Using the Pandas library to explore the data

Learning objectives:

- Use shell commands to navigate to the data folder
- Import Psychopy data into the Jupyter Notebook
- Generate descriptive statistics
- Plot some comparisons by grouping data

We will set up the notebook and load the additional software library we will need - "Pandas".

```
import pandas as pd
```

The Jupyter Notebook understands shell commands as well, we'll use this to find our data. We'll use the exclamation mark to signal that what follows is a shell command rather than Python code.

```
# print working directory
!pwd

/Users/talithaford/Dropbox/Conference-Workshop/ResBaz/2016/psychopy_lesson



# list files in 'data/'
!ls data/

tf1_visneuro_2016_Feb_02_1234.csv    tf_visneuro_2016_Feb_02_1218.psydat
tf1_visneuro_2016_Feb_02_1234.log    tf_visneuro_2016_Feb_02_1233.csv
tf1_visneuro_2016_Feb_02_1234.psydat tf_visneuro_2016_Feb_02_1233.log
tf_visneuro_2016_Feb_02_1218.csv     tf_visneuro_2016_Feb_02_1233.psydat
tf_visneuro_2016_Feb_02_1218.log
```

We can use the 'read_csv' function of Pandas to read the datafile into a pandas DataFrame. A DataFrame will look familiar to anyone who has used Excel or SPSS.

```
# read the csv
df = pd.read_csv('data/tf1_visneuro_2016_Feb_02_1234.csv')
# print the first 5 lines of the DataFrame
df.head()
```

	correct_ans	images	type_of_image	loop.thisRepN	loop.thisTrialN	loop.thisN	loop.thisIndex	key_resp_3.keys	key_resp_
0	up	images/egg_hog.jpeg	hog	0	0	0	6	up	1
1	right	images/cat_eyes.jpeg	cat	0	1	1	3	right	1
2	left	images/gorilla_tongue.jpeg	gorilla	0	2	2	2	left	1
3	up	images/santa_hog.jpeg	hog	0	3	3	9	up	1
4	right	images/ear_muffs_cat.jpeg	cat	0	4	4	4	right	1

```
# drop the first row (under the header)
# the inplace option allows us to makes changes to the dataframe directly
# rather than having to save the output to another variable
df.drop(0, inplace=True)
df.head()
```

	correct_ans	images	type_of_image	loop.thisRepN	loop.thisTrialN	loop.thisN	loop.thisIndex	key_resp_3.keys	key_re
1	right	images/cat_eyes.jpeg	cat	0	1	1	3	right	1
2	left	images/gorilla_tongue.jpeg	gorilla	0	2	2	2	left	1
3	up	images/santa_hog.jpeg	hog	0	3	3	9	up	1
4	right	images/ear_muffs_cat.jpeg	cat	0	4	4	4	right	1
5	up	images/marshmallow_hog.jpeg	hog	0	5	5	8	up	1

```
# drop all colums with NA's
df.dropna(axis=1, inplace=True, how='all')
df.head()
```

	correct_ans	images	type_of_image	loop.thisRepN	loop.thisTrialN	loop.thisN	loop.thisIndex	key_resp_3.keys	key_re
1	right	images/cat_eyes.jpeg	cat	0	1	1	3	right	1
2	left	images/gorilla_tongue.jpeg	gorilla	0	2	2	2	left	1
3	up	images/santa_hog.jpeg	hog	0	3	3	9	up	1
4	right	images/ear_muffs_cat.jpeg	cat	0	4	4	4	right	1
5	up	images/marshmallow_hog.jpeg	hog	0	5	5	8	up	1

```
# save out new dataframe as csv
df.to_csv('data/ID001.csv', index=False)

# to read back in you can use:
#df = pd.read_csv('data/ID001.csv')
```

