nonstat package user guide

2020-04-02

Introduction

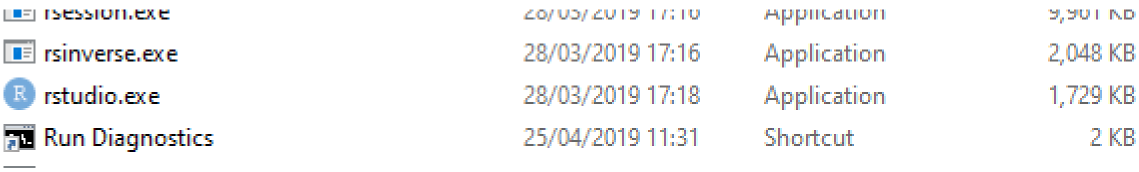
The nonstat R package is designed to allow hydrologists to easily undertake trend and change point detection and non-stationary flood frequency analysis using annual maximum (AMAX) flow data. The package contains a number of functions that can be called by the user and which have associated help files. This document provides more details about the functions, their inputs and outputs, and how to use them. This user guide is supported by two other documents. The first is practitioner guidance ('Allowing for non-stationarity in flood frequency estimation') which aims to introduce practitioners to the concepts of non-stationary flood frequency analysis and guide them when and how they should do it and how to interpret the results. It illustrates the concepts with three case studies. The second is a science report ('Development of Interim National Guidance on Non-Stationary Fluvial Flood Frequency Estimation'), which gives more scientific background to the methods implemented in the nonstat package, as well as nationwide results from applying the methods to the gauging station network in England and Wales.

It is recommended that analysis is carried out using RStudio, a free and open-source integrated development environment (IDE) for R. A version of R of at least 3.5.0 is required.

This documentation contains examples of code for running each function independently and there is therefore some repetition throughout the examples of assigning the same values to the same variables. However, there is also a complete example script at the end of the document for all aspects of analysis that can be carried out using this package and this avoids the repetition. All examples here use a folder called 'Non-stationarity' saved on the C drive, however, you can change the file path to anything you wish. The analysis does not need to be carried out in the same folder in which the nonstat package is saved.

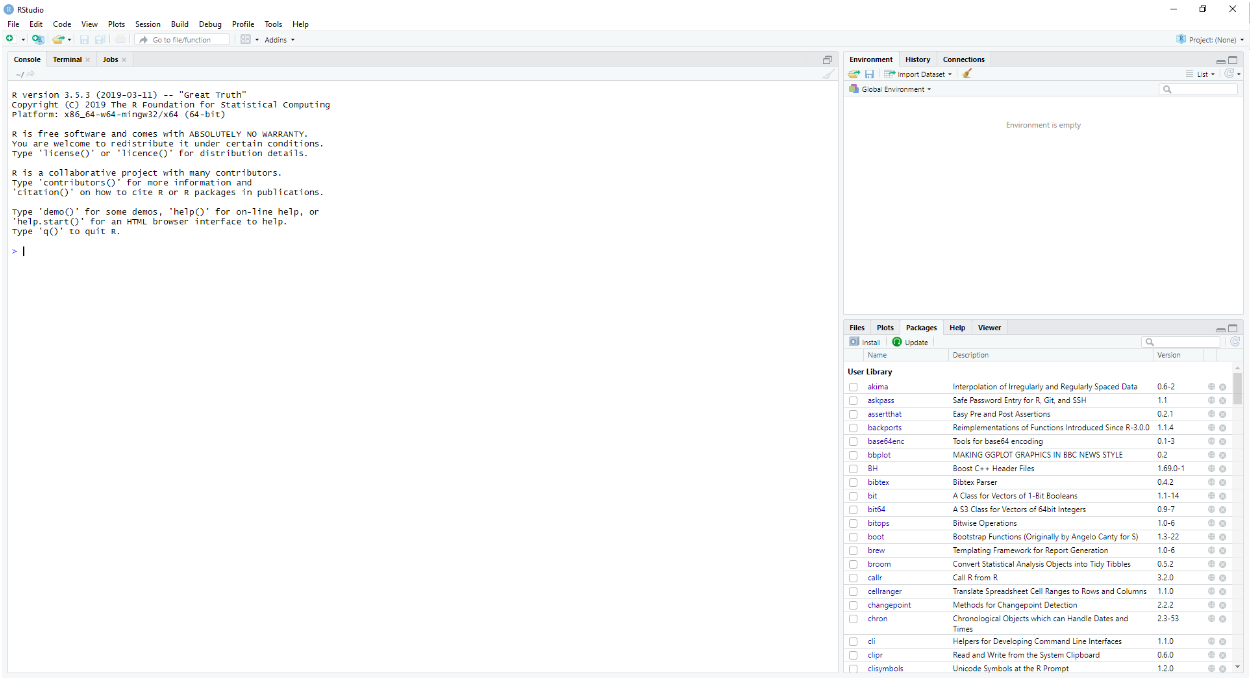
RStudio

Once you have installed RStudio, a shortcut should have been created which you can use to open it. If not, you can find the application in the directory you selected for saving RStudio under 'bin/rstudio.exe' (assuming the default settings have been followed):



```
rsinverse.exe        28/03/2019 17:16    Application    2,048 KB
rstudio.exe          28/03/2019 17:18    Application    1,729 KB
Run Diagnostics      25/04/2019 11:31    Shortcut           2 KB
```

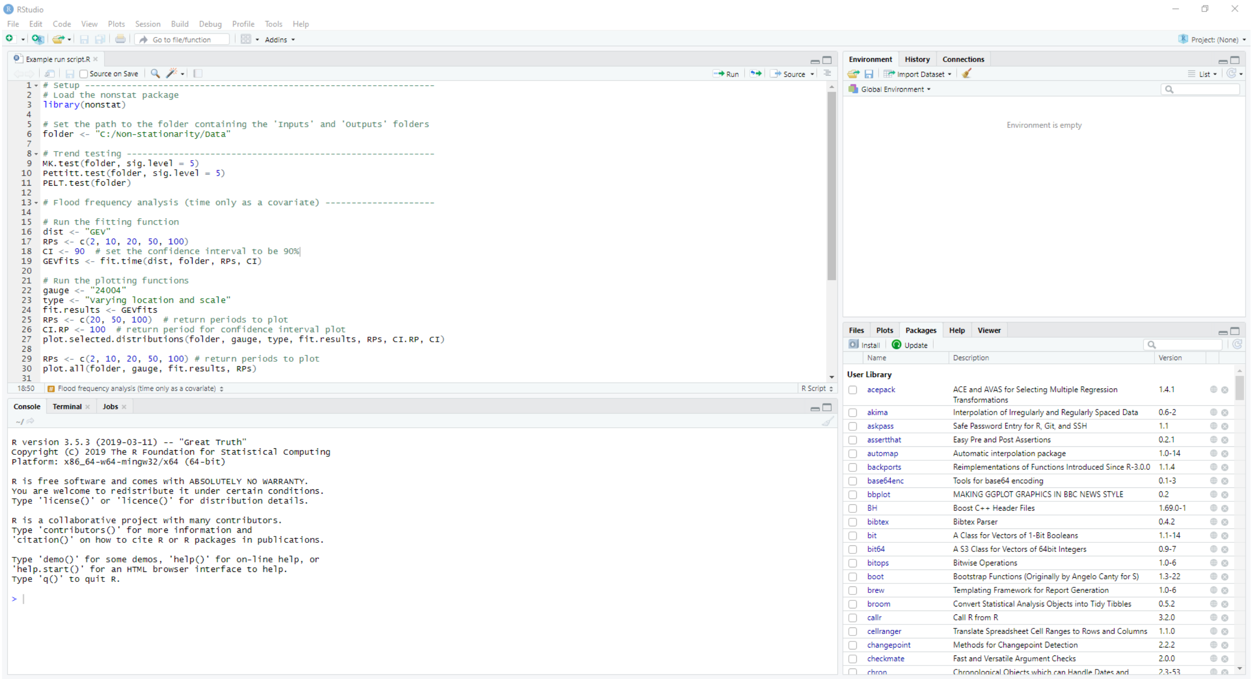
When you first open RStudio, it will look similar to this, although there may also be a box in the top left quadrant:



```
R version 3.5.3 (2019-03-11) -- "Great Truth"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

	Name	Description	Version
	User Library		
☐	akima	Interpolation of Irregularly and Regularly Spaced Data	0.6-2
☐	askpass	Safe Password Entry for R, Git, and SSH	1.1
☐	assertthat	Easy Pre and Post Assertions	0.2.1
☐	backports	Reimplementations of Functions Introduced Since R-3.0.0	1.1.4
☐	base64enc	Tools for base64 encoding	0.1-3
☐	bbplot	MAKING GGPLOT GRAPHICS IN BBC NEWS STYLE	0.2
☐	BH	Boost C++ Header Files	1.69.0-1
☐	bibtex	Bibtex Parser	0.4.2
☐	bit	A Class for Vectors of 1-Bit Booleans	1.1-14
☐	bit64	A S3 Class for Vectors of 64bit Integers	0.9-7
☐	bitops	Bitwise Operations	1.0-6
☐	boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-22
☐	brew	Templating Framework for Report Generation	1.0-6
☐	broom	Convert Statistical Analysis Objects into Tidy Tibbles	0.5.2
☐	callr	Call R from R	3.2.0
☐	cellranger	Translate Spreadsheet Cell Ranges to Rows and Columns	1.1.0
☐	changepoint	Methods for Changepoint Detection	2.2.2
☐	chron	Chronological Objects which can Handle Dates and Times	2.3-53
☐	cli	Helpers for Developing Command Line Interfaces	1.1.0
☐	clipr	Read and Write from the System Clipboard	0.6.0
☐	clisymbols	Unicode Symbols at the R Prompt	1.2.0

To ensure that analyses are reproducible as far as possible, it is recommended that you create and save an R script (in RStudio: File -> New File -> R Script). This script will contain the lines of code that you wish to run. The script will open in the top left quadrant, as in the example below. The hash symbol can be used to comment out parts of the script that should not be run, for example, if you want to add a comment about what you are doing, or don't want to run a certain line of code.



```r
# Setup -----------------------------------------------------------------
# Load the nonstat package
library(nonstat)

# Set the path to the folder containing the 'Inputs' and 'Outputs' folders
folder <- "C:/Non-stationarity/Data"

# Trend testing ---------------------------------------------------------
MK.test(folder, sig.level = 5)
Pettitt.test(folder, sig.level = 5)
PELT.test(folder)

# Flood frequency analysis (time only as a covariate) --------------------

# Run the fitting function
dist <- "GEV"
RPs <- c(2, 10, 20, 50, 100)
CI <- 90  # set the confidence interval to be 90%
GEVfits <- fit.time(dist, folder, RPs, CI)

# Run the plotting functions
gauge <- "24004"
type <- "Varying location and scale"
fit.results <- GEVfits
RPs <- c(20, 50, 100)  # return periods to plot
CI.RP <- 100  # return period for confidence interval plot
plot.selected.distributions(folder, gauge, type, fit.results, RPs, CI.RP, CI)

RPs <- c(2, 10, 20, 50, 100) # return periods to plot
plot.all(folder, gauge, fit.results, RPs)

```

```
R version 3.5.3 (2019-03-11) -- "Great Truth"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

	Name	Description	Version
	User Library		
☐	acepack	ACE and AVAS for Selecting Multiple Regression Transformations	1.4.1
☐	akima	Interpolation of Irregularly and Regularly Spaced Data	0.6-2
☐	askpass	Safe Password Entry for R, Git, and SSH	1.1
☐	assertthat	Easy Pre and Post Assertions	0.2.1
☐	automap	Automatic interpolation package	1.0-14
☐	backports	Reimplementations of Functions Introduced Since R-3.0.0	1.1.4
☐	base64enc	Tools for base64 encoding	0.1-3
☐	bbplot	MAKING GGPLOT GRAPHICS IN BBC NEWS STYLE	0.2
☐	BH	Boost C++ Header Files	1.69.0-1
☐	bibtex	Bibtex Parser	0.4.2
☐	bit	A Class for Vectors of 1-Bit Booleans	1.1-14
☐	bit64	A S3 Class for Vectors of 64bit Integers	0.9-7
☐	bitops	Bitwise Operations	1.0-6
☐	boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-22
☐	brew	Templating Framework for Report Generation	1.0-6
☐	broom	Convert Statistical Analysis Objects into Tidy Tibbles	0.5.2
☐	callr	Call R from R	3.2.0
☐	cellranger	Translate Spreadsheet Cell Ranges to Rows and Columns	1.1.0
☐	changepoint	Methods for Changepoint Detection	2.2.2
☐	checkmate	Fast and Versatile Argument Checks	2.0.0
☐	chron	Chronological Objects which can Handle Dates and	2.3-53

Lines of code in the script can be run in the console by clicking on the line to run and pressing **'Ctrl + Enter'** or by clicking the **'Run'** button at the top of the script. You can save the script as a .R file using File -> Save As. This can then be re-opened if you need to access the script again.

It is generally recommended to start a new analysis in a clean version of RStudio, where nothing is saved in the workspace that may interfere with the analysis. If you get errors that you have not previously encountered from the same function, then it is worth restarting RStudio to check that nothing is saved from previous analyses that may be interfering with the new analysis.

Package installation

The ‘nonstat’ R package is supplied as a **.tar.gz** file.

The ‘nonstat’ package has a number of dependencies (other R packages on which the code depends). These are changepoint, data.table, dplyr, ggplot2, grid, gridExtra, lubridate, reshape2, stats, stringr, texmex and trend. These must be installed first. You can see which packages you have installed in the ‘Packages’ tab, in the bottom right panel (assuming the default RStudio layout). Packages only need installing once, so you do not need to re-install packages that you already have (unless they need updating). If you need to install any dependencies, this can be achieved using the following code (listing all missing packages, rather than the two in this example):

```r
install.packages(c('changepoint', 'trend'))
```

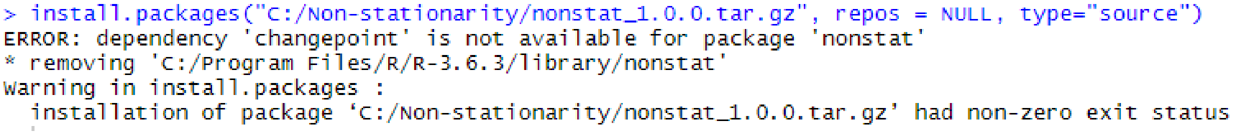
Save the **.tar.gz** file to a directory with an appropriate name. You can use the code below to store that directory and file name in R, replacing the file path with the one to the directory you just created. Note that file and folder paths in R use a forward slash rather than a backslash by default.

```r
path_to_file <- "C:/Non-stationarity/nonstat_1.0.0.tar.gz"
```

Now you can install the ‘nonstat’ package, using the following code. (It would also be possible to put the file path directly into the below line, rather than using ‘path_to_file’, if preferred.)

