statics for data science

Data comes from many sources: sensor measurements, events, text, images, and videos. The Internet of Things (IoT) is spewing out streams of information

Clickstreams are sequences of actions by a user interacting with an app or a web page.

Much of this data is unstructured: images are a collection of pixels, with each pixel containing RGB (red, green, blue) color information

To apply the statistical concepts covered in this book, unstructured raw data must be processed and manipulated into a structured form

One of the commonest forms of structured data is a table with rows and columns—as data might emerge from a relational database or be collected for a study.

- There are two basic types of structured data: numeric and categorical.
- Numeric data comes in two forms:
- continuous, such as wind speed or time duration, and discrete, such as the count of the occurrence of an event.
- Categorical data takes only a fixed set of values, such as a type of TV screen (plasma, LCD, LED, etc.) or a state name (Ala- bama, Alaska, etc.).
- Binary data is an important special case of categorical data that takes on only one of two values, such as 0/1, yes/no, or true/false.
- Another useful type of categorical data is ordinal data in which the categories are ordered;
- ordinal data in which the categories are ordered; an example of this is a numerical rating (1, 2, 3, 4, or 5).

Why do we bother with a taxonomy of data types? know about it very important

Numeric

- Data that are expressed on a numeric scale.

Continuous

- data that can take on any value in an interval (Synonyms: interval, float, numeric)) un countable are exact value

Discrete

- Data that can take on only integer values, such as counts. (Synonyms: integer, count) countable values

Categorical

- Data that can take on only a specific set of values representing a set of possible categories. (Synonyms: enums, enumerated, factors, nominal (example : nominal data sex :1 male , 2 female . order doesn't matter))

Binary

- A special case of categorical data with just two categories of values, e.g., 0/1, true/false. (Synonyms: dichotomous, logical, indicator, Boolean)

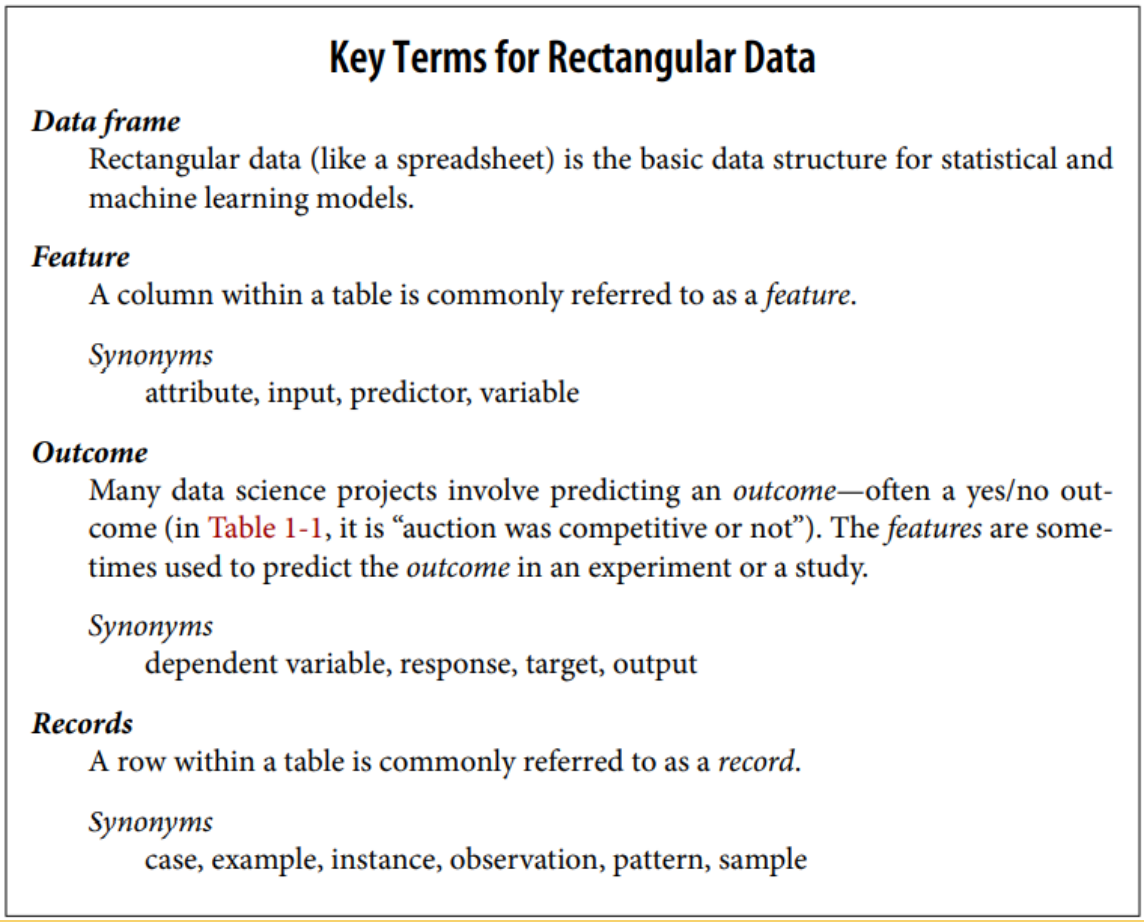
Ordinal

- Categorical data that has an explicit ordering. (Synonym: ordered factor) rating using the numbers order is matter to analysis data
- Knowing that data is categorical can act as a signal telling software how statistical procedures, such as producing a chart or fitting a model, should behave.
- categorical data means all data number or text values help to learn in sklearn.preprocessing.OrdinalEncoder.

Rectangular Data

- The typical frame of reference for an analysis in data science is a rectangular data object, like a spreadsheet or database table.
- Rectangular data is the general term for a two-dimensional matrix with rows indicat- ing records (cases) and columns indicating features (variables);

- rectangular data means it was structured data
- A row within a table is commonly referred to as a record.



Key Terms for Rectangular Data

Data frame

Rectangular data (like a spreadsheet) is the basic data structure for statistical and machine learning models.

Feature

A column within a table is commonly referred to as a *feature*.

Synonyms

attribute, input, predictor, variable

Outcome

Many data science projects involve predicting an *outcome*—often a yes/no outcome (in Table 1-1, it is "auction was competitive or not"). The *features* are sometimes used to predict the *outcome* in an experiment or a study.

Synonyms

dependent variable, response, target, output

Records

A row within a table is commonly referred to as a *record*.

Synonyms

case, example, instance, observation, pattern, sample

-

### NAMES(variable)	### age	### class	### percentage	### year
unknow(row record)	16	8th	80	1992
unknow 2	16	8th	81	1992

Nonrectangular Data Structures

- means unstructured data
- The field view, by contrast, focuses on small units of space and the value of a relevant metric (pixel brightness, for example).
- this is mainly used full machine learning
- The focus of this book is on rectangular data, the fundamental building block of predictive modeling.
- For example, a graph of a social network, such as Facebook or LinkedIn, may represent connections between people on the network. Distribution hubs connected by roads are an example of a physical network.(graph theory)

Graphs in Statistics

- In computer science and information technology, the term graph typically refers to a depiction of the connections among entities, and to the underlying data structure. In statistics, graph is used to refer to a variety of plots and visualizations, not just of connections among entities, and the term applies only to the visualization, not to the data structure.

Key Ideas

- The basic data structure in data science is a rectangular matrix in which rows are records and columns are variables (features).
- Terminology can be confusing; there are a variety of synonyms arising from the different disciplines that contribute to data science (statistics, computer science, and information technology).

Estimates of Location

A basic step in exploring your data is getting a “typical value” for each feature (variable): an estimate of where most of the data is located (i.e., its central tendency means mean, mode median how data is spread).

these are single column data

- **Mean --- averag**

$$MEAN = \frac{\sum_{i=1}^{n} x_i}{n}$$
$$example : sum\ of\ numbers\ /\ total\ numbers$$
$$example : (2+5+6+7+8+9)/6$$

- **Weighted mean --- weighted average**

```
#weighted mean code
xi = np.array([10,20,30,40,50,60])
wi = np.array([50,60,70,8,90,9])

out = xi*wi
z = np.sum(out)
o = np.sum(wi)
weightec = z/o
print(weightec)

oo = np.average(k["Gold"], weights  = k["Silver"])
print(oo)
p = np.average(xi,weights = wi)
print(p)
```

- $$weighted\ MEAN = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$
    - Some values are intrinsically more variable than others, and highly variable observations are given a lower weight. For example, if we are taking the average from multiple sensors and one of the sensors is less accurate, then we might down weight the data from that sensor.
      - The data collected does not equally represent the different groups that we are interested in measuring. For example, because of the way an online experiment was conducted, we may not have a set of data that accurately reflects all groups in the user base. To correct that, we can give a higher weight to the values from the groups that were underrepresented.

- **Median --- 50th percentile center value**

```
> median(state[['Population']])
[1] 4436370
```

- **Percentile --- quantile 75% ,50% 25% these are quantiles**

```
prenctial arrange all data in lower to higher when you enter 75 % you got 25 the values are all
less than < 25
```

- **Weighted median (The value such that one-half of the sum of the weights lies above and below the sorted data.)**
- **Trimmed mean -- truncated mean**

$$TRIMMED \quad MEAN \quad remove \quad the \quad ouliers = \frac{\sum_{i=p+1}^{n-p} x_i}{n - 2_p}$$

$$if \quad you \quad to \quad calculate \quad trimmed \quad mean \quad the \quad data \quad set \quad or \quad values \quad must \quad be \quad in \quad either \quad low \quad to \quad high \quad are \quad high \quad to \quad low$$

```
TRIMMED MEAN basically it remove the outliers if you 10% trimmed means it will removes both sides
10% data
also clear the robust values
data set = [10,20,30,40,50,80,90,100]

 if you 10% trimmed means then you will  it will delete the last and first and numbers
 if you want 25% trimmed means it will delete first and last 5 numbers
```

```
from scipy.stats import trim_mean

print('it was a trimmed mean ',trim_mean(k['Bronze'], 0.1)) #10% data was trimmed
```

- **Robust -- resistant example (1,2,3,5,6, 100) here 100 is robust it will effected the mean(Not sensitive to extreme values)**
- **Outlier --- extreme value**
-

```
          print(data_name.describe()) it will show all
Rank        Gold      Silver      Bronze        Total   Rank by Total

count   93.000000   93.000000   93.000000   93.000000    93.000000       93.000000

mean    46.333333    3.655914    3.634409    4.322581    11.612903       43.494624

std     26.219116    7.022471    6.626339    6.210372    19.091332       24.171769

min      1.000000    0.000000    0.000000    0.000000     1.000000        1.000000

25%     24.000000    0.000000    0.000000    1.000000     2.000000       23.000000

50%     46.000000    1.000000    1.000000    2.000000     4.000000       47.000000

75%     70.000000    3.000000    4.000000    5.000000    11.000000       66.000000

max     86.000000   39.000000   41.000000   33.000000   113.000000       77.000000
```

Example: Location Estimates of Population and Murder Rates

	State	Population	Murder rate	Abbreviation
1	Alabama	4,779,736	5.7	AL
2	Alaska	710,231	5.6	AK
3	Arizona	6,392,017	4.7	AZ
4	Arkansas	2,915,918	5.6	AR
5	California	37,253,956	4.4	CA
6	Colorado	5,029,196	2.8	CO
7	Connecticut	3,574,097	2.4	CT
8	Delaware	897,934	5.8	DE

```
import pandas as pd

state = pd.read_csv('state.csv')
state['population'].mean()
trim_mean(state['population'], 0.1) # here 0.1 means 10% trimmed mean
state['populaion'].median()
```

weighted mean is available with Numpy . for weighted median we can specialized packages

example : Solved Example of Weighted Mean Suppose a marketing firm conducted a survey of 1,000 households to determine the average number of TVs each household owns. The data shows that there are more households with two or three TVs and a few numbers with one of four. Every household in the sample has at least one TV and not a household has more than four. Calculate the mean number of TVs per household.

Number of TVs household	Number of Households
1	73
2	378
3	459
4	90

Finally divide the numerator value by the denominator value.

$$\frac{\sum_{i=1}^{4} w_i x_i}{\sum_{i=1}^{4} w_i}$$

$$= \frac{2566}{1000}$$

$$= 2.566$$

Hence, the mean number of TVs per household in this sample is 2.566

Note: The weighted mean can be easily influenced by an outlier in our data. If we have very high or very low values in our data set, then we cannot rely on the weighted mean.

Here most of the values in this data set are repeated multiple times, we can easily compute the sample mean as a weighted mean. Following are steps to calculate the weighted arithmetic mean

```
import numpy as np

np.average(state['Murder.Rate'], weights=state['Population'])
wquantiles.median(state['Murder.Rate'], weights=state['Population'])
```

- Estimates of Variability :A second dimension, variability, also referred to as dispersion, measures whether the data values are tightly clustered or spread out. At the heart of statistics lies variability (tells how the data is spread)

1. Deviations
2. Variance : The term variance refers to a statistical measurement of the spread between numbers in a data set. More specifically, variance measures how far each number in the set is from the mean (average), and thus from every other number in the set (how data was spread from the mean to the data points) https://www.investopedia.com/ask/answers/021215/what-difference-between-standard-deviation-and-variance.asp

The deviations from the mean are the differences: 1 – 3 = –2, 4 – 3 = 1, 4 – 3 = 1. These deviations tell us how dispersed the data is around the central value.

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{N}$$

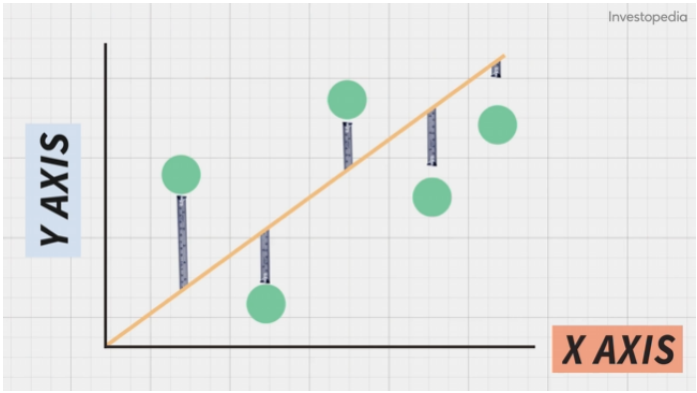
where:

x_i = Each value in the data set

$\overline{x}$ = Mean of all values in the data set

N = Number of values in the data set

1.



$$varaince \quad = s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

2. Standard deviation (tell how devotions are spread it means difference between the values if standard deviation is 10 right side you will add 10 example : the values is 190 add 10 = 200 , you move left side 190 subtract 10 = 180)Standard deviation is a statistical measurement that looks at how far a group of numbers is from the mean.

```
from statistics import variance
from statistics import stdev

print(variance((k['Gold'])))
# stdev is squar root of variance
print(stdev(k['Gold']))
```

$$Standard \quad deviation = \sqrt{variance}$$

3. Mean absolute deviation (this is tell that how data points are spread out from mean . if you got less value it data is spread low if you high value the data is spreader very high) # standard absolute deviation== l1-norm, Manhattan norm (we get the positive values)

$$MAD = \sum_{i-1}^{n} \frac{|x_i - \overline{x}|}{n}$$

Where

x_i = Input data values

$\bar{x}$ = Mean value for a given set of data,

n = Number of data values

x_i	$x_i - \overline{x}$	$\lvert x_i - \overline{x} \rvert$
26	-8	8
46	12	12
56	22	22
45	11	11

==

$$MEAN \quad absolute \quad deviation = \frac{\sum_{i=1}^{n} \mid x_i - \bar{x} \mid}{n}$$

4.
```
from numpy import mean ,absolute
#mean and then absolute devation of data points and substract from and mul by mean you got absolute deavtion
print(mean(absolute(k['Gold']) - mean(k['Gold'])))
```

5. Median absolute deviation from the median (this is tell that how data points are spread out from mean . if you got less value it data is spread low if you high value the data is spreader very high)

$$Median \quad absolute \quad deviation \quad = median(\mid x_1 - m \mid, \mid x_2 - m \mid, ... \mid x_n - m \mid)$$

```
from numpy import median ,absolute
#median and then absolute devation of data points and substract from and mul by median you got absolute deavtion
print(median(absolute(k['Gold']) - median(k['Gold'])))
```

6. Range (max -min) this is called range(data must be in shorted)

```
import numpy as np
#creating a data set
data = np.arange(20,200,4)
```

```
max = data.max()
min = data.min()

range_of_the_value = max - min
print(range_of_the_value)
```

7. Order statistics (Metrics based on the data values sorted from smallest to biggest.)
8. Percentile (it tells the data was below the length we calculate 75th percentile we got value 92 the over all value is 92)(data must be in shorted list)

$$P_x = \frac{x(n + 1)}{100}$$

P_x = The value at which x percentage of data lie below that value

n = Total number of observations

9.

```
q3, q1 = np.percentile(k['Gold'] ,[75,25])

#using pandas

print(data.quantile([0.25,0.5,0.75]))
```

10. Interquartile range (The difference between the 75th percentile and the 25th percentile.) this is also called iQR(data must be in shorted)

```
# tell the how big is middle 50% of data indicates with iqr
iqr = q3-q1
print(iqr)


# you can find the iqr is also quantile  take 25% data and 50% and 75% is lies between
all data is lies between first iqr shows first petal values are lies between like that
```

- **Standard deviation : tell that how much it was spread and tell that length of the points**
-

```
  state['Population'].std()

    state['Population'].quantile(0.75) - state['Population'].quantile(0.25)

robust.scale.mad(state['Population'])

#standed devation , quantile , mad mean abosulate devation
```

Degrees of Freedom, and n or n–1?

If you use the intuitive denominator of n in the variance formula, you will underset-mate the true value of the variance and the standard deviation in the population. This is referred to as a biased estimate. However, if you divide by n – 1 instead of n, the variance becomes an unbiased estimate To fully explain why using n leads to a biased estimate involves the notion of degrees of freedom, which takes into account the number of constraints in computing an esti-mate. In this case, there are n – 1 degrees of freedom since there is one constraint: the standard deviation depends on calculating the sample mean. For most problems, data scientists do not need to worry about degrees of freedom.

Exploring the Data Distribution

1. **Boxplot (this is the tell how outliers are extended)**
2. **Frequency Tables and Histograms **

3. **Histogram**
4. **Density plot (A smoothed version of the histogram, often based on a kernel density estimate)**

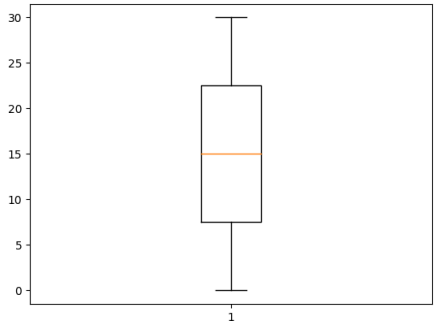
Boxplots, introduced by Tukey [Tukey-1977], are based on percentiles and give a quick way to visualize the distribution of data.

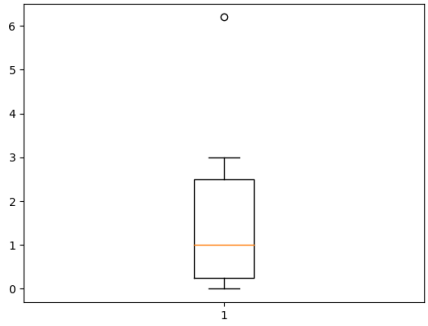
skewness and kurtosis.

Percentiles and Boxplots

```
import pandas  as pd
import matplotlib.pyplot as plt
#s = 'E:\\Medals.xlsx'
#import warnings
#warnings.simplefilter("ignore")
s = pd.read_excel('E:\\Medals.xlsx',engine='openpyxl')
#print(s)
plt.boxplot(s['Total'])
```

```
state['Murder.Rate'].quantile([0.05, 0.25, 0.5, 0.75, 0.95])
# find the perctial of quantile about data how it was going
```





this percentiles plot show how it was spread outliers

we explored how percentiles can be used to measure the spread of the data. Percentiles are also valuable for summarizing the entire distribution. Percentiles are especially valuable for summarizing the tails (the outer range) of the distribution.

Density Plots and Estimates

Frequency Tables and Histograms : frequency table means grouped data are structure data ==

==

vlaue	frequency (how many values)
10 -20	5
20 - 40	10
40-50	1
50-60	2
60-70	10

```
#for frequency table using this commands
po = pd.cut(k['Gold'],10) #10 how must you will sperate
print(po.value_counts())
```

statistical moments :

they are four types of moments to observe in graphs or hist,plot

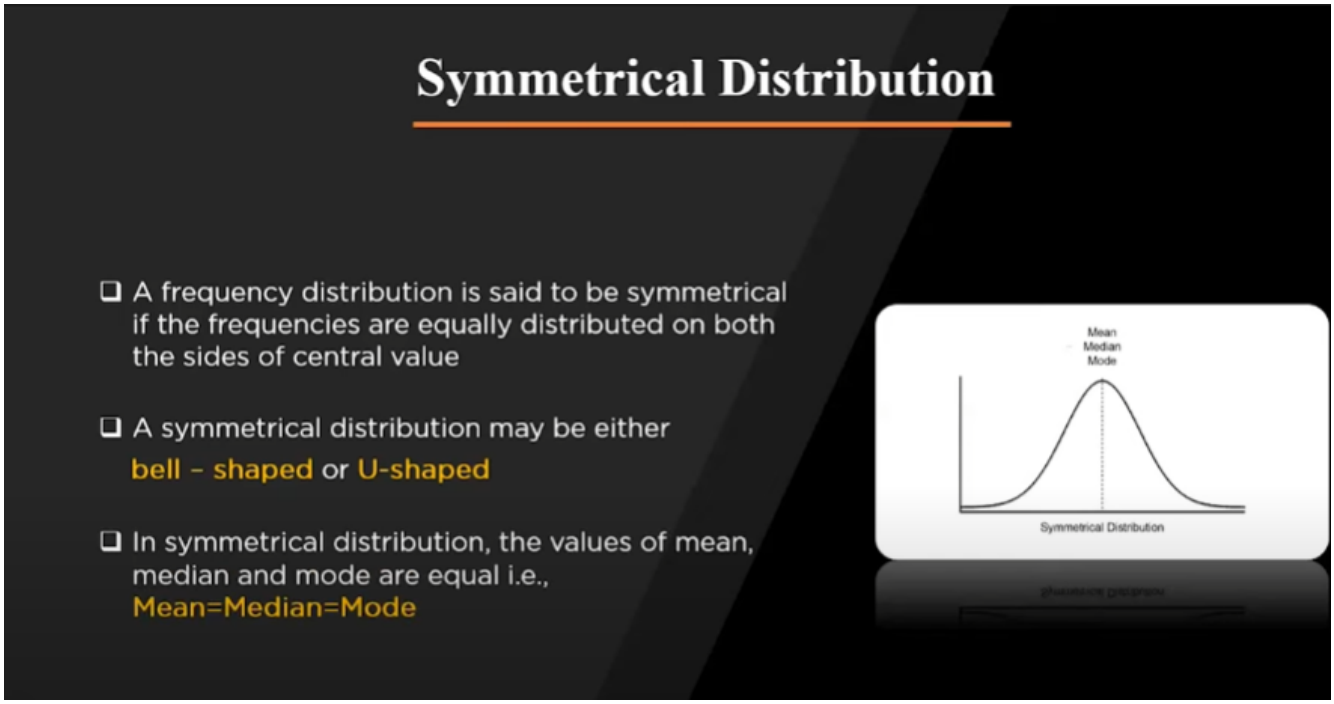
1) first moments (mean) : average of the data

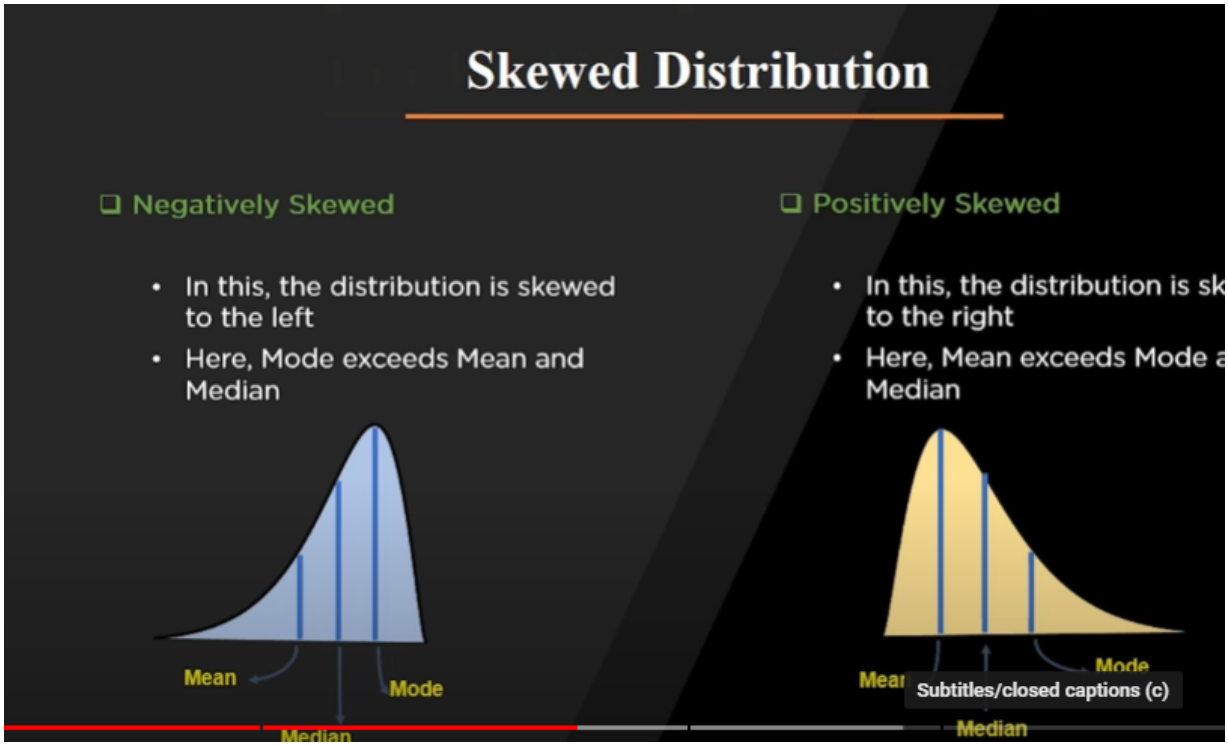
2. second moments : how values are spread and center tendency

