



## DH13: SAS vs. R vs. Python - A Comparison of Data Exploration Techniques

Nathanael Cockburn

# Agenda

- Why this topic?
- Technical points
- A 15 minute data exploration
- Conclusion

# Why this topic?

- Many comparisons of R, Python and SAS are largely academic.
  - One may be a clear winner.
  - Work requirements may take away any choice.
- Basic data exploration is more system agnostic.

# Technical points

- SAS – University edition
- R - Tidyverse, Skimr
- Python - Pandas

# 15 minute data exploration

This presentation will cover:

- Reading in a data file.
- Checking the import.
- Review of metadata.
- Basic overview of data content.
- Checking for simple trends.

# Reading in a datafile

This presentation will use the “cars.csv” file that can be found on the SAS website.

All three languages have simple options to import datafiles.

```
proc import datafile = "/folders/myfolders/data/cars.csv" out = cars;  
    guessingrows = 32767;  
run;
```

```
cars <- read.csv("C:/Users/Nathanael/Documents/R/cars.csv")
```

```
cars = pandas.read_csv("C:/Users/Nathanael/Documents/Python/cars.csv")
```

# Checking the import - SAS

SAS has good options for opening up datasets.

- Checking variable names match the data.
- Checking all observations imported.

|   | <b>Make</b> | <b>Model</b>            | <b>Type</b> | <b>Origin</b> | <b>DriveTrain</b> | <b>MSRP</b> | <b>Inv</b> |
|---|-------------|-------------------------|-------------|---------------|-------------------|-------------|------------|
| 1 | Acura       | MDX                     | SUV         | Asia          | All               | \$36,945    | \$33       |
| 2 | Acura       | RSX Type S 2dr          | Sedan       | Asia          | Front             | \$23,820    | \$21       |
| 3 | Acura       | TSX 4dr                 | Sedan       | Asia          | Front             | \$26,990    | \$24       |
| 4 | Acura       | TL 4dr                  | Sedan       | Asia          | Front             | \$33,195    | \$30       |
| 5 | Acura       | 3.5 RL 4dr              | Sedan       | Asia          | Front             | \$43,755    | \$39       |
| 6 | Acura       | 3.5 RL w/Navigation 4dr | Sedan       | Asia          | Front             | \$46,100    | \$41       |
| 7 | Acura       | NSX coupe 2dr manual S  | Sports      | Asia          | Rear              | \$89,765    | \$79       |
| 8 | Audi        | A4 1.8T 4dr             | Sedan       | Europe        | Front             | \$25,940    | \$23       |

# Checking the import - R

RStudio has a similar option to open datasets.

|   | Make  | Model                   | Type   | Origin | DriveTrain | MSRP  | Invoice | EngineSize | Cylinders | Hors |
|---|-------|-------------------------|--------|--------|------------|-------|---------|------------|-----------|------|
| 1 | Acura | MDX                     | SUV    | Asia   | All        | 36945 | 33337   | 3.5        | 6         | 265  |
| 2 | Acura | RSX Type S 2dr          | Sedan  | Asia   | Front      | 23820 | 21761   | 2.0        | 4         | 200  |
| 3 | Acura | TSX 4dr                 | Sedan  | Asia   | Front      | 26990 | 24647   | 2.4        | 4         | 200  |
| 4 | Acura | TL 4dr                  | Sedan  | Asia   | Front      | 33195 | 30299   | 3.2        | 6         | 270  |
| 5 | Acura | 3.5 RL 4dr              | Sedan  | Asia   | Front      | 43755 | 39014   | 3.5        | 6         | 225  |
| 6 | Acura | 3.5 RL w/Navigation 4dr | Sedan  | Asia   | Front      | 46100 | 41100   | 3.5        | 6         | 225  |
| 7 | Acura | NSX coupe 2dr manual S  | Sports | Asia   | Rear       | 89765 | 79978   | 3.2        | 6         | 290  |
| 8 | Audi  | A4 1.8T 4dr             | Sedan  | Europe | Front      | 25940 | 23508   | 1.8        | 4         | 170  |
| 9 | Audi  | A4 1.8T convertible 2dr | Sedan  | Europe | Front      | 35940 | 32506   | 1.8        | 4         | 170  |

