Session 5:

Python plotting

Andreas Bjerre-Nielsen

Starting Jupyter in another folder

Check out this question on GitHub issues (https://github.com/abjer/sds2019/issues/3) - it points to a great answer on StackExchange

Repeating a quote by Hadley Wickham

*The bad news is that when ever you learn a new skill you're going to suck. It's going to be frustrating. The good news is that is typical and happens to everyone and **it is only temporary**. You can't go from knowing nothing to becoming an expert without going through a period of great frustration and great suckiness.*

Recap

What have we learned about basic Python and Pandas? (e.g. form, operators, methods, IO)

-

- *How do we store numeric variables?*
-

Agenda

1. Background on plotting
2. The Python toolbox for plotting
3. Plots for one variable (Series)
4. Plots for two or more variables (DataFrame):
    - numeric data
    - mixed numeric and categorical data
5. Advanced exploratory plotting

Understanding plotting

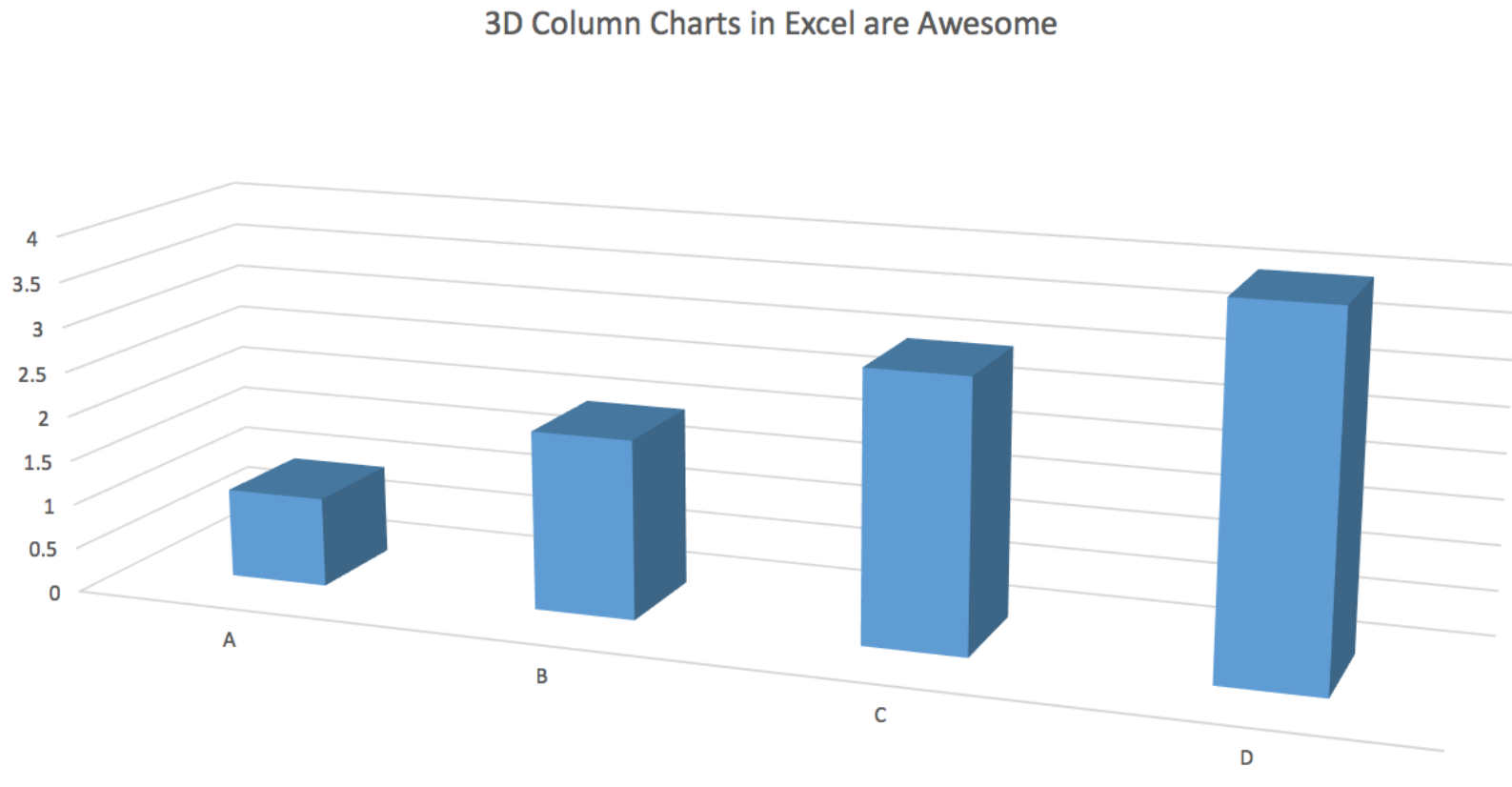
Why we plot

An English adage

> *A picture is worth a thousand words*

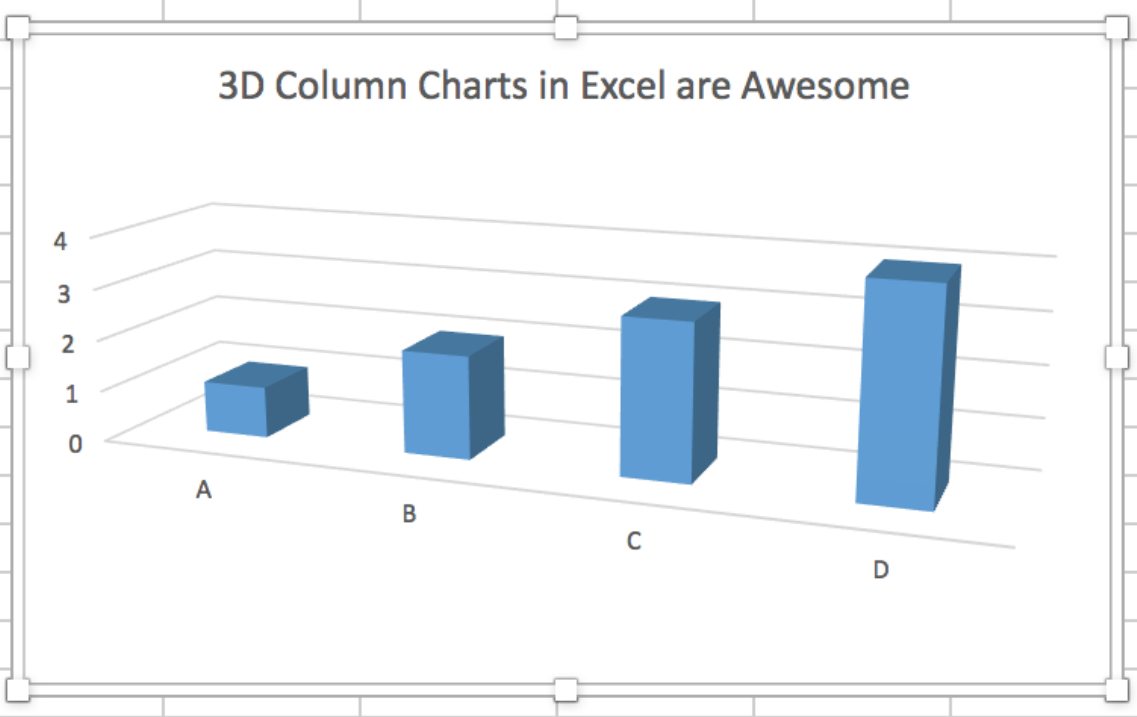
Is that always the case?

