ICQB

Introduction to Computational & Quantitative Biology (G4120)
Fall 2022

Oliver Jovanovic, Ph.D.
Columbia University
Department of Microbiology & Immunology

Lecture 7: Introduction to Statistics
October 25, 2022

Statistics and Information Theory

$$\mathbf{S = k \log W}$$

“Entropy is missing information.”
– Ludwig Boltzmann

Statistics, Probability and Entropy

Entropy is how much information you are missing about an event.

Probability predicts the likelihood of future events.

Statistics is the analysis of the frequency of past events in data.

Probability and Statistics

The Chevalier de Méré (1607-1684) was a French gentleman gambler who noticed that he was losing money on a particular bet. He noticed that he was losing money on even odds on getting at least a double six on 24 rolls of a pair of dice. The Chevalier, an amateur mathematician, thought that based on his calculations, that particular bet had at least a 50% chance of success.

The Chevalier asked his friend Blaise Pascal, a renowned mathematician, to help him find an explanation. Pascal worked with the lawyer and mathematician Pierre de Fermat in a series of letters to calculate the true probability of that particular bet, and showed mathematically that **P (at least one double six in 24 rolls of a pair of dice) = 0.4914.**

The Chevalier had actually noticed this minuscule discrepancy from playing so many hands of dice. Pascal and Fermat's letters laid the foundation for the modern theory of probability and statistics.

Source: The Science of Conjecture: Evidence and Probability Before Pascal by James Franklin

Elements of Probability

Experiment
A set of actions whose results cannot be predicted with certainty, but will produce data. For example, roll a red six sided die and a green six sided die and record the values on the top side of each die.

Outcome
The data obtained as the result of an experiment. For example, you got a red one and a green six.

Sample Space
The set of all simple outcomes of an experiment. For example, all 36 possible outcomes or rolling the dice.

Source: Probability and Statistics by Peggy Strait

Random Variables

Discrete Random Variables
A random value from a discrete sample space. For example, a value of one from the roll of a six sided die.

Continuous Random Variables
A random value from a continuous sample space. For example, the distance from the target marker that an arrow shot at the marker landed.

Event
A random variable takes a specific value or set of values. For example, where (X, Y, Z, ...) denotes a random variable and (x, y, z, ...) denotes a specific value of a variable, X = x.

Sample
A series of observations of a random variable. For example, values of one, six and four from three rolls of a six sided die.

Probability and Distribution of Discrete Variables

Probability
The likelihood of an event to occur. P(X = x)

Probability Mass Function of a Discrete Variable (pmf)
The probability that a random variable will take on each possible value. Also known as a probability distribution. f(x) = P(X = x)

Discrete Probability Distributions
Each type of probability function defines a type of discrete distribution. Frequently used probability functions include uniform, binomial, hypergeometric and Poisson distributions. They are commonly visualized as probability histograms or bar charts.

Binomial Distribution

Binomial Distribution

If you do an experiment where you flip a coin ten times, the number of heads you get follows the binomial distribution B(10, 0.5) where B(n, p). The more flips of the coin, the closer the value gets to exactly 50% of the flips.