Visualising the data

**Now that we have cleaned the data, it might be helpful to visualise it.
To do this we need to import some packages. **

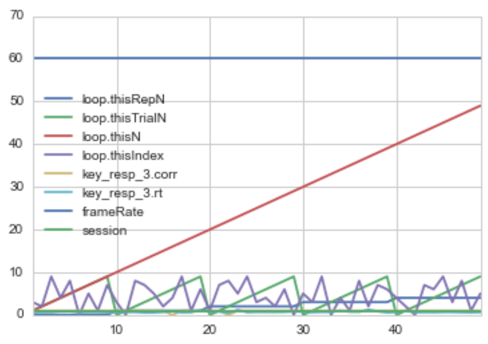
Matplotlib is a Python library that produces quality figures and graphs from data.
Seaborn is a Python library based on matplotlib, which provides additional graphical modifications for matplotlib figures and graphs.

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns # seaborn

# to see the plots in the workspace we use:
%matplotlib inline

# these are seaborn commands, they are used for asthetics
sns.set_color_codes()
sns.set_style('whitegrid')
```

Create a basic plot of all the columns

```
df.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x109768b50>
```



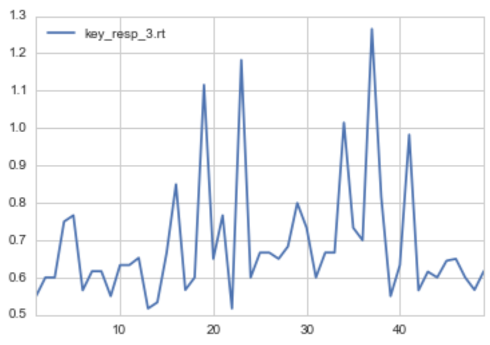
The plot is good, but it doesn't actually tell us much.
Let's used the package matplotlib.pyplot ('plt'), but first, find out what the function 'plot' in the package does.

```
plt.plot?
```

To plot a single column (e.g. reaction time), enter the name of the column (the header) as a **string** in square brackets immediately following the dataframe label as shown below. The column header is the 'index' for that column. In this example, the plot will include the legend.

```
df['key_resp_3.rt'].plot(legend=True)
```

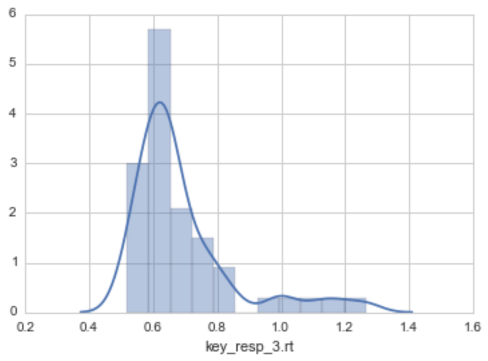
```
<matplotlib.axes._subplots.AxesSubplot at 0x109d68cd0>
```



Seaborn (sns) has a nice distribution plot to visualise whether the responses are normally disributed.

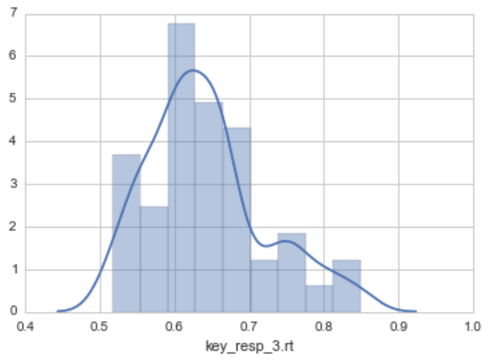
```
sns.distplot(df['key_resp_3.rt'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x109d9de10>
```



It is clear from the graphs that the first answer is much longer than the rest of the answers. This is more likely due to it being the first response and therefore it isn't real data. We should filter it out so it doesn't overly affect the result.

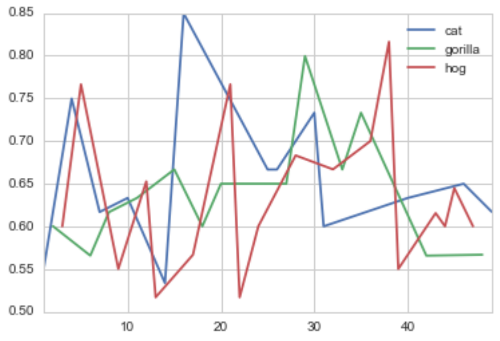
```
df_clean = df[df['key_resp_3.rt'] < .9]


sns.distplot(df_clean['key_resp_3.rt'])




