R-tutorial and Introduction to R Shiny

R-Shiny by RStudio

-A web application framework for R

-No HTML,JavaScript, or CSS required

-Built on a Reactive Programming Model

Reactive Programming- programming with non parallel data streams. Programming paradigm oriented around data flows and propagation of change. If you want to learn more about this take Comp 310 or Comp 333.

Today, the following objectives will be covered:

- What consists of developing an R-Shiny applet
- Basic proficiency in R, to eventually make your own R Shiny Applet
- **We will learn a little bit of how to apply R to the following areas:**
- Areas from Statistics & Probability
- Calculus <Taking a Derivative and Integration>
- Disease Modeling in R
- Combinatorial Algorithms/Dynamic Programming

Examples of R Shiny Applets

Long Run (Category of Probability) -According to Statistics.CalPoly.edu,"one..activity to help students understand change behavior is to observe the runs of consecutive heads or tails in a sequence of coin flips."

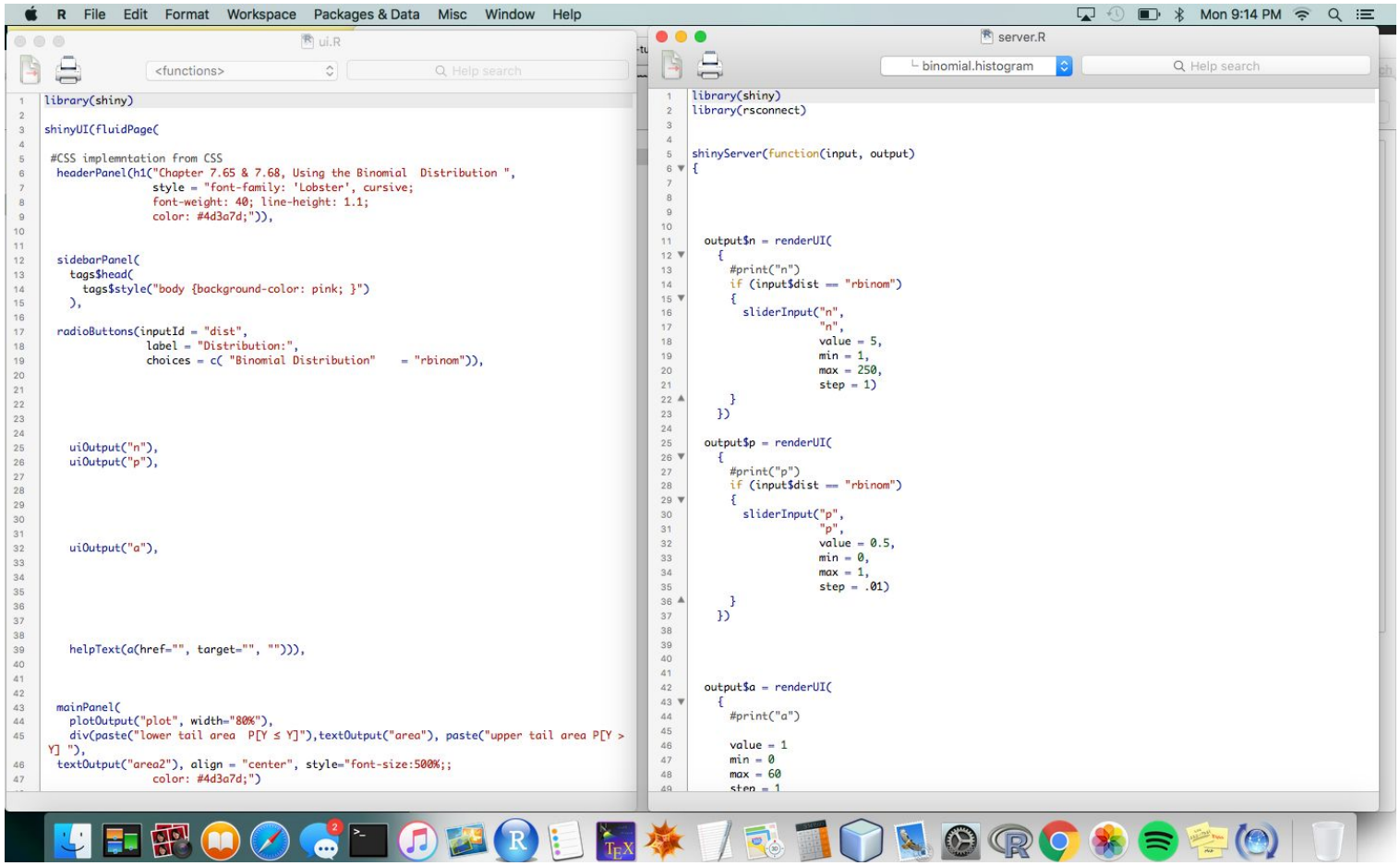
Length/Coverage Optimal Confidence Intervals (Category of Inference)–According to Statistics.CalPoly.edu, "In 2014 Schilling and Doi developed a binomial confidence interval that produces coverage probabilities always at least equal to the stated confidence level and has the highest possible coverage amongst all such methods."