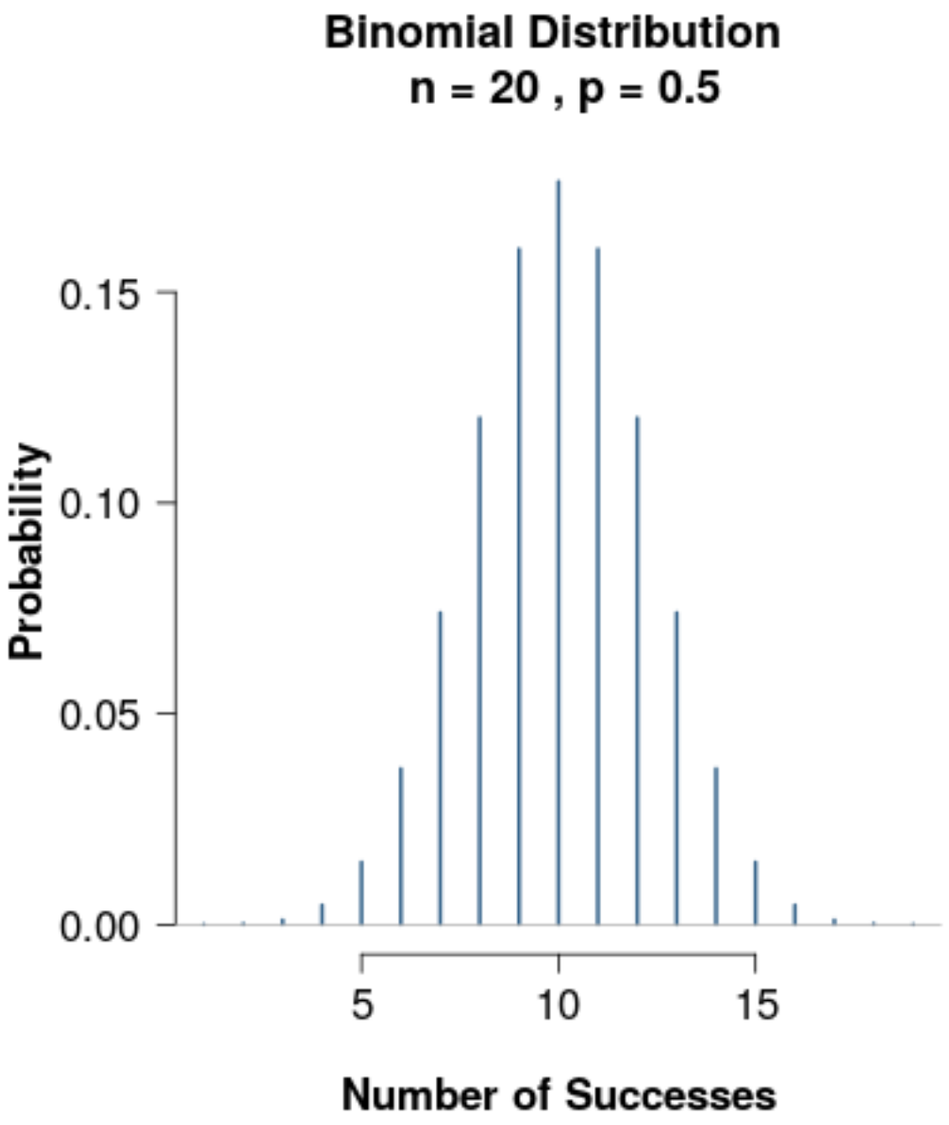
Bernoulli Trials and Distributions

A single experiment is also called a Bernoulli trial and for a single trial, n = 1, the binomial distribution is a Bernoulli distribution.

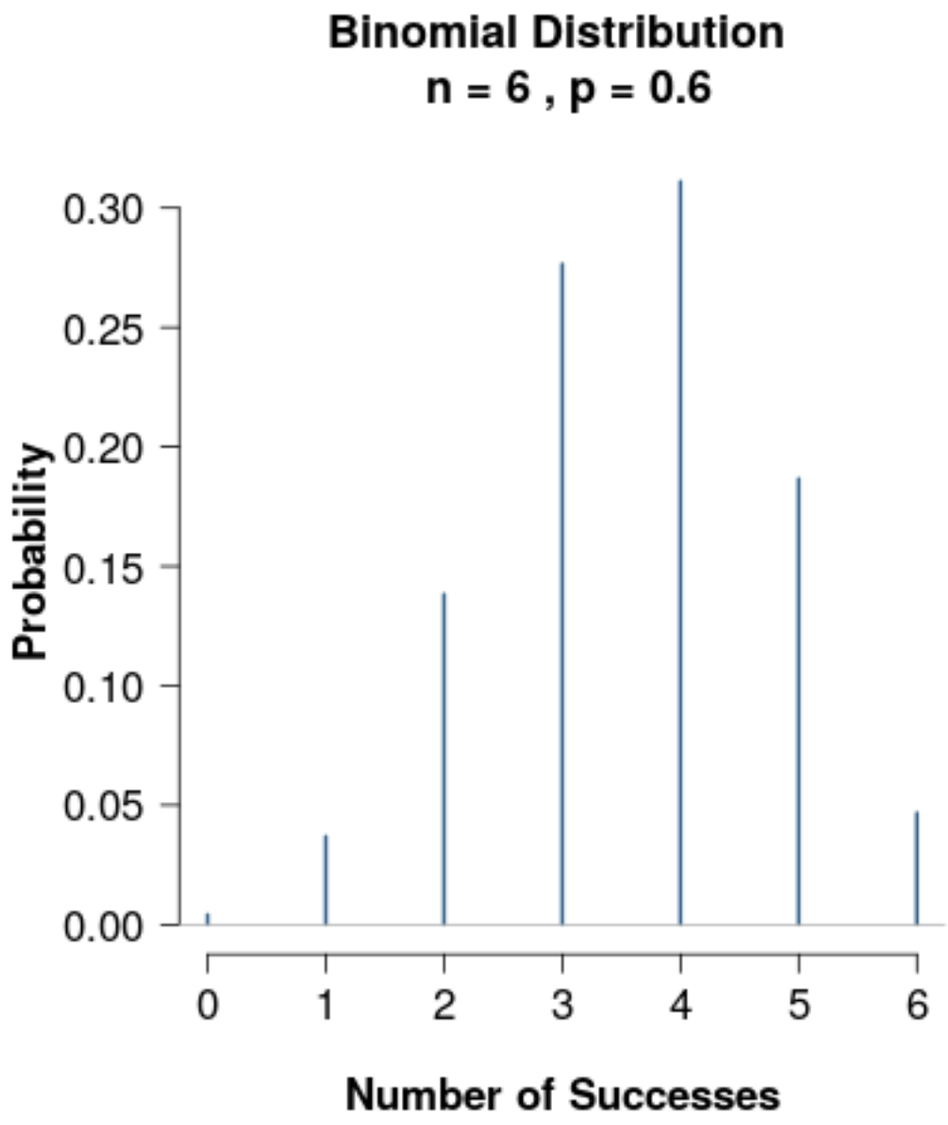
Experiments with Replacement

This can be also used for experiments with replacement, so if you randomly draw six balls from a box with 12 white and 8 black balls in it, but replace the ball drawn each time, the number of white balls follows the binomial distribution B(6, 12/20).

