Data Science

Week 4

Statistics for Data Science

Review: Data Collection

- Data from Public (Open) Dataset
  - When you use the open dataset, you must follow the requirements of using the dataset (copyright, citation, acknowledge, and so on).
- Building a Dataset from Scratch
  - Consider your objectives (limitations if any), identifying your data requirements, and finally organize a data collection plan that synthesizes the most important aspects of your project.

Review: Data Loading and File Formats

Function	Description
`read_csv`	Load delimited data from a file, URL, or file-like object; use comma as default delimiter
`read_table`	Load delimited data from a file, URL, or file-like object; use tab (`'\t'`) as default delimiter
`read_fwf`	Read data in fixed-width column format (i.e., no delimiters)
`read_clipboard`	Version of `read_table` that reads data from the clipboard; useful for converting tables from web pages
`read_excel`	Read tabular data from an Excel XLS or XLSX file
`read_hdf`	Read HDF5 files written by pandas
`read_html`	Read all tables found in the given HTML document
`read_json`	Read data from a JSON (JavaScript Object Notation) string representation
`read_msgpack`	Read pandas data encoded using the MessagePack binary format
`read_pickle`	Read an arbitrary object stored in Python pickle format
`read_sas`	Read a SAS dataset stored in one of the SAS system's custom storage formats
`read_sql`	Read the results of a SQL query (using SQLAlchemy) as a pandas DataFrame
`read_stata`	Read a dataset from Stata file format
`read_feather`	Read the Feather binary file format

Review: Data Cleaning and Preprocessing

- Handling Missing Data
- Removing Duplicates
- Combining and Merging Datasets
- Time Series
- Exercise

Outline

1. Basics of Statistics for data science
2. Relationships between variables
3. Hypothesis testing

Mean

- By far the most common summary statistic is the mean.
- Mean describes the central tendency of the distribution.
- If you have a sample of n values, x_i, the mean, $\bar{x}$, is the sum of the values divided by the number of values:

$$\bar{x} = \frac{1}{n} \sum_i x_i$$

Variance

- Variance is a summary statistic intended to describe the variability or spread of a distribution. The variance of a set of values is

$$S^2 = \frac{1}{n}\sum_i (x_i - \bar{x})^2$$

- The term $x_i - \bar{x}$ is called the ***deviation*** from the mean, so **variance** is the mean squared deviation.
- The square root of variance, S, is the **standard deviation**.

Distribution: Histogram

- One of the best ways to describe a variable is to report the values that appear in the dataset and how many times each value appears. This description is called the distribution of the variable.
- The most common representation of a distribution is a histogram, which is a graph that shows the frequency of each value. In this context, “frequency” means the number of times the value appears.

Python code for Histogram

- In Python, an efficient way to compute frequencies is with a dictionary. Given a sequence of values, t:

```
 8    hist = {}
 9    t = [1, 2, 2, 3, 5]
10    for x in t:
11        hist[x] = hist.get(x, 0) + 1
```

- The result is a dictionary that maps from values to frequencies.

Visualization of Histogram

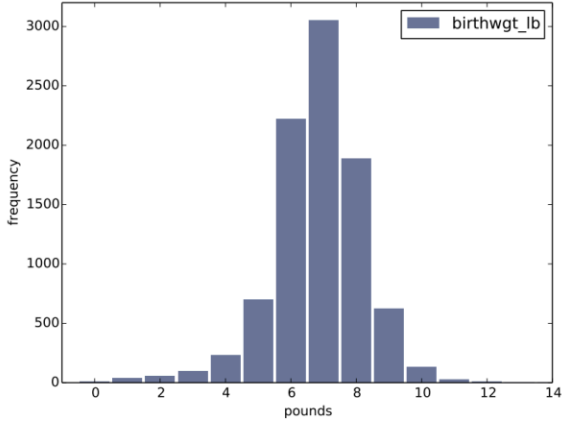
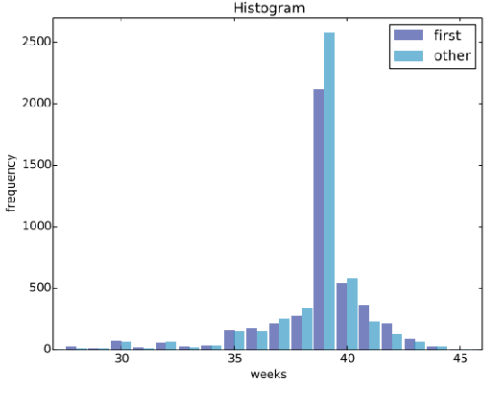


Figure 2.1: Histogram of the pound part of birth weight.

