode'].astype(int, errors='ignore') #Convert floats to integers
#print(FEMA_file['countycode'])

FEMA_dataframe_Num=len(FEMA_file) #Final number of rows of modified FEMA records' file

d = {'Year': [1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1988,1989,1990,1991,1992,1993,1994,1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010,2011,2012,2013,2014,2015,2016,2017,2018,2019], 'Claims': [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]}
draft_dataframe = pd.DataFrame(data=d)
Num_draft_dataframe=len(draft_dataframe)
#print(draft_dataframe)

#Claims per county (1970-2019)
x=FEMA_file.groupby('state').size().reset_index()
#print(x)

states_names_df=x.iloc[:,0]

#Aggregate claims per county per year (1970-2019)
x=FEMA_file.groupby(['state','yearofloss']).size().reset_index()
#print(x)

states_claims_matrix=draft_dataframe.copy()
states_claims_matrix=states_claims_matrix.drop(['Claims'], axis=1)
i=0
while i<len(states_names_df):
    states_name=states_names_df.iloc[i]
    states_claims_matrix[states_name]=draft_dataframe['Claims']
    j=0
    while j<len(x):
        if x.iloc[j,0]==states_name:
            year=x.iloc[j,1]
            z=0
            while z<len(states_claims_matrix):
                if year==states_claims_matrix.iloc[z,0]:
                    states_claims_matrix.iloc[z,i+1]=x.iloc[j,2]
                z+=1
        j+=1
    i+=1

states_claims_matrix=states_claims_matrix.drop(states_claims_matrix.index[49])

states_claims_matrix.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\FEMA claims for STATES\\Claims per state df')
```

11.12. Distribution of FEMA NFIP claims, paid claims, amount paid on building claims and amount paid on contents claims records per Hydrological Unit (region).

```python
import geopandas as gpd
import pandas as pd
import math
from shapely.geometry import Point

#Read FEMA records dataframe
FEMA_file=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\9. FEMA records\\FEMA records dataframe')
print(FEMA_file)
print(FEMA_file.columns)
FEMA_dataframe_Num=len(FEMA_file)

print(FEMA_file['latitude'].isnull().sum().sum())
print(FEMA_file['longitude'].isnull().sum().sum())
print(FEMA_file['countycode'].isnull().sum().sum())
print(FEMA_file['amountpaidonbuildingclaim'].isnull().sum().sum())
print(FEMA_file['amountpaidoncontentsclaim'].isnull().sum().sum())
print(FEMA_file['state'].isnull().sum().sum())
print(FEMA_file['yearofloss'].isnull().sum().sum())

d = {'Year': [1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1988,1989,1990,1991,1992,1993,1994,1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010,2011,2012,2013,2014,2015,2016,2017,2018,2019], 'Claims': [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]}
Claims_dataframe = pd.DataFrame(data=d)
#print(Claims_dataframe)
d = {'Year': [1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1988,1989,1990,1991,1992,1993,1994,1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010,2011,2012,2013,2014,2015,2016,2017,2018,2019], 'Paid claims': [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]}
Paid_claims_dataframe = pd.DataFrame(data=d)
#print(Paid_claims_dataframe)
d = {'Year': [1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1988,1989,1990,1991,1992,1993,1994,1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010,2011,2012,2013,2014,2015,2016,2017,2018,2019], 'Amount paid on building claims': [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]}
Amount_paid_on_building_claims_dataframe = pd.DataFrame(data=d)
#print(Amount_paid_on_building_claims_dataframe)
d = {'Year': [1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1988,1989,1990,1991,1992,1993,1994,1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010,2011,2012,2013,2014,2015,2016,2017,2018,2019], 'Amount paid on contents claims': [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]}
Amount_paid_on_contents_claims_dataframe = pd.DataFrame(data=d)
#print(Amount_paid_on_contents_claims_dataframe)
Num_dataframe=len(Claims_dataframe)

#Dataframes for Region 1
Claims_dataframe_Region_1=Claims_dataframe.copy()
Paid_claims_dataframe_Region_1=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_1=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_1=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 2
Claims_dataframe_Region_2=Claims_dataframe.copy()
Paid_claims_dataframe_Region_2=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_2=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_2=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 3
Claims_dataframe_Region_3=Claims_dataframe.copy()
Paid_claims_dataframe_Region_3=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_3=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_3=Amount_paid_on_contents_claims_dataframe.copy()

```

```
#Dataframes for Region 4
Claims_dataframe_Region_4=Claims_dataframe.copy()
Paid_claims_dataframe_Region_4=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_4=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_4=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 5
Claims_dataframe_Region_5=Claims_dataframe.copy()
Paid_claims_dataframe_Region_5=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_5=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_5=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 6
Claims_dataframe_Region_6=Claims_dataframe.copy()
Paid_claims_dataframe_Region_6=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_6=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_6=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 7
Claims_dataframe_Region_7=Claims_dataframe.copy()
Paid_claims_dataframe_Region_7=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_7=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_7=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 8
Claims_dataframe_Region_8=Claims_dataframe.copy()
Paid_claims_dataframe_Region_8=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_8=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_8=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 9
Claims_dataframe_Region_9=Claims_dataframe.copy()
Paid_claims_dataframe_Region_9=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_9=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_9=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 10
Claims_dataframe_Region_10=Claims_dataframe.copy()
Paid_claims_dataframe_Region_10=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_10=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_10=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 11
Claims_dataframe_Region_11=Claims_dataframe.copy()
Paid_claims_dataframe_Region_11=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_11=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_11=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 12
Claims_dataframe_Region_12=Claims_dataframe.copy()
Paid_claims_dataframe_Region_12=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_12=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_12=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 13
Claims_dataframe_Region_13=Claims_dataframe.copy()
Paid_claims_dataframe_Region_13=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_13=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_13=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 14
Claims_dataframe_Region_14=Claims_dataframe.copy()
Paid_claims_dataframe_Region_14=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_14=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_14=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 15
Claims_dataframe_Region_15=Claims_dataframe.copy()
Paid_claims_dataframe_Region_15=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_15=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_15=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 16
Claims_dataframe_Region_16=Claims_dataframe.copy()
```

```
Paid_claims_dataframe_Region_16=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_16=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_16=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 17
Claims_dataframe_Region_17=Claims_dataframe.copy()
Paid_claims_dataframe_Region_17=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_17=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_17=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 18
Claims_dataframe_Region_18=Claims_dataframe.copy()
Paid_claims_dataframe_Region_18=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_18=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_18=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 19
Claims_dataframe_Region_19=Claims_dataframe.copy()
Paid_claims_dataframe_Region_19=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_19=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_19=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 20
Claims_dataframe_Region_20=Claims_dataframe.copy()
Paid_claims_dataframe_Region_20=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_20=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_20=Amount_paid_on_contents_claims_dataframe.copy()

#Dataframes for Region 21
Claims_dataframe_Region_21=Claims_dataframe.copy()
Paid_claims_dataframe_Region_21=Paid_claims_dataframe.copy()
Amount_paid_on_building_claims_dataframe_Region_21=Amount_paid_on_building_claims_dataframe.copy()
Amount_paid_on_contents_claims_dataframe_Region_21=Amount_paid_on_contents_claims_dataframe.copy()

#Import shapefiles for all regions (HUs)
shapefile_region_1=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles\
    \Shapefiles Basin 1\\WBDHU2.shp')
shapefile_region_2=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles\
    \Shapefiles Basin 2\\WBDHU2.shp')
shapefile_region_3=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles\
    \Shapefiles Basin 3\\WBDHU2.shp')
shapefile_region_4=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles\
    \Shapefiles Basin 4\\WBDHU2.shp')
shapefile_region_5=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles\
    \Shapefiles Basin 5\\WBDHU2.shp')
shapefile_region_6=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles\
    \Shapefiles Basin 6\\WBDHU2.shp')
shapefile_region_7=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles\
    \Shapefiles Basin 7\\WBDHU2.shp')
shapefile_region_8=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles\
    \Shapefiles Basin 8\\WBDHU2.shp')
shapefile_region_9=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles\
    \Shapefiles Basin 9\\WBDHU2.shp')
shapefile_region_10=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles
    \\Shapefiles Basin 10\\WBDHU2.shp')
shapefile_region_11=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles
    \\Shapefiles Basin 11\\WBDHU2.shp')
shapefile_region_12=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles
    \\Shapefiles Basin 12\\WBDHU2.shp')
shapefile_region_13=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles
    \\Shapefiles Basin 13\\WBDHU2.shp')
shapefile_region_14=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles
    \\Shapefiles Basin 14\\WBDHU2.shp')
shapefile_region_15=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles
    \\Shapefiles Basin 15\\WBDHU2.shp')
shapefile_region_16=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles
    \\Shapefiles Basin 16\\WBDHU2.shp')
shapefile_region_17=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles
    \\Shapefiles Basin 17\\WBDHU2.shp')
shapefile_region_18=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles
    \\Shapefiles Basin 18\\WBDHU2.shp')
shapefile_region_19=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles
    \\Shapefiles Basin 19\\WBDHU2.shp')
```

```
shapefile_region_20=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles \\Shapefiles Basin 20\\WBDHU2.shp')
shapefile_region_21=gpd.read_file('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles \\Shapefiles Basin 21\\WBDHU2.shp')

polygon1=shapefile_region_1.iloc[0,15]
polygon2=shapefile_region_2.iloc[0,15]
polygon3=shapefile_region_3.iloc[0,15]
polygon4=shapefile_region_4.iloc[0,15]
polygon5=shapefile_region_5.iloc[0,15]
polygon6=shapefile_region_6.iloc[0,15]
polygon7=shapefile_region_7.iloc[0,15]
polygon8=shapefile_region_8.iloc[0,15]
polygon9=shapefile_region_9.iloc[0,15]
polygon10=shapefile_region_10.iloc[0,15]
polygon11=shapefile_region_11.iloc[0,15]
polygon12=shapefile_region_12.iloc[0,15]
polygon13=shapefile_region_13.iloc[0,15]
polygon14=shapefile_region_14.iloc[0,15]
polygon15=shapefile_region_15.iloc[0,15]
polygon16=shapefile_region_16.iloc[0,15]
polygon17=shapefile_region_17.iloc[0,15]
polygon18=shapefile_region_18.iloc[0,15]
polygon19=shapefile_region_19.iloc[0,15]
polygon20=shapefile_region_20.iloc[0,15]
polygon21=shapefile_region_21.iloc[0,15]

i=0
the_ones_that_were_in_no_watershed=0
while i<FEMA_dataframe_Num:
    year_of_loss=FEMA_file.iloc[i,36]
    longitude=FEMA_file.iloc[i,18]
    latitude=FEMA_file.iloc[i,16]
    p=Point(longitude,latitude)
    amount_paid_on_building_claim1=FEMA_file.iloc[i,27]
    if math.isnan(amount_paid_on_building_claim1)==True:
        amount_paid_on_building_claim=0.0
    else:
        amount_paid_on_building_claim=amount_paid_on_building_claim1
    amount_paid_on_contents_claim1=FEMA_file.iloc[i,28]
    if math.isnan(amount_paid_on_contents_claim1)==True:
        amount_paid_on_contents_claim=0.0
    else:
        amount_paid_on_contents_claim=amount_paid_on_contents_claim1
    state=FEMA_file.iloc[i,33]
    if amount_paid_on_building_claim < 0:
        amount_paid_on_building_claim=amount_paid_on_building_claim*(-1)
    if amount_paid_on_contents_claim < 0:
        amount_paid_on_contents_claim=amount_paid_on_contents_claim*(-1)
    #Find out the dataframe's index for the specific year of loss
    find_out_year_index=0
    while find_out_year_index<Num_dataframe:
        if Claims_dataframe.iloc[find_out_year_index,0]==year_of_loss:
            year_index=find_out_year_index
        find_out_year_index+=1
    if polygon1.contains(p):
        Claims_dataframe_Region_1.iloc[year_index,1]=Claims_dataframe_Region_1.iloc[year_index,1]+1
        if amount_paid_on_building_claim>0:
            Paid_claims_dataframe_Region_1.iloc[year_index,1]=Paid_claims_dataframe_Region_1.iloc[yea
r_index,1]+1
        Amount_paid_on_building_claims_dataframe_Region_1.iloc[year_index,1]=Amount_paid_on_building_
claims_dataframe_Region_1.iloc[year_index,1]+amount_paid_on_building_claim
        Amount_paid_on_contents_claims_dataframe_Region_1.iloc[year_index,1]=Amount_paid_on_contents_
claims_dataframe_Region_1.iloc[year_index,1]+amount_paid_on_contents_claim
    elif polygon2.contains(p):
        Claims_dataframe_Region_2.iloc[year_index,1]=Claims_dataframe_Region_2.iloc[year_index,1]+1
        if amount_paid_on_building_claim>0:
            Paid_claims_dataframe_Region_2.iloc[year_index,1]=Paid_claims_dataframe_Region_2.iloc[yea
r_index,1]+1
        Amount_paid_on_building_claims_dataframe_Region_2.iloc[year_index,1]=Amount_paid_on_building_
claims_dataframe_Region_2.iloc[year_index,1]+amount_paid_on_building_claim
        Amount_paid_on_contents_claims_dataframe_Region_2.iloc[year_index,1]=Amount_paid_on_contents_
claims_dataframe_Region_2.iloc[year_index,1]+amount_paid_on_contents_claim
    elif polygon3.contains(p):
```