3. third moments :skewed distribution

4) fourth moments :Kurtosis distribution

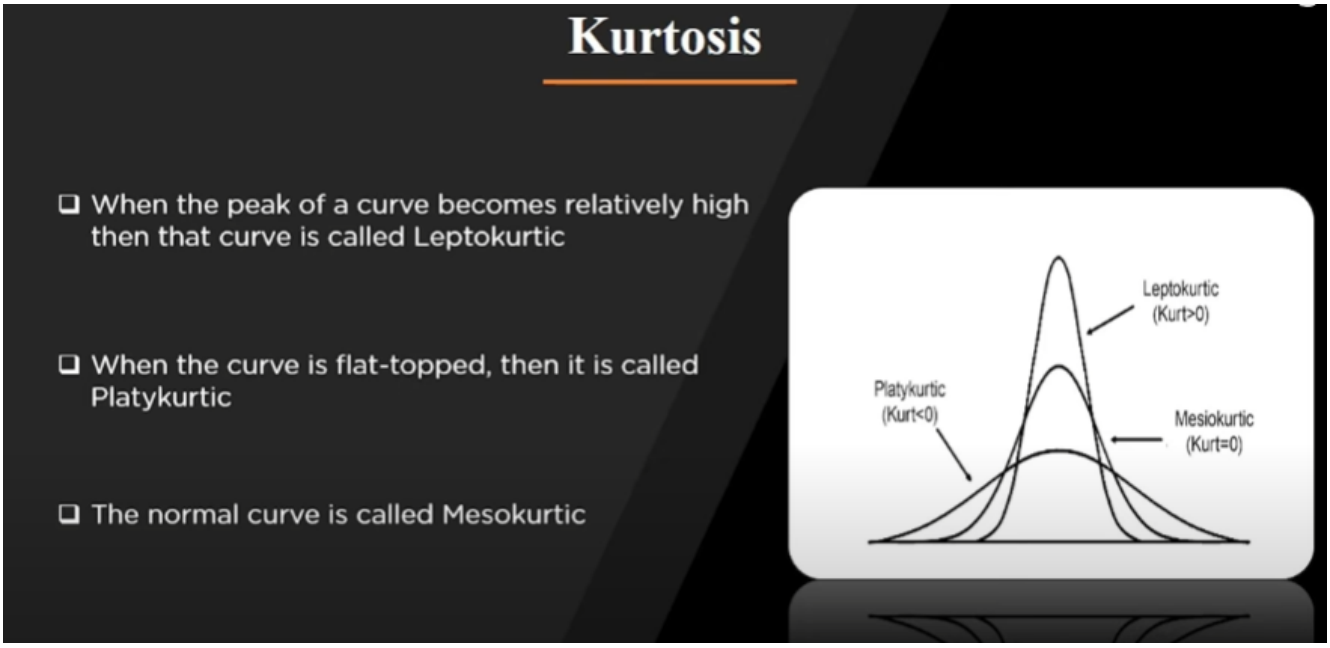
Distribution

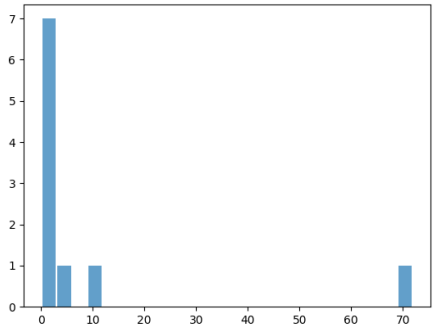




negatively skewed data mean >median > mean , Positively skewed data mean > median > mean

==Kurtosis means find the total degree of outliers ==





frequency table for histogram

In statistical theory, location and variability are referred to as the first and second moments of a distribution. The third and fourth moments are called **skewness and kurtosis**. Skewness refers to whether the data is skewed to larger or smaller values, and *kurtosis indicates the propensity of the data to have extreme values.* Gener- ally, metrics are not used to measure skewness and kurtosis; instead, these are discovered through visual displays such as Fig- ures 1-2 and 1-3.

Density Plots and Estimates

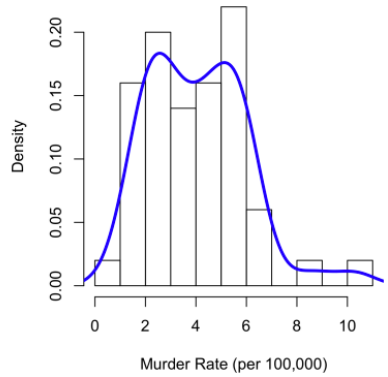
Related to the histogram is a density plot, which shows the distribution of data values as a continuous line. A density plot can be thought of as a smoothed histogram, although it is typically computed directly from the data through a kernel density esti- mate

```
ax = state['Murder.Rate'].plot.hist(density=True, xlim=[0,12], bins=range(1,12))
state['Murder.Rate'].plot.density(ax=ax)
ax.set_xlabel('Murder Rate (per 100,000)')

import pandas  as pd
import matplotlib.pyplot as plt
import seaborn as sns
#s = 'E:\\Medals.xlsx'
#import warnings
#warnings.simplefilter("ignore")
s = pd.read_excel('E:\\Medals.xlsx',engine='openpyxl')
#print(s)

#plt.boxplot(s['Gold'],)
plt.hist(s['Gold'])
plt.plot(s['Gold'])
```

```
plt.title('Desnity plot')
plt.show
```



Kernel Density Estimation (mathisonian.github.io)

Exploring Binary and Categorical Data

mode : most repeated value

```
#mode
import pandas as pd

a  = pd.read_excel('E:\\text.xlsx')

print(a['Gold'].mode())

#most repeated value
```

excepted value

learn about this

**this is like a prediction **

**formula ： EV=∑P(Xi)×XiEV = 0 . 05 300 + 0 . 15 50 + 0 . 80 0 = 22 . 5
**

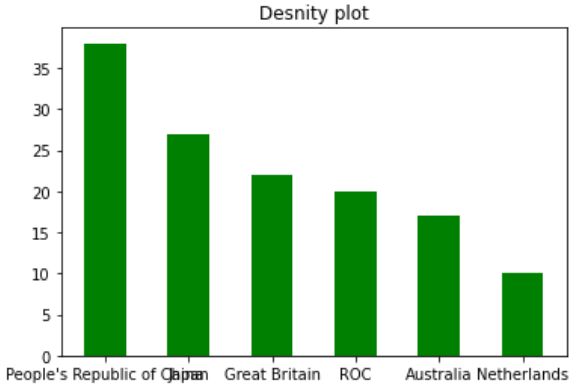
bar charts with label name and values

```
import pandas  as pd
import matplotlib.pyplot as plt
import seaborn as sns


#s = 'E:\\Medals.xlsx'
#import warnings
#warnings.simplefilter("ignore")

s = pd.read_excel('E:\\Medals.xlsx',engine='openpyxl')
#print(s)

#plt.boxplot(s['Gold'],)
#plt.hist(s['Gold'])
#plt.plot(s['Gold'])
plt.bar(s['Team/NOC'][1:7],s['Gold'][1:7], color  = 'green' , width= 0.5 )
plt.title('Desnity plot')
plt.show
```



bar chat are use to both numerical data as categorical data are binding the data showing the x-axis are names and y-axis are numbers

****Pie charts ****

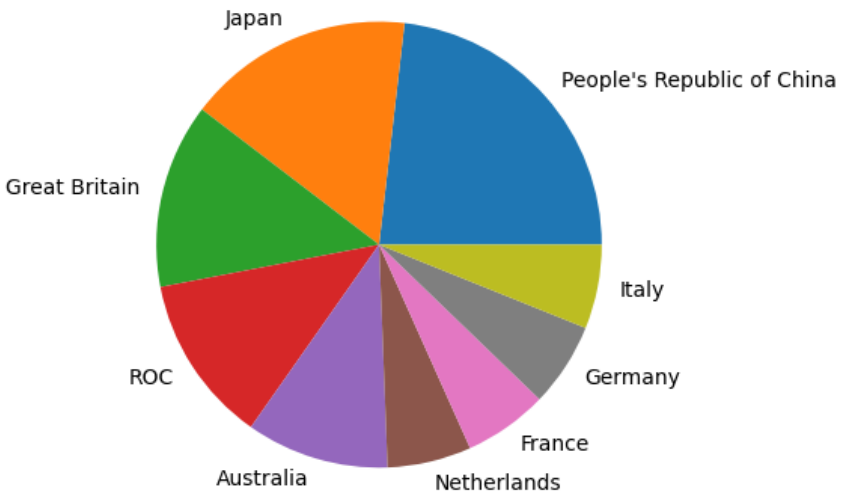
```
import pandas  as pd
import matplotlib.pyplot as plt
import seaborn as sns



#s = 'E:\\Medals.xlsx'
#import warnings
#warnings.simplefilter("ignore")

s = pd.read_excel('E:\\Medals.xlsx',engine='openpyxl')
#print(s)

#plt.boxplot(s['Gold'],)
#plt.hist(s['Gold'])
#plt.plot(s['Gold'])
ss = s['Team/NOC'][1:10]
plt.pie(s['Gold'][1:10] ,labels = ss )
plt.show
```



Expected Value :

we can expected the values for possible before to measure the probability

A special type of categorical data is data in which the categories represent or can be mapped to discrete values on the same scale. A marketer for a new cloud technology, for example, offers two levels of service, one priced at $300/month and another at $50/month. The marketer offers free webinars to generate leads, and the firm figures that 5% of the attendees will sign up for the $300 service, 15% will sign up for the $50 service, and 80% will not sign up for anything. This data can be summed up, for financial purposes, in a single "expected value," which is a form of weighted mean, in which the weights are probabilities.

The expected value is calculated as follows:

1. Multiply each outcome by its probability of occurrence.
2. Sum these values.

In the cloud service example, the expected value of a webinar attendee is thus $22.50 per month, calculated as follows:

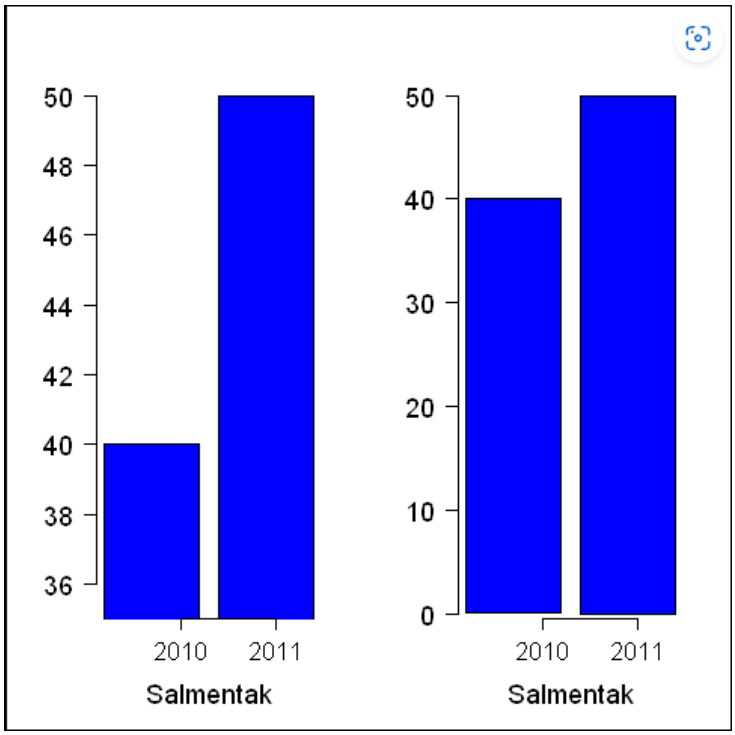
$$EV = (0.05)(300) + (0.15)(50) + (0.80)(0) = 22.5$$

The expected value is really a form of weighted mean: it adds the ideas of future expectations and probability weights, often based on subjective judgment. Expected value is a fundamental concept in business valuation and capital budgeting—for

Probability

means possibility of out comes is there rain today are tomorrow if there is possibility change 50 % chances to rain analysis the last few weeks

No statistics course is complete without a lesson on misleading graphs, which often involves bar charts and pie charts . misleading of leading of graphs are more dangerous below the graph you see that there is slight difference but we see left bar this is called truncate



Correlation :Exploratory data analysis in many modeling projects (whether in data science or in research) involves examining correlation among predictors, and between predictors and a target

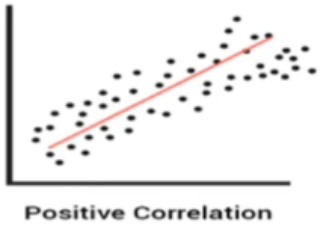
variable.

Correlation is a statistical technique used to determine the degree to which two variables are linear li related

for example : the price and demand of a product is a correlation is linear li related

correlation always between -1 to +1

the two type of correlations positive and negative correlation : positive correlation means both variables are lies between positive are negative relation

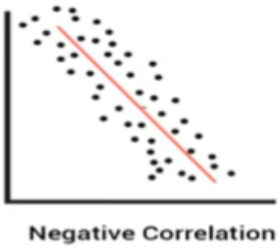


positive slop

for example : if x variable and y variable are going on same direction either positively or negatively

negative correlation both variables are opposite direction x going to positive and y goes to negative

example : if a car decreases speed the time taken to reach the destination increases

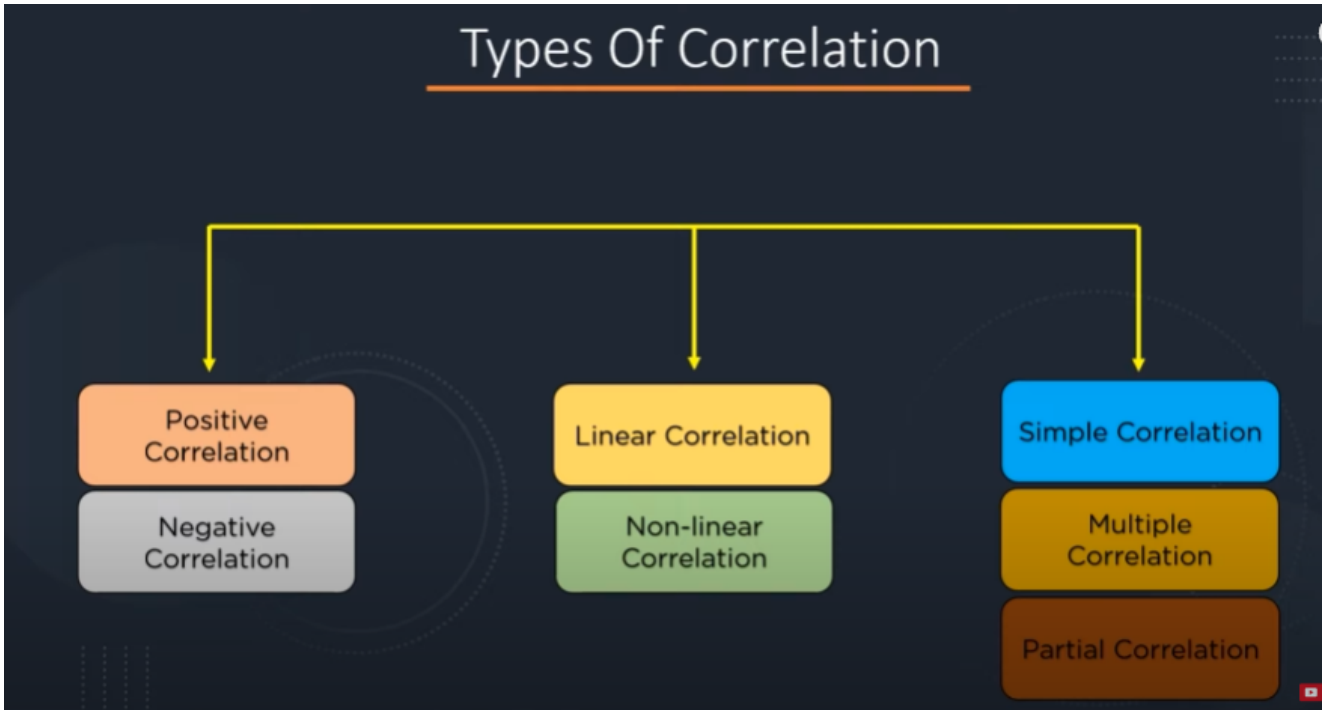


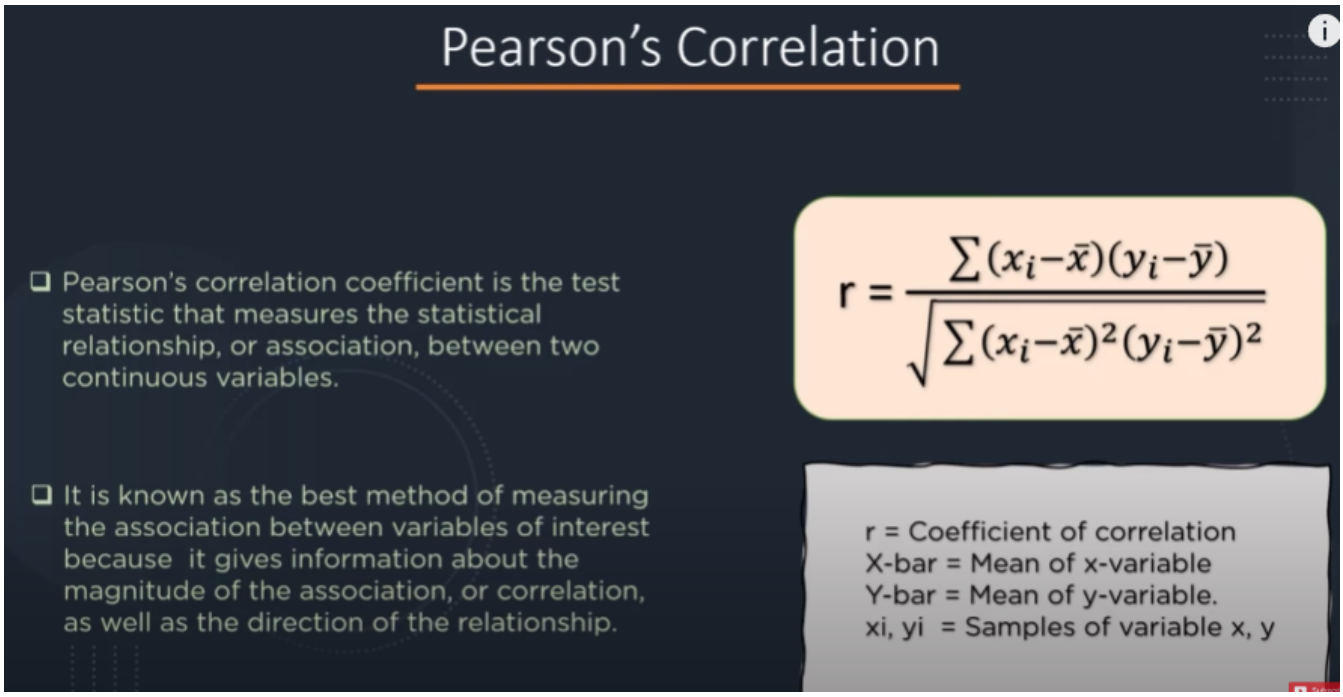
negative slop

the coefficient off correlation (r) is a measure of the strength of the linear relationship between the two variables

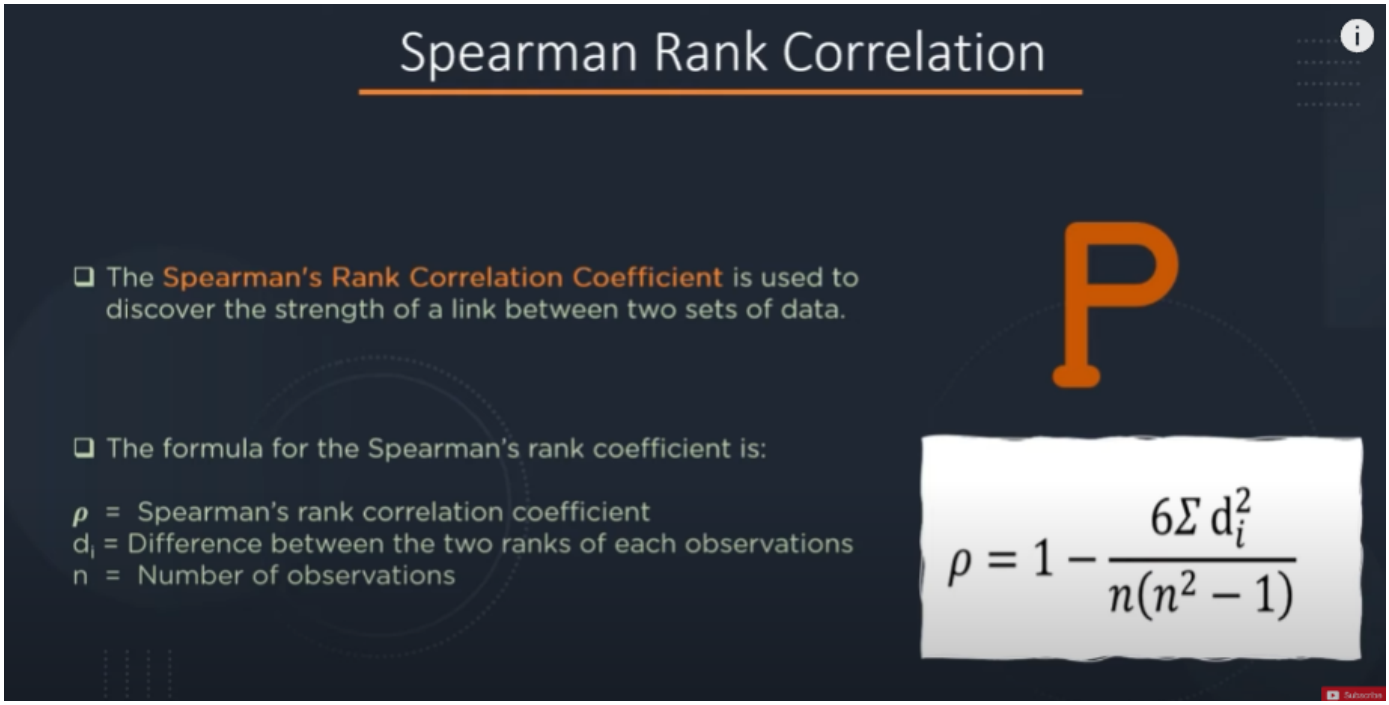
where r -1 strong correlation , r = 0 no correlation , r = +1 strong positive

Correlation does not tell us everything about the data. Mean and standard deviation continues to be important





$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y}$$



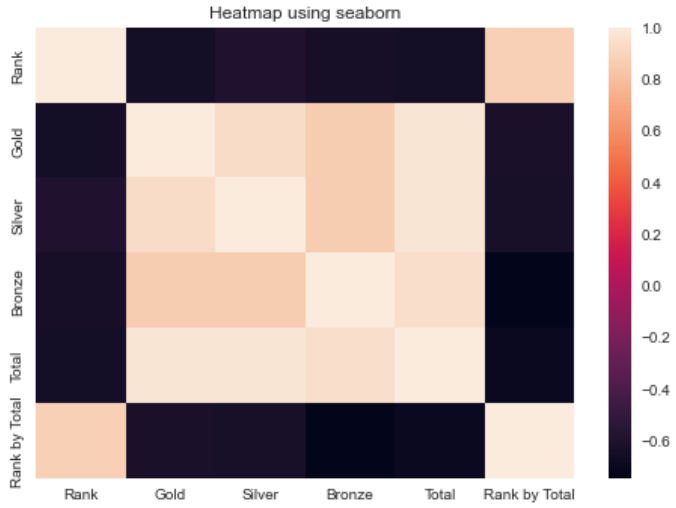
Correlation matrix

A table where the variables are shown on both rows and columns, and the cell values are the correlations between the variables.

