data-vis-report_2

April 13, 2018

1 Data Visualization Report 2

```
Date:    13 April 2018
Theme:   Explorative visual and statistical analysis
Authors: Luca Steyn, Renier Botha
```

```
In [41]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns

         %matplotlib inline
         %load_ext autoreload
         %autoreload 2
         sns.set()

         import sys
         sys.path.append('../')
         import data_utils as util

         import warnings
         warnings.filterwarnings('ignore')
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

1.1 Load Data

Load the data with BASMI and Demographics information.

Also show a small preview to get an intuition for what we are working with...

```
In [42]: df = pd.read_csv('../data/combined_full.csv', index_col=0, parse_dates=[1])
         df.head()
```

```
Out[42]:                  Date  CRS  TWS  LSFS  LFS  IMS   BS   Drug  gender  \
         patient_id
```

```
40          1995-05-09    3    1     6    5    3  3.6  False  Female
40          1995-06-01    3    1     8    5    3  4.0  False  Female
40          1995-06-12    2    1     5    3    2  2.6  False  Female
40          1995-11-02    1    1     3    4    2  2.2  False  Female
40          1996-05-02    2    1     4    3    2  2.4  False  Female

             Age at diagnosis patient_hla_bUnknown7_id   EIBP  \
patient_id
40                 46.255989                Positive  False
40                 46.255989                Positive  False
40                 46.255989                Positive  False
40                 46.255989                Positive  False
40                 46.255989                Positive  False

           patient_condition_subtype  Age  Age_cat
patient_id
40                                AS   48      5.0
40                                AS   48      5.0
40                                AS   48      5.0
40                                AS   48      5.0
40                                AS   49      5.0
```

1.2 Split BASMI data into different cohorts:

- Treatment: Data of patients who while and after they underwent **any** treatment
- No Treatment: Data of patients who **have not** received any treatment

To be clear, consider the following example: Patient 40 entered the study and took no biologics and also did not undergo rehab for the first 12 years. However, after 12 years the patient started using a biologic drug. This means that the first 12 years of measurements of patient 40 will be added to the "no treatment" cohort, and the latter will be added to the "treatment" cohort.

For this report, we merely load the data from disk

```
In [43]: treatment_df = pd.read_csv('../data/treatment_cohort.csv', index_col=0, parse_dates=[1]
         no_treatment_df = pd.read_csv('../data/no_treatment_cohort.csv', index_col=0, parse_dat

         print('Full dataset shape: {}'.format(df.shape))
         print('No-Treatment Dataset shape: {}'.format(no_treatment_df.shape))
         print('Treatment Dataset shape: {}'.format(treatment_df.shape))
```