```
246.         Claims_dataframe_Region_3.iloc[year_index,1]=Claims_dataframe_Region_3.iloc[year_index,1]+1
247.         if amount_paid_on_building_claim>0:
248.             Paid_claims_dataframe_Region_3.iloc[year_index,1]=Paid_claims_dataframe_Region_3.iloc[yea
    r_index,1]+1
249.         Amount_paid_on_building_claims_dataframe_Region_3.iloc[year_index,1]=Amount_paid_on_building_
    claims_dataframe_Region_3.iloc[year_index,1]+amount_paid_on_building_claim
250.         Amount_paid_on_contents_claims_dataframe_Region_3.iloc[year_index,1]=Amount_paid_on_contents_
    claims_dataframe_Region_3.iloc[year_index,1]+amount_paid_on_contents_claim
251.     elif polygon4.contains(p):
252.         Claims_dataframe_Region_4.iloc[year_index,1]=Claims_dataframe_Region_4.iloc[year_index,1]+1
253.         if amount_paid_on_building_claim>0:
254.             Paid_claims_dataframe_Region_4.iloc[year_index,1]=Paid_claims_dataframe_Region_4.iloc[yea
    r_index,1]+1
255.         Amount_paid_on_building_claims_dataframe_Region_4.iloc[year_index,1]=Amount_paid_on_building_
    claims_dataframe_Region_4.iloc[year_index,1]+amount_paid_on_building_claim
256.         Amount_paid_on_contents_claims_dataframe_Region_4.iloc[year_index,1]=Amount_paid_on_contents_
    claims_dataframe_Region_4.iloc[year_index,1]+amount_paid_on_contents_claim
257.     elif polygon5.contains(p):
258.         Claims_dataframe_Region_5.iloc[year_index,1]=Claims_dataframe_Region_5.iloc[year_index,1]+1
259.         if amount_paid_on_building_claim>0:
260.             Paid_claims_dataframe_Region_5.iloc[year_index,1]=Paid_claims_dataframe_Region_5.iloc[yea
    r_index,1]+1
261.         Amount_paid_on_building_claims_dataframe_Region_5.iloc[year_index,1]=Amount_paid_on_building_
    claims_dataframe_Region_5.iloc[year_index,1]+amount_paid_on_building_claim
262.         Amount_paid_on_contents_claims_dataframe_Region_5.iloc[year_index,1]=Amount_paid_on_contents_
    claims_dataframe_Region_5.iloc[year_index,1]+amount_paid_on_contents_claim
263.     elif polygon6.contains(p):
264.         Claims_dataframe_Region_6.iloc[year_index,1]=Claims_dataframe_Region_6.iloc[year_index,1]+1
265.         if amount_paid_on_building_claim>0:
266.             Paid_claims_dataframe_Region_6.iloc[year_index,1]=Paid_claims_dataframe_Region_6.iloc[yea
    r_index,1]+1
267.         Amount_paid_on_building_claims_dataframe_Region_6.iloc[year_index,1]=Amount_paid_on_building_
    claims_dataframe_Region_6.iloc[year_index,1]+amount_paid_on_building_claim
268.         Amount_paid_on_contents_claims_dataframe_Region_6.iloc[year_index,1]=Amount_paid_on_contents_
    claims_dataframe_Region_6.iloc[year_index,1]+amount_paid_on_contents_claim
269.     elif polygon7.contains(p):
270.         Claims_dataframe_Region_7.iloc[year_index,1]=Claims_dataframe_Region_7.iloc[year_index,1]+1
271.         if amount_paid_on_building_claim>0:
272.             Paid_claims_dataframe_Region_7.iloc[year_index,1]=Paid_claims_dataframe_Region_7.iloc[yea
    r_index,1]+1
273.         Amount_paid_on_building_claims_dataframe_Region_7.iloc[year_index,1]=Amount_paid_on_building_
    claims_dataframe_Region_7.iloc[year_index,1]+amount_paid_on_building_claim
274.         Amount_paid_on_contents_claims_dataframe_Region_7.iloc[year_index,1]=Amount_paid_on_contents_
    claims_dataframe_Region_7.iloc[year_index,1]+amount_paid_on_contents_claim
275.     elif polygon8.contains(p):
276.         Claims_dataframe_Region_8.iloc[year_index,1]=Claims_dataframe_Region_8.iloc[year_index,1]+1
277.         if amount_paid_on_building_claim>0:
278.             Paid_claims_dataframe_Region_8.iloc[year_index,1]=Paid_claims_dataframe_Region_8.iloc[yea
    r_index,1]+1
279.         Amount_paid_on_building_claims_dataframe_Region_8.iloc[year_index,1]=Amount_paid_on_building_
    claims_dataframe_Region_8.iloc[year_index,1]+amount_paid_on_building_claim
280.         Amount_paid_on_contents_claims_dataframe_Region_8.iloc[year_index,1]=Amount_paid_on_contents_
    claims_dataframe_Region_8.iloc[year_index,1]+amount_paid_on_contents_claim
281.     elif polygon9.contains(p):
282.         Claims_dataframe_Region_9.iloc[year_index,1]=Claims_dataframe_Region_9.iloc[year_index,1]+1
283.         if amount_paid_on_building_claim>0:
284.             Paid_claims_dataframe_Region_9.iloc[year_index,1]=Paid_claims_dataframe_Region_9.iloc[yea
    r_index,1]+1
285.         Amount_paid_on_building_claims_dataframe_Region_9.iloc[year_index,1]=Amount_paid_on_building_
    claims_dataframe_Region_9.iloc[year_index,1]+amount_paid_on_building_claim
286.         Amount_paid_on_contents_claims_dataframe_Region_9.iloc[year_index,1]=Amount_paid_on_contents_
    claims_dataframe_Region_9.iloc[year_index,1]+amount_paid_on_contents_claim
287.     elif polygon10.contains(p):
288.         Claims_dataframe_Region_10.iloc[year_index,1]=Claims_dataframe_Region_10.iloc[year_index,1]+1

289.         if amount_paid_on_building_claim>0:
290.             Paid_claims_dataframe_Region_10.iloc[year_index,1]=Paid_claims_dataframe_Region_10.iloc[y
    ear_index,1]+1
291.         Amount_paid_on_building_claims_dataframe_Region_10.iloc[year_index,1]=Amount_paid_on_building
    _claims_dataframe_Region_10.iloc[year_index,1]+amount_paid_on_building_claim
292.         Amount_paid_on_contents_claims_dataframe_Region_10.iloc[year_index,1]=Amount_paid_on_contents
    _claims_dataframe_Region_10.iloc[year_index,1]+amount_paid_on_contents_claim
293.     elif polygon11.contains(p):
```

```
294.         Claims_dataframe_Region_11.iloc[year_index,1]=Claims_dataframe_Region_11.iloc[year_index,1]+1

295.         if amount_paid_on_building_claim>0:
296.             Paid_claims_dataframe_Region_11.iloc[year_index,1]=Paid_claims_dataframe_Region_11.iloc[y
    ear_index,1]+1
297.         Amount_paid_on_building_claims_dataframe_Region_11.iloc[year_index,1]=Amount_paid_on_building
    _claims_dataframe_Region_11.iloc[year_index,1]+amount_paid_on_building_claim
298.         Amount_paid_on_contents_claims_dataframe_Region_11.iloc[year_index,1]=Amount_paid_on_contents
    _claims_dataframe_Region_11.iloc[year_index,1]+amount_paid_on_contents_claim
299.     elif polygon12.contains(p):
300.         Claims_dataframe_Region_12.iloc[year_index,1]=Claims_dataframe_Region_12.iloc[year_index,1]+1

301.         if amount_paid_on_building_claim>0:
302.             Paid_claims_dataframe_Region_12.iloc[year_index,1]=Paid_claims_dataframe_Region_12.iloc[y
    ear_index,1]+1
303.         Amount_paid_on_building_claims_dataframe_Region_12.iloc[year_index,1]=Amount_paid_on_building
    _claims_dataframe_Region_12.iloc[year_index,1]+amount_paid_on_building_claim
304.         Amount_paid_on_contents_claims_dataframe_Region_12.iloc[year_index,1]=Amount_paid_on_contents
    _claims_dataframe_Region_12.iloc[year_index,1]+amount_paid_on_contents_claim
305.     elif polygon13.contains(p):
306.         Claims_dataframe_Region_13.iloc[year_index,1]=Claims_dataframe_Region_13.iloc[year_index,1]+1

307.         if amount_paid_on_building_claim>0:
308.             Paid_claims_dataframe_Region_13.iloc[year_index,1]=Paid_claims_dataframe_Region_13.iloc[y
    ear_index,1]+1
309.         Amount_paid_on_building_claims_dataframe_Region_13.iloc[year_index,1]=Amount_paid_on_building
    _claims_dataframe_Region_13.iloc[year_index,1]+amount_paid_on_building_claim
310.         Amount_paid_on_contents_claims_dataframe_Region_13.iloc[year_index,1]=Amount_paid_on_contents
    _claims_dataframe_Region_13.iloc[year_index,1]+amount_paid_on_contents_claim
311.     elif polygon14.contains(p):
312.         Claims_dataframe_Region_14.iloc[year_index,1]=Claims_dataframe_Region_14.iloc[year_index,1]+1

313.         if amount_paid_on_building_claim>0:
314.             Paid_claims_dataframe_Region_14.iloc[year_index,1]=Paid_claims_dataframe_Region_14.iloc[y
    ear_index,1]+1
315.         Amount_paid_on_building_claims_dataframe_Region_14.iloc[year_index,1]=Amount_paid_on_building
    _claims_dataframe_Region_14.iloc[year_index,1]+amount_paid_on_building_claim
316.         Amount_paid_on_contents_claims_dataframe_Region_14.iloc[year_index,1]=Amount_paid_on_contents
    _claims_dataframe_Region_14.iloc[year_index,1]+amount_paid_on_contents_claim
317.     elif polygon15.contains(p):
318.         Claims_dataframe_Region_15.iloc[year_index,1]=Claims_dataframe_Region_15.iloc[year_index,1]+1

319.         if amount_paid_on_building_claim>0:
320.             Paid_claims_dataframe_Region_15.iloc[year_index,1]=Paid_claims_dataframe_Region_15.iloc[y
    ear_index,1]+1
321.         Amount_paid_on_building_claims_dataframe_Region_15.iloc[year_index,1]=Amount_paid_on_building
    _claims_dataframe_Region_15.iloc[year_index,1]+amount_paid_on_building_claim
322.         Amount_paid_on_contents_claims_dataframe_Region_15.iloc[year_index,1]=Amount_paid_on_contents
    _claims_dataframe_Region_15.iloc[year_index,1]+amount_paid_on_contents_claim
323.     elif polygon16.contains(p):
324.         Claims_dataframe_Region_16.iloc[year_index,1]=Claims_dataframe_Region_16.iloc[year_index,1]+1

325.         if amount_paid_on_building_claim>0:
326.             Paid_claims_dataframe_Region_16.iloc[year_index,1]=Paid_claims_dataframe_Region_16.iloc[y
    ear_index,1]+1
327.         Amount_paid_on_building_claims_dataframe_Region_16.iloc[year_index,1]=Amount_paid_on_building
    _claims_dataframe_Region_16.iloc[year_index,1]+amount_paid_on_building_claim
328.         Amount_paid_on_contents_claims_dataframe_Region_16.iloc[year_index,1]=Amount_paid_on_contents
    _claims_dataframe_Region_16.iloc[year_index,1]+amount_paid_on_contents_claim
329.     elif polygon17.contains(p):
330.         Claims_dataframe_Region_17.iloc[year_index,1]=Claims_dataframe_Region_17.iloc[year_index,1]+1

331.         if amount_paid_on_building_claim>0:
332.             Paid_claims_dataframe_Region_17.iloc[year_index,1]=Paid_claims_dataframe_Region_17.iloc[y
    ear_index,1]+1
333.         Amount_paid_on_building_claims_dataframe_Region_17.iloc[year_index,1]=Amount_paid_on_building
    _claims_dataframe_Region_17.iloc[year_index,1]+amount_paid_on_building_claim
334.         Amount_paid_on_contents_claims_dataframe_Region_17.iloc[year_index,1]=Amount_paid_on_contents
    _claims_dataframe_Region_17.iloc[year_index,1]+amount_paid_on_contents_claim
335.     elif polygon18.contains(p):
336.         Claims_dataframe_Region_18.iloc[year_index,1]=Claims_dataframe_Region_18.iloc[year_index,1]+1

