Basic-Data-Manipulation

May 14, 2021

1 Basic data manipulation example

This notebook gives an example of a possible approach to loading and manipulating the data, and flexibly defining explorative plots.

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
```

1.1 Import data

We start by importing the data as a Pandas DataFrame, and printing it out to get a sense for it.

[2]:
```
all_data = pd.read_csv('rat_texture_task.csv')
all_data
```

[2]:

	rat_ID	rat_batch	experiment_phase	learned_statistic	swapped_statistic	session	trial	statistic_intensity	success
0	1	2	1	1	NaN	1	30	0.85	1
1	1	2	1	1	NaN	1	34	0.00	1
2	1	2	1	1	NaN	1	35	0.85	1
3	1	2	1	1	NaN	1	36	0.00	1
4	1	2	1	1	NaN	1	37	0.00	1
...	...	...	...	...					
1336665	6	1	4	4					
1336666	6	1	4	4					
1336667	6	1	4	4					
1336668	6	1	4	4					
1336669	6	1	4	4					

```
...                    ...      ...    ...                  ...       ...
1336665                0.0      181    275                 0.00         0
1336666                3.0      181    277                 0.86         1
1336667                3.0      181    278                 0.23         1
1336668                0.0      181    279                 0.00         0
1336669                0.0      181    280                 0.00         0

[1336670 rows x 9 columns]
```

As a cosmetic enhancement, we rename the values of the `learned_statistic` column from 1, 2, 3, 4 to gamma, beta, theta, and alpha respectively.

[3]:
```
statistic_names = np.array(['gamma', 'beta', 'theta', 'alpha'])
all_data['learned_statistic'] = statistic_names[all_data['learned_statistic'].
 ↪to_numpy()-1]
```

1.2 Transform data

Now we create a new dataframe containing average performance for each rat, statistic, and level for all data from experiment phase 3.

[4]:
```
fraction_correct = all_data\
    .query("experiment_phase == 3")\
    .groupby(["rat_batch", "rat_ID", "learned_statistic",␣
 ↪"statistic_intensity"], as_index=False)[["success"]]\
    .aggregate('mean')
fraction_correct
```

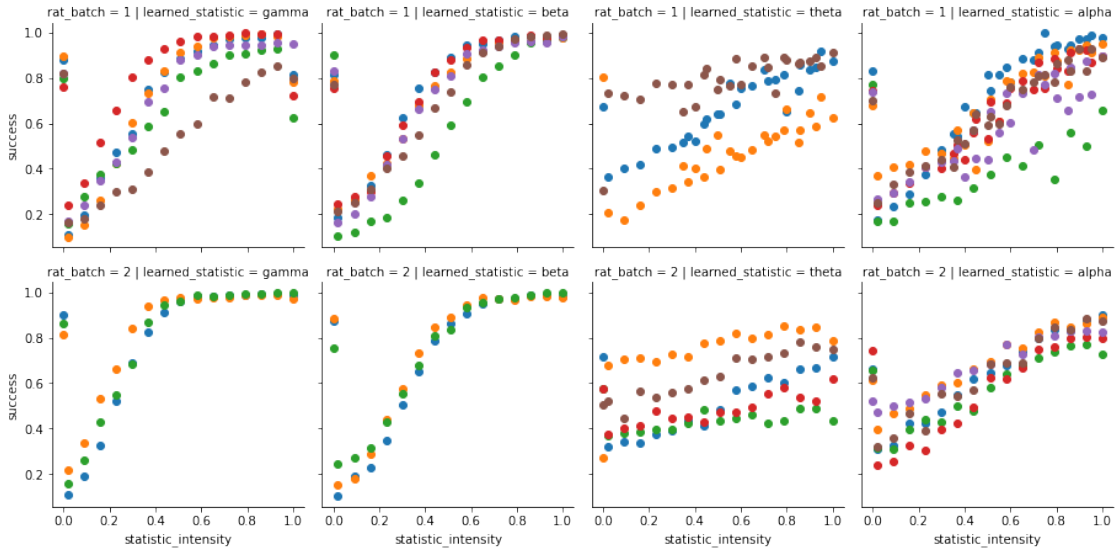
[4]:
```
     rat_batch  rat_ID learned_statistic  statistic_intensity   success
0            1       1             alpha                 0.00  0.831191
1            1       1             alpha                 0.02  0.174603
2            1       1             alpha                 0.09  0.236402
3            1       1             alpha                 0.16  0.289320
4            1       1             alpha                 0.23  0.373494
..         ...     ...               ...                  ...       ...
699          2       6             theta                 0.72  0.718274
700          2       6             theta                 0.79  0.733918
701          2       6             theta                 0.86  0.783862
702          2       6             theta                 0.93  0.757396
703          2       6             theta                 1.00  0.750000

[704 rows x 5 columns]
```

1.3 Visualize data

We can now give an example of generating exploratory plots in a flexible way. Here we separate rats by batch: panels in the top row contain rats in batch 1, while panels in the bottom row contain rats in batch 2. Each column of panels shows data about a different statistic. Within each panel, the X axis is statistic intensity, and the Y axis is the average performance of a rat at that statistic level. Colors are used to distinguish between different rats. Note that we have to specify the order in which the columns (statistics) should be arranged, because if we don't do that they will be sorted in alphabetical order (alpha, beta, gamma, theta), which is typically not how we sort them.

[5]:
```
grid = sns.FacetGrid(fraction_correct, col="learned_statistic", row="rat_batch",␣
 ↪hue="rat_ID", col_order=statistic_names, height=3.2)
grid.map(plt.scatter, 'statistic_intensity', 'success');
```



[6]:
```
%load_ext watermark
%watermark --iversion
```

```
numpy     : 1.20.0
seaborn   : 0.11.1
pandas    : 1.2.1
matplotlib: 3.3.4
```