2023 HAWAII UNIVERSITY INTERNATIONAL CONFERENCES
SCIENCE, TECHNOLOGY & ENGINEERING, ARTS, MATHEMATICS & EDUCATION JUNE 7 - 9, 2023
PRINCE WAIKIKI RESORT, HONOLULU, HAWAII

COMPARISON STUDY OF THRESHOLD MODELING IMPLEMENTION IN PYTHON & R

NAUTH, MIKHAIL
DEPARTMENT OF DATA AND ANALYTICS
NYC HEALTH & HOSPITALS, USA
QUEENS, NEW YORK

Mr. Mikhail Nauth
Department of Data and Analytics
NYC Health + Hospitals, USA
Queens, New York

Comparison Study of Threshold Modeling Implementation in Python & R

Synopsis:

Extreme Value Analysis (EVA) focuses on rare events in large datasets. The Generalized Extreme Value Distribution (GEV) and the Generalized Pareto Distribution (GPD) are two common probability distributions used in (EVA). Threshold Modeling is an important aspect of this analysis. Although R is commonly used for Threshold Modeling, this paper shows that a Python-exclusive library can perform all the necessary functions for this type of analysis.

Comparison Study of Threshold Modeling Implementation in Python & R

Keywords: Statistics, Probability, Python, Finance, Computational Mathematics

Acknowledgement

I would like to acknowledge Dr. Genady Grabarnik, PhD of the St. John's University Department of Mathematics and Computer Science for his advisement and guidance throughout the process of performing the research and preparing this paper. His experience and wisdom in this area were paramount, especially to someone who would be considered a novice.

Abstract

This paper explores the use of Python in performing Extreme Value Analysis, specifically Threshold Modeling, and compares it to the capabilities of R. The study found that Python's SciPy library is capable of performing all the necessary functions related to Threshold Modeling, with comparable accuracy to R but with clearer and more legible visualizations. This result is significant for further research planned in Time Series Analysis and Machine Learning.

1. Introduction / Objective & Goals

Currently, most statistical testing and analyses are done in the R programming language. This includes Extreme Value Analysis (EVA) and more specifically, Threshold Modeling. There is a wide range of statistical R packages that are able to perform these analyses, produce plots, create randomized datasets and output predictive models, such as 'extRemes', 'Rsafd', and 'SpatialExtremes' (Gilleland et al., 2012). However, the Python programing language is more popular and considered more general-purpose. Additionally, in terms of machine learning, Python is the preferred choice (Kumar, 2021).

EVA is especially useful when looking at time series in fields such as climatology and finance. Coupling this analysis with machine learning is much more convenient in Python than R since there are established libraries and algorithms in Python. However, Python isn't as robust as R in the field of EVA.

While there are some libraries in Python that can be manipulated to obtain the same results as in R, many of them are either not fully built, i.e., they don't house all the necessary functions

needed for Threshold Modeling, or they have compatibility issues since they rely on R for their functions. Therefore, the goal of this paper is to show that a Python-exclusive library can perform all the necessary functions related to Threshold Modeling, while matching the results of the R packages.

2. Background & State of the Art

In order to proceed with the Python-exclusive library, it is imperative to understand the definition and purpose of EVA. This section will cover the family of distributions used in EVA and discuss current programming packages.

2.1 Extreme Value Analysis

EVA studies the occurrences of rare events i.e., values in a large dataset with a low probability of occurring. In many fields, these extreme values are treated as outliers. However, in fields such as hydrology, climatology, oceanography, structural engineering, epidemiology and finance keen attention is paid to these extreme events.

For example, in studying temperature magnitudes, the overall distribution of values can be examined, however, in order to determine the likelihood of heat waves or cold outbreaks, the extreme values are the primary focus (Pinheiro & Grotjahn, 2015).

2.2 Extreme Value Distributions

Extreme Value Distributions (EVD) are used in Extreme Value Analysis to model rare events. The majority of probability distributions are concerned primarily with the entire random sample. In the case of the Gaussian distribution, for example, it assigns larger probabilities to values closer to the mean and smaller probabilities to values in the tails. EVD's are concerned with those tails. Typically, the tails of an EVD are much thicker than a standard normal distribution (two-tails) or an exponential distribution (one-tail) (Carmona, 2013). One method used to determine if a dataset would be considered an EVD is a Q-Q plot.