```
Full dataset shape: (14436, 15)
No-Treatment Dataset shape: (11228, 15)
Treatment Dataset shape: (3208, 15)
```

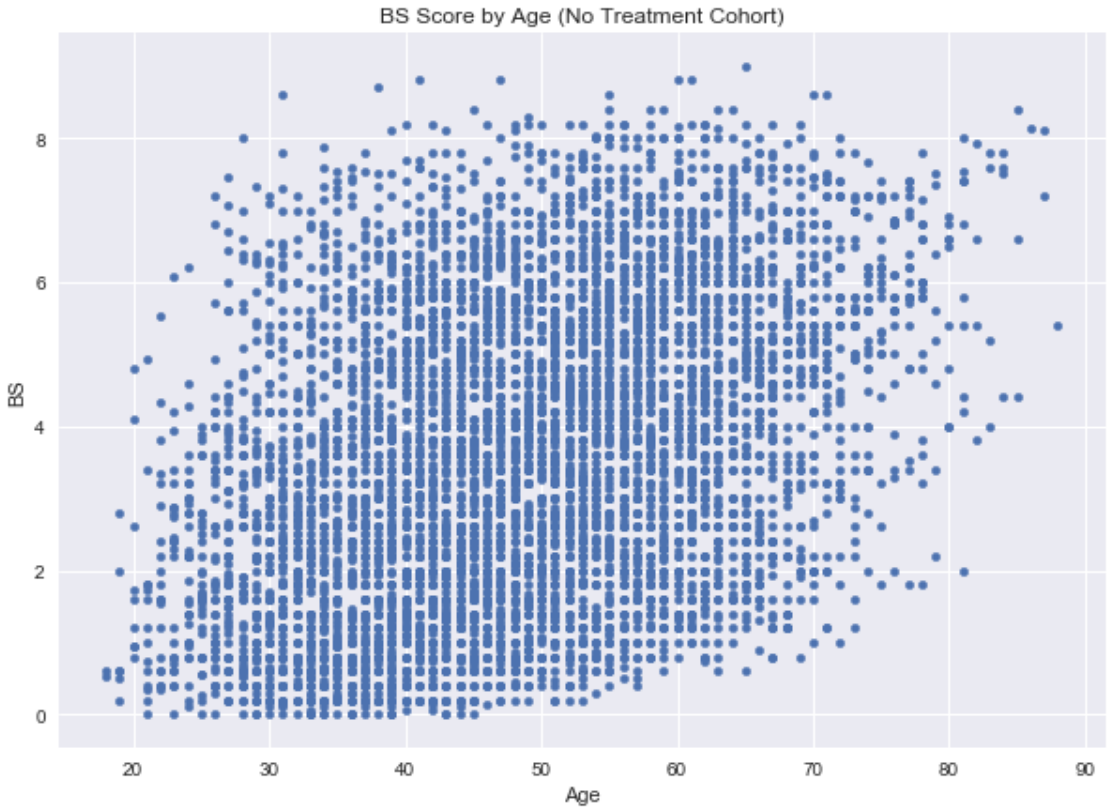
1.3 Show BS Score By Age

Goal: The main objective is to see if there exists any *trend* between BS score and Age.

We will approach this by only considering the "No Treatment" cohort, and plotting the BS score by Age for all patients. The *mean* BS score is taken for every year, to get one score per year for each patient.

```
In [59]: agg_dict = {'BS': 'mean', 'Age': min}
         agg = no_treatment_df.groupby(['patient_id','Age']).agg(agg_dict)
         agg.reset_index(level=1, drop=True)
         agg.plot(x='Age', y='BS', kind='scatter', figsize=(10,7),
                  title='BS Score by Age (No Treatment Cohort)')
```

```
Out[59]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1bc90080>
```

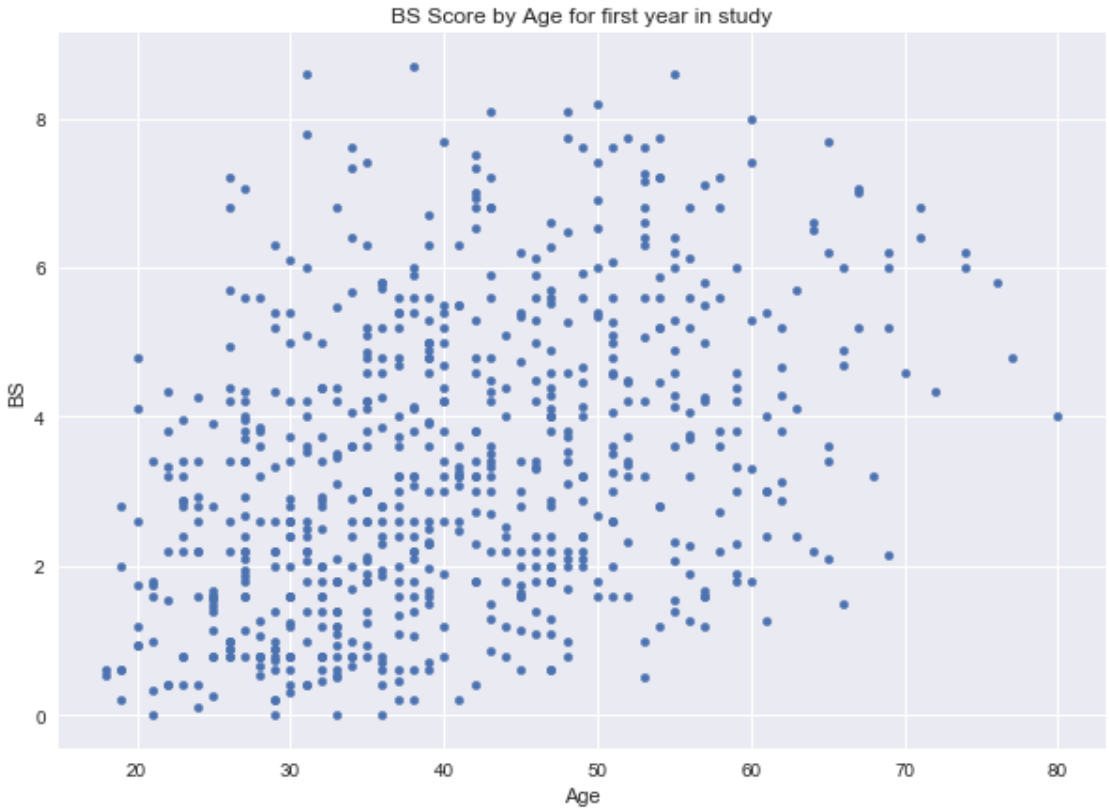


There seems to be a positive trend, but with *a lot* of variation. However, this figure contains subsequent measurements from every patient...

Show BS by Age for first year in study Now take only the BS score of the first year in the study for each patient in the "No Treatment" cohort