Examples of Binomial Distributions



Number of heads from 20 flips of a fair coin



Number of white balls drawn from 6 draws with replacement from 12 white and 8 black balls

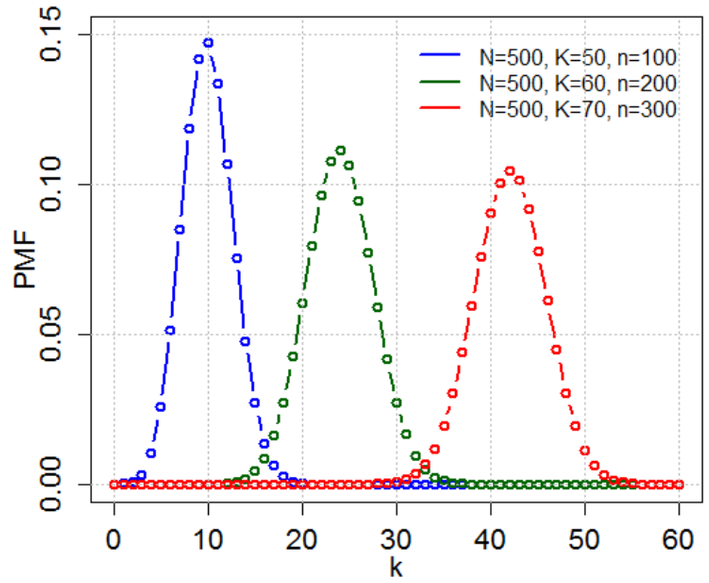
Hypergeometric Distribution

Experiments without Replacement

If you do an experiment without replacement, for example randomly draw six balls without replacement from a box with 12 white and eight black balls, the number of white balls follows the hypergeometric distribution H(6, 12, 20).

Examples of Hypergeometric Distributions

If N is the total number of genes, n is the number of genes in a pathway, and K is the number of differentially expressed genes, the number of differentially expressed genes in the pathway follows the hypergeometric distribution H(K, n, N).



Source: https://commons.wikimedia.org/wiki/File:HypergeometricPDF.png

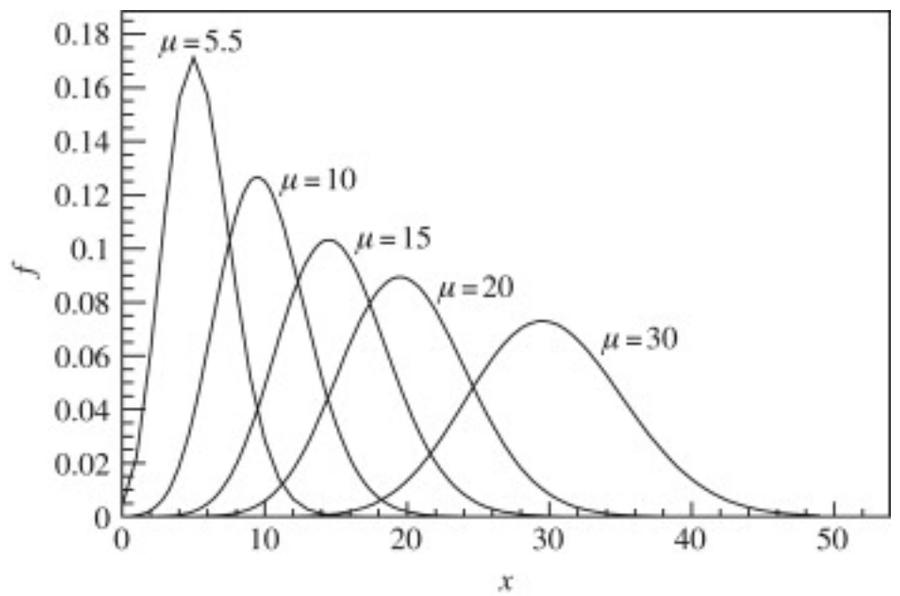
Poisson Distribution

Poisson Distributions

The Poisson distribution expresses the probability of a given number of events occurring in a fixed interval if these events occur with a known constant rate and independently of the time since the last event. It is a limiting case of the binomial distribution.

Examples of Poisson Distributions

The number of decay events per second from a radioactive source, the number of reads aligned to a gene region, etc.