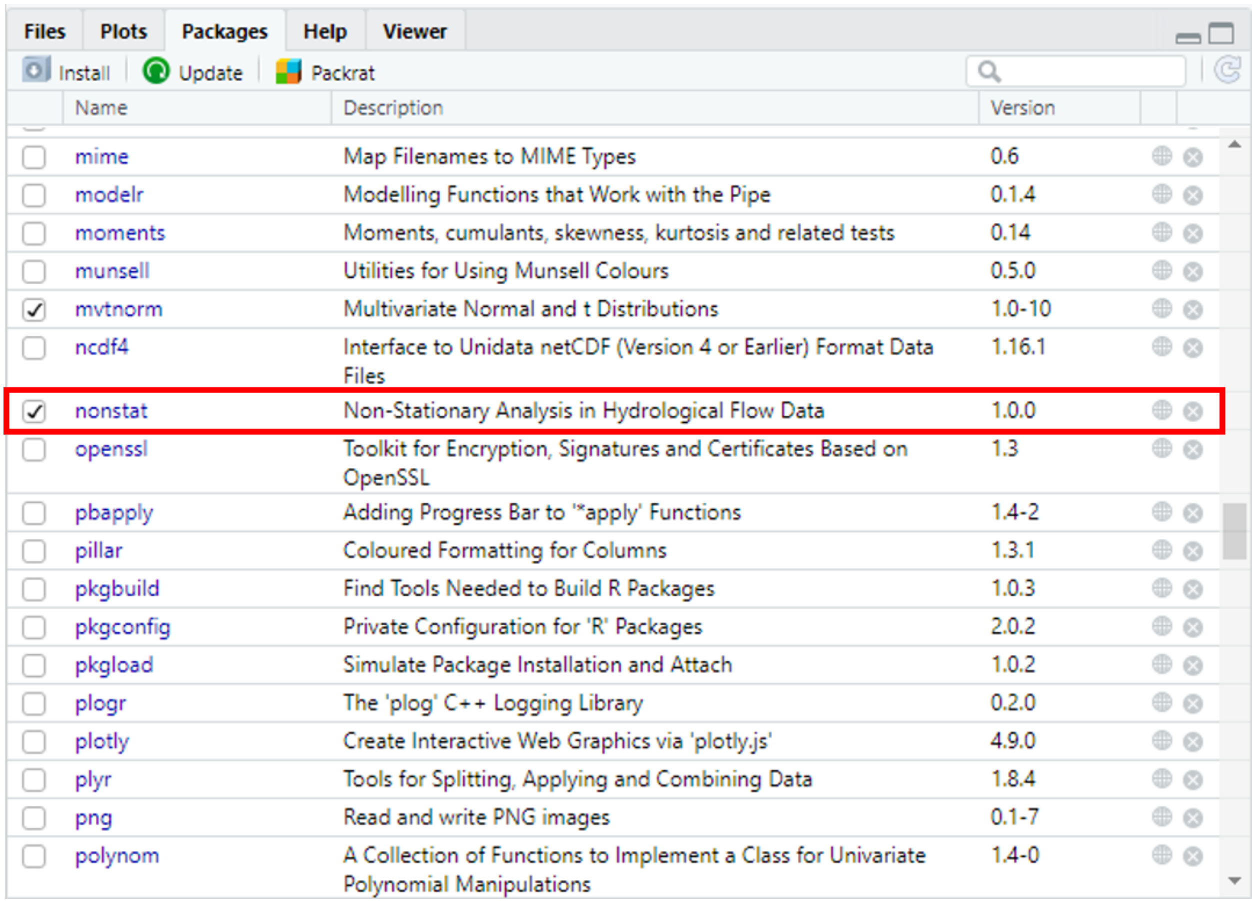
```r
install.packages(path_to_file, repos = NULL, type="source")
```

If any of the package dependencies are still missing, you will get an error message like the following. In this example, the only missing package is ‘changepoint’. This means that the nonstat package has failed to install and you will need to install any missing packages before re-installing the nonstat package.



```
> install.packages("C:/Non-stationarity/nonstat_1.0.0.tar.gz", repos = NULL, type="source")
ERROR: dependency 'changepoint' is not available for package 'nonstat'
* removing 'C:/Program Files/R/R-3.6.3/library/nonstat'
Warning in install.packages :
  installation of package ‘C:/Non-stationarity/nonstat_1.0.0.tar.gz’ had non-zero exit status
```

Once the package is installed, you should be able to see the package present in the ‘Packages’ tab, in the bottom right panel (assuming the default RStudio layout), as shown below:



	Name	Description	Version
☐	mime	Map Filenames to MIME Types	0.6
☐	modelr	Modelling Functions that Work with the Pipe	0.1.4
☐	moments	Moments, cumulants, skewness, kurtosis and related tests	0.14
☐	munsell	Utilities for Using Munsell Colours	0.5.0
☑	mvtnorm	Multivariate Normal and t Distributions	1.0-10
☐	ncdf4	Interface to Unidata netCDF (Version 4 or Earlier) Format Data Files	1.16.1
☑	nonstat	Non-Stationary Analysis in Hydrological Flow Data	1.0.0
☐	openssl	Toolkit for Encryption, Signatures and Certificates Based on OpenSSL	1.3
☐	pbapply	Adding Progress Bar to '*apply' Functions	1.4-2
☐	pillar	Coloured Formatting for Columns	1.3.1
☐	pkgbuild	Find Tools Needed to Build R Packages	1.0.3
☐	pkgconfig	Private Configuration for 'R' Packages	2.0.2
☐	pkgload	Simulate Package Installation and Attach	1.0.2
☐	plogr	The 'plog' C++ Logging Library	0.2.0
☐	plotly	Create Interactive Web Graphics via 'plotly.js'	4.9.0
☐	plyr	Tools for Splitting, Applying and Combining Data	1.8.4
☐	png	Read and write PNG images	0.1-7
☐	polynom	A Collection of Functions to Implement a Class for Univariate Polynomial Manipulations	1.4-0

Using the nonstat package

The functions in the nonstat package can be accessed by loading the package into the current R session by running the following code in the console (or by checking the box to the left of the package name in the 'Packages' list shown above). You will be able to see that other R packages on which nonstat depends are also loaded.

```
library(nonstat)
#> Loading required package: changepoint
#> Loading required package: zoo
#>
#> Attaching package: 'zoo'
#> The following objects are masked from 'package:base':
#>
#>     as.Date, as.Date.numeric
#> Successfully loaded changepoint package version 2.2.2
#>  NOTE: Predefined penalty values changed in version 2.2.  Previous penalty values with a postfix 1 i
#> Loading required package: data.table
#> Loading required package: dplyr
#>
#> Attaching package: 'dplyr'
#> The following objects are masked from 'package:data.table':
#>
```

```
#>     between, first, last
#> The following objects are masked from 'package:stats':
#> 
#>     filter, lag
#> The following objects are masked from 'package:base':
#> 
#>     intersect, setdiff, setequal, union
#> Loading required package: ggplot2
#> Loading required package: grid
#> Loading required package: gridExtra
#> 
#> Attaching package: 'gridExtra'
#> The following object is masked from 'package:dplyr':
#> 
#>     combine
#> Loading required package: lubridate
#> 
#> Attaching package: 'lubridate'
#> The following objects are masked from 'package:data.table':
#> 
#>     hour, isoweek, mday, minute, month, quarter, second, wday,
#>     week, yday, year
#> The following object is masked from 'package:base':
#> 
#>     date
#> Loading required package: reshape2
#> 
#> Attaching package: 'reshape2'
#> The following objects are masked from 'package:data.table':
#> 
#>     dcast, melt
#> Loading required package: stringr
#> Loading required package: texmex
#> Loading required package: mvtnorm
#> Loading required package: trend
```

The help files for each function can be accessed in R by typing '?*function*' where *function* is the function name, for example, '?fit.time'. Note that R is case-sensitive.

Data

Each function in the package is developed to work with flow data in .AM files, such as those found in the National River Flow Archive (NRFA) peak flows dataset. The dates must be in the standard format of 22 Jan 1960 for example; please ensure this is the case if you manually edit any files. Prior to running any functions, you need to create a directory containing two folders: 'Inputs' and 'Outputs'. To tell R what this directory is, you can use the following code:

```
folder <- "C:/Non-stationarity/Data"
```

Every function in the package has the path to this directory as an input argument. If you are using the same folder for the whole analysis then this can just be set once at the beginning of the script.

Data for any gauges you wish to analyse must be in the 'Inputs' folder (many of the functions will loop through all the gauge data (.AM files) saved in this folder). Outputs will automatically be written out to the 'Outputs' folder.

General comments

For outputs from all functions, the gauge numbers come from the station number within the .AM file and therefore that will need changing if multiple versions of the same gauge are used (if only the file name of the .AM file is changed, some of the results files will be overwritten). Outputs will be overwritten when the code is re-run if the gauge numbers are the same, however, output file names can be changed once the file has been produced if required. (Note that a .png file will be overwritten even if the file is open but a .csv file will not and a warning message will be produced, so it is important to make sure that .csv results files are closed before running code to update them.) The gauge name comes from a look-up file ('Gauge_metadata.csv') which is provided with the package and needs to be placed in the 'Inputs' folder. If the gauge ID (number) does not exist in the look-up file then no gauge name will be included in the outputs. The file can be edited to include your gauge if you wish to do so, as long as the format of the file remains the same. The existing IDs are those of the NRFA.

When running the scripts, it is worth keeping an eye out for errors or warnings that may appear in the R console. Errors generally stop the function so that it fails to complete its intended analysis, so it is unlikely that some or all results will have been written out. The problem will therefore need addressing. Some of the functions catch the errors, rather than stopping the analysis, and they are written out in .csv files instead. It is therefore important to check the warning and error files that are written out.
Warnings caution users without stopping the function working, so you will probably get results, but it is possible that there was a problem with the inputs or the computation, for example, something may have been amended slightly to allow the function to work. Some warnings do not require action to be taken, if you are happy that the analysis is not being impacted and the results are correct.
Each function in this package uses the same data pre-processing routine. If the data pre-processing produces warnings or errors for a given gauge, they will be written out in the automatically-generated 'Input data warnings and errors.csv' file. Errors are likely to be related to formatting of the input data, for example, an expected header being missing or one section of the file not ending with an '[END]' row.
A warning message may also be produced if one of the gauges has a missing stage value in the .AM file; this is not a problem as it is only the flow values that are analysed, but it is worth checking that this is the problem in the .AM file to which the warning refers. In the unlikely situation that a flow value is missing in the .AM file for a date (water year) that is included in the file, then that row (water year) is automatically removed from the analysis (to enable the distribution fitting in some of the functions to work).

All water years in the output files are defined as starting in the October of the year stated, e.g. the 2019 water year would be from October 2019 to September 2020.

The x and y axes of all plots are determined automatically based on the data; some are fixed such that multiple plots showing results for different return periods are easily comparable.

Optional parameters do not need to be included to run the function. Output folders and files will automatically be created for each function that is run.

For the code examples in this document, the lines below the code used to run the function starting with '#>' show what appears in the console when the function runs, as an example of what you might see in your console when you run the function.

Trend and change point tests

There are three functions in the package that can carry out trend and change point tests and these are described in this section. All these functions can be run on multiple gauges and a list of the gauges will be printed in the R console. A progress bar will update after each gauge has been analysed.

MK.test

This function carries out the Mann-Kendall trend test on all gauges listed in a folder and uses functionality from the ‘trend’ R package. The Mann-Kendall test is a non-parametric trend test, widely used for monotonic trend testing. The results are automatically written out to the ‘Outputs/Mann-Kendall’ folder. The ‘Mann-Kendall’ folder will be automatically generated if it does not already exist. The function takes the following inputs:

- folder: A path to a folder containing two sub-folders. One must be called ‘Inputs’ and one must be called ‘Outputs’. The ‘Inputs’ folder must contain .AM files for each gauge for which analysis is required. The ‘Outputs’ folder is where the ‘Mann-Kendall’ sub-folder will be created, which is the location to which results files and plots will automatically be written.
- sig.level: The significance level as a percentage for interpretation of results in the output .csv file (e.g. input ‘5’ to get an interpretation of whether trend is significant at the 5% significance level). 5 is the default setting for this parameter.

The function can be run as follows. The progress bar can be seen to update after each gauge has been analysed and the total processing time is also stated. If there are any warnings from running the function, they will also be stated in the console.

```
folder <- "C:/Non-stationarity/Data"
MK.test(folder, sig.level = 5)
#>
  |                                                                 
  |                                                                 |   0%[1] "2 gauging stations queue
#> C:/Non-stationarity/Data/Inputs/24004.AM
#> C:/Non-stationarity/Data/Inputs/76005.AM
#>
#> [1] "C:/Non-stationarity/Data/Inputs/24004.AM"
#>
  |                                                                 
  |================================                                 |  50%[1] "C:/Non-stationarity/Data
#>
  |                                                                 
  |=================================================================| 100%
#> [1] "*** Total processing time:"
#> Time difference of 0.3097801 secs
```

The outputs from running the function are as follows:

‘Mann-Kendall results.csv’: A .csv file containing gauge information and results. Missing years are those in the middle of the AMAX series, not at the start or end, as there is no way of automatically determining that these are missing. The number of years reported is the number of years used in the analysis, excluding rejected years (i.e. those marked as rejected in the .AM file) and rows with missing flow data (if they exist).