```
In [45]: # Get only the first entry for each patient --> Age at entry
         agg_at_entry = agg.groupby('patient_id').apply(lambda x: x.iloc[0])
         agg_at_entry.plot(x='Age',y='BS', kind='scatter', figsize=(10,7),
                           title='BS Score by Age for first year in study')
```

Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x1a19e720b8>



1.3.1 Discussion

There still seems to be positive trend, but with lots of variance.

The goal with this step was to compare our patient BS scores with the *reference intervals* of patients in a study that Raj sent out (link TBC). However, we were unsure how to actually compare this data and would love some feedback about how we can improve this.

1.4 Explore Waiting Times

Goal: Calculate and visualize the time patients spent between measurements.

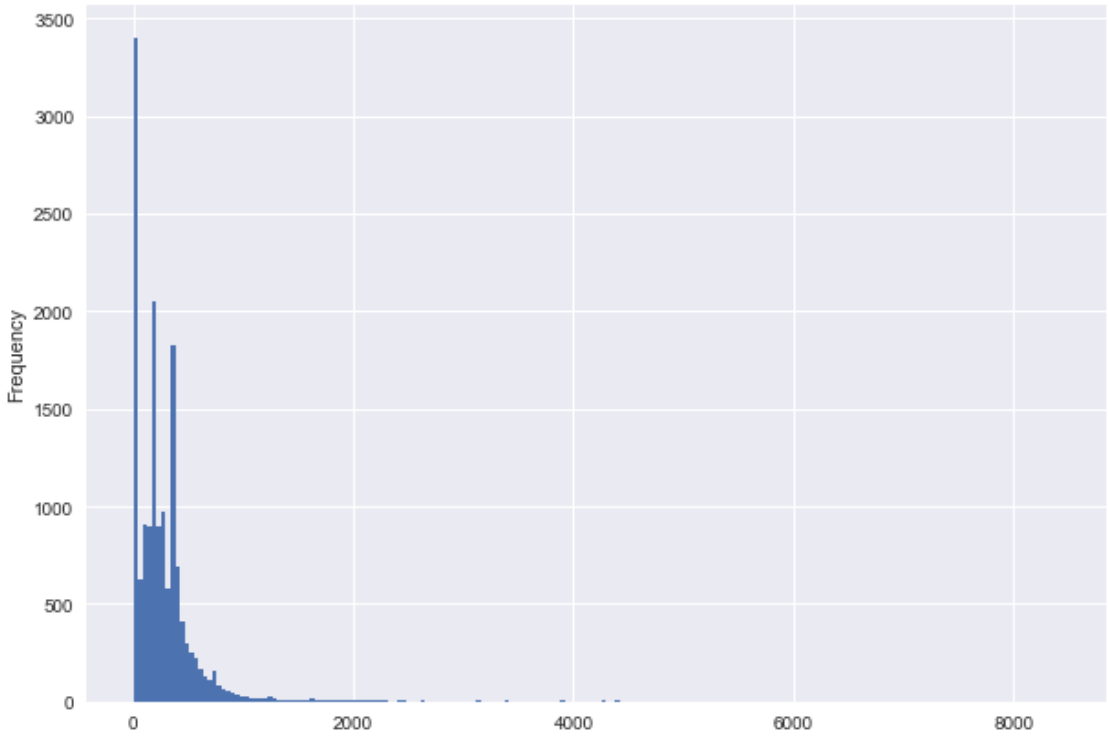
```
In [46]: def get_diff(df_):
             df = df_.copy()
             df['Date_diff'] = df['Date'].diff().dt.days
             df['BS_diff'] = df['BS'].diff()
             return df.iloc[1:] # return from 1-> to drop first nan

