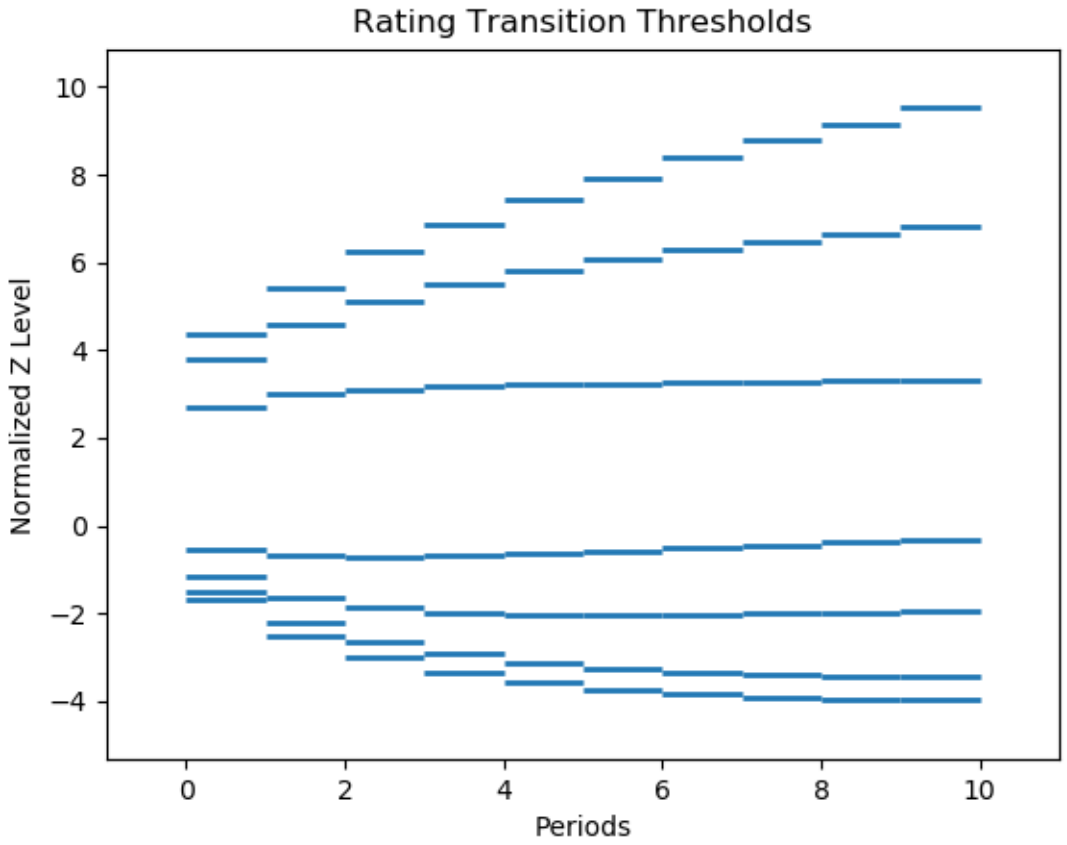
portfolioAnalytics Documentation

Release 0.3.0

Open Risk

Jun 29, 2022

CONTENTS:



NB: portfolioAnalytics is still in alpha release / active development. If you encounter issues please raise them in our github repository

CHAPTER

ONE

PORTFOLIOANALYTICS

portfolioAnalytics is a Python powered library for the calculation of semi-analytic approximations to portfolio credit models

- Author: Open Risk
- License: Apache 2.0
- Code Documentation: Read The Docs
- Mathematical Documentation: Open Risk Manual
- Training: Open Risk Academy
- Development Website: Github

1.1 Functionality

You can use portfolioAnalytics to create semi-analytic loss distributions for a variety of stylized credit portfolios. The library provides semi-analytical functions useful for testing the accuracy of credit portfolio simulation models. The basic formulas are reasonably simple and well known: They underpin the calculation of RWA (risk weighted assets), and in turn required capital, thus ensuring stability for the entire banking systems worldwide. You can also use the library to estimate transition thresholds for stochastic processes

NB: portfolioAnalytics is still in active development. If you encounter issues please raise them in our github repository

1.2 Vasicek Portfolio Models Library

Dependencies: scipy, sympy

1.3 Portfolio Model Examples

Check the jupyter notebooks and python scripts

1.4 Current Functions

- vasicek_base
- vasicek_base_el
- vasicek_base_ul
- vasicek_lim
- vasicek_lim_el
- vasicek_lim_ul
- vasicek_lim_q

