DataViz-Tutorial-DSSG

September 28, 2017

1 Introduction to Python for Data Visualization

DSSG Conference 2017 Alex C. Engler

1.1 Introduction

In this module, you will learn to quickly and flexibly make a wide series of visualizations for exploratory data analysis and communicating to your audience. This module contains a practical introduction to data visualization in Python and covers important rules that any data visualizer should follow.

1.2 The Data is Available Here: http://bit.ly/2wZEHD5

1.3 Learning Objectives

- Learn critical rules about data visualization (using the correct graph types, correctly labeling all visual encodings, properly sourcing data).
- Become familiar with a core base of data visualization tools in Python - specifically matplotlib and seaborn.
- Start to develop the ability to conceptualize what visualizations are going to best reveal various types of patterns in your data.
- Learn more about Illinois administrative data with exploratory analyses.

```
In [50]: # Import packages:
         import pandas as pd

         import matplotlib as mplib
         import matplotlib.pyplot as plt
         import seaborn as sns

         # so images get plotted in the notebook
         %matplotlib inline
```

```
In [6]: # Read in the Data
        # Department of Transportation's (DoT) Fatality Accident Reporting System (FARS) Data:
```

```
In [6]: acc = pd.read_csv("./fars_accident.csv")
        acc.shape
```

```
Out[6]: (62222, 58)
```

1.4 About This Data

Traffic fatalities are on the rise in the United States. 40,200 people died in 2016 due to motor vehicle accidents, according to the National Safety Council. This is a 14 percent increase in the last two years, and the state of Illinois is fairing even worse.

Various experts attribute this increases to distracted driving, more speeding, lax enforcement of seatbelt laws, and a better economy (leading to more driving), but there is little consensus. Even before the dire 2016 numbers, the Obama White House and DoT even issued a call to action to analyze this data and help mitigate the problem.

```
In [9]: # Each row of data is one fatal accident in 2014 or 2015.
        # Look at column names to see what information is available:
        list(acc)
```

```
Out[9]: ['Unnamed: 0',
         'StateName',
         'CountyName',
         'StateFIPSCode',
         'CountyFIPSCode',
         'STATE',
         'ST_CASE',
         'VE_TOTAL',
         'VE_FORMS',
         'PVH_INVL',
         'PEDS',
         'PERNOTMVIT',
         'PERMVIT',
         'PERSONS',
         'COUNTY',
         'CITY',
         'DAY',
         'MONTH',
         'YEAR',
         'DAY_WEEK',
         'HOUR',
         'MINUTE',
         'NHS',
         'RUR_URB',
         'FUNC_SYS',
         'RD_OWNER',
         'ROUTE',
         'TWAY_ID',
         'TWAY_ID2',
         'MILEPT',
```

```
        'LATITUDE',
        'LONGITUD',
        'SP_JUR',
        'HARM_EV',
        'MAN_COLL',
        'RELJCT1',
        'RELJCT2',
        'TYP_INT',
        'WRK_ZONE',
        'REL_ROAD',
        'LGT_COND',
        'WEATHER1',
        'WEATHER2',
        'WEATHER',
        'SCH_BUS',
        'RAIL',
        'NOT_HOUR',
        'NOT_MIN',
        'ARR_HOUR',
        'ARR_MIN',
        'HOSP_HR',
        'HOSP_MN',
        'CF1',
        'CF2',
        'CF3',
        'FATALS',
        'DRUNK_DR',
        'ROAD_FNC']
```

In [83]:
```
## Look at the first 10 rows
print(acc)
```

```
       Unnamed: 0 StateName  CountyName  StateFIPSCode  CountyFIPSCode  STATE  \
0               1   Alabama      Walker              1             127      1
1               2   Alabama   Limestone              1              83      1
2               3   Alabama     Bullock              1              11      1
3               4   Alabama        Dale              1              45      1
4               5   Alabama        Dale              1              45      1
5               6   Alabama    Randolph              1             111      1
6               7   Alabama     Madison              1              89      1
7               8   Alabama   Jefferson              1              73      1
8               9   Alabama      Shelby              1             117      1
9              10   Alabama     Colbert              1              33      1
10             11   Alabama   Limestone              1              83      1
11             12   Alabama    Marshall              1              95      1
12             13   Alabama       Macon              1              87      1
13             14   Alabama     Bullock              1              11      1
14             15   Alabama      Walker              1             127      1
```