Schilling, M., and Doi, J. (2014) 'A Coverage Probability Approach to Finding an Optimal Binomial Confidence Procedure' *The American Statistician* , 68(3), 133--145 (Online access)

Schilling, M. 'The Longest Run of Heads' *The College Mathematics Journal* , 21(3), 196--207

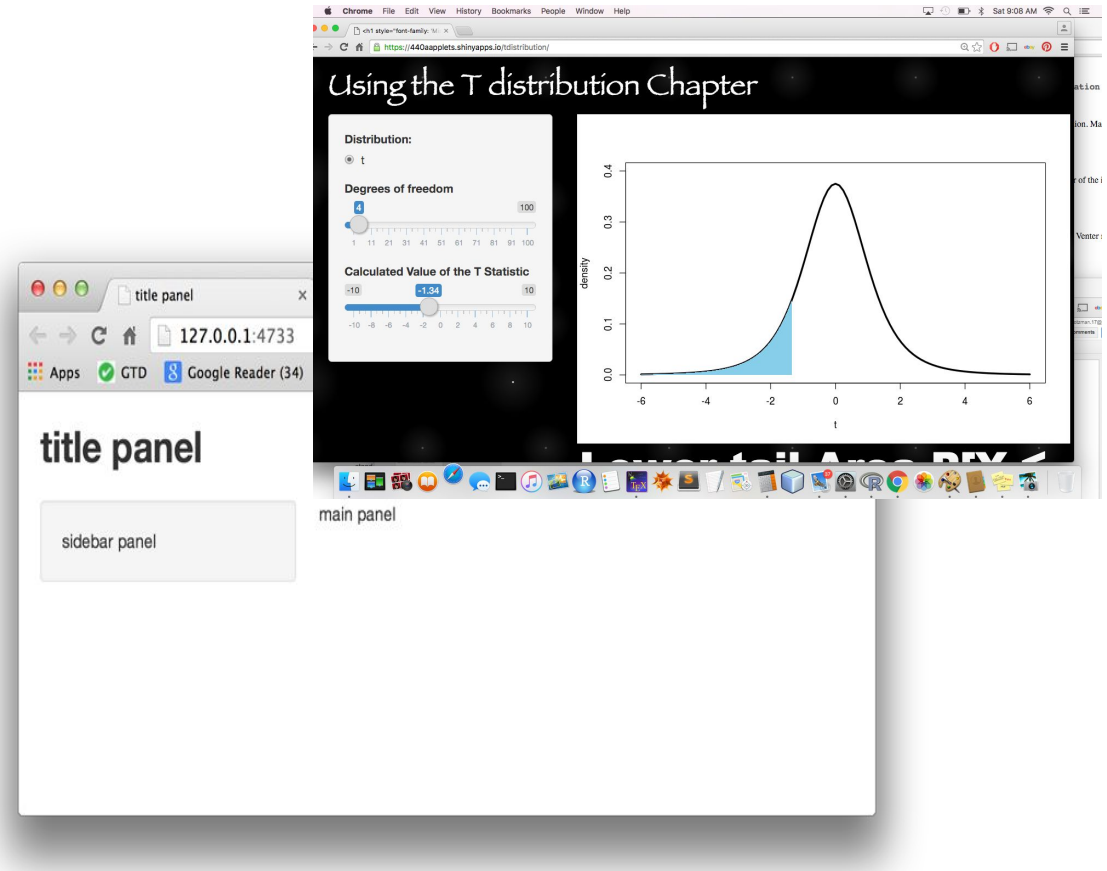
Schilling, M. 'The Surprising Predictability of Long Runs' *Mathematics Magazine* , 85(2), 141--149

What, does an R Shiny Applet Consist of?

