2.1.4 Exploring the data

April 6, 2019

0.1 Exploring the data: Univariate

After defining the research question, your next step toward building a model is to get to know the individual variables through inspection of the data file itself, visualizations and univariate descriptive statistics ("univariate" means pertaining to one variable, rather than many). You'll want to answer questions such as: * How many variables do I have? * How many datapoints? * Which variables are continuous (having an infinite number of possible values, such as age) and which are categorical (having a limited number of values, such as gender). * Do any of my variables have a known probability distribution (normal, Poisson, Gamma, etc)? * Do I have missing data? If so, how much? * How much variance is there in each of my variables? If you've cleaned this dataset yourself, you'll probably already know the answers to most of these questions.

0.2 Exploring the data: Bivariate

Next, examine the pairwise relationships among your data ("bivariate", of course, meaning pertaining to two variables). For our example dataset we will use the European Social Survey, focusing on the data from the year 2012 and including Switzerland (CH) and the Czech Republic (CZ) only. You should be able to grab the csv here. The codebook is available here.

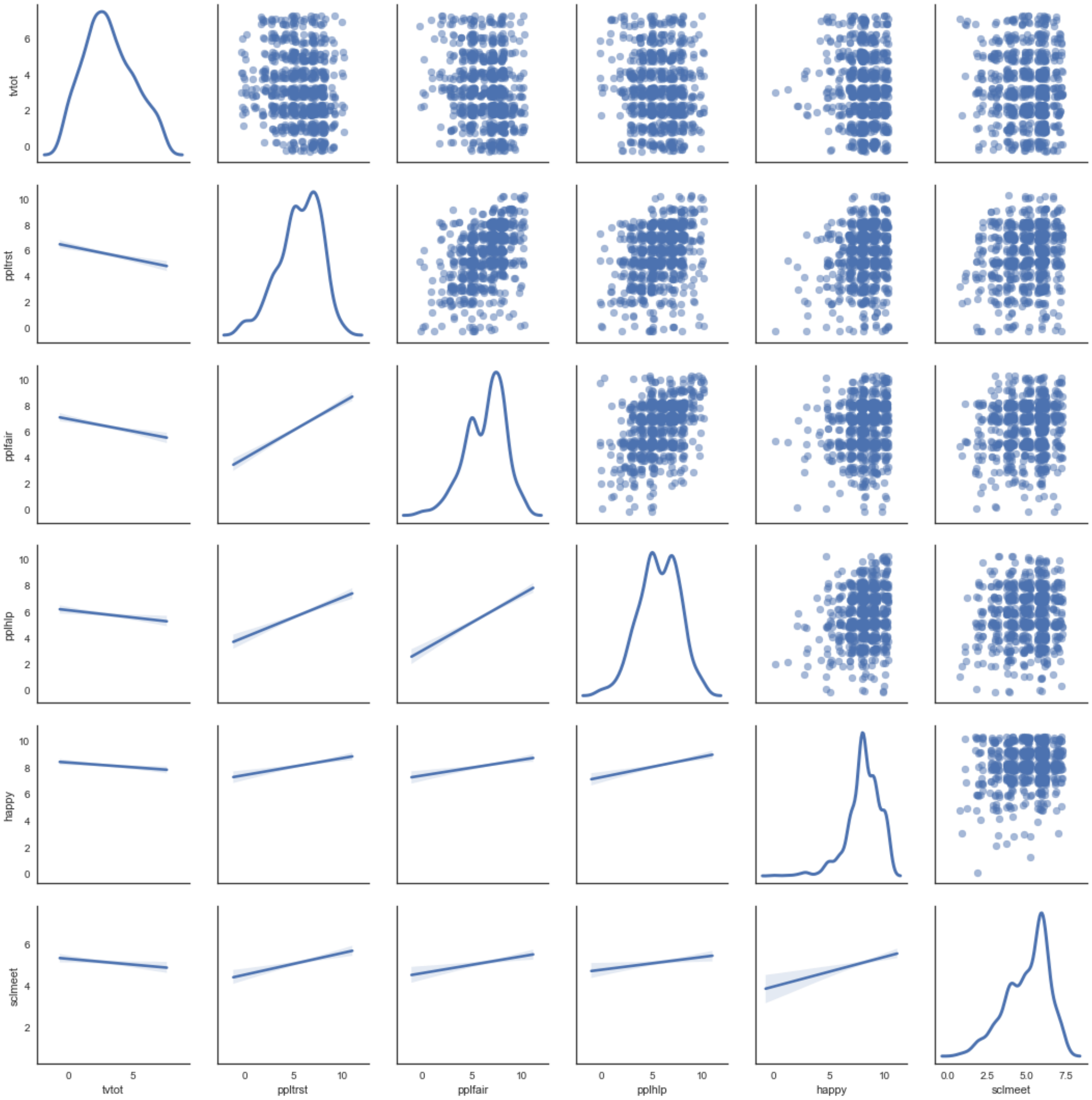
0.2.1 For two continuous variables:

- The scatterplot, which you've already encountered, represents the relationship between two continuous variables. For this sort of wide-scale dataset investigation, we can go a step further and create a scatterplot matrix that shows all the bivariate relationships in one plot made up of subplots.
- The correlation coefficient r is a numeric representation of the linear relationship between two continuous variables. Correlations range from -1 (as values in one variable go up, values in the other go down) to 0 (no relationship between the variables) to 1 (as values in one variable go up, values in the other go up as well).
- Look for: two-dimensional outliers that represent unusual combinations of values

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import scipy.stats as stats
```

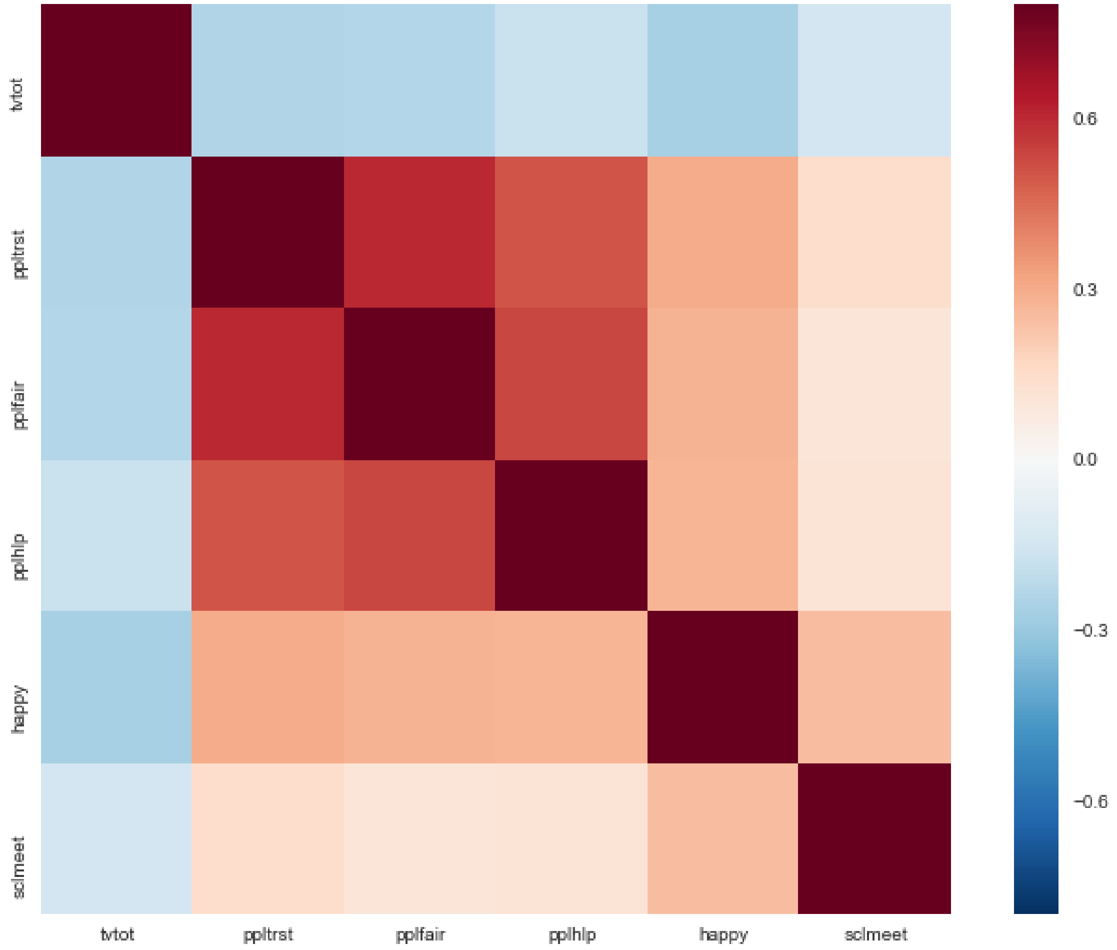
```
In [2]: # Make the scatterplot matrix
        # Set the default plot aesthetics to be prettier.
        sns.set_style("white")

        df = pd.read_csv('https://raw.githubusercontent.com/Thinkful-Ed/data-201-resources/mas