15	16	Alabama	Lee	1	81	1	
16	17	Alabama	Mobile	1	97	1	
17	18	Alabama	Autauga	1	1	1	
18	19	Alabama	Talladega	1	121	1	
19	20	Alabama	Talladega	1	121	1	
20	21	Alabama	St Clair	1	115	1	
21	22	Alabama	Walker	1	127	1	
22	23	Alabama	Bullock	1	11	1	
23	24	Alabama	Barbour	1	5	1	
24	25	Alabama	Tuscaloosa	1	125	1	
25	26	Alabama	Jefferson	1	73	1	
26	27	Alabama	Baldwin	1	3	1	
27	28	Alabama	Autauga	1	1	1	
28	29	Alabama	Houston	1	69	1	
29	30	Alabama	Chilton	1	21	1	
...	...	...	...	...	...	...	
62192	62193	Wyoming	Park	56	29	56	
62193	62194	Wyoming	Big Horn	56	3	56	
62194	62195	Wyoming	Park	56	29	56	
62195	62196	Wyoming	Lincoln	56	23	56	
62196	62197	Wyoming	Fremont	56	13	56	
62197	62198	Wyoming	Sweetwater	56	37	56	
62198	62199	Wyoming	Big Horn	56	3	56	
62199	62200	Wyoming	Sweetwater	56	37	56	
62200	62201	Wyoming	Big Horn	56	3	56	
62201	62202	Wyoming	Laramie	56	21	56	
62202	62203	Wyoming	Laramie	56	21	56	
62203	62204	Wyoming	Laramie	56	21	56	
62204	62205	Wyoming	Johnson	56	19	56	
62205	62206	Wyoming	Laramie	56	21	56	
62206	62207	Wyoming	Lincoln	56	23	56	
62207	62208	Wyoming	Uinta	56	41	56	
62208	62209	Wyoming	Albany	56	1	56	
62209	62210	Wyoming	Campbell	56	5	56	
62210	62211	Wyoming	Laramie	56	21	56	
62211	62212	Wyoming	Sublette	56	35	56	
62212	62213	Wyoming	Natrona	56	25	56	
62213	62214	Wyoming	Campbell	56	5	56	
62214	62215	Wyoming	Fremont	56	13	56	
62215	62216	Wyoming	Fremont	56	13	56	
62216	62217	Wyoming	Goshen	56	15	56	
62217	62218	Wyoming	Big Horn	56	3	56	
62218	62219	Wyoming	Goshen	56	15	56	
62219	62220	Wyoming	Sweetwater	56	37	56	
62220	62221	Wyoming	Platte	56	31	56	
62221	62222	Wyoming	Sweetwater	56	37	56	

	ST_CASE	VE_TOTAL	VE_FORMS	PVH_INVL	...	ARR_HOUR	ARR_MIN	\

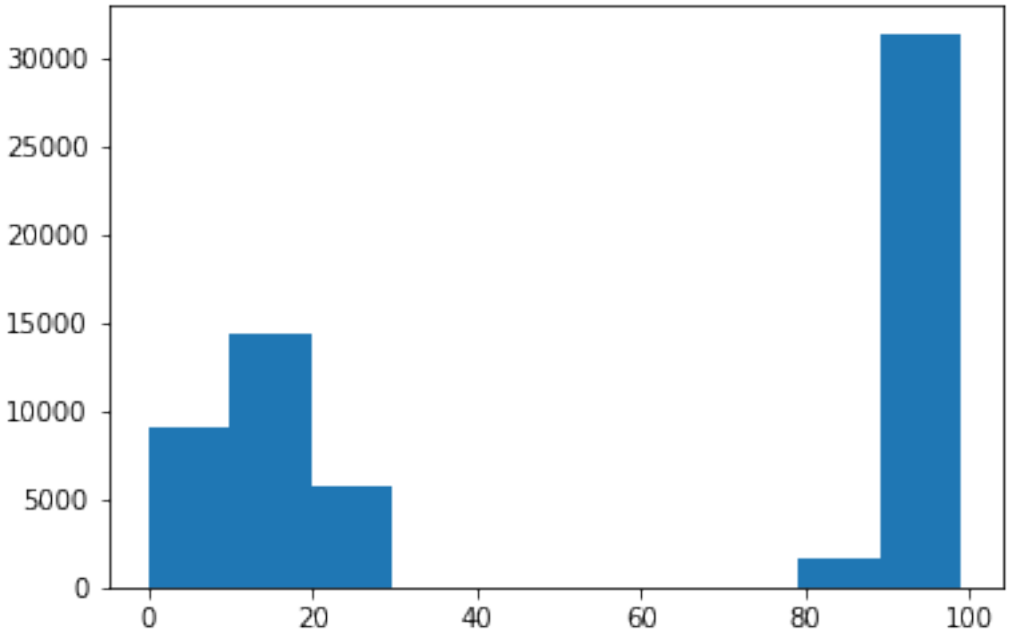
0	10001	1	1	0	...	2	58
1	10002	1	1	0	...	22	20
2	10003	1	1	0	...	1	45
3	10004	1	1	0	...	1	15
4	10005	2	2	0	...	7	16
5	10006	1	1	0	...	10	17
6	10007	1	1	0	...	18	38
7	10008	1	1	0	...	21	48
8	10009	1	1	0	...	8	3
9	10010	2	2	0	...	19	1
10	10011	2	2	0	...	21	20
11	10012	3	3	0	...	6	41
12	10013	1	1	0	...	88	88
13	10014	1	1	0	...	6	29
14	10015	1	1	0	...	8	55
15	10016	2	2	0	...	17	30
16	10017	1	1	0	...	14	50
17	10018	1	1	0	...	19	35
18	10019	1	1	0	...	88	88
19	10020	2	2	0	...	17	18
20	10021	2	2	0	...	14	5
21	10022	1	1	0	...	6	30
22	10023	1	1	0	...	21	25
23	10024	1	1	0	...	6	25
24	10025	1	1	0	...	88	88
25	10026	1	1	0	...	13	42
26	10027	2	2	0	...	14	47
27	10028	1	1	0	...	0	40
28	10029	1	1	0	...	5	52
29	10030	2	2	0	...	13	35
...	...	...	...	...	...	...	...
62192	560102	1	1	0	...	99	99
62193	560103	1	1	0	...	19	9
62194	560104	1	1	0	...	23	8
62195	560105	2	2	0	...	14	54
62196	560106	1	1	0	...	99	99
62197	560107	1	1	0	...	21	17
62198	560108	1	1	0	...	0	14
62199	560109	1	1	0	...	18	41
62200	560110	1	1	0	...	22	57
62201	560111	3	1	2	...	17	49
62202	560112	2	2	0	...	16	36
62203	560113	1	1	0	...	13	30
62204	560114	1	1	0	...	99	99
62205	560115	3	3	0	...	15	8
62206	560116	2	2	0	...	20	21
62207	560117	2	2	0	...	23	30
62208	560118	2	2	0	...	6	48