Table 1-7. Correlation between telecommunication stock returns

	T	CTL	FTR	VZ	LVLT
T	1.000	0.475	0.328	0.678	0.279
CTL	0.475	1.000	0.420	0.417	0.287
FTR	0.328	0.420	1.000	0.287	0.260
VZ	0.678	0.417	0.287	1.000	0.242
LVLT	0.279	0.287	0.260	0.242	1.000

However, most support the visualization of correlation matrices using heatmaps.



means relation between (x and y) it prediction if x increase y increase are not (parties)

code : data.corr()

Other Correlation Estimates

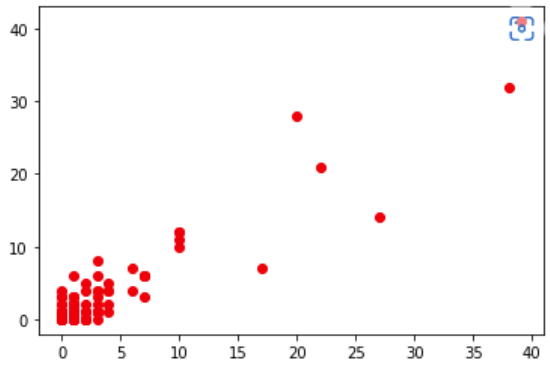
Statisticians long ago proposed other types of correlation coefficients, such as *Spearman's rho* or *Kendall's tau*. These are correlation coefficients based on the rank of the data. Since they work with ranks rather than values, these estimates are robust to outliers and can handle certain types of nonlinearities. However, data scientists can generally stick to Pearson's correlation coefficient, and its robust alternatives, for exploratory analysis. The appeal of rank-based estimates is mostly for smaller data sets and specific hypothesis tests.

Exploring Two or More Variables

one at a time (univariate analysis),important method that compares two variables (bivariate analysis), In this section we look at additional estimates and plots, and at more than two variables (multivariate analysis).

scatter between two variables

```
plt.set_xlabel('golds')
plt.set_ylabel('golds')
plt.scatter(z['Gold'],z['Silver'], color ='red')
plt.show()
```



Exploring Two or More Variables

Key Terms for Exploring Two or More Variables

Contingency table
A tally of counts between two or more categorical variables.

Hexagonal binning
A plot of two numeric variables with the records binned into hexagons.

Contour plot
A plot showing the density of two numeric variables like a topographical map.

Violin plot
Similar to a boxplot but showing the density estimate.

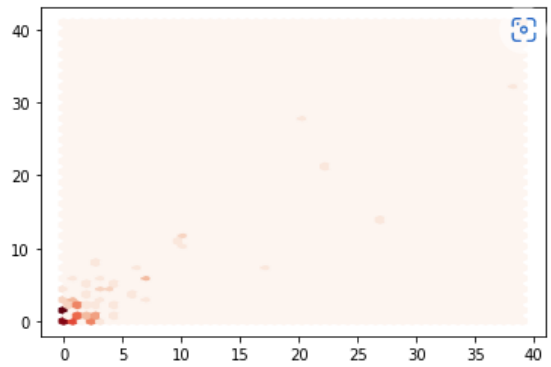
Hexagonal Binning and Contours

(Plotting Numeric Versus Numeric Data)

Scatterplots are fine when there is a relatively small number of data values. The plot of stock returns in Figure 1-7 involves only about 750 points. For data sets with hun- dreds of thousands or millions of records, a scatterplot will be too dense, so we need a different way to

visualize the relationship. To illustrate, consider the data set kc_tax, which contains the tax-assessed values for residential properties in King County, Washington. In order to focus on the main part of the data, we strip out very expensive and very small or large residences using the subset function

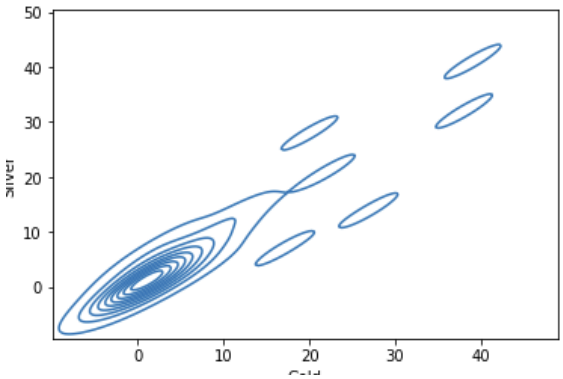
```
plt.hexbin(z['Gold'],z['Silver'],gridsize = 50, cmap ='Reds' )
plt.show()
```



Contour plot

uses contours overlaid onto a scatterplot to visualize the relationship between two numeric variables. The contours are essentially a topographical map to two variables; each contour band represents a specific density of points, increasing as one nears a “peak.” This plot shows a similar story as Figure 1-8: there is a secondary peak “north” of the main peak. This chart was also created using ggplot2 with the built-in geom_density2d function:

```
import seaborn as sn
sn.kdeplot(z['Gold'])
```



Two Categorical Variables

useful way to summarize two categorical variables is a contingency table—a table of counts by category. Table shows the contingency table between the grade of a personal loan and the outcome of that loan. This is taken from data provided by Lending Club, a leader in the peer-to-peer lending business. The grade goes from A (high) to G (low). The outcome is either fully paid, current, late, or charged off (the balance of the loan is not expected to be collected). This table shows the count and row percentages. High-grade loans have a very low late/charge-off percentage as compared with lower-grade loans.

Contingency tables can look only at counts, or they can also include column and total percentages. Pivot tables in Excel are perhaps the most common tool used to create contingency tables

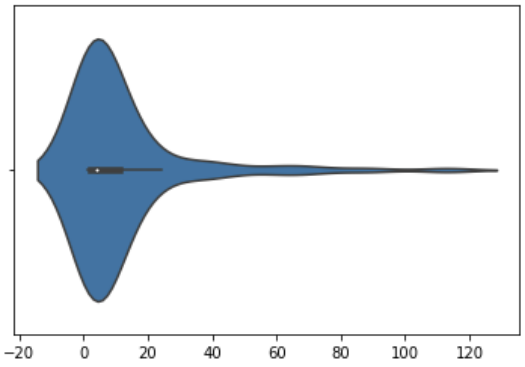
```
table = z.pivot_table(index = 'Team/NOC', columns = 'Gold', aggfunc = lambda x : len(x) ,margins =
True)
table
table.fillna(0)
```

	Bronze										...	Total									
Gold	0	1	2	3	4	6	7	10	17	20	...	6	7	10	17	20	22	27	38	39	All
Team/NOC																					
Argentina	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
Armenia	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
Australia	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1
Austria	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
Azerbaijan	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Ukraine	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
United States of America	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1
Uzbekistan	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
Venezuela	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
All	28.0	22.0	11.0	11.0	5.0	2.0	4.0	4.0	1.0	1.0	...	2.0	4.0	4.0	1.0	1.0	1.0	1.0	1.0	1.0	93

Categorical and Numeric Data

A violin plot, introduced by [Hintze-Nelson-1998], is an enhancement to the boxplot and plots the density estimate with the density on the y-axis. The density is mirrored and flipped over, and the resulting shape is filled in, creating an image resembling a violin. The advantage of a violin plot is that it can show nuances in the distribution that aren't perceptible in a boxplot. On the other hand, the boxplot more clearly shows the outliers in the data. In ggplot2, the function geom_violin can be used to create a violin plot as follows:

```
import seaborn as sn
sn.violinplot(x=z['Total'])
```

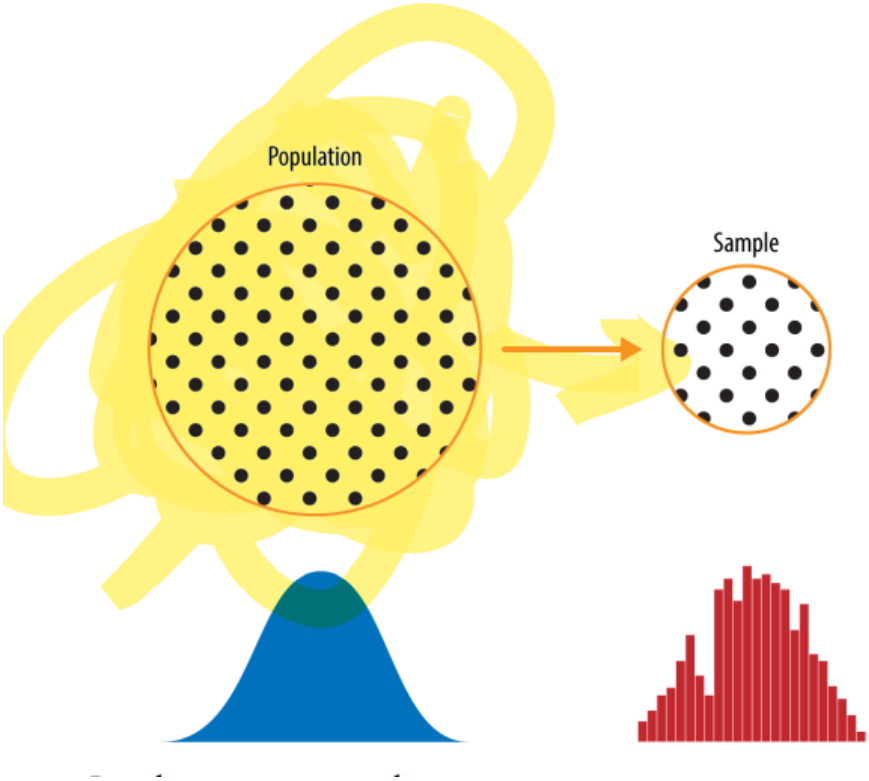


Visualizing Multiple Variables

**ggplot : **https://www.ling.upenn.edu/~joseff/avml2012/ **

Data and Sampling Distributions

The simplest example is flipping a coin: this follows a binomial distribution==. Any real-life binomial situation (buy or don't buy, fraud or no fraud, click or don't click)== can be modeled effectively by a coin (with modified proba-bility of landing heads, of course). In these cases, we can gain additional insight by using our understanding of the population.



Random Sampling and Sample Bias

Data quality often matters more than data quantity when making an estimate or a model based on a sample. Data quality in data science involves completeness, consis- tency of format, cleanliness, and accuracy of individual data points. Statistics adds the notion of representativeness.

****sample** : **a portion(small part) of overall population , usually taken when it either too difficult to obtain or process statistics on the entire dataset

A subset from a larger data set (take a small part)

****Population** : **The total of the datasets (all of the large datasets)

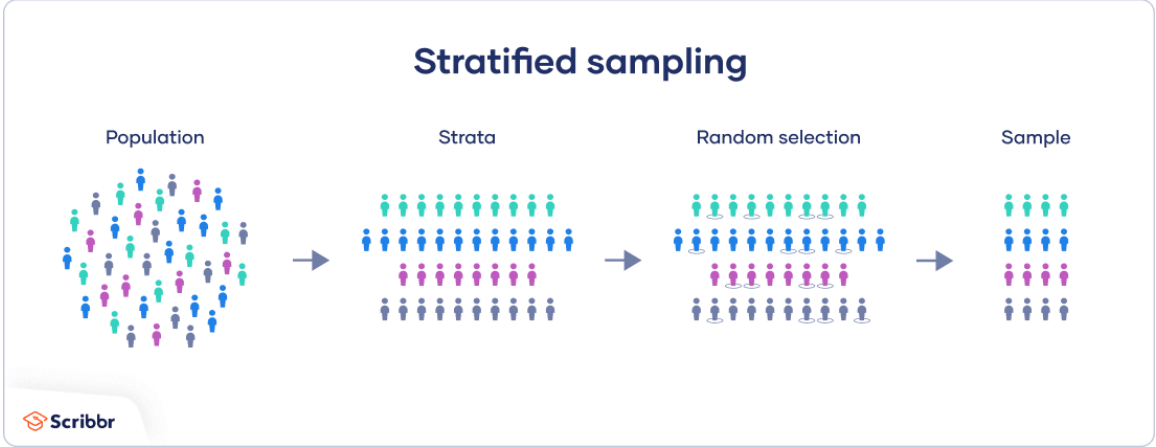
****N(n)** : **the population size (size of all datasets)

****Random sampling** : **create a samples that are taken random picking datasets (portion) of the population (all the datasets)

example : take the all population dataset into portion means parts and take into parts and pickup the one

****Stratified sampling** : **random samples that are taken after you split the population (or) the total data in section or strata (layers)

A stratified random sample, on the other hand, first divides the population into smaller groups, or strata, based on shared characteristics.



example : splitting the population by age group randomly pulling data from each layers

Stratum (pl., strata) : **groups the data spilt into

Simple random sample : **sample the results without breaking them up into layers

A simple random sample is used to represent the entire data population and randomly selects individuals from the population without any other consideration

Bias : Systematic error.

Sample bias : A sample that misrepresents(false the data) the population.

example : pulling one city in Andhra Pradesh when you want to political learn entire country

Self-Selection Sampling Bias

The reviews of restaurants, hotels, cafés, and so on that you read on social media sites like Yelp are prone to bias because the people submitting them are not randomly selected; rather, they themselves have taken the initiative to write. This leads to self-selection bias—the people motivated to write reviews may have had poor experiences, may have an association with the establishment, or may simply be a different type of person from those who do not write reviews. Note that while self-selection samples can be unreliable indicators of the true state of affairs, they may be more reliable in simply comparing one establishment to a similar one; the same self-selection bias might apply to each.

re not archive.zip

Bias

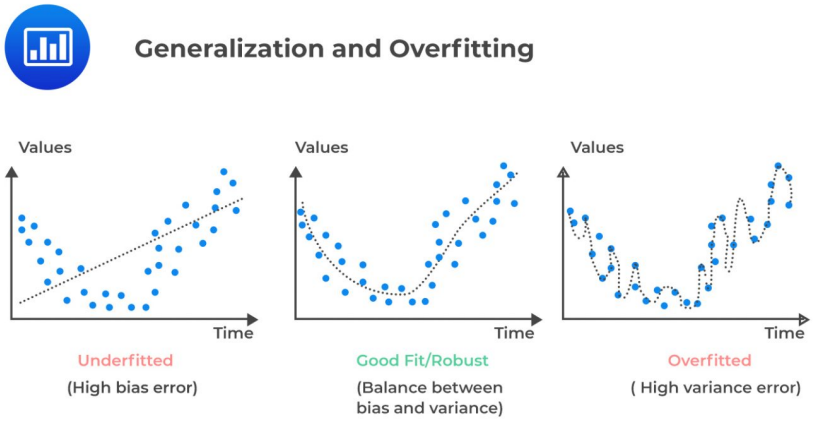
what was the bias

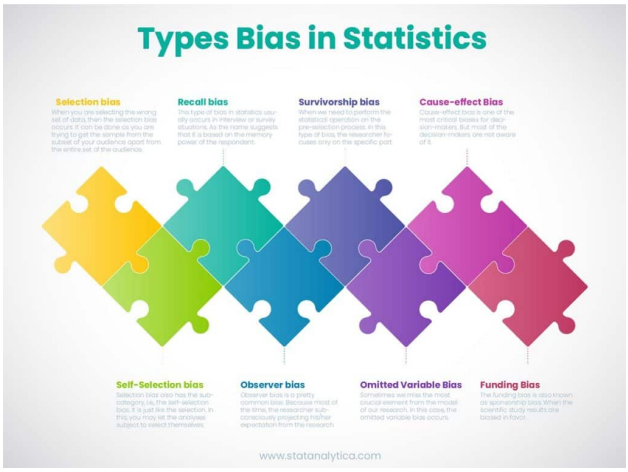
An **unbiased** process will produce error, but it is random and does not tend strongly in any direction

==*Statistical bias* is a systematic tendency which causes differences between results and facts. errors underestimation are overestimation ==

Bias comes in different forms, and may be observable or invisible. When a result does suggest bias (e.g., by reference to a benchmark or actual values), it is often an indica- tor that a statistical or machine learning model has been misspecified, or an impor- tant variable left out.

(bias error means see that how linear predict module are connected to datasets example: fit , outfit , overfit)





Size Versus Quality: When Does Size Matter : In the era of big data, it is sometimes surprising that smaller is better. Time and effort spent on random sampling not only reduces bias but also allows greater attention to data exploration and data quality. For example, missing data and outliers may contain useful information. It might be prohibitively expensive to track down missing values or evaluate outliers in millions of records, but doing so in a sample of several thou- sand records may be feasible.

Sample Mean Versus Population Mean

$$\bar{x} \quad sample \quad mean$$
$$\mu \quad total \quad population$$

Key Ideas

- Even in the era of big data, random sampling remains an important arrow in the data scientist's quiver.
- Bias occurs when measurements or observations are systematically in error because they are not representative of the full population.
- Data quality is often more important than data quantity, and random sampling can reduce bias and facilitate quality improvement that would otherwise be prohibitively expensive.

hypothesis : A hypothesis is **an idea or proposition that can be tested by observations or experiments, about the natural world**

example : **"Students who eat breakfast will perform better on a math exam than students who do not eat breakfast." "Students who experience test anxiety prior to an English exam will get higher scores than students who do not experience test anxiety.

example : experiment output values testing is called hypothesis when we same repeated experiment testing output values we will take one hypothesis values (take one experiment value)

null hypothesis means no difference in experiment values

mean is low the data distribution is very high if mean is high data distribution is low

Selection Bias

basically occurs on non randomized sampling data for target population it effect the result (when you select blind random data it will prefect result)

- Bias that resulting from how observation data are selected

Data snooping

searching through data in order to find something interested on within it . to prevent its important that you split the apart a section of the data this was the test set (when machine learning take a part of the data test the module and apply the all of the data you need to check that testing model and apply model are same sometimes it will get different outputs)

Vast search effect

If you repeatedly run different models and ask different questions with a large data set, you are bound to find something interesting. But is the result you found truly something interesting, or is it the chance outlier?

target shuffling

REGRESSION TO MEAN

reference to the phenomenon that extremes observation tends to be followed by ones closer to the mean . this is important to remember as without remembering this , it can be easy that to assume that an extreme value obtained early on is representative , when it way just an outliers

example : success = talent + luck first day player play very well but we predict the second day it was very difficult second day the luck not favor the day second play like that not playing very back luck work both values are center to the mean

Regression to the mean is a consequence of a particular form of selection bias

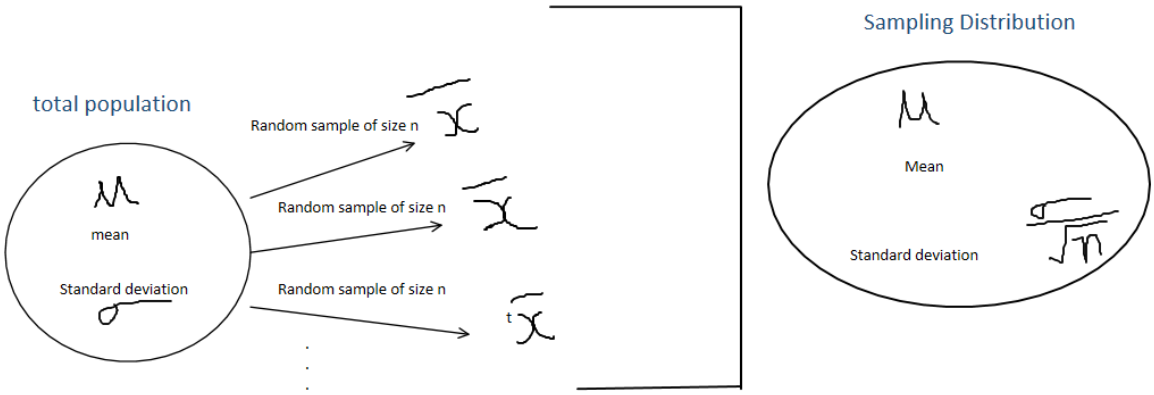
Regression to the mean, meaning to "go back," is distinct from the statistical modeling method of linear regression, in which a linear relationship is estimated between predictor variables and an outcome variable.

https://www.priceactionlab.com/Blog/2012/06/fooled-by-randomness-through-selection-bias/

when it happens

Regression to the mean usually happens because of sampling error. A good sampling technique is to randomly sample from the population. If you don't (i.e. if you asymmetrically sample), then your results may be abnormally high or low for the average and therefore would regress back to the mean. Regression to the mean can also happen because you take a very small, unrepresentative sample (say, the highest 1 percent of the population or the lowest ten percent).

Sampling Distribution of a Statistic



we take random sample of many samples to the population all of that is know as sampling Distribution

Sample statistic

this is normal sample taken by the population

Data distribution (frequency distribution)

means data visitation graphs normal distribution ,binomial distribution

Central limit theorem

if you have a large amount of data take the random sample up to x_n the mean is approximately equal to total number of population

$$\bar{x} = (\mu, \sigma^2/n)$$

here n condition we take (like that n >= 30)

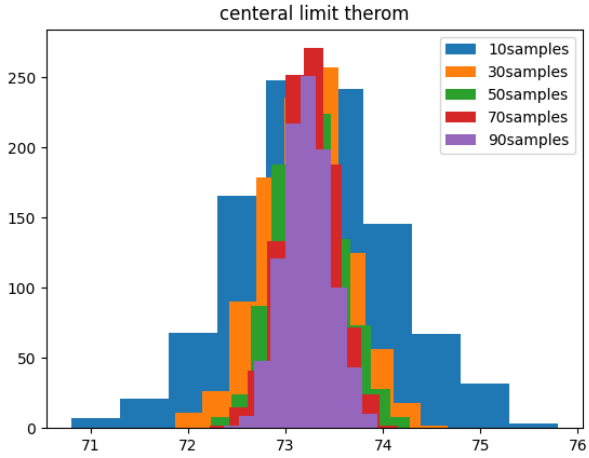
The phenomenon we've just described is termed the central limit theorem. It says that the means drawn from multiple samples will resemble the familiar bell-shaped nor- mal curve even if the source population is not normally distributed, provided that the sample size is large enough and the departure of the data from normality is not too great. The central limit theorem allows normal-approximation formulas like the t-distribution to be used in calculating sampling distributions for inference—that is, confidence intervals and hypothesis tests.

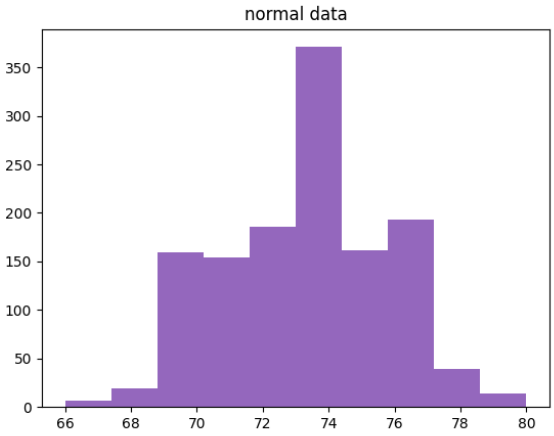
central limit theorem is take the sample arrange that the data into center like a skew plot are bell shaped normal curve we taking 5 ,10 ,20 and 30 samples

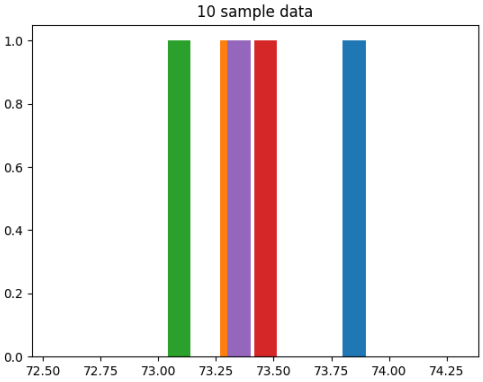
```
import numpy as np , matplotlib.pyplot as plt  ,pandas as pd

s = pd.read_csv('players.csv')
# here take the removing  the some  data like 6-1 and mul by 12 get output like filtering data
s['height'] = s['height'].apply(lambda x: int(x.split('-')[0]) * 12 + int(x.split('-')[1]) if
len(x.split('-'))  == 2 else int(x))
# here sample taken 10 10+20 =30 like thata
sample_amount =  range(10,100,20)

for i in sample_amount :
    sample  = []
    for x in range(1000):
        sample.append(s['height'].sample(i).mean())
    plt.hist(sample)
plt.legend([str(z) + 'samples' for z in sample_amount])
plt.title('centeral limit therom')
plt.show()
```







you take much more of the actual mean going to center and you perfect mean and normal distribution

sample mean is approximate equal to the population mean

Population standard deviation

Formula	Explanation
$\sigma = \sqrt{\frac{\sum (X - \mu)^2}{N}}$	• σ = population standard deviation • $\sum$ = sum of... • X = each value • μ = population mean • N = number of values in the population

sample standard deviation

**standard deviation means data point to point distance **

Formula	Explanation
$s = \sqrt{\frac{\sum (X - \bar{x})^2}{n - 1}}$	• s = sample standard deviation • $\sum$ = sum of... • X = each value • $\bar{x}$ = sample mean • n = number of values in the sample

Standard error

$$standard \quad deviation \quad error = SE \quad (\bar{x}) = \frac{s}{\sqrt{n}}$$

$$s = \ standard \ \ deviation$$
$$n = \ number \ \ of \ \ values \ \ in \ \ the \ \ sample$$

were denominator is big than standard error is low that means you take large amount of sample . here n = total number of samples the error will be low

The standard error can be estimated using a statistic based on the standard deviation s of the sample values, and the sample size n

we estimation the standard error

1. Collect a number of brand-new samples from the population.
2. For each new sample, calculate the statistic (e.g., mean).
3. Calculate the standard deviation of the statistics computed in step 2; use this as your estimate of standard error.

In modern statistics, the bootstrap has become the standard way to estimate standard error. It can be used for virtually any statistic and does not rely on the central limit theorem or other distributional assumptions

Do not confuse standard deviation (which measures the variability of individual data points) with standard error (which measures the variability of a sample metric).