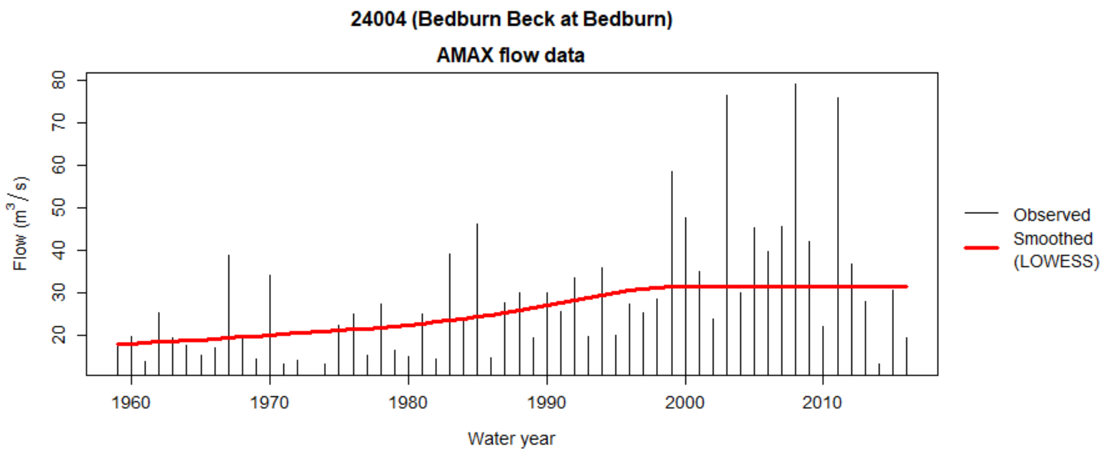
An explanation of the results from the Mann-Kendall test can be found in https://cran.r-project.org/web/packages/trend/vignettes/trend.pdf. The Mann-Kendall Z score (MKZ) is a standardised version of the results, so can be used to compare stations.

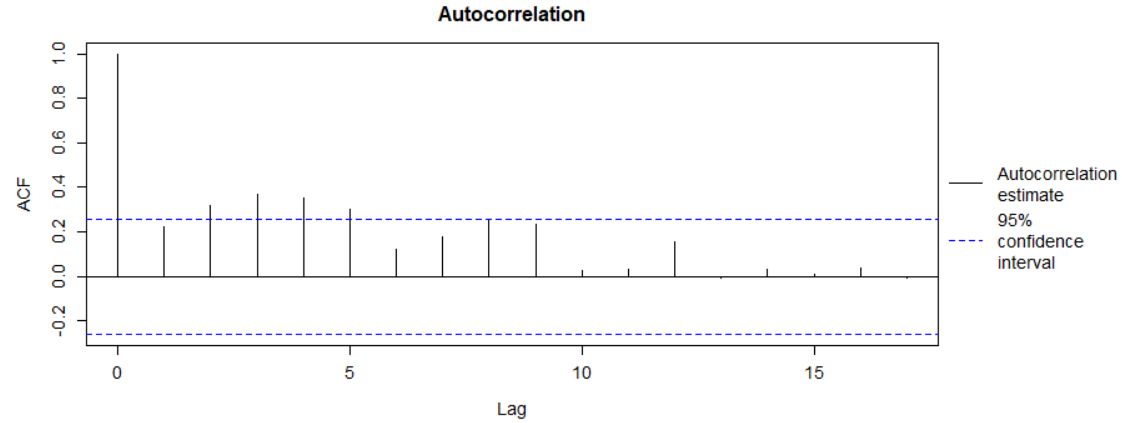
Positive values of MKZ indicate increasing trends and negative ones indicate decreasing trends. The 2-sided p-value can be used to determine whether the results are statistically significant at a given significance level (as specified by the user) and the interpretation is reported in the next column. At a 5% significance level, this means that if the absolute value of MKZ is greater than 1.96 (or if the p-value is less than 0.05) then the null hypothesis (H0) of no trend is rejected. The conventional approach is then to (provisionally) accept a single alternative hypothesis H1, i.e. that there is a trend. For other selected significance levels, the absolute values of MKZ must be greater than the following critical values to reject H0:

- 10% significance level: 1.645
- 2% significance level: 2.33
- 1% significance level: 2.58
- 0.1% significance level: 3.29.

‘*gauge_number* (*gauge name – if exists*) trend plots.png’: One plot per gauge containing two graphs. The first shows a time series of the AMAX data (black lines) and smoothed data (red line), using locally-weighted polynomial regression (LOWESS). The second shows the autocorrelation function. Autocorrelation plots can be used for checking randomness in data. This is determined by computing autocorrelations for data values at a range of time lags. Further information can be found at https://www.itl.nist.gov/div898/handbook/eda/section3/autocopl.htm.
This therefore enables investigation into whether there is significant lag-1 serial correlation in the AMAX time series (i.e. whether the time series is dependent on its past), because the Mann-Kendall test requires serial independence of data. For example, if positive autocorrelation is present, it can increase the probability of detecting trends that do not exist. Therefore, other methods may be more suitable for assessing the significance of trend in the data, such as a modified version of the Mann-Kendall test (not available in the nonstat package).

The plots look like this, where in the second plot, the x-axis is the lag for each autocorrelation estimate and the y-axis is the estimate:





The autocorrelation has been estimated at a range of lags (water years, given that AMAX data are being used). Be aware that missing years are not taken into account (i.e. the function assumes that the values are for consecutive years), so the estimates may not be entirely correct, particularly if there are lots of missing years throughout the time series. The lag-1 autocorrelation indicates the correlation between values in the time series and their preceding value and lag-2 indicates the correlation between values and the value two water years before. The Bedburn Beck example above shows autocorrelations for lags up to 17 years. The ACF at lag-0 is always 1. High values of autocorrelation indicate strong correlation between values at given lags. For time series where the current value is closely related to its preceding value, high values of autocorrelation would be expected for the shorter lags. For time series showing a periodic pattern, e.g. where the current value is closely related to the observation two before it, there would a high value of autocorrelation for lag-2. For time series with no clear patterns, the values of autocorrelation should be relatively small. The blue, horizontal dashed lines on the plot represent lag-wise 95% confidence intervals centred around zero. Autocorrelation estimates at a given lag would be statistically significant at a 5% significance level if they are outside these lines (compared to a null hypothesis of no autocorrelation).

Pettitt.test

This function carries out Pettitt's test, which is designed to detect a sudden change in the mean of a time series. It is a non-parametric test, i.e. it makes no assumption about the distribution followed by the data. Results are automatically written out to the 'Outputs/Pettitt' folder. The 'Pettitt' folder will be automatically generated if it does not already exist. The function takes the following inputs:

- folder: A path to a folder containing two sub-folders. One must be called 'Inputs' and one must be called ‘Outputs’. The ‘Inputs’ folder must contain .AM files for each gauge for which analysis is required. The ‘Outputs’ folder is where the ‘Pettitt’ sub-folder will be created, which is the location to which results files and plots will automatically be written.

- sig.level: The significance level as a percentage for interpretation of results in the output .csv file (e.g. input ‘5’ to get an interpretation of whether the change point is significant at the 5% significance level). 5 is the default setting for this parameter.

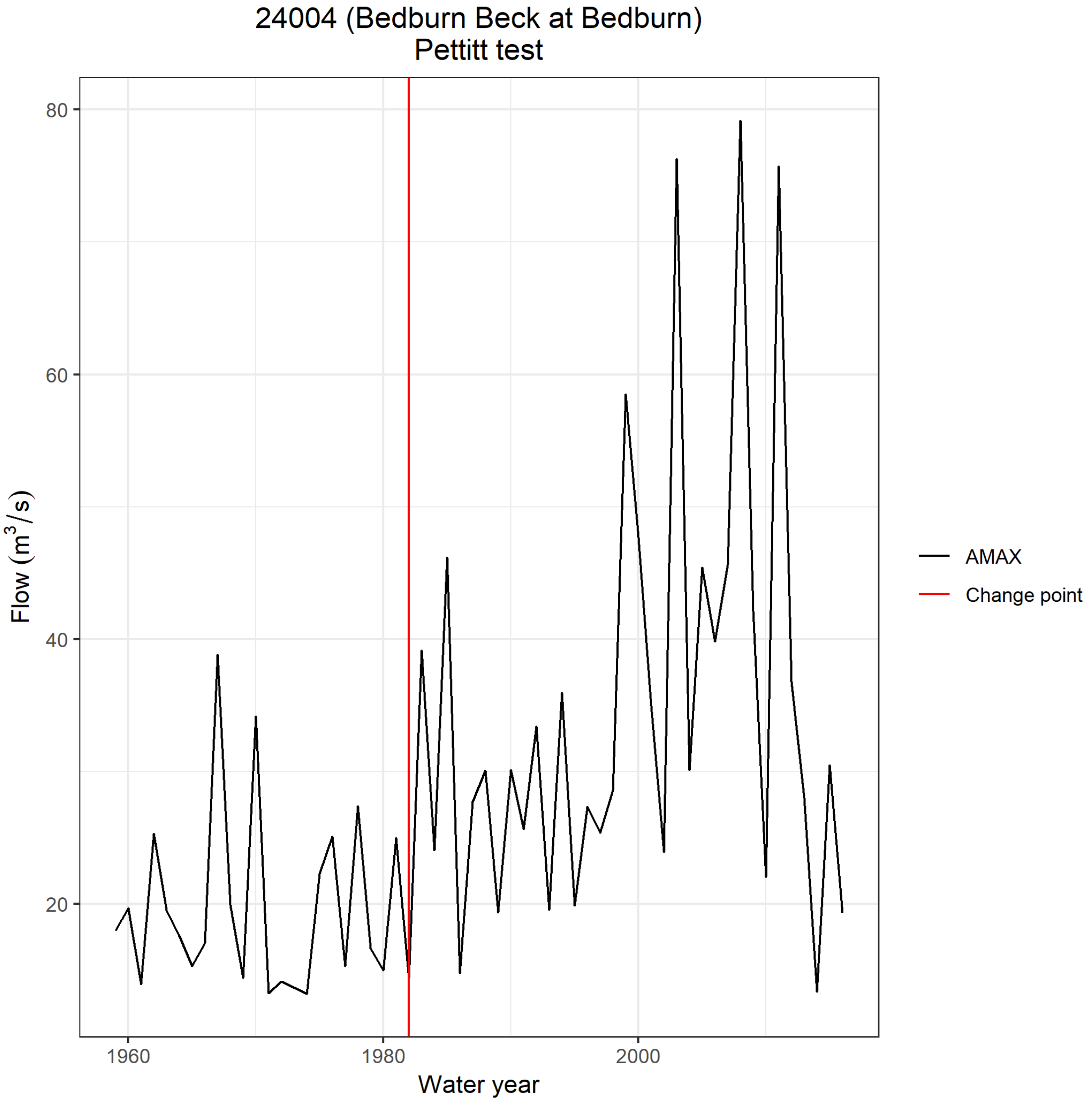
The function can be run as follows. The progress bar can be seen to update after each gauge has been analysed, along with a statement that an image is being saved, and the total processing time is also stated. Any warnings will also be stated in the console.

```
folder <- "C:/Non-stationarity/Data"
Pettitt.test(folder, sig.level = 5)
#>
  |
  |                                                                 |   0%[1] "2 gauging stations queue
#> C:/Non-stationarity/Data/Inputs/24004.AM
#> C:/Non-stationarity/Data/Inputs/76005.AM
#>
#> [1] "C:/Non-stationarity/Data/Inputs/24004.AM"
#> Saving 6.5 x 4.5 in image
#>
  |
  |================================                                 |  50%[1] "C:/Non-stationarity/Data
#> Saving 6.5 x 4.5 in image
#>
  |
  |=================================================================| 100%
#> [1] "*** Total processing time:"
#> Time difference of 0.8955989 secs
```

The outputs from running the function are as follows:

‘Pettitt results.csv’: A .csv file containing the results for all gauges. The Pettitt test always detects a change point and the year of the detected change point is given. The p-value can be used to identify whether the change in the mean AMAX flow from the period before to the period after the change point is significant. In this case, the null hypothesis, H0, is that there is no difference between the means of the earlier and later portions of each AMAX flow series. If the p-value is less than the user-specified significance level (e.g. 5% means that the p-value should be compared with 0.05), then H0 is rejected. The conventional approach is then to (provisionally) accept a single alternative hypothesis H1, i.e. that there is a sudden change. The interpretation is provided in the file. The absolute and percentage change in the mean between the periods before and after the change point is provided, as well as the direction of the change (positive or negative).

‘*gauge_number* (*gauge name – if exists*) Pettitt results.png’: A plot is also generated for each gauge showing the AMAX data with the detected change point shown as a vertical line, if it is significant at the specified significance level. The plot looks like this:



PELT.test

One limitation of Pettitt's test is that it can only detect a single change point. Additionally, it has been criticised for its tendency to classify gradual trends as sudden changes (Rougé et al., 2013). An alternative is the PELT (Pruned Exact Linear Time) test. As with most change point algorithms, PELT (Killick et al., 2012) tries to find the optimal segmentation in a time series. The method prevents identification of too many unrealistic change points. PELT is implemented through the 'changepoint' R package (Killick and Eckley, 2014) and allows detection of one or more change(s) in mean and variance of the AMAX flow data (this function is restricted to two change points; results from national analysis found no gauges where more than two change points were detected). The approach requires an assumption that the data follow a known distributional form. Since the flood time series are not generated from any known distribution, a log transformation is applied to transform the data to approximate Normality (Box and Cox, 1964). In effect, we assume that the AMAX flows follow a log-Normal distribution, which has been used in previous studies (e.g. Prosdocimi et al., 2014). It is worth checking that the log-Normal assumption is reasonable; this can be done by checking the Normal Q-Q plot which is output from the package. Points close to the diagonal line indicate a reasonable log-Normal fit.