R also has the head() function, though this is less attractive.

```
> head(cars)
```

|   | Make  | Model                   | Type  | Origin | DriveTrain | MSRP  | Invoice | EngineSize | Cylinders | Horsepower | MPG_City | MPG_Highway | Weight | Wheelbase |
|---|-------|-------------------------|-------|--------|------------|-------|---------|------------|-----------|------------|----------|-------------|--------|-----------|
| 1 | Acura | MDX                     | SUV   | Asia   | All        | 36945 | 33337   | 3.5        | 6         | 265        | 17       | 23          | 4451   | 106       |
| 2 | Acura | RSX Type S 2dr          | Sedan | Asia   | Front      | 23820 | 21761   | 2.0        | 4         | 200        | 24       | 31          | 2778   | 101       |
| 3 | Acura | TSX 4dr                 | Sedan | Asia   | Front      | 26990 | 24647   | 2.4        | 4         | 200        | 22       | 29          | 3230   | 105       |
| 4 | Acura | TL 4dr                  | Sedan | Asia   | Front      | 33195 | 30299   | 3.2        | 6         | 270        | 20       | 28          | 3575   | 108       |
| 5 | Acura | 3.5 RL 4dr              | Sedan | Asia   | Front      | 43755 | 39014   | 3.5        | 6         | 225        | 18       | 24          | 3880   | 115       |
| 6 | Acura | 3.5 RL w/Navigation 4dr | Sedan | Asia   | Front      | 46100 | 41100   | 3.5        | 6         | 225        | 18       | 24          | 3880   | 115       |

# Checking the import - Python

Python doesn't have a viewer for dataframes, but a similar head() function can be used to check the data.

```
>>> print(cars.iloc[:,0:7].head(5))
```

|   | Make  | Model      | Type      | Origin | DriveTrain | MSRP    | Invoice |
|---|-------|------------|-----------|--------|------------|---------|---------|
| 0 | Acura | MDX        | SUV       | Asia   | All        | 36945.0 | 33337.0 |
| 1 | Acura | RSX Type S | 2dr Sedan | Asia   | Front      | 23820.0 | 21761.0 |
| 2 | Acura | TSX        | 4dr Sedan | Asia   | Front      | 26990.0 | 24647.0 |
| 3 | Acura | TL         | 4dr Sedan | Asia   | Front      | 33195.0 | 30299.0 |
| 4 | Acura | 3.5 RL     | 4dr Sedan | Asia   | Front      | 43755.0 | 39014.0 |

# Review metadata - SAS

SAS has a couple of options for reviewing metadata.

- SASHELP datasets

|    | libname | memname | memtype | name        | type | length |
|----|---------|---------|---------|-------------|------|--------|
| 1  | WORK    | CARS    | DATA    | Make        | char | 13     |
| 2  | WORK    | CARS    | DATA    | Model       | char | 39     |
| 3  | WORK    | CARS    | DATA    | Type        | char | 6      |
| 4  | WORK    | CARS    | DATA    | Origin      | char | 6      |
| 5  | WORK    | CARS    | DATA    | DriveTrain  | char | 5      |
| 6  | WORK    | CARS    | DATA    | MSRP        | num  | 8      |
| 7  | WORK    | CARS    | DATA    | Invoice     | num  | 8      |
| 8  | WORK    | CARS    | DATA    | EngineSize  | num  | 8      |
| 9  | WORK    | CARS    | DATA    | Cylinders   | num  | 8      |
| 10 | WORK    | CARS    | DATA    | Horsepower  | num  | 8      |
| 11 | WORK    | CARS    | DATA    | MPG_City    | num  | 8      |
| 12 | WORK    | CARS    | DATA    | MPG_Highway | num  | 8      |

# Review metadata - SAS

SAS has a couple of options for reviewing metadata.