        # Take a subset of the data to make plots clearer.
        df_czch = df.loc[
            ((df['cntry'] == 'CZ') | (df['cntry'] == 'CH')) & (df['year'] == 6),
            ['cntry', 'tvtot', 'ppltrst', 'pplfair', 'pplhlp', 'happy', 'sclmeet']
        ]

        # Because the variables can only take integer values between 0 and 10, points
        # on the scatterplot will hide one another. To better see all the points, we
        # create a dataframe of random noise (or "jitter") and add it only when plotting.

        # Make a copy of the data to add jitter to and plot.
        df_jittered = df_czch.loc[:, 'tvtot':'sclmeet'].dropna()
        # Making the random noise.
        jitter = pd.DataFrame(
            np.random.uniform(-.3, .3, size=(df_jittered.shape)),
            columns=df_jittered.columns
        )
        # Combine the data and the noise.
        df_jittered = df_jittered.add(jitter)

        # Declare that you want to make a scatterplot matrix.
        g = sns.PairGrid(df_jittered.dropna(), diag_sharey=False)
        # Scatterplot.
        g.map_upper(plt.scatter, alpha=.5)
        # Fit line summarizing the linear relationship of the two variables.
        g.map_lower(sns.regplot, scatter_kws=dict(alpha=0))
        # Give information about the univariate distributions of the variables.
        g.map_diag(sns.kdeplot, lw=3)
        plt.show()

        #Some warnings will show up below because the plot does not include a legend.
```

```
/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5/site-packages/matplotlib/axes/
  warnings.warn("No labelled objects found. "
```



```
In [3]: # Make the correlation matrix.
        corrmat = df_czch.corr()
        print(corrmat)

        # Set up the matplotlib figure.
        f, ax = plt.subplots(figsize=(12, 9))

        # Draw the heatmap using seaborn.
        sns.heatmap(corrmat, vmax=.8, square=True)
        plt.show()
        # Heatmaps are a great way to get a quick visual read on a big correlation matrix.
        # Try to match the correlation numbers to the scatterplots above to get a
        # sense of what each magnitude of correlation can look like.
```

```
            tvtot   ppltrst   pplfair    pplhlp     happy   sclmeet
tvtot    1.000000 -0.240432 -0.233280 -0.176517 -0.265293 -0.144105
ppltrst -0.240432  1.000000  0.605589  0.500378  0.297432  0.148949
pplfair -0.233280  0.605589  1.000000  0.532927  0.275204  0.105886
pplhlp  -0.176517  0.500378  0.532927  1.000000  0.269627  0.109383
happy   -0.265293  0.297432  0.275204  0.269627  1.000000  0.244289
sclmeet -0.144105  0.148949  0.105886  0.109383  0.244289  1.000000
```

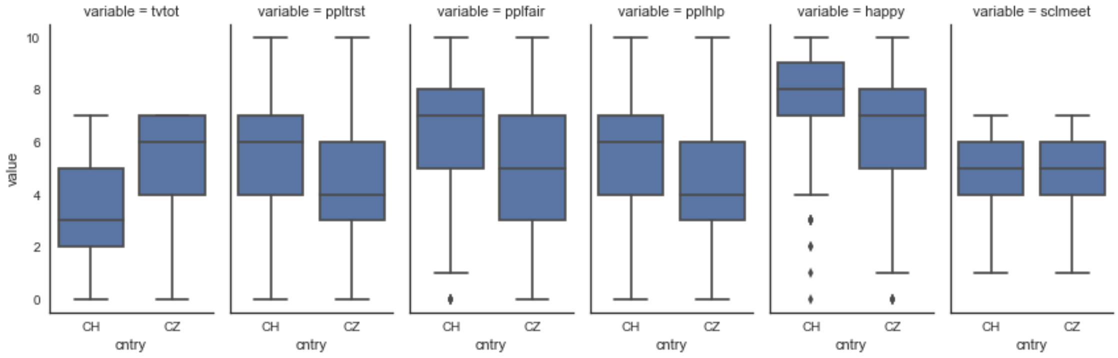


0.2.2 For a continuous and a categorical pair:

- Box, violin, or similar plots estimate the value of the continuous variable for each value of the categorical variable.
- Descriptive statistics, including estimates of central tendency and variance, for the continuous variable at each level of the categorical variable.

- T-tests and ANOVA
- Look for: Outliers in each group, very small groups

```
In [4]: # Plot all the variables with boxplots for each country.