A minimum segment length is required, which prevents additional over-fitting and false positive changes

at short time scales, so 10 years has been chosen here as a suitable length for local stationarity pre- and post-change. This means that the record length of the input data must be at least 20 years (two times the minimum segment length). If the record length of your data is shorter than this, the function will not work and the gauge will be listed in a spreadsheet in the ‘Outputs’ folder: ‘Gauges with record lengths too short to be analysed.csv’.
The PELT.test function carries out the PELT test and takes the following input:

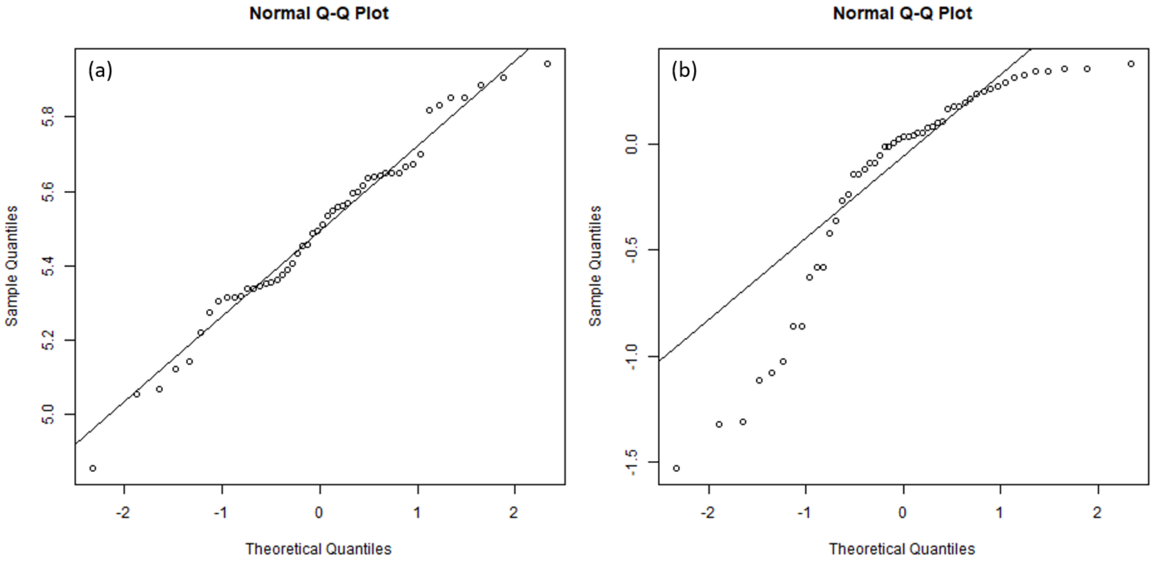
- folder: A path to a folder containing two sub-folders. One must be called ‘Inputs’ and one must be called ‘Outputs’. The ‘Inputs’ folder must contain .AM files for each gauge for which analysis is required. The ‘Outputs’ folder is where the ‘PELT’ sub-folder will be created, which is the location to which results files and plots will automatically be written.

The function can be run as follows. The progress bar can be seen to update after each gauge has been analysed, along with a statement that an image is being saved, and the total processing time is also stated. Any warnings will also be stated in the console.

```
folder <- "C:/Non-stationarity/Data"
PELT.test(folder)
#>
  |
  |                                                                 |   0%[1] "2 gauging stations queue
#> C:/Non-stationarity/Data/Inputs/24004.AM
#> C:/Non-stationarity/Data/Inputs/76005.AM
#>
#> [1] "C:/Non-stationarity/Data/Inputs/24004.AM"
#> Saving 6.5 x 4.5 in image
#>
  |
  |================================                                 |  50%[1] "C:/Non-stationarity/Data
#> Saving 6.5 x 4.5 in image
#>
  |
  |=================================================================| 100%
#> [1] "*** Total processing time:"
#> Time difference of 0.9033248 secs
```

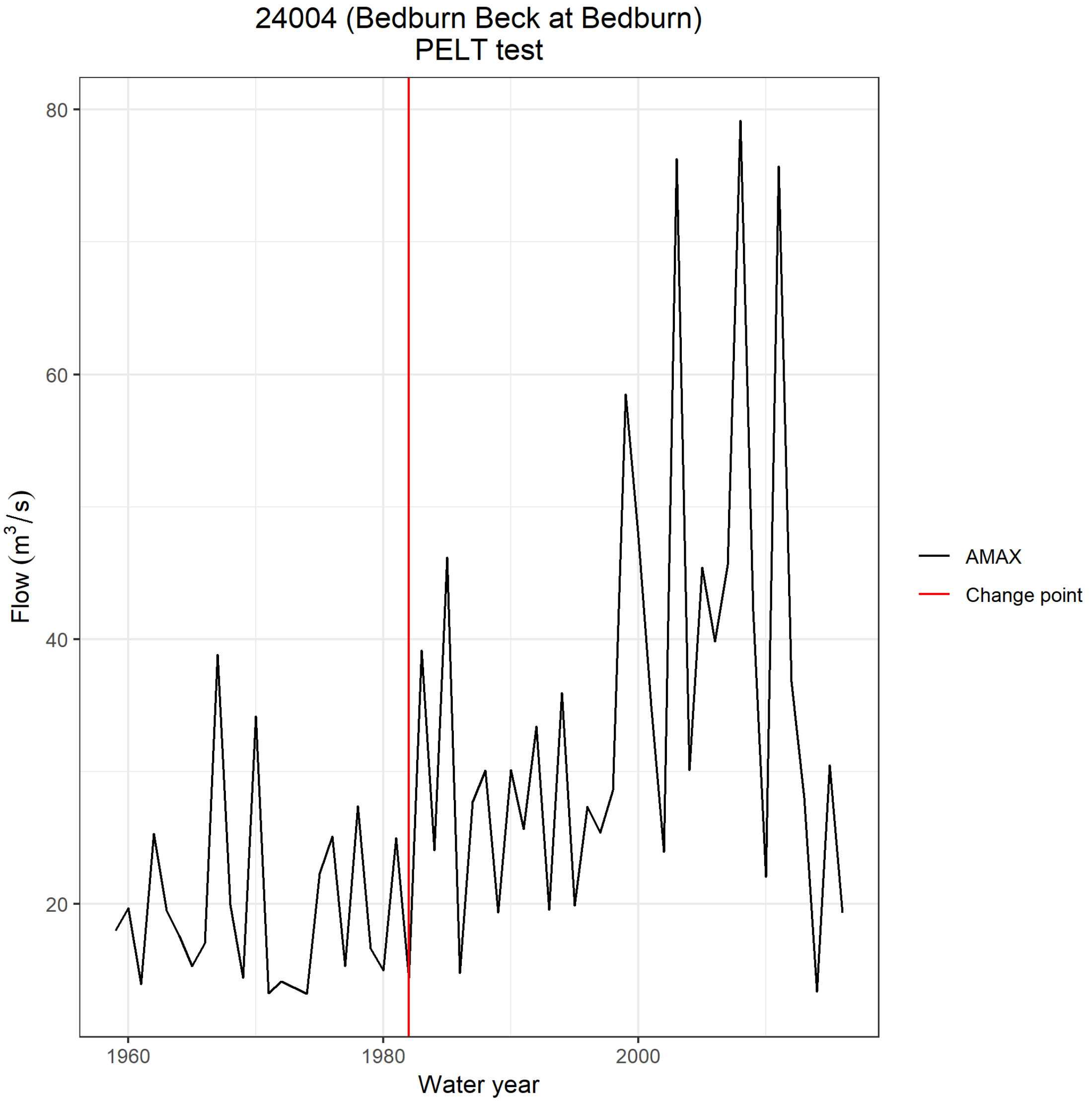
The outputs from running the function are as follows:

‘*gauge_number* (*gauge name – if exists*) Normal Q-Q plot.png’: A Normal Q-Q plot of the log-transformed data. This indicates whether the log-Normal assumption is reasonable; points close to the diagonal line should indicate a reasonable log-Normal fit. If the log-transformed data do not remotely follow a Normal distribution then the PELT results may not be reliable. The following figure shows two Normal Q-Q plots. The points are close to the line in (a) and hence the log-Normal assumption seems to be reasonable. In (b), however, the points are further from the line and it is possible that the log-Normal assumption is not valid and the PELT results may not be reliable.



'PELT results.csv': A .csv file containing the results for all gauges. The year(s) of the detected change point(s) are given (some gauges will have no detected change points). For each detected change point, the absolute and percentage change in the mean and standard deviation between the periods before and after the change point is provided, as well as the direction of the change (positive or negative).

'*gauge_number* (*gauge name – if exists*) PELT results.png': A plot is also generated for each gauge showing the AMAX data with any detected change point shown as a vertical line. The plot looks like this:



Fitting non-stationary models

Traditional statistical methods for flood frequency analysis assume that each AMAX flow in a gauge record represents an independent observation from the same probability distribution as all the other flow values (i.e. observations of past flood events are assumed to represent the behaviour of future events). If this probability distribution is not constant over time (non-stationary), the peak flows are not identically distributed and so the assumption no longer holds. Therefore, if non-stationarity is evident in our AMAX flow data, then design estimates derived from traditional flood frequency analysis may be overestimated or underestimated.

To model non-stationarity, covariates can be incorporated into the parameters of statistical models used in flood frequency analysis. Time (i.e. the year in which a flood happened) may seem to be an obvious choice for a covariate, relating changes in flood frequency to time. However, time is a proxy for some changing physical process that is causing change in flood frequency. An alternative option would be to use physically-based covariates, such as rainfall or a climate index such as the North Atlantic Oscillation. Two reasons are sometimes given for this. The first is that physical covariates help remove some of the year-to-year variability in AMAX flows, enabling better identification of time-based trends and better fit of statistical distributions (Prosdocimi et al. 2014). The second is that they provide a more physically meaningful model of non-stationarity. Some physical covariates may open up the prospect of predicting the future evolution of the flood frequency curve (Sraj et al. 2016). They may not directly represent the physical processes that cause floods, but represent a potential step forward from the simplistic approach of modelling non-stationarity as a change over time.

This package has an option for using only time as a covariate in a statistical distribution, as well as an option for using physical covariates. The former is a simpler option, and can be interpreted as a trend test. Either the Generalised Extreme Value (GEV) or Generalised Logistic (GLO) distributions can be used. The example code in this guidance uses the GEV distribution but the GLO could be used by changing the 'dist' parameter in relevant functions. All model fitting in this package is carried out using functions from the 'texmex' package and all confidence intervals are calculated using bootstrapping. Given that bootstrapping involves an element of random number generation when sampling flows from the model (and when resampling covariates, where applicable), the seed has been set so that it effectively re-uses the same random numbers each time the code is run so that results will be reproducible, even though there is an element of randomness. The fitting function is set to use two cores when bootstrapping.

Typically, flood risk estimates are defined using return periods, but in non-stationary models, this becomes more complicated. This package produces a range of outputs which allow non-stationary models to be used, considering the different ways of defining risk.

Some key definitions for this section are:

Return level - the value (in this analysis, the river flow) associated with a particular return period or probability. Hydrologists often refer to this as the design flood, or as a point on the flood frequency curve. This is used in the practitioner guidance in preference to return level.

Conditional return level – the return level that would be expected if the covariates had a particular combination of values (i.e. conditional on those values). For instance, if the covariate was the water year, there could be a return level conditional on the water year being 2019-20. This is referred to as a **conditional flow estimate** in the practitioner guidance.

This is therefore suitable for use given a model with time covariates, as one could use the conditional return level given the current year.

It is less useful to think the same way when using physical covariates, as for example, the 100-year return level given the 2019 rainfall amount is the expected return level under the (clearly hypothetical) conditions that the rainfall always takes the value that is observed in 2019. In this case, an alternative approach would be to define a marginal return level, as used in Eastoe and Tawn (2009).

Marginal return level – the return level corresponding to the averaged encounter probability, where averaging is over covariates in a period of interest. Specifically it is the return level where the average encounter probability is equal to the reciprocal of the return period. (An encounter probability is the probability of an event occurring in a given number of years.) The practitioner guidance refers to this as an **integrated flow estimate**.

Present-day marginal return level – the present-day expected return level for a particular exceedance probability, but not conditional on any particular value of a covariate such as annual rainfall. It is estimated by calculating the marginal return level, setting the time covariate to be the 'current' water year (last year in the flow and covariate data), and averaging over the full observed distribution of the physical covariates. This is a special case of the **single-year integrated flow estimate** described in the practitioner guidance.

Fitting models where time is the only covariate

fit.time

The fit.time function fits stationary and non-stationary distributions (GEV or GLO) to AMAX flow data for all gauges provided by the user in an 'Inputs' folder. The three non-stationary models have a varying location

parameter, a varying scale parameter and varying both location and scale parameters. The parameters vary with a covariate. This function uses time as a covariate and automatically calculates a value for the covariate to associate with each AMAX flow in the input data for the non-stationary distribution fitting, i.e. the user does not need to supply the time covariate. The time covariate is an index of water year and the values are centred and scaled.
The results can be interpreted as a parametric trend test; there may be some catchments where this indicates non-stationarity, but where no trend is detected by non-parametric trend tests, such as the Mann-Kendall test.
Parameters of the fitted distributions and the results from frequency analysis are automatically written out to .csv files in the 'Outputs/Time covariate models' folder. The 'Time covariate models' sub-folder will be automatically generated if it does not exist.
The function takes the following inputs, as documented in the package help files:

- dist: The statistical distribution to use in the fitting ("GEV" or "GLO").
- folder: A path to a folder containing two sub-folders. One must be called 'Inputs' and one must be called 'Outputs'. The 'Inputs' folder must contain .AM files for each gauge for which analysis is required. The 'Outputs' folder is where the 'Time covariate models' sub-folder will be created, which is the location to which results files and plots will automatically be written.
- RPs: The return periods for which frequency analysis is required.
- CI: The required confidence interval for the results as a percentage, e.g. '90' for 90% is the default.

The function fits a GEV or GLO distribution using maximum likelihood estimation (MLE) for a stationary case and three non-stationary cases (varying location, varying scale and varying both location and scale parameters). The scale parameter is log-transformed to ensure positivity for all possible covariate values.

The outputs from running the function are as follows.