What values do A,B,C,D have?



The shocking answer

A	1
B	2
C	3
D	4



Why are you plotting?

Who's the audience?

Others

- **Explanatory** plots: polished figures to convey your message

You / your team:

- **Exploratory** plots: fast for understanding data - minimal polishing.

How should you plot (1)

*What are some tips for making **explanatory** plots in a report? **(Exam relevant!)***

1. Clear narratives - should convey key point(s)
    - If you to show difference between groups in data make sure it is easy to distinguish them.
2. Self explanatory
    - Contain axis label, title, footnotes in text containing relevant information.
3. Nice appereance
    - Choose the right plot type.
    - Make sure font type, size, colors, line width.
4. Keep simplicity.
    - Anything unnecessary should be removed, see this post (https://www.darkhorseanalytics.com/blog/data-looks-better-naked/).

How should you plot (2)

*What is some practical advice on making **explanatory** plots?*

1. Try out a few plot types, using exploratory analysis - use what works.
2. Apply the *"layered grammer of graphics"*.

    - Start with an empty canvas
    - Fill the necessary things (axis, ticks, bars/lines, labels)

How should you plot (3)

*What are some guidelines on making plots in **general**?*

Be aware of *what* you plot

- numerical vs. non-numeric (categorical)
- raw data vs. model results

Python plotting

Packages for Python plotting (1)

What is the fundamental tool for making plots in Python?

Matplotlib is the fundamental plotting module

- Can make almost any 2d plot.
- Can build publication ready figures.
- Caveat:
    - requires time consuming customization;
    - requires practice.

In [1]:
```
import matplotlib.pyplot as plt
# allow printing in notebook
%matplotlib inline
```

Packages for Python plotting (2)

What are good tools for fast, exploratory plots?

`seaborn` has built-in capabilities to make plots

- Analyzing data, e.g. splitting by subsets
- Make interpolation of data to smooth out noise.

`pandas` can easily convert Series and DataFrames to plots

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns # high level plotting library
```

Packages for Python plotting (3)

Seaborn comes with some illustrative datasets. We load `tips` .

In [3]:
```
tips = sns.load_dataset('tips')
print('Number of rows:',len(tips),'\n')
print(tips.head(3))
```