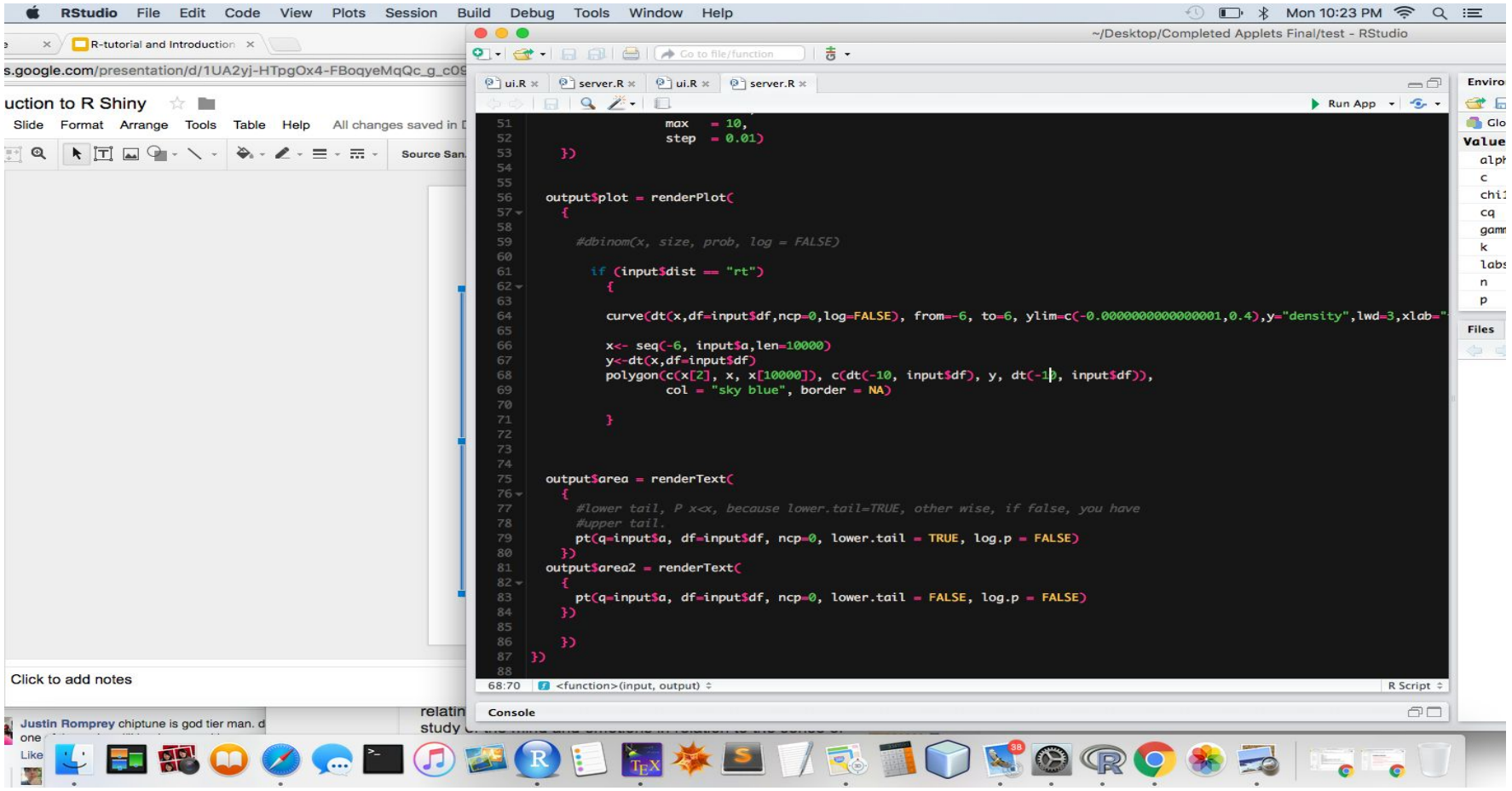


Ui.R allows you to give your applet Esthetic

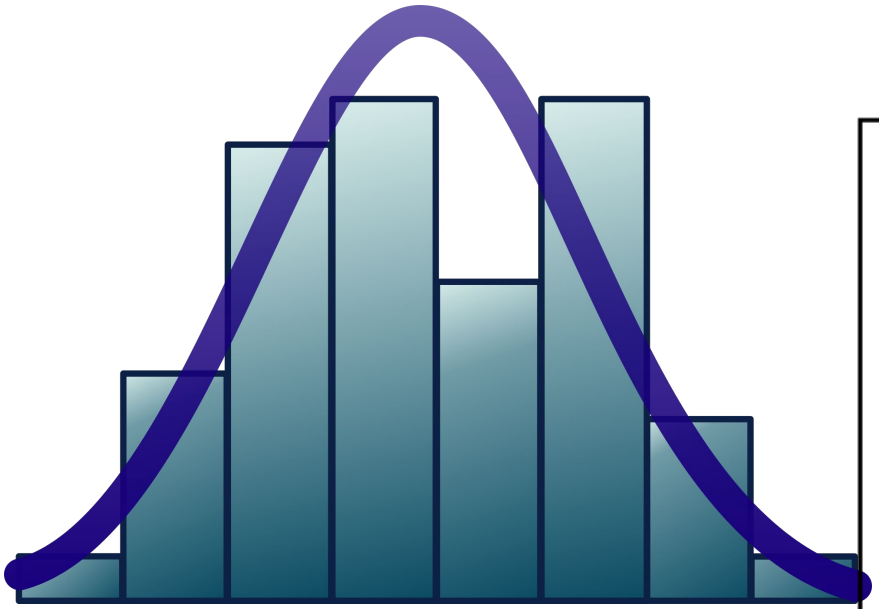
```
# ui.R

shinyUI(fluidPage(
  titlePanel("title panel"),

  sidebarLayout(
    sidebarPanel( "sidebar panel"),
    mainPanel("main panel")
  )
))
```