Probability Mass Function (PMF)

- A probability is a frequency expressed as a fraction of the sample size, n.



Hist. / n

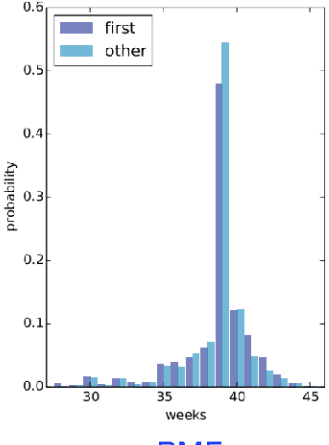
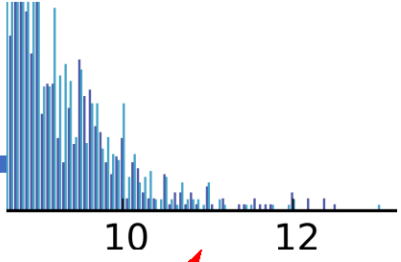


Figure 2.5: Histogram of pregnancy lengths.

From NSFG data

PMF

The Limits of PMFs



PMFs work well if the number of bins is small. But as the number of bins increases, the probability associated with each value gets smaller and the effect of random noise increases.

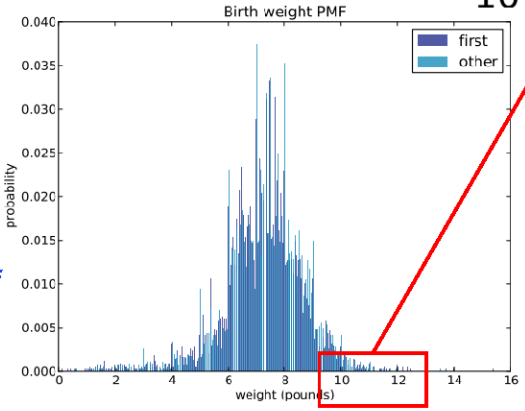


Figure 4.1: PMF of birth weights. This figure shows a limitation of PMFs: they are hard to compare visually.

Cumulative Distribution Function (CDF)

- To evaluate $\mathrm{CDF}(x)$ for a particular value of x, we compute the fraction of values in the distribution less than or equal to x.
- Here's what that looks like as a function that takes a sequence, sample, and a value, x:

```
def EvalCdf(sample, x):
    count = 0.0
    for value in sample:
        if value <= x:
            count += 1

    prob = count / len(sample)
    return prob
```

E.g. sample: [1, 2, 2, 3, 5]
EvalCdf (sample, 0) = 0
EvalCdf (sample, 1) = 0.2
EvalCdf (sample, 2) = 0.6
EvalCdf (sample, 3) = 0.8
EvalCdf (sample, 4) = 0.8
EvalCdf (sample, 5) = 1

Visualization of CDF

E.g. sample: [1, 2, 2, 3, 5]
EvalCdf (sample, 0) = 0
EvalCdf (sample, 1) = 0.2
EvalCdf (sample, 2) = 0.6
EvalCdf (sample, 3) = 0.8
EvalCdf (sample, 4) = 0.8
EvalCdf (sample, 5) = 1

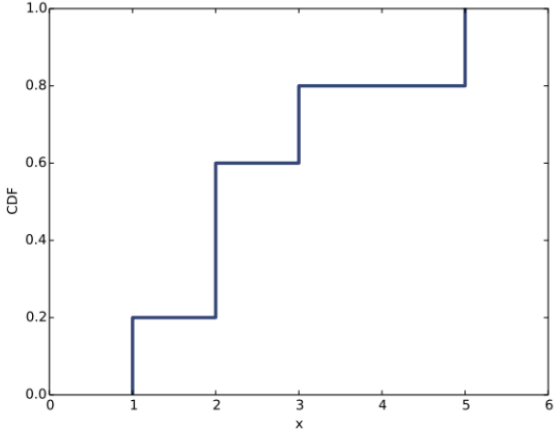


Figure 4.2: Example of a CDF.