The Vasicek Base family produces finite pool loss probabilities and measures (EL, UL)

The Vasicek Lim family produces asymptotic pool loss probabities and measures (EL, UL, Quantile)

1.5 Limitations

The portfolioAnalytics library provides a range of powerful modelling functionalities that are are of relevance in real credit portfolio management activities. Yet achieving the tractability and usability of a semi-analytic calculation suite is not without some tradeoffs. Several simplifications are made (extensively documented in the Mathematical Documentation). Those simplifications imply that when using the portfolioAnalytics models to assess the risk in actual portfolios it is important to assess

CHAPTER

TWO

INSTALLATION

You can install and use the portfolioAnalytics package in any system that supports the Scipy ecosystem of tools

2.1 Dependencies

- portfolioAnalytics requires Python 3
- the thresholds module depends on the Open Risk transitionMatrix and correlationMatrix libraries
- It depends on numerical and data processing Python libraries (Numpy, Scipy, Pandas)
- The Visualization API depends on Matplotlib
- The precise dependencies are listed in the requirements.txt file.
- portfolioAnalytics may work with earlier versions of these packages but this has not been tested

2.2 From PyPi

```
pip3 install pandas
pip3 install matplotlib
pip3 install portfolioAnalytics
```

2.3 From sources

Download the sources to your preferred directory:

```
git clone https://github.com/open-risk/portfolioAnalytics
```

2.4 Using virtualenv

It is advisable to install the package in a virtualenv so as not to interfere with your system's python distribution

```
virtualenv -p python3 tm_test
source tm_test/bin/activate
```

If you do not have pandas already installed make sure you install it first (will also install numpy)

```
pip3 install pandas
pip3 install matplotlib
pip3 install -r requirements.txt
```

Finally issue the install command and you are ready to go!

```
python3 setup.py install
```

2.5 File structure

The distribution has the following structure:

portfolioAnalytics The library source code
- estimators Estimator methods (TODO)
- utils Helper classes and methods
- thresholds Algorithms for calibrating AR(n) process thresholds to input transition rates
- vasicek Collection of portfolio analytic solutions
- creditmetrics Analytic calculation of variance for credit metrics style models

examples Usage examples
datasets Contains a variety of datasets useful for getting started with portfolioAnalytics
tests Testing suite

2.6 Testing Framework

It is a good idea to run the test-suite. Before you get started:

- Adjust the source directory path in portfolioAnalytics/__init__ and then issue the following in at the root of the distribution
- Unzip the data files in the datasets directory

```
python3 test.py
```

CHAPTER

THREE

GETTING STARTED

Check the Examples pages in this documentation

Look at the examples directory for a variety of typical workflows.

For more in depth study, the Open Risk Academy has courses elaborating on the use of the library

- Analysis of Credit Migration using Python portfolioAnalytics:

CHAPTER

FOUR

EXAMPLES

The portfolioAnalytics packages offers quite a lot of functionality. Here we break down some of the main workflows for those getting started. The examples directory includes python scripts and jupyter notebooks to help you get started

- Generating loss distributions analytically
- Estimating thresholds given a multi-period transition matrix set

4.1 Python Scripts

Located in examples/python (For testing purposes all examples can be run using the run_examples.py script located in the root directory)

4.1.1 Examples of calculating variance

Calculating the default rate volatility (variance) of a portfolio requires two inputs: * a set of suitable portfolio data * an asset correlation matrix

4.1.2 Examples of calculating portolio models

The examples directory includes python scripts and jupyter notebooks to help you get started

- Generating loss distributions analytically
- Estimating thresholds given a multi-period transition matrix set

Located in examples/python (For testing purposes all examples can be run using the run_examples.py script located in the root directory)

Portfolio Models

- portolio_model.py

Using the portfolio model library to calculate default probabilities in finite portfolio with N credits