2.2.1 Generalized Extreme Value Distribution

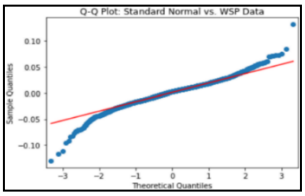

Fig. 1: Q-Q Plot of the Standard Normal Distribution vs. the S&P 500 Weekly Stock Price

Figure 1 compares the standard normal distribution to the S&P 500 Weekly Stock Price data, which displays the heavier tails. This strongly suggests that an EVD would be appropriate to

fit the dataset. In Figure 2 below, the exponential distribution is used to compare its quantiles to the Property Claims Services data, which is a one-tailed distribution. Again, the plot suggests an EVD would be appropriate to fit the data because of the heavier tail.

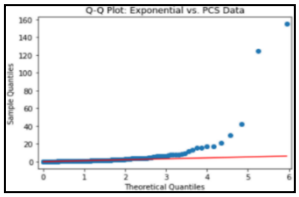

Fig. 2: Q-Q Plot - Exponential Distribution (mean = 1) vs. the PCS Data

The first of these EVD's is the Generalized Extreme Value Distribution (GEV). The GEV is a family of continuous probability distributions, including the Gumbel, Frechet and Weibull distributions. The cumulative distribution function for the GEV is:

$$F_{\mu,\sigma,\xi}(x) = \exp\left\{-\left[1+\xi\left(\frac{x-\mu}{\sigma}\right)\right]^{-1/\xi}\right\} \quad (1)$$

The GEV depends on the location, scale and shape parameters: μ; σ; ξ (Carmona, 2013). The focus here, however, will be on another EVD: the Generalized Pareto Distribution.

2.2.2 Generalized Pareto Distribution

The Generalized Pareto Distribution (GPD) also focuses on extreme values, however, while the GEV focuses on the maximum values at specific intervals, the GPD looks at all data points above a specified threshold. In other words, this distribution is primarily used to model the tails of other distributions (Pinheiro & Grotjahn, 2015). The cumulative density function for the GDP is as follows:

$$F_{\xi}(x) = \begin{cases} 1-(1+\xi x)^{-\frac{1}{\xi}} = 1-\dfrac{1}{(1+\xi x)^{\frac{1}{\xi}}} & if\ x>0, \\ 0 & otherwise \end{cases} \quad (2)$$

The shape parameter, ξ, is the primary parameter for the GPD, however, it is also affected by a shift and scale parameter, m and λ, respectively. With these parameters, the cumulative density function becomes:

$$F_{\xi}(x) = \begin{cases} 1-\left(1+\xi\dfrac{x-m}{\lambda}\right)^{-\frac{1}{\xi}} & for\ \xi \neq 0, \\ 1-\exp\left\{-\dfrac{x-m}{\lambda}\right\} & for\ \xi = 0. \end{cases} \quad (3)$$

It is important to note that if $\xi > 0$, then this is just an ordinary pareto distribution with shape parameter, ξ. And if $\xi = 0$ and $m = 0$, then this corresponds to the exponential distribution with scale parameter, λ.

GPD's allow for a continuous range of possible shape parameters that lead to the ordinary Pareto, exponential and beta distributions. This feature allows the data to essentially decide

which distribution is most appropriate (Carmona, 2013). GPD's have three basic forms, each corresponding to a limiting distribution of exceedance data from a different class of underlying distributions:

- Distributions whose tails decrease exponentially, such as the standard normal distribution, lead to a generalized Pareto shape parameter of zero.
- Distributions whose tails decrease as a polynomial, such as Student's t-distribution, lead to a positive shape parameter.
- Distributions whose tails are finite, such as the beta distribution, lead to a negative shape parameter.

As previously mentioned, the GPD is primarily used to model the tails of other distributions. Additionally, these tails (extreme values) have their own probability distribution (Carmona, 2013). In the case of the GPD, the excess distribution of the values over a specified threshold, l, is defined as:

$$F_l(x) = \mathbb{P}\{X - l \leq x | X > l\} = \frac{F(x+l)-F(l)}{1-F(l)}, x \geq 0. \quad (4)$$

The resulting cumulative density functions is:

$$F_l(x) = 1 - \left[1 + \xi \frac{x}{\lambda+\xi(l-m)}\right]^{-\frac{1}{\xi}} \quad (5)$$

Appendix A lays out the derivation of the cumulative density function for the excess distribution. It is evident that Eq. (5) has the same form as Eq. (3), when $\xi \neq 0$. Therefore, for a GPD, the excess distribution is another GPD with the following parameters:

- $m' = 0$
- $\lambda' = \lambda + \xi(l - m)$
- $\xi' = \xi$

The location parameter is now zero, since there is no shift in the extreme values. The scale is updated since the extreme values are considerably greater than or less than the center of the distribution. Finally, the excess distribution has the same shape parameter as the original.