Outline

1. Basics of Statistics for data science
2. Relationships between variables
3. Hypothesis testing

Covariance

- **Covariance** is a measure of the tendency of two variables to vary together.
- If we have two series, X and Y , their deviations from the mean are

$$dx_i = x_i - \bar{x}$$

$$dy_i = y_i - \bar{y}$$

- where $\bar{x}$ is the sample mean of X and $\bar{y}$ is the sample mean of Y . x_i and y_i are the elements of X and Y.

Calculation of Covariance

- Covariance is the mean of these products:

$$Cov(X,Y) = \frac{1}{n}\sum dx_i \; dy_i$$

- The covariance is maximized if the two vectors are identical, 0 if they are orthogonal (no correlation), and negative if they point in opposite directions.

Correlation

- A **correlation** is a statistic intended to quantify the strength of the relationship between two variables.

- A challenge in measuring correlation is that the variables we want to compare are often not expressed in the same units. (A&B vs. A&C)

- And even if they are in the same units, they come from different distributions.

How to Calculate the Correlation

1. Transform each value to a **standard score**, which is the number of standard deviations from the mean. This transform leads to the "*Pearson product-moment correlation coefficient*."

2. Transform each value to its **rank**, which is its index in the sorted list of values. This transform leads to the "*Spearman rank correlation coefficient*."

Pearson's correlation -1

- Covariance is useful in some computations, but it is seldom reported as a summary statistic because it is hard to interpret.
- Among other problems, its units are the product of the units of X and Y .
- One solution to this problem is to divide the deviations by the standard deviation.

Pearson's correlation -2

- One solution to this problem is to divide the deviations by the standard deviation.

$$p_i = \frac{(x_i - \bar{x})}{S_X} \frac{(y_i - \bar{y})}{S_Y}$$

Where S_X and S_Y are the standard deviations of X and Y . The mean of these products is

$$\rho = \frac{1}{n} \sum p_i \qquad \rho = \frac{Cov(X, Y)}{S_X S_Y}$$

- This value is called **Pearson's correlation**

Nonlinear relationships

- If Pearson's correlation is near 0, it is tempting to conclude that there is no relationship between the variables, but that conclusion is not valid.

- Pearson's correlation only measures linear relationships. If there's a nonlinear relationship, ρ understates its strength.

Example of Pearson's correlation

- The top row shows linear relationships with a range of correlations; you can use this row to get a sense of what different values of ρ look like.
- The second row shows perfect correlations with a range of slopes, which demonstrates that correlation is unrelated to slope.
- The third row shows variables that are clearly related, but because the relationship is nonlinear, the correlation coefficient is 0.

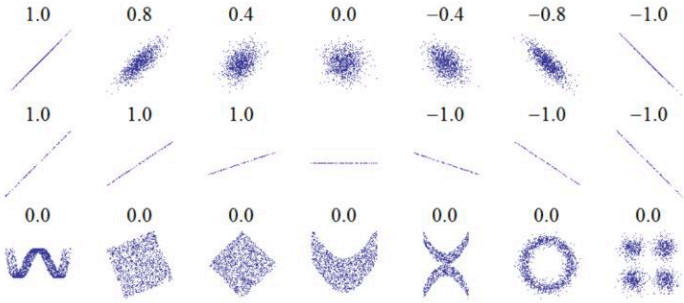


Figure 7.4: Examples of datasets with a range of correlations.

You should always look at a scatter plot of your data before blindly computing a correlation coefficient.

Spearman's rank correlation

- Pearson's correlation works well if the relationship between variables is linear and if the variables are roughly normal.
- But it is not robust in the presence of outliers.
- Spearman's rank correlation is an alternative that mitigates the effect of outliers and skewed distributions.

```
def SpearmanCorr(xs, ys):
    xranks = pandas.Series(xs).rank()
    yranks = pandas.Series(ys).rank()
    return Corr(xranks, yranks)
```

Correlation and causation

- If variables A and B are correlated, there are three possible explanations:
  - A causes B,
  - or B causes A,
  - or some other set of factors causes both A and B.

- These explanations are called "*causal relationships*".