62209	560119	1	1	0	...	8	43
62210	560120	2	2	0	...	13	44
62211	560121	2	2	0	...	17	43
62212	560122	2	2	0	...	99	97
62213	560123	1	1	0	...	23	53
62214	560124	3	3	0	...	7	56
62215	560125	1	1	0	...	99	98
62216	560126	2	2	0	...	22	41
62217	560127	1	1	0	...	22	44
62218	560128	1	1	0	...	7	47
62219	560129	1	1	0	...	99	99
62220	560130	1	1	0	...	99	99
62221	560131	1	1	0	...	99	99

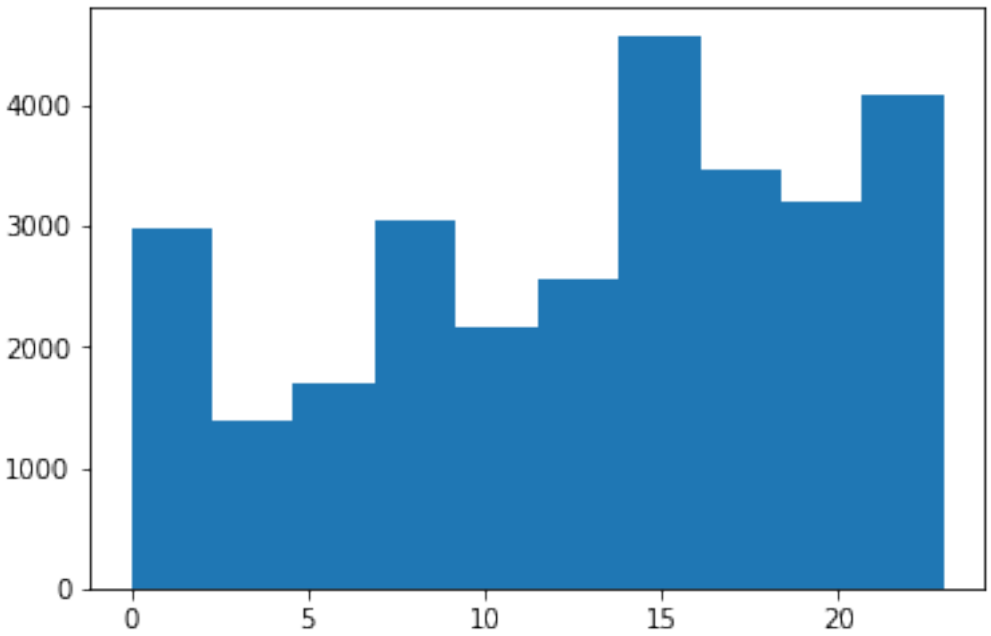
	HOSP_HR	HOSP_MN	CF1	CF2	CF3	FATALS	DRUNK_DR	ROAD_FNC
0	88	88	0	0	0	1	1	NaN
1	88	88	0	0	0	1	0	NaN
2	99	99	0	0	0	1	1	NaN
3	88	88	0	0	0	1	1	NaN
4	88	88	0	0	0	1	0	NaN
5	99	99	0	0	0	1	0	NaN
6	99	99	0	0	0	1	0	NaN
7	99	99	0	0	0	1	0	NaN
8	88	88	0	0	0	1	0	NaN
9	99	99	0	0	0	1	0	NaN
10	88	88	0	0	0	1	0	NaN
11	99	99	0	0	0	1	0	NaN
12	88	88	0	0	0	1	1	NaN
13	88	88	0	0	0	1	1	NaN
14	88	88	0	0	0	1	0	NaN
15	99	99	0	0	0	1	0	NaN
16	99	99	0	0	0	1	0	NaN
17	99	99	0	0	0	1	1	NaN
18	88	88	0	0	0	1	0	NaN
19	99	99	0	0	0	1	0	NaN
20	99	99	27	0	0	1	0	NaN
21	99	99	0	0	0	1	0	NaN
22	88	88	0	0	0	1	1	NaN
23	99	99	0	0	0	1	0	NaN
24	88	88	0	0	0	2	1	NaN
25	88	88	0	0	0	1	0	NaN
26	99	99	0	0	0	1	0	NaN
27	88	88	0	0	0	1	1	NaN
28	88	88	0	0	0	1	0	NaN
29	88	88	0	0	0	5	0	NaN
...	...	...	...	...	...	...	...	...
62192	99	99	0	0	0	1	0	16.0
62193	99	99	0	0	0	1	1	6.0

```
62194       88       88    0    0    0       1         1       2.0
62195       15       14    0    0    0       1         0       2.0
62196       99       99    0    0    0       1         1       6.0
62197       88       88    0    0    0       1         1       4.0
62198       88       88    0    0    0       1         1       4.0
62199       88       88    0    0    0       1         1       4.0
