Workshop B2: A dabbler's introduction to Python
Matt Evans, EMC Actuarial and Analytics
A dabbler's introduction to Python

Tools, Language and Applications
Agenda

• Welcome dabblers!
• The problem with R
• Python dabbling
• Actuarial applications
• Conclusions
The problem with R

• This presentation not intended as part of that phoney war, but…
  – R has a steep learning curve, don’t believe the hype
  – R data structures are not intuitive if you use them infrequently
  – Preferred packages changed in six months (now better, but still need re-learning)
  – Each new algorithm has different package and syntax

• Could Python be the answer? The fanboys say yes.
Python dabbling

• Tools:
  – Anaconda for installing Spyder and Python packages
  – Spyder for editing code

• Trying out the language and packages:
  – Some data generation
  – Popular packages: a GLM fit and XGBoost
Python dabbling: Tools

- Anaconda
  - Installs Spyder
  - Jupyter notebook
  - Rstudio too

- Installs Python packages
  - Most need only a reference but…
  - Some packages (e.g. XGBoost) need separate installation, made easier in Anaconda

- Anaconda something like a “standard” configuration
Python dabbling: Tools

- Spyder
  - Looks a lot like other modern development environments
  - Run and test code
  - Get code hints
  - Examine variables
  - Show output and graphics
Python dabbling: data generation

• This dummy data is designed to be like a claims severity dataset
  – Three factors and a loss severity
  – Factor levels each have a relativity attached, giving expected mean loss
  – Actual loss is simulated with a gamma distribution around the mean

• Distribution *perfect* for fitting a gamma GLM

• How hard is it to do?
Python dabbling: data generation

- The code doesn’t look too bad…
  - Setting up arrays
  - Introducing matrix for interaction
  - DataFrame
  - Sampling from arrays of factor levels
  - Calculating gamma parameters
  - Simulating gamma loss
  - Inferring object types*

```python
# Generate data
cat1 = ['A', 'B']
cat2 = ['1', '2', '3']
cat3 = ['x', 'y', 'z']
rel2 = [1, 4, 6]
rel3 = [1, 1, 1]

# Introduce matrix for interaction
rel23 = [[1, 1, 1],
         [1, 1, 1],
         [1, 1, 1]]

# Test interaction term

# Dataframes, lists and "tuples" not so intuitive
# Relativities for categorical variables

data = pd.DataFrame(columns=['cat1', 'cat2', 'mean', 'loss'])
gamma_data = pd.DataFrame(index=np.arange(0, number_of_rows), columns=['cat1', 'cat2', 'cat3', 'loss_mean', 'loss'])

for i in range(number_of_rows):
    gamma_data.loc[i, 'cat1'] = random.sample(cat1, 1)
    gamma_data.loc[i, 'cat2'] = random.sample(cat2, 1)
    gamma_data.loc[i, 'cat3'] = random.sample(cat3, 1)

shape = 1 # shape held constant to give constant Coefficient of Variation

scale = rel2[cat1.index(gamma_data.loc[i, 'cat1'])] * \
        rel2[cat2.index(gamma_data.loc[i, 'cat2'])] * \
        rel3[cat3.index(gamma_data.loc[i, 'cat3'])]

gamma_data.loc[i, 'loss_mean'] = scale
gamma_data.loc[i, 'loss'] = np.random.gamma(shape, scale, size=None)

# Change loss and loss_mean from object to numerical types (difficult error message without this step)
gamma_data = gamma_data.infer_objects()
```

No end of For loop in Python.

See impact of interaction term on A:2:z value on previous slide.
Python dabbling: data generation VERDICT

• Some problems setting up but actually these were minor
• DataFrames in Python still use occasionally confusing notation (for the dabbler) but R is probably harder
• Code noticeably more object-like
• An easy transition if you are used to VBA
Python dabbling: Popular packages

- Sci-kit learn, the data scientist’s favourite Python package
  - Lots of algorithms all in one consistent package so you can switch easily
  - Regression and Classification algorithms abound

- Statsmodels package, for models closer to R
  - Proper stats with p-values

- XGBoost, Kaggle champion
  - Regression and Classification applications
  - GridSearch - do we need to know what we’re doing any more?
Python dabbling: a GLM fit

• The Sci-kit learn package has a model called *Generalized Linear Model*…
  – But it is only a *linear* model… no link function, not a proper GLM..!

