Spreadsheets vs. Code?

Patrick Yu
Kevin Dalton
Liam McGrath

ANNUAL MEETING

1

Introductions

2

Speakers

- Patrick Yu, FCAS, FSA, MAAA, Senior Director, Actuarial & Analytics, Kaiser Permanente
- Kevin Dalton, Data Scientist, American Modern Insurance Group
- Liam McGrath, ACAS, MAAA, Data Scientist, WTW

3

Agenda

- Why you should use code instead of Excel
- R vs. Python
- How to Setup R/Python
- How to Learn and Use R/Python
- When to use Excel instead of code
- How to build a Tech Stack for Actuaries
- Examples/Q&A

4

Purpose & Scope

5

Goals & Audience

Goals

- ☒ Encourage programming
- ☒ Why & How to get started
- ☒ Practical introduction for actuarial Excel users
- ☐ Turn novices into R/Python users

Target Audience

- ☒ Excel users with repetitive tasks and/or painful data
- ☒ People with basic programming skills (or are willing to learn on their own)
- ☐ Experienced programmers (but comments welcome!)
- ☐ Deep dive into specific problems

6

Why Code Instead of Excel

7

Automating Repetitive Tasks Saves Time

- Most actuarial tasks recur enough times to merit automation
  - Data-heavy, high-touch projects save the most time
  - Quick projects can also benefit
  - Significant time also saved in the review process
- For long tasks, can "set it and forget it"
- Improve quality of life at work
- Flywheel effect – helps you program even better
  - Gives you real-world practice
  - Frees up more time to learn and experiment

8

Automation Reduces Spreadsheet Errors

- Spreadsheet errors are rampant
  - 90%+ of spreadsheets contain errors, with ≈1% cell error rate [1]
  - Not all actuaries follow spreadsheet best practices
  - Not all actuarial teams have time/staff to audit spreadsheets
- Automation reduces error in multiple ways
  - Eliminating manual inputs
  - Eliminating manual formula edits
  - Can build automatic checks and reconciliations

9

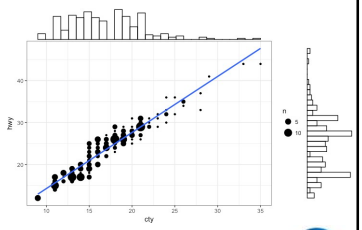
Programming Improves Documentation

- Staff turnover + Poorly-documented spreadsheets = Headaches
- Programming workflows can significantly improve documentation
  - Each action is recorded in the code
  - Code executes in repeatable order
  - Well-written comments can further improve
- Caveat: poorly-written or poorly-commented code can still cause headaches

10

Programming Improves Visualizations

- Allows much richer visualizations than are possible in Excel
- Allows interactive visualizations
- All previously-mentioned advantages also apply to visualizations
  - Save time by automating recurring visualizations
  - Reduce risk of error
  - Improve documentation and repeatability



11

Programming Enables More Sophisticated Analysis

- Handling larger data sets
  - Excel may start to choke at 50-100K rows of data/calculations
  - R works faster and is restricted only by your available RAM
- Handling more sophisticated methods
  - Regression and machine learning methods
  - Automating your own selection or modeling methodology

12

Programming Improves Our Technical Skillset

- Emerging analyses (e.g., telematics) involve large data volumes and sophisticated models infeasible in spreadsheets
- Programming already commonly used in research
- Data science already a significant threat to the actuarial profession

13

R vs. Python

14

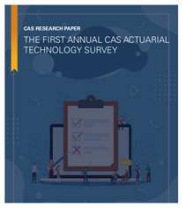
Why R?

- Emerging actuarial standard
  - Most CAS research is done in R
  - R has great actuarial packages
  - SOA Exam PA (≈ CAS Exam 8) now requires R
- Many college students already know it
- But… the best programming language is the one you actually use

15

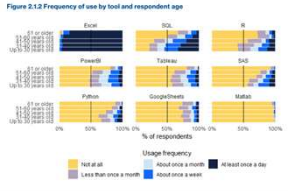
First Annual CAS Actuarial Technology Survey

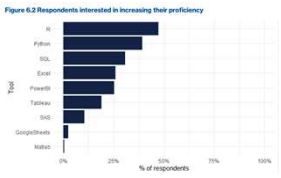
- Released March 16, 2022[2]
- Received 1,294 responses (mix of students and ACAS/FCAS)
- Asked about usage and skill of common tools
- Non-randomized – may be some selection bias
- Excludes proprietary tools (e.g., ResQ, Arius)



16

Survey …





17

Why Python

- Exceedingly popular programming language with easy learning curve
- More common in pure Data Science/Tech Native environments
- Especially suited to Machine Learning and Deep Learning at scale
- Not just for statistical analysis…
- Well suited for integration with Excel
- Actuarial libraries not as robust

18

Which is right for you?