```
Number of rows: 244

   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
```

Plotting one variable

Plot one numeric variable (1)

How did we count categorical data?

- Using `value_counts`.

Can we do something similar with numeric data?

In [4]:
```
tb = tips['total_bill']
cuts = np.arange(0, 100, 20) # range from 0 to 100 with 20 between
tb_cat = pd.cut(tb, cuts) # cut into categorical data
tb_cat.value_counts()
```

Out[4]:
```
(0, 20]     147
(20, 40]     87
(40, 60]     10
(60, 80]      0
Name: total_bill, dtype: int64
```

Plot one numeric variable (2)

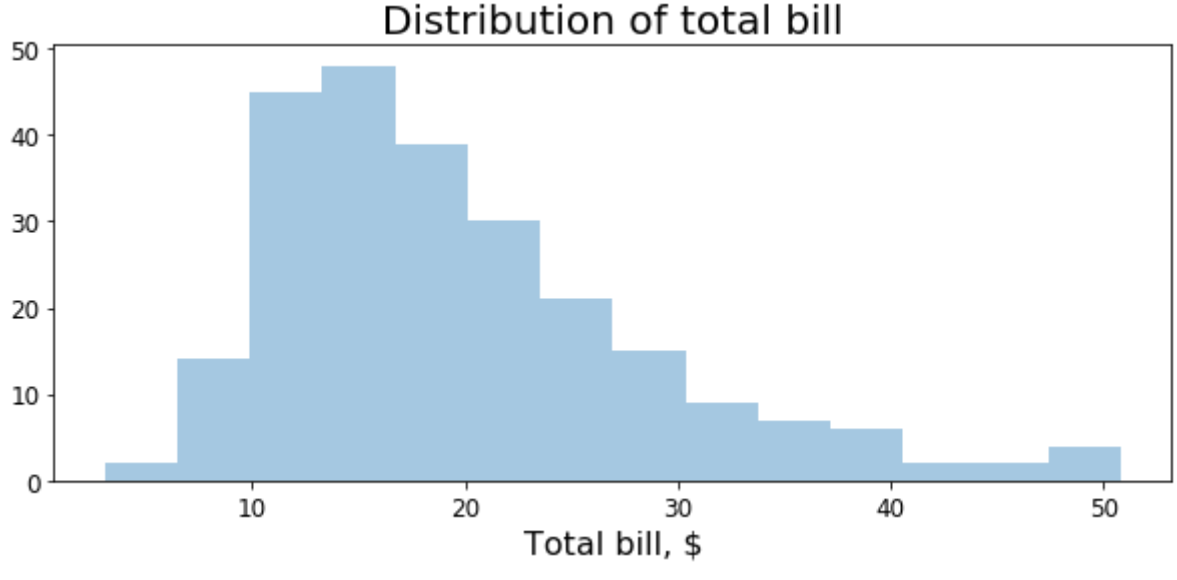
How do we plot the distribution of numerical variables?

We often use the histogram.

- Bins data and counts observations (made from cutting data)
- Example of tips:

In [7]:
```
histplot
```

Out[7]:

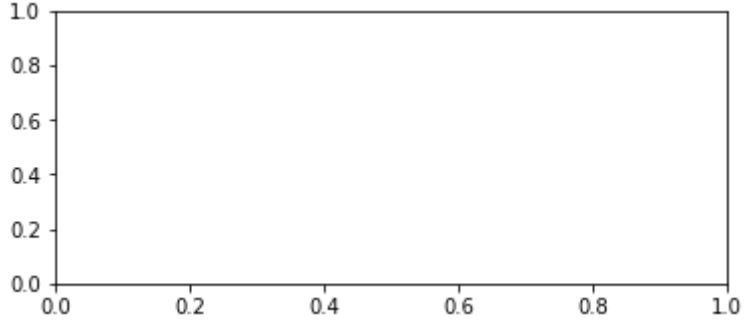


Matplotlib and the grammar of graphics (1)

Where do I start with making a plot?

We will begin with the fundamental and flexible way. We start with our plotting canvas.

In [8]:
```
fig, ax = plt.subplots(figsize = (6, 2.5)) # create placeholder for plot
```



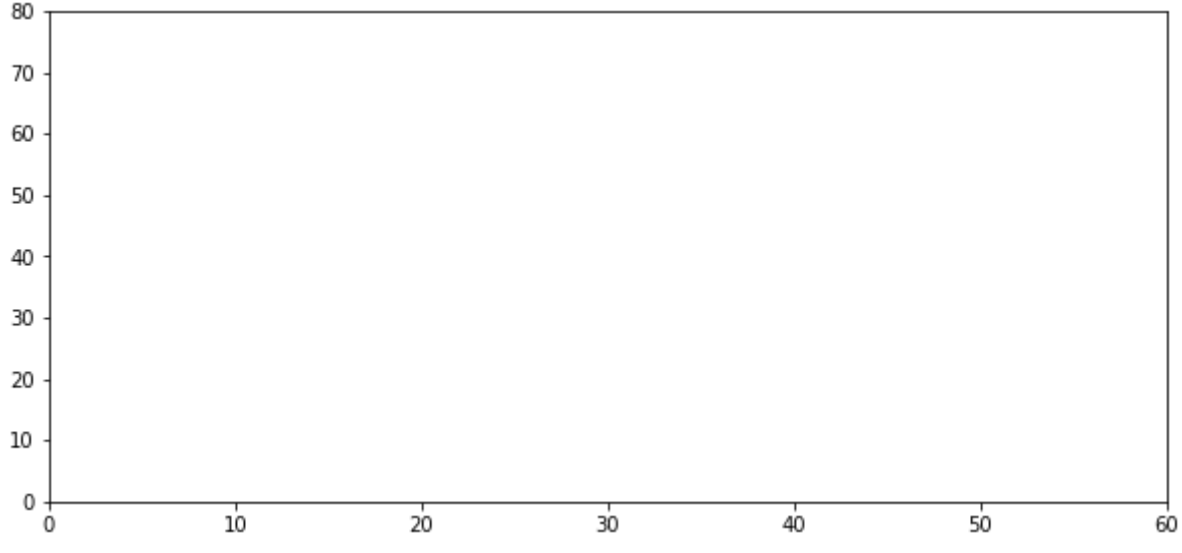
- ax contains most of the chart content as objects:
    - grid axes, labels, shapes we draw etc.
- `fig` the actual plot which is displayed (export to pdf etc.)

Matplotlib and the grammar of graphics (2)

We can modify our canvas, e.g the axis scaling:

In [9]:
```
fig, ax = plt.subplots(figsize = (10, 4.5))
ax.set_xlim([0, 60]) # x-axis cutoffs
ax.set_ylim([0, 80]) # y-axis cutoffs
```

Out[9]:
```
(0, 80)
```

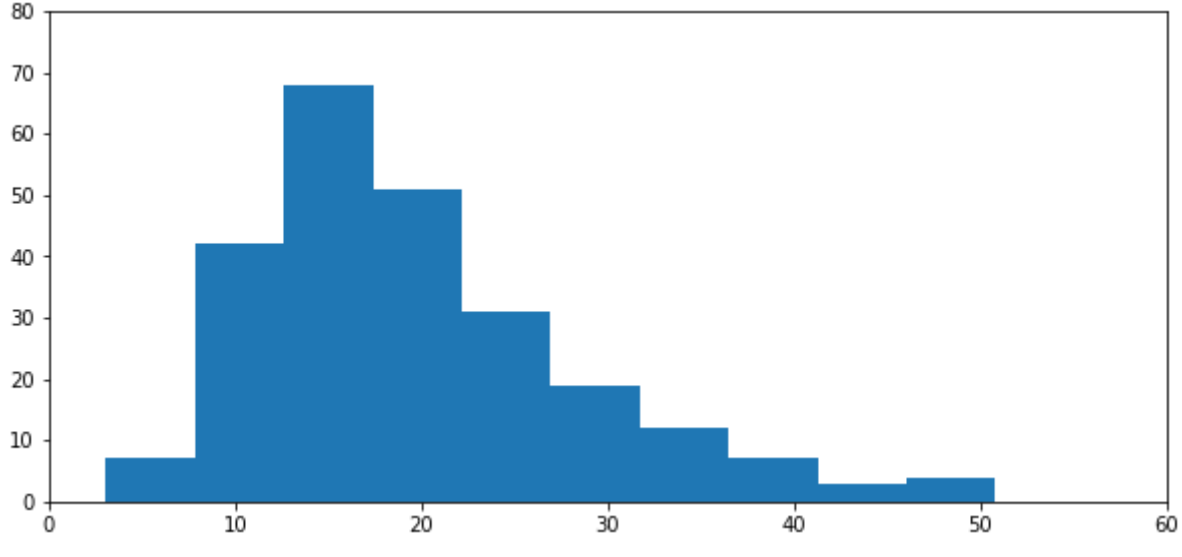


Matplotlib and the grammar of graphics (3)

We can draw plots on the canvas

In [10]:
```
fig, ax = plt.subplots(figsize = (10, 4.5))
ax.set_xlim([0, 60])
ax.set_ylim([0, 80])
ax.hist(tb) # make plot
```

Out[10]:
```
(array([ 7., 42., 68., 51., 31., 19., 12.,  7.,  3.,  4.]),
 array([ 3.07 ,  7.844, 12.618, 17.392, 22.166, 26.94 , 31.714, 36.488,
        41.262, 46.036, 50.81 ]),
 <a list of 10 Patch objects>)
```



Matplotlib and the grammar of graphics (4)

What might we change about our plot?

- We will try customization in the exercises today.

Matplotlib and the grammar of graphics (5)

Can we change matplotlib defaults?

Yes, this may be very useful. For instance plot size.

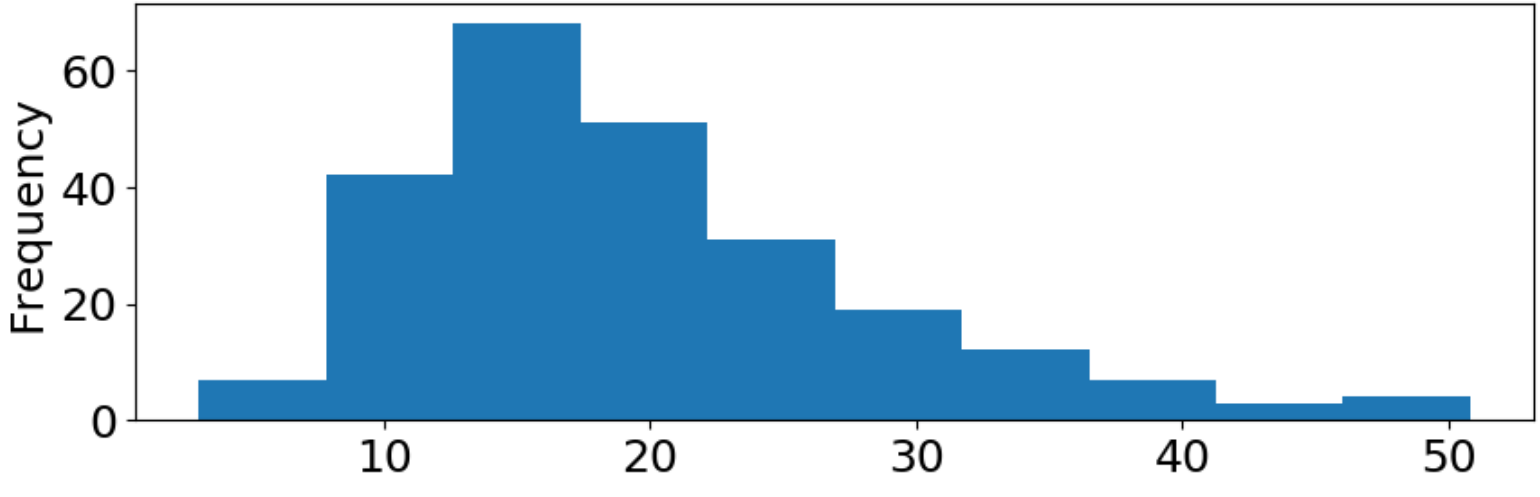
In [40]:
```
plt.style.use('default') # set style (colors, background, size, gridlines etc.)
plt.rcParams['figure.figsize'] = 10, 3 # set default size of plots
plt.rcParams.update({'font.size': 18})
```

Plotting with pandas

Pandas has a quick and dirty implemention. Let's try the code below.

In [41]:
```
tb.plot.hist()
```

Out[41]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x28432964b00>
```