- SASHELP datasets
- Proc contents

```
proc contents data=cars;  
run;
```

| Alphabetic List of Variables and Attributes |             |      |     |         |          |
|---------------------------------------------|-------------|------|-----|---------|----------|
| #                                           | Variable    | Type | Len | Format  | Informat |
| 9                                           | Cylinders   | Num  | 8   | BEST12. | BEST32.  |
| 5                                           | DriveTrain  | Char | 5   | \$5.    | \$5.     |
| 8                                           | EngineSize  | Num  | 8   | BEST12. | BEST32.  |
| 10                                          | Horsepower  | Num  | 8   | BEST12. | BEST32.  |
| 7                                           | Invoice     | Num  | 8   | BEST12. | BEST32.  |
| 15                                          | Length      | Num  | 8   | BEST12. | BEST32.  |
| 11                                          | MPG_City    | Num  | 8   | BEST12. | BEST32.  |
| 12                                          | MPG_Highway | Num  | 8   | BEST12. | BEST32.  |
| 6                                           | MSRP        | Num  | 8   | BEST12. | BEST32.  |
| 4                                           | Make        | Char | 12  | \$12.   | \$12.    |

# Review metadata - R

R can provide a summary of variable information using `str()`.

```
> str(cars)
'data.frame':   428 obs. of  15 variables:
 $ Make      : Factor w/ 38 levels "Acura","Audi",...: 1 1 1 1 1 1 1 2 2 2 ...
 $ Model     : Factor w/ 425 levels "3.5 RL 4dr","3.5 RL w/Navigation 4dr",...: 245
 $ Type      : Factor w/ 6 levels "Hybrid","Sedan",...: 4 2 2 2 2 2 3 2 2 2 ...
 $ Origin    : Factor w/ 3 levels "Asia","Europe",...: 1 1 1 1 1 1 1 2 2 2 ...
 $ DriveTrain: Factor w/ 3 levels "All","Front",...: 1 2 2 2 2 2 3 2 2 2 ...
 $ MSRP      : num  36945 23820 26990 33195 43755 ...
 $ Invoice    : num  33337 21761 24647 30299 39014 ...
 $ EngineSize: num  3.5 2 2.4 3.2 3.5 3.5 3.2 1.8 1.8 3 ...
 $ Cylinders  : num  6 4 4 6 6 6 6 4 4 6 ...
 $ Horsepower: num  265 200 200 270 225 225 290 170 170 220 ...
 $ MPG_City   : num  17 24 22 20 18 18 17 22 23 20 ...
 $ MPG_Highway: num  23 31 29 28 24 24 24 31 30 28 ...
 $ Weight     : num  4451 2778 3230 3575 3880 ...
 $ Wheelbase  : num  106 101 105 108 115 115 100 104 105 104 ...
 $ Length     : num  189 172 183 186 197 197 174 179 180 179 ...
```

# Review metadata - Python

Python can provide a summary of variable information using `info()`.

```
>>> print(cars.info())
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 428 entries, 0 to 427
Data columns (total 15 columns):
Make                428 non-null object
Model               428 non-null object
Type                428 non-null object
Origin              428 non-null object
DriveTrain          428 non-null object
MSRP                428 non-null float64
Invoice             428 non-null float64
```

# Data content - SAS

SAS can provide some summaries to start understanding data content.

- Proc freq for categorical information.

```
proc freq data=cars (keep=make origin type drivetrain);  
run;
```

| Type   | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
|--------|-----------|---------|----------------------|--------------------|
| Hybrid | 3         | 0.70    | 3                    | 0.70               |
| SUV    | 60        | 14.02   | 63                   | 14.72              |
| Sedan  | 262       | 61.21   | 325                  | 75.93              |
| Sports | 49        | 11.45   | 374                  | 87.38              |
| Truck  | 24        | 5.61    | 398                  | 92.99              |
| Wagon  | 30        | 7.01    | 428                  | 100.00             |

| Origin | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
|--------|-----------|---------|----------------------|--------------------|
| Asia   | 158       | 36.92   | 158                  | 36.92              |
| Europe | 123       | 28.74   | 281                  | 65.66              |
| USA    | 147       | 34.35   | 428                  | 100.00             |

| DriveTrain | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
|------------|-----------|---------|----------------------|--------------------|
| All        | 92        | 21.50   | 92                   | 21.50              |
| Front      | 226       | 52.80   | 318                  | 74.30              |
| Rear       | 110       | 25.70   | 428                  | 100.00             |

# Data content - SAS

SAS can provide some summaries to start understanding data content.