Here are some things to consider:

- Do you have programming experience?
- What do your colleagues use?
- What problems are you trying to solve?
- How important are visualizations?

19

Setup R/Python

20

R & RStudio

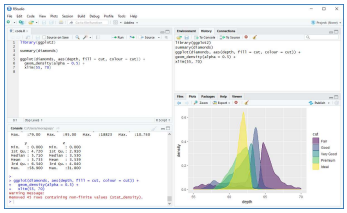
RStudio – Easiest way to get R and full featured R IDE

Free for individual users - a standard for R development

Easy to integrate with Python and SQL

Integrated Package & Notebook environment

Batteries very much included



21

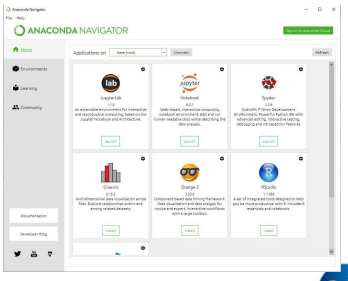
Python

Installing Python has the potential to be much more complicated - *caveat emptor*

Anaconda – Easiest way to get Python and Notebook environment up and running

Easy to integrate with R and SQL

Free for individual users



22

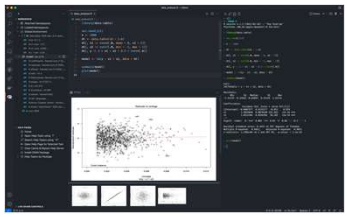
Visual Studio Code

Visual Studio Code – Free and highly performant IDE for R & Python & much, much more!

A great alternative to RStudio & Jupyter Notebooks for development

Code editor-centric development tool, simple & intuitive

Most popular IDE – a great choice for experienced programmers



23

Dealing with IT Challenges

- Get local admin rights (preferably permanent)
- May need to whitelist R/RStudio/R packages/etc. in your virus scanner
- May need to install packages off company VPN
- Avoid installing anything in C:\Program Files or other similarly sensitive folders

24

Learn & Use R/Python

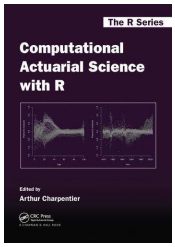
25

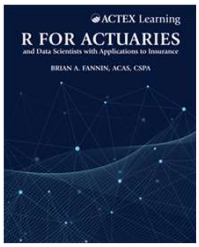
R

- R Studio Education – A great resource for **Beginner**, **Intermediate** & **Advanced** users.
- {Swirl} – Learn R from within R.
- R Bloggers – Comprehensive R news and tutorials.
- R for Data Science – How to do data science with R!
- Actuarial Data Science – Great R resources for intersection of Data & Actuarial Science
- Impatient R – A detailed no-nonsense tutorial for R beginners, from the author of the "The R Inferno"

26

Actuarial Focused References





27

Common All-Purpose R Packages

- tidyverse – A collection of packages for data wrangling and visualization, programming, and more that utilize a user-friendly syntax
- lubridate – Convenient and robust handling of date-time data
- shiny – Interactive web applications for sharing results and analysis tools
- readxl/writexl – Read and write data from/to Excel
- openxlsx – Write, style, and edit Excel worksheets
- rmarkdown – Turn analyses into high quality documents, reports, presentations and dashboards

28

Common Actuarial R Packages

- raw – R Actuarial Workshops
- actuar – Actuarial Functions and Heavy Tailed Distributions
- chainLadder – Statistical Methods and Models for Claims Reserving in General Insurance
- finCal – Time Value of Money, Time Series Analysis and Computational Finance
- imaginator – Simulate General Insurance Policies and Losses
- MortalityLaws – Parametric Mortality Models, Life Tables and HMD
- CAS Datasets

29

Chain Ladder Example

```
# Extract from Vignette

suppressPackageStartupMessages
(library(ChainLadder))
library(ChainLadder)

B <- BootChainLadder(Triangle
= RAA, R = 999, process.distr
= "gamma")
plot(B)
```

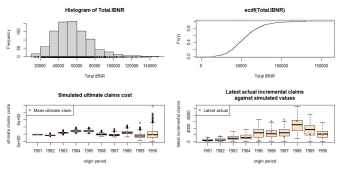


Figure 1: Bootstrap chain-ladder

30

Python

- NumPy – The fundamental package for scientific computing with Python.
- Pandas – Extremely powerful data analysis and manipulation tool.
- MatPlotLib – Comprehensive Python visualization tools.
- Seaborn – Statistical data visualization.
- Google's Python Class – World-class Python basics for free.
- Python Data Science – How to do data science with Python!
- Python for Actuaries – Not free but a good resource.

