Introduction to data science

Patrick Shafto

Department of Math and Computer Science

Plan for today

- Basic stats

Basic statistics

- Chi square
- t-test
- correlation
- regression

Basic question:

- Are X and Y related?
  - HARD!
  - How to decide related vs not?
  - What kind of variables?
  - What can we assume to be true?

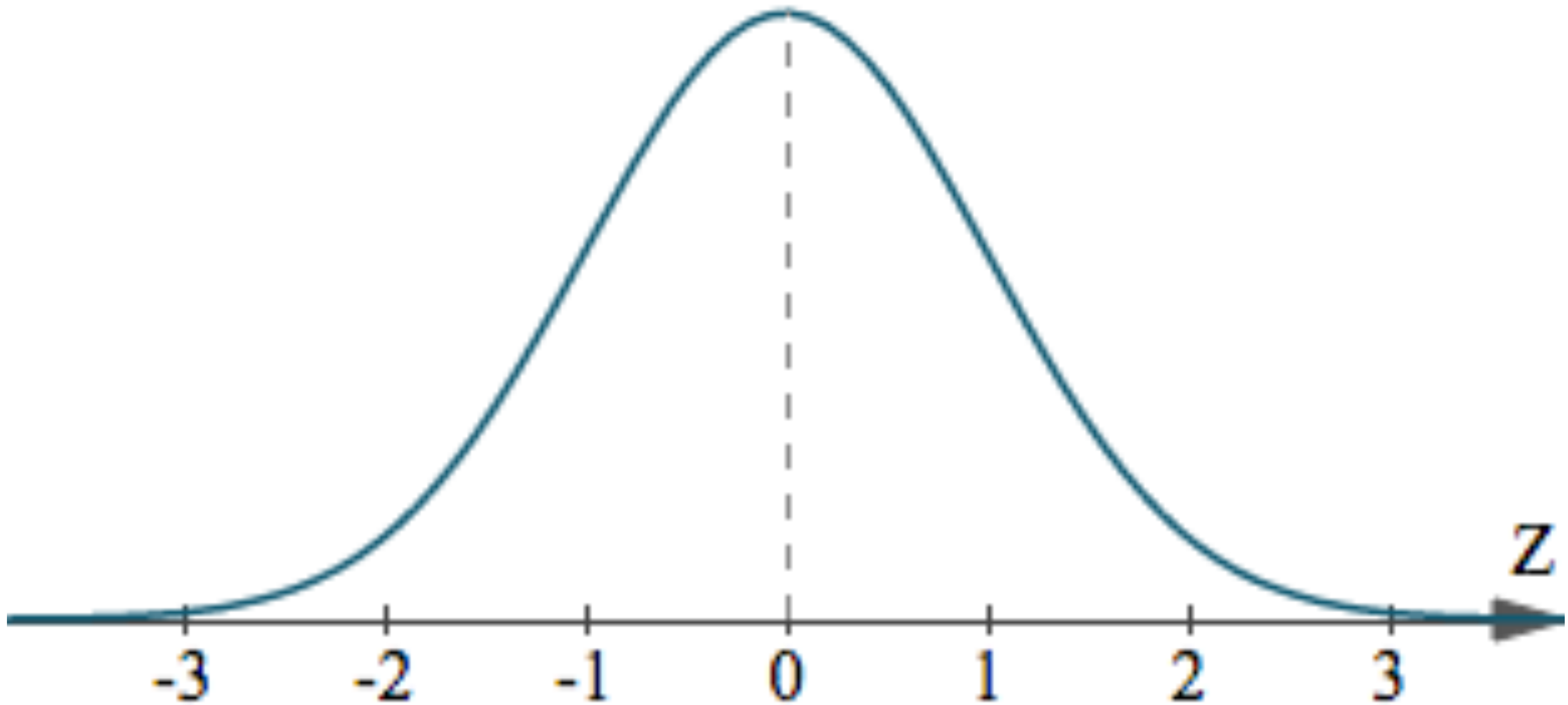
Neyman-Pearson statistical inference

- Measure how far your sample is from what one would expect based on random chance
- Decide on a cut-off that is “far enough” (alpha)
  - This is typically 0.05. Meaning that there is a 0.05 chance or less of this event or more extreme assuming random sampling