- Proc freq for categorical information.
- Proc means for numeric data.

```
proc means data = cars (keep=cylinders horsepower weight);  
run;
```

| Variable   | N   | Mean        | Std Dev     | Minimum    | Maximum     |
|------------|-----|-------------|-------------|------------|-------------|
| Cylinders  | 426 | 5.8075117   | 1.5584426   | 3.0000000  | 12.0000000  |
| Horsepower | 428 | 215.8855140 | 71.8360316  | 73.0000000 | 500.0000000 |
| Weight     | 428 | 3577.95     | 758.9832146 | 1850.00    | 7190.00     |

# Data content - R

R can drop out some simple summary statistics very easily using `summary()`.

```
> summary(cars)
```

| Make              | Model                      | Type       | Origin     | DriveTrain | MSRP           |
|-------------------|----------------------------|------------|------------|------------|----------------|
| Toyota : 28       | C240 4dr : 2               | Hybrid: 3  | Asia :158  | All : 92   | Min. : 10280   |
| Chevrolet : 27    | C320 4dr : 2               | Sedan :262 | Europe:123 | Front:226  | 1st Qu.: 20334 |
| Mercedes-Benz: 26 | G35 4dr : 2                | Sports: 49 | USA :147   | Rear :110  | Median : 27635 |
| Ford : 23         | 3.5 RL 4dr : 1             | SUV : 60   |            |            | Mean : 32775   |
| BMW : 20          | 3.5 RL w/Navigation 4dr: 1 | Truck : 24 |            |            | 3rd Qu.: 39205 |
| Audi : 19         | 300M 4dr : 1               | Wagon : 30 |            |            | Max. :192465   |
| (Other) :285      | (Other) :419               |            |            |            |                |

| Cylinders      | Horsepower    | MPG_City      | MPG_Highway   | Weight       | Wheelbase     | Length        |
|----------------|---------------|---------------|---------------|--------------|---------------|---------------|
| Min. : 3.000   | Min. : 73.0   | Min. :10.00   | Min. :12.00   | Min. :1850   | Min. : 89.0   | Min. :101.0   |
| 1st Qu.: 4.000 | 1st Qu.:165.0 | 1st Qu.:17.00 | 1st Qu.:24.00 | 1st Qu.:3104 | 1st Qu.:103.0 | 1st Qu.:106.0 |
| Median : 6.000 | Median :210.0 | Median :19.00 | Median :26.00 | Median :3474 | Median :107.0 | Median :108.0 |
| Mean : 5.808   | Mean :215.9   | Mean :20.06   | Mean :26.84   | Mean :3578   | Mean :108.2   | Mean :109.0   |
| 3rd Qu.: 6.000 | 3rd Qu.:255.0 | 3rd Qu.:21.25 | 3rd Qu.:29.00 | 3rd Qu.:3978 | 3rd Qu.:112.0 | 3rd Qu.:111.0 |
| Max. :12.000   | Max. :500.0   | Max. :60.00   | Max. :66.00   | Max. :7190   | Max. :144.0   | Max. :112.0   |
| NA's :2        |               |               |               |              |               |               |

# Data content - R

R can drop out some simple summary statistics very easily using `summary()` or `skim()`.

```
-- Variable type: factor -----  
# A tibble: 5 x 6  
  skim_variable n_missing complete_rate ordered n_unique top_counts  
*   <chr>          <int>         <dbl> <lgl>      <int> <chr>  
1 Make              0             1 FALSE      38 Toy: 28, Che: 27, Mer: 26, For: 23  
2 Model             0             1 FALSE     425 C24: 2, C32: 2, G35: 2, 3.5: 1  
3 Type              0             1 FALSE      6 Sed: 262, SUV: 60, Spo: 49, Wag: 30  
4 Origin            0             1 FALSE      3 Asi: 158, USA: 147, Eur: 123  
5 DriveTrain        0             1 FALSE      3 Fro: 226, Rea: 110, All: 92
```

# Data content - R

R can drop out some simple summary statistics very easily using `summary()` or `skim()`.

```
-- Variable type: numeric -----  
# A tibble: 10 x 11  
  skim_variable n_missing complete_rate   mean    sd    p0    p25    p50    p75    p100 hist  
*   <chr>          <int>         <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>  
1 MSRP              0           1 32775. 19432. 10280 20334. 27635 39205 192465   
2 Invoice            0           1 30015. 17642.  9875 18866 25294. 35710. 173560   
3 EngineSize        0           1    3.20   1.11   1.3   2.38    3    3.9    8.3   
4 Cylinders         2         0.995    5.81   1.56    3     4     6     6    12   
5 Horsepower        0           1    216.   71.8   73    165   210   255   500   
6 MPG_City          0           1    20.1   5.24   10    17    19   21.2   60   
7 MPG_Highway       0           1    26.8   5.74   12    24    26   29    66   
8 Weight            0           1   3578.  759.  1850  3104  3474. 3978.  7190   
9 Wheelbase         0           1    108.   8.31   89    103   107   112   144   
10 Length           0           1    186.  14.4  143   178   187   194   238 
```