- Correlation does not imply causation

So what can you do to provide evidence of causation? -1

1. Use time.

If A comes before B, then A can cause B.

The order of events can help us infer the direction of causation.

But it does not preclude the possibility that something else causes both A and B.

So what can you do to provide evidence of causation? -2

2. Use randomness.

E.g. Causation: treatment → Response?

A randomized controlled trial is a type of scientific (often medical) experiment that aims to reduce certain sources of bias when testing the effectiveness of new treatments;

This is accomplished by randomly allocating subjects to two or more groups, treating them differently, and then comparing them with respect to a measured response.

https://en.wikipedia.org/wiki/Randomized_controlled_trial

Real World Use Case of Correlation

- Let's imagine you lend a large sum of money to a company named ABC for a year. ABC promises to give you your money with interest back in a years time. You are worried that the company ABC might default and to protect yourself from that risk, you decide to buy insurance from an insurance company named XYZ.
- Now let's also assume that everyone who has lent money to ABC has also bought the insurance from the insurance company XYZ.
- Can you see what will happen if ABC defaults?

https://medium.com/fintechexplained/did-you-know-the-importance-of-finding-correlations-in-data-science-1fa3943debc2

Real World Use Case of Correlation

- If ABC defaults then everyone will reach out to XYZ and expect to get their money back from them. As a consequence, XYZ might default and you would lose your money.
- This is because there is a strong positive correlation between the companies ABC and XYZ.
- If we knew the correlation up-front, we would have bought insurance from another company and would have saved ourselves from losing the money!
- What I have just explained above is the concept of financial trade known as CDS (Credit default swap) and the risk is known as WWR (wrong-way risk) Correlation Risk.

https://medium.com/fintechexplained/did-you-know-the-importance-of-finding-correlations-in-data-science-1fa3943debc2

Outline

1. Basics of Statistics from the point of view of data science
2. Relationships between variables
3. Hypothesis testing (Basics)

A Dataset for this Class

- Since 1973 the U.S. Centers for Disease Control and Prevention (CDC) have conducted the National Survey of Family Growth (NSFG), which is intended to gather information on family life, marriage and divorce, pregnancy, infertility, use of contraception, and men's and women's health.
- The survey results are used to plan health services and health education programs, and to do statistical studies of families, fertility, and health." See http://cdc.gov/nchs/nsfg.htm
- We will use data collected by this survey to investigate whether first babies tend to come late, and other questions.

Hypothesis testing

- The fundamental question we want to address is whether the effects we see in a **sample** are likely to appear in the larger **population**.

Population ≡ all possible values

Sample ≡ a portion of the population

- It is a common question in data science.

Example of hypothesis testing

- For example, in the example data we see a difference in mean pregnancy length for first babies and others.
- For example, in the NSFG sample we see a difference in mean pregnancy length for first babies and others.
- We would like to know if that effect reflects a real difference for women in the U.S., or if it might appear in the sample by chance.

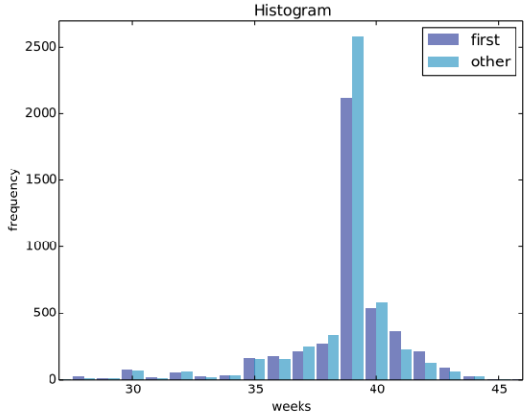


Figure 2.5: Histogram of pregnancy lengths.

The goal of hypothesis testing

- The goal of classical hypothesis testing is to answer the question,

"Given a sample and an apparent effect, what is the probability of seeing such an effect by chance?"

How we answer that question:

- The first step is to quantify the size of the apparent effect by choosing a **test statistic (mean, probability…)**.
- The second step is to define a **null hypothesis**, which is a model of the system based on the assumption that the apparent effect is not real.
- The third step is to compute a ***p*-value**, which is the probability of seeing the apparent effect if the null hypothesis is true.
- The last step is to interpret the result. If the p-value is low, the effect is said to be **statistically significant**, which means that it is unlikely to have occurred by chance.