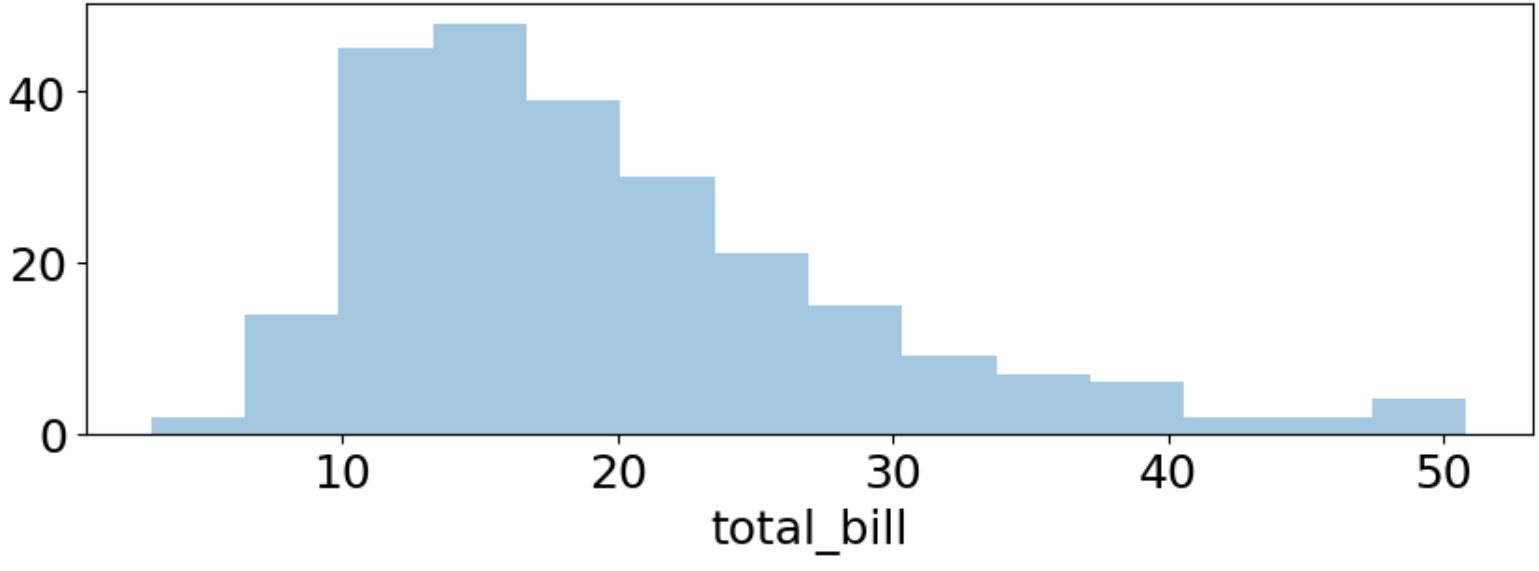


Plotting with Seaborn (1)

The module Seaborn is great for fast plots that look good.

In [42]:
```
sns.distplot(tb, kde=False)  # histogram for seaborn, what is KDE?
```

Out[42]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2843358e438>
```



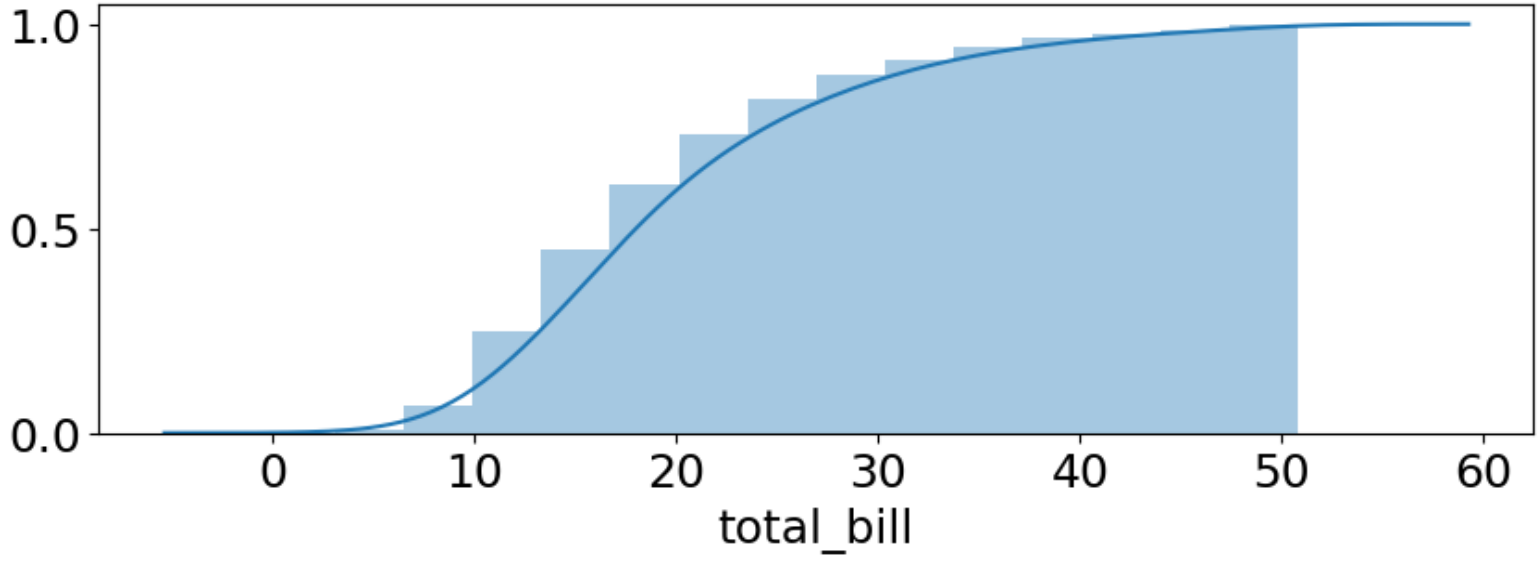
Plotting with Seaborn (2)

How can we use Seaborn for cumulative plots?

Yes, we just need some arguments.

In [43]:
```
sns.distplot(tb, hist_kws={'cumulative': True}, kde_kws={'cumulative': True})
```

Out[43]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x28432928978>
```



Summing up

How did our tools perform?