        # Restructure the data so we can use FacetGrid rather than making a boxplot
        # for each variable separately.
        df_long = df_czch
        df_long = pd.melt(df_long, id_vars=['cntry'])

        g = sns.FacetGrid(df_long, col="variable", size=4, aspect=.5)
        g = g.map(sns.boxplot, "cntry", "value")
        plt.show()

        # Descriptive statistics by group.
        print(df_czch.groupby('cntry').describe())

        # Test whether group differences are significant.
        for col in df_czch.loc[:,'tvtot':'sclmeet'].columns:
            print(col)
            print(stats.ttest_ind(
                df_czch[df_czch['cntry'] == 'CZ'][col].dropna(),
                df_czch[df_czch['cntry'] == 'CH'][col].dropna()
            ))
```

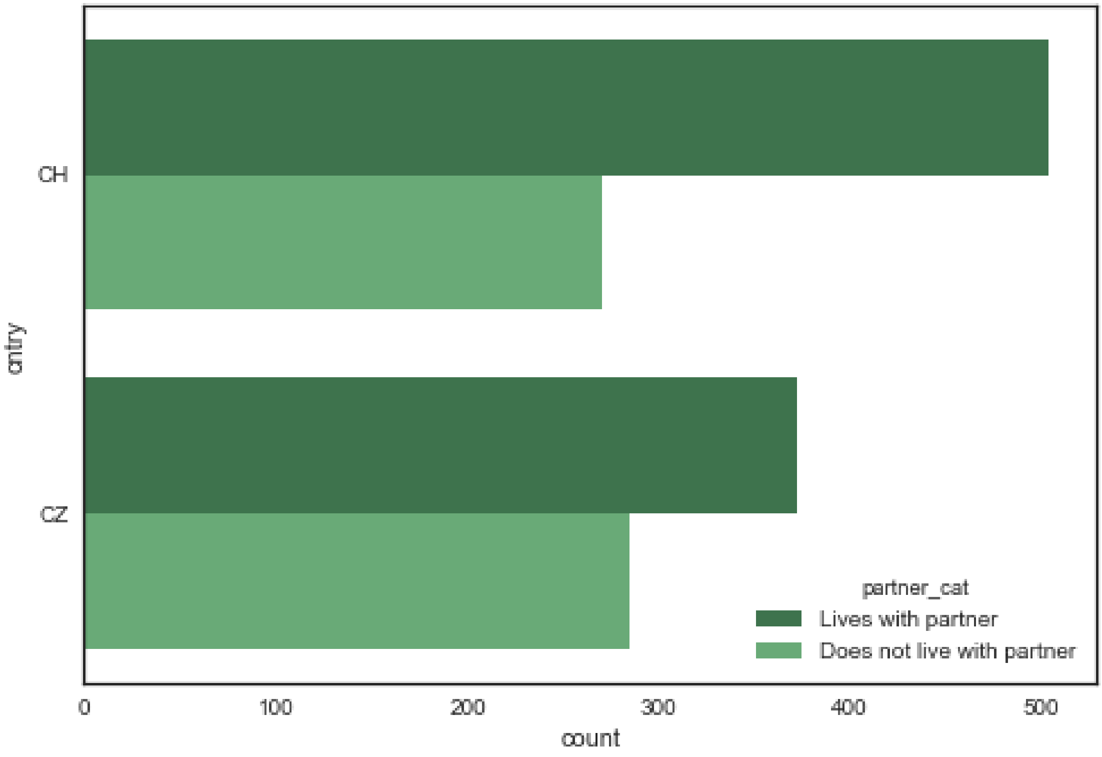