62200       99       99    0    0    0       1         1       4.0
62201       18       17   15   23    0       1         0      12.0
62202       16       49    0    0    0       1         0      16.0
62203       14       22    0    0    0       1         0       2.0
62204       88       88    0    0    0       1         1       4.0
62205       15       28    0    0    0       3         0      11.0
62206       21        5    0    0    0       1         0       2.0
62207        0        1   19    0    0       1         0       1.0
62208       88       88    0    0    0       1         0       2.0
62209       88       88    0    0    0       1         1       2.0
62210       14       18    0    0    0       1         0      16.0
62211       18       26    0    0    0       1         0       2.0
62212       99       97    0    0    0       1         0       1.0
62213       88       88    0    0    0       3         0       3.0
62214        8       21    0    0    0       1         1      13.0
62215       88       88    0    0    0       1         0       6.0
62216       88       88    0    0    0       1         2      16.0
62217       22       56    0    0    0       1         1       2.0
62218       88       88    0    0    0       1         0       6.0
62219       88       88    0    0    0       1         1       1.0
62220       99       99    0    0    0       1         0       1.0
62221       99       99    0    0    0       1         1       6.0

[62222 rows x 58 columns]
```

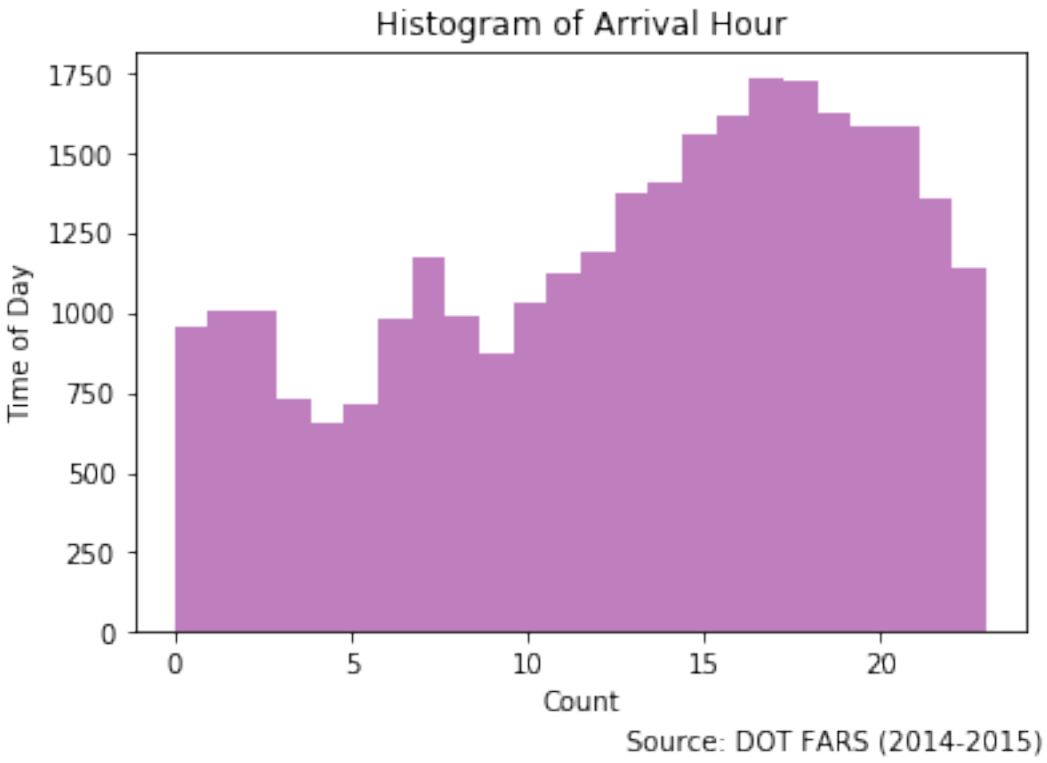
```
In [40]: # Data Visuaulzation can reveal errors in our data:
         plt.hist(acc.ARR_HOUR)
         plt.show()
```