-

-

Plotting one categorical variable

What is categorical data?

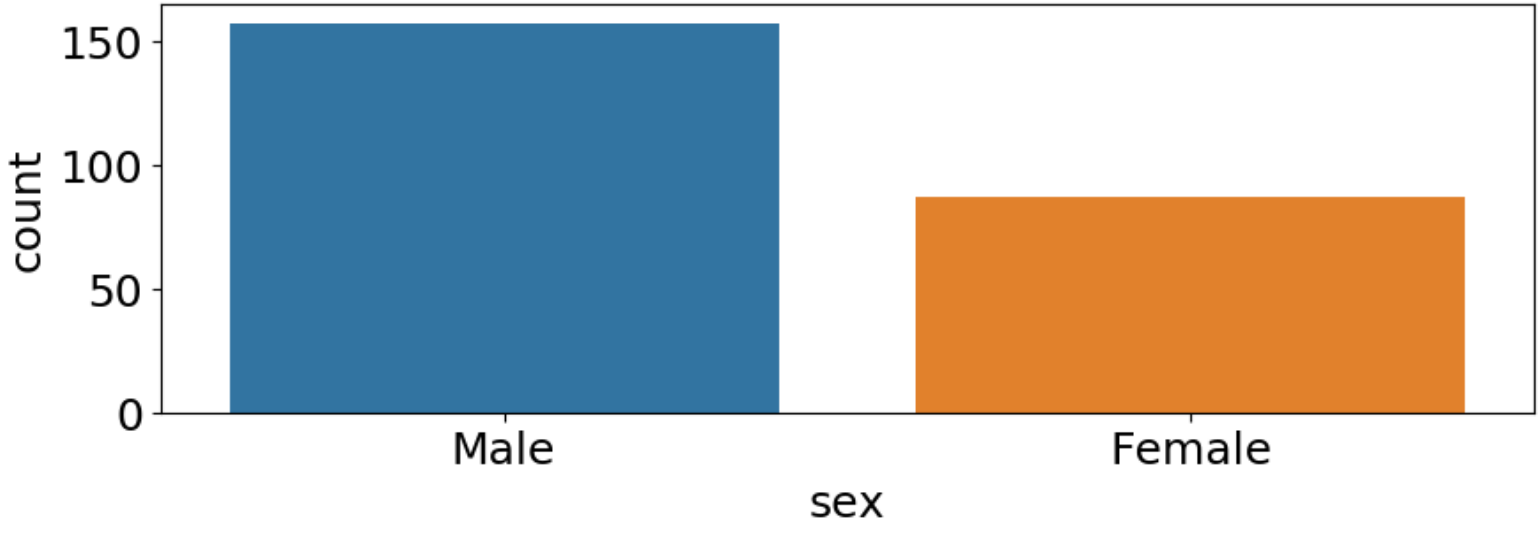
Categorical is non-numeric data (this afternoon).

Plotting one categorical variable (2)

How can we plot categorical data? Pie chart is ugly..