Source: Physics and Engineering of Radiation Detection (2nd ed.), 2015

Probability and Distribution of Continuous Variables

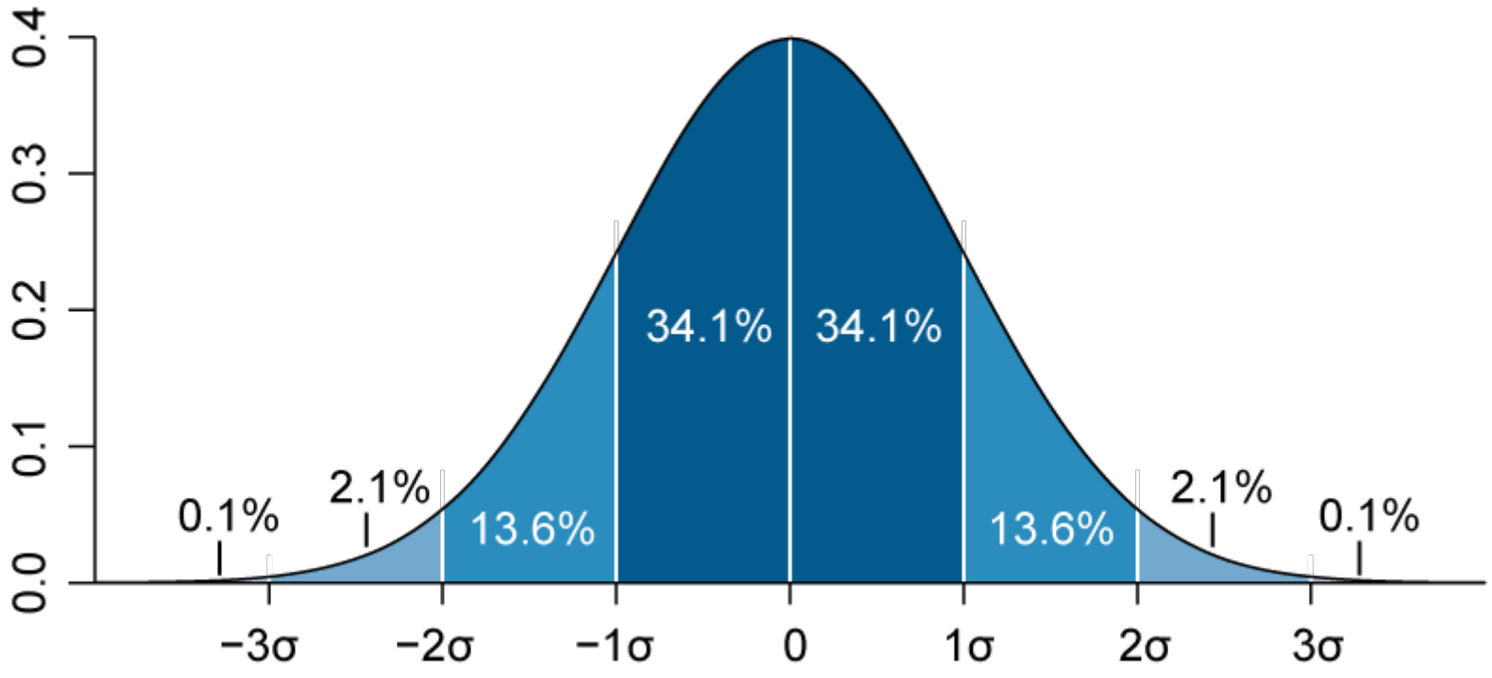
Probability Density Function of a Continuous Variable (pdf)
The collected probabilities of all possible values of that variable. For continuous random variables, a probability histogram is created by rounding off the values, and then a probability density curve that is the limit of the probability histogram is created. The function f(x) that is defined by this curve is the probability density function of X.

Cumulative Distribution Function (cdf)
Given a pdf, a cumulative distribution function can be defined that predicts the probability that the value of a particular continuous random variable will be no greater than a given number. $F(x) = P(X \geq x)$.

Continuous Distributions
Frequently used continuous random variable distributions include uniform, exponential, normal and Student's t. The normal distribution is also known as the Gauss, Gaussian, Laplace-Gauss or bell curve distribution.

Normal Distribution

A commonly used continuous probability distribution, also known as the Gauss, Gaussian, Laplace–Gauss or bell curve distribution. It is particularly useful due to the central limit function, which postulates that observations of random variables become normally distributed when the number of observations is sufficiently large.