Neyman-Pearson statistical inference

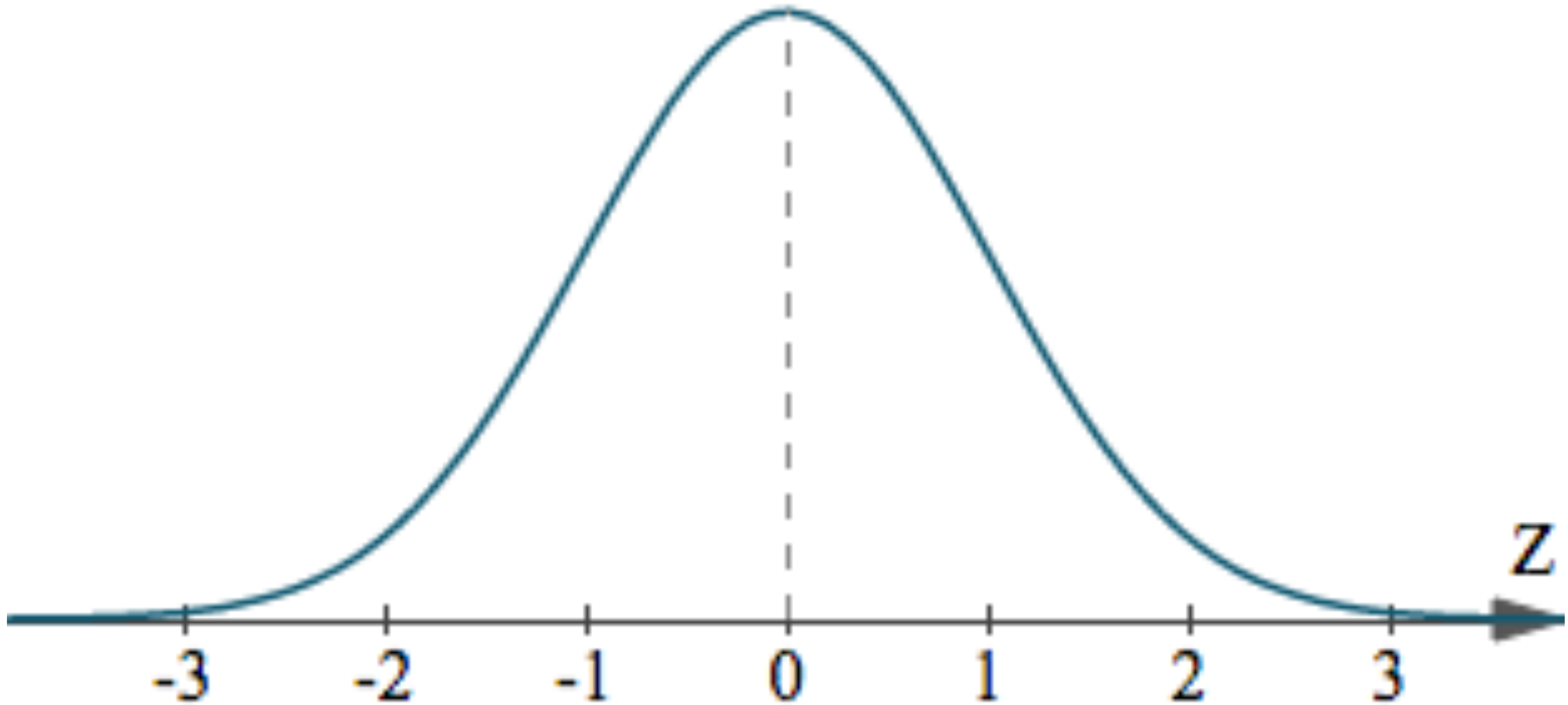
- Comparing two hypotheses:
    - Null hypothesis: There is no relationship/ difference between the groups
    - Alternative hypothesis: There is a relationship/ difference
- Neyman-Pearson only tests the first hypothesis!