4.1.3 Examples of calculation thresholds

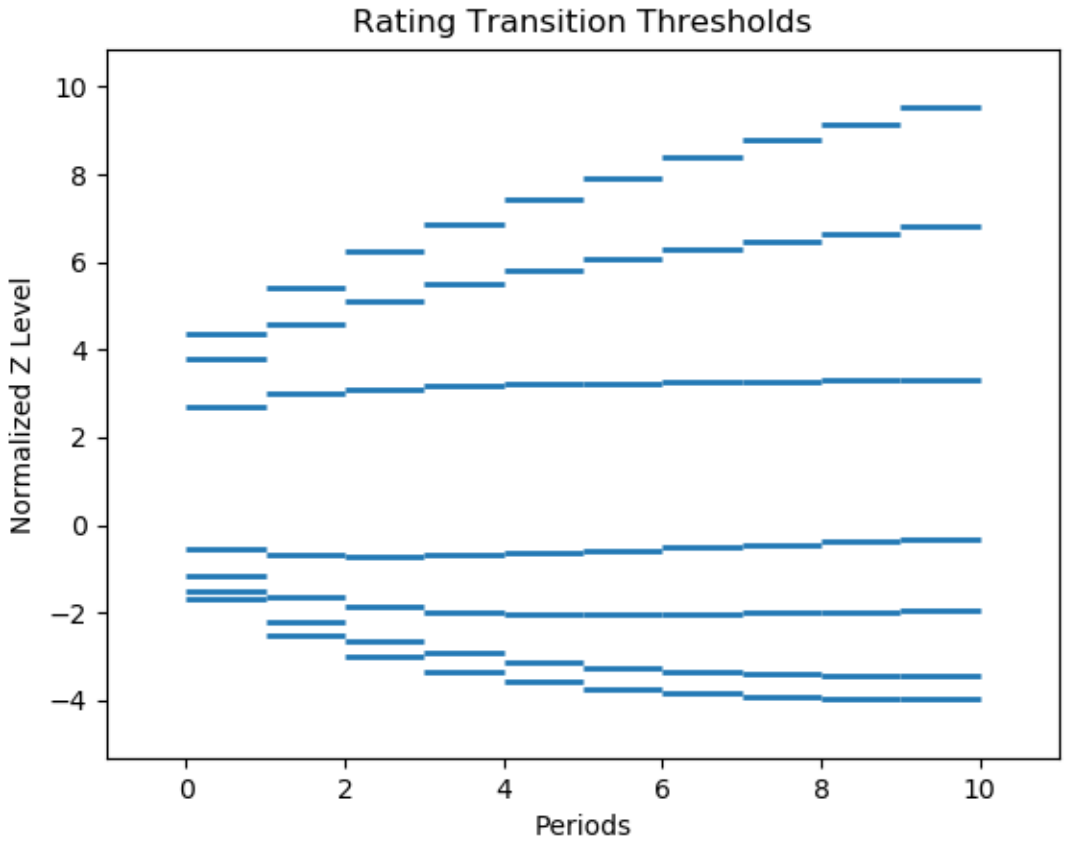
Estimating thresholds given a multi-period transition matrix set. Example of using portfolioAnalytics to compute multi-period transition thresholds compatible with a given transition matrix.

Rating Transition Thresholds

- calculate_thresholds.py
- validated_thresholds.py
- visualize_thresholds.py

  Validate computed thresholds.

The mathematical framework documented in Multi-Period Transition Thresholds



4.1.4 Examples of visualizing thresholds

The examples directory includes python scripts and jupyter notebooks to help you get started

- Generating loss distributions analytically
- Estimating thresholds given a multi-period transition matrix set