         diffs = basmi_df.groupby('patient_id').apply(get_diff)[['Date','Date_diff','BS_diff']]
         diffs.reset_index(level=0, drop=True, inplace=True)
```

```
# Plot the diffs
diffs['Date_diff'].plot(kind='hist', bins=200, figsize=(10,7))
print('\n\nA preview of the data sorted by Date_diff: (Note the value of Date_diff)\n')
diffs.sort_values(by='Date_diff').head()
```

```
A preview of the data sorted by Date_diff: (Note the value of Date_diff)
```

Out[46]:

patient_id	Date	Date_diff	BS_diff
7766	1996-09-12	1.0	0.0
3936	1989-02-18	1.0	0.0
3936	1989-01-31	1.0	0.0
1716	1987-11-13	1.0	-0.6
3824	1997-02-27	1.0	0.0

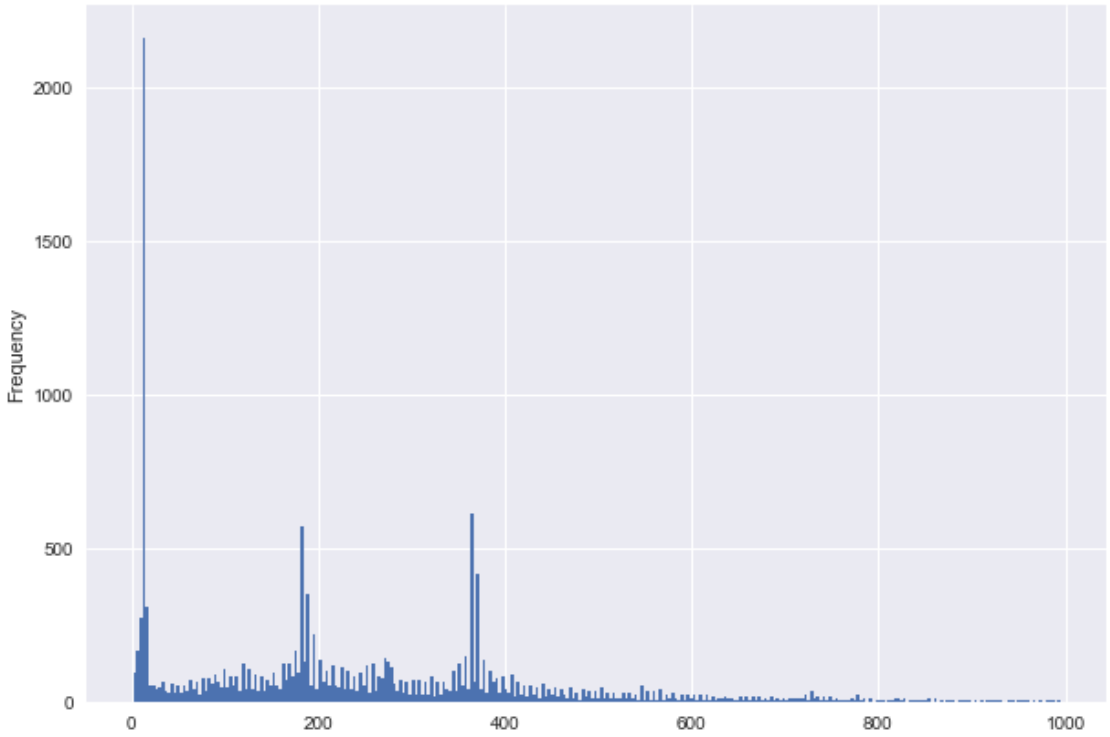


Interesting, an *overwhelming* number of patients had measurements 1 day apart, which was unexpected. @Raj any comment on that?

Lets see how the graph looks if we plot for all values greater than 1 and less than 1000:

In [47]:
```
filtered = diffs[(diffs['Date_diff'] > 1) & (diffs['Date_diff'] < 1000)]
filtered['Date_diff'].plot(kind='hist',bins=300, figsize=(10,7))
```

