Seaborn Pack

Instructor Notes, YB, RIT

Seaborn Pack for Visualization and EDA

Table of Contents

Import Essential Packs and Data

- This tutorial will introduce many packs. Some will have bugs and won't work well. No worries. Try to fix, share on the Slack. It is ok to skip some. let me know if you know better or useful ones. Enjoy! YB

In [ ]:
```
# import these two packs
import pandas as pd
import seaborn as sns
```

In [ ]:
```
# import data (our data set, not sns's dataset!)
df_titanic = pd.read_csv('train-titanic.csv',
                         index_col='PassengerId') #comma-seperated values=csv
```

In [ ]:
```
# Raw data: X in N by p
# N=891, p=11
# Dimension

df_titanic.shape
```

In [ ]:
```
# Summary table as EDA
# Univariate summary

df_titanic.describe(include='all')
```

Seaborn Pack in Brief

- Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.
- Training: I will go over briefly
    1. Overview of Seaborn Pack:Main resource, Functions in the pack AND Gallery
        - “relational”, “distributional”, and “categorical” modules ==> see the figure
        - “axes-level” and “figure-level” functions ==> interface with matplotlib through a seaborn object, usually a FacetGrid
    2. Many data sets are available: visit
- For example, displot() is the figure-level function for the distributions module. Its default behavior is to draw a histogram, using the same code as histplot() behind the scenes.
- Optional: How to do specifying figure sizes? Please visit the links to learn more.

Goals in EDA in General (so think of the aspects that relate to viz)

1. to describe/monitor the data:
    - column-wise
    - row-wise
    - missingness
    - data types
    - weird, unusuals, doublicates
2. to summarize and plot the data to see the associations or patterns:
    - raw data
    - univariate summaries
    - bivariate summaries
        - associations
        - correlations
        - patterns
    - multivariate summaries
        - mix of data types-related plots, summaries
    - advanced and interactive
        - dynamic, interactive,
        - raw, aggregated, projected

Thought Question: Can a pack do all these?

Seaborn Pack with Examples

Let's go over some of the chunks found on the web and Gallery. I listed below some.

In [ ]:
```
# Here's an example of what seaborn can do:

# Import seaborn
import seaborn as sns

# Apply the default theme
# Seaborn uses matplotlib to draw its plots
```

```
# This affect show all matplotlib plots look
sns.set_theme()
```

In [ ]:
```
# Load dataset
tips = sns.load_dataset('tips')
tips.shape
```

In [ ]:
```
# Let's run 6 codes useful after importing a dataset
# Use Pandas
tips.info()
tips.head()
tips.tail()
tips.describe(include='all')

#try others
```

In [ ]:
```
# The goal is to explore when to get good tips. When are people generous in restraurants?
# Scatterplot by including a linear regression model (and its uncertainty) using lmplot()
# Correlation between total bill and tips
sns.lmplot(data=tips,
           x="total_bill",
           y="tip"
           ) #add col="time", hue="smoker"
```

In [ ]:
```
# Informative distributional summaries
# Dist of tips

sns.displot(data=tips,
            x="tip",
            kde=True) # include col="time", kde=True (density estimate)
```

In [ ]:
```
# Specialized plots for categorical data

sns.catplot(data=tips,
            kind="bar", #"swarm", "strip", "swarm", "box", "violin", "boxen", "point", "bar", or "count"
            x="day", y="tip",
            hue="smoker")
```

In [ ]:
```
# Or you could show only the mean value and its confidence interval within each nested category:

sns.catplot(data=tips, kind="bar", x="day", y="total_bill", hue="smoker")
```

In [ ]:
```
# catplot: kernel density estimation to represent the underlying distribution

sns.catplot(data=tips, kind="violin", x="day", y="total_bill",
            hue="smoker", split=True)
```

In [ ]:
```
# Composite views onto multivariate datasets

sns.jointplot(data=tips, x="tip", y="total_bill")
```

In [ ]:
```
# Composite views onto multivariate datasets

sns.jointplot(data=tips, x="tip", y="total_bill", hue='day')
```

In [ ]:
```
# Composite views onto multivariate datasets
# Use for large data: hex, kde in kind

sns.jointplot(data=tips, x="tip", y="total_bill", kind='hex') #kind='kde', 'hist', 'hex'
```

In [ ]:
```
# Joint and marginal distributions for all pairwise relationships and for each variable

sns.pairplot(data=tips, hue="day") #hue="smoker" ategorical variable
```

Relplot

- The function relplot() is named that way because it is designed to visualize many different statistical relationships.
- Let's create a visualization with relplot:

In [ ]:
```
sns.relplot(
    data=tips,
    x="tip", y="total_bill", hue="day") #kind='line' for time-series
```

In [ ]:
```
# Load an example dataset
tips = sns.load_dataset("tips")

# More complex relplot example
sns.relplot(
    data=tips,
    x="tip", y="total_bill",
    hue="smoker",
```

```
    col="time",
    style="smoker", size="size")
```

In [ ]:
```
# The relplot() function has a convenient kind parameter that lets you easily switch to this alternate repres

dots = sns.load_dataset("dots")
sns.relplot(
    data=dots, kind="line",
    x="time", y="firing_rate", col="align",
    hue="choice", size="coherence", style="choice",
    facet_kws=dict(sharex=False),
)
```

References

- Seaborn pack
- Seaborn tutorial
- Seaborn tutorial with Dataset