In the upcoming sections, this phenomenon will be used to model the extreme values of some datasets. Additionally, the method for determining the threshold, l, will be discussed.

2.3 Threshold Modeling in R and Python

There is a plethora of libraries in R that have some capabilities to perform Threshold Modeling. These include 'extRemes', 'Rsafd', 'SpatialExtremes', 'POT', and 'texmex' to name a few (Gilleland, 2012). In the following sections, 'SpatialExtremes' is used to generate random

samples in R, while 'Rsafd' is used to perform the threshold modeling.

In Python, however, options are limited. For example, the 'Tensorflow' library has functionality that can return random samples from a GPD given location, scale and shape parameters (Abadi, 2016). However, it is unable to fit a GPD to a dataset. The 'Scikit-extremes' library seems promising because it contains documentation for the GEV distribution. Nevertheless, the functionality for the Generalized Pareto Distribution still needs to be built (Correoso, 2019).

The well-known library of SciPy is capable of generating random samples from a GPD and can fit a dataset, as well as return the maximum likelihood estimators for each parameter. However, there is no built-in functionality for modeling extreme values (Virtanen, 2020). Finally, there is a library called 'thresholdmodeling' that is designed to perform threshold modeling. However, there is no functionality for generating random samples from a GPD and it can only fit a GPD to an excess distribution *above* a specified threshold. Lastly, the functions in this library rely on R functions from the 'POT' package to operate (Lemos et al, 2013).

3. Implementation of Threshold Modeling in Python

Using some real-world datasets, the upcoming sections will compare the outputs of threshold modeling in R with the results of Python libraries, as well as creating new functionality in SciPy.

3.1 One-Tailed Distribution

The Property Claim Services (PCS) Index is the year-to-date aggregate amount of total damage reported in the United States to the insurance industry. Each index value represents $100 million worth of damage. For example, a value of 72.4 for the national index in 1966, means that $(72.4x100) million (i.e., $7.24 billion) in damage was recorded that year (Carmona, 2013).

Table 1: First 5 rows of the PCS Data

Timestamp	Amount (in $100 million)
13	4
16	0.07
46	0.35
60	0.25
87	0.36

The first column in Table 1 contains timestamps for the catastrophic events, which are identified as codes. The focus, however, will be on the 2nd column: the amount of total damage. With this dataset, the focus is on modeling the larger costs of total damage (extreme values). The GPD will be used here, but is a GPD appropriate for this dataset?

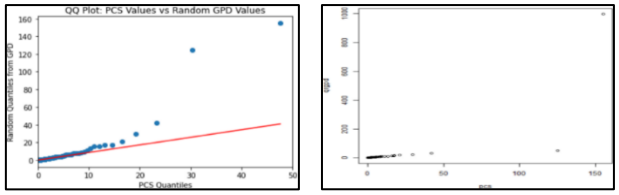

Fig. 3: Q-Q Plot of PCS dataset vs. random GPD distribution using (a) SciPy and (b) 'Rsafd'

Based on the Q-Q plots in Figure 3, the PCS data closely follows a straight line except for a few outliers. Therefore, it is safe to say that a GPD would be appropriate for fitting this dataset. It is important to note that in Figure 3b, there is no straight-line indicator as in Figure 3a.

3.2 Threshold Modeling

The threshold value for the PCS dataset will be set to 4. Thus, any value above 4 will be considered an extreme value. Later on, determining this value will be discussed.

The code block in Appendix B outputs the scale (lambda, λ) and shape (xi, ξ) parameters for the extreme values (greater than the threshold of 4) for the PCS dataset.

The first Python library to which these results will be compared is the 'thresholdmodeling' library, mentioned earlier. Initial testing of the PCS data with this library returned large and clunky outputs, which will not be included here. However, further investigation into the underlying functions of this package revealed a dependency on R. The primary operations were completed by the 'POT' package. The code snippet in Appendix C is used to allow the 'POT' package to run in Python.

