Open Actuarial Documentation

Release

Robert Spaul

Jan 19, 2018

Contents

1 Introduction **1**
- 1.1 Getting Started . . . 1
- 1.2 Discussion and Contributing . . . 1
- 1.3 Areas for focus . . . 1

2 Discussion and Contributing **5**
- 2.1 Contributing to this wiki . . . 5
- 2.2 Editing the site . . . 5

3 Core statistical tools **9**

4 Getting started with Python **11**
- 4.1 Why Python? . . . 11
- 4.2 Getting Started . . . 11
- 4.3 Getting help . . . 12
- 4.4 Installing Python . . . 12
- 4.5 Learning the basics . . . 12
- 4.6 Python for Data Analysis . . . 13

5 Data **15**
- 5.1 General Insurance/Property & Casualty . . . 15

6 Domain specific tools and models **17**
- 6.1 General Insurance/Property & Casualty . . . 17
- 6.2 Life . . . 17

7 Introduction **19**
- 7.1 Getting Started . . . 19
- 7.2 Discussion and Contributing . . . 19
- 7.3 Areas for focus . . . 19

CHAPTER 1

Introduction

Open Actuarial is a group for the promotion of open approaches to actuarial problems.

This organisation aims to:

- Identify the main areas where open tools can benefit actuarial analyses
- Support and publicise existing open tools that can help actuarial analysis
- Aid in the development of new tools, especially where gaps in the available toolset exist
- Educate the actuarial community on the availability, benefits and occasional downsides of open tools

1.1 Getting Started

Resources on how to get started with programming.

- *Getting started with Python*
- Getting started with R (TODO)

1.2 Discussion and Contributing

- *Help improve this site*

1.3 Areas for focus

Below are some of the areas where the actuarial community can benefit from open tools:

1.3.1 Core statistical tools

- core probabilistic functions ala Actuar including blended distributions e.g. mixed exponential, mixed pareto, etc
- curve fitting
- dependency modelling and parameterisation
- data manipulation and charting tools

1.3.2 Domain specific data dictionaries and datasets (real and dummy)

- Standardised data dictionaries/glossaries
- Tools for the transformation of data from formats typically used in administration systems to those suitable for analysis
- Real datasets
- Dummy datasets (useful where either real data is unavailable for commercial reasons and for ensuring that actuarial analyses work appropriately)

1.3.3 Statistical analysis

- tools for generalised linear regression, mixed models, survival analysis, bayesian analysis, machine learning

1.3.4 Reporting tools

- Web based, pdf-able reports (to improve on Excel based reporting)
- web apps
- Notebooks
- Web based presentations

1.3.5 Development and production environments

Applying DevOps principles to Actuarial work:

- Collaboration and Version control (Git, TFS, Mecurical, SVN)
- Testing (unit tests, speed tests, test-driven development)
- Continuous integration
- Push button deployments
- Development, Testing and Production environments (virtualisation and containers)
- 'Big Data' architectures

1.3.6 Domain specific tools and models

- General Insurance/Property & Casualty
- Life Insurance
- Investments

CHAPTER 2

Discussion and Contributing

2.1 Contributing to this wiki

To add to this wiki either:

1. Add an 'issue' explaining your suggestion to the Issue Tracker of the site's repository on GitHub. Someone else can then pick up the suggestion and incorporate it into the site. (very simple to do, good for anyone not yet comfortable with GitHub or Sphinx)
2. *Edit the site* yourself by submitting a pull-request (requires knowledge of GitHub and Sphinx but significantly reduces the workload for the Core Team)

2.2 Editing the site

This site is built using Sphinx, a Python package that creates static websites such as wikis or project documentation. The site is made up of a collection of rst and markdown text files with Sphinx turning the text files into a full website.

The site's code is version controlled using Git and hosted on GitHub.

A Conda virtual environment is used manage the build environment and reduce the chance of build errors.

This site follows the standard steps for contributing to a GitHub based open-source project.

2.2.1 First time setup

1. Fork the site's repository to your personal GitHub account.
2. Clone your fork to your local machine:

```
cd desired_folder # change directory to the location where you will keep the␣
↪project

# where user_name is your github username
```

```
git clone git@github.com:user_name/open-actuarial-site.git # if using ssh or
git clone https://github.com/user_name/open-actuarial-site.git # if using https
```

3. Add the main project as upstream (allows you to pull in changes made by others):

```
# add upstream (location of main repository)
git remote add upstream  git@github.com:open-actuarial/open-actuarial-site.git #␣
↪if using ssh
git remote add upstream https://github.com/open-actuarial/open-actuarial-site.git
↪# if using https
```

4. Move into project directory:

```
cd OpenActuarialSite # move into the site's working directory
```

5. Set up the environment:

```
pip install conda # if you do not have conda installed already (you will do if␣
↪you have installed Anaconda)
conda env create -f wiki_env.yml # creates a virtual environment using the yml␣
↪file for the list of packages

# start the virtual environment
activate wiki # for windows
source activate wiki # for MacOS or Linux
```