337.         if amount_paid_on_building_claim>0:
```

```
            Paid_claims_dataframe_Region_18.iloc[year_index,1]=Paid_claims_dataframe_Region_18.iloc[year_index,1]+1
        Amount_paid_on_building_claims_dataframe_Region_18.iloc[year_index,1]=Amount_paid_on_building_claims_dataframe_Region_18.iloc[year_index,1]+amount_paid_on_building_claim
        Amount_paid_on_contents_claims_dataframe_Region_18.iloc[year_index,1]=Amount_paid_on_contents_claims_dataframe_Region_18.iloc[year_index,1]+amount_paid_on_contents_claim
    elif polygon19.contains(p):
        Claims_dataframe_Region_19.iloc[year_index,1]=Claims_dataframe_Region_19.iloc[year_index,1]+1

        if amount_paid_on_building_claim>0:
            Paid_claims_dataframe_Region_19.iloc[year_index,1]=Paid_claims_dataframe_Region_19.iloc[year_index,1]+1
        Amount_paid_on_building_claims_dataframe_Region_19.iloc[year_index,1]=Amount_paid_on_building_claims_dataframe_Region_19.iloc[year_index,1]+amount_paid_on_building_claim
        Amount_paid_on_contents_claims_dataframe_Region_19.iloc[year_index,1]=Amount_paid_on_contents_claims_dataframe_Region_19.iloc[year_index,1]+amount_paid_on_contents_claim
    elif polygon20.contains(p):
        Claims_dataframe_Region_20.iloc[year_index,1]=Claims_dataframe_Region_20.iloc[year_index,1]+1

        if amount_paid_on_building_claim>0:
            Paid_claims_dataframe_Region_20.iloc[year_index,1]=Paid_claims_dataframe_Region_20.iloc[year_index,1]+1
        Amount_paid_on_building_claims_dataframe_Region_20.iloc[year_index,1]=Amount_paid_on_building_claims_dataframe_Region_20.iloc[year_index,1]+amount_paid_on_building_claim
        Amount_paid_on_contents_claims_dataframe_Region_20.iloc[year_index,1]=Amount_paid_on_contents_claims_dataframe_Region_20.iloc[year_index,1]+amount_paid_on_contents_claim
    elif polygon21.contains(p):
        Claims_dataframe_Region_21.iloc[year_index,1]=Claims_dataframe_Region_21.iloc[year_index,1]+1

        if amount_paid_on_building_claim>0:
            Paid_claims_dataframe_Region_21.iloc[year_index,1]=Paid_claims_dataframe_Region_21.iloc[year_index,1]+1
        Amount_paid_on_building_claims_dataframe_Region_21.iloc[year_index,1]=Amount_paid_on_building_claims_dataframe_Region_21.iloc[year_index,1]+amount_paid_on_building_claim
        Amount_paid_on_contents_claims_dataframe_Region_21.iloc[year_index,1]=Amount_paid_on_contents_claims_dataframe_Region_21.iloc[year_index,1]+amount_paid_on_contents_claim
    else:
        the_ones_that_were_in_no_watershed+=1
print('the_ones_that_were_in_no_watershed',the_ones_that_were_in_no_watershed)

Claims_dataframe_Region_1.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Claims Region_1')
Paid_claims_dataframe_Region_1.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Paid claims Region_1')
Amount_paid_on_building_claims_dataframe_Region_1.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
 Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_1'
)
Amount_paid_on_contents_claims_dataframe_Region_1.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
 Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_1'
)
Claims_dataframe_Region_2.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Claims Region_2')
Paid_claims_dataframe_Region_2.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Paid claims Region_2')
Amount_paid_on_building_claims_dataframe_Region_2.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
 Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_2'
)
Amount_paid_on_contents_claims_dataframe_Region_2.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
 Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_2'
)
Claims_dataframe_Region_3.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Claims Region_3')
Paid_claims_dataframe_Region_3.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Paid claims Region_3')
Amount_paid_on_building_claims_dataframe_Region_3.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
 Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_3'
)
Amount_paid_on_contents_claims_dataframe_Region_3.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
```

```
    Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_3'
    )
Claims_dataframe_Region_4.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA rec
    ords - Results for every Basin and every State\\Regions\\Claims Region_4')
Paid_claims_dataframe_Region_4.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEM
    A records - Results for every Basin and every State\\Regions\\Paid claims Region_4')
Amount_paid_on_building_claims_dataframe_Region_4.to_pickle('C:\\Users\\Konstantinos\\Documents\\THES
    IS\\Python\\11. FEMA records -
    Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_4'
    )
Amount_paid_on_contents_claims_dataframe_Region_4.to_pickle('C:\\Users\\Konstantinos\\Documents\\THES
    IS\\Python\\11. FEMA records -
    Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_4'
    )
Claims_dataframe_Region_5.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA rec
    ords - Results for every Basin and every State\\Regions\\Claims Region_5')
Paid_claims_dataframe_Region_5.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEM
    A records - Results for every Basin and every State\\Regions\\Paid claims Region_5')
Amount_paid_on_building_claims_dataframe_Region_5.to_pickle('C:\\Users\\Konstantinos\\Documents\\THES
    IS\\Python\\11. FEMA records -
    Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_5'
    )
Amount_paid_on_contents_claims_dataframe_Region_5.to_pickle('C:\\Users\\Konstantinos\\Documents\\THES
    IS\\Python\\11. FEMA records -
    Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_5'
    )
Claims_dataframe_Region_6.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA rec
    ords - Results for every Basin and every State\\Regions\\Claims Region_6')
Paid_claims_dataframe_Region_6.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEM
    A records - Results for every Basin and every State\\Regions\\Paid claims Region_6')
Amount_paid_on_building_claims_dataframe_Region_6.to_pickle('C:\\Users\\Konstantinos\\Documents\\THES
    IS\\Python\\11. FEMA records -
    Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_6'
    )
Amount_paid_on_contents_claims_dataframe_Region_6.to_pickle('C:\\Users\\Konstantinos\\Documents\\THES
    IS\\Python\\11. FEMA records -
    Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_6'
    )
Claims_dataframe_Region_7.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA rec
    ords - Results for every Basin and every State\\Regions\\Claims Region_7')
Paid_claims_dataframe_Region_7.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEM
    A records - Results for every Basin and every State\\Regions\\Paid claims Region_7')
Amount_paid_on_building_claims_dataframe_Region_7.to_pickle('C:\\Users\\Konstantinos\\Documents\\THES
    IS\\Python\\11. FEMA records -
    Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_7'
    )
Amount_paid_on_contents_claims_dataframe_Region_7.to_pickle('C:\\Users\\Konstantinos\\Documents\\THES
    IS\\Python\\11. FEMA records -
    Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_7'
    )
Claims_dataframe_Region_8.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA rec
    ords - Results for every Basin and every State\\Regions\\Claims Region_8')
Paid_claims_dataframe_Region_8.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEM
    A records - Results for every Basin and every State\\Regions\\Paid claims Region_8')
Amount_paid_on_building_claims_dataframe_Region_8.to_pickle('C:\\Users\\Konstantinos\\Documents\\THES
    IS\\Python\\11. FEMA records -
    Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_8'
    )
Amount_paid_on_contents_claims_dataframe_Region_8.to_pickle('C:\\Users\\Konstantinos\\Documents\\THES
    IS\\Python\\11. FEMA records -
    Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_8'
    )
Claims_dataframe_Region_9.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA rec
    ords - Results for every Basin and every State\\Regions\\Claims Region_9')
Paid_claims_dataframe_Region_9.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEM
    A records - Results for every Basin and every State\\Regions\\Paid claims Region_9')
Amount_paid_on_building_claims_dataframe_Region_9.to_pickle('C:\\Users\\Konstantinos\\Documents\\THES
    IS\\Python\\11. FEMA records -
    Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_9'
    )
Amount_paid_on_contents_claims_dataframe_Region_9.to_pickle('C:\\Users\\Konstantinos\\Documents\\THES
    IS\\Python\\11. FEMA records -
    Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_9'
    )
```

```
Claims_dataframe_Region_10.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Claims Region_10')
Paid_claims_dataframe_Region_10.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Paid claims Region_10')
Amount_paid_on_building_claims_dataframe_Region_10.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
 Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_10
')
Amount_paid_on_contents_claims_dataframe_Region_10.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
 Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_10
')
Claims_dataframe_Region_11.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Claims Region_11')
Paid_claims_dataframe_Region_11.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Paid claims Region_11')
Amount_paid_on_building_claims_dataframe_Region_11.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
 Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_11
')
Amount_paid_on_contents_claims_dataframe_Region_11.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
 Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_11
')
Claims_dataframe_Region_12.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Claims Region_12')
Paid_claims_dataframe_Region_12.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Paid claims Region_12')
Amount_paid_on_building_claims_dataframe_Region_12.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
 Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_12
')
Amount_paid_on_contents_claims_dataframe_Region_12.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
 Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_12
')
Claims_dataframe_Region_13.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Claims Region_13')
Paid_claims_dataframe_Region_13.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Paid claims Region_13')
Amount_paid_on_building_claims_dataframe_Region_13.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
 Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_13
')
Amount_paid_on_contents_claims_dataframe_Region_13.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
 Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_13
')
Claims_dataframe_Region_14.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Claims Region_14')
Paid_claims_dataframe_Region_14.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Paid claims Region_14')
Amount_paid_on_building_claims_dataframe_Region_14.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
 Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_14
')
Amount_paid_on_contents_claims_dataframe_Region_14.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
 Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_14
')
Claims_dataframe_Region_15.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Claims Region_15')
Paid_claims_dataframe_Region_15.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Paid claims Region_15')
Amount_paid_on_building_claims_dataframe_Region_15.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
 Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_15
')
Amount_paid_on_contents_claims_dataframe_Region_15.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
 Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_15
')
Claims_dataframe_Region_16.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Claims Region_16')
```

```
Paid_claims_dataframe_Region_16.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Paid claims Region_16')
Amount_paid_on_building_claims_dataframe_Region_16.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_16')
Amount_paid_on_contents_claims_dataframe_Region_16.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_16')
Claims_dataframe_Region_17.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Claims Region_17')
Paid_claims_dataframe_Region_17.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Paid claims Region_17')
Amount_paid_on_building_claims_dataframe_Region_17.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_17')
Amount_paid_on_contents_claims_dataframe_Region_17.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_17')
Claims_dataframe_Region_18.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Claims Region_18')
Paid_claims_dataframe_Region_18.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Paid claims Region_18')
Amount_paid_on_building_claims_dataframe_Region_18.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_18')
Amount_paid_on_contents_claims_dataframe_Region_18.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_18')
Claims_dataframe_Region_19.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Claims Region_19')
Paid_claims_dataframe_Region_19.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Paid claims Region_19')
Amount_paid_on_building_claims_dataframe_Region_19.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_19')
Amount_paid_on_contents_claims_dataframe_Region_19.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_19')
Claims_dataframe_Region_20.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Claims Region_20')
Paid_claims_dataframe_Region_20.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Paid claims Region_20')
Amount_paid_on_building_claims_dataframe_Region_20.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_20')
Amount_paid_on_contents_claims_dataframe_Region_20.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_20')
Claims_dataframe_Region_21.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Claims Region_21')
Paid_claims_dataframe_Region_21.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Paid claims Region_21')
Amount_paid_on_building_claims_dataframe_Region_21.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Amount paid on building claims dataframe Region_21')
Amount_paid_on_contents_claims_dataframe_Region_21.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_21')

#Sum buildings and contents compensations to one dataframe
sum_compensations_regions=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
```

```
    Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_1'
    )
Num=len(sum_compensations_regions)
i=0
while i<Num:
    sum_compensations_regions.iloc[i,1]=0.0
    i+=1
sum_compensations_states=sum_compensations_regions.copy()
sum_compensations_regions.columns=['Year','Sum compensations (buildings and contents)']
sum_compensations_states.columns=['Year','Sum compensations (buildings and contents)']
#print(sum_compensations_regions)
print(sum_compensations_states)

i=1
while i<22:
    sum_compensations_regions1=sum_compensations_regions.copy()
    i_str=str(i)
    file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
     Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_'+
    i_str)
    file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records -
     Results for every Basin and every State\\Regions\\Amount paid_on contents claims dataframe Region_'+
    i_str)
    j=0
    Num=len(file1)
    while j<Num:
        sum_compensations_regions1.iloc[j,1]=file1.iloc[j,1]+file2.iloc[j,1]
        j+=1
    sum_compensations_regions1.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEM
    A records -
     Results for every Basin and every State\\Regions\\Sum of buildings and contents amount paid datafram
    e Region_'+str(i))
#    print(sum_compensations_regions1)
    i+=1
```

11.13. Plot of the USA map with Spearman correlation for all the selected gauge locations and all thresholds between annual collective risk of a specific gauge locations and the aggregated FEMA claims of the Hydrological Unit (region) that this gauge location belongs to.