Normal distribution



Assume we are talking about IQ scores.
The mean is 100, standard deviation is 15.
Thus, -1 on this plot corresponds to 85.

The region beyond which .05 of the mass resides is +/- 1.96.



Say we do a study in which we aim to make people smarter, and the average of our group is had a (standardized score) of 3. What can we say?

If the average had a standardized score of 1.
What could we say?

Statistical decisions: Knowing the possibilities

Truth

Decision

	Null hypothesis is **true**	Null hypothesis is **false**
reject Null hypothesis		
accept Null hypothesis		

Statistical decisions: Knowing the possibilities

Truth

Decision

	Null hypothesis is **true**	Null hypothesis is **false**
reject Null hypothesis	Type 1 error (alpha=.05)	
accept Null hypothesis		Type 2 error

Statistical decisions: Knowing the possibilities

Truth

Decision

	Null hypothesis is **true**	Null hypothesis is **false**
reject Null hypothesis	Type 1 error (alpha=.05)	Power
accept Null hypothesis	1-alpha	Type 2 error

Basic question:

- Are variables X and Y related?
  - HARD!
  - How to decide related vs not?
  - What kind of variables?
  - What can we assume to be true?

Kinds of variables

- Categorical
- Continuous
- (also others, frequency, etc)

Kinds of variables

- Categorical: Chi squared
  - (test for independence)
- Categorical (two) & Continuous: T-test
  - (independent samples)
- Continuous: Correlation / regression

Two categorical variables: Chi squared

```
df.groupby('Sex').Survived.value_counts()
```

```
Sex     Survived
female  1           233
        0            81
male    0           468
        1           109
Name: Survived, dtype: int64
```

Two categorical variables: Chi squared

$$\chi^2 = \sum \frac{(observed - expected)^2}{expected}$$

$$cellExpected = \frac{rowTotal * colTotal}{n}$$

Two categorical variables: Chi squared

	Republican	Democrat	Totals
M	215	143	358
F	19	64	83
Totals	234	207	441

Two categorical variables: Chi squared

```
house = [ [ 215, 143 ], [ 19, 64 ] ]
chi2, p, ddof, expected = scipy.stats.chi2_c
msg = "Test Statistic: {}\np-value: {}\nDegr
print( msg.format( chi2, p, ddof ) )
print( expected )
```