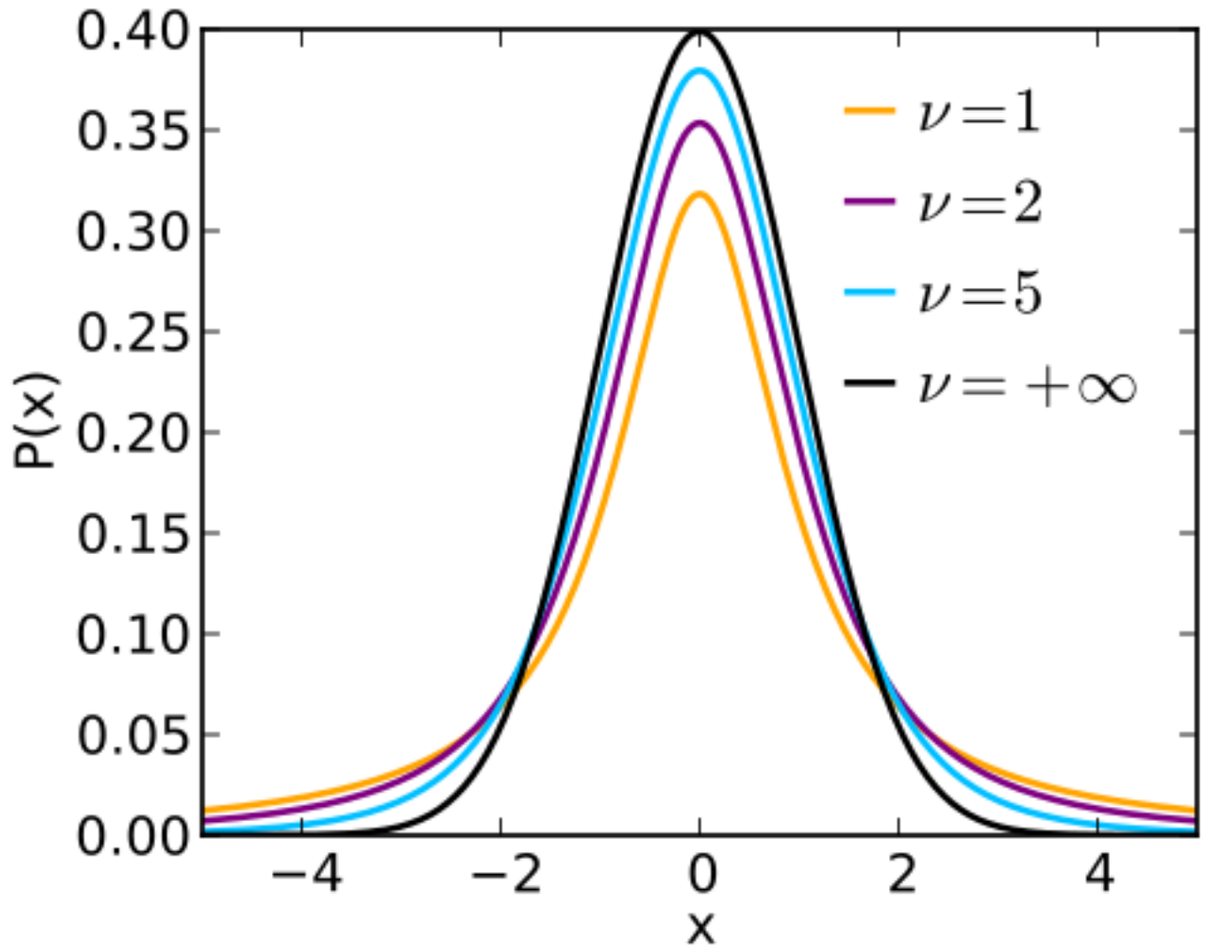


In a normal distribution, the 68-95-99.7 rule describes the percentage of values that fall within roughly one, two or three standard deviations of the mean.

Source: https://commons.wikimedia.org/wiki/File:Standard_deviation_diagram.svg

Student’s t Distribution

A continuous probability distribution used to estimate the mean of a normally distributed population when the sample size (n) is small and the population standard deviation is unknown.



pdf of various t-distributions with v = n-1 degrees of freedom

Source: https://upload.wikimedia.org/wikipedia/commons/4/41/Student_t_pdf.svg

Expected Value and Standard Deviation

Expected Value E(X)
For a discrete random variable, the expected value is the probability weighted average of all its possible values. For example, for a die with six sides, it is 3.5. For a continuous random variable, the expected value is the integral with respect to its probability density function.

Standard deviation (σ)
A measure of the amount of variation in a set of values. It measures the typical distance from the mean, a low standard deviation (sigma) means the values tend to be close to the mean, a high standard deviation means the values tend to be far from the mean.

Significance
In biomedical sciences, a confidence level of the order of a two sigma effect (95%) or a P value of .05, or one in 20, is often considered significant. In particle physics, a five sigma effect, or a P value of 3x10-7, or one in 3.5 million is considered a discovery.

Sample Mean and Median

Sample Mean
The arithmetic mean, or average, of a sample of values drawn from the population.

Median
The middle value (or average of the two middle values).
{1,2,3,3,**4**,5,5,7,8} Median = 4
{1,2,3,**3,4**,5,7,8} Median = (3+4)/2 = 3.5
For a continuous probability distribution, the median is the value where a number is equally likely to fall above or below it.

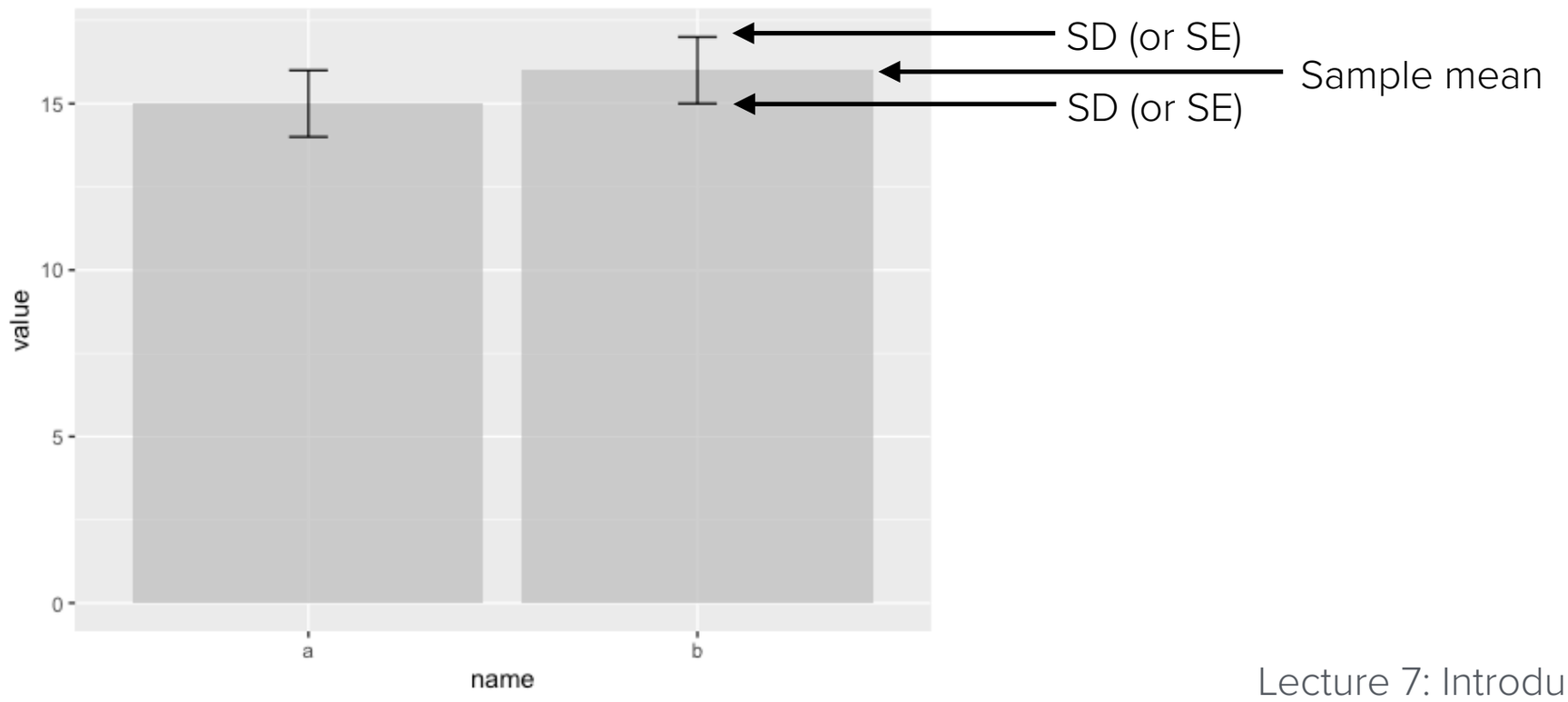
Mean vs. Median
{1,2,3,3,4,5,5,7,8} Mean = 4.2, Median = 4
{1,2,3,3,4,5,5,7,80} Mean = 12.2, Median = 4
Note that the median is less sensitive to outliers.

