Statistical Inference

Python & Statistics Bootcamp

NaLette M. Brodnax

The Institute for Quantitative Social Science
Harvard University

June 5, 2018

Set up

Jupyter notebook

1. Launch Jupyter from the command line:
   `jupyter notebook`, or from the Anaconda Navigator graphical interface
2. Navigate to the browser where your notebook is running
3. Create a new Python 3 notebook called `Inference`

Statistical inference

Estimation

Process of estimating the value of a population parameter from information obtained from a sample

- Point estimation involves using a sample to find the best estimate of a parameter of interest, such as the mean
- Interval estimation involves using a sample to find a range of parameter values
- Confidence refers to a measure of how certain we are that our interval estimate contains the true population parameter

Repeated, random sampling allows us to easily calculate or approximate the probabilities associated with different outcomes

Evaluating claims

A statistical hypothesis is a claim about a population parameter

- We cannot say whether a hypothesis is true or false without examining the entire population
- We can compare two hypotheses (e.g., status quo vs an alternative) and use evidence from a sample to say which one is less likely

Correlation is a measure of the strength of a relationship between two variables

- Positive $\longrightarrow$as x increases, y increases
- Negative $\longrightarrow$as x decreases, y decreases
- Zero $\longrightarrow$no relationship between x and y
- Linear $\longrightarrow$the correlation is constant across all values of x
- Non-linear $\longrightarrow$the correlation varies with x

Hypothesis testing

Use a test statistic to estimate the probability

1. State the **null** and **alternative** hypotheses
2. Compute the **test statistic**
3. Compare it to the **critical value** to **reject or fail to reject** the null hypothesis

Hypothesis testing

Use a test statistic to estimate the probability

1. State the **null** and **alternative** hypotheses
2. Compute the **test statistic**
3. Compare it to the **critical value** to **reject or fail to reject** the null hypothesis

$$\text{Test Statistic} = \frac{\text{observed value} - \text{expected value}}{\text{standard error}}$$

Population Parameter **Sample Parameter**	Mean, μ Mean, $\bar{X}$	Proportion, p, $(q = 1 - p)$ Proportion, $\hat{p}$, $(\hat{q} = \hat{p} - 1)$
Assumptions	random sample N > 30 or normal	binomial experiment $np \geq 5$ and $nq \geq 5$
Test	$Z = \frac{\bar{X} - \mu}{\sigma/\sqrt{n}}$	$Z = \frac{\hat{p} - p}{\sqrt{pq/n}}$
Distribution	Standard Normal	Standard Normal

Hypothesis testing

Example: Evaluate the claim that among 9-year-olds, boys are taller than girls, given a sample of 60 girls with mean height of 123.5 cm ($\sigma^2 = 98$) and a sample of 50 boys with mean height of 126.5 cm ($\sigma^2 = 120$)

- Parameter of interest: average height, μ
- Null hypothesis: no difference, $\mu_{boys} = \mu_{girls}$
- Alternative hypothesis (claim): $\mu_{boys} \neq \mu_{girls}$
- Test statistic: Z Test, standard normal distribution
- Critical value: $Z < -1.96$ or $Z > 1.96$
- Assumptions: random sample, N > 30