• The Statsmodels package does a proper GLM
  – Code does *two* fits
  – One with interaction; one without

```python
# Fit Gamma GLM (Statsmodels)

# Log link function for multiplicative relativities
import statsmodels.api as sm
import statsmodels.formula.api as smf
import statsmodels.genmod.families.links as link

fmla='loss ~ cat1 + cat2 + cat3'
mod = smf.glm(formula=fmla, data=gamma_data_train, family=sm.families.Gamma(link='loglog'))
res = mod.fit()
print(res.summary())

print('Gamma GLM simple form Relativities - exponentiated parameters')
print(np.exp(res.params))

print("GAMMA GLM simple RMSE: %.2f" % math.sqrt(np.mean((res.predict(gamma_data_test[['cat1','cat2','cat3']]) - gamma_data_test['loss']) ** 2)))

fmla='loss ~ cat1 + cat2 + cat3 + cat2*cat3'
mod = smf.glm(formula=fmla, data=gamma_data_train, family=sm.families.Gamma(link='loglog'))
res = mod.fit()
print(res.summary())

print('Gamma GLM interaction form Relativities - exponentiated parameters')
print(np.exp(res.params))

print("GAMMA GLM interaction RMSE: %.2f" % math.sqrt(np.mean((res.predict(gamma_data_test[['cat1','cat2','cat3']]) - gamma_data_test['loss']) ** 2)))
```

Interaction not allowed for. So more like modelling in the real world where we don’t have full knowledge of risk.

**PERFECT INFORMATION**
Python dabbling: a GLM fit – did it work?

- Our simple model does OK, even though it doesn’t “know” about the interaction
- Relativities quite close, with the interaction load falling into cat2[T.2]

Note the RMSE on 25% hold out sample
  - RMSE 15.66

---

Original Relativities

<table>
<thead>
<tr>
<th>Cat1</th>
<th>Relativity</th>
<th>Cat3</th>
<th>Relativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1.0</td>
<td>x</td>
<td>1.0</td>
</tr>
<tr>
<td>B</td>
<td>5.0</td>
<td>y</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>z</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Cat2 Relativity

<table>
<thead>
<tr>
<th></th>
<th>Cat2:Cat3</th>
<th>Cat3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>4.0</td>
<td>1.0</td>
<td>1.2</td>
</tr>
<tr>
<td>6.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

This would be ~4 without the interaction term.

---

```
Generalized Linear Model Regression Results

Model: GLM  Df Residuals: 7494
Model Family: Gamma  Df Model: 5
Link Function: log  Scale: 1.015135814432834
Date: Thu, 31 May 2018  Deviance: 8713.5
Time: 22:15:36  Pearson chi2: 7.61e+03
No. Iterations: 7

+----------------+--------+--------+-------+-----------------+--------+--------+--------+--------+
|                  | coef   | std err| z     | P>|z|  [0.025]  | 0.975  |
|-----------------|--------|--------|-------|-----------------|--------|--------|--------|--------|
| Intercept       | -0.0337| 0.028  | -1.18 | 0.233           | 0.997  | 0.027  |
| cat1[T.0]       | 1.6099 | 0.023  | 66.14 | 0.000           | 1.000  | 1.000  |
| cat2[T.2]       | 1.4591 | 0.029  | 49.12 | 0.000           | 1.000  | 1.000  |
| cat2[T.3]       | 1.7977 | 0.028  | 62.90 | 0.000           | 1.000  | 1.000  |
| cat3[T,y]       | -0.0025| 0.028  | -0.09 | 0.925           | 0.997  | 0.027  |
| cat3[T,z]       | 0.0380 | 0.029  | 1.32  | 0.185           | 0.997  | 0.027  |

Gamma GLM simple form Relativities - exponentiated parameters

|                  | coef   | std err | z     | P>|z|  [0.025]  | 0.975  |
|-----------------|--------|--------|-------|-----------------|--------|--------|--------|--------|
| Intercept       | 0.97149|        |       |                 |        |        |        |        |
| cat1[T.0]       | 5.00213|        |       |                 |        |        |        |        |
| cat2[T.2]       | 4.36999|        |       |                 |        |        |        |        |
| cat2[T.3]       | 6.09385|        |       |                 |        |        |        |        |
| cat3[T,y]       | 0.99743|        |       |                 |        |        |        |        |
| cat3[T,z]       | 1.83873|        |       |                 |        |        |        |        |

type: float64  GAMMA GLM simple RMSE: 15.66
```
Python dabbling: a GLM fit – did it work?

• Our “perfect information” model does better – it fits interaction

• Relativities quite close, even the interaction loading

• Note the RMSE is about 0.1% reduced (not a lot!)
  – RMSE 15.64
Python dabbling: XGBoost