'*gauge_number* (*gauge name – if exists*) GEV|GLO results.csv': A results spreadsheet containing the conditional return levels (flow estimates) from the fitted models. The first columns show the AMAX flow data used in the analysis. The 'WaterYearIndex' column is the automatically generated data representing time and the 'ScaledWaterYearIndex' column is the centred and scaled version of this which is used as a covariate to represent time in the model fitting. For each model, the 'Fit' column is the type of fit and the subsequent columns are the results from that fitted model (including the user-specified confidence intervals). The results from fitting a stationary distribution are the same for every year, but for the non-stationary cases, the results vary with time (e.g. the 100-year flow in 1960 may be different from the 100-year flow in 2015), because they are conditional on a given year. Note that the results estimates here are taken from the median of the bootstrap sample and the confidence intervals are taken from the relevant quantiles of the bootstrap sample.

'GEV|GLO distribution parameters and fits.csv': A spreadsheet containing the parameters from the distribution fitting is also produced. There is one row per model, so four rows per gauging station.

The first columns provide information about each station. Missing years are those in the middle of the AMAX series, not at the start or end, as there is no way of automatically determining that these are missing. The number of years reported is the number of years used in the analysis, excluding rejected years and rows with missing flow data (if they exist). The 'Fit' column gives the model type that corresponds to the parameters in that row. The next columns are the parameters of the fitted distribution. For a stationary fit, the location, ln(scale) and shape parameters are provided, along with their standard errors. The scale parameter (σ) can be obtained by taking the exponential of the value provided (which is ln(σ) or ϕ). For the non-stationary fits, the outputs change for the varying parameter(s). If the location parameter is varying then the location parameter (μ) is replaced with μ_0 and μ_1 (mu0 and mu1 in the column names) and if the scale parameter is varying then the ln(scale) parameter (ϕ) is replaced with ϕ_0 and ϕ_1 (phi0 and phi1 in the column names), as described in the following equations where the parameters vary with time, t:
$\mu(t) = \mu_0 + \mu_1 t,$
$ln(\sigma(t)) = \phi(t) = \phi_0 + \phi_1 t.$
Therefore, if the parameter is varying, then the column names should be read as mu0 and phi0, but if the parameter is not varying, they should be read as location and ln(scale).

The AIC and BIC are the Akaike Information Criterion and the Bayesian Information Criterion and represent the goodness of fit of the model. They are both methods of selecting the best fitting model for each station and establish a trade-off between the goodness of fit and the simplicity of the model. The lowest value indicates the best fit. The BIC tends to give preference to simpler models. It is worth noting that there can be small differences in the AIC or BIC value between different models for the same catchment and yet the models themselves can produce quite different results. It is therefore recommended that the results are assessed alongside diagnostic plots and other methods of judging goodness of fit to determine which model is most suitable.

Another approach to find best fitting models is to use likelihood ratio testing. This is a preferred approach when comparing a small number of candidate models, when the likelihood ratio can be calculated for each nested pair of models. It is impractical when comparing hundreds of models, which can be the case when several covariates are being considered, but it is suitable for this case where there are only four models to compare. The last three columns therefore identify the best fit per gauge based on the AIC value, the BIC value and likelihood ratio testing.

'*gauge_number* (*gauge name – if exists*) GEV|GLO diagnostic plots – *fit type*.png': This contains three plots. The first is of the observed AMAX data, the second is a probability-probability (or P-P) plot and the third is a quantile-quantile (or Q-Q) plot. Residual P-P plots and residual Q-Q plots are specific instances of P-P and Q-Q plots. They are used because there are covariates in the models, so for a given dataset (flow record) (with accompanying covariates), there is no such thing as "the model quantiles" since each set of covariates defines a different fitted GLO or GEV distribution. Therefore, a P-P or Q-Q plot cannot be constructed for the original data, as they don't come from a common distribution. Instead, the data points are first transformed to a common distribution by constructing model residuals using the fitted model parameters for each data point. These residuals do come from a common distribution and therefore meaningful P-P and Q-Q plots can be constructed. A well fitting model will have points lying close to the unit diagonal on both the P-P and Q-Q plots.
The plot title contains information about the type of fit, i.e. stationary or non-stationary. Information about covariates included in the model is in the form of the regression equation of covariates used in the location and scale parameters. For this fit.time function:

- If the model is stationary, then no covariates are included, so the equation is just "~1" for both the location and scale parameters because they are constant.
- If the model is non-stationary with only the location parameter varying with time (i.e. ScaledWaterYearIndex, the covariate that is automatically-generated within the package to represent time), then the equation for the location parameter is "~1 + ScaledWaterYearIndex". The scale parameter stays as "~1" as this is not varying with a covariate.
- If the model is non-stationary with only the scale parameter varying with time (i.e. ScaledWaterYearIndex), then the equation for the scale parameter is "~1 + ScaledWaterYearIndex". The location parameter stays as "~1" as this is not varying with a covariate.
- If the model is non-stationary with both the location and scale parameters varying with time (i.e. ScaledWaterYearIndex), then the equation for both is "~1 + ScaledWaterYearIndex".

The Q-Q plots for all four models are also provided in '*gauge_number* (*gauge name – if exists*) GEV|GLO diagnostic plots – Q-Q plots for all models.png' for easy comparison.

There are also several files containing information about warnings and errors produced during the analysis. These are as follows:

'GEV|GLO diagnostic plot warnings and errors.csv': If there were any warnings or errors produced during the generation of the diagnostic plots then they will be stated in here. There is one column for warnings and one for errors for each type of model. It will be empty if there were no warnings or errors.

'GEV|GLO fit time function warnings and errors.csv': If there were any warnings or errors produced within the function used to fit the distributions and calculate return levels, they will be stated in here.

'GEV|GLO model fitting warnings and errors.csv': If there were any warnings or errors produced during the actual model fitting, they will be stated in here. There is one column for warnings and one for errors for each type of model.

If there are any errors, then any results from the process which caused the error will be missing.

It is possible that the message 'Ratio of bias to standard error is high' will appear in the RStudio console window during the model fitting. This is an indication of the variability of the bootstrapped estimates and can be ignored in this instance.

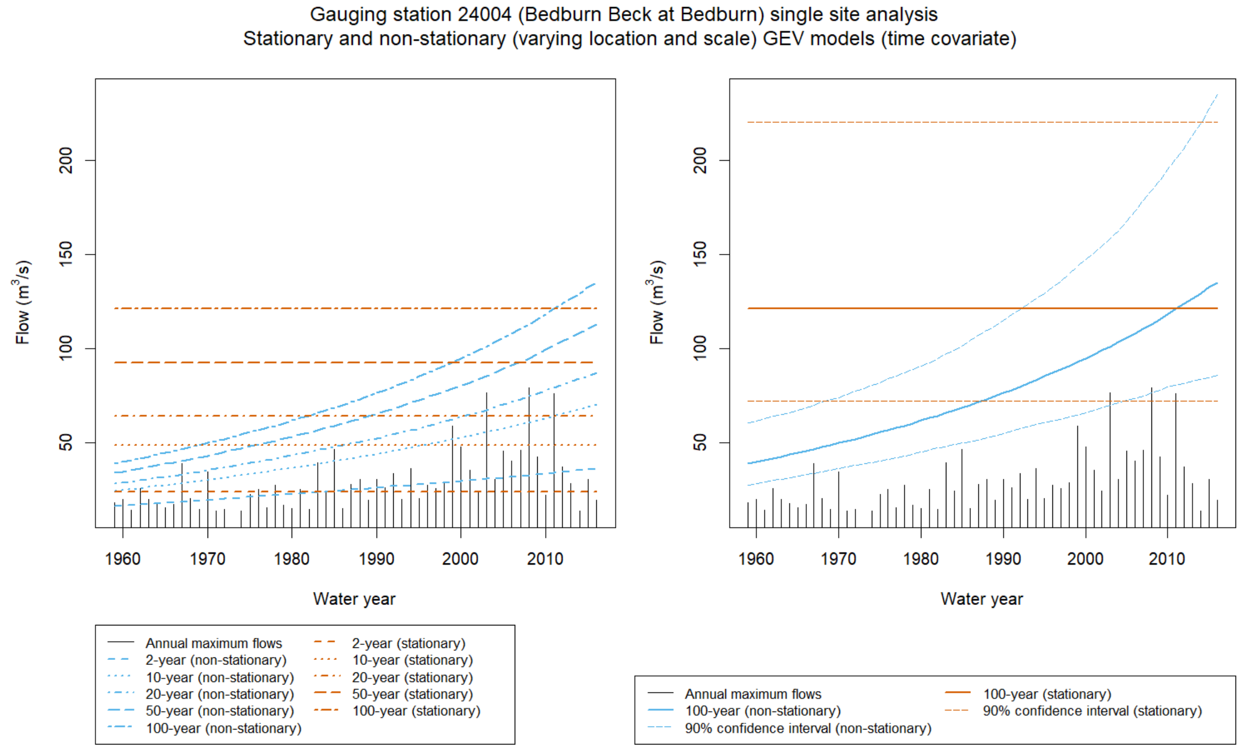
Once this function has been run, there are two functions that can be used for plotting the results. These are 'plot.selected.distributions' and 'plot.all'. The example code to run the 'fit.time' function is provided with the example plotting code in the 'plot.all' section.

plot.selected.distributions

This function plots the results from the fitted stationary distribution and one of the fitted non-stationary distributions selected by the user (e.g. the best fitting non-stationary model based on likelihood ratio testing) for one user-selected gauge using the outputs from fit.time for a range of return periods. It also plots the confidence interval for one return period on a second plot. These plots will be automatically written out to a .png file in the 'Outputs/Time covariate models' folder. The function takes the following inputs, as documented in the package help files:

- folder: A path to a folder containing two sub-folders. One must be called 'Inputs' and one must be called 'Outputs'. The 'Outputs' folder is where the 'Time covariate models' sub-folder will have been created, which is the location to which the plots will automatically be written.
- gauge: The number of the gauge to plot; this must be one of the .AM files to whose data a model was fitted using the fit.time function.
- type: The type of non-stationary distribution to plot. Options are 'Varying location', 'Varying scale' and 'Varying location and scale'.
- fitResults: The output from the fitting function (fit.time). This could either be from a saved location or will already be in the R environment if fit.time has just been run and saved to a named object.
- RPs: The return periods to plot (the maximum number of return periods allowed is five).
- CI.RP: The return period for which a confidence interval is to be plotted. Must be in RPs.
- CI: The required confidence interval as a percentage to plot for the results from the distribution fitting, e.g. '90' for 90% is the default.
- plotTitle: This is an optional parameter. The code will automatically generate a plot title, but this can be used if you wish to change the title of the output plot. To split a title over two lines use a backslash followed immediately by 'n' in the character string where you want the line break to be.

The output from the function is a .png file containing two plots ('*gauge_number* (*gauge name – if exists*) GEV|GLO user-specified distribution results and confidence interval plots.png'). The plots should look like this:



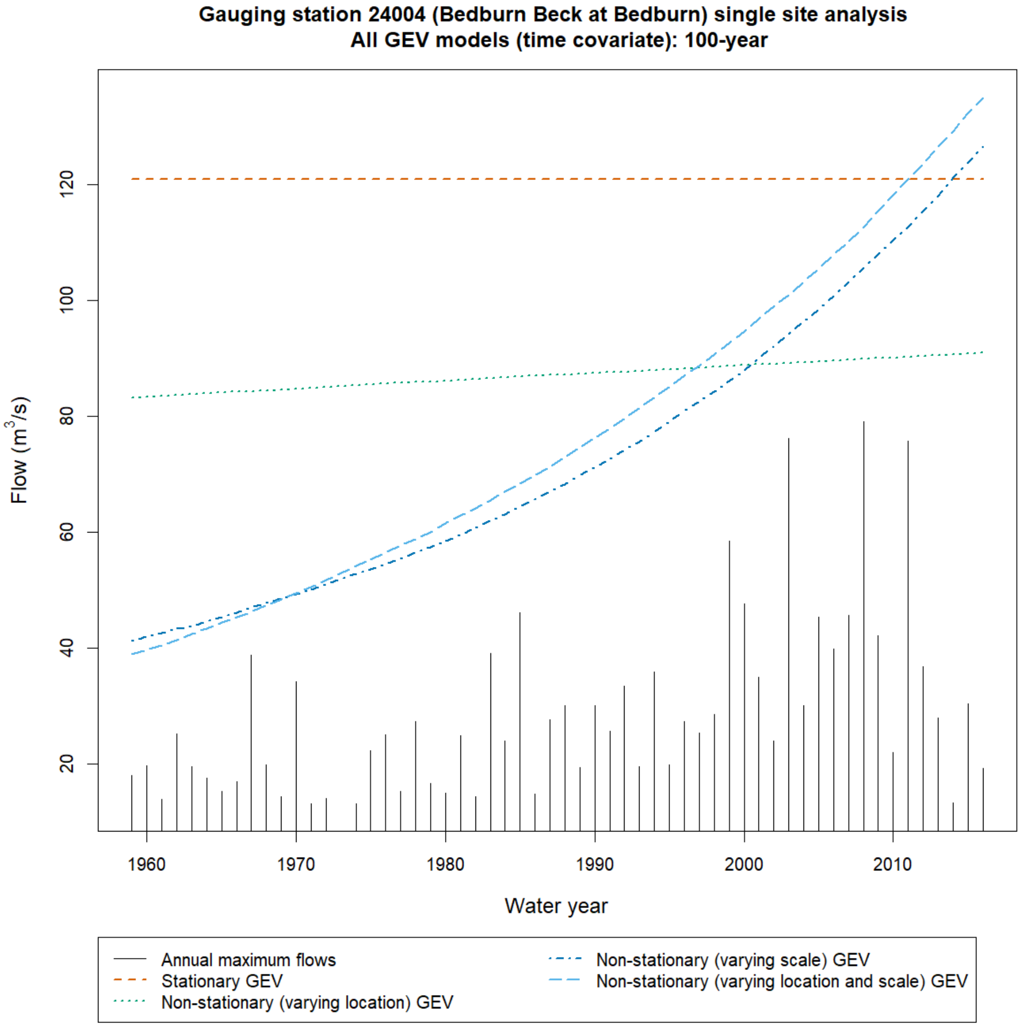
The first plot shows the return levels from the fitted stationary model and the conditional return levels from one user-selected non-stationary model for specified return periods. In the example above, the varying location and scale parameters model results have been plotted, as stated in the plot title. The second plot shows confidence intervals for the return levels for one specified return period (in this example, for the 100-year return period). It can be seen that the non-stationary results are varying over time, with the 100-year flow increasing from 39 to 135 m^3/s over the period of record.

plot.all

This function plots the results from the fitted stationary and three non-stationary models (varying location, varying scale and varying location and scale parameters) with time as a covariate for a user-specified gauge. There will be one plot per specified return period. The plots will be automatically written out to a .png file in the 'Outputs/Time covariate models' folder. The function takes the following inputs, as documented in the package help files:

- folder: A path to a folder containing two sub-folders. One must be called 'Inputs' and one must be called 'Outputs'. The 'Outputs' folder is where the 'Time covariate models' sub-folder will have been created, which is the location to which results files and plots will automatically be written.
- gauge: The number of the gauge to plot; this must be one of the .AM files to whose data a model was fitted using the fit.time function.
- fitResults: The output from the fitting function (fit.time). This could either be from a saved location or will already be in the R environment if fit.time has just been run and saved to a named object.
- RPs: The return periods to plot.
- plotTitle: This is an optional parameter. The code will automatically generate a plot title, but this can be used if you wish to change the title of the output plot. To split a title over two lines use a backslash followed immediately by 'n' in the character string where you want the line break to be. This should only be used when one return period has been specified, if you wish to manually include the return period value in the title. The automatically-generated title contains the appropriate return period.