In [66]:
```
acc_lim = acc[(acc.ARR_HOUR < 80)]

plt.hist(acc_lim.ARR_HOUR)
plt.show()
```



```
In [67]: ## We can change options within the hist function (e.g. number of bins, color, transp
         plt.hist(acc_lim.ARR_HOUR, bins=24, facecolor="purple", alpha=0.5)

         ## And we can affect the plot options too:
         plt.title ('Histogram of Arrival Hour')
         plt.xlabel('Count')
         plt.ylabel('Time of Day')

         ## And add Data sourcing:
         ### xy are measured in percent of axes length, from bottom left of graph:
         plt.annotate('Source: DOT FARS (2014-2015)', xy=(0.55,-0.2), xycoords="axes fraction"

         ## We use plt.show() to display the graph once we are done setting options:
         plt.show()
```



1.4.1 A Note on Data Sourcing

Data sourcing is a critical aspect of any data visualization. Although here we are simply referencing the agencies that created the data, it is ideal to provide as direct of a path as possible for the viewer to find the data the graph is based on. When this is not possible (e.g. the data is sequestered), directing the viewer to documentation or methodology for the data is a good alternative. Regardless, providing clear sourcing for the underlying data is an **absolutely requirement** of any respectable visualization, and further builds trusts and enables reproducibility.

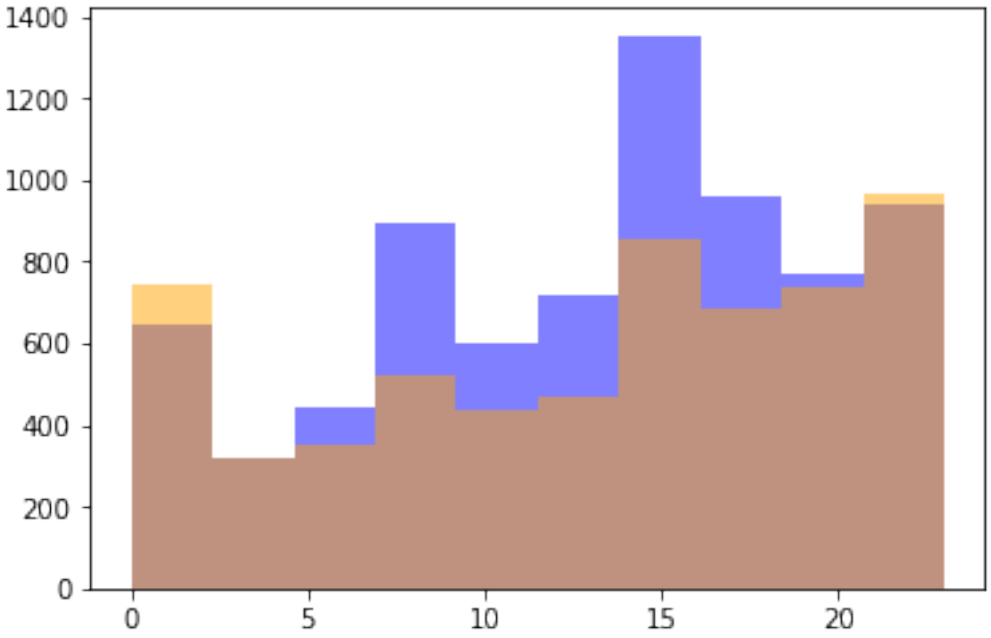
```
In [43]: acc_lim.RUR_URB.value_counts()

Out[43]: 1.0    7657
         2.0    6094
         8.0     603
         6.0      37
         9.0      13
         Name: RUR_URB, dtype: int64
```

1.4.2 Layering in Matplotlib

This functionality - where we can make consecutive changes to the same plot - also allows us to layer on multiple plots. By default, the first graph you create will be at the bottom, with ensuing graphs on top.

