FindTheTail Documentation

Release 1.1

Frederik Strothmann

Oct 08, 2019

Contents:

Find The Tail is a procedure that is able to separate a given data set into subsets which belong to the body or the tail of an unknown underlying parent distribution. The data belonging to the tail are used to fit a generalized Pareto distribution as tail model. The tail model is evaluated further to determine the high quantiles, for risk assessment.

CHAPTER 1

Theory

In many disciplines, there is often a need to adapt a statistical model to existing data to be able to make statements regarding uncertain future outcomes. In particular, when assessing risks, an estimate of major losses must be based on events that, despite having a low probability of occurrence, have a high impact. Since the actual distribution of data – the parent distribution – is generally unknown, statisticians can fit a generalized Pareto distribution (GPD) to the data belonging to the tail.

For a very large class of parent distribution functions, the generalized Pareto distribution (GPD) can be used as a model for the tail. A certain threshold divides the parent distribution into two areas: a body and a tail region. Above the threshold, analyses are performed using the GPD as a model for the tail.

This FTT procedure (Find-The-Tail) provides a suitable and efficient method for determining the optimal threshold.

FTT uses a specially weighted MSE teststatistic AU^2 for determining the optimal value. The weights are chosen, such that the deviations (between EDF and the fitted GPD) at high quantiles are stronger penalized than at lower quantiles. This upper tail teststatistic is evaluated successive over a sorted timeseries.

Teststatistic:

$$AU_n^2 = \frac{1}{2}n - \sum_{i=1}^{n}\left[2F(x_{(i)}) + \frac{2(n-i)+1}{n}\ln\left(1 - F(x_{(i)})\right)\right]$$

With n is the sampe length, $x_{(i)}$ is the descending sorted data and F is the fitted GPD.

For further information see Reference.

CHAPTER 2

How it works

The FTT algorithm consists of the following 9 steps.

1. The loss data are sorted in descending order $x_1 \geq x_2 \geq \ldots \geq x_n$.
2. Starting with the largest losses x_1, x_2 (k = 2) the generalized Pareto distribution $\hat{F}(x)$ (= Estimation) is adapted to the data.
3. With this distribution the measure of deviation AU_k^2 is calculated.
4. Then the next smaller loss is added $(k \to k+1)$ and the parameters of the generalized Pareto distribution are re-estimated.
5. It continues with point (3) until the last loss value has been processed $(k = n)$.
6. Depending on k, a time series of the deviation measure results: AU_k^2.
7. The minimal deviation at a particular $k^* \in [1, n]$ indicates the best fit of a model for the tail to the given loss data and the associated x_{k^*} corresponds to the sought threshold u.
8. To assess the result, the confidence level is determined. (This describes how large the likelihood of a wrong decision would be, if the adapted distribution and thus the decision for the threshold were rejected)
9. For further assurance, the statistics of the standard goodness-of-fit test (CM and AD test) are evaluated.

Note:

1. FTT strictly separates the procedure for detecting the threshold and the goodness of fit to evaluate the quality of the fit.
2. There is no need for any external parameters, all information are gained form the data alone.

CHAPTER 3

Installation and Requirements

3.1 Installation

The package can be installed from pypi using

```
pip install findthetail
```

3.2 Requirements

The following Packages are required.

- numpy
- matplotlib
- scipy
- jinja2

CHAPTER 4

Input/Output

4.1 Input

The input has to be a 1-dimensional time series. The data has to in the form of a numpy array. There is no need for any preperation of the data in the form of sorting or making sure no value is given more than twice. All this prepatrions will be done by the FTT procedure.

The preperation procedure of consits of two steps:

1. Check if any value exists more than once, if so add small random number below the significant digit of the data
2. Sort the data in descending order

4.2 Output

From the information of the analysis a html report is generated which contains all important information. For a more detailed infomation on the report see the next section.

CHAPTER 5

Reports

The report consists of five parts.

- Tail Detection
- Goodness-of-Fit Tests
- Fit Values
- Risk assessment
- Plots

5.1 Tail Detection

In this part contains the size of the data set that was analyzed, the minimal value for the AU^2 as well as information on the estimated tail.

5.2 Goodness-of-Fit Tests

Goodness-of-Fit values for the Cramér-von Mises and Anderson-Darling teststatistics.