```
import geopandas as gpd
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from scipy.stats import spearmanr
from mpl_toolkits.axes_grid1 import make_axes_locatable

threshold_df_data = {'Text': ['90%','95%','98%','99%'],'Numerical': [0.9, 0.95, 0.98, 0.99]}
threshold_df = pd.DataFrame(data=threshold_df_data)
Num_threshold_df=len(threshold_df)
#print(threshold_df)

threshold_loop=0
while threshold_loop<Num_threshold_df:
    threshold=threshold_df.iloc[threshold_loop,0]

    #Read collective risk S dataframe
    Collective_Risk_S=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\1. Results
    for the Observed\\'+threshold+'\\Collective Risk S CAMELS 1980-2014')
    Collective_Risk_S=Collective_Risk_S.drop(Collective_Risk_S.index[35])
    #print(Collective_Risk_S)

    #Claims: Shapefile for USA states and stations (with correlation)
    shapefile='C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\10. Shapefiles\\States\\states.shp
    '
    stations1=gpd.read_file(shapefile)
    ax=stations1.plot(color='white', edgecolor='black',linewidth=0.3, figsize=(16,16))
    ax.set_xlim(xmin=-130, xmax=-65)
```

```
    ax.set_ylim(ymin=20, ymax=55)
    region_index=1
    while region_index<19:
    #    sns.set()
        region_index_str=str(region_index)
        Claims=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Claims Region_'+region_index_str)
        i=0
        while i<10:
            Claims=Claims.drop(Claims.index[0])
            Claims=Claims.reset_index(drop=True)
            i+=1
        i=0
        while i<5:
            Claims=Claims.drop(Claims.index[35])
            Claims=Claims.reset_index(drop=True)
            i+=1
        stations=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Region '+region_index_str+' - All stations with criteria (1980-2014 and 10% missing values')
    #    print(stations)
        Num_cols=len(stations)
        corr_coef = pd.DataFrame(np.zeros((Num_cols, 3)))
        i=0
        while i<Num_cols:
            name=stations.iloc[i,1]
            S_column=Collective_Risk_S[name]
            PaidClaims=Claims['Claims']
            corr,p_value=spearmanr(S_column, PaidClaims)
            corr_coef.iloc[i,0]=name
            corr_coef.iloc[i,1]=corr
            corr_coef.iloc[i,2]=p_value
            i+=1
    #    print(corr_coef)
        Stations=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Gauge_info - 671 stations')
        Num_stations=len(Stations)
        #print(Stations)
        lat_lon=corr_coef.copy()
        lat_lon.columns=['Name','Lat','Lon']
        Num_lat_lon=len(lat_lon)
        #print(lat_lon)
        i=0
        j=0
        while i<Num_lat_lon:
            j=0
            while j<Num_stations:
                if lat_lon.iloc[i,0]==Stations.iloc[j,1]:
                    lat_lon.iloc[i,1]=Stations.iloc[j,3]
                    lat_lon.iloc[i,2]=Stations.iloc[j,4]
                j+=1
            i+=1
        lat_lon['Corr. coef.']=corr_coef[1]
        #print(lat_lon)
        x=lat_lon.iloc[:,2]
        y=lat_lon.iloc[:,1]
        color=lat_lon.iloc[:,3]
        plt.scatter(x,y, c=lat_lon.iloc[:,3], marker='o', edgecolor='black', linewidth=0.2, alpha=1, cmap=plt.cm.get_cmap('rainbow', 5))
        plt.clim(-0.25, 1);
        region_index+=1
    plt.title('Collective risk (S) vs Cumulative Claims - Threshold '+threshold,fontsize=19, y=1.05)
    plt.xlabel('Longitude',fontsize=18)
    plt.ylabel('Latitude',fontsize=18)
    plt.tick_params(axis='both',which='both',labelsize=17,zorder=20)

    divider = make_axes_locatable(ax)
    cax = divider.append_axes("right", size="5%", pad=0.05)

    v1 = np.linspace(-0.25, 1, 6)
    cbar = plt.colorbar(ticks=v1, cax=cax)
    cbar.ax.tick_params(labelsize=17)
    cbar.set_label('Correlation coefficient', fontsize=17, labelpad=20)
```

```
    plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\12. Correlation with FEMA records\\Spearman\\Observed\\'+threshold+'\\MAP WITH STATES\\USA States map (Spearman) - Claims '+threshold+'.png')
    threshold_loop+=1
```

11.14. Boxplots for all thresholds of the Spearman correlation coefficient between collective risk of every gauge location and FEMA's claims records of the hydrological unit that a specific gauge location belongs.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from scipy.stats import spearmanr

def boxplot(x, filename):
    file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\12. Correlation with FEMA records\\Spearman\\Spearman - HU claims and S (90%)')
    file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\12. Correlation with FEMA records\\Spearman\\Spearman - HU claims and S (95%)')
    file3=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\12. Correlation with FEMA records\\Spearman\\Spearman - HU claims and S (98%)')
    file4=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\12. Correlation with FEMA records\\Spearman\\Spearman - HU claims and S (99%)')

    # Create data
    collectn_1 = file1.iloc[x,1:100]
    collectn_11 = file1.iloc[x,101]
    collectn_2 = file2.iloc[x,1:100]
    collectn_22 = file2.iloc[x,101]
    collectn_3 = file3.iloc[x,1:100]
    collectn_33 = file3.iloc[x,101]
    collectn_4 = file4.iloc[x,1:100]
    collectn_44 = file4.iloc[x,101]

    # Combine these different collections into a list
    data_to_plot = [collectn_1, collectn_11, collectn_2, collectn_22, collectn_3, collectn_33, collectn_4, collectn_44]

    # Create a figure instance
    fig=plt.figure(figsize=(11,6),dpi=200)
    ax = fig.add_subplot(111)
    medianprops = dict(linestyle='-', linewidth=2, color='red')
    ax.set_axisbelow(True)
    ax.set_title('Boxplot of the Spearman correlation coefficient between \nHydrological Unit Claims and Collective Risk S (Gauge ID: '+filename+')',fontsize=14)
    ax.set_ylabel('Spearman correlation coefficient',fontsize=13)
#    plt.tick_params(axis='x',which='x')
    ax.tick_params(axis='x', labelsize=10,zorder=20,rotation=0)
    bp = ax.boxplot(data_to_plot,medianprops=medianprops,patch_artist=True,labels=['90% Shuffled','90% Observed','95% Shuffled','95% Observed','98% Shuffled','98% Observed','99% Shuffled','99% Observed'])
    ax.grid(color='black', linestyle='-', linewidth=0.1)
    colors = ['skyblue','blue', 'moccasin','green', 'lightsalmon','tan', 'tan','pink']
    for patch, color in zip(bp['boxes'], colors):
        patch.set_facecolor(color)
    plt.yticks(np.arange(-0.6, 1.01, 0.1))
    plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\12. Correlation with FEMA records\\Spearman\\Boxplots HU\\Spearman S-HUCLAIMS Boxplot '+filename+'.png')
    plt.close()


threshold_df_data = {'Text': ['90%','95%','98%','99%'],'Numerical': [0.9, 0.95, 0.98, 0.99]}
threshold_df = pd.DataFrame(data=threshold_df_data)
Num_threshold_df=len(threshold_df)
#print(threshold_df)

df_filenames=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Timeseries 1980-2014 and 10% limit for missing values')
Num_cols_filenames = len (df_filenames)
```

```
i=0
while i<Num_cols_filenames:
    filename=df_filenames.iloc[i,0]
    filename1=filename.replace(".mat","")
    df_filenames.iloc[i,0]=filename1
    i+=1
df_filenames1=df_filenames.copy()
df1=df_filenames.copy()
i=1
while i<101:
    i_str=str(i)
    df_filenames1.columns = [i_str]
    df1[i_str]=df_filenames1[i_str]
    i+=1
df1['Observed']=df_filenames1[i_str]
#print(df1)
#print(df_filenames)

threshold_loop=0
while threshold_loop<Num_threshold_df:
    threshold=threshold_df.iloc[threshold_loop,1]
    threshold_text=threshold_df.iloc[threshold_loop,0]

    spearman_corr_coef=df1.copy()

    i=0
    while i<Num_cols_filenames:
        filename=df_filenames.iloc[i,0]

        stations_df=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\Gauge_info - 671 stations')
        Num_stations=len(stations_df)
        stations_index=0
        while stations_index<Num_stations:
            if filename==stations_df.iloc[stations_index,1]:
                HU=stations_df.iloc[stations_index,0]
                HU_str=str(HU)
                break
            stations_index+=1
        HU_claims_df=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\11. FEMA records - Results for every Basin and every State\\Regions\\Claims Region_'+HU_str)
        HU_claims_df=HU_claims_df.iloc[10:45,:]
        HU_claims_df=HU_claims_df.reset_index(drop=True)
        HU_claims_df=HU_claims_df.drop(['Year'], axis=1)
#         print(HU_claims_df)

        observed_S_df=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\1. Results for the Observed\\'+threshold_text+'\\Collective Risk S CAMELS 1980-2014')
        observed_S_df=observed_S_df[filename]
        observed_S_df=observed_S_df.drop(observed_S_df.index[35])
#         print(observed_S_df)

        z=0
        k=len(observed_S_df)
        HU_claims_df_observed=HU_claims_df.copy()
        while z<k:
            if observed_S_df.iloc[z]==0 or HU_claims_df_observed.iloc[z,0]==0:
                observed_S_df=observed_S_df.drop(observed_S_df.index[z])
                HU_claims_df_observed=HU_claims_df_observed.drop(HU_claims_df_observed.index[z])
                observed_S_df=observed_S_df.reset_index(drop=True)
                HU_claims_df_observed=HU_claims_df_observed.reset_index(drop=True)
                k=k-1
            else:
                z+=1

        corr1,p_value1=spearmanr(observed_S_df, HU_claims_df_observed)
        spearman_corr_coef.iloc[i,101]=corr1

        shuffle_index=1
        while shuffle_index<101:
            HU_claims_df_shuffled=HU_claims_df.copy()
            shuffled_S_df=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\4. Results for the Shuffled of the timeseries with criteria\\'+threshold_text+'\\'+filename+' Collective Risk for 100 Shuffled')
```

```
            shuffled_S=shuffled_S_df.iloc[:,shuffle_index]
            z=0
            k=len(shuffled_S)
            while z<k:
                if shuffled_S.iloc[z]==0 or HU_claims_df_shuffled.iloc[z,0]==0:
                    shuffled_S=shuffled_S.drop(shuffled_S.index[z])
                    HU_claims_df_shuffled=HU_claims_df_shuffled.drop(HU_claims_df_shuffled.index[z])

                    shuffled_S=shuffled_S.reset_index(drop=True)
                    HU_claims_df_shuffled=HU_claims_df_shuffled.reset_index(drop=True)
                    k=k-1
                else:
                    z+=1
#
            corr1,p_value1=spearmanr(shuffled_S, HU_claims_df_shuffled)
            spearman_corr_coef.iloc[i,shuffle_index]=corr1

            shuffle_index+=1

        i+=1
#        print(i)
#    print(spearman_corr_coef)
    spearman_corr_coef.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\12. Correlation with FEMA records\\Spearman\\Spearman - HU claims and S ('+threshold_text+')')

    threshold_loop+=1

#Plot Boxplot
i=0
while i<Num_cols_filenames:
    filename=df_filenames.iloc[i,0]
    boxplot(i, filename)
    i+=1
```