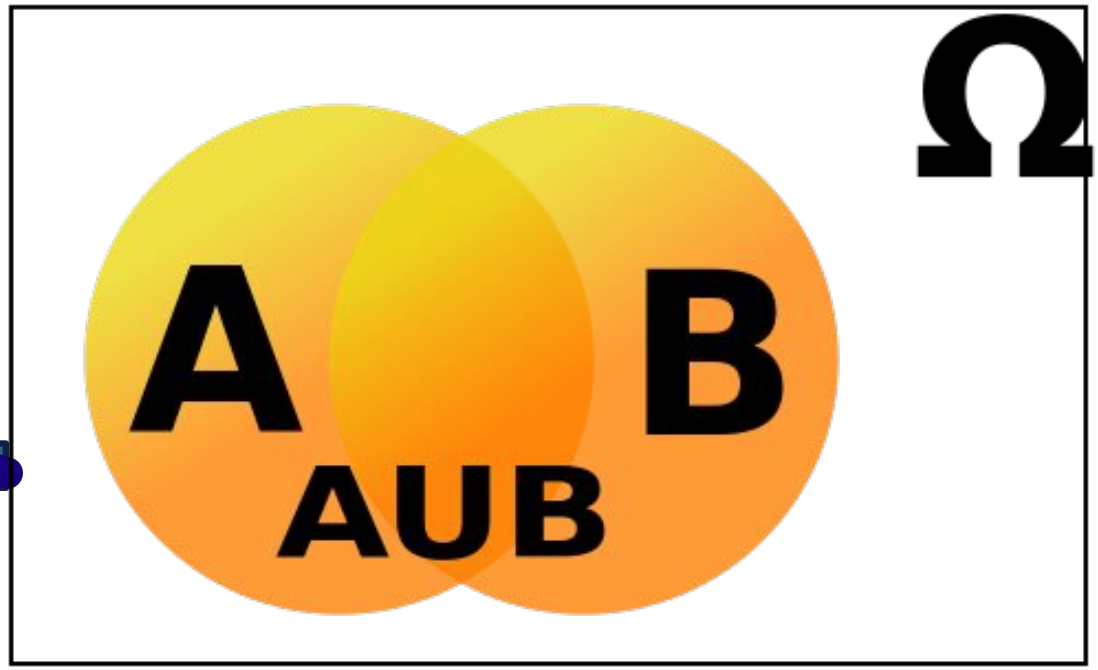


Server.R- give the applet functionality



Category: Statistics & Probability





The Basics

Computing mean
x <- c(x1.x2,....xn)
 mean(x)

Computing standard deviation
X <- c(x1,....xn)
sd(X)

Computing the median
X <- c(x1....xn)
median(X)

Continuation….

To determine the mode, you have to do something different.

You have to install the package, modeest, which provide estimators of the mode of univariate unimodal (and sometimes multimodal) data and values of the modes of usual probability distributions.

Modeest, allows us to return the most frequent value in a given numeric vector.

```
install.packages("modeest")

library(modeest)

mySamples <- c(x1,...xn)

mlv(mySamples, method = "mfv")
```

*mfv signifies most frequent value.

Continuation...

factorial(num)

We can also determine the binomial coefficient:

The number of combination of length k within n numbers (n choose k),

So you can type in :

choose (n,k)

Union and Intersection of numbers:

Define a function with a set of numbers,

Function <- c(x1,.....xn)

Function1 <- c(x2,....xn)

union(Function, Function1)

Same Approach can go for Intersection

intersect(a:b,a:b)

Creating histograms, stem and leaf plots, scatter plots, pie charts

Creating a histogram

# Define the function vector with 5 values
function<- c(x1,x2,..xn)

colors = c("color1",....colorn)