Below, we see the 2005 histogram, in blue, is beneath the 2015 histogram, in orange. You might also notice that the distribution of income for offenders has shifted over that ten year period.

```
In [68]: plt.hist(acc_lim[acc_lim["RUR_URB"] == 1].ARR_HOUR, facecolor="blue", alpha=0.5)
         plt.hist(acc_lim[acc_lim["RUR_URB"] == 2].ARR_HOUR, facecolor="orange", alpha=0.5)
         plt.show()
```



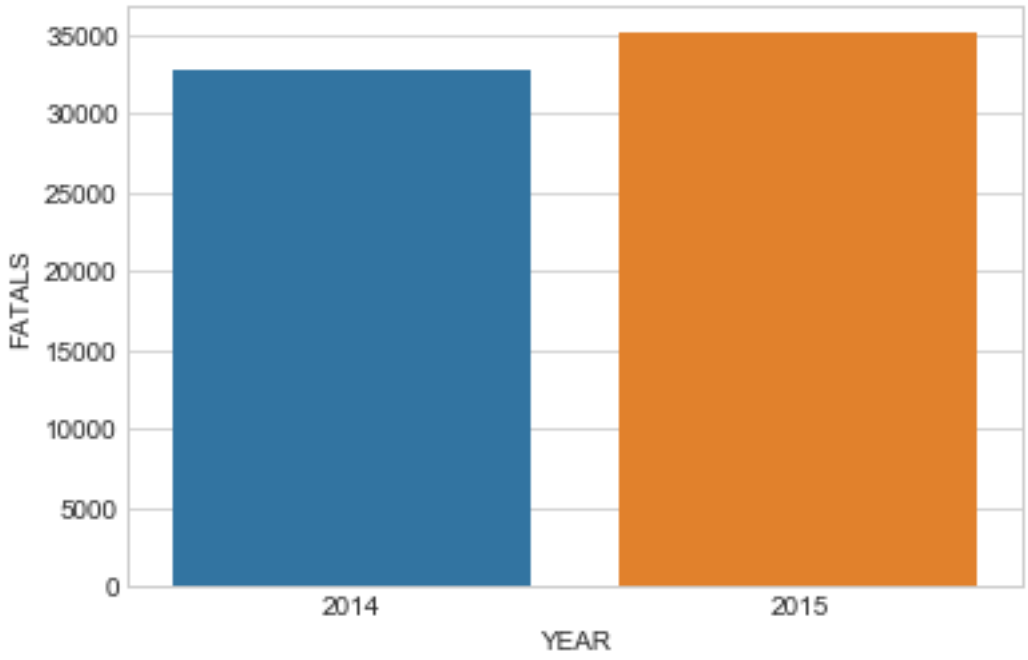
1.4.3 Our First Chart in seaborn

Below, we quickly use pandas to create an aggregation of our wages data - the average wages by year. Then we pass the data to the barplot function in the `seaborn` function, which recall we imported as `sns` for short.

```
In [73]: agg = acc.groupby(['YEAR'])['FATALS'].sum().reset_index()
         agg.head(10)
```

```
Out[73]:    YEAR  FATALS
         0  2014   32744
         1  2015   35092
```

```
In [79]: ## Barplot function

         # Seaborn's set_style function allows us to set many aesthetic parameters.
         sns.set_style("whitegrid")

         # Note we can reference column names (in quotes) in the specified data:
         sns.barplot(data=agg, x='YEAR', y='FATALS')
         plt.show()
```



1.5 Choosing a Data Visualization Package

There are many excellent data visualiation modules available in Python, but for the tutorial we will stick to the tried and true combination of `matplotlib` and `seaborn`. You can read more about different options for data visualization in Python in the Section **??** section at the bottom of this notebook.

`matplotlib` is very expressive, meaning it has functionality that can easily account for fine-tuned graph creation and adjustment. However, this also means that `matplotlib` is somewhat more complex to code.

`seaborn` is a higher-level visualization module, which means it is much less expressive and flexible than matplotlib, but far more concise and easier to code.

It may seem like we need to choose between these two approaches, but this is not the case! Since `seaborn` is itself written in `matplotlib` (you will sometimes see `seaborn` be called a `matplotlib` 'wrapper'), we can use `seaborn` for making graphs quickly and then `matplotlib` for specific adjustments. When you see `plt` referenced in the code below, we are using `matplotlib`'s pyplot submodule.