Hypothesis Testing

Construct data

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.axes as ax

mean_girls = 123.5
sd_girls = np.sqrt(98)
n_girls = 60

mean_boys = 126.2
sd_boys = np.sqrt(120)
n_boys = 50

ht_girls = np.random.normal(loc=mean_girls, scale=
    sd_girls, size=n_girls)
ht_boys = np.random.normal(loc=mean_boys, scale=
    sd_boys, size=n_boys)
```

Hypothesis Testing

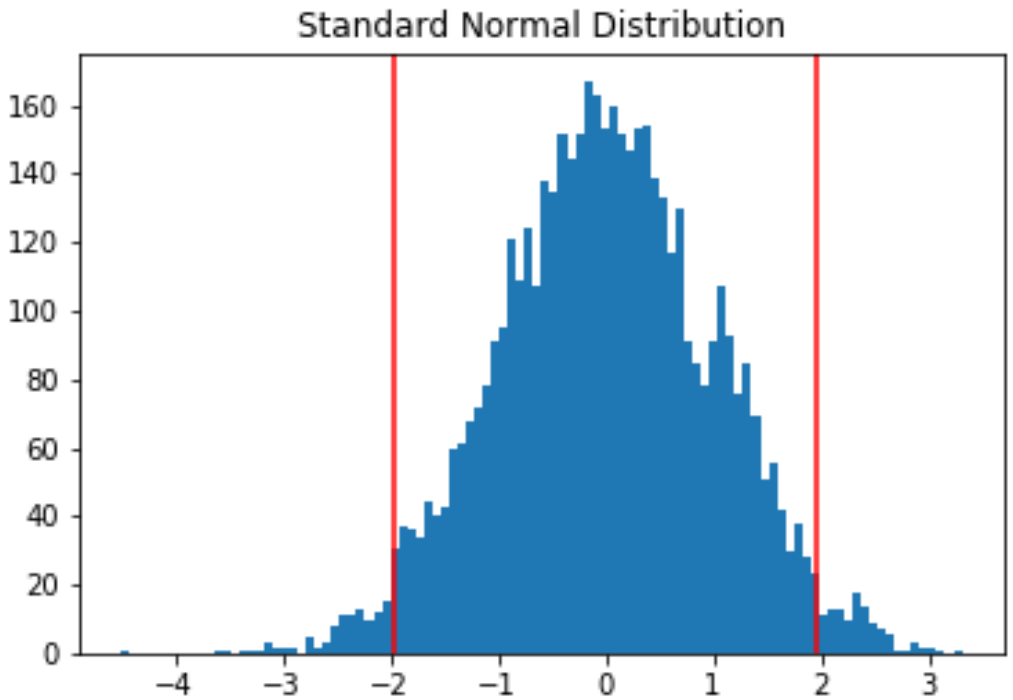
Test: $Z = \frac{(\bar{X}_g - \bar{X}_b) - (\mu_g - \mu_b)}{\sqrt{\frac{\sigma_g^2}{n_g} + \frac{\sigma_b^2}{n_b}}}$

```
obs_diff = mean_girls - mean_boys
exp_diff = 0
std_err = np.sqrt(((sd_girls**2)/n_girls) + ((
    sd_boys**2)/n_boys))

Z = (obs_diff - exp_diff)/std_err
```

Plot the standard normal distribution

```
stdn_data = np.random.randn(5000)
stdn = plt.figure()
plt.hist(stdn_data, bins=100)
plt.axvline(x=-1.96, color='r')
plt.axvline(x=1.96, color='r')
plt.title("Standard Normal Distribution")
plt.show()
```



Linear regression

Correlation between two variables is the parameter of interest

- Estimate by fitting a **regression line**, also known as the line of best fit, which minimizes the sum of the squares of the distance from each point to the line
- Many regression techniques, including ordinary least squares and maximum likelihood

Ordinary least squares

- Population model: $y = \beta_0 + \beta_1 X$
- Sample model: $\hat{y} = \hat{\beta}_0 + \hat{\beta_1} X + \epsilon$
- Interpretation: the intercept $\hat{\beta}_0$ is the estimated average value of $\hat{y}$ and $\hat{\beta}_1$ is the estimated change in y corresponding to a one-unit change in X

Note: Correlation does not imply causation!

The statsmodels package

Statsmodels provides routines for statistical testing and modeling

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
```

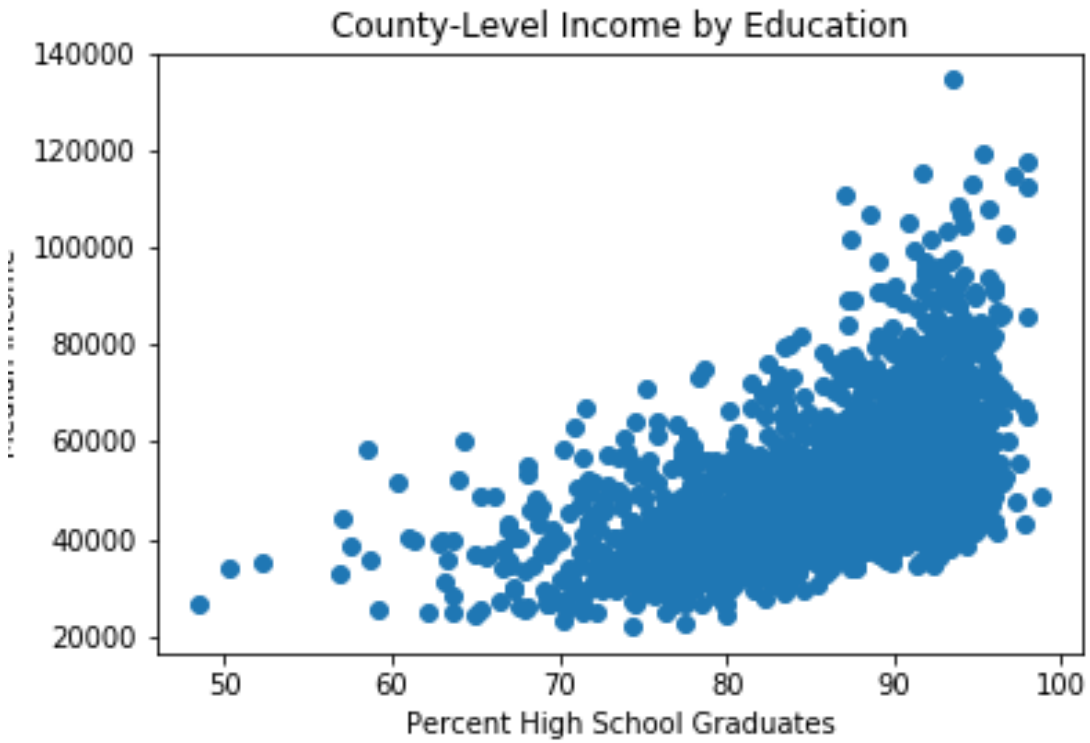
```
demo = pd.read_csv("county_demographics_2016.csv")
demo.head()
```