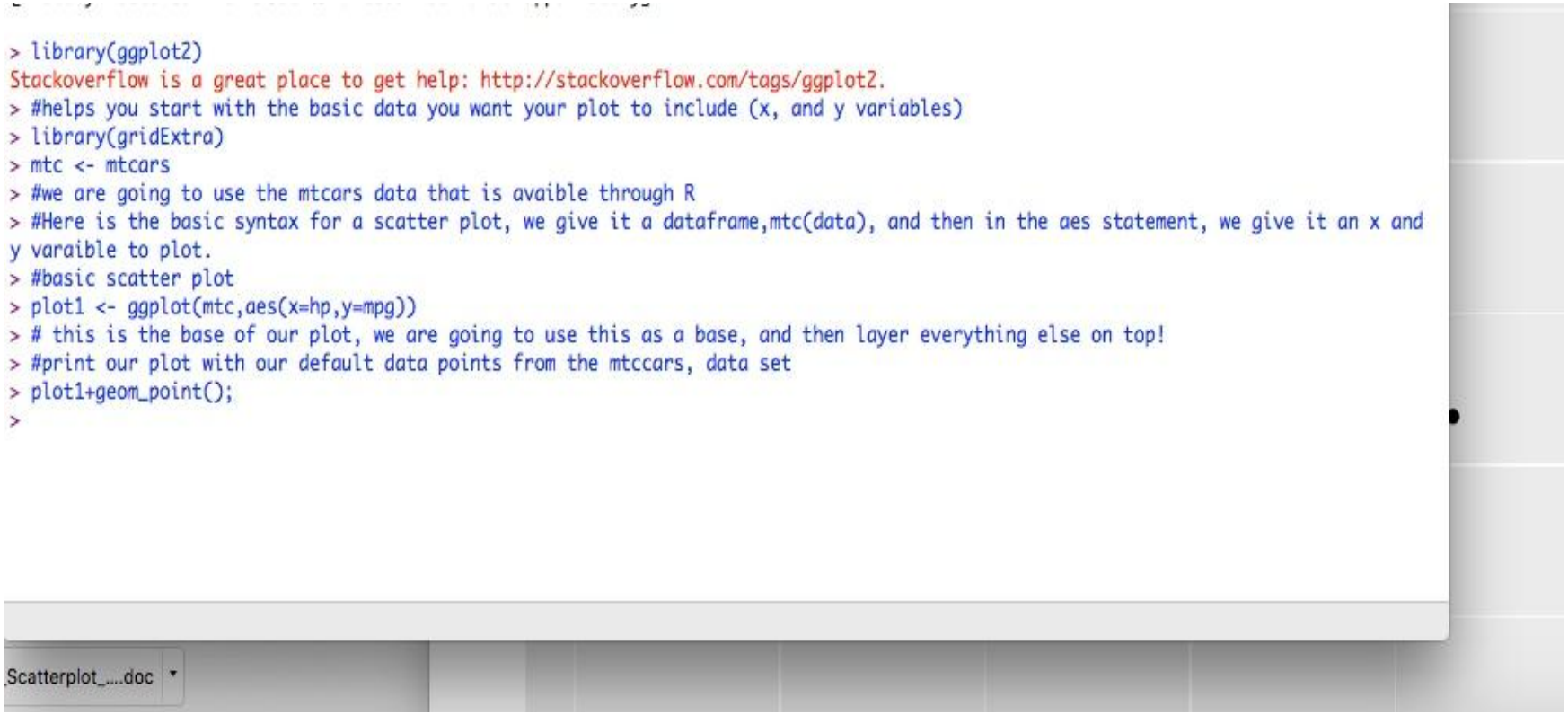
Create a histogram for function

hist(function, col=colors)

Stem and Leaf Plot

```
x <- c(x1,x2….,xn)

stem(x,scale=?)
```

Scatter Plots



Pie Charts

```
# Create data for the graph.
x <- c(x1,..xn)
labels <- c(“label1”,... ,“label_n”)

# Give the chart file a name.
png(file = "example.jpg")

# Plot the chart.
pie(x,labels)

# Save the file.
dev.off()
```

R functions for Probablity Distributions

Distribution	Functions			
Beta	pbeta	qbeta	dbeta	rbeta
Binomial	pbinom	qbinom	dbinom	rbinom
Cauchy	pcauchy	qcauchy	dcauchy	rcauchy
Chi-Square	pchisq	qchisq	dchisq	rchisq
Exponential	pexp	qexp	dexp	rexp
F	pf	qf	df	rf
Gamma	pgamma	qgamma	dgamma	rgamma
Geometric	pgeom	qgeom	dgeom	rgeom
Hypergeometric	phyper	qhyper	dhyper	rhyper
Logistic	plogis	qlogis	dlogis	rlogis
Log Normal	plnorm	qlnorm	dlnorm	rlnorm
Negative Binomial	pnbinom	qnbinom	dnbinom	rnbinom
Normal	pnorm	qnorm	dnorm	rnorm
Poisson	ppois	qpois	dpois	rpois
Student t	pt	qt	dt	rt
Studentized Range	ptukey	qtukey	dtukey	rtukey
Uniform	punif	qunif	dunif	runif
Weibull	pweibull	qweibull	dweibull	rweibull
Wilcoxon Rank Sum Statistic	pwilcox	qwilcox	dwilcox	rwilcox
Wilcoxon Signed Rank Statistic	psignrank	qsignrank	dsignrank	rsignrank

-Every distribution that R handles has four functions. There is a root name. For example the root name for a binomial distribution is binom. This root is then prefixed by one of the following letters:

- **p for "probability", (c. d. f.)**
- **q for "quantile", the inverse c. d. f.**
- **d for "density", the density function p. d. f**
- **r for "random", a random variable having the specified distribution**