`seaborn` also improves on `matplotlib` in important ways, such as the ability to more easily visualize regression model results, creating small multiples, enabling better color palettes, and improve default aesthetics. From seaborn's documentation:

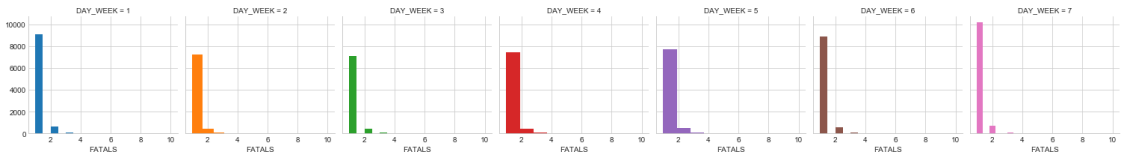
> If matplotlib 'tries to make easy things easy and hard things possible', seaborn tries to make a well-defined set of hard things easy too.

1.5.1 Seaborn and matplotlib

Below, we use seaborn for setting an overall aesthetic style and then faceting (created small multiples). We then use matplotlib to set very specific adjustments - things like adding the title, adjusting the locations of the plots, and sizing th graph space. This is a pretty protoyptical use of the power of these two libraries together.

More on Seaborn's set_style function. More on matplotlib's figure (fig) API.

```
In [113]: ## Seaborn offers a powerful tool called FacetGrid for making small multiples of mat

          ### Create an empty set of grids:
          facet_histograms = sns.FacetGrid(acc, col='DAY_WEEK', hue='DAY_WEEK')

          ## "map' a histogram to each grid:
          facet_histograms = facet_histograms.map(plt.hist, 'FATALS')

          ## Data Sourcing:
          plt.show()
```




```
In [127]: ## Simple Aggregation
          agg2 = acc.groupby(['StateName','YEAR'])['FATALS'].sum().reset_index()
          agg2.head(10)
```

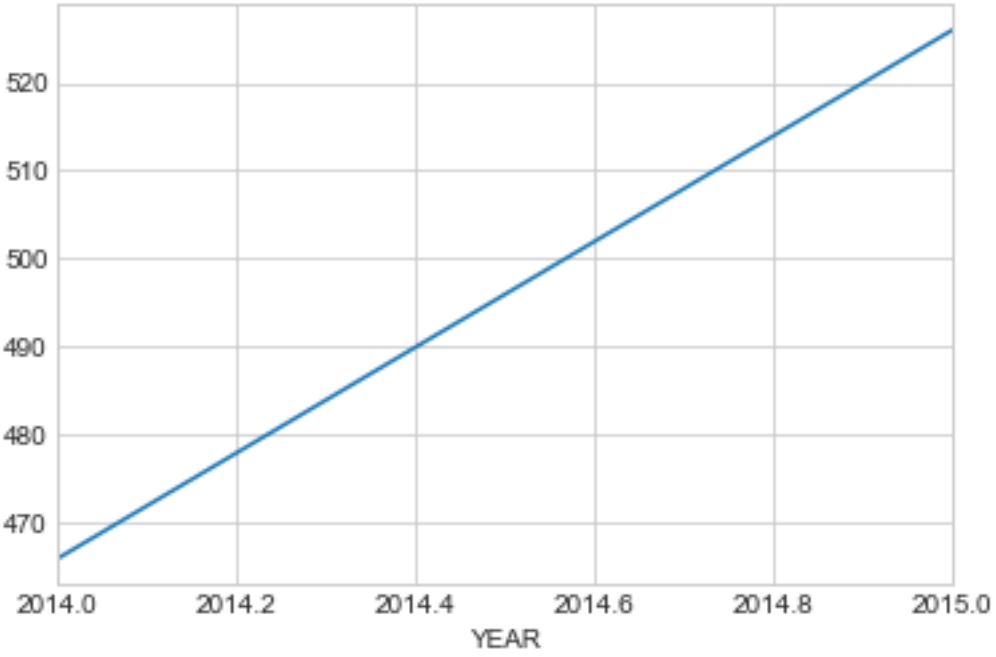
Out[127]:

	StateName	YEAR	FATALS
0	Alabama	2014	820
1	Alabama	2015	849

```
2       Alaska  2014      73
3       Alaska  2015      65
4      Arizona  2014     773
5      Arizona  2015     891
6     Arkansas  2014     466
7     Arkansas  2015     526
8   California  2014    3102
9   California  2015    3176
```

```
In [150]: ## Easy line chart in seaborn:
          agg_lim = agg2[agg2["StateName"] == "Arkansas"]
          sns.tsplot(data=agg_lim['FATALS'], time=agg_lim['YEAR'])
```

```
Out[150]: <matplotlib.axes._subplots.AxesSubplot at 0x10f0c2310>
```



1.6 Quiz:

Can you make a grid of small multiple line charts or scatterplots?