11.15. Collective risk method on Larissa's precipitation mechanisms. Shuffle time series. Calculation of annual collective risk, annual peak, *Average Yi*, the duration and the number of the over-threshold events for all thresholds. Calculation and plot of the ECDF diagrams of collective risk, return intervals and duration of over-threshold events. Box plot of the correlation between *Average Yi* and *N*. Box plot of correlation between collective risk and actual compensations. Application on the observed as well as the shuffled time series for all thresholds.

```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from scipy.stats import pearsonr
from scipy.stats import spearmanr

def missing_values_percentage(df_TS):
    df_withoutNaN=df_TS.dropna(axis=0,how='any', thresh=None, subset=None, inplace=False)
    Num_cols1=len(df_withoutNaN)
    Num_cols2=len(df_TS)
    missing_values_percentage=1-Num_cols1/Num_cols2
    return missing_values_percentage

#Estimation of the dataframe events_df (when an event occured)
def Get_events_df(df_TS, threshold):
    #Drop NaN (missing values)
    df_withoutNaN=df_TS.dropna(axis=0,how='any', thresh=None, subset=None, inplace=False)
    df_sorted_noNaN=df_withoutNaN.sort_values(by=['Precipitation'])
#    df_sorted_noNaN=df_withoutNaN.sort_values(3, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last')
    Num_cols = len (df_withoutNaN)
    position=int(threshold*Num_cols)
```

```
    threshold_value=df_sorted_noNaN.iloc[position,3]
    events_df=df_withoutNaN
    events_df.columns = ['Year', 'Month', 'Day', 'Precipitation']
    events_df.drop(events_df.loc[events_df['Precipitation'] < threshold_value].index, inplace=True)
    return events_df

#Estimation of the dataframe without ones (continuous days of that events occur)
def Get_events_df_without_ones(events_df):
    Num_cols = len (events_df)
    events_df_without_ones=events_df.take([0])
    i=1
    while i<=Num_cols-1:
        if events_df.index[i]-1==events_df.index[i-1]:
            i+=1
        else:
            append_row=events_df.iloc[i].copy()
            events_df_without_ones=events_df_without_ones.append(append_row)
            i+=1
    return events_df_without_ones

#Estimation of the returning_interval_df
def Get_return_interval_df(events_df_without_ones):
    Num_cols = len (events_df_without_ones)
    return_interval_df = {'Return interval': [events_df_without_ones.index[0]]}
    return_interval_df = pd.DataFrame(data=return_interval_df)
    i=1
    while i<=(Num_cols-1):
        j=events_df_without_ones.index[i]-events_df_without_ones.index[i-1]
        append_row = {'Return interval': [j]}
        append_row1 = pd.DataFrame(data=append_row)
        return_interval_df=return_interval_df.append(append_row1)
        i+=1
    return_interval_df=return_interval_df.reset_index(drop=True)
    return return_interval_df

#Estimation of the collective_risk_S_df
def collective_risk_function(df_TS, events_df):
    first_year=df_TS.iloc[0,0]
    Num_cols = len (df_TS)-1
    last_year=df_TS.iloc[Num_cols,0]
    collective_risk_S_df = {'Year': [0], 'Collective Risk S': [0.0]}
    collective_risk_S_df = pd.DataFrame(data=collective_risk_S_df)
    a=first_year
    while a<=last_year-1:
        m={'Year': [0], 'Collective Risk S': [0.0]}
        m = pd.DataFrame(data=m)
        append_row=m.iloc[0].copy()
        collective_risk_S_df=collective_risk_S_df.append(append_row)
        a+=1
    collective_risk_S_df=collective_risk_S_df.reset_index(drop=True)
    a=first_year
    i=0
    while a<=last_year:
        collective_risk_S_df.xs(i)['Year'] = a
        a+=1
        i+=1
    a=first_year
    i=0
    j=0
    Num_cols = len (events_df)
    while a<=last_year:
        while j<=(Num_cols-1):
            if events_df.iloc[j,0]==a:
                collective_risk_S_df.xs(i)['Collective Risk S'] = collective_risk_S_df.iloc[i,1] + ev
ents_df.iloc[j,3]
                j+=1
            else:
                j+=1
        j=0
        i+=1
        a+=1
    return collective_risk_S_df

#Estimation of the number_of_events_df
```

```
96. def number_of_events_function(df_TS, events_df):
97.     first_year=df_TS.iloc[0,0]
98.     Num_cols = len (df_TS)-1
99.     last_year=df_TS.iloc[Num_cols,0]
100.    number_of_events_df = {'Year': [0], 'Number of events': [0.0]}
101.    number_of_events_df = pd.DataFrame(data=number_of_events_df)
102.    a=first_year
103.    while a<=last_year-1:
104.        m={'Year': [0], 'Number of events': [0.0]}
105.        m = pd.DataFrame(data=m)
106.        append_row=m.iloc[0].copy()
107.        number_of_events_df=number_of_events_df.append(append_row)
108.        a+=1
109.    number_of_events_df=number_of_events_df.reset_index(drop=True)
110.    a=first_year
111.    i=0
112.    while a<=last_year:
113.        number_of_events_df.xs(i)['Year'] = a
114.        a+=1
115.        i+=1
116.    a=first_year
117.    i=0
118.    j=0
119.    Num_cols = len (events_df)
120.    while a<=last_year:
121.        while j<=(Num_cols-1):
122.            if events_df.iloc[j,0]==a:
123.                number_of_events_df.xs(i)['Number of events'] = number_of_events_df.iloc[i,1] + 1
124.                j+=1
125.            else:
126.                j+=1
127.        j=0
128.        i+=1
129.        a+=1
130.    return number_of_events_df
131.
132.#Estimation of the annual_peak_ONLY_for_events_df
133.def annual_peak_only_for_events_function(df_TS, events_df):
134.    first_year=df_TS.iloc[0,0]
135.    Num_cols = len (df_TS)-1
136.    last_year=df_TS.iloc[Num_cols,0]
137.    annual_peak_only_for_events_df = {'Year': [0], 'Annual peak (Block maxima) only for events': [0.0
    ]}
138.    annual_peak_only_for_events_df = pd.DataFrame(data=annual_peak_only_for_events_df)
139.    a=first_year
140.    while a<=last_year-1:
141.        m={'Year': [0], 'Annual peak (Block maxima) only for events': [0.0]}
142.        m = pd.DataFrame(data=m)
143.        append_row=m.iloc[0].copy()
144.        annual_peak_only_for_events_df=annual_peak_only_for_events_df.append(append_row)
145.        a+=1
146.    annual_peak_only_for_events_df=annual_peak_only_for_events_df.reset_index(drop=True)
147.    a=first_year
148.    i=0
149.    while a<=last_year:
150.        annual_peak_only_for_events_df.xs(i)['Year'] = a
151.        a+=1
152.        i+=1
153.    a=first_year
154.    i=0
155.    j=0
156.    Num_cols = len (events_df)
157.    max=0
158.    while a<=last_year:
159.        while j<=(Num_cols-1):
160.            if events_df.iloc[j,0]==a:
161.                if events_df.iloc[j,3]>max:
162.                    annual_peak_only_for_events_df.xs(i)['Annual peak (Block maxima) only for events'
    ] = events_df.iloc[j,3]
163.                    max=events_df.iloc[j,3]
164.                    j+=1
165.                else:
166.                    j+=1
167.            else:
```

```
                j+=1
        max=0
        j=0
        i+=1
        a+=1
    return annual_peak_only_for_events_df

#Estimation of the annual_peak_for_ALL_df
def annual_peak_for_all_function(df_TS, events_df):
    first_year=df_TS.iloc[0,0]
    Num_cols = len (df_TS)-1
    last_year=df_TS.iloc[Num_cols,0]
    annual_peak_for_all_df = {'Year': [0], 'Annual peak (Block maxima) for all': [0.0]}
    annual_peak_for_all_df = pd.DataFrame(data=annual_peak_for_all_df)
    a=first_year
    while a<=last_year-1:
        m={'Year': [0.0], 'Annual peak (Block maxima) for all': [0.0]}
        m = pd.DataFrame(data=m)
        append_row=m.iloc[0].copy()
        annual_peak_for_all_df=annual_peak_for_all_df.append(append_row)
        a+=1
    annual_peak_for_all_df=annual_peak_for_all_df.reset_index(drop=True)
    a=first_year
    i=0
    while a<=last_year:
        annual_peak_for_all_df.xs(i)['Year'] = a
        a+=1
        i+=1
    a=first_year
    i=0
    j=0
    Num_cols = len (df_TS)
    max=0
    while a<=last_year:
        while j<=(Num_cols-1):
            if df_TS.iloc[j,0]==a:
                if df_TS.iloc[j,3]>max:
                    annual_peak_for_all_df.xs(i)['Annual peak (Block maxima) for all'] = df_TS.iloc[j
,3]
                    max=df_TS.iloc[j,3]
                    j+=1
                else:
                    j+=1
            else:
                j+=1
        max=0
        j=0
        i+=1
        a+=1
    return annual_peak_for_all_df

#Estimation of the average_Yi_df
def average_Yi_function(df_TS, events_df):
    first_year=df_TS.iloc[0,0]
    Num_cols = len (df_TS)-1
    last_year=df_TS.iloc[Num_cols,0]
    average_Yi_df = {'Year': [0], 'Average Yi': [0.0]}
    average_Yi_df = pd.DataFrame(data=average_Yi_df)
    a=first_year
    while a<=last_year-1:
        m={'Year': [0], 'Average Yi': [0.0]}
        m = pd.DataFrame(data=m)
        append_row=m.iloc[0].copy()
        average_Yi_df=average_Yi_df.append(append_row)
        a+=1
    average_Yi_df=average_Yi_df.reset_index(drop=True)
    a=first_year
    i=0
    while a<=last_year:
        average_Yi_df.xs(i)['Year'] = a
        a+=1
        i+=1

    a=first_year
```

```
    i=0
    j=0
    Num_cols = len (events_df)
    while a<=last_year:
        r=0
        u=0
        while j<=(Num_cols-1):
            if events_df.iloc[j,0]==a:
                r=r + events_df.iloc[j,3]
                u+=1
                j+=1
            else:
                j+=1
        if u!=0:
            average_Yi_df.xs(i)['Average Yi'] =r/u
        j=0
        i+=1
        a+=1
    return average_Yi_df

def ecdf_S(collective_risk_S_df):
    ecdf_S_df=collective_risk_S_df.drop(['Year'], axis=1)
    ecdf_S_df.columns = ['Large sorted S']
    ecdf_S_df=ecdf_S_df.sort_values(by=['Large sorted S'],ascending=False)
    ecdf_S_df=ecdf_S_df.reset_index(drop=True)
    Num_cols=len(ecdf_S_df)
    x=ecdf_S_df.copy()
    ecdf_S_df['ECDF']=x['Large sorted S']
    i=0
    while i<Num_cols:
        ecdf_S_df.iloc[i,1]=1-((ecdf_S_df.index[i]+1)/(Num_cols+1))
        i+=1
    return ecdf_S_df

def ecdf_r(return_interval_df):
    ecdf_r_df=return_interval_df
    ecdf_r_df.columns = ['Large sorted r']
    ecdf_r_df=ecdf_r_df.sort_values(by=['Large sorted r'],ascending=False)
    ecdf_r_df=ecdf_r_df.reset_index(drop=True)
    Num_cols=len(ecdf_r_df)
    x=ecdf_r_df.copy()
    ecdf_r_df['ECDF']=x['Large sorted r']
    i=0
    while i<Num_cols:
        ecdf_r_df.iloc[i,1]=1-((ecdf_r_df.index[i]+1)/(Num_cols+1))
        i+=1
    return ecdf_r_df

#Create 100 shuffled timeseries for observed timeseries
def Shuffle_100_timeseries(df_TS):
    df_TS.columns = ['Year', 'Month', 'Day', 'Precipitation']
    shuffled_df=df_TS
    shuffled_df=shuffled_df.drop(['Year', 'Month', 'Day'], axis=1)
    x=shuffled_df
    i=1
    while i<101:
        x = x.sample(frac=1, axis=1).sample(frac=1).reset_index(drop=True)
        shuffled_df[i]=x['Precipitation']
        i+=1
    shuffled_df.insert(loc=0, column='Day', value=df_TS['Day'])
    shuffled_df.insert(loc=0, column='Month', value=df_TS['Month'])
    shuffled_df.insert(loc=0, column='Year', value=df_TS['Year'])
    return shuffled_df

def Get_duration_df(return_interval_df):
    ret_int_index=0
    while ret_int_index<len(return_interval_df):
        if return_interval_df.iloc[ret_int_index,0]!=1:
            return_interval_df.iloc[ret_int_index,0]=0.0
        ret_int_index+=1
#    print(return_interval_df)
    ret_int_index=1
    Num_return_interval_df=len(return_interval_df)
    while ret_int_index<Num_return_interval_df:
```

```
        if return_interval_df.iloc[ret_int_index,0]==1.0:
            return_interval_df.iloc[ret_int_index-1,0]=return_interval_df.iloc[ret_int_index-1,0]+1
            return_interval_df=return_interval_df.drop(return_interval_df.index[ret_int_index])
            return_interval_df=return_interval_df.reset_index(drop=True)
            Num_return_interval_df=Num_return_interval_df-1
        else:
            ret_int_index+=1
    ret_int_index=0
    duration_df=return_interval_df.copy()
    duration_df.columns=['Duration of event']
    while ret_int_index<len(duration_df):
        duration_df.iloc[ret_int_index,0]=duration_df.iloc[ret_int_index,0]+1
        ret_int_index+=1
    return duration_df

def ecdf_duration(duration_df):
    duration_df.columns = ['Large sorted duration']
    duration_df=duration_df.sort_values(by=['Large sorted duration'],ascending=False)
    duration_df=duration_df.reset_index(drop=True)
    Num_cols=len(duration_df)
    x=duration_df.copy()
    duration_df['ECDF']=x['Large sorted duration']
    i=0
    while i<Num_cols:
        duration_df.iloc[i,1]=1-((duration_df.index[i]+1)/(Num_cols+1))
        i+=1
    return duration_df

def boxplot_Yi_N(correlation_label):
    file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 1999-2017\\3. Average Yi - N correlation\\Larissa Average Yi-N '+correlation_label+' (90%) - no zeros')
    file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 1999-2017\\3. Average Yi - N correlation\\Larissa Average Yi-N '+correlation_label+' (95%) - no zeros')
    file3=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 1999-2017\\3. Average Yi - N correlation\\Larissa Average Yi-N '+correlation_label+' (98%) - no zeros')
    file4=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 1999-2017\\3. Average Yi - N correlation\\Larissa Average Yi-N '+correlation_label+' (99%) - no zeros')
    # Create data
    collectn_1 = file1.iloc[0,1:100]
    collectn_11 = file1.iloc[0,0]
    collectn_2 = file2.iloc[0,1:100]
    collectn_22 = file2.iloc[0,0]
    collectn_3 = file3.iloc[0,1:100]
    collectn_33 = file3.iloc[0,0]
    collectn_4 = file4.iloc[0,1:100]
    collectn_44 = file4.iloc[0,0]
    # combine these different collections into a list
    data_to_plot = [collectn_1, collectn_11, collectn_2, collectn_22, collectn_3, collectn_33, collectn_4, collectn_44]
    # Create a figure instance
    fig=plt.figure(figsize=(11,6),dpi=200)
    ax = fig.add_subplot(111)
    medianprops = dict(linestyle='-', linewidth=2, color='red')
    ax.set_axisbelow(True)
    ax.set_title('Boxplot of the '+correlation_label+' correlation coefficient between \nAverage Yi and Number of Events (N) in Larissa Station (precipitation)',fontsize=14)
    ax.set_ylabel(correlation_label+' correlation coefficient',fontsize=13)
#    plt.tick_params(axis='x',which='x')
    ax.tick_params(axis='x', labelsize=10,zorder=20,rotation=0)
    bp = ax.boxplot(data_to_plot,medianprops=medianprops,patch_artist=True,labels=['90% Shuffled','90% Observed','95% Shuffled','95% Observed','98% Shuffled','98% Observed','99% Shuffled','99% Observed'])
    ax.grid(color='black', linestyle='-', linewidth=0.1)