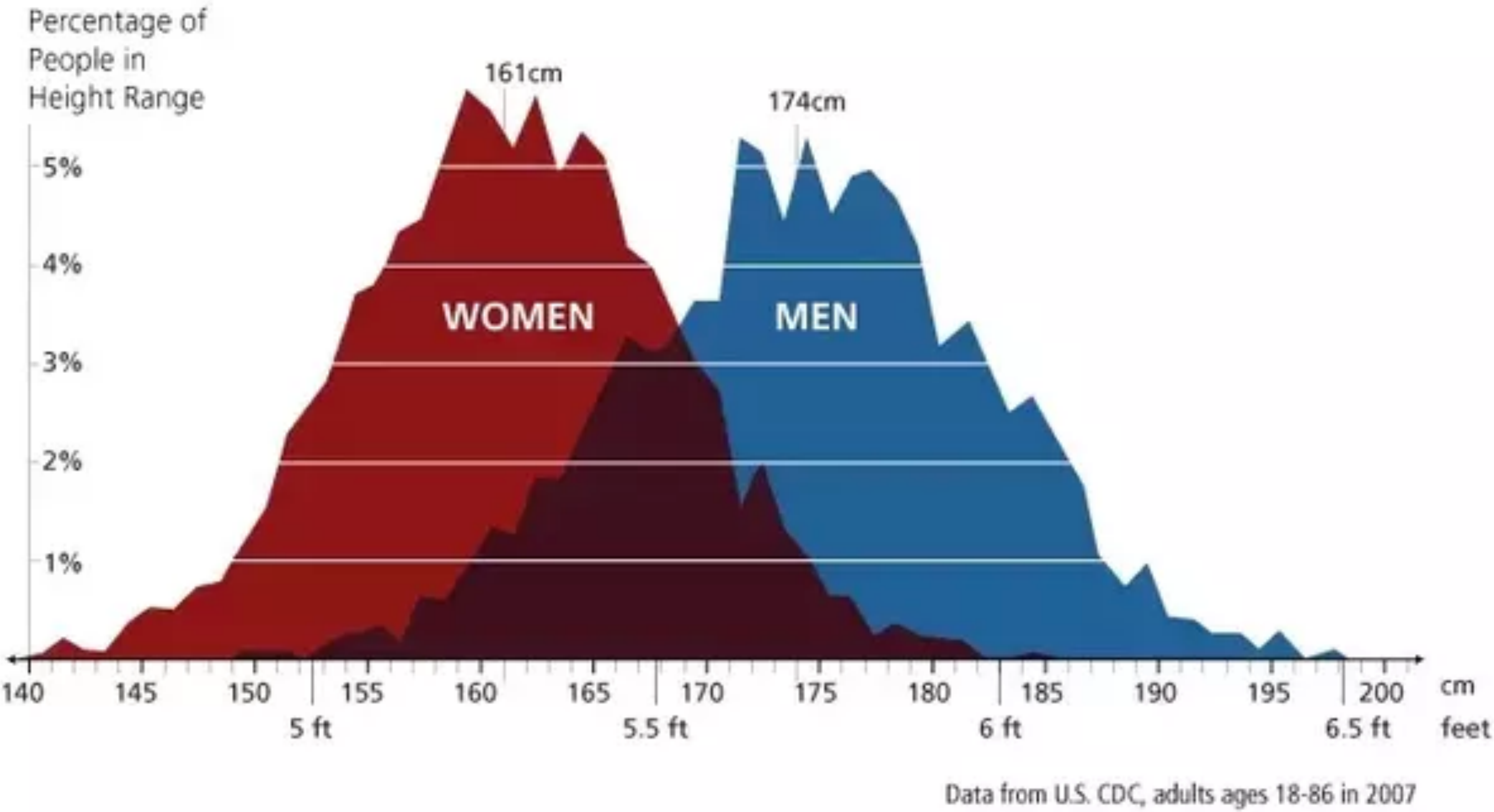
```
Test Statistic: 35.8877686481
p-value: 2.09016744218e-09
Degrees of Freedom: 1


[[ 189.95918367  168.04081633]
 [  44.04081633   38.95918367]]
```

Basic statistics

- Chi sqaured:
  - http://nbviewer.jupyter.org/github/ipython-books/cookbook-code/blob/master/notebooks/chapter07_stats/04_correlation.ipynb
  - pd.crosstab
  - pd.chi2_contingency

Categorical (two) and continuous: t-test

- Categorical variable (independent)
    - e.g. Men and women
- Continuous variable (dependent)
    - Height
- Do the groups differ?