Optional: embed plots into Jupyter notebook

```
%matplotlib inline
```

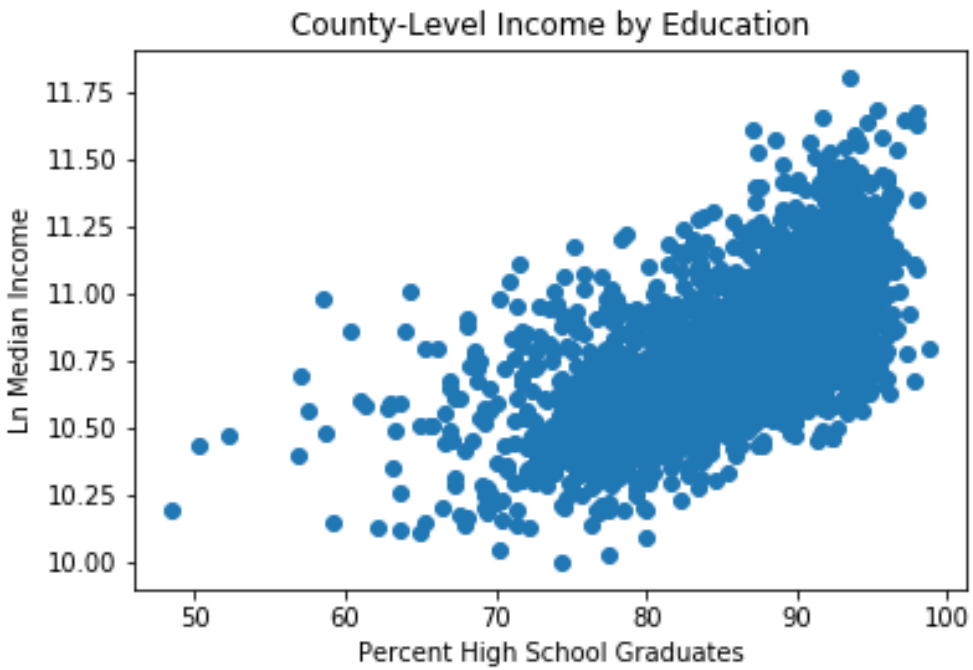
Exploratory visualizations

```
fig_inc = plt.figure()
plt.scatter(demo['pct_highschool'], demo['
    median_income'])
plt.title('County-Level Income by Education')
plt.xlabel('Percent High School Graduates')
plt.ylabel('Median Income')
plt.show()
```



Variable transformation

```
fig_ln_inc = plt.figure()
plt.scatter(demo['pct_highschool'], np.log(demo['
    median_income']))
plt.title('County-Level Income by Education')
plt.xlabel('Percent High School Graduates')
plt.ylabel('Ln Median Income')
plt.show()
fig_ln_inc.savefig('fig_ln_inc.png')
```



Linear regression

Ordinary least squares with statsmodels

```
demo = demo[['pct_highschool', 'median_income']].
    dropna()
demo.head()
```

```
model = smf.ols(formula='np.log(median_income) ~
    pct_highschool', data=demo)
est = model.fit()
est.summary()
```

Questions?