    colors = ['skyblue','blue', 'moccasin','green', 'lightsalmon','tan', 'tan','pink']
    for patch, color in zip(bp['boxes'], colors):
        patch.set_facecolor(color)
    plt.yticks(np.arange(-1.0, 1.01, 0.2))
    plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 1999-2017\\3. Average Yi - N correlation\\'+correlation_label+' Boxplot Larissa.png')
    plt.close()
```

```
376.
377.def boxplot_compensations(correlation_label):
378.    file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 19
    99-2017\\4. Compensations correlation\\Larissa compensations '+correlation_label+' (90%) -
     no zeros')
379.    file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 19
    99-2017\\4. Compensations correlation\\Larissa compensations '+correlation_label+' (95%) -
     no zeros')
380.    file3=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 19
    99-2017\\4. Compensations correlation\\Larissa compensations '+correlation_label+' (98%) -
     no zeros')
381.    file4=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 19
    99-2017\\4. Compensations correlation\\Larissa compensations '+correlation_label+' (99%) -
     no zeros')
382.    # Create data
383.    collectn_1 = file1.iloc[0,1:100]
384.    collectn_11 = file1.iloc[0,0]
385.    collectn_2 = file2.iloc[0,1:100]
386.    collectn_22 = file2.iloc[0,0]
387.    collectn_3 = file3.iloc[0,1:100]
388.    collectn_33 = file3.iloc[0,0]
389.    collectn_4 = file4.iloc[0,1:100]
390.    collectn_44 = file4.iloc[0,0]
391.    # combine these different collections into a list
392.    data_to_plot = [collectn_1, collectn_11, collectn_2, collectn_22, collectn_3, collectn_33, collec
    tn_4, collectn_44]
393.    # Create a figure instance
394.    fig=plt.figure(figsize=(11,6),dpi=200)
395.    ax = fig.add_subplot(111)
396.    medianprops = dict(linestyle='-', linewidth=2, color='red')
397.    ax.set_axisbelow(True)
398.    ax.set_title('Boxplot of the '+correlation_label+' correlation coefficient between \nCollective R
    isk S and compensations in Larissa Station (precipitation)',fontsize=14)
399.    ax.set_ylabel(correlation_label+' correlation coefficient',fontsize=13)
400.#    plt.tick_params(axis='x',which='x')
401.    ax.tick_params(axis='x', labelsize=10,zorder=20,rotation=0)
402.    bp = ax.boxplot(data_to_plot,medianprops=medianprops,patch_artist=True,labels=['90% Shuffled','90
    % Observed','95% Shuffled','95% Observed','98% Shuffled','98% Observed','99% Shuffled','99% Observed'
    ])
403.    ax.grid(color='black', linestyle='-', linewidth=0.1)
404.    colors = ['skyblue','blue', 'moccasin','green', 'lightsalmon','tan', 'tan','pink']
405.    for patch, color in zip(bp['boxes'], colors):
406.        patch.set_facecolor(color)
407.    plt.yticks(np.arange(-1.0, 1.01, 0.2))
408.    plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 1999-
    2017\\4. Compensations correlation\\Compensations '+correlation_label+' Boxplot Larissa.png')
409.    plt.close()
410.
411.#Results of the observed time series 1955-2018
412.df = pd.read_excel('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\0. Scripts\\PRCT Larissa.xlsx
    ', sheet_name = '1669565')
413.#print(df)
414.df_TS=df[['DATE','PRCP']]
415.Num=len(df_TS)
416.i=0
417.while i<Num:
418.    df_TS.iloc[i,1]=df_TS.iloc[i,1]*25.4
419.    i+=1
420.#    print(i)
421.
422.df_TS['DATE'] =  pd.to_datetime(df_TS['DATE'], format='%m/%d/%Y')
423.df_TS = df_TS.sort_values(by=['DATE'], ascending=[True])
424.df_TS.set_index('DATE', inplace=True)
425.df_TS = df_TS.resample('D').asfreq().reset_index()
426.
427.df_TS['Year'] = pd.DatetimeIndex(df_TS['DATE']).year
428.df_TS['Month'] = pd.DatetimeIndex(df_TS['DATE']).month
429.df_TS['Day'] = pd.DatetimeIndex(df_TS['DATE']).day
430.df_TS['Precipitation']=df_TS['PRCP']
431.df_TS=df_TS.drop(['DATE'], axis=1)
432.df_TS=df_TS.drop(['PRCP'], axis=1)
433.
434.Num=len(df_TS)
435.i=0
```

```
while i<Num:
    if df_TS.iloc[i,0]==2019:
        df_TS=df_TS.drop(df_TS.index[i])
        df_TS=df_TS.reset_index(drop=True)
        Num=Num-1
    else:
        i+=1
#print(df_TS)
df_TS.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa df_TS')
df_TS=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa df_TS'
)

d = {'Year': [1955,1956,1957,1958,1959,1960,1961,1962,1963,1964,1965,1966,1967,1968,1969,1970,1971,19
72,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1988,1989,1990,1991,199
2,1993,1994,1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010,2011,2012
,2013,2014,2015,2016,2017,2018,0]}
final_prototype_df = pd.DataFrame(data=d)
Collective_Risk_S_FOR_ALL_df=final_prototype_df.copy()
number_of_events_FOR_ALL_df=final_prototype_df.copy()
annual_peak_for_all_FOR_ALL_df=final_prototype_df.copy()
average_Yi_FOR_ALL_df=final_prototype_df.copy()

threshold_df_data = {'Text': ['90%','95%','98%','99%'],'Numerical': [0.9, 0.95, 0.98, 0.99]}
threshold_df = pd.DataFrame(data=threshold_df_data)
Num_threshold_df=len(threshold_df)
#print(threshold_df)

threshold_loop=0
while threshold_loop<Num_threshold_df:
    threshold=threshold_df.iloc[threshold_loop,1]
    threshold_text=threshold_df.iloc[threshold_loop,0]
    i=0
    while i<1:
        filename1='Larissa'
        missing_values_per=missing_values_percentage(df_TS)
#        print(missing_values_percentage)
        events_df=Get_events_df(df_TS, threshold)
    #    print(events_df)
        events_df_without_ones=Get_events_df_without_ones(events_df)
        print(events_df_without_ones)
        return_interval_df=Get_return_interval_df(events_df_without_ones)
    #    print (return_interval_df)
        collective_risk_S_df=collective_risk_function(df_TS, events_df)
        collective_risk_S_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Laris
    sa\\1. Results for observed and shuffled\\'+threshold_text+' Collective Risk S '+filename1+' 1955-
    2018')
#        print(collective_risk_S_df)
        number_of_events_df=number_of_events_function(df_TS, events_df)
        number_of_events_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Lariss
    a\\1. Results for observed and shuffled\\'+threshold_text+' Number of events N '+filename1+' 1955-
    2018')
    #    print(number_of_events_df)
        annual_peak_only_for_events_df=annual_peak_only_for_events_function(df_TS, events_df)
    #    print(annual_peak_only_for_events_df)
        annual_peak_for_all_df=annual_peak_for_all_function(df_TS, events_df)
        annual_peak_for_all_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Lar
    issa\\1. Results for observed and shuffled\\'+threshold_text+' Annual Peak '+filename1+' 1955-
    2018')
    #    print(annual_peak_for_all_df)
        average_Yi_df=average_Yi_function(df_TS, events_df)
        average_Yi_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1.
    Results for observed and shuffled\\'+threshold_text+' Annual Average Yi '+filename1+' 1955-2018')
    #    print(average_Yi_df)
        ecdf_S_df=ecdf_S(collective_risk_S_df)
    #    print(ecdf_S_df)
        ecdf_S_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Resu
    lts for observed and shuffled\\'+threshold_text+' ecdf_S '+filename1)
        ecdf_r_df=ecdf_r(return_interval_df)
    #    print(ecdf_r_df)
        ecdf_r_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Resu
    lts for observed and shuffled\\'+threshold_text+' ecdf_r '+filename1)
        return_interval_df=Get_return_interval_df(events_df)
#        print(return_interval_df)
        duration_df_obs=Get_duration_df(return_interval_df)
```

```
        duration_df_obs.columns=['Observed Duration']
        ecdf_duration_df=ecdf_duration(duration_df_obs)
#         print(ecdf_duration_df)
        ecdf_duration_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Results for observed and shuffled\\Duration\\'+filename1+' ECDF DURATION OBSERVED '+threshold_text)
        shuffled_100_df=Shuffle_100_timeseries(df_TS)
    #    print(shuffled_100_df)
        shuffled_100_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Shuffled_100 '+filename1)
        i+=1
    threshold_loop+=1


#Results of the shuffled time series 1955-2018
how_many_shuffled_timeseries=100

threshold_loop=0
while threshold_loop<Num_threshold_df:
    threshold=threshold_df.iloc[threshold_loop,1]
    threshold_text=threshold_df.iloc[threshold_loop,0]
    i=0
    l=0
    while i<1:
        filename1='Larissa'
        pickle_file = pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Shuffled_100 '+filename1)
        j=0
        while j+3<how_many_shuffled_timeseries+3:
            df_TS=pickle_file[['Year', 'Month', 'Day',j+1]].copy()
            df_withoutNaN=df_TS.dropna(axis=0,how='any', thresh=None, subset=None, inplace=False)
    #         print(df_withoutNaN)
            df_sorted_noNaN=df_withoutNaN.sort_values(j+1, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last')
            Num_cols = len (df_withoutNaN)
            position=int(threshold*Num_cols)
            threshold_value=df_sorted_noNaN.iloc[position,3]
            events_df=df_withoutNaN
            events_df.columns = ['Year', 'Month', 'Day', 'Precipitation']
            events_df.drop(events_df.loc[events_df['Precipitation'] < threshold_value].index, inplace=True)
    #         print(events_df)
            events_df_without_ones=Get_events_df_without_ones(events_df)
    #        print(events_df_without_ones)
            return_interval_df=Get_return_interval_df(events_df_without_ones)
    #        print (return_interval_df)
            collective_risk_S_df=collective_risk_function(df_TS, events_df)
    #        print(collective_risk_S_df)
            number_of_events_df=number_of_events_function(df_TS, events_df)
    #        print(number_of_events_df)
            annual_peak_only_for_events_df=annual_peak_only_for_events_function(df_TS, events_df)
    #        print(annual_peak_only_for_events_df)
            annual_peak_for_all_df=annual_peak_for_all_function(df_TS, events_df)
    #        print(annual_peak_for_all_df)
            average_Yi_df=average_Yi_function(df_TS, events_df)
    #        print(average_Yi_df)
            return_interval_df=Get_return_interval_df(events_df)
    #        print(return_interval_df)
            duration_df_shuffled=Get_duration_df(return_interval_df)
            duration_df_shuffled.columns=['Shuffled Duration']
            ecdf_duration_df=ecdf_duration(duration_df_shuffled)
    #        print(ecdf_duration_df)
            ecdf_duration_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Results for observed and shuffled\\Duration\\'+filename1+' ECDF DURATION SHUFFLED '+str(j+1)+' '+threshold_text)

            if j==0:
                ecdf_S_df=ecdf_S(collective_risk_S_df)
    #             print(ecdf_S_df)
                ecdf_r_df=ecdf_r(return_interval_df)
    #             print(ecdf_r_df)
                l+=2
            else:
                u=ecdf_S(collective_risk_S_df)
```

```
                u.columns = ['Large sorted S', 'ECDF']
                ecdf_S_df[l]=u['Large sorted S']
                ecdf_S_df[l+1]=u['ECDF']
    #            print(ecdf_S_df)
                u=ecdf_r(return_interval_df)
                u.columns = ['Large sorted r', 'ECDF']
                ecdf_r_df[l]=u['Large sorted r']
                ecdf_r_df[l+1]=u['ECDF']
    #            print(ecdf_r_df)
                l+=2
            Collective_Risk_S_FOR_ALL_df[j]=collective_risk_S_df['Collective Risk S']
            number_of_events_FOR_ALL_df[j]=number_of_events_df['Number of events']
            annual_peak_for_all_FOR_ALL_df[j]=annual_peak_for_all_df['Annual peak (Block maxima) for all']
            average_Yi_FOR_ALL_df[j]=average_Yi_df['Average Yi']
#            print(filename1+' shuffle:'+str(j))
            j+=1

        ecdf_S_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Results for observed and shuffled\\'+threshold_text+' 100 ecdf_S '+filename1)
        ecdf_r_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Results for observed and shuffled\\'+threshold_text+' 100 ecdf_r '+filename1)
        Collective_Risk_S_FOR_ALL_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Results for observed and shuffled\\'+threshold_text+' Collective risk for 100 shuffled '+filename1)
        number_of_events_FOR_ALL_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Results for observed and shuffled\\'+threshold_text+' Number of events N for 100 shuffled '+filename1)
        annual_peak_for_all_FOR_ALL_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Results for observed and shuffled\\'+threshold_text+' Annual peak for 100 shuffled '+filename1)
        average_Yi_FOR_ALL_df.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Results for observed and shuffled\\'+threshold_text+' Annual Average Yi  for 100 Shuffled for 100 shuffled '+filename1)
        i+=1
    threshold_loop+=1

#Plot of ECDF for collective risk (S), return intervals (r) and duration of over-threshold events 1955-2018
threshold_loop=0
while threshold_loop<Num_threshold_df:
    threshold=threshold_df.iloc[threshold_loop,1]
    threshold_text=threshold_df.iloc[threshold_loop,0]
#    Export ECDF-S diagram
    j=1
    while j<=1:
        filename1='Larissa'
        file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Results for observed and shuffled\\'+threshold_text+' 100 ecdf_S Larissa')
        file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Results for observed and shuffled\\'+threshold_text+' ecdf_S Larissa')
        sns.set()
        sns.set_style("ticks")
        plt.figure(figsize=(8,5),dpi=200)
        for i in  np.arange(1,200,2):
            ecdf_S_=file1.iloc[:,i]
            colrisk=file1.iloc[:,i-1]
            plt.plot(colrisk,ecdf_S_,label='',color='darkgrey')
        plt.plot(colrisk,ecdf_S_,label='Shuffled',color='darkgrey')
        plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='Observed', marker='o', markersize=3, color='r')
        plt.title('Larissa station - Threshold '+threshold_text,fontsize=13, y=1.01)
        plt.ylabel("ECDF",fontsize=14)
        plt.xlabel("Collective Risk (S)",fontsize=14)
        plt.yticks(np.arange(0, 1.1, 0.1))
        plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
        plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(0.95, 0.4),loc=1,ncol=1,fontsize=14)
        plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
        plt.tight_layout()
        plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\2. ECDF diagrams for S and r\\Linear ECDF_S '+filename1+' '+threshold_text+'.png',facecolour='white')
        plt.close()
        j+=1