The outputs from the function are .png files (one per return period) containing plots showing results from all four fitted distributions (‘*gauge_number* (*gauge name – if exists*) GEV|GLO all distribution plots (*RP*-year).png’). They should look like the following. It can be seen that the results from the different models can vary hugely.



All these functions could be run as follows.

```r
# Set up data to use as inputs to the fit.time function
dist <- "GEV"
folder <- "C:/Non-stationarity/Data"
RPs <- c(2, 10, 20, 50, 100)
CI <- 90
# Run the fitting function, saving the outputs to an object called 'GEVfits'.
GEVfits <- fit.time(dist, folder, RPs, CI)
#>
  |
  |                                                                      |   0%[1] "2 gauging stations queue
#> C:/Non-stationarity/Data/Inputs/24004.AM
#> C:/Non-stationarity/Data/Inputs/76005.AM
#>
#> [1] "C:/Non-stationarity/Data/Inputs/24004.AM"
#> Loading required namespace: parallel
#> Replicate 100
#> Replicate 200
#> Replicate 300
#> Replicate 400
#> Replicate 500
#> Replicate 600
#> Replicate 700
#> Replicate 800
#> Replicate 900
#> Replicate 1000
#> Replicate 100
#> Replicate 200
#> Replicate 300
#> Replicate 400
#> Replicate 500
#> Replicate 600
#> Replicate 700
#> Replicate 800
#> Replicate 900
#> Replicate 1000
#> Ratio of bias to standard error is high
#> Replicate 100
#> Replicate 200
#> Replicate 300
#> Replicate 400
#> Replicate 500
#> Replicate 600
#> Replicate 700
#> Replicate 800
#> Replicate 900
#> Replicate 1000
#> Ratio of bias to standard error is high
#> Replicate 100
#> Replicate 200
#> Replicate 300
#> Replicate 400
#> Replicate 500
#> Replicate 600
#> Replicate 700
#> Replicate 800
#> Replicate 900
#> Replicate 1000
#> Ratio of bias to standard error is high
#>
  |
  |================================                                 |  50%[1] "C:/Non-stationarity/Data
#> Replicate 100
#> Replicate 200
#> Replicate 300
#> Replicate 400
#> Replicate 500
#> Replicate 600
#> Replicate 700
#> Replicate 800
#> Replicate 900
#> Replicate 1000
#> Replicate 100
#> Replicate 200
#> Replicate 300
#> Replicate 400
#> Replicate 500
#> Replicate 600
#> Replicate 700
#> Replicate 800
#> Replicate 900
#> Replicate 1000
#> Ratio of bias to standard error is high
#> Replicate 100
#> Replicate 200
#> Replicate 300
#> Replicate 400
#> Replicate 500
#> Replicate 600
#> Replicate 700
#> Replicate 800
#> Replicate 900
#> Replicate 1000
#> Ratio of bias to standard error is high
#> Replicate 100
#> Replicate 200
#> Replicate 300
#> Replicate 400
#> Replicate 500
#> Replicate 600
#> Replicate 700
#> Replicate 800
#> Replicate 900
#> Replicate 1000
#> Ratio of bias to standard error is high
#>
  |
  |=================================================================| 100%
#> [1] "*** Total processing time:"
#> Time difference of 1.492201 mins
```

These results are then saved in your R environment and can then be used in the plotting functions. If you wish to save the results so that they can be used later in the plotting functions (in a new R session), this can be achieved by saving the results as a .RDS file (a serialized version of the data which is saved with gzip compression by default).

```
# Save fitting results as a .RDS file
saveRDS(GEVfits, "Outputs/Time covariate only/GEV fits from fit time function.RDS")
# To read this back into R at a later date:
GEVfits <- readRDS("Outputs/Time covariate only/GEV fits from fit time function.RDS")
```

Once the fitting function has been run, you can run the plotting functions.

```
# Plot results from one of the fitted models
# Set up data to use as inputs to the plot.selected.distributions function
folder <- "C:/Non-stationarity/Data"  # this does not need to be run if 'folder' is
                                      # already in your R environment from a previous step
gauge <- "24004"
type <- "Varying location and scale"  # choose from 'Varying location',
                                      # 'Varying scale' and 'Varying location and scale'
fitResults <- GEVfits
RPs <- c(2, 10, 20, 50, 100)
CI.RP <- 100  # return period for which a confidence interval is required
CI <- 90  # set to calculate 90% confidence interval
# Run the function
plot.selected.distributions(folder, gauge, type, fitResults, RPs, CI.RP, CI)
#> pdf
#>   2

# Plot results from all of the fitted models
# Set up data to use as inputs to the plot.all function
folder <- "C:/Non-stationarity/Data"
gauge <- "24004"
fit.results <- GEVfits
RPs <- c(2, 10, 20, 50, 100)  # return periods to plot
# Run the function
plot.all(folder, gauge, fit.results, RPs)
# Note that the names of the input data do not have to match the names of the
# function arguments. Here, fit.results is being used as an input to the fitResults
# parameter. This could also be included as fitResults = fit.results, rather than
# just fit.results. Specifying the argument like this would be required if the
# arguments were given in the wrong order in the function, e.g.
# plot.all(fitResults = fit.results, folder = folder, RPs = RPs, gauge = gauge).
# This would also work.
```

If you wish to plot results for more than one gauge at a time, it is possible to write some code to loop through each gauge. This would only be possible for plot.selected.distributions if the same type of non-stationary model is to be fitted for each gauge. The following code could be used to loop through the gauges.

```
# Example of looping through gauges for plotting

# Set parameters
dist <- "GEV"
folder <- "C:/Non-stationarity/Data"
RPs <- c(2, 10, 20, 50, 100)
CI <- 90
CI.RP <- 100
type <- "Varying location"

# Fit models
GEVfits <- fit.time(dist, folder, RPs, CI)
fitResultsGEV <- GEVfits

# List gauges to loop through
# The following line of code lists all .AM files in your 'Inputs' folder
gauges <- list.files(paste0(folder, "/Inputs"), pattern = ".AM", full.names = TRUE)
```

```
# Read in the metadata file in your 'Inputs' folder
metadata <- read.csv(paste0(folder, "/Inputs/Gauge_metadata.csv"),
                     stringsAsFactors = FALSE)

# Loop through gauges
for (i in 1:length(gauges)){

  print(gauges[i])

  # Pre-process the data to extract the gauge number from within the .AM file
  # (rather than from the file name). The former has been used in the analysis.
  # This function is an internal function of the package and hence is not
  # documented in this guidance, however there is a help file available.
  preprocessedData <- import.preprocess.data(gauges[i], metadata = metadata)

  # If pre-processing has not produced an error then plot results
  if (!is.null(preprocessedData$res)){

    gauge <- preprocessedData$res$gaugeNumber  # extract number of gauge to plot

    plot.selected.distributions(folder, gauge, type, fitResultsGEV, RPs, CI.RP, CI)

    plot.all(folder, gauge, fitResultsGEV, RPs)

  }

}
```

Fitting models with physical covariates

The functions described in this section allow users to incorporate physical covariates into flood frequency analysis.

One important consideration is the effect of collinearity and confounding variables on results when including multiple covariates. It is desirable that covariates included in a non-stationary statistical model are independent, otherwise computational issues may arise during inference and the fitted model may suffer from problems with interpretability. When time is one of the covariates, it is recommended that this is achieved by detrending the physical covariates based on the time covariate. This can be achieved in the fit.phys.cov function by using the detrend.cov argument.

There are two steps to carry out analysis using physical covariates in this package. The first is fitting of models (varying the parameters by a range of covariate combinations) to flow data for all gauges in the 'Inputs' folder. The second is obtaining return level estimates (flows) for each gauge individually using one of the fitted models for that gauge.

Unlike the time covariate, physical covariates need to be supplied by the user in a .csv file (see the example file *24004_covariates.csv*). This should be included in the 'Inputs' folder with the flow data. The gauge number in the covariates file name must match the one in the .AM file, so if it has been amended in the .AM file, e.g. *24004x*, then the covariates file should be called *24004x_covariates.csv*. A value of the covariate must be associated with every AMAX flow (e.g. a value of annual or seasonal rainfall would be required for every water year in the flow record). The flow and covariate data should therefore cover the same time period, although the package will automatically trim both datasets to use only the overlapping time period. The file must contain a maximum of seven physical covariates, otherwise the fit.phys.cov function will produce an error. Gauges without a covariate data file will be skipped.

The example file contains a range of covariates. The NAO and EA indices were obtained from NOAA (https://www.cpc.ncep.noaa.gov/data/teledoc/telecontents.shtml). The indices are calculated from anomalies that are standardised by monthly means and standard deviations calculated over a 1950-2000 baseline period.
In all cases, annual values of the covariates were calculated using water years.
Each rainfall covariate was centred and scaled by subtracting the mean of the covariate time series from each observation in the record and then dividing the resulting values by the standard deviation of the time series. This can be carried out using the 'scale' function in R. The other covariates were already standardised in a similar way. It is recommended that all covariates are centred and scaled.

fit.phys.cov

As with the fit.time function, the fit.phys.cov function fits stationary and non-stationary GEV or GLO distributions to AMAX flow data for all gauges provided by the user in an 'Inputs' folder. The non-stationary models have varying location and/or scale parameters that can vary with one or more physical covariate(s) and/or time. Outputs from the function are automatically written out to the 'Outputs/Physical covariates models' folder. The 'Physical covariates models' sub-folder will be automatically generated if it does not exist. Time (a centred and scaled index of water year calculated within the function and given the name "ScaledWaterYearIndex") is automatically included as a covariate alongside the physical covariates, although the function fits a range of models with various combinations of covariates, so time will not be included in every model fitted.

The function takes the following inputs, as documented in the package help files:

- dist: The statistical distribution to use in the fitting ("GEV" or "GLO").
- folder: A path to a folder containing two sub-folders. One must be called 'Inputs' and one must be called 'Outputs'. The 'Inputs' folder must contain .AM files and .csv files containing the physical covariates data for each gauge for which analysis is required. The 'Outputs' folder is where the 'Physical covariates models' sub-folder will be created, which is the location to which results files will automatically be written.
- detrend.cov: If included (i.e. not NULL), the function detrends all the covariates based on the covariate in this string, usually time (represented by "ScaledWaterYearIndex" in this function). This is recommended for models where a physical covariate and time may both be included in the location or scale parameter at the same time (options 1 and 2 below). The default is "ScaledWaterYearIndex" (i.e. by default, all the physical covariates are detrended based on the time covariate), but this can be changed to NULL if you do not wish to detrend the covariate data. It could also be set to any covariate included in the covariates input .csv file (the string here must match the name in the .csv file).
- option: Must be 1, 2 (default) or 3. Option 1 fits models using all possible combinations of covariates supplied in the .csv file (this can produce tens of thousands of fitted models if many covariates are considered, for example, ~65,000 models for 7 physical covariates plus time); Option 2 fits models using combinations which only include one physical covariate and/or time (this would produce 88 fitted models for 7 physical covariates plus time); Option 3 fits models using combinations which include one covariate at a time in the location and/or scale parameter.
- best_fit_method: Optional. Method of automatically selecting a best fit model that can be used in the next step if desired. Must be "AIC" or "BIC" (default).

The function fits a GEV or GLO distribution using maximum likelihood estimation for a stationary case and a number of non-stationary cases (depending on the option selected). The scale parameter is log-transformed to ensure positivity for all possible covariate values.

The outputs from running the function are as follows.

'*gauge_number* (*gauge name – if exists*) GEV|GLO fitting information (physical covariates).csv': A spreadsheet containing the information about each fitted model. There is one row per model. The locFit column gives the regression equation of covariates included in the location parameter for that model. The scaleFit column gives the regression equation of covariates included in the scale parameter for that model. For example:

- If the location or scale parameter is constant (i.e. no covariate is included in it), then the equation for that parameter is "~ 1". If the model is stationary, then no covariates are included in either parameter, so the equation is just "~ 1" for both the location and scale parameter because they are constant.
- If the model is non-stationary with the location and/or scale parameter varying with time (i.e. ScaledWaterYearIndex, the covariate that is automatically-generated within the package to represent time), then the equation for the varying parameter/s is "~ 1 + ScaledWaterYearIndex".
- If the model is non-stationary with the location and/or scale parameter varying with a physical covariate (e.g. annual rainfall, which may be called Rain_WY in the input covariates file and hence also in the outputs), then the equation for the varying parameter/s is "~ 1 + Rain_WY".
- If the model is non-stationary with the location and/or scale parameter varying with time and a physical covariate, then the equation for the varying parameter/s may be "~ 1 + Rain_WY + ScaledWaterYearIndex".

Additional covariates are included in the equation as extra terms. If the location and/or scale parameter was varying with the maximum of seven physical covariates allowed in the package, as well as time, then the equation would have nine terms: "~1 + Covariate1 + Covariate2 + Covariate3 + Covariate4 + Covariate5 + Covariate6 + Covariate7 + ScaledWaterYearIndex".