1. Test statistic

- The first step is to quantify the size of the apparent effect by choosing a **test statistic**.

- In the NSFG example, the apparent effect is *a difference in pregnancy length between first babies and others*, so a natural choice for the test statistic is the difference in **means** between the two groups.

2. Null hypothesis

- The second step is to define a **null hypothesis**, which is a model of the system based on the assumption that the apparent effect is NOT real.

- In the NSFG data, we saw (from the histogram) that the mean pregnancy lengths for first babies and other babies are slightly different.

- In the NSFG example the null hypothesis (H_0) is that there is no difference between the pregnancy lengths of first babies and others; that is, that pregnancy lengths for both groups have the same distribution.

3. compute a *p-value*

- The third step is to compute a ***p-value***, which is the probability of seeing the apparent effect if the null hypothesis is true.

- In the NSFG example, we would compute the actual difference in means, then compute the probability of seeing a difference under the null hypothesis.

4. Interpretation for *p*-value -1

- The last step is to interpret the result. If the *p-value* is low, the effect is said to be **statistically significant**, which means that it is unlikely to have occurred by chance (***H_0 is false in this case***).

- In that case we infer that the effect is more likely to appear in the larger population.

4. Interpretation for *p*-value -2

- What is the probability of the observed test statistic … **when H_0 *is true*?**
  - *P*-value is large→ Observed effect is not a small probability event → Observed effect is likely to have occurred by chance

- Thus, smaller and smaller *P*-values provide stronger and stronger evidence against H_0

Interpretation for *p*-value

Conventions*

$p > 0.10 \Rightarrow$ non-significant evidence against H_0

$0.05 < p \leq 0.10 \Rightarrow$ marginally significant evidence

$0.01 < p \leq 0.05 \Rightarrow$ significant evidence against H_0

$p \leq 0.01 \Rightarrow$ highly significant evidence against H_0

Examples

$P = 0.27 \Rightarrow$ non-significant evidence against H_0

$p = 0.01 \Rightarrow$ highly significant evidence against H_0

How we answer that question:

- The third step is to compute a **p-value**, which is the probability of seeing the apparent effect if the null hypothesis is true.

- In the NSFG example, we would compute the actual difference in means, then compute the probability of seeing a difference as big, or bigger, under the null hypothesis.

Outline

1. Basics of Statistics from the point of view of data science
2. Relationships between variables
3. Hypothesis testing (Extension)

Two-sided test statistic

- The first step is to quantify the size of the apparent effect by choosing a **test statistic**.

- In the NSFG example, the apparent effect is a difference in pregnancy length between first babies and others, so a natural choice for the test statistic is the difference in **means** between the two groups. This is two-sided test statistic

Two-sided

```
def TestStatistic(self, data):
    group1, group2 = data
    test_stat = abs(group1.mean() - group2.mean())
    return test_stat
```

Two-side test statistic using absolute difference in means

- Choosing the best test statistic depends on what question you are trying to address.

- For example, if the relevant question is whether pregnancy lengths are different for first babies, then it makes sense to test the absolute difference in means, as we did in the previous slides.

Other test statistics - one-sided

- If we had some reason to think that first babies are likely to be late, then we would not take the absolute value of the difference
- We would use this test statistic:

```
class DiffMeansOneSided(DiffMeansPermute):

    def TestStatistic(self, data):
        group1, group2 = data
        test_stat = group1.mean() - group2.mean()
        return test_stat
```

one-sided

Other test statistics - standard deviation

- We can use the same framework to test for a difference in **standard deviation**.
- We may saw some evidence that first babies are more likely to be early or late, and less likely to be on time.
- So we might hypothesize that the standard deviation is higher.

```
def TestStatistic(self, data):
    group1, group2 = data
    test_stat = group1.std() - group2.std()
    return test_stat
```

Errors

- In classical hypothesis testing, an effect is considered statistically significant if the p-value is below some threshold, commonly 5%. This procedure raises two questions:
- If the effect is actually due to chance, what is the probability that we will wrongly consider it significant? This probability is the false positive rate (type Ⅰ error).
- If the effect is real, what is the chance that the hypothesis test will fail? This probability is the false negative rate (type Ⅱ error).