In Appendix D, the 'POT' package output matches the initial run in R. The 'POT' function of 'fitgpd' requires the type of estimation that will be used. Some examples of these include L-moments, moments and maximum goodness-of-fit estimators. Only the maximum likelihood estimator ('mle') will be used. The final library to test will be the SciPy library. It is important to note here than in both the 'Rsafd' and 'POT' runs, the location parameter is assumed to be 0. In SciPy, the location parameter has to be initialized to 0 because the GPD function attempts to fit all parameters with respect to the default GPD.

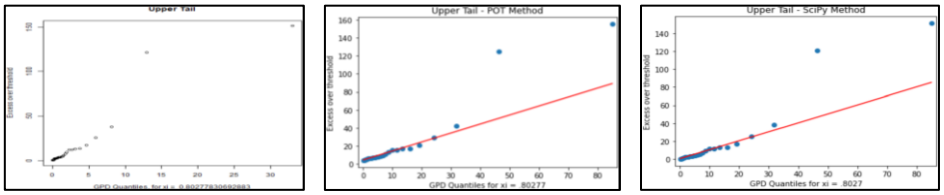

Fig. 4: Q-Q Plot from (a) 'Rsafd', (b) 'POT' and (c) SciPy of excess values from PCS dataset

The SciPy library's output in Appendix E is extremely close to the previous runs from the R

packages. However, this does not necessarily mean that a GPD is the most appropriate distribution for this extreme value dataset. In order to confirm that a GPD is desirable, the Q-Q plot of the upper tail is examined. Figure 4 above displays the Q-Q plots from the 'Rsafd' package, 'POT' package and SciPy library, respectively. For all 3 plots, most of the points fall on a straight line, which is a strong indication that a GPD is appropriate.

3.2.1 Shape Plots

In the previous section, the threshold value of 4 was not arbitrarily chosen. Before choosing a threshold value, the following must be considered:

- The threshold can't be too large because there will be too few points to fit.
- The threshold can't be too small because points from the center of the distribution may be included (Carmona, 2013).

The best way to choose this value is via a shape plot. The shape plot graphs the estimates of the shape parameter, as it changes with the values of the threshold used to produce the estimates. In other words, the threshold, l, is the independent variable, while the shape parameter, ξ, is the dependent variable. The shape plot from the 'Rsafd' package in R is shown in Figure 5.

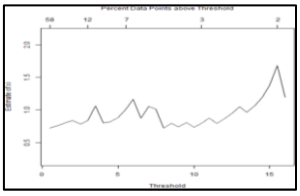

Fig. 5: Shape plot of PCS dataset from 'Rsafd' in R

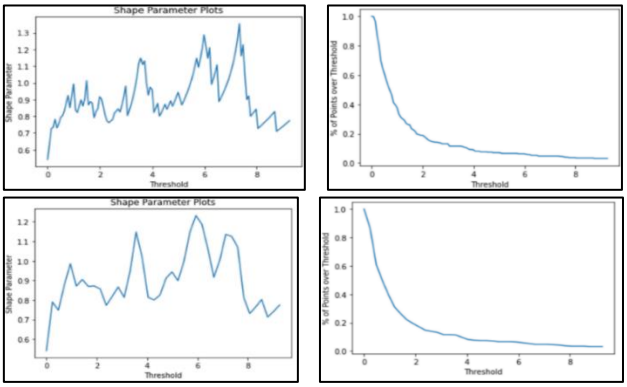

Fig. 6: Shape plot and threshold level from (a) 'POT' package and (b) SciPy library

In the 'thresholdmodeling' library in Python, there is a function called Parameter Stability plot, which is the same as the shape plot. However, because of the outdated underlying functions, it returns errors. Upon further investigation, the Parameter Stability plot loops through various thresholds, fits a GPD to each threshold from the 'POT' package and plots the threshold value vs. the shape parameter. Using this logic, shape plot functions were created in Python using both the 'POT' package and the SciPy library, where various thresholds were looped through

and plotted vs. the respective resulting shape parameter. Figure 6 above displays the respective shape plots with the portion of data points above the threshold mark. For the threshold range of 0-8, all shape plots are extremely similar. Since about 10% of the points are over the threshold of 4, this is a valid choice.