Standard Deviation Versus Standard Error

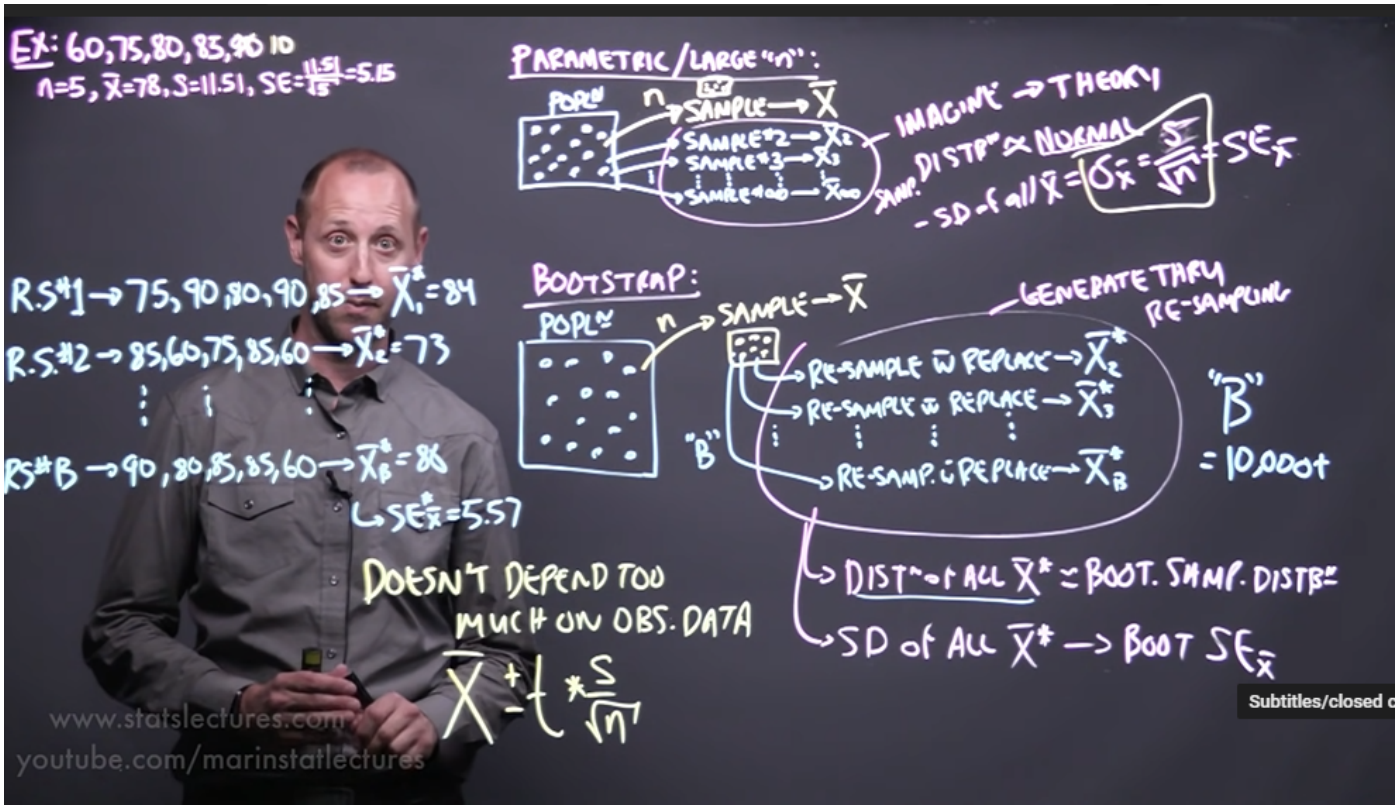
Do not confuse standard deviation (which measures the variability of individual data points) with standard error (which measures the variability of a sample metric).

how we far form the excite value

```
# to find the standard error in pandas
# this is unstructure data covert into structure data using series and get the standard error
pd.Series(dataset_name).sem()
# this code is normal formula
s['height'].sem()
```

https://onlinestatbook.com/stat_sim/sampling_dist/

THE Bootstrap

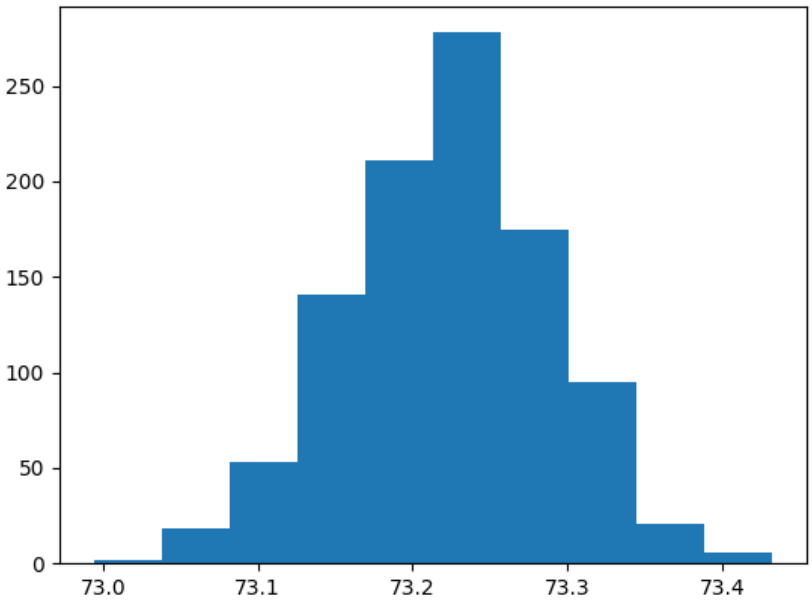


expiation : https://www.youtube.com/watch?v=O_Fj4q8lgmc

bootstrap is : you have large set of the population take a sample data means x-bar and in that sample data you take it into resample x-bar* (bootstrap sample)

- example : you take a resample data and put back into data in resample and you take other data and put back into resample and that resample standard error is approximately equal sample standard error , mean also
- resample mean you can also plot is called bootstrap mean
- you have thousand of resample data the bootstrap standard error is approximately equal to sample standard error
- you bootstrap approached values are nearly identical to large sample theory 27/06/2022 12:49
- Bootstrapping actually creates bigger samples out of your existing data and can even do this in multiple repetitions to allow significance testing. Bootstrapping respects the distribution that your existing data have, so you don't need to make any assumptions regarding the distribution of your data. This is important because in most statistical approaches there are such assumptions, which sometimes we don't even realise or check before we apply them
- https://stats.stackexchange.com/questions/19675/what-normality-assumptions-are-required-for-an-unpaired-t-test-and-when-are-the
- https://www.investopedia.com/ask/answers/073115/what-assumptions-are-made-when-conducting-ttest.asp
- **Bootstrap sample : A sample taken with replacement from an observed data set**
- An exhaustive permutation test
  A bootstrap permutation test
- **Resampling : The process of taking repeated samples from observed data; includes both boot- strap and permutation (shuffling) procedures.(using the resample data using taking data reusing the data) there is also possibility to present in outliers**
- ```
  from sklearn.utils import resample
                                                  s = pd.read_csv('players.csv')
                                                  s['height'] = s['height'].apply(lambda x:
  int(x.split('-')[0]) * 12 + int(x.split('-')[1]) if len(x.split('-'))  == 2 else int(x))
                                                  data = s['height]
                                                  results = []
                                                  #taking the thousand samples and you will take
  the  100 and 50 as you like that
                                                  for nrepeat in range(1000):
  ```
  ```

```
                                                    sample = resample(data)
                                                    results.append(sample.mean())
                                                    results = pd.Series(results)
                                                    print('Bootstrap Statistics:')
                                                    print(f'original: {data.mean()}')
                                                    #bias
                                                    print(f'bias: {results.mean() -
data.mean()}')
                                                    print(f'std : {results.std()}')
```



this is the hist of all 1000 resample mean

every thing are in normal distribution like that

Resampling Versus Bootstrapping

Sometimes the term *resampling* is used synonymously with the term *bootstrapping*, as just outlined. More often, the term *resampling* also includes permutation procedures (see "Permutation Test" on page 97), where multiple samples are combined and the sampling may be done without replacement. In any case, the term *bootstrap* always implies sampling with replacement from an observed data set.

Key Ideas

- The bootstrap (sampling with replacement from a data set) is a powerful tool for assessing the variability of a sample statistic.
- The bootstrap can be applied in similar fashion in a wide variety of circumstances, without extensive study of mathematical approximations to sampling distributions.
- It also allows us to estimate sampling distributions for statistics where no mathematical approximation has been developed.
- When applied to predictive models, aggregating multiple bootstrap sample predictions (bagging) outperforms the use of a single model.

Confidence Interval (total population data lie between learn about this)

The total population probability values lies between - and + (values) when the normal distribution we know the standard devotion

we taken the sample data and find the total number of the population data lies between these values . lots of time we need to find 95% and 99% of data

$$\bar{x} + z \cdot \frac{s}{\sqrt{n}}$$
$$\bar{x} - z \cdot \frac{s}{\sqrt{n}}$$
$$here \quad \bar{x} \quad we \quad taken \quad the \quad sample \quad mean \quad of the \quad data$$
$$z \quad values \quad present \quad in \quad data$$
$$s \quad standarddeviation \quad \sigma \quad of \quad total \quad population$$
$$\sqrt{n} \quad total(number) \quad of \quad sample \quad you \quad taken$$

```
import numpy as np , matplotlib.pyplot as plt  ,pandas as pd
from sklearn.utils import resample
import scipy.stats as sp
s = pd.read_csv('players.csv')
s['height'] = s['height'].apply(lambda x: int(x.split('-')[0]) * 12 + int(x.split('-')[1]) if
len(x.split('-'))  == 2 else int(x))

st = s['height']
#define sample data
np.random.seed(0)
data = st.sample( frac= 0.60)
k = sp.norm.interval(alpha = 0.95 , loc = np.mean(data), scale =sp.sem(data) )

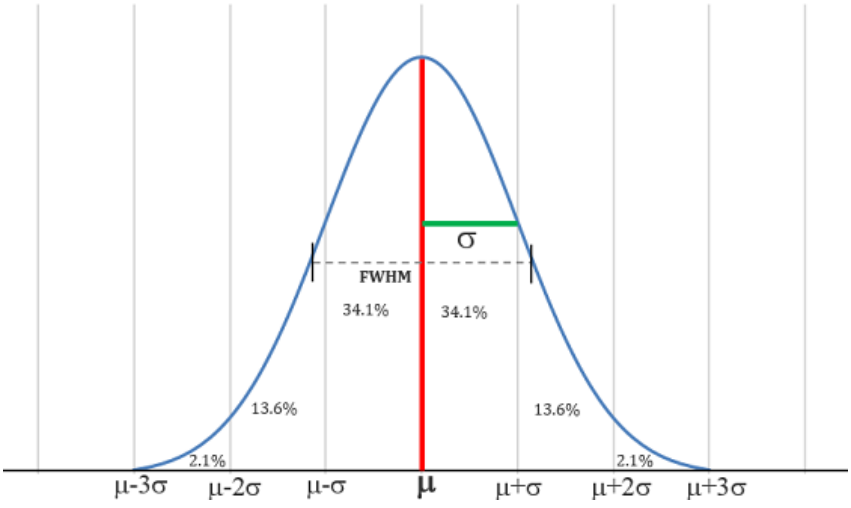
print(k)

output values : (73.04488079566644, 73.38222917875811) the total population lies between in this
```

For a data scientist, a confidence interval is a tool that can be used to get an idea of how variable a sample result might be. Data scientists would use this information not to publish a scholarly paper or submit a result to a regulatory agency (as a researcher might) but most likely to communicate the potential error in an estimate, and perhaps to learn whether a larger sample is needed.

Normal Distribution

**The bell-shaped normal distribution is iconic in traditional statistics. this also a Gaussian distribution **



In a normal distribution or Gaussian distribution , 68% of the data lies within one standard deviation of the mean, and 95% lies within two standard deviations.

It is a common misconception that the normal distribution is called that because most data follows a normal distribution—that is, it is the normal thing. Most of the variables used in a typical data science project—in fact, most raw data as a whole—are not normally distributed: see "Long-Tailed Distributions"

Error

The difference between a data point and a predicted or average value.(you predict any data value tell that result is not good prediction is called predict error)

Standardize

Subtract the mean and divide by the standard deviation. also called standard scores or normal deviates . It tells us how far from the mean we are in terms of standard deviations. (mean to standard deviation distance)

$$z = \frac{X - \mu}{\sigma}$$

z-score

The result of standardizing an individual data point. the standardize and z -score are same and formula : using the z score table we find the -1.5 or other values

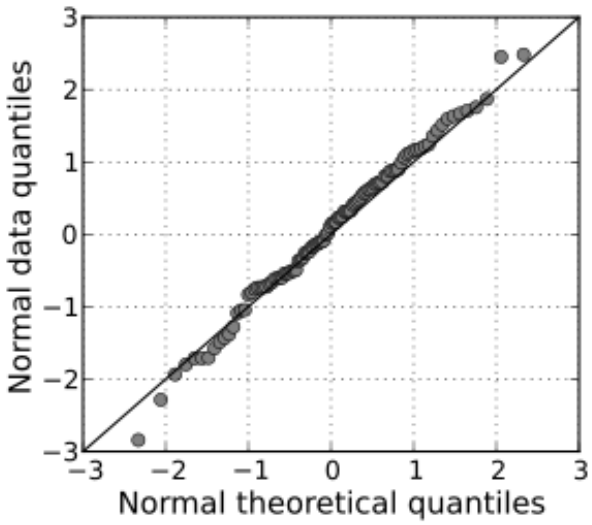
Standard normal

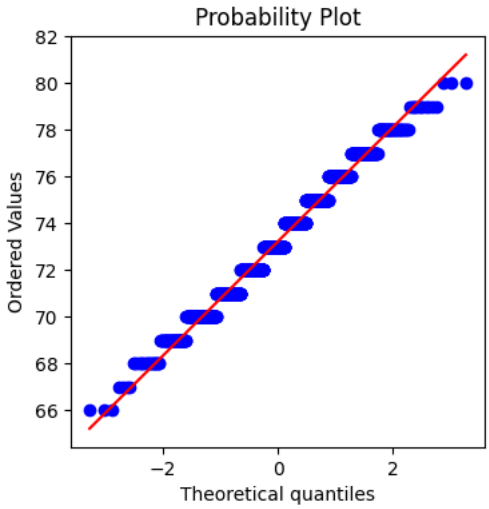
A normal distribution with mean = 0 and standard deviation = 1.

https://www.youtube.com/watch?v=4Fta6KQ1QHQ

qq-plot

A plot to visualize how close a sample distribution is to a specified distribution, e.g., the normal distribution.



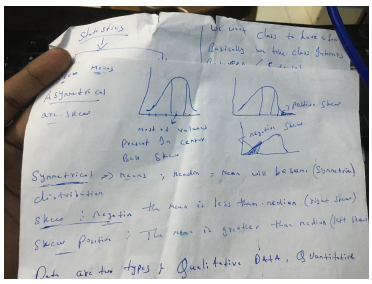



```
import numpy as np , matplotlib.pyplot as plt  ,pandas as pd
from sklearn.utils import resample
import scipy.stats as sp
s = pd.read_csv('players.csv')
s['height'] = s['height'].apply(lambda x: int(x.split('-')[0]) * 12 + int(x.split('-')[1]) if
len(x.split('-'))  == 2 else int(x))

st = s['height']
fix,ax = plt.subplots(figsize = (4,4))
sample = st
sp.probplot(sample, plot =ax )
plt.show()
```

It is a common misconception that the normal distribution is called that because most data follows a normal distribution—that is, it is the normal thing. Most of the variables used in a typical data science project—in fact, most raw data as a whole—are not normally distributed: see “Long-Tailed Distributions” on page 73. The utility of the normal distribution derives from the fact that many statistics are normally distributed in their sampling distribution. Even so, assumptions of normality are generally a last resort, used when empirical probability distributions, or bootstrap distributions, are not available.

Long-Tailed Distribution



Tail : The long narrow portion of a frequency distribution, where relatively extreme values occur at low frequency.

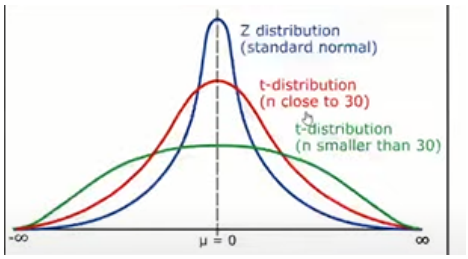
Skew : Where one tail of a distribution is longer than the other.

While the normal distribution is often appropriate and useful with respect to the dis- tribution of errors and sample statistics, it typically does not characterize the distribu- tion of raw data. Sometimes, the distribution is highly skewed (asymmetric), such as with income data; or the distribution can be discrete, as with binomial data. Both symmetric and asymmetric distributions may have long tails. The tails of a distribu- tion correspond to the extreme values (small and large). Long tails, and guarding against them, are widely recognized in practical work. Nassim Taleb has proposed the black swan theory, which predicts that anomalous events, such as a stock market crash, are much more likely to occur than would be predicted by the normal distribution. A good example to illustrate the long-tailed nature of data is stock return

Most data is not normally distributed. • Assuming a normal distribution can lead to underestimation of extreme events (“black swans”).

Student’s t-Distribution

The t-distribution is a normally shaped distribution, except that it is a bit thicker and longer on the tails. It is used extensively in depicting distributions of sample statistics. Distributions of sample means are typically shaped like a t-distribution, and there is a family of t-distributions that differ depending on how large the sample is. The larger the sample, the more normally shaped the t-distribution becomes.



in t - distribution have degree of freedom

$$\bar{x} \pm t_{n-1}\alpha \cdot \frac{s}{\sqrt{n}}$$

• The t-distribution is actually a family of distributions resembling the normal dis- tribution but with thicker tails. • The t-distribution is widely used as a reference basis for the distribution of sample means, differences between two sample means, regression parameters, and more.

Binomial Distribution

basically tells us click or not click , like head and toss example of the binomial distribution

https://www.youtube.com/watch?v=K0JLSf2D1rg

```
import scipy.stats as sp
sp.binom.pmf(2, n=5, p=0.1)
sp.binom.cdf(2, n=5, p=0.1)
```

- where the probability of success for each trial is 0.1
- 2 number of the success we do like
- n number of times to do test
- The Probability Mass Function (PMF) tells us the probability that an event will happen a certain number of times given a certain number of samples and probability for success while a Cumulative Density Function or CDF tells us the probability that an even will happen a certain number or fewer times.

The variance is n × p 1 − p . With a large enough number of trials (particularly when p is close to 0.50), the binomial distribution is virtually indistinguishable from the normal distribution. In fact, calculating binomial probabilities with large sample sizes is computationally demanding, and most statistical procedures use the normal distribution, with mean and variance, as an approximation.

Chi-Square Distribution

An important idea in statistics is departure from expectation, especially with respect to category counts. Expectation is defined loosely as "nothing unusual or of note in the data" (e.g., no correlation between variables or predictable patterns)

It is useful for determining whether multiple treatments (an "A/B/C… test") differ from one another in their effects

https://www.youtube.com/watch?v=YrhlQB3mQFI

https://www.youtube.com/watch?v=qYOMO83Z1WU

https://www.statology.org/yates-continuity-correction/

https://www.youtube.com/watch?v=hkDfi5j-6IM

The chi-square distribution is the distribution of this statistic under repeated resampled draws from the null model—see "Chi-Square Test for a detailed algorithm, and the chi-square formula for a data table. A low chi-square value for a set of counts indicates that they closely follow the expected distribution. A high chi-square indicates that they differ markedly from what is expected. There are a variety of chi-square distributions associated with different degrees of freedom (e.g., number of observations—see "Degrees of Freedom" .

most important topic we learn

Degree of freedom

means (n-1) here n = total number of the samples and subtract one

you have one possible of out comes means you one degree of freedom

two possible of out comes you two degree of freedom

Key Ideas

- The chi-square distribution is typically concerned with counts of subjects or items falling into categories.
- The chi-square statistic measures the extent of departure from what you would expect in a null model.

F-Distribution

A common procedure in scientific experimentation is to test multiple treatments across groups—say, different fertilizers on different blocks of a field. This is similar to the A/B/C test referred to in the chi-square distribution (see "Chi-Square Distribution" on page 80), except we are dealing with measured continuous values rather than counts. In this case we are interested in the extent to which differences among group means are greater than we might expect under normal random variation. The F-statistic measures this and is the ratio of the variability among the group means to the variability within each group (also called residual variability). This comparison is termed an *analysis of variance* (see "ANOVA" on page 118). The distribution of the F-statistic is the frequency distribution of all the values that would be produced by randomly permuting data in which all the group means are equal (i.e., a null model). There are a variety of F-distributions associated with different degrees of freedom (e.g., numbers of groups—see "Degrees of Freedom" on page 116). The calculation of F is illustrated in the section on ANOVA. The F-statistic is also used in linear regression to compare the variation accounted for by the regression model to the overall variation in the data. F-statistics are produced automatically by *R* and *Python* as part of regression and ANOVA routines.

Key Ideas

- The F-distribution is used with experiments and linear models involving measured data.
- The F-statistic compares variation due to factors of interest to overall variation.

https://www.youtube.com/watch?v=r1ueoHA_KCQ

Poisson and Related Distributions

Poisson Distributions

From prior aggregate data (for example, number of flu infections per year), we can estimate the average number of events per unit of time or space (e.g., infections per day, or per census unit). We might also want to know how different this might be from one unit of time/space to another. The Poisson distribution tells us the distribution of events per unit of time or space when we sample many such units. It is useful when addressing queuing questions such as "How much capacity do we need to be 95% sure of fully processing the internet traffic that arrives on a server in any five-second period?"

The key parameter in a Poisson distribution is λ, or lambda. This is the mean number of events that occurs in a specified interval of time or space. The variance for a Poisson distribution is also λ.

A common technique is to generate random numbers from a Poisson distribution as part of a queuing simulation. The `rpois` function in *R* does this, taking only two arguments—the quantity of random numbers sought, and lambda:

```
rpois(100, lambda=2)
```

The corresponding `scipy` function is `stats.poisson.rvs`:

```
stats.poisson.rvs(2, size=100)
```

This code will generate 100 random numbers from a Poisson distribution with $\lambda = 2$. For example, if incoming customer service calls average two per minute, this code will simulate 100 minutes, returning the number of calls in each of those 100 minutes.

we can estimate the average number of events per unit of time or space (e.g., infections per day, or per census unit). We might also want to know how different this might be from one unit of time/space to another.

The key parameter in a Poisson distribution is λ, or lambda. This is the mean number of events that occurs in a specified interval of time or space. The variance for a Pois- son distribution is also λ

stats.poisson.rvs(2, size=100)

This code will generate 100 random numbers from a Poisson distribution with λ = 2. For example, if incoming customer service calls average two per minute, this code will simulate 100 minutes, returning the number of calls in each of those 100 minutes.

Exponential Distribution

Using the same parameter λ that we used in the Poisson distribution, we can also model the distribution of the time between events: time between visits to a website or between cars arriving at a toll plaza. It is also used in engineering to model time to failure, and in process management to model, for example, the time required per service call. The *R* code to generate random numbers from an exponential distribution takes two arguments: `n` (the quantity of numbers to be generated) and `rate` (the number of events per time period). For example:

```
rexp(n=100, rate=0.2)
```

In the function `stats.expon.rvs`, the order of the arguments is reversed:

```
stats.expon.rvs(0.2, size=100)
```

This code would generate 100 random numbers from an exponential distribution where the mean number of events per time period is 0.2. So you could use it to simulate 100 intervals, in minutes, between service calls, where the average rate of incoming calls is 0.2 per minute.

A key assumption in any simulation study for either the Poisson or exponential distribution is that the rate, λ, remains constant over the period being considered. This is rarely reasonable in a global sense; for example, traffic on roads or data networks varies by time of day and day of week. However, the time periods, or areas of space, can usually be divided into segments that are sufficiently homogeneous so that analysis or simulation within those periods is valid.

Using the same parameter λ that we used in the Poisson distribution, we can also model the distribution of the time between events: time between visits to a website or between cars arriving at a toll plaza.