Sample SD and SE of the Mean

Sample Standard Deviation
S or SD is the standard deviation of a sample of values drawn from the population, and provides an unbiased estimate of σ, E[S] = σ

Standard Error of the Mean
SE = S / sqrt N

Bar Charts with Error Bars



R

The R programming language was released in 1993 by Ross Ihaka and Robert Gentleman, statisticians at the University of Aukland in New Zealand. Their original goal was to develop a statistics language suitable for teaching in their Mac computer lab. The language's name is a reference to the S programming language for statistics, which was one of their inspirations, and also refers to the first names of the authors.

The reference implementation of R is primarily written in C, Fortran and R. It is free and open source, released under the GNU General Public License, and supported by a community of open source developers at the Comprehensive R Archive Network, which serves as a repository for R and free third party R software, and currently contains over 18,500 packages.

R is an interpreted language, and primarily supports procedural programming with functions, but has some object oriented functionality. It supports matrix arithmetic, a wide variety of data structures useful in math and statistics, math symbols, and a variety of graphing functions.

R has become one of the most popular programming languages used by statisticians and data miners, and is well established in bioinformatics. The current Bioconductor repository contains 2,140 free, open source bioinformatics and genomics packages for R.

Source: **https://www.r-project.org** and **https://cran.r-project.org** and **https://bioconductor.org**

RStudio

RStudio is a free, open source, Integrated Development Environment (IDE) for the R programming language that provides R with a simple graphical user interface and useful development tools. It runs on Mac, Windows and Linux.

The source editor features R specific highlighting, code completion, and smart indentation, and allows you directly run R code from it. Help and documentation are built in, along with the ability to quickly jump to function definitions. Additional support for development is provided by an interactive debugger, support for version control systems (Git and Subversion), and package authoring and documentation tools.

RStudio can simultaneously display multiple panes, typically a source code editor pane, an interactive console pane (like Terminal), a workspace pane, and a plotting pane. Interactive graphics can be created with a number of packages.

Source: https://www.rstudio.com

Working with RStudio

You will typically be working in a project with its own working directory. When not in a project, you can set the working directory under **Tools > Global Options...** You can customize the appearance, pane order and tabs of RStudio here as well.

The Source code editor pane (collapsed in upper left by default) is useful for larger projects, otherwise the interactive Console pane (by default on the left below it) can be used directly. The Terminal tab provides system shell access. The Background Jobs tab is used to start, manage and display jobs to be run in the background. To open the Source editor, select R Script from the File menu or the + button in the upper left, or select Move Focus to Source from the View Menu. In the Source editor, **command** and **enter** (Mac) or **control** and **enter** (Win) sends the current line of code to the Console and runs it. Multiple lines of text can be selected beforehand. The Run button also serves the same function.

In both the Source editor and interactive Console, **tab** acts as an auto-complete function to suggest file or function names and **option** and **-** (Mac) or **alt** and **-** (Win) is a shortcut for the frequently used **<–** assignment characters. In the interactive Console, **up arrow** brings up previous commands, while **command** and **up arrow** (Mac) or **control** and **up arrow** (Win) brings up a historical list of previous commands.

The Workspace pane (upper right by default) has an Environment tab which displays anything created during an R session, including values, objects or functions, a History tab that stores all previous commands run, a Connections tab that displays current data sources and allows you to connect to databases, and a Tutorial tab for the **`learnr`** tutorial package.

The View pane (by default below the workspace pane) has a Files tab for displaying the directory structure, a Plots tab for displaying graphs (the arrows move between multiple graphs), a Packages tab that shows loaded packages, a Help tab for displaying documentation, a Viewer tab for displays local HTML and web content, and a Presentation tab for displaying slides with Markdown syntax and LaTeX or MathML equations.