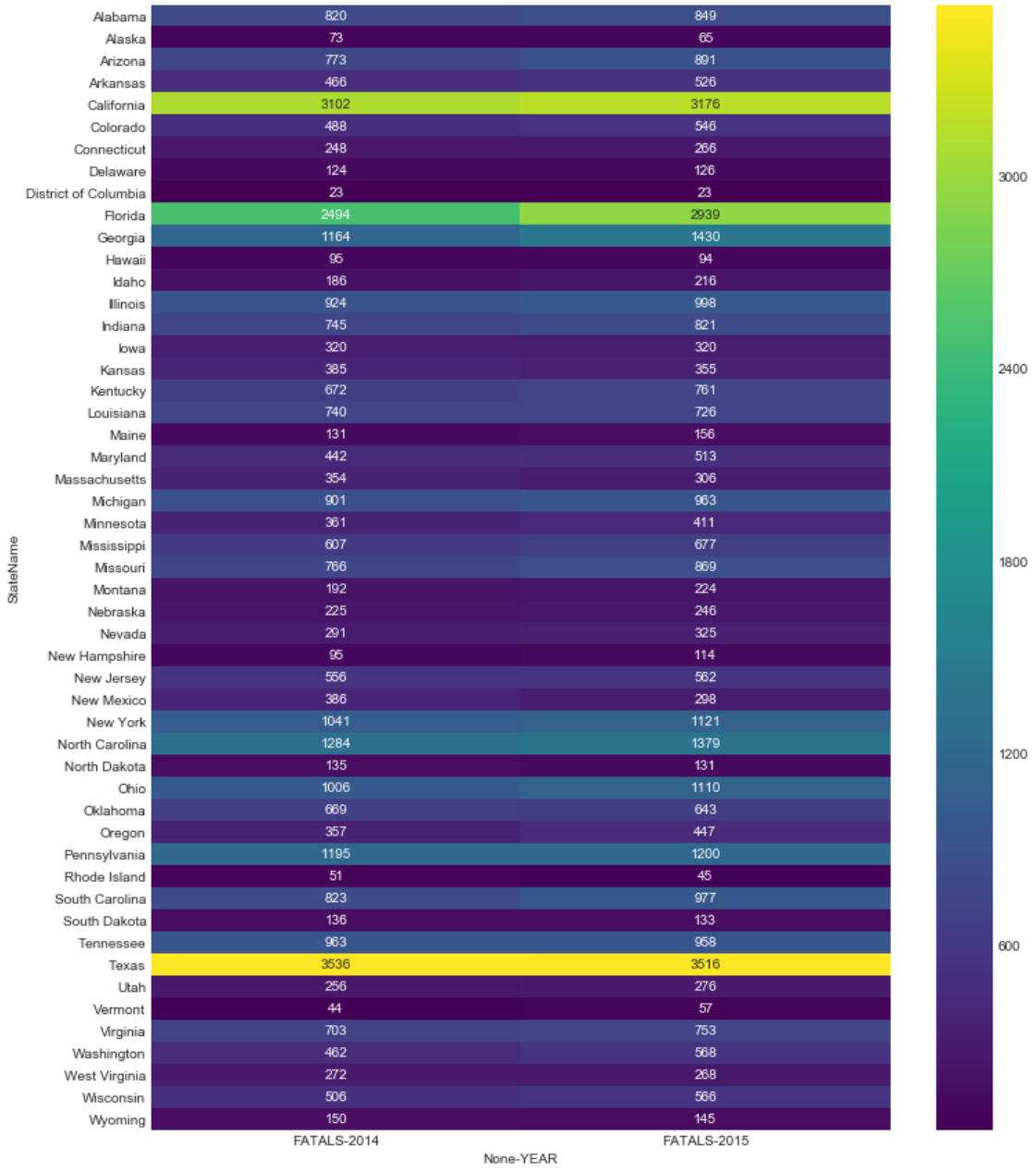
1.7 Visual Encodings

We often start with charts that use 2-dimensional position (like a scatterplot) or that use height (like histograms and bar charts). This is because these visual encodings - the visible mark that represents the data - are particularly perceptually strong. This means that when humans view these visual encodings, they are more accurate in estimating the underlying numbers than encodings like size (think circle size in a bubble chart) or angle (e.g. pie chart).

For more information on visual encodings and data visualization theory, see:

- Designing Data Visualizations, Chapter 4 by Julie Steele and Noah Iliinsky
- Now You See It - book by Stephen Few

In [149]:
```
### Reading Documentation and Learning New Charts - Heatmap
## Knowing the  expected data format is important!
## Heatmap requires a reformat to wide formatted data

agg2_wide = agg2.pivot("StateName", "YEAR")
agg2_wide

plt.figure(figsize=(12, 15))
sns.heatmap(agg2_wide, annot=True, fmt='g', cmap="viridis")
plt.show()
```



More information:

- seaborn heatmap documentation
- matplotlib color map documentation

1.8 Next Steps:

Check out the seaborn data visualization gallery and see if you can implement an interesting visualization.

1.9 Learn More:

A Dramatic Tour Through Pythons Data Visualization Landscape