```
stats.expon.rvs(0.2, size=100)
```

in this instance we re simulating the number of minutes between phone calls assuming that we receive 0.2 a call per minute

These are similar in use case to Poisson distributions except they focus on outputting the time between each occurrence given a mean number of instances per time period and a number of time periods to simulate:

https://www.youtube.com/watch?v=8M_VRCc9rMY

Estimating the Failure Rate

In many applications, the event rate, λ, is known or can be estimated from prior data. However, for rare events, this is not necessarily so. Aircraft engine failure, for example, is sufficiently rare (thankfully) that, for a given engine type, there may be little data on which to base an estimate of time between failures. With no data at all, there is little basis on which to estimate an event rate. However, you can make some guesses: if no events have been seen after 20 hours, you can be pretty sure that the rate is not 1 per hour. Via simulation, or direct calculation of probabilities, you can assess different hypothetical event rates and estimate threshold values below which the rate is very unlikely to fall. If there is some data but not enough to provide a precise, reliable estimate of the rate, a goodness-of-fit test (see "Chi-Square Test" on page 124) can be applied to various rates to determine how well they fit the observed data.

Weibull Distribution :

how many times possible occurs failure when flight engine failure this time is called Weibull distribution

```
stats.weibull_min.rvs(1.5, scale=5000, size=100)
```

https://www.youtube.com/watch?v=8M_VRCc9rMY

• For events that occur at a constant rate, the number of events per unit of time or space can be modeled as a Poisson distribution.

• You can also model the time or distance between one event and the next as an exponential distribution.

• A changing event rate over time (e.g., an increasing probability of device failure) can be modeled with the Weibull distribution.

chapter 3

A/B Testing (a/b/c/d testing as you wish but you it increasing the complexity)

Key Terms for A/B Testing

Treatment
Something (drug, price, web headline) to which a subject is exposed.

Treatment group
A group of subjects exposed to a specific treatment.

Control group
A group of subjects exposed to no (or standard) treatment.

Randomization
The process of randomly assigning subjects to treatments.

Subjects
The items (web visitors, patients, etc.) that are exposed to treatments.

Test statistic
The metric used to measure the effect of the treatment.

example : a/b testing is a like when you take one tablet two same features models(slightly different versions) A and B , A is give some 1000 peoples and B is given authored thousand peoples . both of the b like 700 people and a like 300 people (we stop the production of a or discontinue a and we use the a)

most of time we use a/b testing in data science products recommendation using machine learning models

and we created two websites layout we want see people which layout likes we take add button to survey that data add to dango and sql we see that data and change the layout the website

we used mean and standard deviation check the results

Revenue/page view with price A: mean = 3.87, SD = 51.10

Revenue/page view with price B: mean = 4.11, SD = 62.98

Multivariate tests (MVT) : A multivariate test (MVT) tests variants of two or more elements simultaneously to see which combination creates the best outcome

Redirect tests :A redirect test, (a.k.a. *split URL test)*, is a type of A/B test that allows you to test separate web pages against each other

https://www.kaggle.com/code/myzziah/e-commerce-a-b-testing-full-experiment/notebook

Hypothesis Tests /significance testing

hypothesis and null Hypothesis both are equal

Null hypothesis
The hypothesis that chance is to blame.

Alternative hypothesis
Counterpoint to the null (what you hope to prove).

One-way test
Hypothesis test that counts chance results only in one direction.

Two-way test
Hypothesis test that counts chance results in two directions.

when you want to prove Hypothesis is write we need to prove null Hypothesis is wrong

hypothesis means some time when random events coin flips test 1000 times some where it comes 9 heads at a time human think that it was not random nature hypothesis prove that it was random event

null hypothesis and alternative hypothesis are like same you have data about table A and tablet B result of the data is tablet A is work well then tablet b

then you prove tablet B is null hypothesis automatically prove that A is working well (A is reject the null hypothesis)

One-Way Versus Two-Way Hypothesis Tests

One more consideration is whether we should run a one or two-tailed hypothesis test. If you look at the image below from statisticshowto.com then you'll see a two tailed rejection region (because we're rejecting the null hypothesis) on either side of the graph in yellow. The center is the default mean (what the null hypothesis claims the alternative treatment won't significantly deviate from). In a two-tailed test like the one below, we're typically just testing that the alternative hypothesis is

significantly different from the null hypothesis without being too concerned about the direction in

which it's different. In a one-tail test, we can also test the direction in which the alternative

treatment is different. It is important to understand that the one-tail test also doesn't test either the left or the right side of the graph (of course) so we need to be confident that only one direction away from the mean needs to be tested and that we tested the correct direction.

https://www.youtube.com/watch?v=XHPIEp-3yC0

• A null hypothesis is a logical construct embodying the notion that nothing special has happened, and any effect you observe is due to random chance.

• The hypothesis test assumes that the null hypothesis is true, creates a “null model” (a probability model), and tests whether the effect you observe is a reasonable outcome of that model

Resampling

pair plot

Resampling in statistics means to repeatedly sample values from observed data, with a general goal of assessing random variability in a statistic. It can also be used to assess and improve the accuracy of some machine-learning models (e.g., the predictions from decision tree models built on multiple bootstrapped data sets can be averaged in a process known as bagging—see

There are two main types of resampling procedures: the bootstrap and permutation tests

Permutation test

The procedure of combining two or more samples together and randomly (or exhaustively) reallocating the observations to resamples. up to repeat R times

we combine two are more models and you got one random(permutation group) shuffle sample and you take other values we pick the other models

Exhaustive and Bootstrap Permutation Tests

• An exhaustive permutation test

In an exhaustive permutation test, instead of just randomly shuffling and dividing the data, we actually figure out all the possible ways it could be divided. This is practical only for relatively small sample sizes. With a large number of repeated shuffling, the random permutation test results approximate those of the exhaustive permutation test, and approach them in the limit

take random sample of large data

• A bootstrap permutation test

Permutation Tests: The Bottom Line for Data Science

Permutation tests are useful heuristic procedures for exploring the role of random variation. They are relatively easy to code, interpret, and explain, and they offer a use- ful detour around the formalism and “false determinism” of formula-based statistics, in which the precision of formula “answers” tends to imply unwarranted certainty. One virtue of resampling, in contrast to formula approaches, is that it comes much closer to a one-size-fits-all approach to inference. Data can be numeric or binary. Sample sizes can be the same or different. Assumptions about normally distributed data are not needed.

Key Ideas

- In a permutation test, multiple samples are combined and then shuffled.
- The shuffled values are then divided into resamples, and the statistic of interest is calculated.
- This process is then repeated, and the resampled statistic is tabulated.
- Comparing the observed value of the statistic to the resampled distribution allows you to judge whether an observed difference between samples might occur by chance.

```python
def permutation_function(dataframe, data_col, grouping_col, iterations, statistic_functi
on): # resamples without replacement each group and outputs a dictionary
    output = {}

    groups = dataframe[grouping_col].value_counts().index
    values = dataframe[grouping_col].value_counts()

    for x in range(iterations):
        for i in groups:
            sample_amount = dataframe[grouping_col].value_counts()[i]
            sample_data = dataframe[data_col].sample(sample_amount, replace=False).to_li
st()
            sample_metric = statistic_function(sample_data)
            try:
                output[i].append(sample_metric)
            except (KeyError, AttributeError):
                output[i] = [sample_metric]

    final_output = pd.DataFrame(output)
    return final_output
```

Example: Web Stickiness

A company selling a relatively high-value service wants to test which of two web presentations does a better selling job. Due to the high value of the service being sold, sales are infrequent and the sales cycle is lengthy; it would take too long to accumulate enough sales to know which presentation is superior. So the company decides to measure the results with a proxy variable, using the detailed interior page that describes the service.

A *proxy* variable is one that stands in for the true variable of interest, which may be unavailable, too costly, or too time-consuming to measure. In climate research, for example, the oxygen content of ancient ice cores is used as a proxy for temperature. It is useful to have at least *some* data on the true variable of interest, so the strength of its association with the proxy can be assessed.

One potential proxy variable for our company is the number of clicks on the detailed landing page. A better one is how long people spend on the page. It is reasonable to think that a web presentation (page) that holds people's attention longer will lead to more sales. Hence, our metric is average session time, comparing page A to page B.

Due to the fact that this is an interior, special-purpose page, it does not receive a huge number of visitors. Also note that Google Analytics, which is how we measure session time, cannot measure session time for the last session a person visits. Instead of deleting that session from the data, though, Google Analytics records it as a zero, so the data requires additional processing to remove those sessions. The result is a total of 36 sessions for the two different presentations, 21 for page A and 15 for page B. Using `ggplot`, we can visually compare the session times using side-by-side boxplots:

```
ggplot(session_times, aes(x=Page, y=Time)) +
  geom_boxplot()
```

The pandas boxplot command uses the keyword argument by to create the figure:

```
ax = session_times.boxplot(by='Page', column='Time')
ax.set_xlabel('')
ax.set_ylabel('Time (in seconds)')
plt.suptitle('')
```

The boxplot, shown in Figure 3-3, indicates that page B leads to longer sessions than page A. The means for each group can be computed in *R* as follows:

```
mean_a <- mean(session_times[session_times['Page'] == 'Page A', 'Time'])
mean_b <- mean(session_times[session_times['Page'] == 'Page B', 'Time'])
mean_b - mean_a
[1] 35.66667
```

In *Python*, we filter the pandas data frame first by page and then determine the mean of the Time column:

```
mean_a = session_times[session_times.Page == 'Page A'].Time.mean()
mean_b = session_times[session_times.Page == 'Page B'].Time.mean()
mean_b - mean_a
```

Page B has session times that are greater than those of page A by 35.67 seconds, on average. The question is whether this difference is within the range of what random chance might produce, i.e., is statistically significant. One way to answer this is to apply a permutation test—combine all the session times together and then repeatedly shuffle and divide them into groups of 21 (recall that $n_A = 21$ for page A) and 15 ($n_B = 15$ for page B).

To apply a permutation test, we need a function to randomly assign the 36 session times to a group of 21 (page A) and a group of 15 (page B). The *R* version of this function is:

```
perm_fun <- function(x, nA, nB)
{
  n <- nA + nB
  idx_b <- sample(1:n, nB)
  idx_a <- setdiff(1:n, idx_b)
  mean_diff <- mean(x[idx_b]) - mean(x[idx_a])
  return(mean_diff)
}
```

The *Python* version of this permutation test is the following:

```
def perm_fun(x, nA, nB):
    n = nA + nB
    idx_B = set(random.sample(range(n), nB))
    idx_A = set(range(n)) - idx_B
    return x.loc[idx_B].mean() - x.loc[idx_A].mean()
```

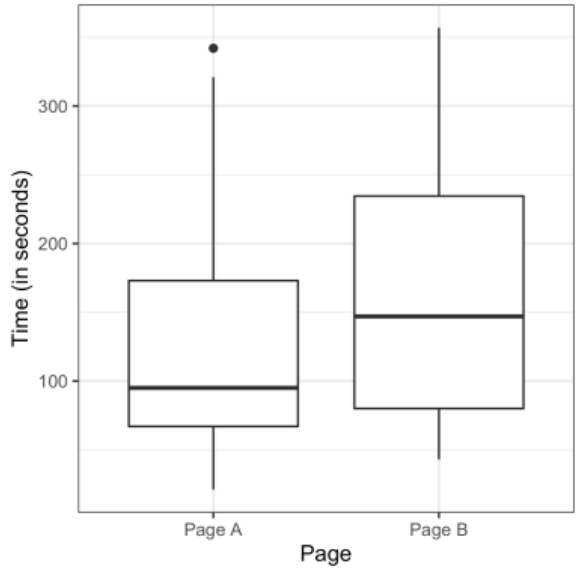


Figure 3-3. Session times for web pages A and B

This function works by sampling (without replacement) n_B indices and assigning them to the B group; the remaining n_A indices are assigned to group A. The difference between the two means is returned. Calling this function R = 1,000 times and specifying n_A = 21 and n_B = 15 leads to a distribution of differences in the session times that can be plotted as a histogram. In *R* this is done as follows using the `hist` function:

```
perm_diffs <- rep(0, 1000)
for (i in 1:1000) {
  perm_diffs[i] = perm_fun(session_times[, 'Time'], 21, 15)
}
hist(perm_diffs, xlab='Session time differences (in seconds)')
abline(v=mean_b - mean_a)
```

In *Python*, we can create a similar graph using `matplotlib`:

```
perm_diffs = [perm_fun(session_times.Time, nA, nB) for _ in range(1000)]

fig, ax = plt.subplots(figsize=(5, 5))
ax.hist(perm_diffs, bins=11, rwidth=0.9)
ax.axvline(x = mean_b - mean_a, color='black', lw=2)
```

```
ax.text(50, 190, 'Observed\ndifference', bbox={'facecolor':'white'})
ax.set_xlabel('Session time differences (in seconds)')
ax.set_ylabel('Frequency')
```

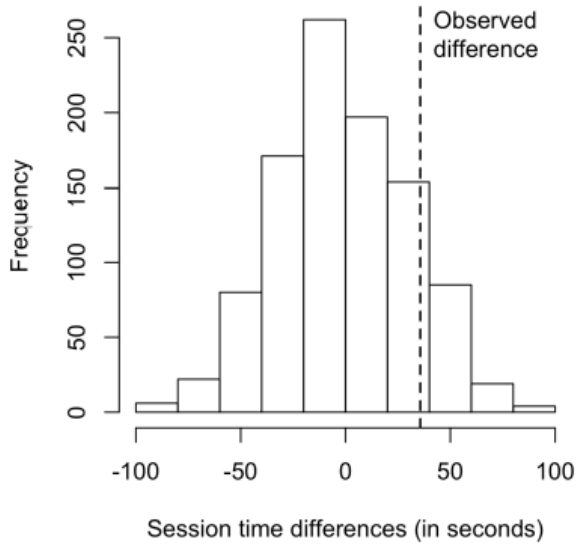
The histogram, in Figure 3-4 shows that mean difference of random permutations often exceeds the observed difference in session times (the vertical line). For our results, this happens in 12.6% of the cases:

```
mean(perm_diffs > (mean_b - mean_a))
---
0.126
```

As the simulation uses random numbers, the percentage will vary. For example, in the *Python* version, we got 12.1%:

```
np.mean(perm_diffs > mean_b - mean_a)
---
0.121
```

This suggests that the observed difference in session time between page A and page B is well within the range of chance variation and thus is not statistically significant.



Exhaustive and Bootstrap Permutation Tests

In addition to the preceding random shuffling procedure, also called a *random permutation test* or a *randomization test*, there are two variants of the permutation test:

- An *exhaustive permutation test*
- A *bootstrap permutation test*

In an exhaustive permutation test, instead of just randomly shuffling and dividing the data, we actually figure out all the possible ways it could be divided. This is practical only for relatively small sample sizes. With a large number of repeated shufflings, the random permutation test results approximate those of the exhaustive permutation test, and approach them in the limit. Exhaustive permutation tests are also sometimes called *exact tests*, due to their statistical property of guaranteeing that the null model will not test as "significant" more than the alpha level of the test (see "Statistical Significance and p-Values" on page 103).

In a bootstrap permutation test, the draws outlined in steps 2 and 3 of the random permutation test are made *with replacement* instead of without replacement. In this way the resampling procedure models not just the random element in the assignment of treatment to subject but also the random element in the selection of subjects from a population. Both procedures are encountered in statistics, and the distinction between them is somewhat convoluted and not of consequence in the practice of data science.

Permutation Tests: The Bottom Line for Data Science

Permutation tests are useful heuristic procedures for exploring the role of random variation. They are relatively easy to code, interpret, and explain, and they offer a useful detour around the formalism and "false determinism" of formula-based statistics, in which the precision of formula "answers" tends to imply unwarranted certainty.

One virtue of resampling, in contrast to formula approaches, is that it comes much closer to a one-size-fits-all approach to inference. Data can be numeric or binary. Sample sizes can be the same or different. Assumptions about normally distributed data are not needed.

Statistical Significance and p-Values

Statistical significance is how statisticians measure whether an experiment (or even a study of existing data) yields a result more extreme than what chance might produce. If the result is beyond the realm of chance variation, it is said to be statistically significant

p-value : we need to prove that null hypothesis and hypothesis one of them is wrong then it was prove if you prove null hypothesis the value is correct either wrong

Given a chance model that embodies the null hypothesis, the p-value is the probability of obtaining results as unusual or extreme as the observed results.

scipy.stats.chi2_contingency

Alpha :

The alpha value is the probability threshold for statistical significance. The most common alpha value is **p = 0.05, but 0.1, 0.01, and even 0.001** are sometimes used. It's best to look at the papers published in your field to decide which alpha value to use.

The probability threshold of "unusualness" that chance results must surpass for actual outcomes to be deemed statistically significant.

> ***Type 1 error***
> Mistakenly concluding an effect is real (when it is due to chance).
>
> ***Type 2 error***
> Mistakenly concluding an effect is due to chance (when it is real).

Table 3-2. 2×2 table for ecommerce experiment results

Outcome	Price A	Price B
Conversion	200	182
No conversion	23,539	22,406

Price A converts almost 5% better than price B (0.8425% = 200/(23539+200)*100, versus 0.8057% = 182/(22406+182)*100—a difference of 0.0368 percentage points), big enough to be meaningful in a high-volume business. We have over 45,000 data points here, and it is tempting to consider this as "big data," not requiring tests of statistical significance (needed mainly to account for sampling variability in small samples). However, the conversion rates are so low (less than 1%) that the actual meaningful values—the conversions—are only in the 100s, and the sample size needed is really determined by these conversions. We can test whether the difference in conversions between prices A and B is within the range of *chance variation*, using a resampling procedure. By chance variation, we mean the random variation produced by a probability model that embodies the null hypothesis that there is no difference between the rates (see "The Null Hypothesis" on page 94). The following permutation procedure asks, "If the two prices share the same conversion rate, could chance variation produce a difference as big as 5%?"

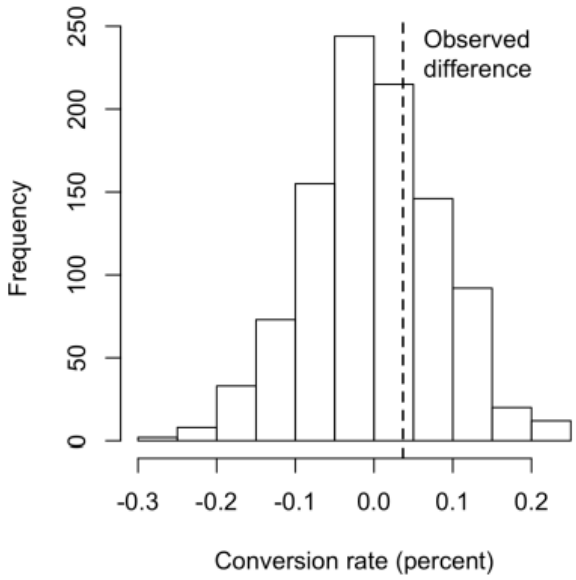
1. Put cards labeled 1 and 0 in a box: this represents the supposed shared conversion rate of 382 ones and 45,945 zeros = 0.008246 = 0.8246%.
2. Shuffle and draw out a resample of size 23,739 (same *n* as price A), and record how many 1s.
3. Record the number of 1s in the remaining 22,588 (same *n* as price B).
4. Record the difference in proportion of 1s.
5. Repeat steps 2–4.
6. How often was the difference >= 0.0368?

The corresponding *Python* code is:

```
obs_pct_diff = 100 * (200 / 23739 - 182 / 22588)
print(f'Observed difference: {obs_pct_diff:.4f}%')
conversion = [0] * 45945
conversion.extend([1] * 382)
conversion = pd.Series(conversion)

perm_diffs = [100 * perm_fun(conversion, 23739, 22588)
              for _ in range(1000)]

fig, ax = plt.subplots(figsize=(5, 5))
ax.hist(perm_diffs, bins=11, rwidth=0.9)
ax.axvline(x=obs_pct_diff, color='black', lw=2)
ax.text(0.06, 200, 'Observed\ndifference', bbox={'facecolor':'white'})
ax.set_xlabel('Conversion rate (percent)')
ax.set_ylabel('Frequency')
```



p-Value

In this case, we didn't need to use a permutation test to get a p-value. Since we have a binomial distribution, we can approximate the p-value. In *R* code, we do this using the function `prop.test`:

```
> prop.test(x=c(200, 182), n=c(23739, 22588), alternative='greater')

        2-sample test for equality of proportions with continuity correction

data:  c(200, 182) out of c(23739, 22588)
X-squared = 0.14893, df = 1, p-value = 0.3498
alternative hypothesis: greater
95 percent confidence interval:
 -0.001057439  1.000000000
sample estimates:
     prop 1      prop 2
0.008424955 0.008057376
```

The argument `x` is the number of successes for each group, and the argument `n` is the number of trials.

The method `scipy.stats.chi2_contingency` takes the values as shown in Table 3-2:

```
survivors = np.array([[200, 23739 - 200], [182, 22588 - 182]])
chi2, p_value, df, _ = stats.chi2_contingency(survivors)

print(f'p-value for single sided test: {p_value / 2:.4f}')
```

The normal approximation yields a p-value of 0.3498, which is close to the p-value obtained from the permutation test.

Data Science and p-Values

The work that data scientists do is typically not destined for publication in scientific journals, so the debate over the value of a p-value is somewhat academic. For a data scientist, a p-value is a useful metric in situations where you want to know whether a model result that appears interesting and useful is within the range of normal chance variability. As a decision tool in an experiment, a p-value should not be considered controlling, but merely another point of information bearing on a decision. For example, p-values are sometimes used as intermediate inputs in some statistical or machine learning models—a feature might be included in or excluded from a model depending on its p-value.

Key Ideas

- Significance tests are used to determine whether an observed effect is within the range of chance variation for a null hypothesis model.
- The p-value is the probability that results as extreme as the observed results might occur, given a null hypothesis model.
- The alpha value is the threshold of "unusualness" in a null hypothesis chance model.
- Significance testing has been much more relevant for formal reporting of research than for data science (but has been fading recently, even for the former).

t-Tests

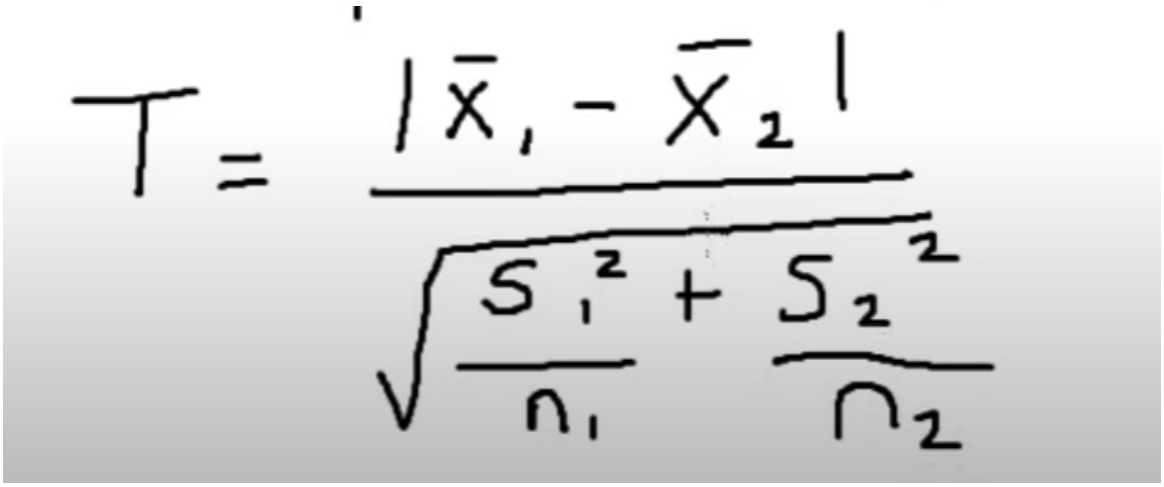
There are numerous types of significance tests, depending on whether the data comprises count data or measured data, how many samples there are, and what's being measured. A very common one is the *t-test*, named after Student's t-distribution, originally developed by W. S. Gosset to approximate the distribution of a single sample mean (see "Student's t-Distribution" on page 75).

Key Terms for t-Tests

Test statistic
A metric for the difference or effect of interest.

t-statistic
A standardized version of common test statistics such as means.

t-distribution
A reference distribution (in this case derived from the null hypothesis), to which the observed t-statistic can be compared.



The function `scipy.stats.ttest_ind` can be used in *Python*:

```
res = stats.ttest_ind(session_times[session_times.Page == 'Page A'].Time,
                      session_times[session_times.Page == 'Page B'].Time,
                      equal_var=False)
print(f'p-value for single sided test: {res.pvalue / 2:.4f}')
```

The alternative hypothesis is that the session time mean for page A is less than that for page B. The p-value of 0.1408 is fairly close to the permutation test p-values of 0.121 and 0.126 (see "Example: Web Stickiness" on page 98).

In a resampling mode, we structure the solution to reflect the observed data and the hypothesis to be tested, not worrying about whether the data is numeric or binary, whether or not sample sizes are balanced, sample variances, or a variety of other factors. In the formula world, many variations present themselves, and they can be bewildering. Statisticians need to navigate that world and learn its map, but data scientists do not—they are typically not in the business of sweating the details of hypothesis tests and confidence intervals the way a researcher preparing a paper for presentation might.

Key Ideas

- Before the advent of computers, resampling tests were not practical, and statisticians used standard reference distributions.
- A test statistic could then be standardized and compared to the reference distribution.
- One such widely used standardized statistic is the t-statistic.

Multiple Testing

As we've mentioned previously, there is a saying in statistics: "Torture the data long enough, and it will confess." This means that if you look at the data through enough

different perspectives and ask enough questions, you almost invariably will find a statistically significant effect.

For example, if you have 20 predictor variables and one outcome variable, all randomly generated, the odds are pretty good that at least one predictor will (falsely) turn out to be statistically significant if you do a series of 20 significance tests at the alpha = 0.05 level. As previously discussed, this is called a Type 1 error. You can calculate this probability by first finding the probability that all will correctly test nonsignificant at the 0.05 level. The probability that one will correctly test nonsignificant is 0.95, so the probability that all 20 will correctly test nonsignificant is 0.95 × 0.95 × 0.95…, or 0.9520 = 0.36.1 The probability that at least one predictor will (falsely) test significant is the flip side of this probability, or 1 – (probability that all will be nonsignificant) = 0.64. This is known as alpha inflation. This issue is related to the problem of overfitting in data mining, or "fitting the model to the noise." The more variables you add, or the more models you run, the greater the probability that something will emerge as "significant" just by chance

Key Terms for Multiple Testing

Type 1 error
Mistakenly concluding that an effect is statistically significant.

False discovery rate
Across multiple tests, the rate of making a Type 1 error.

Alpha inflation
The multiple testing phenomenon, in which *alpha*, the probability of making a Type 1 error, increases as you conduct more tests.

Adjustment of p-values
Accounting for doing multiple tests on the same data.

Overfitting
Fitting the noise.

In supervised learning tasks, a holdout set where models are assessed on data that the model has not seen before mitigates this risk. In statistical and machine learning tasks not involving a labeled holdout set, the risk of reaching conclusions based on statistical noise persists.

In statistics, there are some procedures intended to deal with this problem in very specific circumstances. For example, if you are comparing results across multiple treatment groups, you might ask multiple questions. So, for treatments A–C, you might ask:

- Is A different from B?
- Is B different from C?
- Is A different from C?

Or, in a clinical trial, you might want to look at results from a therapy at multiple stages. In each case, you are asking multiple questions, and with each question, you are increasing the chance of being fooled by chance. Adjustment procedures in statistics can compensate for this by setting the bar for statistical significance more stringently than it would be set for a single hypothesis test. These adjustment procedures typically involve "dividing up the alpha" according to the number of tests. This results in a smaller alpha (i.e., a more stringent bar for statistical significance) for each test. One such procedure, the Bonferroni adjustment, simply divides the alpha by the number of comparisons. Another, used in comparing multiple group means, is Tukey's "honest significant difference," or *Tukey's HSD*. This test applies to the maximum difference among group means, comparing it to a benchmark based on the *t-distribution* (roughly equivalent to shuffling all the values together, dealing out resampled groups of the same sizes as the original groups, and finding the maximum difference among the resampled group means).

However, the problem of multiple comparisons goes beyond these highly structured cases and is related to the phenomenon of repeated data "dredging" that gives rise to the saying about torturing the data. Put another way, given sufficiently complex data, if you haven't found something interesting, you simply haven't looked long and hard enough. More data is available now than ever before, and the number of journal articles published nearly doubled between 2002 and 2010. This gives rise to lots of opportunities to find something interesting in the data, including multiplicity issues such as:

- Checking for multiple pairwise differences across groups
- Looking at multiple subgroup results ("we found no significant treatment effect overall, but we did find an effect for unmarried women younger than 30")
- Trying lots of statistical models
- Including lots of variables in models
- Asking a number of different questions (i.e., different possible outcomes)

False Discovery Rate

The term *false discovery rate* was originally used to describe the rate at which a given set of hypothesis tests would falsely identify a significant effect. It became particularly useful with the advent of genomic research, in which massive numbers of statistical tests might be conducted as part of a gene sequencing project. In these cases, the term applies to the testing protocol, and a single false "discovery" refers to the outcome of a hypothesis test (e.g., between two samples). Researchers sought to set the parameters of the testing process to control the false discovery rate at a specified level. The term has also been used for classification in data mining; it is the misclassification rate within the class 1 predictions. Or, put another way, it is the probability that a "discovery" (labeling a record as a "1") is false. Here we typically are dealing with the case where 0s are abundant and 1s are interesting and rare (see Chapter 5 and "The Rare Class Problem" on page 223).

For a variety of reasons, including especially this general issue of "multiplicity," more research does not necessarily mean better research. For example, the pharmaceutical company Bayer found in 2011 that when it tried to replicate 67 scientific studies, it could fully replicate only 14 of them. Nearly two-thirds could not be replicated at all.

In any case, the adjustment procedures for highly defined and structured statistical tests are too specific and inflexible to be of general use to data scientists. The bottom line for data scientists on multiplicity is:

- For predictive modeling, the risk of getting an illusory model whose apparent efficacy is largely a product of random chance is mitigated by cross-validation (see "Cross-Validation" on page 155) and use of a holdout sample.
- For other procedures without a labeled holdout set to check the model, you must rely on:
  - Awareness that the more you query and manipulate the data, the greater the role that chance might play.
  - Resampling and simulation heuristics to provide random chance benchmarks against which observed results can be compared.

Key Ideas

- Multiplicity in a research study or data mining project (multiple comparisons, many variables, many models, etc.) increases the risk of concluding that something is significant just by chance.
- For situations involving multiple statistical comparisons (i.e., multiple tests of significance), there are statistical adjustment procedures.
- In a data mining situation, use of a holdout sample with labeled outcome variables can help avoid misleading results.

Degrees of Freedom

In the documentation and settings for many statistical tests and probability distributions, you will see a reference to "degrees of freedom." The concept is applied to statistics calculated from sample data, and refers to the number of values free to vary. For example, if you know the mean for a sample of 10 values, there are 9 degrees of freedom (once you know 9 of the sample values, the 10th can be calculated and is not free to vary). The degrees of freedom parameter, as applied to many probability distributions, affects the shape of the distribution.

The number of degrees of freedom is an input to many statistical tests. For example, degrees of freedom is the name given to the $n - 1$ denominator seen in the calculations for variance and standard deviation. Why does it matter? When you use a sample to estimate the variance for a population, you will end up with an estimate that is slightly biased downward if you use n in the denominator. If you use $n - 1$ in the denominator, the estimate will be free of that bias.

Key Terms for Degrees of Freedom

n or sample size
: The number of observations (also called *rows* or *records*) in the data.

d.f.
: Degrees of freedom.

A large share of a traditional statistics course or text is consumed by various standard tests of hypotheses (t-test, F-test, etc.). When sample statistics are standardized for use in traditional statistical formulas, degrees of freedom is part of the standardization calculation to ensure that your standardized data matches the appropriate reference distribution (t-distribution, F-distribution, etc.).

Is it important for data science? Not really, at least in the context of significance testing. For one thing, formal statistical tests are used only sparingly in data science. For another, the data size is usually large enough that it rarely makes a real difference for a data scientist whether, for example, the denominator has n or $n - 1$. (As n gets large, the bias that would come from using n in the denominator disappears.)

There is one context, though, in which it is relevant: the use of factored variables in regression (including logistic regression). Some regression algorithms choke if exactly redundant predictor variables are present. This most commonly occurs when factoring categorical variables into binary indicators (dummies). Consider the variable "day of week." Although there are seven days of the week, there are only six degrees of freedom in specifying day of week. For example, once you know that day of week is not Monday through Saturday, you know it must be Sunday. Inclusion of the Mon–Sat indicators thus means that *also* including Sunday would cause the regression to fail, due to a *multicollinearity* error.

Key Ideas

- The number of degrees of freedom (d.f.) forms part of the calculation to standardize test statistics so they can be compared to reference distributions (t-distribution, F-distribution, etc.).
- The concept of degrees of freedom lies behind the factoring of categorical variables into $n - 1$ indicator or dummy variables when doing a regression (to avoid multicollinearity).

[What Are Degrees of Freedom in Statistics? (minitab.com)]

ANOVA

Suppose that, instead of an A/B test, we had a comparison of multiple groups, say A/B/C/D, each with numeric data. The statistical procedure that tests for a statistically significant difference among the groups is called analysis of variance, or ANOVA

Key Terms for ANOVA

Pairwise comparison
A hypothesis test (e.g., of means) between two groups among multiple groups.

Omnibus test
A single hypothesis test of the overall variance among multiple group means.

Decomposition of variance
Separation of components contributing to an individual value (e.g., from the overall average, from a treatment mean, and from a residual error).

F-statistic
A standardized statistic that measures the extent to which differences among group means exceed what might be expected in a chance model.

SS
"Sum of squares," referring to deviations from some average value.

```
observed_variance = four_sessions.groupby('Page').mean().var()[0]
print('Observed means:', four_sessions.groupby('Page').mean().values.ravel())
print('Variance:', observed_variance)

def perm_test(df): df = df.copy()
  df['Time'] = np.random.permutation(df['Time'].values)
  return df.groupby('Page').mean().var()[0]

perm_variance = [perm_test(four_sessions) for _ in range(3000)]
print('Pr(Prob)', np.mean([var > observed_variance for var in perm_variance]))
```

the stickiness of four web pages, defined as the number of seconds a visitor spent on the page. The four pages are switched out so that each web visitor receives one at random. There are a total of five visitors for each page, and in Table 3-3, each column is an independent set of data. The first viewer for page 1 has no connection to the first viewer for page 2. Note that in a web test like this, we can- not fully implement the classic randomized sampling design in which each visitor is selected at random from some huge population. We must take the visitors as they come. Visitors may systematically differ depending on time of day, time of week, sea- son of the year, conditions of their internet, what device they are using, and so on. These factors should be considered as potential bias when the experiment results are reviewed.

Table 3-3. Stickiness (in seconds) of four web pages

	Page 1	Page 2	Page 3	Page 4
	164	178	175	155
	172	191	193	166
	177	182	171	164
	156	185	163	170
	195	177	176	168
Average	172	185	176	162
Grand average				173.75

Now we have a conundrum (see Figure 3-6). When we were comparing just two groups, it was a simple matter; we merely looked at the difference between the means of each group. With four means, there are six possible comparisons between groups:

- Page 1 compared to page 2
- Page 1 compared to page 3
- Page 1 compared to page 4
- Page 2 compared to page 3
- Page 2 compared to page 4
- Page 3 compared to page 4

The more such *pairwise* comparisons we make, the greater the potential for being fooled by random chance (see "Multiple Testing" on page 112). Instead of worrying about all the different comparisons between individual pages we could possibly make, we can do a single overall test that addresses the question, "Could all the pages have the same underlying stickiness, and the differences among them be due to the random way in which a common set of session times got allocated among the four pages?"

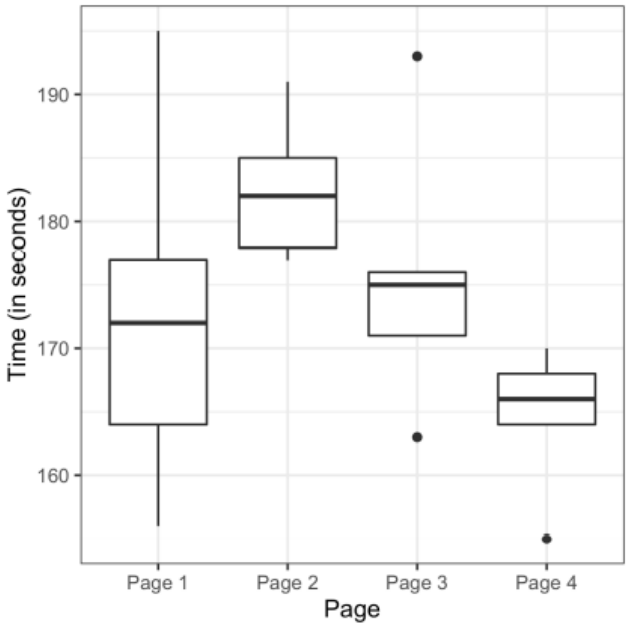


Figure 3-6. Boxplots of the four groups show considerable differences among them

The procedure used to test this is ANOVA. The basis for it can be seen in the following resampling procedure (specified here for the A/B/C/D test of web page stickiness):

1. Combine all the data together in a single box.
2. Shuffle and draw out four resamples of five values each.
3. Record the mean of each of the four groups.
4. Record the variance among the four group means.
5. Repeat steps 2–4 many (say, 1,000) times.

What proportion of the time did the resampled variance exceed the observed variance? This is the p-value.

This type of permutation test is a bit more involved than the type used in "Permutation Test" on page 97. Fortunately, the `aovp` function in the `lmPerm` package computes a permutation test for this case:

The p-value, given by `Pr(Prob)`, is 0.09278. In other words, given the same underlying stickiness, 9.3% of the time the response rate among four pages might differ as much as was actually observed, just by chance. This degree of improbability falls short of the traditional statistical threshold of 5%, so we conclude that the difference among the four pages could have arisen by chance.

The column `Iter` lists the number of iterations taken in the permutation test. The other columns correspond to a traditional ANOVA table and are described next.

In *Python*, we can compute the permutation test using the following code:

```
observed_variance = four_sessions.groupby('Page').mean().var()[0]
print('Observed means:', four_sessions.groupby('Page').mean().values.ravel())
print('Variance:', observed_variance)

def perm_test(df):
    df = df.copy()
    df['Time'] = np.random.permutation(df['Time'].values)
    return df.groupby('Page').mean().var()[0]

perm_variance = [perm_test(four_sessions) for _ in range(3000)]
print('Pr(Prob)', np.mean([var > observed_variance for var in perm_variance]))
```

F-Statistic

Just like the t-test can be used instead of a permutation test for comparing the mean of two groups, there is a statistical test for ANOVA based on the *F-statistic*. The F-statistic is based on the ratio of the variance across group means (i.e., the treatment effect) to the variance due to residual error. The higher this ratio, the more statistically significant the result. If the data follows a normal distribution, then statistical theory dictates that the statistic should have a certain distribution. Based on this, it is possible to compute a p-value.

The `statsmodels` package provides an ANOVA implementation in *Python*:

```
model = smf.ols('Time ~ Page', data=four_sessions).fit()

aov_table = sm.stats.anova_lm(model)
aov_table
```

The output from the *Python* code is almost identical to that from *R*.

`Df` is "degrees of freedom," `Sum Sq` is "sum of squares," `Mean Sq` is "mean squares" (short for mean-squared deviations), and `F value` is the F-statistic. For the grand average, sum of squares is the departure of the grand average from 0, squared, times 20 (the number of observations). The degrees of freedom for the grand average is 1, by definition.

For the treatment means, the degrees of freedom is 3 (once three values are set, and then the grand average is set, the other treatment mean cannot vary). Sum of squares for the treatment means is the sum of squared departures between the treatment means and the grand average.

For the residuals, degrees of freedom is 20 (all observations can vary), and SS is the sum of squared difference between the individual observations and the treatment means. Mean squares (MS) is the sum of squares divided by the degrees of freedom.

The F-statistic is MS(treatment)/MS(error). The F value thus depends only on this ratio and can be compared to a standard F-distribution to determine whether the differences among treatment means are greater than would be expected in random chance variation.

Decomposition of Variance

Observed values in a data set can be considered sums of different components. For any observed data value within a data set, we can break it down into the grand average, the treatment effect, and the residual error. We call this a "decomposition of variance":

1. Start with grand average (173.75 for web page stickiness data).
2. Add treatment effect, which might be negative (independent variable = web page).
3. Add residual error, which might be negative.

Thus the decomposition of the variance for the top-left value in the A/B/C/D test table is as follows:

1. Start with grand average: 173.75.
2. Add treatment (group) effect: –1.75 (172 – 173.75).
3. Add residual: –8 (164 – 172).
4. Equals: 164.

Two-Way ANOVA

The A/B/C/D test just described is a "one-way" ANOVA, in which we have one factor (group) that is varying. We could have a second factor involved—say, "weekend versus weekday"—with data collected on each combination (group A weekend, group A weekday, group B weekend, etc.). This would be a "two-way ANOVA," and we would handle it in similar fashion to the one-way ANOVA by identifying the "interaction effect." After identifying the grand average effect and the treatment effect, we then separate the weekend and weekday observations for each group and find the difference between the averages for those subsets and the treatment average.

You can see that ANOVA and then two-way ANOVA are the first steps on the road toward a full statistical model, such as regression and logistic regression, in which multiple factors and their effects can be modeled (see Chapter 4).

Key Ideas

- ANOVA is a statistical procedure for analyzing the results of an experiment with multiple groups.
- It is the extension of similar procedures for the A/B test, used to assess whether the overall variation among groups is within the range of chance variation.

- A useful outcome of ANOVA is the identification of variance components associated with group treatments, interaction effects, and errors.

Chi-Square Test

Web testing often goes beyond A/B testing and tests multiple treatments at once. The chi-square test is used with count data to test how well it fits some expected distribution. The most common use of the *chi-square* statistic in statistical practice is with $r \times c$ contingency tables, to assess whether the null hypothesis of independence among variables is reasonable (see also "Chi-Square Distribution" on page 80).

The chi-square test was originally developed by Karl Pearson in 1900. The term *chi* comes from the Greek letter X used by Pearson in the article.

Key Terms for Chi-Square Test

Chi-square statistic
A measure of the extent to which some observed data departs from expectation.

Expectation or expected
How we would expect the data to turn out under some assumption, typically the null hypothesis.

$r \times c$ means "rows by columns"—a 2 × 3 table has two rows and three columns.

Chi-Square Test: A Resampling Approach

Suppose you are testing three different headlines—A, B, and C—and you run them each on 1,000 visitors, with the results shown in Table 3-4.

Table 3-4. Web testing results for three different headlines

	Headline A	Headline B	Headline C
Click	14	8	12
No-click	986	992	988

The headlines certainly appear to differ. Headline A returns nearly twice the click rate of B. The actual numbers are small, though. A resampling procedure can test whether the click rates differ to an extent greater than chance might cause. For this test, we need to have the "expected" distribution of clicks, and in this case, that would be under the null hypothesis assumption that all three headlines share the same click rate, for an overall click rate of 34/3,000. Under this assumption, our contingency table would look like Table 3-5.

Table 3-5. Expected if all three headlines have the same click rate (null hypothesis)

	Headline A	Headline B	Headline C
Click	11.33	11.33	11.33
No-click	988.67	988.67	988.67

The *Pearson residual* is defined as:

$$R = \frac{\text{Observed} - \text{Expected}}{\sqrt{\text{Expected}}}$$

R measures the extent to which the actual counts differ from these expected counts (see Table 3-6).

Table 3-6. Pearson residuals

	Headline A	Headline B	Headline C
Click	0.792	−0.990	0.198
No-click	−0.085	0.106	−0.021

The chi-square statistic is defined as the sum of the squared Pearson residuals:

$$X = \sum_{i}^{r} \sum_{j}^{c} R^2$$

where r and c are the number of rows and columns, respectively. The chi-square statistic for this example is 1.666. Is that more than could reasonably occur in a chance model?

We can test with this resampling algorithm:

1. Constitute a box with 34 ones (clicks) and 2,966 zeros (no clicks).
2. Shuffle, take three separate samples of 1,000, and count the clicks in each.
3. Find the squared differences between the shuffled counts and the expected counts and sum them.
4. Repeat steps 2 and 3, say, 1,000 times.
5. How often does the resampled sum of squared deviations exceed the observed? That's the p-value.

To run a permutation test in *Python*, use the following implementation:

```
box = [1] * 34
box.extend([0] * 2966)
random.shuffle(box)

def chi2(observed, expected):
    pearson_residuals = []
    for row, expect in zip(observed, expected):
        pearson_residuals.append([(observe - expect) ** 2 / expect
                                  for observe in row])
    # return sum of squares
    return np.sum(pearson_residuals)

expected_clicks = 34 / 3
expected_noclicks = 1000 - expected_clicks
expected = [34 / 3, 1000 - 34 / 3]
chi2observed = chi2(clicks.values, expected)

def perm_fun(box):
    sample_clicks = [sum(random.sample(box, 1000)),
                     sum(random.sample(box, 1000)),
                     sum(random.sample(box, 1000))]
    sample_noclicks = [1000 - n for n in sample_clicks]
    return chi2([sample_clicks, sample_noclicks], expected)