2.2.2 Making changes

1. Fetch upstream updates from main GitHub repository (in case anyone else has made changes since you last worked on the project):

```
git pull upstream master
```

2. Create a new branch for your suggestion:

```
git branch branch_name # where branch_name is the name of the feature or bug
```

3. Make the changes by adding new pages to the wiki (as rst or md files), editing existing ones or a combination of both

4. Test that the change has the desired effect by rebuilding the wiki using Make HTML:

```
make html # from windows CMD not Git Bash
```

5. Commit the change to the feature branch:

```
git add file_name # to stage a change to a single file, or
git add . # if staging changes to all files
git commit -m "commit_message"
```

6. Push the changes to your forked GitHub repository:

```
git pull upstream master # (to check for changes by others)
git push origin branch_name
```

7. Submit a pull request on main repository

8. The change will be reviewed by the Core Team - if revisions are requested further commits may be required
9. Pull request is accepted or rejected

2.2.3 If using two-factor authentication (2FA)

On windows, GitHub for Windows makes it a lot easier to use 2FA. If using the command line then you will use ssh. See Settings on your GitHub account to set up ssh keys.

Once ssh keys are set up:

1. Test ssh connection (from GitHub articles):

```
ssh -T git@github.com
# Attempts to ssh to GitHub
```

2. Launch ssh agent:

```
eval $(ssh-agent -s)
```

3. Add private key (if not already added):

```
ssh-add ~/.ssh/id_rsa
# assumes key is already set up
```

CHAPTER 3

Core statistical tools

1. actuar (R)

   https://github.com/cran/actuar

2. copula (R)

CHAPTER 4

Getting started with Python

4.1 Why Python?

Python is one of the most popular programming languages around due to its:

- Simple and easy to read syntax making it quick to learn
- Huge range of packages allowing it be used for a huge range of applications
- Active open community of developers
- Use behind many of the most popular websites in the world (Google, YouTube, Instagram, Dropbox, Reddit)

It is widely regarded as the main language for machine learning and is comparable to R for other forms of data analysis.

4.2 Getting Started

Each person has their own preferences when learning something new so the route advised below may not suit everyone.

1. *Install* it on your machine
2. Start by *learning the basics* of the language and the syntax.
3. Familiarise yourself with the *Python packages* that will be useful for actuarial analyses
4. Force yourself to use it for some real work, something simple that you may normally have used Excel for
5. Learn some intermediate Python (e.g. generators, docstrings, decorators, etc) so that you can implement projects more efficiently
6. Read others code to understand how to get the most out of the language

There are plenty of options for in-person training courses if you feel that

Following the steps above should get you well on your way but don't stop there:

1. Seek out other Python users for tips

2. Watch *videos*
3. Look at projects available online for ideas
4. Join a local community (e.g PyData) to find Python users in your area and learn from them
5. Look at Kaggle competition entries and blogs
6. Follow leading Python data scientists on Twitter for ideas

4.3 Getting help

You will get stuck as with any new tool. Inevitably there will be someone who has experienced your issue before so persevere and google your way through it:

1. Search the web for answers - StackOverFlow is an amazing website for questions and dedicated volunteers to help solve issues. Usually someone will have already encountered your problem and someone else will have pointed out a solution.
2. Read the documentation - at least read the getting started guides for which ever area you are stuck on.
3. Don't go it alone - persuade someone else in your organisation to learn with you
4. Find someone in your organisation who can help

4.4 Installing Python

Windows does not come with Python by default. Linux and MacOS users already have Python on their machines. However the version installed may not be suited for scientific analysis.

The easiest way to get started is to install Anaconda, which at its core is a Python installation customised for scientific use with lots of bells and whistles that will be useful along the way. It isn't a single program which may confuse newcomers but a collection of tools which are most easily launched through the Anaconda Navigator (one of the programs installed).

Once Anaconda is installed you have several ways of actually writing code and running it:

1. Launch a Jupyter Notebook where code is written in cells and the output displayed below the cell
2. Launch Spyder for an Interactive Development Environment (IDE) similar to RStudio or Visual Studio.
3. Write code in any other compatible text editor or IDE (PyCharm, JupyterLab, Eclipse, Visual Studio with Python for Visual Studio, etc)

4.5 Learning the basics

There are lots of websites offering free online courses covering the basics most of which allow you to run code in the browser so that you don't need to even worry about installing the software. Here are just a few:

- Codecademy - Learn Python
- Datacamp - Intro to Python for Data Science
- Udacity - Programming Foundations with Python

Look for a course that covers topics such as Lists, Dicts, Functions, Loops and Classes.