R Functions and Statements

Functions in R are written as the name of the function, directly followed by parentheses, e.g. **`mean()`**. No special characters are required following a function, simply a carriage return as the end of the line.

To use a function, provide the appropriate arguments in the parentheses, e.g. **`mean(1:10)`** or **`mean(data)`**

Statements in R are normally written one to a line, with curly brackets used to group statements.

You can define your own unique functions in R using the syntax:

```
uniquefunctionname <- function(arg1, arg2, ... ){
statements
return(object)
}
```

Help

Typing a question mark directly in front of a function name displays its documentation, e.g. **`?mean`**

Typing the name of the function without parentheses returns its code in the console, e.g. **`mean`**

`example(functionname)` displays examples of how the function is used, e.g. **`example(mean)`**, while **`browseVignette()`** and **`demo()`** may provide additional details for certain packages and their functions.

`args(functionname)` displays a list of the function's arguments, e.g. **`args(mean)`**

`??searchterm` searches all of R's help documentation for the search pattern, it can be prefixed to limit the search.

R Syntax

R supports a number of data types, including scalar variables, vector variables (numeric, character, logical), lists, matrices, and data frames. Variable names may not start with a digit, and can contain underscores, but not most other special characters or spaces, and R is case sensitive. Anything following a **#** character is considered a comment.

Assigning Values
In R, **<–** is usually used to assign a value to a variable, not the **=** symbol e.g. **`x <- 1`**. The shortcut for an assignment is **option** and **-** (Mac) or **alt** and **-** (Win). Most other mathematical operators function as expected.

The combine function, **`c()`** is used to assign multiple same type values to a variable, e.g. **`x_vec <- c(1, 2, 3)`**. The colon symbol **`:`**, is an operator that generates regular sequences, e.g. **`x_vector <- 1:10`** then can ask **`mean(x_vector)`**.

The list function, **`list()`** is used to assign multiple different types of values to a variable, e.g. **`x_list <- list(1, "two", 3.0)`**. Note that unlike Python, the first value in an R list is at position 1, not 0, and R assumes that 1 and 3.0 are both numeric data types, not an integer and a float.

Matrices and Data Frames
A matrix consists of rows and columns of the same data type, e.g. **`x_matrix <- matrix(c(1, 2, 3, 4), nrow=2, ncol=2)`**. By default, a matrix fills by column. The **`dim()`** function returns the dimensions of a matrix.

A data frame has rows and columns, but can store different data types in each column, and each column must have a name, preferably unique, e.g. **`x_dataframe <- data.frame(a = 1:5, b = 6:10)`**. Note how the colon operator is used to generate a range of numbers and the equal sign is properly used here as an equality operator.

Operations and Loops
Operations, including loops, are generally applied in R with the **`apply()`** function, although if and if...else statements are also supported. The third-party **`plyr`** package lets you easily apply operations to part of a data set. The colon operator can be used as an iterator in for loops, e.g **`for (x in 1:10)`**.

R Standard Library

Viewing Data

`attach()` attaches a list or data frame to the R search path for easy searching.
`edit()` displays an editable version of an object.
`print()` displays the values of an object.

Graphing Data

`barplot()` creates a bar plot.
`boxplot()` creates a box plot.
`heatmap()` creates a heat map.
`hist()` creates a histogram.
`plot()` creates a scatter plot.

Statistics

`mean()` calculates the arithmetic mean.
`sd()` calculates the standard deviation.
`summary()` provides summary statistics if used with a data frame.
`table()` cross-tabluation and table creation.
`t.test()` one and two sample t-tests on vectors of data (Student's t-Test).
`var, cov` and **`cor`** compute the variance of x and the covariance or correlation of x and y.

R Input and Output

Input

`test_data <- read.csv("filename.txt", header=FALSE)` will read the data in the named CSV file into a data frame. Note that a header is normally expected, and in some cases you may need to add **`stringsAsFactor=FALSE`** to prevent strings from being interpreted as statistical values by R. You can also provide a URL instead of a file name (assuming the data at the URL is in standard CSV format). **`read.csv()`** is identical to **`read2.csv()`** and **`read.table()`** except with a different set of defaults.

In RStudio, you can simply use **Import Dataset** in the **Environment** tab.

Many third party packages exist for reading data in particular file formats, databases or online data sources.

Output

`save.image()` will save the entire workspace to a file named .RData

To save data in a R compact binary file, use **`save(data, file="filename.rda")`** and to load the object, use **`load("filename.rda")`**.

To save data in R text format, use **`dump(data, "filename.Rdmpd")`** and to load it, use **`source("filename.Rdmpd")`**.

`write.csv(test_data, "test.csv")` will save data to a CSV text file. To omit headers, add the argument **`row.names=F`**. Use **`write(test_data, "test.txt")`** to save data to a regular text file.