In [44]:
```
sns.countplot(x='sex', data=tips)
```

Out[44]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x28432af9f98>
```



Plots of two numeric variables

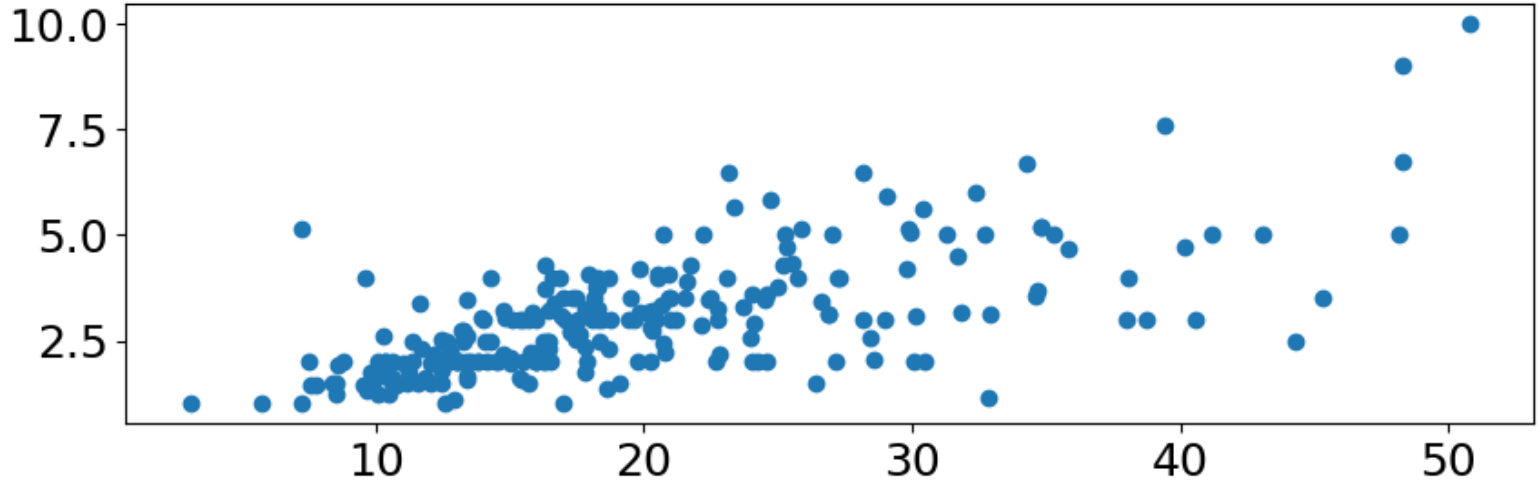
Two numeric variables (1)

How do we plot two numeric variables?

If we have little data we can make a point cloud, i.e. a scatter plot.

In [46]:
```
plt.scatter(x=tips['total_bill'], y=tips['tip'])
```

Out[46]:
```
<matplotlib.collections.PathCollection at 0x28432a7ec88>
```



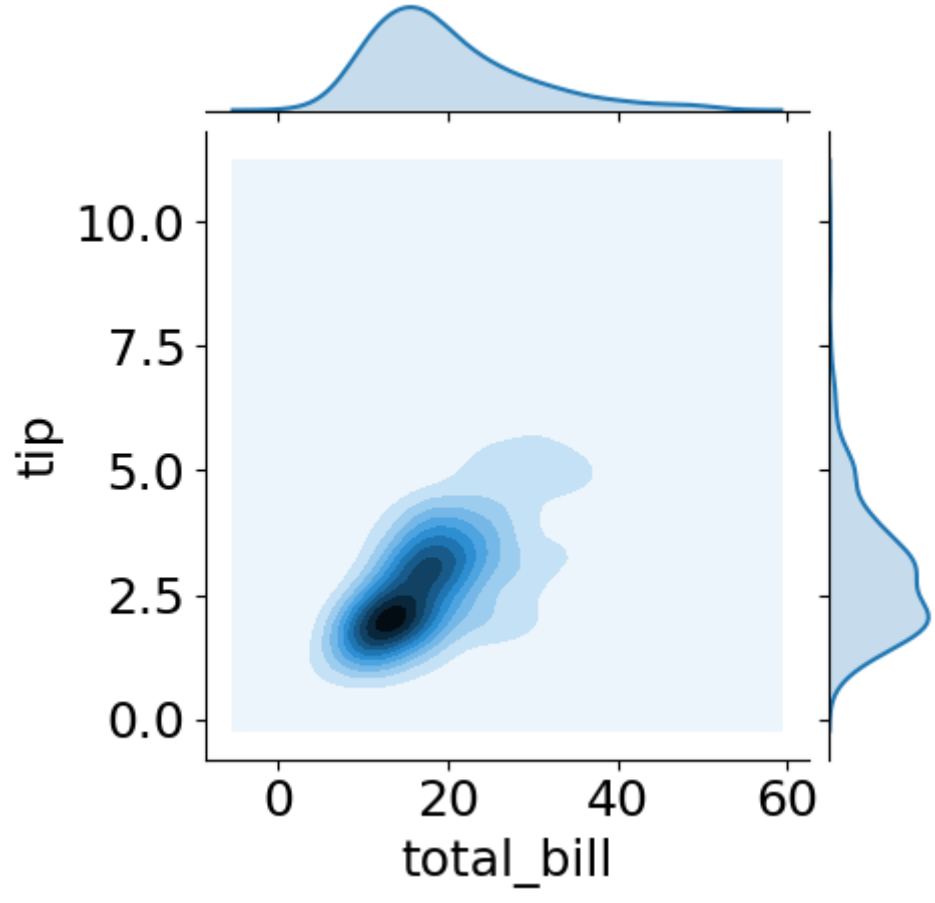
Two numeric variables (2)

Quiz: How might we alter the scatter plot?

We can interpolate the data:

In [47]:
```
sns.jointplot(x='total_bill', y='tip', data=tips, kind='kde', size=5) # hex
```

Out[47]:
```
<seaborn.axisgrid.JointGrid at 0x28432a7ec18>
```



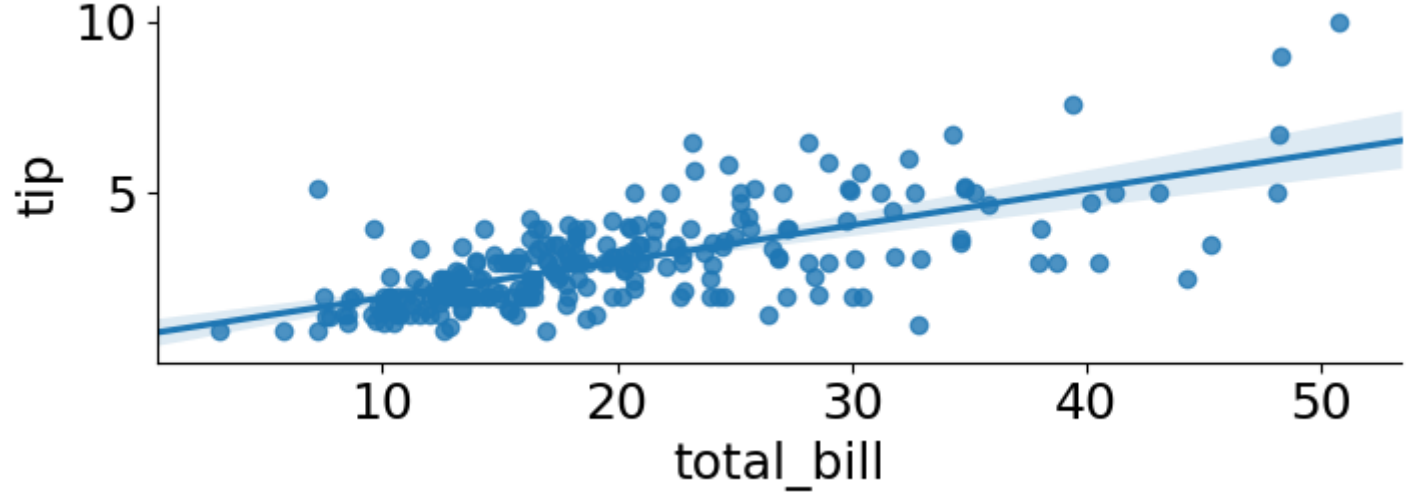
Two numeric variables (3)

What if we want to see the linear relationship?