Summary

1. Basics of Statistics from the point of view of data science

2. Relationships between variables

3. Hypothesis testing

Reference

- McKinney, Wes. Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. " O'Reilly Media, Inc.", 2012.

- Downey, A., 2014. *Think stats: exploratory data analysis*. " O'Reilly Media, Inc.".

Data Science

Week 5
Data Visualization

```python
# for Data Sciecnce Week-4
"""This source code is created by referring to the book
"Think Stats", by Allen B. Downey"""

import numpy as np
import pandas as pd
import random

def RandomSeed(x):
    """Initialize the random and np.random generators. x: int seed """
    random.seed(x)
    np.random.seed(x)

def MakeFrames():
    """Reads pregnancy data and partitions first babies and others.
    returns: DataFrames (all live births, first babies, others)"""

    # load the data "preg" from csv file
    preg = pd.read_csv('2002FemPreg.csv', index_col=0)
    # create new data "live" by picking up the data whose outcome column is 1
    live = preg[preg.outcome == 1]
    # create new data "firsts" by picking up the data whose birthord column is 1
    firsts = live[live.birthord == 1]
    # create new data "others" by picking up the data whose birthord column is 1
    others = live[live.birthord != 1]

    return live, firsts, others

class DiffMeansPermute():
    def __init__(self, data):
        """Initializes.
        It builds a representation of the null hypothesis,
        then passes the data to TestStatistic, which computes the
        size of the effect in the sample.
        data: data in whatever form is relevant"""
        self.data = data
        self.MakeModel()
        # self.actual is the effect observed from the sample.
        self.actual = self.TestStatistic(data)
        self.test_stats = None


    def PValue(self, iters=1000):
        """Computes the distribution of the test statistic and p-value.
        iters: number of iterations
        PValue computes the probability of the apparent effect
        under the null hypothesis. It takes as a parameter iters,
        which is the number of simulations to run.
        returns: float p-value """

        # The first line generates simulated data, computes test statistics
        # for the simulated data in each iteration
        # The simulated data is explained in RunModel()
        self.test_stats = [self.TestStatistic(self.RunModel())
                           for _ in range(iters)]

        # count the number of iterations whose
        # test statistic of simulated sample >= test statistic of actual sample.
        # It means that the effect in the simulated data is
        # more obvious than actual sample.
        count = sum(1 for x in self.test_stats if x >= self.actual)
        # return the probability
        return float(count) / float(iters)

    def MaxTestStat(self):
        """Returns the largest test statistic seen during simulations."""
```

```
67            return max(self.test_stats)
68
69        def TestStatistic(self, data):
70            """Computes the test statistic.
71            data: data in whatever form is relevant """
72            group1, group2 = data
73            test_stat = abs(group1.mean() - group2.mean())
74            return test_stat
75
76        def MakeModel(self):
77            """Build a model of the null hypothesis."""
78            group1, group2 = self.data
79            self.n, self.m = len(group1), len(group2)
80            # Stack arrays in sequence horizontally (column wise).
81            self.pool = np.hstack((group1, group2))
82
83        def RunModel(self):
84            """Run the model of the null hypothesis.
85            ***In the real experiment, we should collect the new sample data
86            ***for each iteration, compare the test statistic to calculate pvalue.
87            ***Here, we generate the new sample data (simulated data)
88            ***by shuffle the data we have.
89            ***Some data moves from "firsts" category to "others" category
90            *** e.g. (birthord: 1-->other values)
91            ***and some data moves from "others" category to "firsts" category
92            ***e.g. (birthord: other values to 1)
93            ***
94            returns: simulated data (pick up samples from population)"""
95            np.random.shuffle(self.pool)
96            data = self.pool[:self.n], self.pool[self.n:]
97            return data
98
99    def main():
100       RandomSeed(100)
101
102       # Load the data from csv file
103       live, firsts, others = MakeFrames()
104       data = firsts.prglngth.values, others.prglngth.values
105       # Hypothesis test
106       ht = DiffMeansPermute(data)
107       # Out put the p-value
108       print (ht.PValue())
109
110       # reuslt is 0.173
111       # the null hypothesis is likely to be true.
112       # there is no difference between the pregnancy lengths
113       # of first babies and others
114
115   if __name__ == "__main__":
116       main()
```