```

```
    #Export ECDF-r diagram
    j=1
    while j<=1:
        filename1='Larissa'
        file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Results for observed and shuffled\\'+threshold_text+' 100 ecdf_r Larissa')
        file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Results for observed and shuffled\\'+threshold_text+' ecdf_r Larissa')
        sns.set()
        sns.set_style("ticks")
        plt.figure(figsize=(8,5),dpi=200)
        for i in  np.arange(1,200,2):
            ecdf_ri=file1.iloc[:,i]
            retint=file1.iloc[:,i-1]
            plt.plot(retint,ecdf_ri,label='',color='darkgrey')
        plt.plot(retint,ecdf_ri,label='Shuffled',color='darkgrey')
        plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='Observed', marker='o', markersize=3, color='r')
        plt.title('Larissa station - Threshold '+threshold_text,fontsize=13, y=1.01)
        plt.ylabel("ECDF",fontsize=14)
        plt.xlabel("Return Interval (r)",fontsize=14)
        plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
        plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(0.95, 0.4),loc=1,ncol=1,fontsize=14)
        plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
        plt.tight_layout()
        plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\2. ECDF diagrams for S and r\\Linear ECDF_r '+filename1+' '+threshold_text+'.png',facecolour='white')
        plt.close()
        j+=1

#    Export LOG-LOG ECDF-S diagram
    j=1
    while j<=1:
        filename1='Larissa'
        file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Results for observed and shuffled\\'+threshold_text+' 100 ecdf_S Larissa')
        file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Results for observed and shuffled\\'+threshold_text+' ecdf_S Larissa')
        sns.set()
        sns.set_style("ticks")
        plt.figure(figsize=(8,5),dpi=200)
        for i in  np.arange(1,200,2):
            ecdf_S_=file1.iloc[:,i]
            colrisk=file1.iloc[:,i-1]
            plt.plot(colrisk,ecdf_S_,label='',color='darkgrey')
        plt.plot(colrisk,ecdf_S_,label='Shuffled',color='darkgrey')
        plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='Observed', marker='o', markersize=3, color='r')
        plt.title('Larissa station - Threshold '+threshold_text,fontsize=13, y=1.01)
        plt.ylabel("ECDF",fontsize=14)
        plt.xlabel("Collective Risk (S)",fontsize=14)
        plt.xscale('log')
        plt.yscale('log')
        labels = ['0.05', '0.1', '0.2', '0.5', '1.0']
        nthreads = [0.05, 0.1, 0.2, 0.5, 1.0]
        plt.yticks(nthreads, labels)
        plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
        plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(0.95, 0.4),loc=1,ncol=1,fontsize=14)
        plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
        plt.tight_layout()
        plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\2. ECDF diagrams for S and r\\Log ECDF_S '+filename1+' '+threshold_text+'.png',facecolour='white')
        plt.close()
        j+=1

    #Export LOG-LOG ECDF-r diagram
    j=1
    while j<=1:
        filename1='Larissa'
        file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Results for observed and shuffled\\'+threshold_text+' 100 ecdf_r Larissa')
        file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Results for observed and shuffled\\'+threshold_text+' ecdf_r Larissa')
        sns.set()
```

```
        sns.set_style("ticks")
        plt.figure(figsize=(8,5),dpi=200)
        for i in  np.arange(1,200,2):
            ecdf_ri=file1.iloc[:,i]
            retint=file1.iloc[:,i-1]
            plt.plot(retint,ecdf_ri,label='',color='darkgrey')
        plt.plot(retint,ecdf_ri,label='Shuffled',color='darkgrey')
        plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='Observed', marker='o', markersize=3, color='r
')
        plt.title('Larissa station - Threshold '+threshold_text,fontsize=13, y=1.01)
        plt.ylabel("ECDF",fontsize=14)
        plt.xlabel("Return Interval (r)",fontsize=14)
        plt.xscale('log')
        plt.yscale('log')
        labels = ['0.05', '0.1', '0.2', '0.5', '1.0']
        nthreads = [0.05, 0.1, 0.2, 0.5, 1.0]
        plt.yticks(nthreads, labels)
        plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
        plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(0.95, 0.4),loc=1,ncol=1,fontsize=14)
        plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
        plt.tight_layout()
        plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\2. ECDF diagram
    s for S and r\\Log ECDF_r '+filename1+' '+threshold_text+'.png',facecolour='white')
        plt.close()
        j+=1

#    Export 1/(1-f) ECDF-S diagram
    j=1
    while j<=1:
        filename1='Larissa'
        file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Res
    ults for observed and shuffled\\'+threshold_text+' 100 ecdf_S Larissa')
        file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Res
    ults for observed and shuffled\\'+threshold_text+' ecdf_S Larissa')
        sns.set()
        sns.set_style("ticks")
        plt.figure(figsize=(8,5),dpi=200)
        for i in  np.arange(1,200,2):
            ecdf_S_=1/(1-file1.iloc[:,i])
            colrisk=file1.iloc[:,i-1]
            plt.plot(colrisk,ecdf_S_,label='',color='darkgrey')
        plt.plot(colrisk,ecdf_S_,label='Shuffled',color='darkgrey')
        plt.plot(file2.iloc[:,0],1/(1-
    file2.iloc[:,1]),label='Observed', marker='o', markersize=3, color='r')
        plt.title('Larissa station - Threshold '+threshold_text,fontsize=13, y=1.01)
        plt.ylabel("1/(1-ECDF)",fontsize=14)
        plt.xlabel("Collective Risk (S)",fontsize=14)
        plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
        plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(1.005, 0.215),loc=1,ncol=1,fontsize=14)
        plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
        plt.tight_layout()
        plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\2. ECDF diagram
    s for S and r\\1--(1-ECDF) ECDF_S '+filename1+' '+threshold_text+'.png',facecolour='white')
        plt.close()
        j+=1

    #Export 1/(1-f) ECDF-r diagram
    j=1
    while j<=1:
        filename1='Larissa'
        file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Res
    ults for observed and shuffled\\'+threshold_text+' 100 ecdf_r Larissa')
        file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Res
    ults for observed and shuffled\\'+threshold_text+' ecdf_r Larissa')
        sns.set()
        sns.set_style("ticks")
        plt.figure(figsize=(8,5),dpi=200)
        for i in  np.arange(1,200,2):
            ecdf_ri=1/(1-file1.iloc[:,i])
            retint=file1.iloc[:,i-1]
            plt.plot(retint,ecdf_ri,label='',color='darkgrey')
        plt.plot(retint,ecdf_ri,label='Shuffled',color='darkgrey')
        plt.plot(file2.iloc[:,0],1/(1-
    file2.iloc[:,1]),label='Observed', marker='o', markersize=3, color='r')
```

```
        plt.title('Larissa station - Threshold '+threshold_text,fontsize=13, y=1.01)
        plt.ylabel("1/(1-ECDF)",fontsize=14)
        plt.xlabel("Return Interval (r)",fontsize=14)
        plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
        plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(1, 1),loc=1,ncol=1,fontsize=14)
        plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
        plt.tight_layout()
        plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\2. ECDF diagram
    s for S and r\\1--(1-ECDF) ECDF_r '+filename1+' '+threshold_text+'.png',facecolour='white')
        plt.close()
        j+=1

    #Export ECDF-Duration diagram
    j=1
    while j<=1:
        filename1='Larissa'
        file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Res
    ults for observed and shuffled\\Duration\\'+filename1+' ECDF DURATION OBSERVED '+threshold_text)
        sns.set()
        sns.set_style("ticks")
        plt.figure(figsize=(8,5),dpi=200)
        shuffle_index=1
        while shuffle_index<=how_many_shuffled_timeseries:
            file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1.
     Results for observed and shuffled\\Duration\\'+filename1+' ECDF DURATION SHUFFLED '+str(shuffle_inde
    x)+' '+threshold_text)
            plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='',color='darkgrey')
            shuffle_index+=1
        plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='Shuffled',color='darkgrey')
        plt.plot(file1.iloc[:,0],file1.iloc[:,1],label='Observed', marker='o', markersize=3, color='r
    ')
        plt.title('Larissa station - Threshold '+threshold_text,fontsize=13, y=1.01)
        plt.ylabel("ECDF",fontsize=14)
        plt.xlabel("Duration of events (days)",fontsize=14)
        plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
        plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(1, 0.22),loc=1,ncol=1,fontsize=14)
        plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
        plt.tight_layout()
        plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\2. ECDF diagram
    s for S and r\\Linear ECDF Duration '+filename1+' '+threshold_text+'.png',facecolour='white')
        plt.close()
        j+=1

    #Export LOG-LOG ECDF-Duration diagram
    j=1
    while j<=1:
        filename1='Larissa'
        file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Res
    ults for observed and shuffled\\Duration\\'+filename1+' ECDF DURATION OBSERVED '+threshold_text)
        sns.set()
        sns.set_style("ticks")
        plt.figure(figsize=(8,5),dpi=200)
        shuffle_index=1
        while shuffle_index<=how_many_shuffled_timeseries:
            file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1.
     Results for observed and shuffled\\Duration\\'+filename1+' ECDF DURATION SHUFFLED '+str(shuffle_inde
    x)+' '+threshold_text)
            plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='',color='darkgrey')
            shuffle_index+=1
        plt.plot(file2.iloc[:,0],file2.iloc[:,1],label='Shuffled',color='darkgrey')
        plt.plot(file1.iloc[:,0],file1.iloc[:,1],label='Observed', marker='o', markersize=3, color='r
    ')
        plt.title('Larissa station - Threshold '+threshold_text,fontsize=13, y=1.01)
        plt.ylabel("ECDF",fontsize=14)
        plt.xlabel("Duration of events (days)",fontsize=14)
        plt.xscale('log')
        plt.yscale('log')
        labels = ['0.05', '0.1', '0.2', '0.5', '1.0']
        nthreads = [0.05, 0.1, 0.2, 0.5, 1.0]
        plt.yticks(nthreads, labels)
        plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
        plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(1, 0.22),loc=1,ncol=1,fontsize=14)
        plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
        plt.tight_layout()
```

```
814.        plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\2. ECDF diagram
    s for S and r\\LOG ECDF Duration '+filename1+' '+threshold_text+'.png',facecolour='white')
815.        plt.close()
816.        j+=1
817.
818.    #Export 1/(1-f) ECDF-Duration diagram
819.    j=1
820.    while j<=1:
821.        filename1='Larissa'
822.        file1=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1. Res
    ults for observed and shuffled\\Duration\\'+filename1+' ECDF DURATION OBSERVED '+threshold_text)
823.        sns.set()
824.        sns.set_style("ticks")
825.        plt.figure(figsize=(8,5),dpi=200)
826.        shuffle_index=1
827.        while shuffle_index<=how_many_shuffled_timeseries:
828.            file2=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\1.
     Results for observed and shuffled\\Duration\\'+filename1+' ECDF DURATION SHUFFLED '+str(shuffle_inde
    x)+' '+threshold_text)
829.            plt.plot(file2.iloc[:,0],1/(1-file2.iloc[:,1]),label='',color='darkgrey')
830.            shuffle_index+=1
831.        plt.plot(file2.iloc[:,0],1/(1-file2.iloc[:,1]),label='Shuffled',color='darkgrey')
832.        plt.plot(file1.iloc[:,0],1/(1-
    file1.iloc[:,1]),label='Observed', marker='o', markersize=3, color='r')
833.        plt.title('Larissa station - Threshold '+threshold_text,fontsize=13, y=1.01)
834.        plt.ylabel("1/(1-ECDF)",fontsize=14)
835.        plt.xlabel("Duration of events (days)",fontsize=14)
836.        plt.grid(b=True, which='major',axis='both', color='0.90',linestyle='-',zorder=2)
837.        plt.legend(frameon=1,fancybox=1,bbox_to_anchor=(1.005, 1.008),loc=1,ncol=1,fontsize=14)
838.        plt.tick_params(axis='both',which='both',labelsize=14,zorder=20)
839.        plt.tight_layout()
840.        plt.savefig('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\2. ECDF diagram
    s for S and r\\1--(1-ECDF) ECDF Duration '+filename1+' '+threshold_text+'.png',facecolour='white')
841.        plt.close()
842.        j+=1
843.
844.    threshold_loop+=1
845.
846. #Boxplot of Average Yi -
     N Pearson and Spearman correlation coefficient for observed and shuffled 1955-2018
847.a = np.zeros(shape=(1,101))
848.zero_df = pd.DataFrame(a)
849.#print(df)
850.
851.pearson_corr_coef=zero_df.copy()
852.spearman_corr_coef=zero_df.copy()
853.
854.threshold_loop=0
855.while threshold_loop<Num_threshold_df:
856.    threshold=threshold_df.iloc[threshold_loop,1]
857.    threshold_text=threshold_df.iloc[threshold_loop,0]
858.
859.    i=0
860.    while i<1:
861.        filename='Larissa'
862.        Yi=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 1
    999-
    2017\\1. Results for observed and shuffled\\'+threshold_text+' Annual Average Yi  for 100 Shuffled fo
    r 100 shuffled Larissa')
863.        N=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 19
    99-
    2017\\1. Results for observed and shuffled\\'+threshold_text+' Number of events N for 100 shuffled La
    rissa')
864.        Yi=Yi.drop(['Year'], axis=1)
865.        N=N.drop(['Year'], axis=1)
866.        N=N.drop(N.index[len(N)-1])
867.        Yi=Yi.drop(Yi.index[len(Yi)-1])
868.        j=0
869.        while j<100:
870.            Yi_column=Yi[j]
871.            N_column=N[j]
872.            z=0
873.            k=len(Yi)
874.            while z<k:
```

```
                if N_column.iloc[z]==0:
                    N_column=N_column.drop(N_column.index[z])
                    Yi_column=Yi_column.drop(Yi_column.index[z])
                    N_column=N_column.reset_index(drop=True)
                    Yi_column=Yi_column.reset_index(drop=True)
                    k=k-1
                else:
                    z+=1
            corr1,p_value1=pearsonr(N_column, Yi_column)
            pearson_corr_coef.iloc[i,j+1]=corr1
            corr2,p_value1=spearmanr(N_column, Yi_column)
            spearman_corr_coef.iloc[i,j+1]=corr2
            j+=1
        Yi=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 1999-2017\\1. Results for observed and shuffled\\'+threshold_text+' Annual Average Yi Larissa 1955-2018')
        N=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 1999-2017\\1. Results for observed and shuffled\\'+threshold_text+' Number of events N Larissa 1955-2018')
        Yi=Yi.drop(['Year'], axis=1)
        N=N.drop(['Year'], axis=1)
        N=N.drop(N.index[len(N)-1])
        Yi=Yi.drop(Yi.index[len(Yi)-1])
        Yi_column=Yi.copy()
        N_column=N.copy()
        z=0
        k=len(Yi)
        while z<k:
            if N_column.iloc[z,0]==0:
                N_column=N_column.drop(N_column.index[z])
                Yi_column=Yi_column.drop(Yi_column.index[z])
                N_column=N_column.reset_index(drop=True)
                Yi_column=Yi_column.reset_index(drop=True)
                k=k-1
            else:
                z+=1
        corr1,p_value1=pearsonr(N_column, Yi_column)
        pearson_corr_coef.iloc[i,0]=corr1
        corr2,p_value1=spearmanr(N_column, Yi_column)
        spearman_corr_coef.iloc[i,0]=corr2

        i+=1
#    print(pearson_corr_coef)
#    print(spearman_corr_coef)
    pearson_corr_coef.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 1999-2017\\3. Average Yi - N correlation\\Larissa Average Yi-N Pearson ('+threshold_text+') - no zeros')
    spearman_corr_coef.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 1999-2017\\3. Average Yi - N correlation\\Larissa Average Yi-N Spearman ('+threshold_text+') - no zeros')

    threshold_loop+=1

correlation_label='Spearman'
boxplot_Yi_N(correlation_label)
correlation_label='Pearson'
boxplot_Yi_N(correlation_label)


#Boxplot of collective risk and compensations Pearson and Spearman correlation coefficient for observed and shuffled 1999-2017
compensations= {'Year': [1999,2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010,2011,2012,2013,2014,2015,2016,2017],'Compensations': [13493.73,64195,293.22,569450.823,138472.615,46883.5,1878.74,12410.36,30717.94,642.6,261853.335,29461.93,0,217043.16,321.25,6830.66,6148.71,230394.12,311303.87]}
compensations_df = pd.DataFrame(data=compensations)

a = np.zeros(shape=(1,101))
zero_df = pd.DataFrame(a)
#print(df)

pearson_corr_coef=zero_df.copy()
spearman_corr_coef=zero_df.copy()

threshold_loop=0
```

```
938.while threshold_loop<Num_threshold_df:
939.    threshold=threshold_df.iloc[threshold_loop,1]
940.    threshold_text=threshold_df.iloc[threshold_loop,0]
941.#
942.    #without zeros
943.    i=0
944.    while i<1:
945.        filename='Larissa'
946.        Coll_risk=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 1999-
    2017\\1. Results for observed and shuffled\\'+threshold_text+' Collective risk for 100 shuffled Larissa')
947.        Coll_risk=Coll_risk.drop(['Year'], axis=1)
948.        Coll_risk['Compensations']=Coll_risk[0]
949.        Coll_risk=Coll_risk.drop(Coll_risk.index[len(Coll_risk)-1])
950.        index=0
951.        while index<len(Coll_risk):
952.            Coll_risk.iloc[index,100]=compensations_df.iloc[index,1]
953.            index+=1
954.        j=0
955.        while j<100:
956.            Coll_risk_column=Coll_risk[j]
957.            Comp_df=Coll_risk.iloc[:,100]
958.            z=0
959.            k=len(Coll_risk)
960.            while z<k:
961.                if Comp_df.iloc[z]==0 or Coll_risk_column.iloc[z]==0:
962.                    Comp_df=Comp_df.drop(Comp_df.index[z])
963.                    Coll_risk_column=Coll_risk_column.drop(Coll_risk_column.index[z])
964.                    Comp_df=Comp_df.reset_index(drop=True)
965.                    Coll_risk_column=Coll_risk_column.reset_index(drop=True)
966.                    k=k-1
967.                else:
968.                    z+=1
969.            corr1,p_value1=pearsonr(Comp_df, Coll_risk_column)
970.            pearson_corr_coef.iloc[i,j+1]=corr1
971.            corr2,p_value1=spearmanr(Comp_df, Coll_risk_column)
972.            spearman_corr_coef.iloc[i,j+1]=corr2
973.            j+=1
974.        Coll_risk_obs=pd.read_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 1999-
    2017\\1. Results for observed and shuffled\\'+threshold_text+' Collective Risk S Larissa 1955-2018')
975.        Coll_risk_obs=Coll_risk_obs.drop(['Year'], axis=1)
976.        Coll_risk['Observed']=Coll_risk_obs['Collective Risk S']
977.        Coll_risk_column=Coll_risk.iloc[:,101]
978.        Comp_df=Coll_risk.iloc[:,100]
979.        z=0
980.        k=len(Coll_risk_column)
981.        while z<k:
982.            if Comp_df.iloc[z]==0 or Coll_risk_column.iloc[z]==0:
983.                Comp_df=Comp_df.drop(Comp_df.index[z])
984.                Coll_risk_column=Coll_risk_column.drop(Coll_risk_column.index[z])
985.                Comp_df=Comp_df.reset_index(drop=True)
986.                Coll_risk_column=Coll_risk_column.reset_index(drop=True)
987.                k=k-1
988.            else:
989.                z+=1
990.        corr1,p_value1=pearsonr(Comp_df, Coll_risk_column)
991.        pearson_corr_coef.iloc[i,0]=corr1
992.        corr2,p_value1=spearmanr(Comp_df, Coll_risk_column)
993.        spearman_corr_coef.iloc[i,0]=corr2
994.
995.        i+=1
996.#    print(pearson_corr_coef)
997.#    print(spearman_corr_coef)
998.    pearson_corr_coef.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 1999-2017\\4. Compensations correlation\\Larissa compensations Pearson ('+threshold_text+') - no zeros')
999.    spearman_corr_coef.to_pickle('C:\\Users\\Konstantinos\\Documents\\THESIS\\Python\\14. Larissa\\Larissa 1999-2017\\4. Compensations correlation\\Larissa compensations Spearman ('+threshold_text+') - no zeros')
1000.
1001.           threshold_loop+=1
```

```
1002.
1003.       correlation_label='Spearman'
1004.       boxplot_compensations(correlation_label)
1005.       correlation_label='Pearson'
1006.       boxplot_compensations(correlation_label)
```