31

Python Distfit Example

```
# Extracted from distfit examples
import matplotlib.pyplot as plt
from distfit import distfit
import numpy as np
# Example data
X = np.random.normal(0, 3, 2000)
dist = distfit(todf=True)
results = dist.fit_transform(X)
# Plot

[distfit] >fit..

[distfit] >transform..

[distfit] >[norm        ] [0.00 sec] [RSS:
0.00292556] [loc=-0.019 scale=3.011]

[distfit] >[expon       ] [0.0 sec] [RSS:
0.17196] [loc=-9.499 scale=9.480]

[distfit] >[pareto      ] [0.14 sec] [RSS:
0.188392] [loc=-14751138.582
scale=14751129.083]

[distfit] >[dweibull    ] [0.02 sec] [RSS:
0.00798933] [loc=-0.017 scale=2.578]

[distfit] >[t           ] [0.09 sec] [RSS:
0.00292552] [loc=-0.019 scale=3.011
```

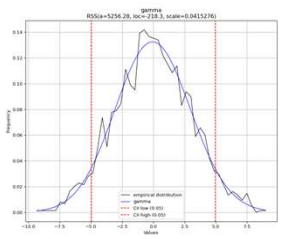
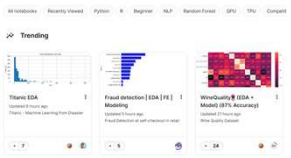


Figure 2: Distribution Fit

32

Strategies for both R and Python

- Read code written by others, and try to replicate
  - Colleagues, online tutorials, Kaggle
- Someone else has likely encountered the same problem before you
  - Google, StackOverflow, other forums
- Follow a set coding style
  - The tidyverse R style guide
  - PEP 8 Python style guide



Kaggle Notebooks

33

When to use Excel instead of Code

34

Speed/Convenience

- Experimenting or prototyping a process
  - Ex: "sketch out" a series of calculations
- Doing a one-time task quickly
  - Ex: pivoting data for quick, high-level analysis

35

Practical Limitations

- End user (e.g., accounting department) requires output in Excel
- Work needs to be reviewed by non-programming resource
- Inputs require manual entry
- Calculations/models don't work well with tables

36

Building Tech Stacks for Actuaries

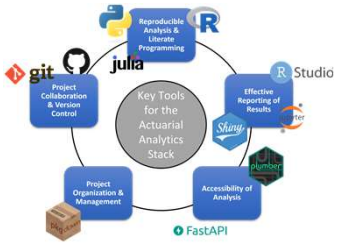
37

Key Tools

- Reproducible Analysis & Literate Programming
- Effective Communication of Results
- Accessibility of Analysis
- Project Organization & Management
- Project Collaboration & Version Control

38

Actuarial Analytics Stack



39

Reproducible Analysis & Literate Programming

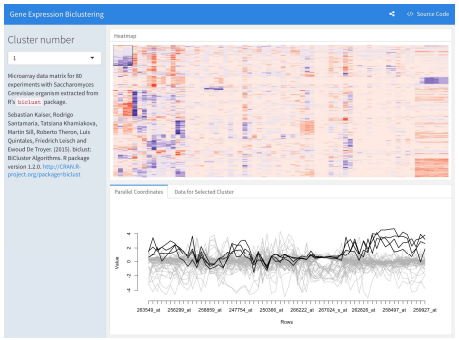
- Accessible language for analytics tasks
- Deep mathematical & analytic libraries
- Easily Read, Clean, Transform & Analyze Data
- Scaleable
- Highly automateable

40

Effective Reporting of Results

- Easily extend code and analysis to results
- Self-documenting code
- Rich data visualizations
- Customization
- Stakeholder Utility

41



42

Accessibility of Analysis

- Expose existing code as a service available to others
- Embedding scientific computing & reproducible research
- Integrate 3rd party services
- Access Management
- Plumber [R]
- OpenCPU [R]
- FastAPI [Python]
- Flask [Python]

43

Project Organization & Management

- R Projects & R Packages
- Best practices for organizing data projects
- *Automagic* automation of project documentation and project structure
- usethis – A workflow package: it automates repetitive tasks that arise during project setup and development.
- pkgdown – A quick and easy way to build documentation for your package.
- projecttemplate – Automate project creation.

44

Project Collaboration & Version Control

- Store & coordinate project work across teams
- Centralize development of your analytics codebase & coordinate changes across teams
- Track, review & implement code changes systematically
- Continuous Integration/Continuous Delivery (CI/CD)

45

Examples

Example 1

Example 2

Example 3

46

Questions, Answers, and Storytime

47

This presentation was coded in R and created by

quarto®

48

Notes

1. "Errors in Operational Spreadsheets," Journal of Organizational and End User Computing, 21(3), 24-36, July-September 2009
http://mba.tuck.dartmouth.edu/spreadsheet/product_pubs_files/errors.pdf

2. https://www.casact.org/articles/cas-releases-results-actuarial-technology-survey.html

49