Collect finite samples from population

Categorical (two) and continuous: t-test

$$t = \frac{\bar{x}_1 - \bar{x}_2}{S_{diff}}$$

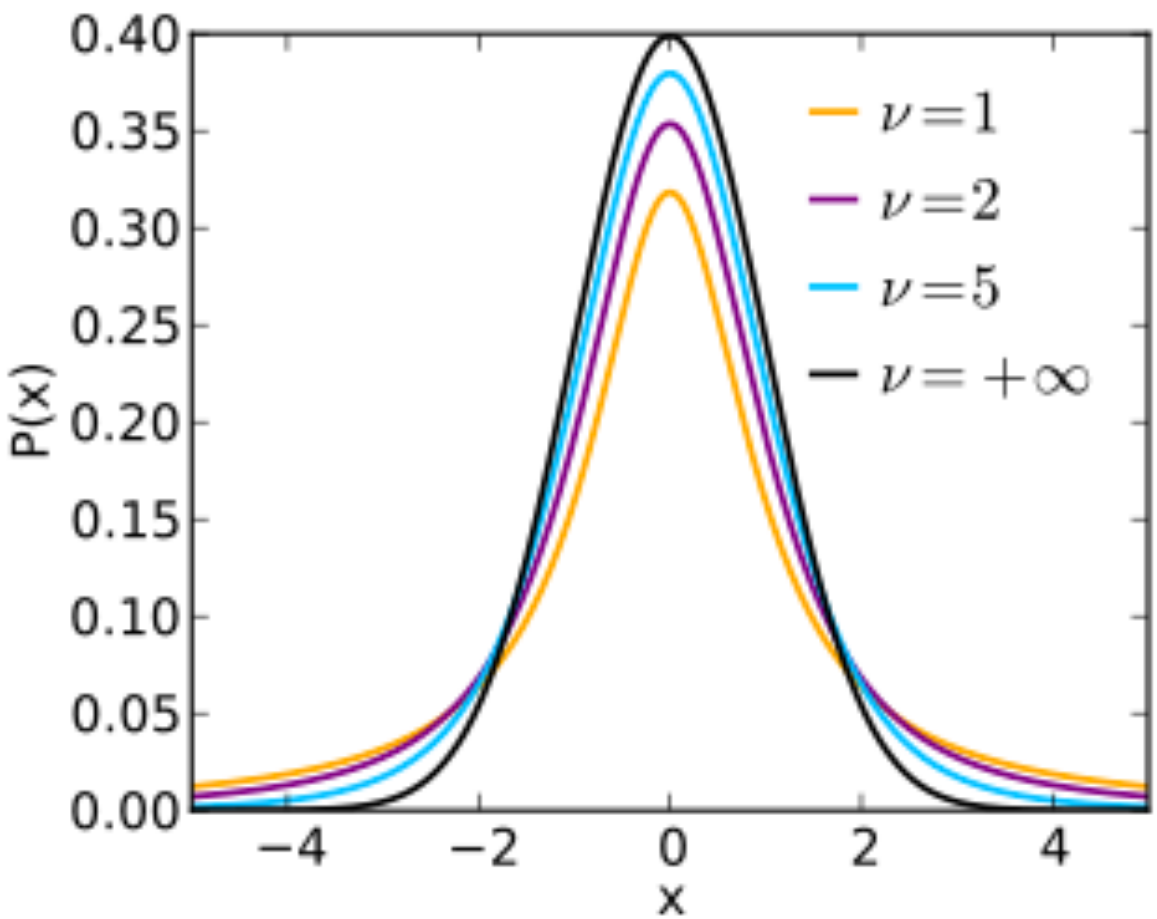
$$S_p^2 = \frac{df_1}{df_{tot}} S_1^2 + \frac{df_2}{df_{tot}} S_2^2$$

Pooled estimate of variance

$$S_{diff} = \sqrt{\frac{S_p^2}{n_1} + \frac{S_p^2}{n_2}}$$

Standard deviation of differences

Categorical (two) and continuous: t-test



Categorical (two) and continuous: t-test

```
>>> female_viq = data[data['Gender'] == 'Female']['VIQ']
>>> male_viq = data[data['Gender'] == 'Male']['VIQ']
>>> stats.ttest_ind(female_viq, male_viq)
(-0.77261617232750124, 0.44452876778583217)
```