The warnings and errors columns give any warnings or errors that occurred during the fitting of the model (this is more likely when option 1 is selected as there may be many covariates in one model). Models with warnings or errors should not be selected for calculating return levels. Therefore, it is important to check this spreadsheet before running the 'res.phys.cov' function. The AIC and BIC are as described for the models with only time as a covariate. This file also contains columns which rank the models in terms of AIC and BIC. If detrend.cov has been set, this is reported in the 'DetrendingCovariate' column, and the 'CovariatesOption' column records which option was selected when the function was run.

The 'params' column gives the parameters of the fitted distribution. For a stationary fit, the location, ln(scale) and shape parameters are provided. However, as for the models where time is the only covariate, for the non-stationary fits, the outputs change depending on the varying parameter(s). If the location parameter is varying then the location parameter (μ) is replaced with a μ value for the intercept and each covariate. This is equivant to using μ_0 to μ_n (as used in the fit.time function results where there was only one covariate and hence there was only a μ_0 and a μ_1). If the scale parameter is varying then the ln(scale) parameter (ϕ) is replaced with a ϕ value for the intercept and each covariate, equivalent to ϕ_0 to ϕ_n. This is described in the following equations where the parameters vary with a vector of covariates, x:
$\mu(x) = \mu_0 + \mu_1 x_1 + ... + \mu_n x_n$,
$ln(\sigma(x)) = \phi(x) = \phi_0 + \phi_1 x_1 + ... + \phi_n x_n$,
where n is the number of covariates included in the parameter.
For example, in the above equation, if covariate x_1 represents time and covariate x_2 is a measure of rainfall, μ_1 would express how floods change over time, once variations in rainfall are taken into account by μ_2 (Prosdocimi et al., 2014).

'*gauge_number* (*gauge name – if exists*) GEV|GLO model fits (physical covariates).RDS': This is for reference purposes, if any information from the fitting process needs to be accessed again in the future. It is a list containing information for each gauge analysed. For each gauge, the following information is saved: dist (distribution used - taken from the input argument), model_fits (described below), all_data_raw (the data used in the model fitting - not detrended), gaugeName (name of the gauge) and detrend.cov (the string from the input argument). Most of these are also returned from the function so that they can be used in the next step (see code example).

The model_fits element is another list containing the following information: best_model (details of the best fitting model selected based on AIC or BIC, as specified by the user), best_model_formula (details of the formula used to fit best_model), stat_model (details of the fitted stationary model) and all_models_metadata (a metadata file associated with fitting models to all combinations of covariates selected by the user). The latter is the same as the .csv file described above.

Once this function has been run, the res.phys.cov function can be run for each gauge to obtain return levels (flows) based on one of the fitted models, selected by the user.

res.phys.cov

The res.phys.cov function uses one of the fitted non-stationary models from the fit.phys.cov function to estimate conditional, marginal and present-day marginal return levels (as described in the 'Fitting non-stationary models' section of this document). Stationary return levels are also estimated based on the fitted stationary model. The function also outputs plots of the results. The following inputs are required:

- folder: As for the fit.phys.cov function. The results files will automatically be written out to 'Outputs/Physical covariates models'.
- physCovModelFits: Output from fit.phys.cov function.
- gauge: Number of gauge to analyse. Must have been used in the fit.phys.cov function and must match the number in the fitting information .csv file output from the fit.phys.cov function (so if the number has been edited then it should be the edited number, e.g. 24004x rather than 24004).
- RPs: The return periods for which frequency analysis is required. A maximum of eight return periods can be plotted on the 'model results' plot - if more are listed here then only the first eight will be selected for the plot. All return periods will be used for the frequency analysis and for the 'encounter probability' plot.
- CI: The required confidence interval for the results as a percentage, e.g. '90' for 90% is the default.
- model_type: Must be "Best" (default) or "User". This is for the user to select whether to use the best fit model based on AIC or BIC (whichever they specified in the fit.phys.cov function) or a user-selected model to calculate return levels (and then plot). The selected model should ideally be a non-stationary model as results will automatically be written out for the stationary model. Some results will not be available for a stationary model.
- locFit: This is only required if model_type = "User". It is the regression equation of covariates to include in the location parameter (copy from the 'GEV|GLO fitting information (physical covariates)' .csv file produced by the fit.phys.cov function for the gauge), e.g. locFit = "~ 1 + NAO_SON + ScaledWaterYearIndex".
- scaleFit: This is only required if model_type = "User". It is the regression equation of covariates to include in the scale parameter (copy from the 'GEV|GLO fitting information (physical covariates)' .csv file produced by the fit.phys.cov function for the gauge), e.g. scaleFit = "~ 1 + ScaledWaterYearIndex".
- present_day: Indicates whether to calculate the present-day marginal return level. Set to TRUE to calculate the present-day marginal return level as well as the marginal return level, otherwise set to FALSE. This can only be TRUE if time ("ScaledWaterYearIndex") is included as a covariate in the selected model. If it is set to TRUE and "ScaledWaterYearIndex" is missing from the model covariates then it will automatically be reset to FALSE.

The outputs from running the function are as follows.

'*gauge_number* (*gauge name – if exists*) GEV|GLO model.RDS': A .RDS file is returned which contains information relating to the analysis for the gauge (from the model used to estimate return levels to the results of the frequency analysis). This can be opened in R and referred to at a later date if required. Much of the information is also written out in a range of .csv files, as described below.

'*gauge_number* (*gauge name – if exists*) GEV|GLO model formula.csv': This contains the regression equations used in the location and scale parameters for the selected model (i.e. which covariates have been included). See previous section for explanation of these equations.

'*gauge_number* (*gauge name – if exists*) GEV|GLO conditional return levels.csv': This is similar to the '*gauge_number* (*gauge name – if exists*) GEV|GLO results.csv' output from the fit.time function. The

Flow column is the observed AMAX flow data and the WaterYear column is the water year associated with each flow. Other columns at the beginning show the covariate data used in the model fitting (detrended, if detrend.cov has been used). If time is included as a covariate in the selected model, then the WaterYearIndex column is the automatically-generated index representing time and the ScaledWaterYearIndex column is the centred and scaled version of this, which was actually used as the time covariate in the model fitting. The conditional return levels (and their confidence intervals) are given in the remaining columns (these are taken from the median of the bootstrap sample). These vary based on whichever covariates have been used in the model selected in this function for calculating return levels. As previously mentioned, given that multiple covariates may have been used, these results can be less useful than those derived from stationary models or the non-stationary models with only time as a covariate. For example, the return level may be conditional on a specific water year and rainfall amount.

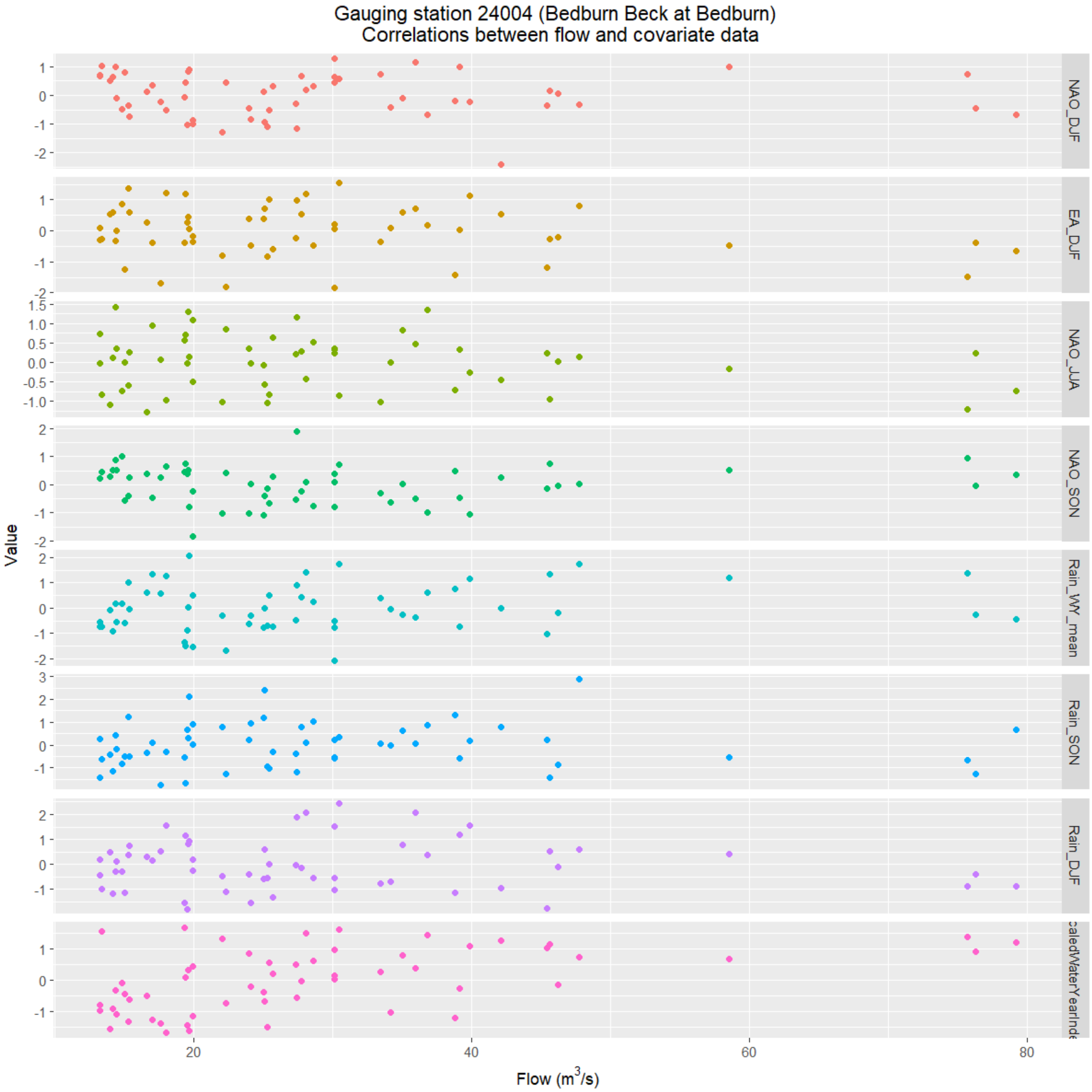
'*gauge_number* (*gauge name – if exists*) GEV|GLO marginal return levels.csv': This provides the marginal return levels (and confidence intervals) for each return period. Unlike the conditional return levels, there is only one value per return period. It indicates the return level corresponding to a particular averaged exceedance probability (the encounter probability) during the entire period of record. For example, the 100-year marginal return level is the value expected to be exceeded once every 100 years assuming the covariate distribution in the observed period is unchanged. There may be a higher chance of that exceedance occurring in the second half of the record, if the conditional return levels are higher in that half. These are calculated from the fitted model, rather than the median of the bootstrap sample; the confidence intervals are from the relevant quantiles of the bootstrap sample. The same applies for the present-day marginal return levels below.

'*gauge_number* (*gauge name – if exists*) GEV|GLO present-day marginal return levels.csv': This provides the present-day marginal return levels (and confidence intervals) for each return period. This is similar to the marginal return levels, except that the time covariate has been set to be the last year in the input flow and covariate data (representing the present-day) and then the averaging is done over the full observed distribution of the physical covariates.

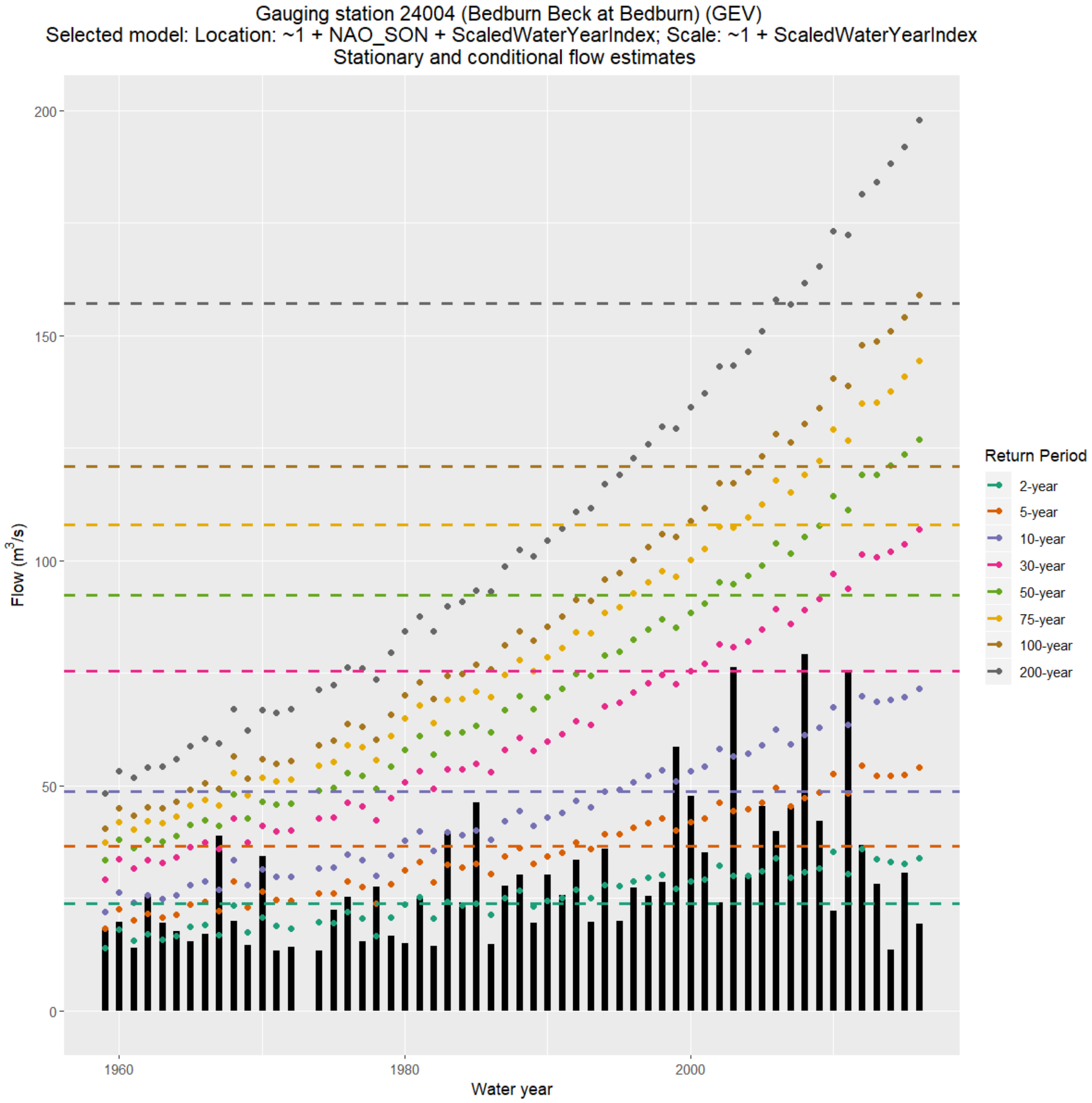
'*gauge_number* (*gauge name – if exists*) GEV|GLO stationary return levels.csv': This provides the return levels for each return period based on the stationary model. This is the usual definition of a return level (the flow associated with a particular return period).