4.6 Python for Data Analysis

A lot can be done with just core Python but its real power comes from making use of all the optional Python packages (think libraries in R or toolboxes in MatLab) that will make your life much easier and allow you to concentrate on your analyses. The following packages are potentially useful across a wide range of actuarial projects:

- Numpy - a foundational package that most other scientific packages in Python utilise. It introduces an array data type and a whole set of functions and classes for working with them. This package makes array calculations to run extremely quickly as most of the underlying code is in C and makes use of Intel's math kernels. It is unlikely you will need to use Numpy directly, at least initially, as other packages are more useful but you should make note of the basics.
- Pandas - for working with data tables (importing, editing, grouping, pivoting, merging, joining, reshaping, time series). This makes working with actuarial data very straightforward and this fantastic package is extremely well documented with a 10 minutes in Pandas section which gets you up and running quickly.
- SciPy - mathematics package covering integration, optimization, interpolation, linear algebra and statistics. Very useful especially the statistics classes and functions.
- Matplotlib - the standard plotting library for creating a wide range of plots. The documentation is a needs some work but it can do anything you need it to if you persevere.

Other packages to note:

- Bokeh - relatively new plotting library that creates interactive plots suited to display in the browser
- Seaborn - a plotting library geared towards statistics
- PyMC3 - Bayesian analysis (also consider PyStan, PyTorch)
- Lifelines - survival analysis
- Statsmodels - statistical models (tests, regression, time series)
- scikit-learn - - machine learning algorithms including neural networks

There are many online courses that focus on Python for data science, for example:

- Udacity - Intro to Data Analysis
- edX - Python for Data Science
- Coursera - Introduction to Data Science in Python

There are several good books that lay the foundations:

- Python for Data Analysis, Wes McKinney - written by the main author of Pandas
- Python Data Science Handbook, Jake VanderPlas - written by a widely regarded data scientist, the book is freely available online on GitHub (consider purchasing if you like it)

Video series:

- pyvideo
- PyData channel on YouTube
- Data School

CHAPTER 5

Data

5.1 General Insurance/Property & Casualty

1. Loss reserving data from NAIC schedule P
2. R Actuarial Workshop datasets

   "In order to facilitate R instruction for actuaries, we have organized several sets of publicly available data of interest to non-life actuaries. In addition, we suggest a set of packages, which most practicing actuaries will use routinely. Finally, there is an R markdown skeleton for basic reserve analysis."

3. CAS with R Datasets (R)

   A collection of datasets, originally for the book 'Computational Actuarial Science with R' edited by Arthur Charpentier. Now, the package contains a large variety of actuarial datasets.

4. Accidents on roads of Victoria, Australia (Kaggle competition)
5. French auto claims data (Kaggle competition)

CHAPTER 6

Domain specific tools and models

6.1 General Insurance/Property & Casualty

1. ChainLadder, R

   "ChainLadder is an R package providing methods and models which are typically used in insurance claims reserving"

2. Interface for ChainLadder using Shiny, R
3. Stochastic Loss Reserving Models using MCMC (R)

   Code for https://www.casact.org/pubs/monographs/papers/01-Meyers.PDF

6.2 Life

1. Pyliferisk (Python)

   Pyliferisk is a python library for life actuarial calculations, simple, powerful and easy-to-use.

2. Life contingencies (R)
3. demography (R)

   The R package demography provides functions for demographic analysis including: lifetable calculations; Lee-Carter modelling; functional data analysis of mortality rates, fertility rates, net migration numbers; and stochastic population forecasting.

CHAPTER 7

Introduction

Open Actuarial is a group for the promotion of open approaches to actuarial problems.

This organisation aims to:

- Identify the main areas where open tools can benefit actuarial analyses
- Support and publicise existing open tools that can help actuarial analysis
- Aid in the development of new tools, especially where gaps in the available toolset exist
- Educate the actuarial community on the availability, benefits and occasional downsides of open tools

7.1 Getting Started

Resources on how to get started with programming.

- *Getting started with Python*
- Getting started with R (TODO)

7.2 Discussion and Contributing

- *Help improve this site*

7.3 Areas for focus

Below are some of the areas where the actuarial community can benefit from open tools:

7.3.1 Core statistical tools

- core probabilistic functions ala Actuar including blended distributions e.g. mixed exponential, mixed pareto, etc
- curve fitting
- dependency modelling and parameterisation
- data manipulation and charting tools

7.3.2 Domain specific data dictionaries and datasets (real and dummy)

- Standardised data dictionaries/glossaries
- Tools for the transformation of data from formats typically used in administration systems to those suitable for analysis
- Real datasets
- Dummy datasets (useful where either real data is unavailable for commercial reasons and for ensuring that actuarial analyses work appropriately)

7.3.3 Statistical analysis

- tools for generalised linear regression, mixed models, survival analysis, bayesian analysis, machine learning

7.3.4 Reporting tools

- Web based, pdf-able reports (to improve on Excel based reporting)
- web apps
- Notebooks
- Web based presentations

7.3.5 Development and production environments

Applying DevOps principles to Actuarial work:

- Collaboration and Version control (Git, TFS, Mecurical, SVN)
- Testing (unit tests, speed tests, test-driven development)
- Continuous integration
- Push button deployments
- Development, Testing and Production environments (virtualisation and containers)
- 'Big Data' architectures

7.3.6 Domain specific tools and models

- General Insurance/Property & Casualty
- Life Insurance
- Investments