Statistics Examples with R

```
x <- runif(10)  #Uniform distribution, also try 10000
summary(x)

y <-sample(1:100, 10, replace=T)
summary(y)
print(y)

y <- sample(1:100, 100, replace=T)
summary(y)
mean(y)
sd(y)
print(y)

y <- sample(1:100, 10000, replace=T)
summary(y)
sd(y)

z <- rnorm(10)    #Normal distribution, also try 10000
summary(z)
sd(z)             #Expected value is 1.0
```

Probability with R

Coin Tosses

```
rbinom(1,1,0.5)              #Binomial distribution
rbinom(1,100,0.5)
rbinom(1,10000,0.5)
rbinom(1,1000000,0.5)
rbinom(1,100,0.5)            #View results of 100 flips
```

Random DNA

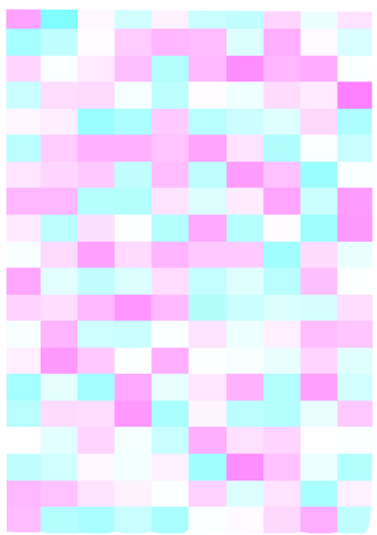
```
x <- sample(c("A","C","G","T"),10,replace=T)
x

x <- sample(c("A","C","G","T"),100,replace=T)
x
write(x, "100rtest.txt")     #To avoid one character per line use
y = paste(x, collapse="")    #paste with an empty collapse argument
y                            #to remove the default blank space
write(y, “100rdna.txt”).    #then write the data to a file
```

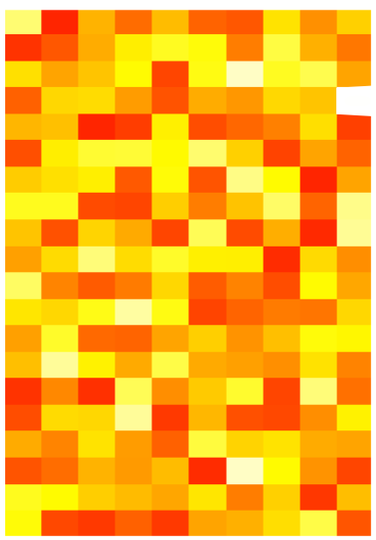
Heat Maps with R

The following R source code generates a random matrix of 10 columns and 20 rows containing 200 random integers between 1 and 100, then views the randomly generated data. It then creates a heat map using the default cyan to purple **heatmap** colors (note that there is no line break in the third line).

```
hm <- matrix(sample(1:100, 200, replace=T), ncol=10)
hm
hm_heatmap <- heatmap(hm, Rowv=NA, Colv=NA, col =
cm.colors(256), scale="column", margins=c(10,20))
```

Use **col = heat.colors(256)** for more temperature-like colors. Third-party packages such as **gplots** (with **heatmap.2**), **ggplot2** or **RColorBrewer** offer far more options.





Plotting Examples with R

```
a <- sample(1:100, 100, replace=T)    #Random sample with replacement
plot(a)
b <- rnorm(100, mean=50, sd=9)
plot(b)                               #looks similar, but check the y axis!
c <- rnorm(100, mean=50, sd=3)
plot(c)                               #looks similar, but check the y axis!
```

How to display this on a single plot? The lines() function will add lines to an existing plot.

```
a <- sample(1:100, 10000, replace=T) #Increased n for visibility
b <- rnorm(10000, mean=50, sd=9)
c <- rnorm(10000, mean=50, sd=3)

plot(a, type="l",col="red")
lines(b, col="green")
lines(c, col="blue")
```

Histogram of normal distribution?

```
x <- rnorm(100)
hist(x, col="lightblue", freq=T)
summary(x)

x <- rnorm(100)                          #can vary n
hist(x, col="lightblue", freq=F)         #density instead of frequency
curve(dnorm(x, mean=0, sd=1), add=T, col="darkblue")    #adds a density curve
```

Installing R Packages

Additional third-party packages for R can be browsed for at the Comprehensive R Archive Network (CRAN) at **https://cran.r-project.org** or searched and browsed for at **https://www.rdocumentation.org** (which also includes package documentation and download statistics).

Installation of third-party packages is handled by the built-in **`install.packages()`** function, e.g. **`install.packages("ggplot2")`**. Run **`update.packages()`** beforehand to make sure your other packages are up to date. In RStudio, you can simply use **Tools > Install Packages...**

R packages are installed into *libraries*, which are directories containing a subdirectory for each package installed there. To load a package, use the **`library()`** function, e.g. **`library("ggplot2")`**

Some notable third-party R packages include:

ggplot2, a plotting system featuring a variety of plots, statistical transformations and display options.

plotly, an interface to interactive online plot.ly graphs.

plyr, tools for splitting, applying and combining data (or **dplyr** for data frames).

shiny, an interactive web application framework for Shiny servers or cloud hosting.

stringr, makes R string functions more consistent, simpler and easier to use.

References

An Introduction to R free at:
https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf

swirl, a free interactive console tutorial package for R:
`https://swirlstats.com`
`https://github.com/swirldev/swirl_courses`
`install.packages("swirl")`
`install_course("R Programming”)`
`install_course("The R Programming Environment")`
`install_course("Getting and Cleaning Data")`

learnr, a built-in tutorial package for R (in the RStudio Tutorials tab):
`https://rstudio.github.io/learnr`
`install.packages(“learnr")`

Introductory Statistics with R by Peter Dalgaard