3.2.2 Estimations

With a threshold selected and a GPD fit to the excess values, a new GPD can be estimated and compared to the original dataset.

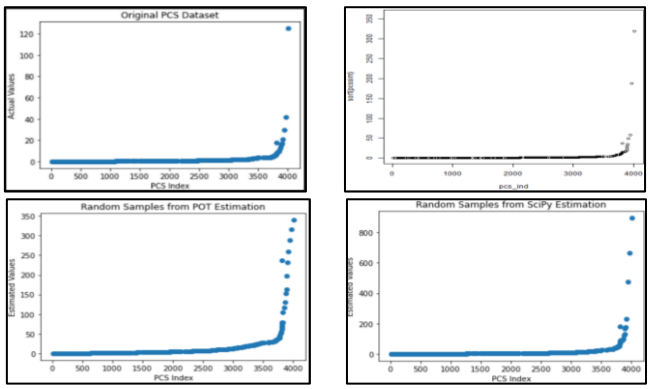

Fig. 7: (a) Original PCS data vs. Estimated data from (b) 'Rsafd', (c) 'POT' and (d) SciPy

The original dataset's shape is estimated well by all 3 libraries. As explained earlier, the scale of the estimated datasets changes by a factor of $\lambda' = \lambda + \xi(l - m)$. While the estimated dataset from the SciPy library does produce a few outliers, the majority of data points matches the other R-based packages.

3.3 Two-Tailed Distribution

In order for the potential Python-exclusive library to be considered thorough, it needs to be able to perform the appropriate analyses on two-tailed distributions, as well. The weekly closing values of the S&P 500 index will be examined here; more specifically the weekly log-returns (WSP), which makes more sense considering continuous time discounting (Carmona, 2013).

Table 2: (a) Weekly closing values of S&P 500, (b) Weekly log-return of S&P 500

Index	Return
1	59.5
2	58.38
3	57.38
4	55.61
5	55.98

Index	Log-Return
1	-0.019
2	-0.01728
3	-0.03133
4	0.006631
5	-0.00933

The first step is to obtain a shape plot in order to determine the appropriate threshold. It is important to keep in mind that there can be two shape plots: one for the upper and one for the lower thresholds. Figure 8 shows the shape plots for the upper threshold of the WSP data.