```
                  happy     pplfair      pplhlp     ppltrst     sclmeet  \
cntry
CH    count  770.000000  770.000000  771.000000  773.000000  770.000000
      mean     8.088312    6.350649    5.726329    5.677878    5.080519
      std      1.437920    1.964352    2.004947    2.130701    1.339122
      min      0.000000    0.000000    0.000000    0.000000    1.000000
      25%      7.000000    5.000000    4.000000    4.000000    4.000000
      50%      8.000000    7.000000    6.000000    6.000000    5.000000
      75%      9.000000    8.000000    7.000000    7.000000    6.000000
      max     10.000000  10.000000  10.000000  10.000000   7.000000
CZ    count  646.000000 649.000000 654.000000 651.000000 655.000000
      mean     6.770898   4.959938   4.503058   4.362519   4.648855
      std      2.053582   2.244664   2.285011   2.409738   1.667845
      min      0.000000   0.000000   0.000000   0.000000   1.000000
      25%      5.000000   3.000000   3.000000   3.000000   4.000000
      50%      7.000000   5.000000   4.000000   4.000000   5.000000
      75%      8.000000   7.000000   6.000000   6.000000   6.000000
      max     10.000000  10.000000  10.000000  10.000000   7.000000

                  tvtot
cntry
CH    count  772.000000
      mean     3.141192
      std      1.913706
      min      0.000000
      25%      2.000000
      50%      3.000000
      75%      5.000000
      max      7.000000
CZ    count  653.000000
      mean     5.142420
      std      1.851775
      min      0.000000
      25%      4.000000
      50%      6.000000
      75%      7.000000
      max      7.000000
tvtot
Ttest_indResult(statistic=19.962218108197227, pvalue=2.3215763985820789e-78)
ppltrst
Ttest_indResult(statistic=-10.928900558029875, pvalue=9.3940290175498276e-27)
pplfair
Ttest_indResult(statistic=-12.444422221030575, pvalue=8.1706858237367176e-34)
pplhlp
Ttest_indResult(statistic=-10.762634615325524, pvalue=5.0297991350713549e-26)
happy
Ttest_indResult(statistic=-14.142750296144351, pvalue=1.4274009995900834e-42)
sclmeet
Ttest_indResult(statistic=-5.4169092111441302, pvalue=7.1108682182128696e-08)
```

Judging by the boxplots, the 'happy' variable has some potential outliers for Switzerland that we should keep an eye on. The two countries appear to differ significantly for all the variables measured. ### For two categorical variables: * A barplot, with the height of each bar indicating the number of datapoints in each combination of levels of the two variables * A table of counts, giving the number of datapoints for each combination of levels of the two variables * Chi-square tests that indicate whether one combination of levels is significantly smaller or larger than the rest.

* Look for: Subgroups with very small counts relative to the other subgroups

```
In [5]: # Add a second categorical variable.
        df_2cat = df.loc[
            ((df['cntry'] == 'CZ') | (df['cntry'] == 'CH')) & (df['year'] == 6),
            ['cntry','partner']
        ]

        # Create a text variable so that output is easier to interpret.
        df_2cat['partner_cat'] = None
        df_2cat.loc[df_2cat['partner'] == 1, 'partner_cat'] = 'Lives with partner'
        df_2cat.loc[df_2cat['partner'] == 2, 'partner_cat'] = 'Does not live with partner'

        # Plot counts for each combination of levels.
        sns.countplot(y="cntry", hue="partner_cat", data=df_2cat, palette="Greens_d")
        plt.show()

        # Table of counts
        counttable = pd.crosstab(df_2cat['cntry'], df_2cat['partner_cat'])
        print(counttable)

        # Test will return a chi-square test statistic and a p-value. Like the t-test,
        # the chi-square is compared against a distribution (the chi-square
        # distribution) to determine whether the group size differences are large
        # enough to reflect differences in the population.
        print(stats.chisquare(counttable, axis=None))
```



```
partner_cat  Does not live with partner  Lives with partner
cntry
CH                                  270                 503
CZ                                  284                 372
Power_divergenceResult(statistic=96.399580125962217, pvalue=9.2365935866454386e-21)
```

It appears that in Switzerland, a higher proportion of people live with a partner than in the Czech Republic.

0.3 Interpreting pairwise plots and statistics

These pairwise plots and statistics tell you important things: * Which of your variables are redundant with one another, and which provide unique information. It appears that the three 'ppl' variables (trust, fairness, and helpfulness) cluster together, with correlations at or above .5. The other continuous variables are more weakly correlated and thus less redundant. The country respondents live in appears to share significant variance with all the other variables. * Unusual two-dimensional datapoints that could skew analyses. These include outliers, subgroups with very small variance relative to other subgroups, and subgroups with very small Ns. Aside from the previously-identified potential outliers in "happy" for Switzerland, no other concerns leapt out in this overview.

0.4 Starting to focus in

At this point, you should decide on a strategy for dealing with unusual two-dimensional datapoints. This could include discarding the data, winsorizing, or collapsing very small or very similar subgroups into one larger category. Then it's time to move on to feature engineering.