'*gauge_number* (*gauge name – if exists*) GEV|GLO model diagnostic plots.png': This contains the diagnostic plots relating to the selected model. These should be checked to make sure that the selected model has a good fit, rather than just relying on the AIC or BIC value. The plots are the same as those from the fit.time function, but the titles will also contain physical covariates in the location and scale parameter equations, as appropriate.

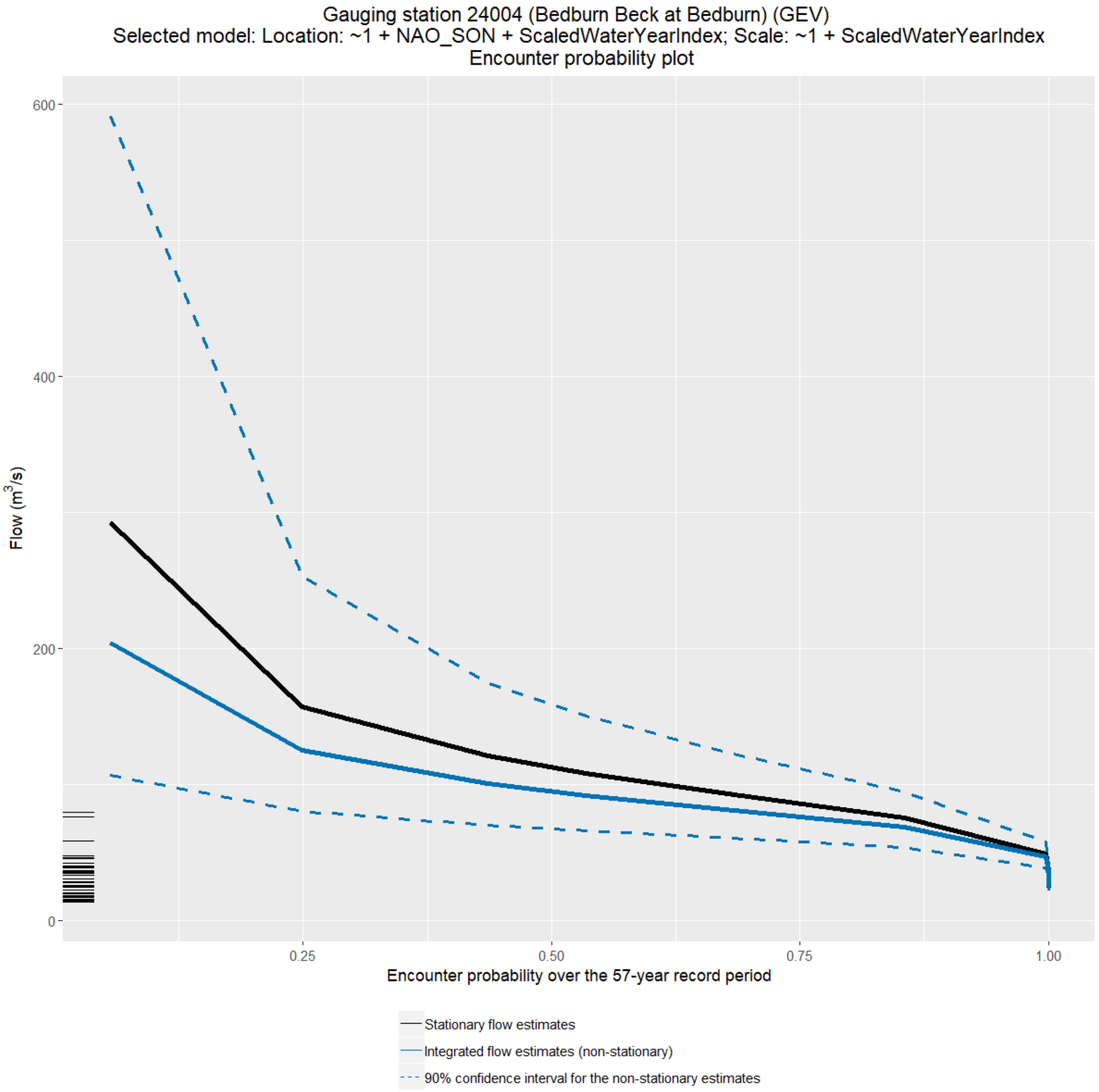
'*gauge_number* (*gauge name – if exists*) GEV|GLO correlation plot.png': This is a plot of the flow data plotted against all covariates in the input file (detrended, if detrend.cov has been used) and can be used to see whether flow is correlated with the covariates used. The plot will look like the following.



‘*gauge_number* (*gauge name – if exists*) GEV|GLO model results plot.png’: This is a plot of flow plotted against conditional return levels from the non-stationary model and stationary return levels from the stationary model for the observed period. An example can be seen below. The dots are the conditional return levels and these will move up and down each year, depending on the covariates. The stationary return levels are the dashed horizontal lines. If the selected model is stationary then only the stationary return levels will be plotted as conditional return levels will not be available.



‘*gauge_number* (*gauge name – if exists*) GEV|GLO encounter probability plot.png’: This is a plot of flow against encounter probability, in this case, the probability of the flow being exceeded over the full period of record. An example can be seen below. The plot shows both the stationary return levels (‘Stationary flow estimates’) and marginal return levels (referred to as ‘Integrated flow estimates’, as in the practitioner guidance), including a confidence interval for the latter. The observed AMAX flows are plotted as short black lines on the y-axis. If the selected model is stationary, then marginal return levels will not be available and therefore only the stationary results will be plotted, along with their confidence interval. The smoothness of the curve and range of results shown on the plot will depend on the number of return periods selected by the user because the plot uses the same stationary and marginal return levels that are output in the .csv files.



Any warnings and errors associated with the analysis will be written out in the R console. Due to the use of bootstrapping to calculate confidence intervals, 200 (the number of bootstrap samples used) models are fitted using maximum likelihood estimation. There are therefore likely to be some errors and warnings associated with some of these models.

You may get messages stating that there were a number of warnings or errors when fitting the model for a particular bootstrap sample. This is for information only. You may then also get a message stating that there are errors or warnings in more than 10 of the bootstrap models for at least one return period. This means that there may be issues with at least 5% of the 200 bootstrap models used to find marginal or present-day marginal (present-day will be stated in the message if applicable) return level confidence intervals.

In addition, you may get a message stating the number of errors or warnings that occurred when calculating marginal return levels from the 200 models in order to calculate the confidence intervals over all the required return periods. Bootstrap samples with errors or warnings will probably result in NAs and hence not be used to calculate the confidence interval. There may also be additional errors from the model fitting that appear in the console, such as ‘Error in solve.default(o$hessian)’ or ‘Error in diag(o$cov)’. These imply poor-fitting of the model. Therefore, if there are a large number of errors or warnings then it may be that less confidence should be placed in the values of the confidence interval. It is worth noting that non-convergence is indicative of an issue somewhere in the model assumptions on which the estimation process is based, so it is useful to understand the context of the data and the model being fit to it.

Information about the bootstrapping results, warnings and errors is also saved in the .RDS file under ‘margRLsAllInfo’. If you do need to refer to this information, the explanation for each output can be found in the ‘BootstrapMargRLs’ function help file (this function is only used within other functions and hence is not included in this document). The information includes the marginal return levels for each bootstrap sample and return period, so you can see where the problems are. It is possible that there may be some poorly-fitting bootstrap models that have been used to produce results for lower return periods but have resulted in NAs for higher return periods, in which case, it is likely that any results produced are not sensible. If this is a problem with lots of the bootstrap samples, then the confidence interval for each return period may be less accurate. Additionally, any models containing warnings or errors in the ‘fitting information’ .csv file that is output from the fit.phys.cov function should not be used. If any confidence intervals are coming out the same as the estimate, this also implies that there were problems with the model fitting and that the results should not be used.

These two steps for using physical covariates can be run as in the example below.

```
# Fit models using physical covariates
folder <- "C:/Non-stationarity/Data"
dist <- "GEV"
# Run function and save outputs to an object named 'physCovModelFits'
physCovModelFits <- fit.phys.cov(dist, folder, detrend.cov = "ScaledWaterYearIndex",
                                 option = 2, best_fit_method = "BIC")
#>
  |
  |                                                                  |   0%[1] "2 gauging stations queue
#> C:/Non-stationarity/Data/Inputs/24004.AM
#> C:/Non-stationarity/Data/Inputs/76005.AM
#>
#> [1] "C:/Non-stationarity/Data/Inputs/24004.AM"
#> Replicate 100
#> Replicate 200
#> Replicate 300
#> Replicate 400
#> Replicate 500
#> Replicate 600
#> Replicate 700
#> Replicate 800
#> Replicate 900
#> Replicate 1000
#> Ratio of bias to standard error is high
#> Replicate 100
#> Replicate 200
#> Replicate 300
#> Replicate 400
#> Replicate 500
#> Replicate 600
#> Replicate 700
#> Replicate 800
#> Replicate 900
#> Replicate 1000
#>
```

```
  |
  |================================                                 |  50%[1] "C:/Non-stationarity/Data
#> Replicate 100
#> Replicate 200
#> Replicate 300
#> Replicate 400
#> Replicate 500
#> Replicate 600
#> Replicate 700
#> Replicate 800
#> Replicate 900
#> Replicate 1000
#> Ratio of bias to standard error is high
#> Replicate 100
#> Replicate 200
#> Replicate 300
#> Replicate 400
#> Replicate 500
#> Replicate 600
#> Replicate 700
#> Replicate 800
#> Replicate 900
#> Replicate 1000
#>
  |
  |=================================================================| 100%
#> [1] "*** Total processing time:"
#> Time difference of 1.102384 mins

# Obtain return levels based on one of the fitted models
# Set inputs
gauge <- "24004"
RPs <- c(2, 10, 20, 50, 100)
CI <- 90
# Run function
res.phys.cov(folder, physCovModelFits, gauge, RPs, CI, model_type = "Best",
             locFit = NULL, scaleFit = NULL, present_day = TRUE)
#> null device
#>           1
```

Complete example script to run all functions

```
# Setup -------------------------------------------------------------------
# Install package dependencies
install.packages(c('changepoint', 'trend'))

# Install the nonstat package from saved location
path_to_file <- "C:/Non-stationarity/nonstat_1.0.0.tar.gz"
install.packages(path_to_file, repos = NULL, type="source")

# Load the nonstat package
```

```
library(nonstat)

# Set the path to the folder containing the 'Inputs' and 'Outputs' folders
folder <- "C:/Non-stationarity/Data"

# Trend testing ---------------------------------------------------------
MK.test(folder, sig.level = 5)
Pettitt.test(folder, sig.level = 5)
PELT.test(folder)

# Flood frequency analysis (time only as a covariate) ---------------------

# Run the fitting function
dist <- "GEV"
RPs <- c(2, 10, 20, 50, 100)
CI <- 90  # set the confidence interval to be 90%
GEVfits <- fit.time(dist, folder, RPs, CI)

# Run the plotting functions
gauge <- "24004"
type <- "Varying location and scale"
fit.results <- GEVfits
RPs <- c(20, 50, 100)  # return periods to plot
CI.RP <- 100  # return period for confidence interval plot
plot.selected.distributions(folder, gauge, type, fit.results, RPs, CI.RP, CI)

RPs <- c(2, 10, 20, 50, 100) # return periods to plot
plot.all(folder, gauge, fit.results, RPs)

# Flood frequency analysis (physical covariates) --------------------------

# Run the fitting function
detrend.cov <- "ScaledWaterYearIndex"
option <- 2
best_fit_method <- "BIC"
physCovModelFits <- fit.phys.cov(dist, folder, detrend.cov, option, best_fit_method)

# Obtain return levels based on one of the fitted models
gauge <- "24004"
RPs <- c(2, 10, 20, 50, 100)
res.phys.cov(folder, physCovModelFits, gauge, RPs, CI, model_type = "Best",
             locFit = NULL, scaleFit = NULL, present_day = TRUE)
```

References

Box, G. E. & Cox, D. R. (1964). An analysis of transformations. Journal of the Royal Statistical Society: Series B (Methodological), 26(2), 211-243.

Eastoe, E. F. & Tawn, J. A. (2009). Modelling non-stationary extremes with application to surface level ozone. Appl. Statist. 58, 22–45.

Killick, R., Fearnhead, P. & Eckley, I.A. (2012). Optimal detection of changepoints with a linear computational cost. Journal of the American Statistical Association, 107(500), 1590-1598.

Killick, R. & Eckley, I.A. (2014). changepoint: An R package for changepoint analysis. Journal of Statistical Software, 58(3), 1-19.

Prosdocimi, I., Kjeldsen, T. R. & Svensson, C. (2014). Non-stationarity in annual and seasonal series of peak flow and precipitation in the UK. Natural Hazards and Earth System Sciences, 14(5), 1125-1144.

Rougé, C., Ge, Y. & Cai, X. (2013). Detecting gradual and abrupt changes in hydrological records. Advances in Water Resour., 53, 33–44.

Šraj, M., Viglione, A., Parajka, J. & Blöschl, G. (2016). The influence of non-stationarity in extreme hydrological events on flood frequency estimation. J. Hydrol. Hydromech., 64(4), 426–437.