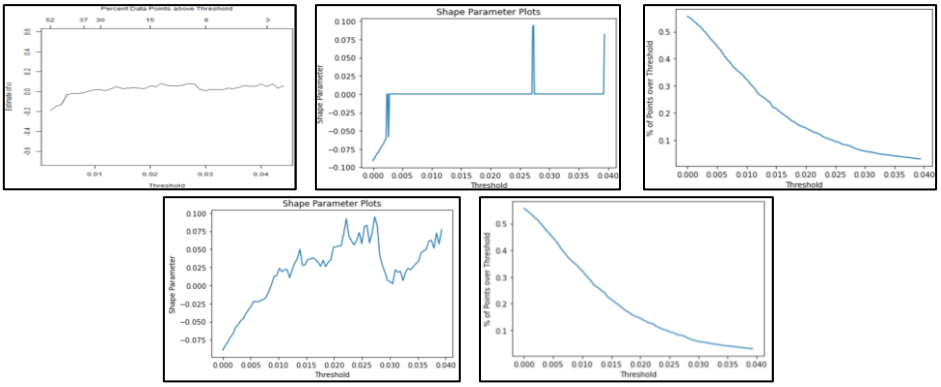

Fig. 8: Upper shape plot of WSP Data from (a) 'Rsafd', (b), 'POT' and (c) SciPy

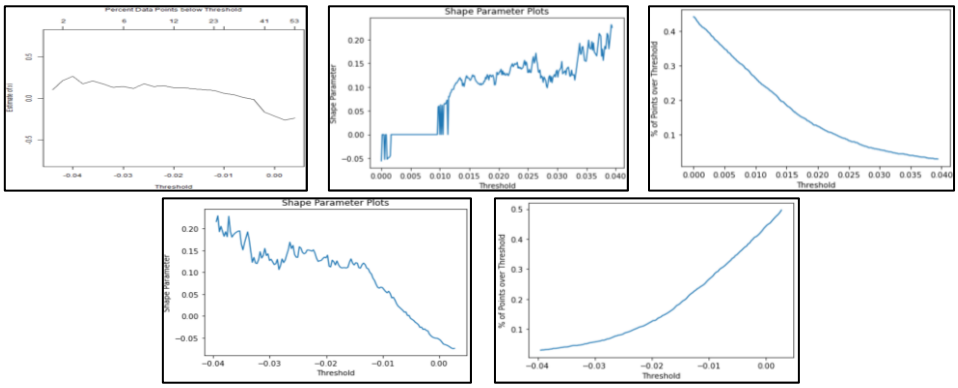

Fig. 9: Lower shape plot of WSP Data from (a) 'Rsafd', (b), 'POT' and (c) SciPy

Comparing the shape plot from 'Rsafd' and SciPy in Figure 8, it is evident that both plots are very similar. However, the shape plot from 'POT' does not return anything of value. The major setback with the 'POT' package is that when the threshold produces a shape parameter close to zero, it isn't able to plot the output correctly. It is unclear what specifically in the function is causing this. Nevertheless, based on the clearer shape plots, 0.02 will be used as the threshold level.

In Figure 9, the shape plots from SciPy and 'Rsafd' are once again showing strong similarities. The shape plot from 'POT' is also showing some similarity at higher thresholds, which in actuality is lower thresholds (since the 'POT' package only does upper thresholds, the WSP data had to be 'flipped' in order for the analysis to be performed here). Once again, when the shape parameter is close to zero, the shape plot from 'POT' isn't very informative. Based on the shape plots, -0.02 is a strong threshold value to use.

Looking at the Q-Q plots for the upper and lower tails in Figure 10, it is clear that a GPD would be a suitable fit for the respective datasets.

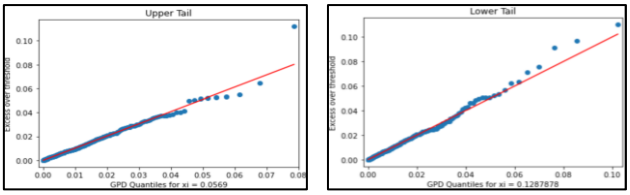

Fig. 10: Upper and lower tail Q-Q plot of WSP Data

Completing the analysis, Figure 11 displays the original WSP dataset with the estimated datasets from 'Rsafd', 'POT' and SciPy. Similar to the PCS data analysis, the SciPy estimation for the WSP data proves to be just as reliable as the others.

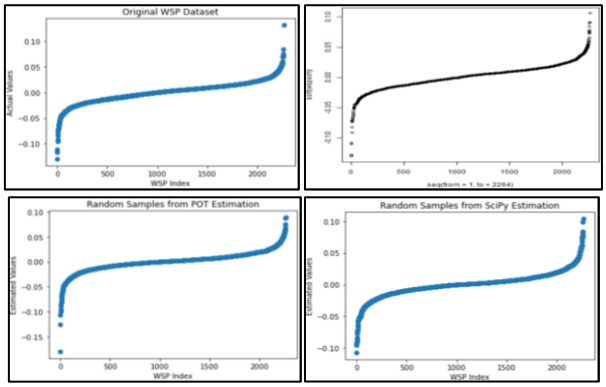

Fig. 11: (a) Original WSP data vs. Estimated data from (b) 'Rsafd', (c) 'POT' and (d) SciPy

4. Conclusion & Future Work

In the attempt to build a Python-exclusive library for performing Extreme Value Analysis, experimental functions were compared to already developed R-based functions. Based on the numerical and graphical results, there is sufficient evidence that the SciPy library in Python is capable of performing the necessary functions equivalent to its R counterparts. These functions include generating random samples from GPD parameters, creating shape plots, fitting a Generalized Pareto Distribution to a dataset (one-tailed or two-tailed) and plotting estimated data based on given parameters.

The impetus of this work is based in Time Series Analysis. Thus, further work is planned to extend Threshold Modeling testing to include this analysis. There is a number of Python libraries that are used in Time Series Analysis and this work will merge Extreme Value Analysis with time series' types such as Auto Regressive and Moving Average.

5. References

Gilleland, Eric, et al. A Software Review for Extreme Value Analysis. Springerlink.com, 20 July 2012, https://link.springer.com/content/pdf/10.1007/s10687-012-0155-0.pdf.