We use the linear model plot:

In [51]:
```
sns.lmplot(x='total_bill', y='tip', data=tips, size=3, aspect=2.5)
```

Out[51]:
```
<seaborn.axisgrid.FacetGrid at 0x28434f5a780>
```

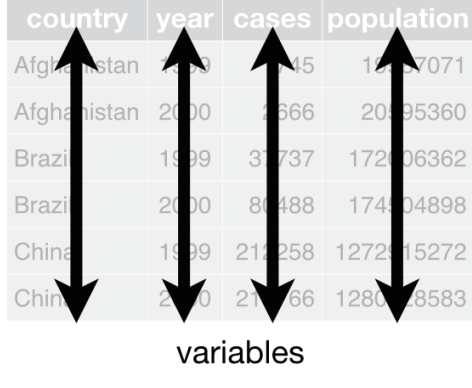


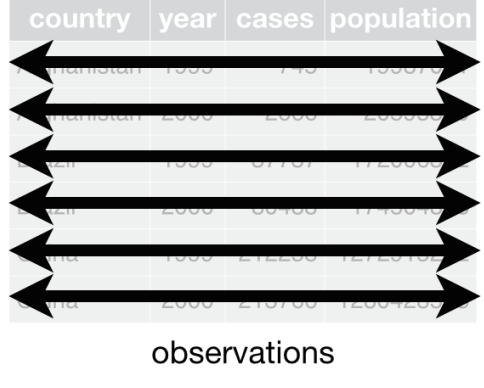
Plots with mixed variables

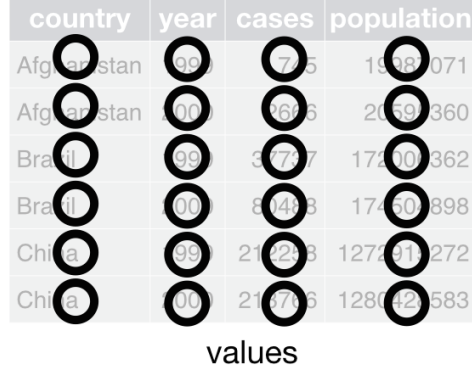
Table format

How did we define a tidy/long table?

One row for each observation







Mixed types - numeric, categorical (1)

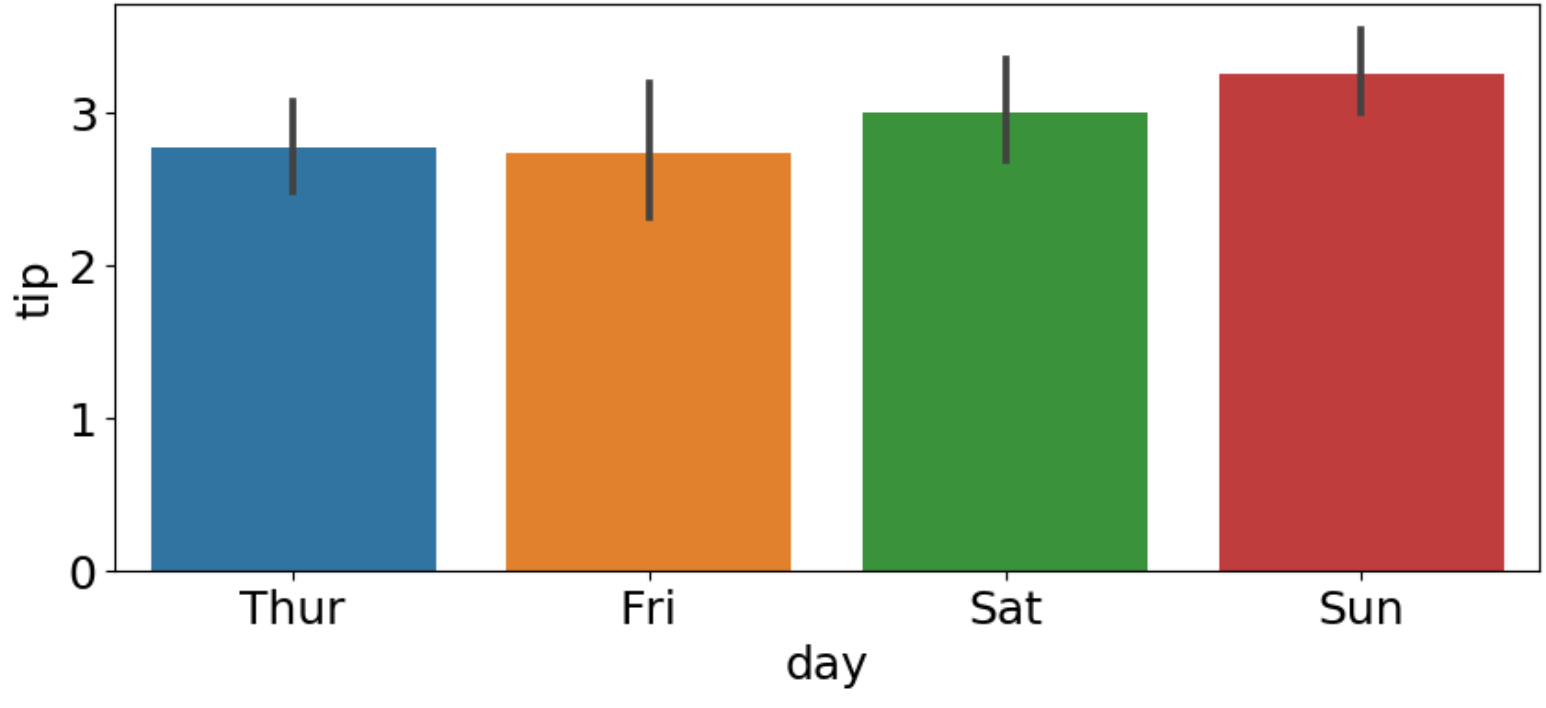
How might we use categorical variables?

- We can split data and make plots based on subsets of data!

Mixed types - numeric, categorical (2)

Let's make a plot the mean tips - distinguish by weekday:

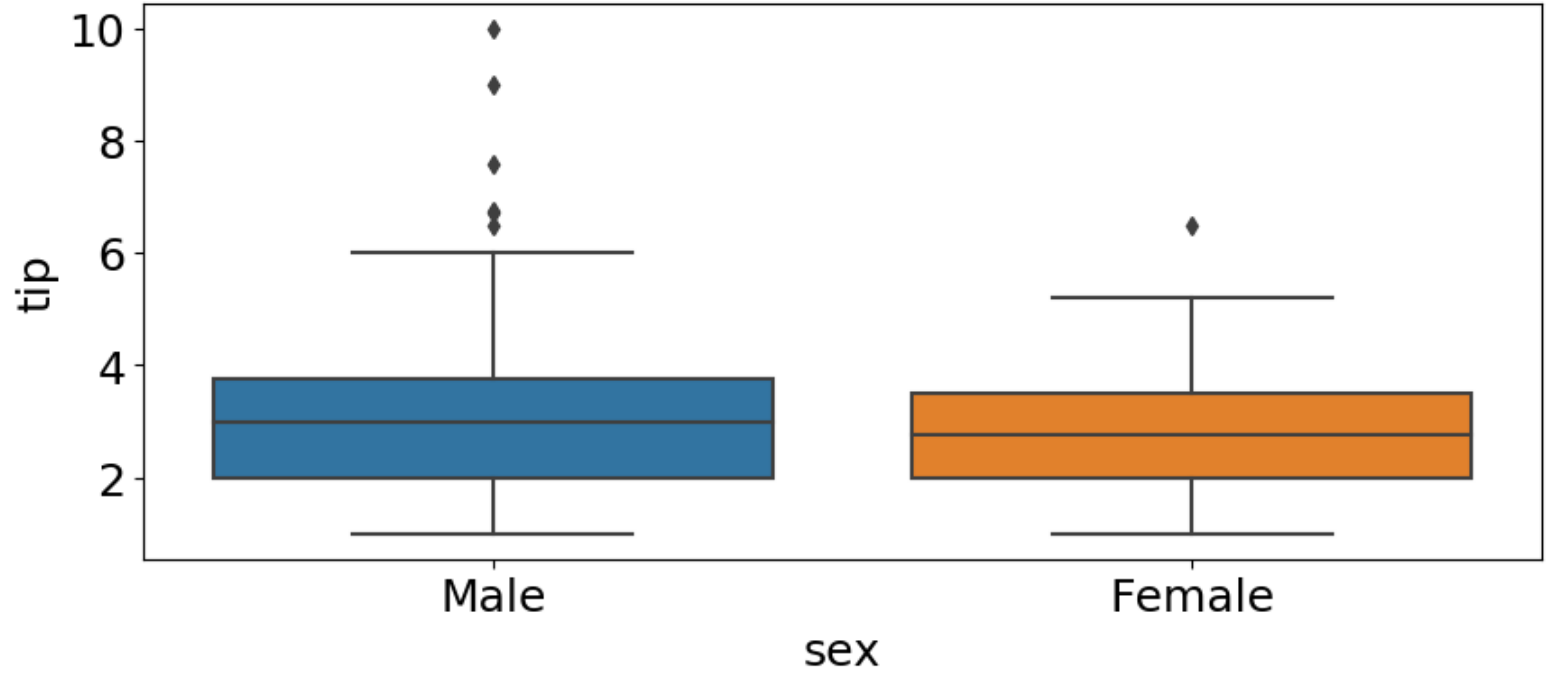
In [22]:
```
f = sns.barplot(x='day', y='tip', data=tips) # hue='sex'
```



Mixed types - numeric, categorical (3)

Let's make a plot the tip quartiles - distinguish by sex:

In [23]:
```