perm_chi2 = [perm_fun(box) for _ in range(2000)]
```



```
resampled_p_value = sum(perm_chi2 > chi2observed) / len(perm_chi2)
print(f'Observed chi2: {chi2observed:.4f}')
print(f'Resampled p-value: {resampled_p_value:.4f}')
```

Chi-Square Test: Statistical Theory

Asymptotic statistical theory shows that the distribution of the chi-square statistic can be approximated by a *chi-square distribution* (see "Chi-Square Distribution" on page 80). The appropriate standard chi-square distribution is determined by the *degrees of freedom* (see "Degrees of Freedom" on page 116). For a contingency table, the degrees of freedom are related to the number of rows (r) and columns (c) as follows:

$$\text{degrees of freedom} = (r-1) \times (c-1)$$

The chi-square distribution is typically skewed, with a long tail to the right; see Figure 3-7 for the distribution with 1, 2, 5, and 20 degrees of freedom. The further out on the chi-square distribution the observed statistic is, the lower the p-value.

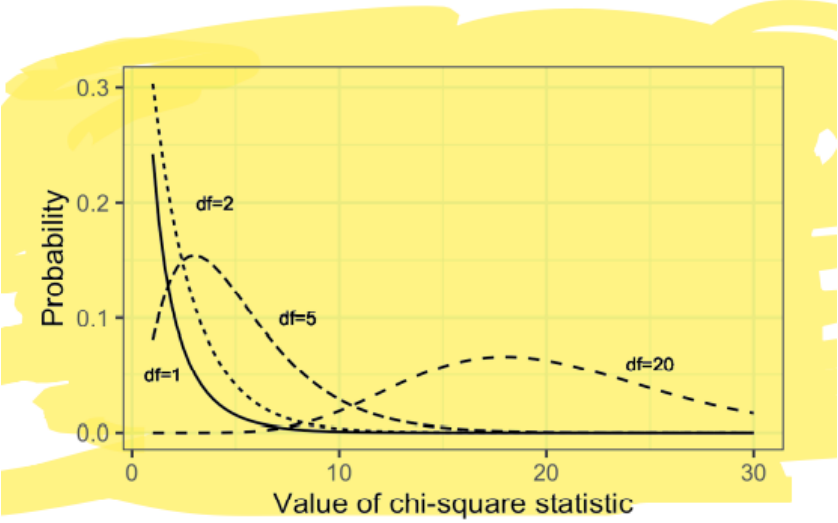


Figure 3-7. Chi-square distribution with various degrees of freedom

Relevance for Data Science

The chi-square test, or Fisher's exact test, is used when you want to know whether an effect is for real or might be the product of chance. In most classical statistical applications of the chi-square test, its role is to establish statistical significance, which is typically needed before a study or an experiment can be published. This is not so important for data scientists. In most data science experiments, whether A/B or A/B/C…, the goal is not simply to establish statistical significance but rather to arrive at the best treatment. For this purpose, multi-armed bandits (see "Multi-Arm Bandit Algorithm" on page 131) offer a more complete solution.

One data science application of the chi-square test, especially Fisher's exact version, is in determining appropriate sample sizes for web experiments. These experiments often have very low click rates, and despite thousands of exposures, count rates might be too small to yield definitive conclusions in an experiment. In such cases, Fisher's exact test, the chi-square test, and other tests can be useful as a component of power and sample size calculations (see "Power and Sample Size" on page 135).

Chi-square tests are used widely in research by investigators in search of the elusive statistically significant p-value that will allow publication. Chi-square tests, or similar resampling simulations, are used in data science applications more as a filter to determine whether an effect or a feature is worthy of further consideration than as a formal test of significance. For example, they are used in spatial statistics and mapping to determine whether spatial data conforms to a specified null distribution (e.g., are crimes concentrated in a certain area to a greater degree than random chance would allow?). They can also be used in automated feature selection in machine learning, to assess class prevalence across features and identify features where the prevalence of a certain class is unusually high or low, in a way that is not compatible with random variation.

Key Ideas

- A common procedure in statistics is to test whether observed data counts are consistent with an assumption of independence (e.g., propensity to buy a particular item is independent of gender).
- The chi-square distribution is the reference distribution (which embodies the assumption of independence) to which the observed calculated chi-square statistic must be compared.

Further Reading

- R. A. Fisher's famous "Lady Tasting Tea" example from the beginning of the 20th century remains a simple and effective illustration of his exact test. Google "Lady Tasting Tea," and you will find a number of good writeups.
- Stat Trek offers a good tutorial on the chi-square test.

Multi-Arm Bandit Algorithm

Multi-arm bandits offer an approach to testing, especially web testing, that allows explicit optimization and more rapid decision making than the traditional statistical approach to designing experiments.

Key Terms for Multi-Arm Bandits

Multi-arm bandit
: An imaginary slot machine with multiple arms for the customer to choose from, each with different payoffs, here taken to be an analogy for a multitreatment experiment.

Arm
: A treatment in an experiment (e.g., "headline A in a web test").

Win
: The experimental analog of a win at the slot machine (e.g., "customer clicks on the link").

A traditional A/B test involves data collected in an experiment, according to a specified design, to answer a specific question such as, "Which is better, treatment A or treatment B?" The presumption is that once we get an answer to that question, the experimenting is over and we proceed to act on the results.

You can probably perceive several difficulties with that approach. First, our answer may be inconclusive: "effect not proven." In other words, the results from the experiment may suggest an effect, but if there is an effect, we don't have a big enough sample to prove it (to the satisfaction of the traditional statistical standards). What decision do we take? Second, we might want to begin taking advantage of results that come in prior to the conclusion of the experiment. Third, we might want the right to change our minds or to try something different based on additional data that comes in after the experiment is over. The traditional approach to experiments and hypothesis tests dates from the 1920s and is rather inflexible. The advent of computer power and software has enabled more powerful flexible approaches. Moreover, data science (and business in general) is not so worried about statistical significance, but concerned more with optimizing overall effort and results.

Bandit algorithms, which are very popular in web testing, allow you to test multiple treatments at once and reach conclusions faster than traditional statistical designs. They take their name from slot machines used in gambling, also termed one-armed bandits (since they are configured in such a way that they extract money from the gambler in a steady flow). If you imagine a slot machine with more than one arm, each arm paying out at a different rate, you would have a multi-armed bandit, which is the full name for this algorithm.

Your goal is to win as much money as possible and, more specifically, to identify and settle on the winning arm sooner rather than later. The challenge is that you don't know at what overall rate the arms pay out—you only know the results of individual pulls on the arms. Suppose each "win" is for the same amount, no matter which arm.

What differs is the probability of a win. Suppose further that you initially try each arm 50 times and get the following results:

> Arm A: 10 wins out of 50
> Arm B: 2 win out of 50
> Arm C: 4 wins out of 50

One extreme approach is to say, "Looks like arm A is a winner—let's quit trying the other arms and stick with A." This takes full advantage of the information from the initial trial. If A is truly superior, we get the benefit of that early on. On the other hand, if B or C is truly better, we lose any opportunity to discover that. Another extreme approach is to say, "This all looks to be within the realm of chance—let's keep pulling them all equally." This gives maximum opportunity for alternates to A to show themselves. However, in the process, we are deploying what seem to be inferior treatments. How long do we permit that? Bandit algorithms take a hybrid approach: we start pulling A more often, to take advantage of its apparent superiority, but we don't abandon B and C. We just pull them less often. If A continues to outperform, we continue to shift resources (pulls) away from B and C and pull A more often. If, on the other hand, C starts to do better, and A starts to do worse, we can shift pulls from A back to C. If one of them turns out to be superior to A and this was hidden in the initial trial due to chance, it now has an opportunity to emerge with further testing.

Now think of applying this to web testing. Instead of multiple slot machine arms, you might have multiple offers, headlines, colors, and so on being tested on a website. Customers either click (a "win" for the merchant) or don't click. Initially, the offers are shown randomly and equally. If, however, one offer starts to outperform the others, it can be shown ("pulled") more often. But what should the parameters of the algorithm that modifies the pull rates be? What "pull rates" should we change to, and when should we change?

Here is one simple algorithm, the epsilon-greedy algorithm for an A/B test:

1. Generate a uniformly distributed random number between 0 and 1.
2. If the number lies between 0 and epsilon (where epsilon is a number between 0 and 1, typically fairly small), flip a fair coin (50/50 probability), and:
   a. If the coin is heads, show offer A.
   b. If the coin is tails, show offer B.
3. If the number is ≥ epsilon, show whichever offer has had the highest response rate to date.

Epsilon is the single parameter that governs this algorithm. If epsilon is 1, we end up with a standard simple A/B experiment (random allocation between A and B for each

subject). If epsilon is 0, we end up with a purely *greedy* algorithm—one that chooses the best available immediate option (a local optimum). It seeks no further experimentation, simply assigning subjects (web visitors) to the best-performing treatment.

A more sophisticated algorithm uses "Thompson's sampling." This procedure "samples" (pulls a bandit arm) at each stage to maximize the probability of choosing the best arm. Of course you don't know which is the best arm—that's the whole problem!—but as you observe the payoff with each successive draw, you gain more information. Thompson's sampling uses a Bayesian approach: some prior distribution of rewards is assumed initially, using what is called a *beta distribution* (this is a common mechanism for specifying prior information in a Bayesian problem). As information accumulates from each draw, this information can be updated, allowing the selection of the next draw to be better optimized as far as choosing the right arm.

Bandit algorithms can efficiently handle 3+ treatments and move toward optimal selection of the "best." For traditional statistical testing procedures, the complexity of decision making for 3+ treatments far outstrips that of the traditional A/B test, and the advantage of bandit algorithms is much greater.

Key Ideas

- Traditional A/B tests envision a random sampling process, which can lead to excessive exposure to the inferior treatment.
- Multi-arm bandits, in contrast, alter the sampling process to incorporate information learned during the experiment and reduce the frequency of the inferior treatment.
- They also facilitate efficient treatment of more than two treatments.
- There are different algorithms for shifting sampling probability away from the inferior treatment(s) and to the (presumed) superior one.

Further Reading

- An excellent short treatment of multi-arm bandit algorithms is found in *Bandit Algorithms for Website Optimization*, by John Myles White (O'Reilly, 2012). White includes *Python* code, as well as the results of simulations to assess the performance of bandits.
- For more (somewhat technical) information about Thompson sampling, see "Analysis of Thompson Sampling for the Multi-armed Bandit Problem" by Shipra Agrawal and Navin Goyal.

Power and Sample Size

If you run a web test, how do you decide how long it should run (i.e., how many impressions per treatment are needed)? Despite what you may read in many guides to web testing, there is no good general guidance—it depends, mainly, on the frequency with which the desired goal is attained.

Key Terms for Power and Sample Size

Effect size
: The minimum size of the effect that you hope to be able to detect in a statistical test, such as "a 20% improvement in click rates."

Power
: The probability of detecting a given effect size with a given sample size.

Significance level
: The statistical significance level at which the test will be conducted.

One step in statistical calculations for sample size is to ask "Will a hypothesis test actually reveal a difference between treatments A and B?" The outcome of a hypothesis test—the p-value—depends on what the real difference is between treatment A and treatment B. It also depends on the luck of the draw—who gets selected for the groups in the experiment. But it makes sense that the bigger the actual difference between treatments A and B, the greater the probability that our experiment will reveal it; and the smaller the difference, the more data will be needed to detect it. To distinguish between a .350 hitter and a .200 hitter in baseball, not that many at-bats are needed. To distinguish between a .300 hitter and a .280 hitter, a good many more at-bats will be needed.

Power is the probability of detecting a specified *effect size* with specified sample characteristics (size and variability). For example, we might say (hypothetically) that the probability of distinguishing between a .330 hitter and a .200 hitter in 25 at-bats is 0.75. The effect size here is a difference of .130. And "detecting" means that a hypothesis test will reject the null hypothesis of "no difference" and conclude there is a real effect. So the experiment of 25 at-bats ($n = 25$) for two hitters, with an effect size of 0.130, has (hypothetical) power of 0.75, or 75%.

You can see that there are several moving parts here, and it is easy to get tangled up in the numerous statistical assumptions and formulas that will be needed (to specify sample variability, effect size, sample size, alpha-level for the hypothesis test, etc., and to calculate power). Indeed, there is special-purpose statistical software to calculate power. Most data scientists will not need to go through all the formal steps needed to report power, for example, in a published paper. However, they may face occasions

where they want to collect some data for an A/B test, and collecting or processing the data involves some cost. In that case, knowing approximately how much data to collect can help avoid the situation where you collect data at some effort, and the result ends up being inconclusive. Here's a fairly intuitive alternative approach:

1. Start with some hypothetical data that represents your best guess about the data that will result (perhaps based on prior data)—for example, a box with 20 ones and 80 zeros to represent a .200 hitter, or a box with some observations of "time spent on website."
2. Create a second sample simply by adding the desired effect size to the first sample—for example, a second box with 33 ones and 67 zeros, or a second box with 25 seconds added to each initial "time spent on website."
3. Draw a bootstrap sample of size *n* from each box.
4. Conduct a permutation (or formula-based) hypothesis test on the two bootstrap samples and record whether the difference between them is statistically significant.
5. Repeat the preceding two steps many times and determine how often the difference was significant—that's the estimated power.

Sample Size

The most common use of power calculations is to estimate how big a sample you will need.

For example, suppose you are looking at click-through rates (clicks as a percentage of exposures), and testing a new ad against an existing ad. How many clicks do you need to accumulate in the study? If you are interested only in results that show a huge difference (say, a 50% difference), a relatively small sample might do the trick. If, on the other hand, even a minor difference would be of interest, then a much larger sample is needed. A standard approach is to establish a policy that a new ad must do better than an existing ad by some percentage, say, 10%; otherwise, the existing ad will remain in place. This goal, the "effect size," then drives the sample size.

For example, suppose current click-through rates are about 1.1%, and you are seeking a 10% boost to 1.21%. So we have two boxes: box A with 1.1% ones (say, 110 ones and 9,890 zeros), and box B with 1.21% ones (say, 121 ones and 9,879 zeros). For starters, let's try 300 draws from each box (this would be like 300 "impressions" for each ad). Suppose our first draw yields the following:

Box A: 3 ones
Box B: 5 ones

Right away we can see that any hypothesis test would reveal this difference (5 versus 3) to be well within the range of chance variation. This combination of sample size (n = 300 in each group) and effect size (10% difference) is too small for any hypothesis test to reliably show a difference.

So we can try increasing the sample size (let's try 2,000 impressions), and require a larger improvement (50% instead of 10%).

For example, suppose current click-through rates are still 1.1%, but we are now seeking a 50% boost to 1.65%. So we have two boxes: box A still with 1.1% ones (say, 110 ones and 9,890 zeros), and box B with 1.65% ones (say, 165 ones and 9,868 zeros). Now we'll try 2,000 draws from each box. Suppose our first draw yields the following:

Box A: 19 ones
Box B: 34 ones

A significance test on this difference (34–19) shows it still registers as "not significant" (though much closer to significance than the earlier difference of 5–3). To calculate power, we would need to repeat the previous procedure many times, or use statistical software that can calculate power, but our initial draw suggests to us that even detecting a 50% improvement will require several thousand ad impressions.

In summary, for calculating power or required sample size, there are four moving parts:

- Sample size
- Effect size you want to detect
- Significance level (alpha) at which the test will be conducted
- Power

Specify any three of them, and the fourth can be calculated. Most commonly, you would want to calculate sample size, so you must specify the other three. With *R* and *Python*, you also have to specify the alternative hypothesis as "greater" or "larger" to get a one-sided test; see "One-Way Versus Two-Way Hypothesis Tests" on page 95 for more discussion of one-way versus two-way tests. Here is *R* code for a test involving two proportions, where both samples are the same size (this uses the `pwr` package):

```
effect_size = ES.h(p1=0.0121, p2=0.011)
pwr.2p.test(h=effect_size, sig.level=0.05, power=0.8, alternative='greater')
--
     Difference of proportion power calculation for binomial distribution
                                                 (arcsine transformation)

              h = 0.01029785
              n = 116601.7
      sig.level = 0.05
```

```
effect_size = sm.stats.proportion_effectsize(0.0121, 0.011)
analysis = sm.stats.TTestIndPower()
result = analysis.solve_power(effect_size=effect_size,
                              alpha=0.05, power=0.8, alternative='larger')
print('Sample Size: %.3f' % result)
--
Sample Size: 116602.393
```

```
            power = 0.8
      alternative = greater

NOTE: same sample sizes
```

The function `ES.h` calculates the effect size. We see that if we want a power of 80%, we require a sample size of almost 120,000 impressions. If we are seeking a 50% boost (`p1=0.0165`), the sample size is reduced to 5,500 impressions.

The `statsmodels` package contains several methods for power calculation. Here, we use `proportion_effectsize` to calculate the effect size and `TTestIndPower` to solve for the sample size:

```
effect_size = sm.stats.proportion_effectsize(0.0121, 0.011)
analysis = sm.stats.TTestIndPower()
result = analysis.solve_power(effect_size=effect_size,
                              alpha=0.05, power=0.8, alternative='larger')
print('Sample Size: %.3f' % result)
--
Sample Size: 116602.393
```

Key Ideas

- Finding out how big a sample size you need requires thinking ahead to the statistical test you plan to conduct.
- You must specify the minimum size of the effect that you want to detect.
- You must also specify the required probability of detecting that effect size (power).
- Finally, you must specify the significance level (alpha) at which the test will be conducted.

Further Reading

- *Sample Size Determination and Power* by Thomas Ryan (Wiley, 2013) is a comprehensive and readable review of this subject.
- Steve Simon, a statistical consultant, has written a very engaging narrative-style post on the subject.

Weighted Regression

Weighted regression is used by statisticians for a variety of purposes; in particular, it is important for analysis of complex surveys. Data scientists may find weighted regression useful in two cases:

- Inverse-variance weighting when different observations have been measured with different precision; the higher variance ones receiving lower weights.
- Analysis of data where rows represent multiple cases; the weight variable encodes how many original observations each row represents.

Python

```
house['Year'] = [int(date.split('-')[0]) for date in house.DocumentDate]
house['Weight'] = house.Year - 2005
```

We can compute a weighted regression with the `lm` function using the `weight` argument:

```
house_wt <- lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
                 Bedrooms + BldgGrade,
               data=house, weight=Weight)
round(cbind(house_lm=house_lm$coefficients,
            house_wt=house_wt$coefficients), digits=3)
```

```
                house_lm    house_wt
(Intercept)   -521871.368 -584189.329
SqFtTotLiving     228.831     245.024
SqFtLot            -0.060      -0.292
Bathrooms       -19442.840  -26085.970
Bedrooms        -47769.955  -53608.876
BldgGrade       106106.963  115242.435
```

The coefficients in the weighted regression are slightly different from the original regression.

Most models in `scikit-learn` accept weights as the keyword argument `sample_weight` in the call of the `fit` method:

```
predictors = ['SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms', 'BldgGrade']
outcome = 'AdjSalePrice'

house_wt = LinearRegression()
house_wt.fit(house[predictors], house[outcome], sample_weight=house.Weight)
```

Key Ideas

- Multiple linear regression models the relationship between a response variable Y and multiple predictor variables $X_1, ..., X_p$.
- The most important metrics to evaluate a model are root mean squared error (RMSE) and R-squared (R^2).
- The standard error of the coefficients can be used to measure the reliability of a variable's contribution to a model.
- Stepwise regression is a way to automatically determine which variables should be included in the model.
- Weighted regression is used to give certain records more or less weight in fitting the equation.

Factor Variables in Regression

Factor variables, also termed *categorical* variables, take on a limited number of discrete values. For example, a loan purpose can be "debt consolidation," "wedding," "car," and so on. The binary (yes/no) variable, also called an *indicator* variable, is a special case of a factor variable. Regression requires numerical inputs, so factor variables need to be recoded to use in the model. The most common approach is to convert a variable into a set of binary *dummy* variables.

Key Terms for Factor Variables

Dummy variables
Binary 0–1 variables derived by recoding factor data for use in regression and other models.

Reference coding
The most common type of coding used by statisticians, in which one level of a factor is used as a reference and other factors are compared to that level.

Synonym
treatment coding

One hot encoder
A common type of coding used in the machine learning community in which all factor levels are retained. While useful for certain machine learning algorithms, this approach is not appropriate for multiple linear regression.

Deviation coding
A type of coding that compares each level against the overall mean as opposed to the reference level.

Synonym
sum contrasts

Dummy Variables Representation

In *Python*, we can convert categorical variables to dummies using the pandas method `get_dummies`:

```
pd.get_dummies(house['PropertyType']).head() ❶
pd.get_dummies(house['PropertyType'], drop_first=True).head() ❷
```

❶ By default, returns one hot encoding of the categorical variable.

❷ The keyword argument `drop_first` will return $P - 1$ columns. Use this to avoid the problem of multicollinearity.

In certain machine learning algorithms, such as nearest neighbors and tree models, one hot encoding is the standard way to represent factor variables (for example, see "Tree Models" on page 249).

In the regression setting, a factor variable with P distinct levels is usually represented by a matrix with only $P - 1$ columns. This is because a regression model typically includes an intercept term. With an intercept, once you have defined the values for $P - 1$ binaries, the value for the Pth is known and could be considered redundant. Adding the Pth column will cause a multicollinearity error (see "Multicollinearity" on page 172).

The method `get_dummies` takes the optional keyword argument `drop_first` to exclude the first factor as *reference*:

```
predictors = ['SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms',
              'BldgGrade', 'PropertyType']

X = pd.get_dummies(house[predictors], drop_first=True)

house_lm_factor = LinearRegression()
house_lm_factor.fit(X, house[outcome])

print(f'Intercept: {house_lm_factor.intercept_:.3f}')
print('Coefficients:')
for name, coef in zip(X.columns, house_lm_factor.coef_):
    print(f' {name}: {coef}')
```

The output from the *R* regression shows two coefficients corresponding to `PropertyType`: `PropertyTypeSingle Family` and `PropertyTypeTownhouse`. There is no coefficient of `Multiplex` since it is implicitly defined when `PropertyTypeSingle Family == 0` and `PropertyTypeTownhouse == 0`. The coefficients are interpreted as relative to `Multiplex`, so a home that is `Single Family` is worth almost $85,000 less, and a home that is `Townhouse` is worth over $150,000 less.[4]

chapter 5 classficiation

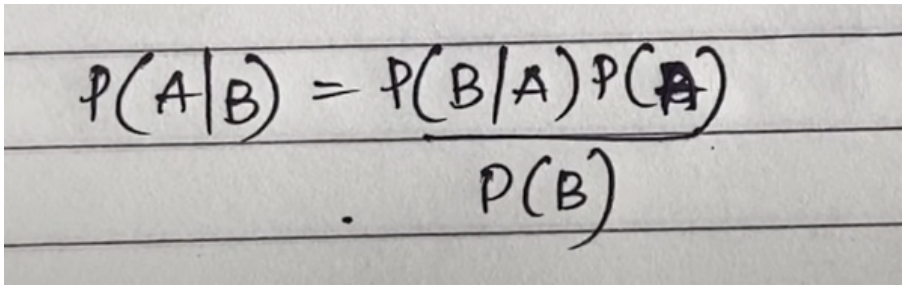
data science scientist are often taken the automatic decision for the business problems is email attacking are fishing it is a customer like this journey is a web user likely click on advertisement this is all using the classifications

regression is used basically continue variables like numbers 123456 and sone classification is classification is most of the times deals with categorical variables that converts into dummy variables like one and zero one oh it's fishing or not fishing click are not click most of the times used in the statistics classification probability theorems the probability theorem is there fishing or not fishing is a malware the spam mail not mail and so on

Naive Bayes

naive bayes classification based on bayes theorem is used one is given and we find the output another one output is called bayes therom

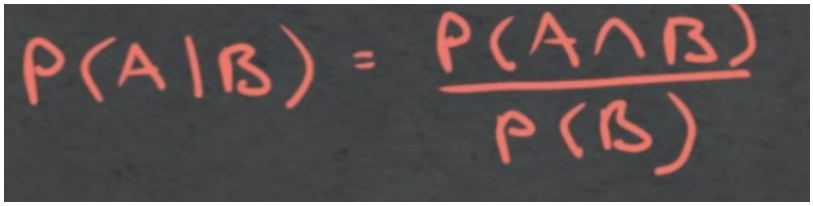
there is no dependency on the other future means every variable as a independent on its own



bayes theorem formula

Conditional probability

condition property means p(A|B)



When a predictor category is absent in the training data, the algo- rithm assigns zero probability to the outcome variable in new data, rather than simply ignoring this variable and using the information from other variables, as other methods might. Most implementa- tions of Naive Bayes use a smoothing parameter (Laplace Smooth- ing) to prevent thiss

Gaussian Maximum Likelihood Classification: Assume each class has a Gaussian distribution,

estimate the distributions from the data, then classify new observations to the class with the maximum likelihood.

lda and qda

Discriminant Analysis basically use dimension redaction

What is its role in a discriminant analysis?

Discriminant function analysis is **used to determine which variables discriminate between two or more naturally occurring groups**. ... Discriminant Analysis could then be used to determine which variable(s) are the best predictors of students' subsequent educational choice.

What is discriminant analysis in research methodology?

Discriminant analysis is a **versatile statistical method often used by market researchers to classify observations into two or more groups or categories**. In other words, discriminant analysis is used to assign objects to one group among a number of known groups.

What is multiple discriminant analysis give some examples?

For example, three brands of computers, Computer A, Computer B and Computer C can be the categorical dependent variable. ... **If the dependent variable has three or more than three categories**, then the type used is multiple discriminant analysis.

LDA AND PSA

lda purpose provide separation boundary between them

PSA purpose capture maximum variance

lda used in single access basically is used in single axis what is very difficult to understand more than 5 or 6 dimensions variables at the time is used to PSA lda means linear discriminant analysis

Linear discriminant analysis should not be confused with Latent Dirichlet Allocation, also referred to as LDA. Latent DirichletAllo- cation is used in text and natural language processing and is unre- lated to linear discriminant analysis.

Covariance Matrix

To understand discriminant analysis, it is first necessary to introduce the concept of covariance between two or more variables.

The covariance measures the relationship between two variables x and z. Denote the mean for each variable by x and z (see “Mean” on page 9). The covariance sx, z between x and z is given by:

$$s_{x,z} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(z_i - \bar{z})}{n-1}$$

positive values indicate a positive relationship and negative values indicate a negative relationship. Correlation, however, is constrained to be between –1 and 1, whereas covariance scale depends on the scale of the variables x and z

Fisher's Linear Discriminant

For simplicity, let's focus on a classification problem in which we want to predict a binary outcome y using just two continuous numeric variables (x, z). Technically, discriminant analysis assumes the predictor variables are normally distributed continuous variables, but, in practice, the method works well even for nonextreme departures from normality, and for binary predictors. Fisher's linear discriminant distinguishes variation *between* groups, on the one hand, from variation *within* groups on the other. Specifically, seeking to divide the records into two groups, linear discriminant analysis (LDA) focuses on maximizing the "between" sum of squares $SS_{between}$ (measuring the variation between the two groups) relative to the "within" sum of squares SS_{within} (measuring the within-group variation). In this case, the two groups correspond to the records (x_0, z_0) for which $y = 0$ and the records (x_1, z_1) for which $y = 1$. The method finds the linear combination $w_x x + w_z z$ that maximizes that sum of squares ratio:

$$\frac{SS_{between}}{SS_{within}}$$

The between sum of squares is the squared distance between the two group means, and the within sum of squares is the spread around the means within each group, weighted by the covariance matrix. Intuitively, by maximizing the between sum of

https://www.youtube.com/watch?v=2EjGJXzsC2U

Key Ideas

- Discriminant analysis works with continuous or categorical predictors, as well as with categorical outcomes.
- Using the covariance matrix, it calculates a *linear discriminant function*, which is used to distinguish records belonging to one class from those belonging to another.
- This function is applied to the records to derive weights, or scores, for each record (one weight for each possible class), which determines its estimated class.

Logistic Regression

Logistic regression is analogous to multiple linear regression (see Chapter 4), except the outcome is binary. Various transformations are employed to convert the problem to one in which a linear model can be fit

Key Terms for Logistic Regression

Logit

The function that maps class membership probability to a range from ± ∞ (instead of 0 to 1).

Synonym

Log odds (see below)

Odds

The ratio of "success" (1) to "not success" (0).

Log odds

The response in the transformed model (now linear), which gets mapped back to a probability.

logistic regression basically used in zeros and ones or a same as multi linear regression used take multiple values and its depends upon output are used as used as track the variables . vision based on s shape what's the time use the log or same likely the functions like like up and down curves

$$p = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q$$

this is used in deep learning

$$p = \frac{1}{1 + e^{-\left(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q\right)}}$$

This transform ensures that the p stays between 0 and 1.

To get the exponential expression out of the denominator, we consider *odds* inste of probabilities. Odds, familiar to bettors everywhere, are the ratio of "successes" to "nonsuccesses" (0). In terms of probabilities, odds are the probability of an eve divided by the probability that the event will not occur. For example, if the probabil that a horse will win is 0.5, the probability of "won't win" is (1 – 0.5) = 0.5, and t odds are 1.0:

$$\text{Odds}(Y = 1) = \frac{p}{1 - p}$$

We can obtain the probability from the odds using the inverse odds function:

$$p = \frac{\text{Odds}}{1 + \text{Odds}}$$

We combine this with the logistic response function, shown earlier, to get:

$$\text{Odds}(Y = 1) = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q}$$

$$\log\left(\text{Odds}(Y = 1)\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q$$

Logistic Regression and the GLM

In *Python*, we use the `scikit-learn` class `LogisticRegression` from `sklearn.linear_model`. The arguments `penalty` and `C` are used to prevent overfitting by L1 or L2 regularization. Regularization is switched on by default. In order to fit without regularization, we set `C` to a very large value. The `solver` argument selects the used minimizer; the method `liblinear` is the default:

```
predictors = ['payment_inc_ratio', 'purpose_', 'home_', 'emp_len_',
              'borrower_score']
outcome = 'outcome'
X = pd.get_dummies(loan_data[predictors], prefix='', prefix_sep='',
                   drop_first=True)
y = loan_data[outcome]

logit_reg = LogisticRegression(penalty='l2', C=1e42, solver='liblinear')
logit_reg.fit(X, y)
```

Generalized Linear Models

Generalized linear models (GLMs) are characterized by two main components:

- A probability distribution or family (binomial in the case of logistic regression)
- A link function—i.e., a transformation function that maps the response to the predictors (logit in the case of logistic regression)

Logistic regression is by far the most common form of GLM. A data scientist will encounter other types of GLMs. Sometimes a log link function is used instead of the logit; in practice, use of a log link is unlikely to lead to very different results for most applications. The Poisson distribution is commonly used to model count data (e.g., the number of times a user visits a web page in a certain amount of time). Other families include negative binomial and gamma, often used to model elapsed time (e.g., time to failure). In contrast to logistic regression, application of GLMs with these models is more nuanced and involves greater care. These are best avoided unless you are familiar with and understand the utility and pitfalls of these methods.

Predicted Values from Logistic Regression build

predict output using logistic regression

$$\hat{p} = \frac{1}{1 + e^{-\hat{Y}}}$$

In *Python*, we can convert the probabilities into a data frame and use the `describe` method to get these characteristics of the distribution:

```
pred = pd.DataFrame(logit_reg.predict_log_proba(X),
                    columns=loan_data[outcome].cat.categories)
pred.describe()
```

The probabilities are directly available using the `predict_proba` methods in `scikit-learn`:

```
pred = pd.DataFrame(logit_reg.predict_proba(X),
                    columns=loan_data[outcome].cat.categories)
pred.describe()
```

These are on a scale from 0 to 1 and don't yet declare whether the predicted value is default or paid off. We could declare any value greater than 0.5 as default. In practice, a lower cutoff is often appropriate if the goal is to identify members of a rare class

Interpreting the Coefficients and Odds Ratios

odd values is finding by calculating using sum of yes values and sum of no values

odd ration is a/b = first 0dd /second odd

find the correct are not using 1) ci-square test , 2) fisher-exact test 3)the wald test

$$\text{odds ratio} = \frac{\text{Odds}(Y = 1 \mid X = 1)}{\text{Odds}(Y = 1 \mid X = 0)}$$

https://www.youtube.com/watch?v=8nm0G-1uJzA

Fitting the model

and the model must be fit using maximum likelihood estimation (MLE).Maximum likelihood estimation is a process that tries to find the model that is most likely to have produced the data we see. The MLE finds the solution such that the estimated log odds best describes the observed outcome.

The mechanics of the algorithm involve a quasiNewton optimization that iterates between a scoring step (Fisher's scoring), based on the current parameters, and an update to the parameters to improve the fit.

The package `statsmodels` has an implementation for generalized linear model (`GLM`) that provides similarly detailed information:

```
y_numbers = [1 if yi == 'default' else 0 for yi in y]
logit_reg_sm = sm.GLM(y_numbers, X.assign(const=1),
                      family=sm.families.Binomial())
logit_result = logit_reg_sm.fit()
logit_result.summary()
```

Many other concepts for linear regression carry over to the logistic regression setting (and other GLMs). For example, you can use stepwise regression, fit interaction terms, or include spline terms. The same concerns regarding confounding and correlated variables apply to logistic regression (see "Interpreting the Regression Equation"

The formula interface of `statsmodels` also supports these extensions in *Python*:

```
import statsmodels.formula.api as smf
formula = ('outcome ~ bs(payment_inc_ratio, df=4) + purpose_ + ' +
           'home_ + emp_len_ + bs(borrower_score, df=4)')
model = smf.glm(formula=formula, data=loan_data, family=sm.families.Binomial())
results = model.fit()
```

Analysis of residuals

```
ggplot(df, aes(x=payment_inc_ratio, y=partial_resid, solid = FALSE)) +
  geom_point(shape=46, alpha=0.4) +
  geom_line(aes(x=payment_inc_ratio, y=terms),
            color='red', alpha=0.5, size=1.5) +
  labs(y='Partial Residual')
```

The resulting plot is displayed in Figure 5-4. The estimated fit, shown by the line, goes between two sets of point clouds. The top cloud corresponds to a response of 1 (defaulted loans), and the bottom cloud corresponds to a response of 0 (loans paid off). This is very typical of residuals from a logistic regression since the output is binary. The prediction is measured as the logit (log of the odds ratio), which will always be some finite value. The actual value, an absolute 0 or 1, corresponds to an infinite logit, either positive or negative, so the residuals (which get added to the fitted value) will never equal 0. Hence the plotted points lie in clouds either above or below the fitted line in the partial residual plot. Partial residuals in logistic regression, while less valuable than in regression, are still useful to confirm nonlinear behavior and identify highly influential records.

There is currently no implementation of partial residuals in any of the major *Python* packages. We provide *Python* code to create the partial residual plot in the accompanying source code repository.

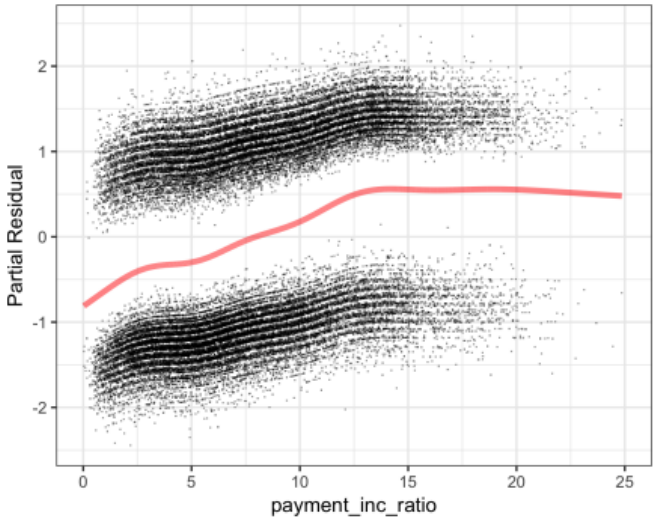


Figure 5-4. Partial residuals from logistic regression

Key Ideas

- Logistic regression is like linear regression, except that the outcome is a binary variable.
- Several transformations are needed to get the model into a form that can be fit as a linear model, with the log of the odds ratio as the response variable.
- After the linear model is fit (by an iterative process), the log odds is mapped back to a probability.
- Logistic regression is popular because it is computationally fast and produces a model that can be scored to new data with only a few arithmetic operations.

Evaluating Classification Models

It is common in predictive modeling to train a number of different models, apply each to a holdout sample, and assess their performance

this evolution of classification modules use show the procedures of most accurate and useful product models

which model produces the most accurate and useful predictions

Key Terms for Evaluating Classification Models

Accuracy
The percent (or proportion) of cases classified correctly.

Confusion matrix
A tabular display (2×2 in the binary case) of the record counts by their predicted and actual classification status.

Sensitivity
The percent (or proportion) of all 1s that are correctly classified as 1s.

Synonym
Recall

Specificity
The percent (or proportion) of all 0s that are correctly classified as 0s.

Precision
The percent (proportion) of predicted 1s that are actually 1s.

ROC curve
A plot of sensitivity versus specificity.

Lift
A measure of how effective the model is at identifying (comparatively rare) 1s at different probability cutoffs.

Accuracy is simply a measure of total error:

$$\text{accuracy} = \frac{\sum \text{TruePositive} + \sum \text{TrueNegative}}{\text{SampleSize}}$$

In most classification algorithms, each case is assigned an “estimated probability of being a 1.”3 The default decision point, or cutoff, is typically 0.50 or 50%. If the proba- bility is above 0.5, the classification is “1”; otherwise it is “0.” An alternative default cutoff is the prevalent probability of 1s in the data.

Confusion Matrix

At the heart of classification metrics is the confusion matrix. The confusion matrix is a table showing the number of correct and incorrect predictions categorized by type of response.

To illustrate the confusion matrix, consider the logistic_gam model that was trained on a balanced data set with an equal number of defaulted and paid-off loans

Following the usual conventions, Y = 1 corresponds to the event of inter- est (e.g., default), and Y = 0 corresponds to a negative (or usual) event (e.g., paid off). The following computes the confusion matrix for the logistic_gam model applied to

In *Python*:

```
pred = logit_reg.predict(X)
pred_y = logit_reg.predict(X) == 'default'
true_y = y == 'default'
true_pos = true_y & pred_y
true_neg = ~true_y & ~pred_y
false_pos = ~true_y & pred_y
false_neg = true_y & ~pred_y
```

3 Not all methods provide unbiased estimates of probability. In most cases, it is sufficient that the method provide a ranking equivalent to the rankings that would result from an unbiased probability estimate; the cutoff method is then functionally equivalent.


```
conf_mat = pd.DataFrame([[np.sum(true_pos), np.sum(false_neg)],
                         [np.sum(false_pos), np.sum(true_neg)]],
                        index=['Y = default', 'Y = paid off'],
                        columns=['Yhat = default', 'Yhat = paid off'])
conf_mat
```

The predicted outcomes are columns and the true outcomes are the rows. The diagonal elements of the matrix show the number of correct predictions, and the off diagonal elements show the number of incorrect predictions. For example, 14,295 defaulted loans were correctly predicted as a default, but 8,376 defaulted loans were incorrectly predicted as paid off.

Precision, Recall, and Specificity

$$\text{precision} = \frac{\sum \text{TruePositive}}{\sum \text{TruePositive} + \sum \text{FalsePositive}}$$

$$\text{recall} = \frac{\sum \text{TruePositive}}{\sum \text{TruePositive} + \sum \text{FalseNegative}}$$

$$\text{specificity} = \frac{\sum \text{TrueNegative}}{\sum \text{TrueNegative} + \sum \text{FalsePositive}}$$

```
conf_mat[2, 2] / sum(conf_mat[2,])
```

Here is the equivalent code to calculate the metrics in *Python*:

```
conf_mat = confusion_matrix(y, logit_reg.predict(X))
print('Precision', conf_mat[0, 0] / sum(conf_mat[:, 0]))
print('Recall', conf_mat[0, 0] / sum(conf_mat[0, :]))
print('Specificity', conf_mat[1, 1] / sum(conf_mat[1, :]))

precision_recall_fscore_support(y, logit_reg.predict(X),
                                labels=['default', 'paid off'])
```

`scikit-learn` has a custom method `precision_recall_fscore_support` that calculates precision and recall/specificity all at once.

ROC Curve

we will take threshold values

You can see that there is a trade-off between recall and specificity. Capturing more 1s generally means misclassifying more 0s as 1s. The ideal classifier would do an excel- lent job of classifying the 1s, without misclassifying more 0s as 1s.

The metric that captures this trade-off is the “Receiver Operating Characteristics” curve, usually referred to as the ROC curve. The ROC curve plots recall (sensitivity) on the y-axis against specificity on the x-axis.4 The ROC curve shows the trade-off between recall and specificity as you change the cutoff to determine how to classify a record. Sensitivity (recall) is plotted on the y-axis, and you may encounter two forms in which the x-axis is labeled

The curve looks identical whichever way it is done. The process to compute the ROC curve is:

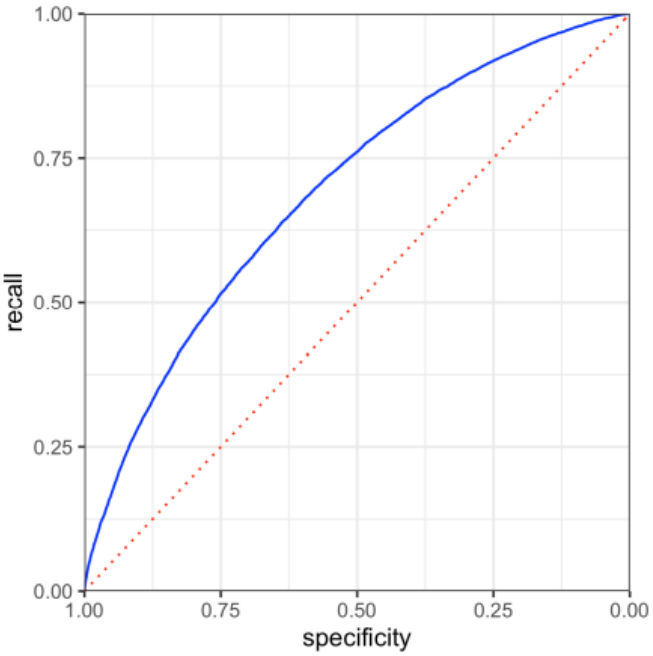
1. Sort the records by the predicted probability of being a 1, starting with the most probable and ending with the least probable.
2. Compute the cumulative specificity and recall based on the sorted records.

```
fpr, tpr, thresholds = roc_curve(y, logit_reg.predict_proba(X)[:,0],
                                 pos_label='default')
roc_df = pd.DataFrame({'recall': tpr, 'specificity': 1 - fpr})

ax = roc_df.plot(x='specificity', y='recall', figsize=(4, 4), legend=False)
ax.set_ylim(0, 1)
ax.set_xlim(1, 0)
ax.plot((1, 0), (0, 1))
ax.set_xlabel('specificity')
ax.set_ylabel('recall')
```

Precision-Recall Curve

In addition to ROC curves, it can be illuminating to examine the precision-recall (PR) curve. PR curves are computed in a similar way except that the data is ordered from least to most probable and cumulative precision and recall statistics are computed. PR curves are especially useful in evaluating data with highly unbalanced outcomes.



AUC

The ROC curve is a valuable graphical tool, but by itself doesn't constitute a single measure for the performance of a classifier. The ROC curve can be used, however, to produce the area underneath the curve (AUC) metric. AUC is simply the total area under the ROC curve. The larger the value of AUC, the more effective the classifier. An AUC of 1 indicates a perfect classifier: it gets all the 1s correctly classified, and it doesn't misclassify any 0s as 1s

In *Python*, we can either calculate the accuracy as shown for *R* or use `scikit-learn`'s function `sklearn.metrics.roc_auc_score`. You will need to provide the expected value as 0 or 1:

```
print(np.sum(roc_df.recall[:-1] * np.diff(1 - roc_df.specificity)))
print(roc_auc_score([1 if yi == 'default' else 0 for yi in y],
                    logit_reg.predict_proba(X)[:, 0]))
```

The model has an AUC of about 0.69, corresponding to a relatively weak classifier.

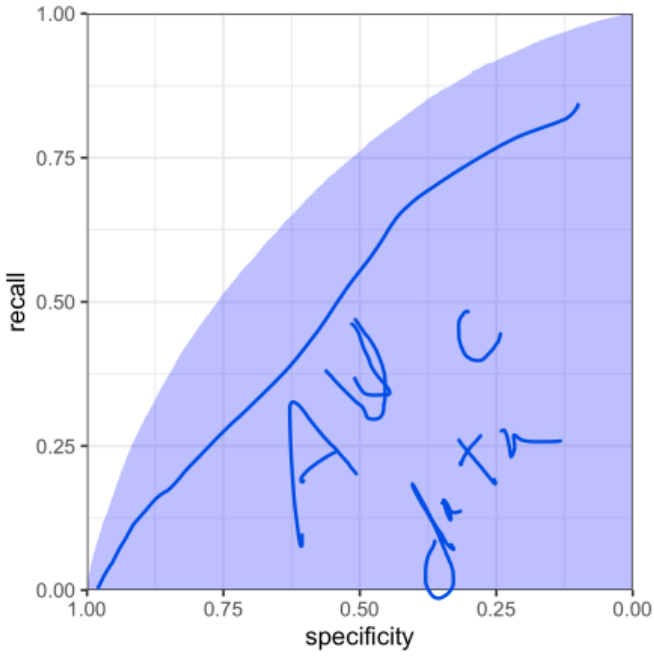


Figure 5-7. Area under the ROC curve for the loan data

Lift

Using the AUC as a metric to evaluate a model is an improvement over simple accuracy, as it can assess how well a classifier handles the trade-off between overall accuracy and the need to identify the more important 1s. But it does not completely address the rare-case problem, where you need to lower the model's probability cutoff below 0.5 to avoid having all records classified as 0. In such cases, for a record to be classified as a 1, it might be sufficient to have a probability of 0.4, 0.3, or lower. In effect, we end up overidentifying 1s, reflecting their greater importance.

Changing this cutoff will improve your chances of catching the 1s (at the cost of misclassifying more 0s as 1s). But what is the optimum cutoff?

The concept of lift lets you defer answering that question. Instead, you consider the records in order of their predicted probability of being 1s. Say, of the top 10% classified as 1s, how much better did the algorithm do, compared to the benchmark of simply picking blindly? If you can get 0.3% response in this top decile instead of the 0.1% you get overall by picking randomly, the algorithm is said to have a *lift* (also called *gains*) of 3 in the top decile. A lift chart (gains chart) quantifies this over the range of the data. It can be produced decile by decile, or continuously over the range of the data.

To compute a lift chart, you first produce a *cumulative gains chart* that shows the recall on the y-axis and the total number of records on the x-axis. The *lift curve* is the ratio of the cumulative gains to the diagonal line corresponding to random selection. *Decile gains charts* are one of the oldest techniques in predictive modeling, dating from the days before internet commerce. They were particularly popular among direct mail professionals. Direct mail is an expensive method of advertising if applied indiscriminately, and advertisers used predictive models (quite simple ones, in the early days) to identify the potential customers with the likeliest prospect of payoff.

Uplift

Sometimes the term *uplift* is used to mean the same thing as lift. An alternate meaning is used in a more restrictive setting, when an A/B test has been conducted and the treatment (A or B) is then used as a predictor variable in a predictive model. The uplift is the improvement in response predicted *for an individual case* with treatment A versus treatment B. This is determined by scoring the individual case first with the predictor set to A, and then again with the predictor toggled to B. Marketers and political campaign consultants use this method to determine which of two messaging treatments should be used with which customers or voters.

Key Ideas

- Accuracy (the percent of predicted classifications that are correct) is but a first step in evaluating a model.
- Other metrics (recall, specificity, precision) focus on more specific performance characteristics (e.g., recall measures how good a model is at correctly identifying 1s).
- AUC (area under the ROC curve) is a common metric for the ability of a model to distinguish 1s from 0s.
- Similarly, lift measures how effective a model is in identifying the 1s, and it is often calculated decile by decile, starting with the most probable 1s.

https://www.youtube.com/watch?v=A_ZKMsZ3f3o

Strategies for Imbalanced Data

The previous section dealt with evaluation of classification models using metrics that go beyond simple accuracy and are suitable for imbalanced data—data in which the outcome of interest (purchase on a website, insurance fraud, etc.) is rare. In this section, we look at additional strategies that can improve predictive modeling perfor- mance with imbalanced data.

Key Terms for Imbalanced Data

Undersample
Use fewer of the prevalent class records in the classification model.

Synonym
Downsample

Oversample
Use more of the rare class records in the classification model, bootstrapping if necessary.

Synonym
Upsample

Up weight or down weight
Attach more (or less) weight to the rare (or prevalent) class in the model.

Data generation
Like bootstrapping, except each new bootstrapped record is slightly different from its source.

z-score
The value that results after standardization.

K
The number of neighbors considered in the nearest neighbor calculation.

Undersampling basically use both data are not equal in machine learning classification algorithms suppose her take it to variables like this is to the this is more than 2,00,000 and no is less than 5000 this is where basically under sampling bye no then what we do OK take the data what was the two lakh data we take the randomly 500 and stately randomly 500 that it will be called does this is a solution of under sampling

Undersampling

If you have enough data, as is the case with the loan data, one solution is to *undersample* (or downsample) the prevalent class, so the data to be modeled is more balanced between 0s and 1s. The basic idea in undersampling is that the data for the dominant class has many redundant records. Dealing with a smaller, more balanced data set yields benefits in model performance, and it makes it easier to prepare the data and to explore and pilot models.

How much data is enough? It depends on the application, but in general, having tens of thousands of records for the less dominant class is enough. The more easily distinguishable the 1s are from the 0s, the less data needed.

The loan data analyzed in "Logistic Regression" on page 208 was based on a balanced training set: half of the loans were paid off, and the other half were in default. The predicted values were similar: half of the probabilities were less than 0.5, and half were greater than 0.5. In the full data set, only about 19% of the loans were in default, as shown in *R*:

```
mean(full_train_set$outcome=='default')
[1] 0.1889455
```

In *Python*:

```
print('percentage of loans in default: ',
      100 * np.mean(full_train_set.outcome == 'default'))
```

What happens if we use the full data set to train the model? Let's see what this looks like in *R*:

```
full_model <- glm(outcome ~ payment_inc_ratio + purpose_ + home_ +
                            emp_len_+ dti + revol_bal + revol_util,
                  data=full_train_set, family='binomial')
pred <- predict(full_model)
mean(pred > 0)
[1] 0.003942094
```

And in *Python*:

```
predictors = ['payment_inc_ratio', 'purpose_', 'home_', 'emp_len_',
              'dti', 'revol_bal', 'revol_util']
outcome = 'outcome'
X = pd.get_dummies(full_train_set[predictors], prefix='', prefix_sep='',
                   drop_first=True)
y = full_train_set[outcome]

full_model = LogisticRegression(penalty='l2', C=1e42, solver='liblinear')
full_model.fit(X, y)
print('percentage of loans predicted to default: ',
      100 * np.mean(full_model.predict(X) == 'default'))
```

Oversampling and Up/Down Weighting

One criticism of the undersampling method is that it throws away data and is not using all the information at hand. If you have a relatively small data set, and the rarer class contains a few hundred or a few thousand records, then undersampling the dominant class has the risk of throwing out useful information. In this case, instead of downsampling the dominant case, you should oversample (upsample) the rarer class by drawing additional rows with replacement (bootstrapping).

You can achieve a similar effect by weighting the data. Many classification algorithms take a weight argument that will allow you to up/down weight the data. For example, apply a weight vector to the loan data using the `weight` argument to `glm` in *R*:

```
wt <- ifelse(full_train_set$outcome=='default',
             1 / mean(full_train_set$outcome == 'default'), 1)
full_model <- glm(outcome ~ payment_inc_ratio + purpose_ + home_ +
                            emp_len_+ dti + revol_bal + revol_util,
                  data=full_train_set, weight=wt, family='quasibinomial')
pred <- predict(full_model)
mean(pred > 0)
[1] 0.5767208
```

Most `scikit-learn` methods allow specifying weights in the `fit` function using the keyword argument `sample_weight`:

```
default_wt = 1 / np.mean(full_train_set.outcome == 'default')
wt = [default_wt if outcome == 'default' else 1
      for outcome in full_train_set.outcome]

full_model = LogisticRegression(penalty="l2", C=1e42, solver='liblinear')
full_model.fit(X, y, sample_weight=wt)
print('percentage of loans predicted to default (weighting): ',
      100 * np.mean(full_model.predict(X) == 'default'))
```

The weights for loans that default are set to $\frac{1}{p}$, where p is the probability of default. The nondefaulting loans have a weight of 1. The sums of the weights for the default-

if we have less sample data increase the data using bootstrap method

Data Generation

Data Generation

A variation of upsampling via bootstrapping (see "Oversampling and Up/Down Weighting" on page 232) is *data generation* by perturbing existing records to create new records. The intuition behind this idea is that since we observe only a limited set of instances, the algorithm doesn't have a rich set of information to build classification "rules." By creating new records that are similar but not identical to existing records, the algorithm has a chance to learn a more robust set of rules. This notion is similar in spirit to ensemble statistical models such as boosting and bagging (see Chapter 6).

The idea gained traction with the publication of the *SMOTE* algorithm, which stands for "Synthetic Minority Oversampling Technique." The SMOTE algorithm finds a record that is similar to the record being upsampled (see "K-Nearest Neighbors" on page 238) and creates a synthetic record that is a randomly weighted average of the original record and the neighboring record, where the weight is generated separately for each predictor. The number of synthetic oversampled records created depends on the oversampling ratio required to bring the data set into approximate balance with respect to outcome classes.

There are several implementations of SMOTE in *R*. The most comprehensive package for handling unbalanced data is `unbalanced`. It offers a variety of techniques, including a "Racing" algorithm to select the best method. However, the SMOTE algorithm is simple enough that it can be implemented directly in *R* using the `FNN` package.

The *Python* package `imbalanced-learn` implements a variety of methods with an API that is compatible with `scikit-learn`. It provides various methods for over- and undersampling and support for using these techniques with boosting and bagging classifiers.

Cost-Based Classification

In practice, accuracy and AUC are a poor man's way to choose a classification rule. Often, an estimated cost can be assigned to false positives versus false negatives, and it is more appropriate to incorporate these costs to determine the best cutoff when classifying 1s and 0s. For example, suppose the expected cost of a default of a new loan is C and the expected return from a paid-off loan is R. Then the expected return for that loan is:

$$\text{expected return} = P(Y = 0) \times R + P(Y = 1) \times C$$

Instead of simply labeling a loan as default or paid off, or determining the probability of default, it makes more sense to determine if the loan has a positive expected return. Predicted probability of default is an intermediate step, and it must be combined with the loan's total value to determine expected profit, which is the ultimate planning metric of business. For example, a smaller value loan might be passed over in favor of a larger one with a slightly higher predicted default probability.

Exploring the Predictions

A single metric, such as AUC, cannot evaluate all aspects of the suitability of a model for a situation. Figure 5-8 displays the decision rules for four different models fit to the loan data using just two predictor variables: `borrower_score` and `payment_inc_ratio`. The models are linear discriminant analysis (LDA), logistic linear regression, logistic regression fit using a generalized additive model (GAM), and a tree model (see "Tree Models" on page 249). The region to the upper left of the lines corresponds to a predicted default. LDA and logistic linear regression give nearly identical results in this case. The tree model produces the least regular rule, with two steps. Finally, the GAM fit of the logistic regression represents a compromise between the tree model and the linear model.

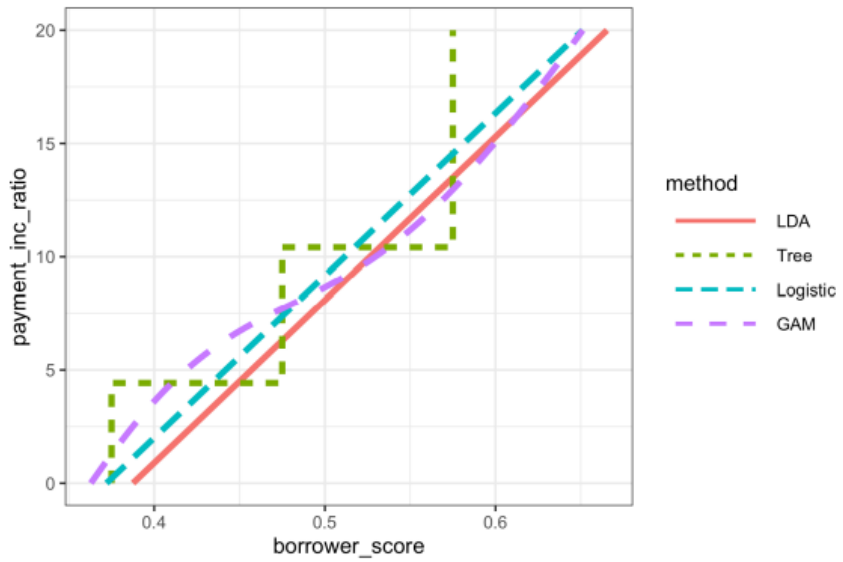


Figure 5-8. Comparison of the classification rules for four different methods

It is not easy to visualize the prediction rules in higher dimensions or, in the case of the GAM and the tree model, even to generate the regions for such rules.

In any case, exploratory analysis of predicted values is always warranted.

Key Ideas

- Highly imbalanced data (i.e., where the interesting outcomes, the 1s, are rare) are problematic for classification algorithms.
- One strategy for working with imbalanced data is to balance the training data via undersampling the abundant case (or oversampling the rare case).
- If using all the 1s still leaves you with too few 1s, you can bootstrap the rare cases, or use SMOTE to create synthetic data similar to existing rare cases.
- Imbalanced data usually indicates that correctly classifying one class (the 1s) has higher value, and that value ratio should be built into the assessment metric.