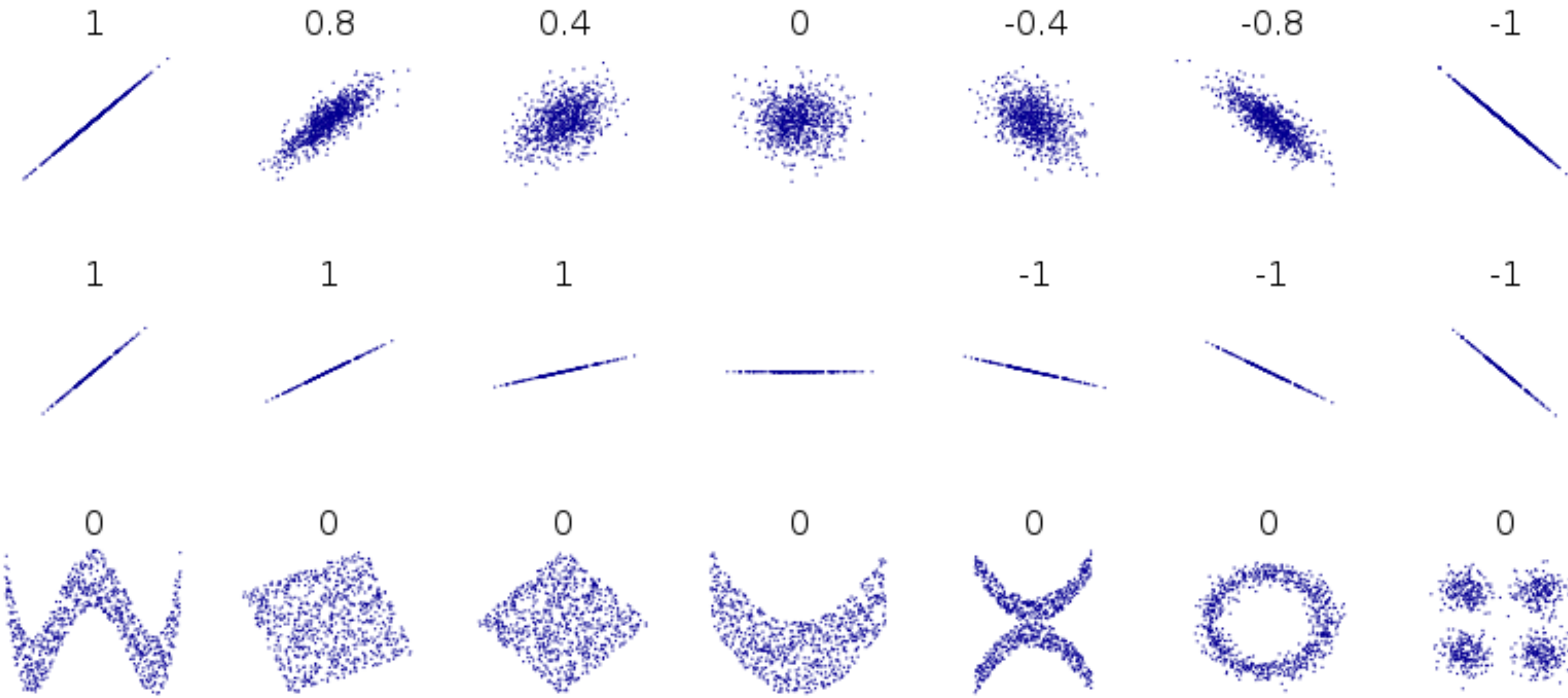
T-test

- Overview:
  - http://www.scipy-lectures.org/packages/statistics/index.html#hypothesis-testing-comparing-two-groups

Correlation and linear regression

- Two continuous variables
  - Height and weight
- Correlation: Are the variables related?
- Regression: Does X predict Y?

Correlation and linear regression



Correlation

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}},$$

Correlation and linear regression

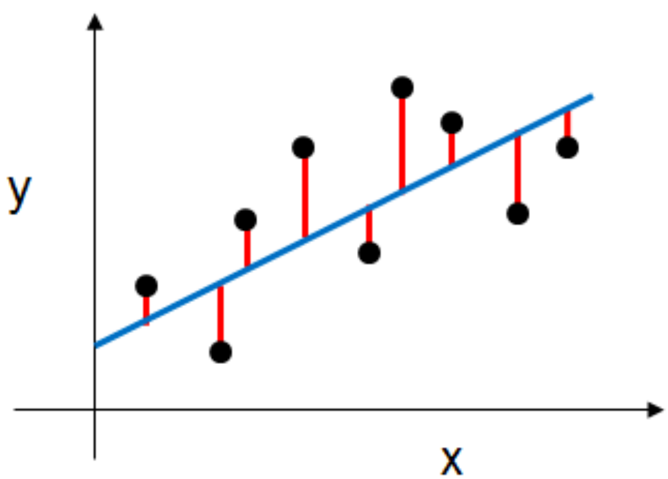
- linear regression:
  - https://github.com/justmarkham/DAT4/blob/master/notebooks/08_linear_regression.ipynb
  - up through “how well does the model fit the data”