```
Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x10f652e10>
```



Some interesting peaks there...lets take a look at the actual frequencies of the top 10 intervals

```
In [58]: print('Top 10 occuring waiting times:')
         filtered['Date_diff'].value_counts().head(10)
```

```
Top 10 occuring waiting times:
```

```
Out[58]: 13.0     1380
         12.0      569
         364.0     401
         182.0     322
         371.0     314
         189.0     197
         14.0      182
         18.0      177
         11.0      162
         183.0     130
         Name: Date_diff, dtype: int64
```

Waiting Time Analysis: So there are 1380 patients who had measurements taken 13 days apart and 570 who were measured 12 days apart.

Knowing that rehab patients are measured 14 days apart, is it possible that these patients exited the rehab earlier and were measured 11/12/13 days apart instead of 14?

Another interesting thing: The 3rd highest frequency period is 364 days - exactly one year apart, the and the 4th is 182, which is half a year apart.

@Raj, were you expecting these insights?

2 t_test On Rehab patient data

Next we do a t-test on data from patients who underwent rehab...

Question Is there a *significant* difference in BS scores before and after rehab sessions?

To answer this question, we will run a "t-test" on the data from before and after patients underwent rehab.

- We will identify the rehab sessions by measurements spaced exactly 14 days apart.
- Then we will run t-tests with different assumptions, and testing these assumptions as we progress.

The end goal is run a non-parametric test, but we will get there by first running simpler tests, thus leading up to the NP test and ensuring that the results make sense.

NOTE For future work we could include measurements 11/12/13 days apart...

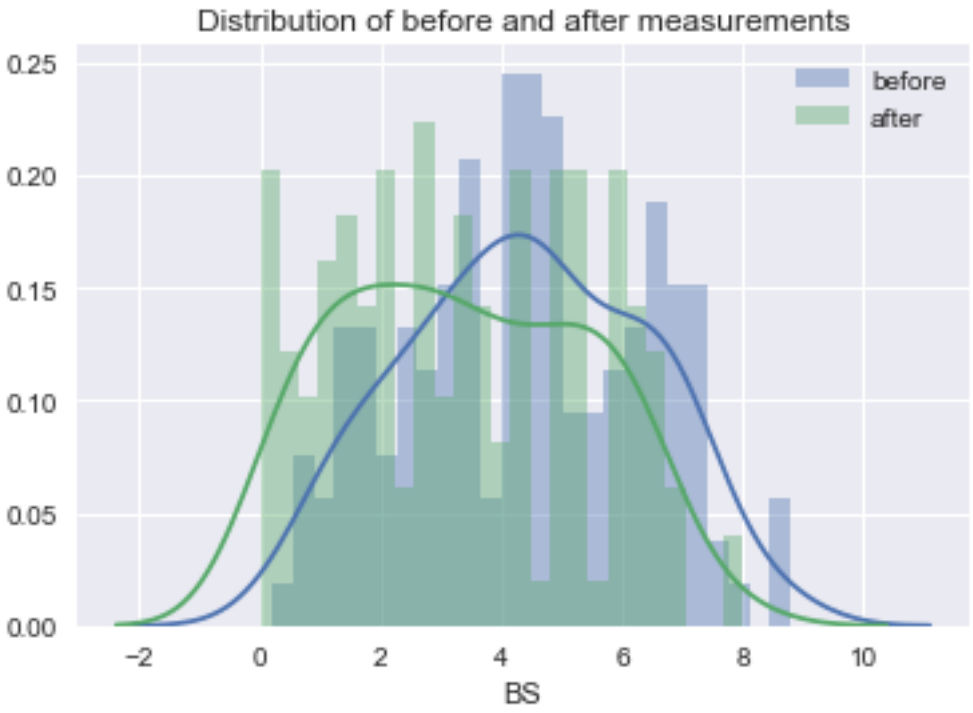
```
In [49]: # Load the rehab data
         rehab_df = pd.read_csv('../data/rehab_data.csv', index_col=0)