Kumar, V. (2021, December 11). *Python Vs R: What's Best for Machine Learning - Towards Data Science*. Medium. https://towardsdatascience.com/python-vs-r-whats-best-for-

machine-learning-93432084b480

Lemos, Iago, et al. “Thresholdmodeling: A Python Package for Modeling Excesses over a Threshold Using the Peak-Over-Threshold Method and the Generalized Pareto Distribution.” Journal of Open Source Software, no. 46, The Open Journal, Feb. 2020, p. 2013. Crossref, doi:10.21105/joss.02013.

Pinheiro, Marielle, and Richard Grotjahn. An Introduction to Extreme Value Statistics. http://grotjahn.ucdavis.edu/EWEs/extremes_primer_v9_22_15.pdf. Accessed 6 Sept. 2021.

Carmona, René. Statistical Analysis of Financial Data in R. 2nd ed., Springer Science & Business Media, 2013.

Abadi, Martín, Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., … others. (2016). Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (pp. 265–283).

Correoso, Kiko. (2019). *Welcome to scikit-extremes's documentation!ℑ*. Welcome to scikit-extremes's documentation! - skextremes documentation. Retrieved February 12, 2023, from https://scikit-extremes.readthedocs.io/en/latest/

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

6. Appendices

Appendix A – Cumulative Density Function for Excess Distribution

$$F_l(x) = \frac{1-\left(1+\xi\left(\frac{x+l-m}{\lambda}\right)\right)^{-\frac{1}{\xi}}-(1-(1+\xi\left(\frac{l-m}{\lambda}\right))^{-\frac{1}{\xi}})}{1-\left(1-\left(1+\xi\left(\frac{l-m}{\lambda}\right)\right)^{-\frac{1}{\xi}}\right)}$$

$$F_l(x) = 1 - \left[\frac{\left(1+\xi\left(\frac{x+l-m}{\lambda}\right)\right)}{\left(1+\xi\left(\frac{l-m}{\lambda}\right)\right)}\right]^{-\frac{1}{\xi}}$$

$$F_l(x) = 1 - \left[1 + \frac{\xi x}{(\lambda+\xi l-\xi m)}\right]^{-\frac{1}{\xi}}$$

$$F_l(x) = 1 - \left[1 + \xi\frac{x}{\lambda+\xi(l-m)}\right]^{-\frac{1}{\xi}} \quad (5)$$

$$F_l(x) = F_{m',\lambda',\xi'}(x)$$

Appendix B – Code Block from R package, 'Rsafd'

```
pcsest = gpd.tail(pcs, one.tail = T, upper = 4) #PCS data only has
positive values so only one tail is used: upper
print(pcsest$upper.par.ests) #scale and shape parameters
print(pcsest$n.upper.exceed) #number of points that are above the
threshold
print(1-pcsest$p.less.upper.thresh) #proportion of data points above
threshold
```

```
   lambda        xi
4.5014926 0.8027783
[1] 31
[1] 0.08136483
```

Appendix C – Importing R functionality for 'thresholdmodeling' library in Python

```
from rpy2.robjects.packages import importr
import rpy2.robjects.packages as rpackages
from rpy2.robjects.vectors import FloatVector
import rpy2

base = importr('base')
utils = importr('utils')
utils.chooseCRANmirror(ind=1)
utils.install packages('POT') #installing POT package
```

Appendix D – Output from 'POT' package

```
POT = importr('POT')
pcsfit = POT.fitgpd(FloatVector(pcs), 4, est = 'mle')
print(pcsfit[0]) #scale and shape
print(pcsfit[12]) #number of points above the threshold
print(pcsfit[13]) #proportion of points above the specified threshold
```

```
    scale     shape
4.5015133 0.8027741
[1] 31
[1] 0.08136483
```

Appendix E – Output from SciPy library

```
pcsupp1 = pcs[pcs>4]
pcsupp = pcsupp1 - 4
pcsuppfit = genpareto.fit(pcsupp, floc = 0)
print('Shape = {0}, scale = {1} and location = 0.'.format(pcsuppfit[0],
pcsuppfit[2]))
print('Number of points over threshold =', len(pcsupp)) #number of
points
print('Proportion of points over threshold =', len(pcsupp)/len(pcs))
#proportion of points
```

```
Shape = 0.8027661376803175, scale = 4.501542945509427 and location = 0.
Number of points over threshold = 31
Proportion of points over threshold = 0.08136482939632546
```