Linear regression

- Simple linear regression is an approach for predicting a quantitative response using a single feature (or "predictor" or "input variable"). It takes the following form:
- y = \beta_0 + \beta_1(x)
- What does each term represent?
    - y is the response
    - x is the feature
    - \beta_0 is the intercept
    - \beta_1 is the coefficient for x

Linear regression

- Coefficients are estimated using the least squares criterion, which means we are find the line (mathematically) which minimizes the sum of squared residuals (or "sum of squared errors"):



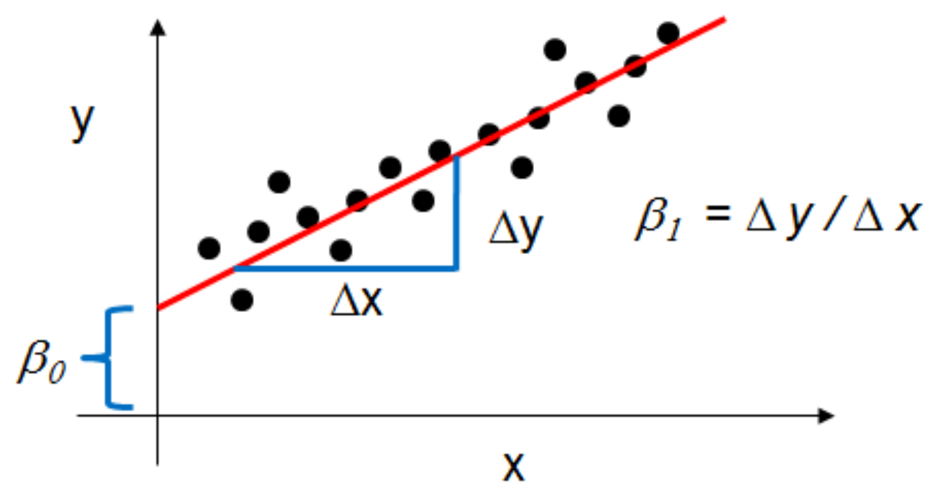
Model Prediction

$$SS_{residuals} = \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

Observed Result

Linear regression

- How do the model coefficients relate to the least squares line?
- \beta_0 is the intercept (the value of y when x=0)
- \beta_1 is the slope (the change in y divided by change in x)



Linear regression

```
In [5]: # this is the standard import if you're using "formula notation" (similar to R)
        import statsmodels.formula.api as smf

        # create a fitted model in one line
        lm = smf.ols(formula='Sales ~ TV', data=data).fit()

        # print the coefficients
        lm.params
```

```
Out[5]: Intercept    7.032594
        TV           0.047537
        dtype: float64
```

Linear regression

In [5]:
```
# this is the standard import if you're using "formula notation" (similar to R)
import statsmodels.formula.api as smf

# create a fitted model in one line
lm = smf.ols(formula='Sales ~ TV', data=data).fit()

# print the coefficients
lm.params
```

Out[5]:
```
Intercept    7.032594
TV           0.047537
dtype: float64
```

In [8]:
```
# use the model to make predictions on a new value
lm.predict(X_new)
```

Out[8]:
```
array([ 9.40942557])
```

Linear regression

In [5]:
```python
# this is the standard import if you're using "formula notation" (similar to R)
import statsmodels.formula.api as smf

# create a fitted model in one line
lm = smf.ols(formula='Sales ~ TV', data=data).fit()

# print the coefficients
lm.params
```

Out[5]:
```
Intercept    7.032594
TV           0.047537
dtype: float64
```

In [13]:
```python
# print the p-values for the model coefficients
lm.pvalues
```

Out[13]:
```
Intercept    1.406300e-35
TV           1.467390e-42
dtype: float64
```

Linear regression

In [5]:
```
# this is the standard import if you're using "formula notation" (similar to R)
import statsmodels.formula.api as smf

# create a fitted model in one line
lm = smf.ols(formula='Sales ~ TV', data=data).fit()

# print the coefficients
lm.params
```

Out[5]:
```
Intercept    7.032594
TV           0.047537
dtype: float64
```

In [14]:
```
# print the R-squared value for the model
lm.rsquared
```

Out[14]:
```
0.611875050850071
```

HW

- Work on projects!