<matplotlib.axes._subplots.AxesSubplot at 0x109e5f650>
```



It's still not looking particularly normal but better. Maybe less than 0.3 would be better still...

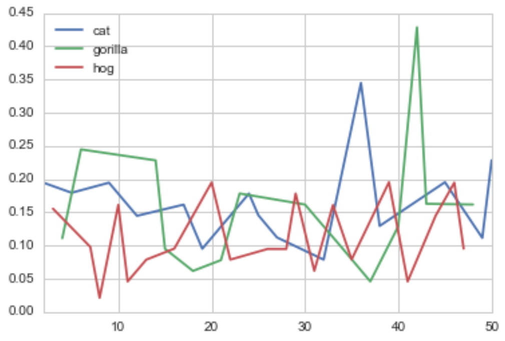
We can plot the reaction time by the type of image by including the column name 'typeofimage' in parentheses before the index.

```
df_clean.groupby('type_of_image')['key_resp_3.rt'].plot(legend=True)




type_of_image
cat        Axes(0.125,0.125;0.775x0.775)
gorilla    Axes(0.125,0.125;0.775x0.775)
hog        Axes(0.125,0.125;0.775x0.775)
Name: key_resp_3.rt, dtype: object
```



Creating a variable that groups reation time by the animal type gives some more functionality

```
animal_type_group = df_clean.groupby('type_of_image')['key_resp_3.rt']


# then we can plot that variable
animal_type_group.plot(legend = True)




type_of_image
cat        Axes(0.125,0.125;0.775x0.775)
gorilla    Axes(0.125,0.125;0.775x0.775)
hog        Axes(0.125,0.125;0.775x0.775)
Name: key_resp_4.rt, dtype: object
```

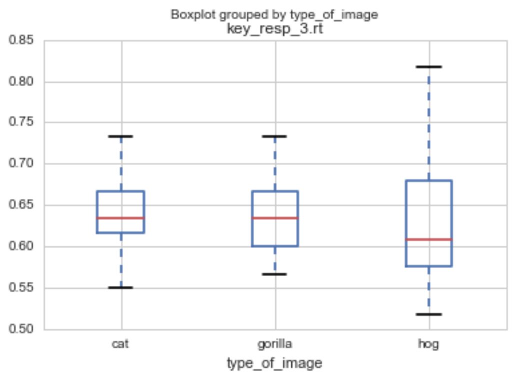


This doesn't tell us much, but we can now print out the statistics for that variable.

```
animal_type_group.mean()




type_of_image
cat        0.653995
gorilla    0.639744
hog        0.634237
Name: key_resp_3.rt, dtype: float64




print('Mean: ',animal_type_group.mean())
print('SD: ',animal_type_group.std())

('Mean: ', type_of_image
cat        0.653995
gorilla    0.639744
hog        0.634237
Name: key_resp_3.rt, dtype: float64)
('SD: ', type_of_image
cat        0.084515
gorilla    0.068218
hog        0.086569
Name: key_resp_3.rt, dtype: float64)
```

We can also visualise the reaction time by the type of image as a boxplot

```
df_clean.boxplot('key_resp_3.rt',by = 'type_of_image',)



<matplotlib.axes._subplots.AxesSubplot at 0x10a165690>
```



and save plot to the current working directory

```
plt.savefig('ID001_boxplot.png')


<matplotlib.figure.Figure at 0x10a87a390>
```