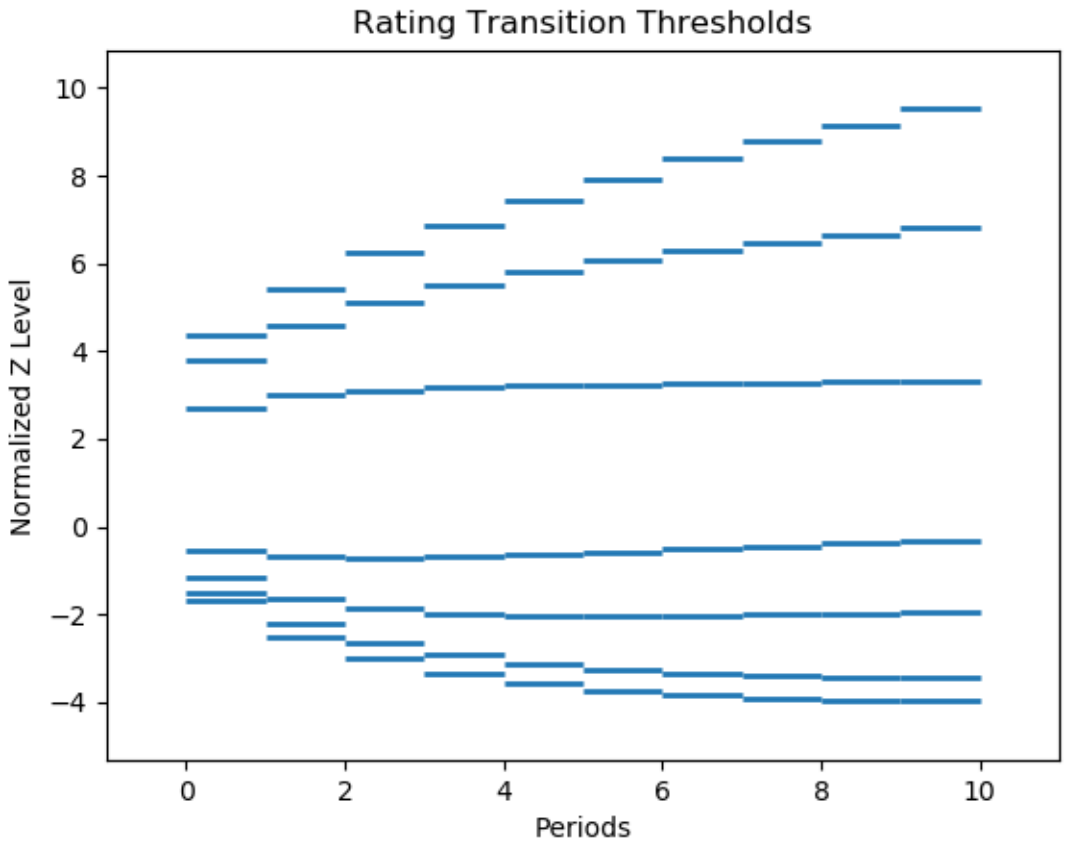
Located in examples/python (For testing purposes all examples can be run using the run_examples.py script located in the root directory)

Visualizing Rating Transition Thresholds

- visualize_thresholds.py

Example of using portfolioAnalytics to compute multi-period transition thresholds compatible with a given transition matrix. Validate computed thresholds.

The mathematical framework documented in Multi-Period Transition Thresholds



4.1.5 Examples of validating transition thresholds

The examples directory includes python scripts and jupyter notebooks to help you get started

- Generating loss distributions analytically
- Estimating thresholds given a multi-period transition matrix set

Located in examples/python (For testing purposes all examples can be run using the run_examples.py script located in the root directory)

Validating Transition Thresholds

- validate_thresholds.py

Example of using portfolioAnalytics to compute multi-period transition thresholds compatible with a given transition matrix. Validate computed thresholds.

4.2 Jupyter Notebooks

- Examples.ipynb
- Portfolio_Examples.ipynb

4.3 Open Risk Academy Scripts

TODO

CHAPTER

FIVE

API

The portfolioAnalytics package structure and API.

Warning: The library is still being expanded / refactored. Significant structure and API changes are likely.

5.1 portfolioAnalytics Package

The core module

This module is part of the portfolioAnalytics package.

5.2 portfolioAnalytics Subpackages

5.2.1 portfolioAnalytics.vasicek subpackage

Functions

The Vasicek subpackage implements currently the following:

- vasicek_base implements a finite homogeneous pool
- vasicek_base_el implements the expected loss for the vasicek_base case
- vasicek_base_ul implements the standard deviation for the vasicek_base case
- vasicek_lim implements the limiting case for large N
- vasicek_lim_el implements the expected loss for the vasicek_lim case
- vasicek_lim_ul implements the standard deviation for the vasicek_lim case
- vasicek_lim_q implements the quantile for the vasicek_lim case

Vasicek Base Distribution

Vasicek Base Discrete distribution.

param N
The number of entities in the portfolio

param k
The number of defaults

param p
The probability of default (uniform across the portfolio)

param rho
The asset correlation parameter

return
The probability of k defaults

Vasicek Base Distribution Expected Loss

Expected Loss for the Vasicek Base distribution.

param N
The number of entities in the portfolio

param p
The probability of default

param rho
The asset correlation (not needed here)

return
The average default rate / loss

Vasicek Base Distribution Unexpected Loss

Unexpected Loss (Standard Deviation) for the Vasicek Base distribution.

param N
The number of entities in the portfolio

param p
The probability of default

param rho
The asset correlation (not needed here)

return
The default rate volatility (UL)

Vasicek Limit Distribution

The Large-N limit of the Vasicek Distribution.

param theta
The target default rate

param p
The probability of default

param rho
The asset correlation

return
Cumulative probability

Vasicek Limit Distribution Expected Loss

The expected loss of the large n limit of the Vasicek distribution.

param p
The probability of default

param rho
The asset correlation (not needed)

return
The expected default rate

Vasicek Limit Distribution Unexpected Loss

The unexpected loss of the large n limit of the Vasicek distribution.

param p
The probability of default

param rho
The asset correlation

return
The default rate volatility

Vasicek Limit Distribution Quantile Loss

The quantile of the large-n Limit of the Vasicek distribution.

param alpha
The desired quantile

param p
The probability of default

param rho
The asset correlation

return
The default rate at that confidence level

5.2.2 portfolioAnalytics.creditmetrics subpackage

CreditMetrics Style Functions

Inverse normal function

Inverse normal function.