# Data content - Python

Python has a similar option to R, with `describe()` for numeric variables.

```
>>> print(cars.describe())
```

|       | MSRP          | Invoice       | ... | Wheelbase  | Length     |
|-------|---------------|---------------|-----|------------|------------|
| count | 428.000000    | 428.000000    | ... | 428.000000 | 428.000000 |
| mean  | 32774.855140  | 30014.700935  | ... | 108.154206 | 186.362150 |
| std   | 19431.716674  | 17642.117750  | ... | 8.311813   | 14.357991  |
| min   | 10280.000000  | 9875.000000   | ... | 89.000000  | 143.000000 |
| 25%   | 20334.250000  | 18866.000000  | ... | 103.000000 | 178.000000 |
| 50%   | 27635.000000  | 25294.500000  | ... | 107.000000 | 187.000000 |
| 75%   | 39205.000000  | 35710.250000  | ... | 112.000000 | 194.000000 |
| max   | 192465.000000 | 173560.000000 | ... | 144.000000 | 238.000000 |

```
[8 rows x 10 columns]
```

# Data content - Python

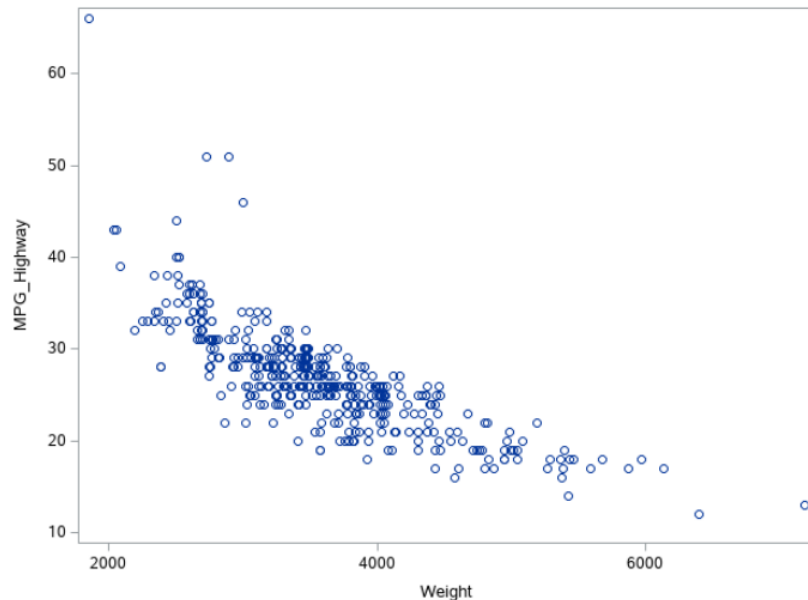
Python has a similar option to R, with `describe()` for numeric variables and `value_counts()` for categorical ones.

```
>>> print(cars["Origin"].value_counts())  
Asia      158  
USA       147  
Europe    123  
Name: Origin, dtype: int64
```

# Simple trends - SAS

Easiest way to spot simple trends with limited time, is by plotting data.

```
.....  
proc sgplot data = cars;  
    scatter x = weight y = mpg_highway;  
run;
```