- This regression and classification algorithm uses gradient boosted trees
- It wins a lot of competitions on Kaggle, the data science website, and doesn’t do too badly here either
- RMSE on 25% hold out sample:
  - XGBOOST RMSE: 15.65
    - This is better than the typical real world GLM model with imperfect information, without knowing anything
- A simple test with dummy data only, but encouraging, alongside anecdotal evidence

```python
# Fit XGBoost
import xgboost
from sklearn.grid_search import GridSearchCV

xgb = xgboost.XGBRegressor(n_estimators=300,
                           learning_rate=0.1,
                           gamma=0.15,
                           subsample=0.75,
                           colsample_bytree=0.7,
                           max_depth=4)

xgb.fit(one_hot_train, gamma_data_train[['loss']]) # X and y
print("XGBOOST RMSE: %.2f" % math.sqrt(np.mean((xgb.predict(one_hot_test) - gamma_data_test['loss']) ** 2)))

# Grid search
parameters = {'nthread':[1],
              'n_estimators': [910,920], #number of trees, change it to 1000 for better results
              'learning_rate': [0.10,0.15], #so called 'eta' value
              'max_depth': [4,5,6,7],
              'gamma': [0.1,0.15],
              'subsample': [0.75],
              'colsample_bytree': [0.5,0.6,0.7],
              'seed': [1337]}

g = GridSearchCV(xgb, parameters, n_jobs=10,
                 verbose=0,
                 cv=10,
                 scoring='neg_mean_squared_error',
                 refit=True)
g.fit(one_hot_train, gamma_data_train[['loss']]) # X and y
print("XGBOOST RMSE: %.2f" % math.sqrt(np.mean((g.predict(one_hot_test) - gamma_data_test['loss']) ** 2)))
```

GridSearch uses cross-validation to optimise parameters, offering route to further improvement
Python dabbling: Popular packages VERDICT

- Sci-kit learn a bit disappointing as tests with a wide range of regressors failed to get near GLM result, however…
  - Cross validation and train/test facilities are useful
  - Many regressors can be tried out and trained with GridSearch, which should improve results

- Statsmodels works very well
  - Output as good as R
  - Fast to run
  - Model formulae standard and intuitive
Python dabbling: Popular packages VERDICT

• XGBoost
  – On our admittedly simple test, this algorithm gets very close to a GLM with perfect information
    • Full GLM knows structure \textit{and} error distribution
    • XGBoost uses only cross-validation to improve fit
  – In the real world our data is full of hidden interactions and effects... could we get closer with this?
  – Could this approach work well where...:
    • We have little data?
    • We are not resourced to do full GLM modelling?
    • We suspect non-linear characteristics?
    • We are less worried about a "black-box"?

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLM without interaction term</td>
<td>15.66</td>
</tr>
<tr>
<td>GLM with interaction term</td>
<td>15.64</td>
</tr>
<tr>
<td>XGBoost</td>
<td>15.65</td>
</tr>
<tr>
<td>Best Sci-kit (1st pass only)</td>
<td>16+</td>
</tr>
</tbody>
</table>
Actuarial applications

- Data wrangling
- Pricing with GLM, GAM, XGBoost and various others
- Pricing using curve fitting and simulation
- Reserving with ChainLadder* package now ported from R to Python
- Capital Modelling
- Reporting and Graphics
Strengths and weaknesses

• Python is from world of IT:
  – Is proper programming, and can be deployed easily in modern IT structures
  – Has a whole load of other stuff in it, not just stats
  – Is supported by wider pool of talent
  – But is weaker as not so statistical (..?)

• R is from world of academic stats
  – Closer to SAS and SPSS
  – Proper statistics with p-values!
  – Rich library of packages
  – Good at data handling
  – But harder to learn if you’re used to VBA
  – And more difficult to deploy (..?)
Conclusions

• On balance, I will likely look to Python first for new projects
  – Data handling and syntax are more intuitive for this VBA-soaked actuary
  – A friendlier introduction to the new data science tools available
  – Python is used in QGIS, open source mapping software

• Others may feel differently…
  – Do you have a lot of SAS experience in your team?
  – Would your team emphasise “proper” statistics?
  – GLM/GAM are more established in R
EMC Actuarial & Analytics

About Us

- Peter England
  - Capital
  - Reserving
  - IFRS 17
  - Stochastic/statistical modelling
  - Research

  - peter@emc-actuarial.com

- Matthew Evans
  - Pricing
  - Reserving
  - Data Science
  - InsurTech

  - matthew@emc-actuarial.com
References

• Anaconda installer  https://www.anaconda.com/download/
  – Spyder code editor and console
  – Jupiter notebooks
• Sci-kit learn  http://scikit-learn.org/stable/
• Kaggle competitions  https://www.kaggle.com/
• XGBoost Champion regression algorithm
• MatPlotLib Python graphing
The views expressed in this [publication/presentation] are those of invited contributors and not necessarily those of the IFoA. The IFoA do not endorse any of the views stated, nor any claims or representations made in this [publication/presentation] and accept no responsibility or liability to any person for loss or damage suffered as a consequence of their placing reliance upon any view, claim or representation made in this [publication/presentation].

The information and expressions of opinion contained in this publication are not intended to be a comprehensive study, nor to provide actuarial advice or advice of any nature and should not be treated as a substitute for specific advice concerning individual situations. On no account may any part of this [publication/presentation] be reproduced without the written permission of the IFoA [or authors, in the case of non-IFoA research].