12. R scripts

The following scripts, originally created by K. Tay (2019), are modified in order to make the graphical and diagrammatic visualization of the FEMA's NFIP claims records in chapter 5.

12.1. Load libraries and dataset

```
library(tidyverse)
library(choroplethr)
library(choroplethrMaps)
df <- read_csv("openfema_claims20190331.csv",
               col_types = cols(asofdate = col_date(format = "%Y-%m-%d"),
                                dateofloss = col_date(format = "%Y-%m-%d"),
                                originalconstructiondate = col_date(format = "%Y-%m-%d"),
                                originalnbdate = col_date(format = "%Y-%m-%d")))
str(df)
```

12.2. Select the columns we analyze and remove specific rows

```
# select just the columns we want and remove rows with no state info or
# negative claim amounts
small_df <- df %>%
    select(year = yearofloss, state, countycode,
           claim_bldg = amountpaidonbuildingclaim,
           claim_contents = amountpaidoncontentsclaim,
           claim_ICC = amountpaidonincreasedcostofcomplianceclaim) %>%
    filter(!is.na(state)) %>%
    replace_na(list(claim_bldg = 0, claim_contents = 0, claim_ICC = 0)) %>%
    filter(!(claim_bldg < 0 | claim_contents < 0 | claim_ICC < 0))
 # filter for just years between 1978 and 2018 inclusive,
 # add total cost column
 small_df <- small_df %>% filter(year >= 1978 & year <= 2018) %>%
     mutate(total_cost = claim_bldg + claim_contents + claim_ICC)
```

12.3. Histogram of the number of claims per year

```
# histogram of year
with(small_df, hist(year, breaks = c(min(year):max(year)),
                    xlab = "Year"))
```

12.4. Number of claims by state

```
state_no_claims_df <- small_df %>% group_by(state) %>%
    summarize(claim_cnt = n())
# map to state name
data(state.regions)
state_no_claims_df$region <- plyr::mapvalues(state_no_claims_df$state,
                                             from = state.regions$abb,
                                             to = state.regions$region)
# rename columns for choroplethr
names(state_no_claims_df) <- c("state", "value", "region")
```

```
# basic map
state_choropleth(state_no_claims_df)
# above and below median
state_choropleth(state_no_claims_df, num_colors = 2)
# continuous scale
state_choropleth(state_no_claims_df, num_colors = 1)

# zoom in on gulf states: have a shoreline on gulf of mexico
gulf_states <- c("texas", "louisiana", "mississippi", "alabama", "florida")
state_choropleth(state_no_claims_df, num_colors = 1,
                 title = "No. of claims by state (Gulf states)", legend = "No. of claims",
                 zoom = gulf_states)
# bar plot of no. of claims
ggplot(data = state_no_claims_df %>% arrange(desc(value)) %>% head(n = 10)) +
    geom_bar(aes(x = reorder(state, -value), y = value), stat = "identity",
             fill = "gray", col = "black") +
    geom_abline(intercept = 100000, slope = 0, col = "red", lty = 2) +
    labs(x = "State", y = "No. of claims", title = "Top 10 states by no. of claims")
##########
# make plot of total claim amounts by state
##########
state_claims_amt_df <- small_df %>% group_by(state) %>%
    summarize(value = log10(sum(total_cost)))
# bar plot of claim amt
ggplot(data = state_claims_amt_df %>% arrange(desc(value)) %>% head(n = 10)) +
    geom_bar(aes(x = reorder(state, -value), y = 10^value), stat = "identity",
             fill = "gray", col = "black") +
    geom_abline(intercept = 10^9, slope = 0, col = "red", lty = 2) +
    scale_y_continuous(labels = function(x) format(x, scientific = TRUE)) +
    labs(x = "State", y = "Total claims", title = "Top 10 states by total claims")
```

12.5. Number of claims by county

```
county_summary_df <- small_df %>% group_by(countycode) %>%
    summarize(claim_cnt = n(), claim_amt = sum(total_cost)) %>%
    mutate(countycode = as.numeric(countycode))
names(county_summary_df) <- c("region", "value", "claim_amt")
county_choropleth(county_summary_df)
# change fill color for NAs to be white
county_choropleth(county_summary_df, title = "No. of claims by county") +
    scale_fill_brewer(na.value = "white")
# zoom into florida only
county_choropleth(county_summary_df, title = "No. of claims by county (Florida)",
                  state_zoom = "florida") +
    scale_fill_brewer(na.value = "white")
# zoom into gulf
county_choropleth(county_summary_df, title = "No. of claims by county (Gulf states)",
                  state_zoom = gulf_states) +
    scale_fill_brewer(na.value = "white")
```