Variance calculation

Variance calculation.

Expected Loss Calculation

Credit Metrics Expected Loss Calculation.

Loss Volatility (Unexpected Loss)

Credit Metrics Loss Volatility (Standard Deviation of Loss)

5.2.3 portfolioAnalytics.estimators subpackage

Todo: This subpackage is still to be written (planned for 0.4)

5.2.4 portfolioAnalytics.thresholds subpackage

Warning: The Thresholds subpackage has a dependency on transitionMatrix that need to be installed in the same python environment

portfolioAnalytics.thresholds.model module

This module is part of the portfolioAnalytics package.

class portfolioAnalytics.thresholds.model.**ConditionalTransitionMatrix**(**args: Any*, ***kwargs: Any*)

Bases: TransitionMatrixSet

The ConditionalTransitionMatrix object stores a family of TransitionMatrix objects as a time ordered list.

Its main functionality is to allow conditioning (stressing) the values in accordance with a predefined model

fit(*AR_Model*, *Scenario*, *rho*, *ri*)

Calculate conditional transition rates given thresholds and stochastic model

plot_densities(*period=None*, *state=0*)

Plot densities.

print_matrix(*format_type='Standard'*, *accuracy=2*, *state=None*)

Pretty print a threshold matrix set

Parameters

- **format_type** (*str*) – formating options (Standard, Percent)
- **accuracy** (*int*) – number of decimals to display

class portfolioAnalytics.thresholds.model.**ThresholdSet**(*ratings=None*, *periods=None*, *TMSet=None*, *json_file=None*)

Bases: object

The Threshold set object stores a multiperiod migration/default threshold structure as a numpy array.

Todo: Separate integration method from transition data

fit(*AR_Model*, *ri*, *dt=1.0*)

Fit Thresholds given autoregressive model and transition matrix given the initial state ri.

Note: The threshold corresponding to the starting rating is set by convention to NaN. The threshold corresponding to a defaulted state is set by convention to - Infinity. These values are stored in memory as numpy NaN and Infinity value respectively. They are serialized as strings "nan" and "-inf" respectively.

from_json(*json_file*)

Read from JSON.

plot(*rating*)

Plot.

print(*format_type='Standard'*, *accuracy=2*)

Pretty print a threshold matrix set

Parameters

- **format_type** (*str*) – formating options (Standard, Percent)
- **accuracy** (*int*) – number of decimals to display

to_json(*json_file=None*, *accuracy=5*)

Convert to JSON.

validate(*AR_Model*)

Validate calculated Thresholds given autoregressive model

The comparison is accomplished by producing the implied transition matrix and setting against the input set

Todo: Automate the comparison when a new set is produced

portfolioAnalytics.thresholds.model.**integrate_f**(*ff*, *x*, *an*, *dx*, *dt*, *mu*, *phi_1*)

Integrate F.

portfolioAnalytics.thresholds.model.**integrate_g**(*ff*, *x*, *an*, *dx*, *dt*, *mu*, *phi_1*)

Integrate G.

ThresholdSet

The Threshold set object stores a multiperiod migration/default threshold structure as a numpy array.

Todo: Separate integration method from transition data

Integrate G

Integrate G.

Integrate F

Integrate F.

5.2.5 portfolioAnalytics.utils subpackage

portfolioAnalytics.utils contents

This module contains various helper classes and functions that do not fit into any of the main modules of the library

portfolioAnalytics.utils Submodules

portfolioAnalytics.utils.converters module

Converter utilities to help switch between various formats.

`portfolioAnalytics.utils.converters.`**`datetime_to_float`**(*dataframe*)

Datetime to float. f .. _Datetime_to_float:

Converts dates from string format to the canonical float format

Parameters
dataframe – Pandas dataframe with dates in string format

Returns
Pandas dataframe with dates in float format

Return type
object

Note: The date string must be recognizable by the pandas to_datetime function.

portfolioAnalytics.utils.dataset_portfolio module

This module provides simple functionality for holding portfolio data for calculation purposes.