5.3 Fit Values

Fitted values of the generalized Pareto distribution (GPD).

5.4 Risk assessment

This part contains the Value-at-Risk and the Conditional-Value-at-Risk for the 0.95, 0.97, 0.99 and 0.999 quantiles.

5.5 Plots

A plot of the input data, a plot of the three teststatistics (AU^2, Cramér-von Mises and Anderson-Darling) for the different tails and a plot of the fitted tail.

CHAPTER 6

Examples

To show how FTT procedure works, we provide two examples using public domain data sets from the field of hydrology, which are commonly used for model testing.

6.1 Load FindTheTail

[1]:
```
import findthetail.ftt as ftt
import pandas as pd
import numpy as np
```

[2]:
```
# disable warning to keep the output clean
# the warning result form divergences in the logarithms in the test statistics
np.warnings.filterwarnings('ignore')
```

6.2 Read data

[3]:
```
river_nidd_data = pd.read_csv('data/river_nidd_data.csv', names=['x'])
```

6.3 Load data into model and run analyse

[4]:
```
# instanciate the Ftt (Find the tail) class with data,
# number of mc steps and a name for the report
mod = ftt.Ftt(river_nidd_data['x'].values, mc_steps=500, data_name='River Nidd')
mod.run_analysis()
```

```
Runnig fit
Running Montecarlo simulation
Calculating q and cvar
Generating plots
```

6.4 Generate report

[5]:
```
mod.report_html()
```

The report can be found here

6.5 Example River Wheaton

The data consits of 72 exceedances of flood peaks in m^3s^1 fo the Wheaton River near Carcross in Xukon Terrtory, Canda. The 72 exceedances, for the years 1958 to 1984, are rounded to one decimal place. This data set is commonly used in hydrology for model testing.

[1]:
```
import findthetail.ftt as ftt
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

[2]:
```
# disable warning to keep the output clean
# the warning result form divergences in the logarithms in the test statistics
np.warnings.filterwarnings('ignore')
```

6.6 Load data

[3]:
```
data1 = pd.read_excel('data/river_wheaton_data.xlsx', index_col='Position')
```

6.7 Load data into model and run analyse

[4]:
```
# instanciate the Ftt (Find the tail) class with data,
# number of mc steps, a name for the report
# and the number of threads to use for the montecarlo simulation
mod = ftt.Ftt(data1['x'].values, mc_steps=500, data_name='River Wheaton', threads=2)
# additional quantils can be calculated by giving the run_analysis
# function an array with specific p-values
mod.run_analysis(p_values=np.array([0.95, 0.99, 0.9999]))
```

```
Runnig fit
Running Montecarlo simulation
Calculating q and cvar
Generating plots
```

Take a look at the p-value.

[5]:
```
mod.p_value_a2
```

[5]:
```
0.678
```

The error of the p-value is given by $\sigma = \frac{1}{\sqrt{\#MC_{steps}}}$

[6]:
```
mod.mc_error
```

[6]:
```
0.03162277660168379
```

Using the `run_montecarlo_simulation` function the p-value can be calculated with a smaller error. The `threads` argument specifies the number of threads to use for the simulation. With a value of 4 for threads and 1000 mc_steps 4000 additional montecarlo points will be generated.

[7]:
```
mod.run_montecarlo_simulation(mc_steps=1000, threads=4)
```

```
        999/1000
```

[8]:
```
mod.mc_error
```

[8]:
```
0.01414213562373095
```

6.8 Generate report

[9]:
```
mod.report_html()
```

The report can be found here

CHAPTER 7

Cooperation Partners

FTT was made in cooperation from:

CHAPTER 8

Release Notes

8.1 1.1

Small changes for the plotting and some performance optimization for the montecarlo simulation.

8.1.1 Plotting

Changed from the plt interface to the fig interface for all plots.

8.1.2 Montecarlo simulation

Added multiprocessing for the montecarlo simulation. `__init__()` now takes a new keyword argument `threads`. `threads` specifies how many threads should be used for the montecalo simulation, the default value is 1. The function `run_montecarlo_simulation` also has the keyword `threads`, to specify the number us used threads. If no value is set the default value, set in the `__init__()` will be used.

8.1.3 Examples

The river wheaton example as been updated to show the usage of the multiprocessing for the montecarlo simulation.