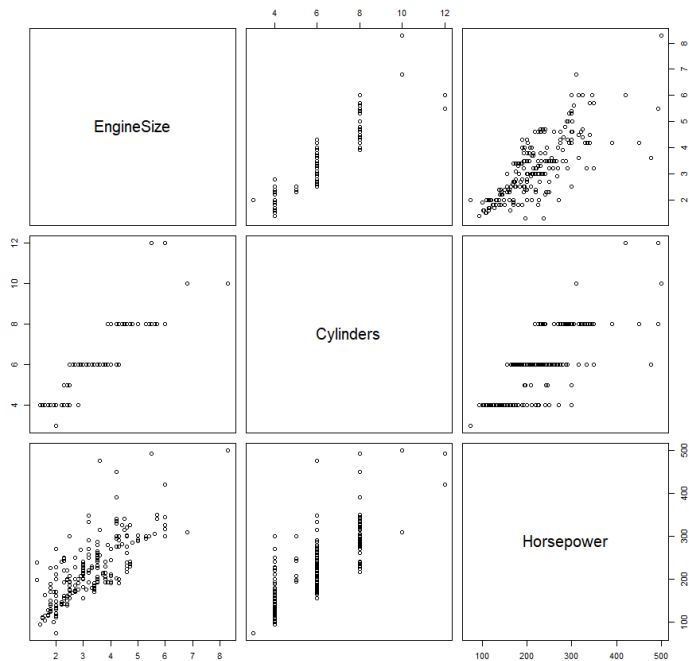
# Simple trends - SAS

Easiest way to spot simple trends with limited time, is by plotting data. However, code starts to get more involved as you go on.

```
%macro plot(x=, y=);  
    proc sql noprint;  
        select distinct name into :var1-:var10  
        from sashelp.vcolumn  
        scatter x = &x y where type = "num"  
    quit;  
run;  
%mend;  
  
%plot(x=weight,y=mpg_highway);  
%plot(x=cylinders,y=horsepower);  
  
%macro multiplot;  
    %do i = 1 %to 10;  
        %do j = 1 %to 10;  
            %plot(x=&&var&i,y=&&var&j)  
        %end;  
    %end;  
%mend;  
%multiplot;
```

# Simple trends - R

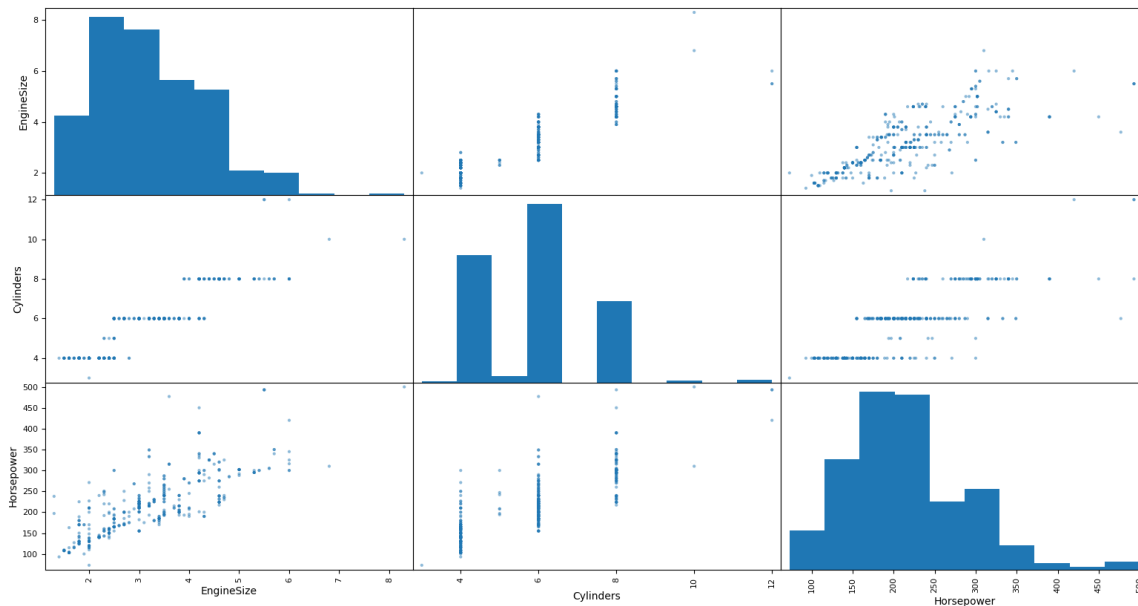
R has a built in function for plotting variable combinations – `pairs()`.



# Simple trends - Python

Python has a similar function to R.

```
scatter_matrix(cars.iloc[:,7:10])  
plt.show()
```



# Conclusions

- SAS is powerful, but not designed for quick use.
- Python has some good tools, but they feel a little clunky.
- R is really well set up for this sort of fast data review.

# Further information

- Accompanying paper goes into functions in more detail.
- Any questions to [nathanael.cockburn@quanticate.com](mailto:nathanael.cockburn@quanticate.com)

# Thank you for listening