f = sns.boxplot(x='sex', y='tip', data=tips)
```

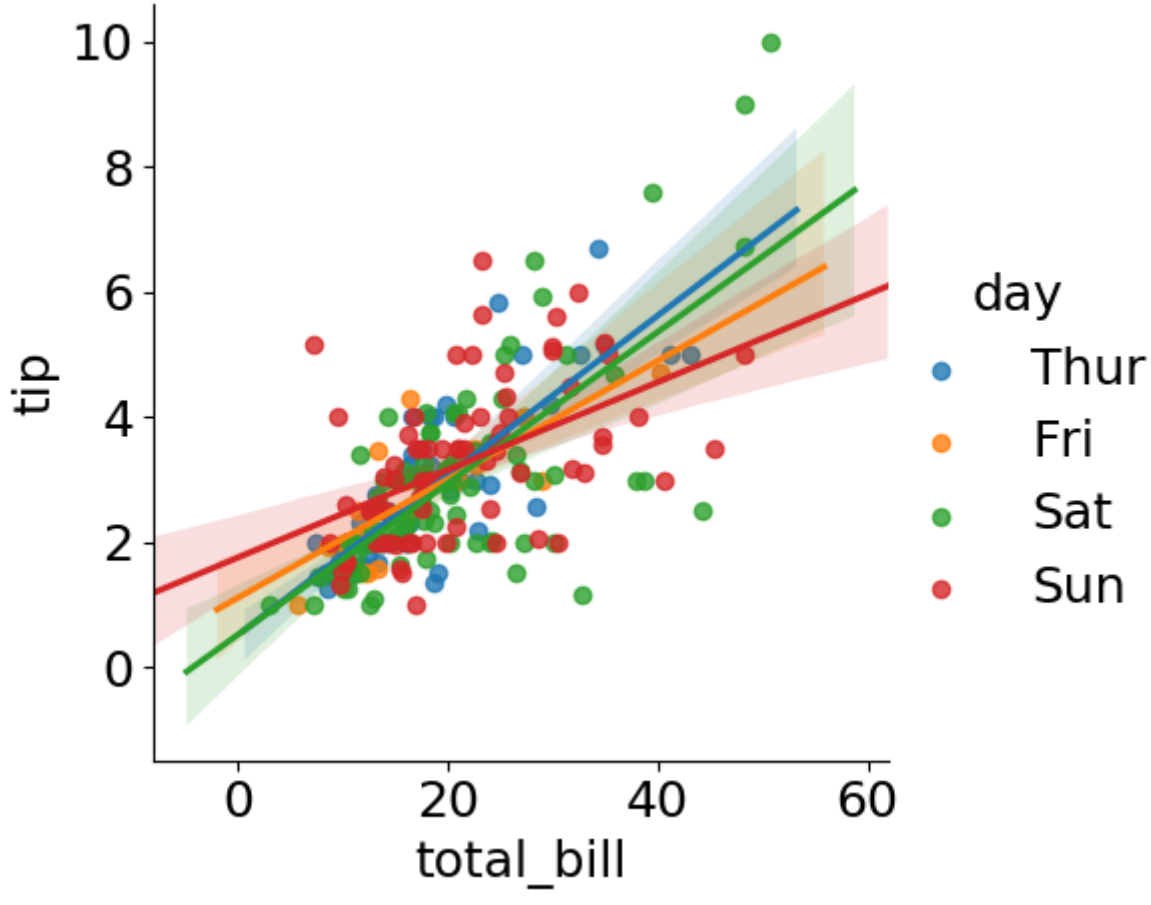


Mixed types - numeric, categorical (4)

Let's make a linear model plot the - distinguish slope by sex:

In [26]:
```
sns.lmplot('total_bill', 'tip',hue='day',data=tips)
```

Out[26]:
```
<seaborn.axisgrid.FacetGrid at 0x284309fbc50>
```



Advanced exploratory plotting

Plot grids (1)

How can we plot the relationship for more than two variables?

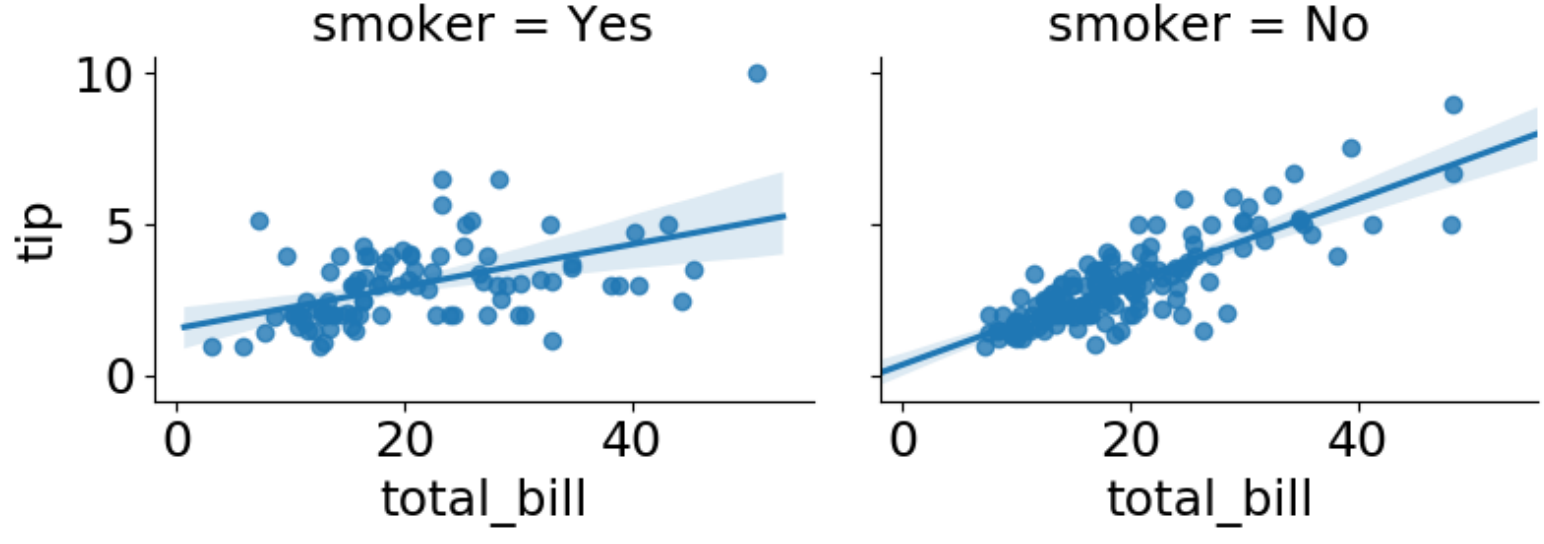
In [27]:
```
sns.pairplot(tips, height=2.3) # make hist and scatter for all
```

Out[27]:
```
<seaborn.axisgrid.PairGrid at 0x28430a51c88>
```

Plot grids (2)

Can we split the data to investigate heterogeneous relationships?

In [32]:
```
g = sns.FacetGrid(tips, col='smoker', height=3.2, aspect=1.3) #row='sex'
g = g.map(sns.regplot, 'total_bill', 'tip')
```



Can we say anything about smokers tipping behavior?

The end

Return to Agenda