- *Portfolio* implements a simple portfolio data container

class portfolioAnalytics.utils.portfolio.**Portfolio**(*psize=0*, *rating=[]*, *exposure=[]*, *factor=[]*)

Bases: object

The Portfolio object implements a simple portfolio data structure. See loan tape for more general structures.

loadjson(*data*)

Load portfolio data from JSON object.

The data format for the input json object is a list of dictionaries as follows

```
[{"ID":"1","PD":"0.015","EAD":"40","FACTOR":0},
  ...
 {"ID":"2","PD":"0.286","EAD":"20","FACTOR":0}]
```

preprocess_portfolio()

Produce some portfolio statistics like total number of entities and exposure weighted average probability of default :return:

portfolioAnalytics.utils.bivariatenormal module

Implementation of the bivariate normal function.

portfolioAnalytics.utils.bivariatenormal.**BivariateNormalDensity**(*a*, *b*, *rho*)

Bivariate Normal Density.

portfolioAnalytics.utils.bivariatenormal.**BivariateNormalDistribution**(*a*, *b*, *rho*)

Bivariate Normal Distribution.

based on Z. Drezner, “Computation of the bivariate normal integral”, Mathematics of Computation 32, pp. 277-279, 1978. uses 8-point Gaussian quadrature

portfolioAnalytics.utils.bivariatenormal.**N**(*x*)

Standard Normal.

portfolioAnalytics.utils.bivariatenormal.**Phi_Sum**(*a*, *b*, *rho*)

Phi Sum Helper Function.

portfolioAnalytics.utils.bivariatenormal.**rhoc**(*a*, *b*, *rho*)

Rho_c Helper Function.

CHAPTER

SIX

TESTING

Testing portfolioAnalytics has two major components:

- normal code testing aiming to certify the correctness of code execution
- algorithm testing aiming to validate the correctness of algorithmic implementation

Note: In general algorithmic testing is not as precise as code testing and may be more subject to uncertainties such as numerical accuracy.

6.1 Running all the examples

Running all the examples is a quick way to check that everything is installed properly, all paths are defined etc. At the root of the distribution:

```
python3 run_examples.py
```

Warning: The script might generate a number of files / images at random places within the distribution

6.2 Test Suite

The testing framework is based on unittest.

Then run all tests

```
python3 test.py
```

For an individual test:

```
pytest tests/test_TESTNAME.py
```

CHAPTER

SEVEN

ROADMAP

portfolioAnalytics aims to become the most intuitive and versatile tool to analyse discrete transition data. This roadmap lays out the general upcoming steps in this journey. The Todo List has some more concrete tasks that may have been already raised as issue son github.

7.1 0.3.X

The 0.3.X family of releases will focus on rounding out a number of functionalities already introduced

- Further documenting the existing functionality
- Further tests, of both code and algorithms

7.2 0.4.X

7.2.1 Provide additional functions for default rate distributions

- Complete the universe of analytic solutions for Gaussian models * Include some interesting special cases (e.g., large pool + single exposure) * Include some tractable inhomogeneous problems
- Calculate more statistical moments (e.g., skew, kurtosis)
- Expand to non-Gaussian distributions

7.2.2 Expand to loss distributions that include recovery risk

7.2.3 Make a more robust implementation

The functions should ultimately be coded to a high standard of robustness: * input validation * exception handling * controlled accuracy

7.2.4 Introduce testing framework

CHAPTER

EIGHT

TODO LIST

Feature requests, bug reports and any other issues are welcome to log at the Github Repository

CHAPTER

NINE

CHANGELOG

PLEASE NOTE THAT THE API IS STILL VERY UNSTABLE AS MORE USE CASES / FEATURES ARE ADDED REGULARLY

9.1 v0.3.2 (XX-XX-2022)

- release on PyPI

9.2 v0.3.1 (23-03-2020)

- Improved the documentation of the vasicek module
- Fixed the broken API of the variance examples

9.3 v0.3.0 (17-06-2019)

- Expanding sphinx documentation to cover all classes function

9.4 v0.2.1 (04-04-2019)

- Including credit metrics style variance calculation

9.5 v0.2.0 (29-03-2019)

- Refactoring to include threshold model functionality (formerly with transitionMatrix library)

9.6 v0.1.1 (11-07-2017)

- Training: Notebook examples

9.7 v0.1.0 (16-04-2017)

- First public release of the package

CHAPTER

TEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

R

T

V