         # Split the data into the measurements before and after rehab
         before = rehab_df.iloc[::2]['BS']
         after = rehab_df.iloc[1::2]['BS']
         before = before[:-1]
```

2.0.1 Simple 2-sided t-test

Notice that the test statistic is positive (indicating before - after). Python always performs a two-sided test for symmetrical tests such as the t-test. Therefore, the p-value must be dividd by two to get the one-sided p-value.

```
In [50]: from scipy import stats

         t, p = stats.ttest_ind(before, after)

         print('t = {}'.format(t))
         print('p = {}'.format(p/2))

         fig = plt.figure()
         sns.distplot(before, bins=25, label='before')
         sns.distplot(after, bins=25, label='after')
         plt.title('Distribution of before and after measurements')
         plt.legend()
         plt.show()
```

```
t = 4.434264190752112
p = 6.448029677903826e-06
```



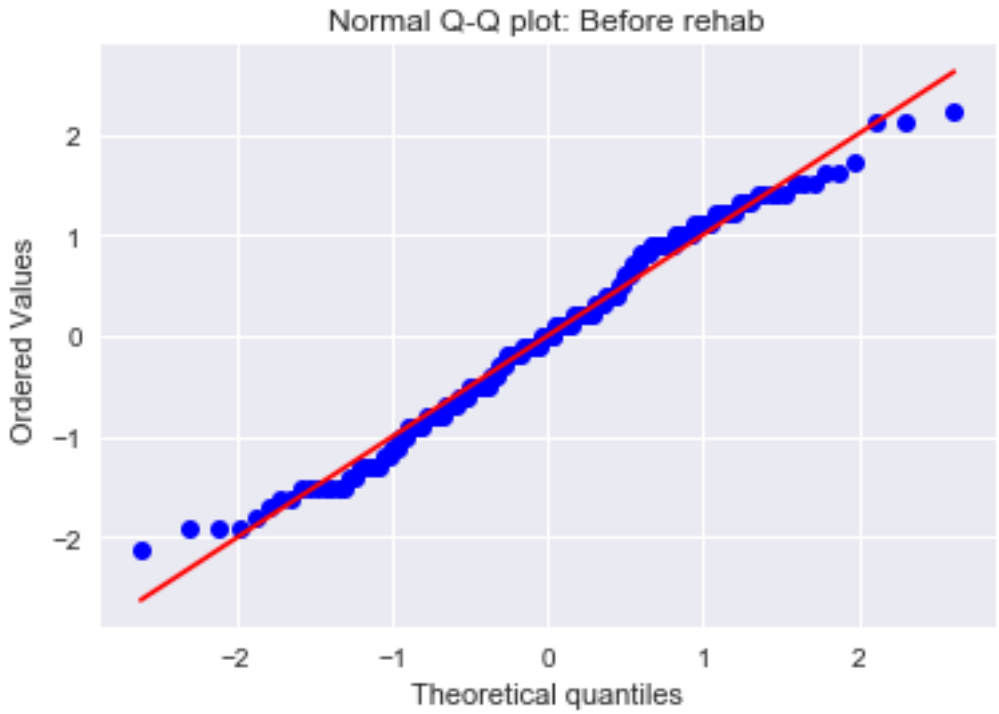
2.0.2 Check the assumptions of the t-test.

The assumptions are: - Both samples are normally distributed - Samples are independent

2.0.3 Visual check for normality: Q-Q plot

We expect a straight line through the origin if the data is normally distributed.

```
In [51]: Z_before = (before - np.mean(before))/np.std(before)
         stats.probplot(Z_before, dist='norm', plot=plt)
         plt.title("Normal Q-Q plot: Before rehab")
         plt.show()
```




```
In [52]: Z_after = (after - np.mean(after))/np.std(after)
         stats.probplot(Z_after, dist='norm', plot=plt)
         plt.title("Normal Q-Q plot: After rehab")
         plt.show()
```