I will demonstrate with the Normal Distribution

```
starting httpd help server ... done
> dnorm(1, mean = 0, sd = 1, log = FALSE)
[1] 0.2419707
> #dnorm function returns the height of the normal curve at some value along x axis
> dnorm(1,mean=0,sd=1,log=FALSE)
[1] 0.2419707
> #pnorm function is the cdf, it returns the area  below the given value of x
> pnorm(1,mean=0,sd=1,log=FALSE)
[1] 0.8413447
> #the values for default for mean and sd are set to 0 and 1. But these values can be set to something else in the case of d norm
> #to find the area above the cutoff x, either substract 1 or set lower.tail=FALSE
> 1-pnorm(1)
[1] 0.1586553
> pnorm(1,mean=0,sd=1,lower.tail=FALSE)
[1] 0.1586553
> #finally to get quantiles or critical values, you can use the qnorm function,
> #p=.05, one tailed (upper)
> qnorm(.95)
[1] 1.644854
> qnorm(c(.025,.975))
[1] -1.959964  1.959964
>
```

Confidence Intervals

Confidence Interval-a range of values so defined that there is a specific probability that the value of a parameter lies within it.Interval estimate combined with a probability statement.

-We are able to compute confidence intervals from multiple simulated data sets

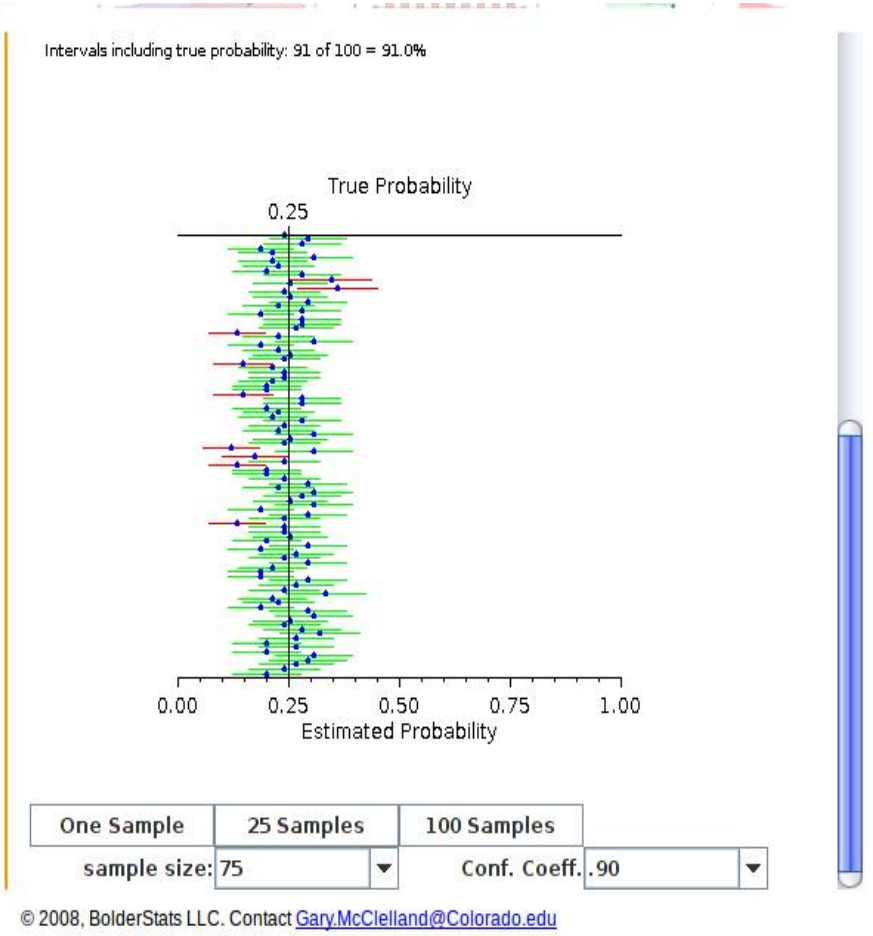
-To do this we can use the command CIsim

-This function automates the calculation of coverage rates for exploring the robustness of confidence interval methods

This data frame will return the following:

**A data frame with variables lower, upper, estimate, cover ('Yes' or 'No'), and sample is returned invisibly.*

Coverage percentile will be shown on the R console



What do we need to take into consideration in order to compute this, and use the CISim command

n:size of each sample

samples:number of samples to simulate

dist:function used to draw random samples

args:arguments required by rdist

plot:one of "print", "return" or "none" describing whether a plot should be printed, returned, or no generated at all.

estimand:true value of the parameter being estimated

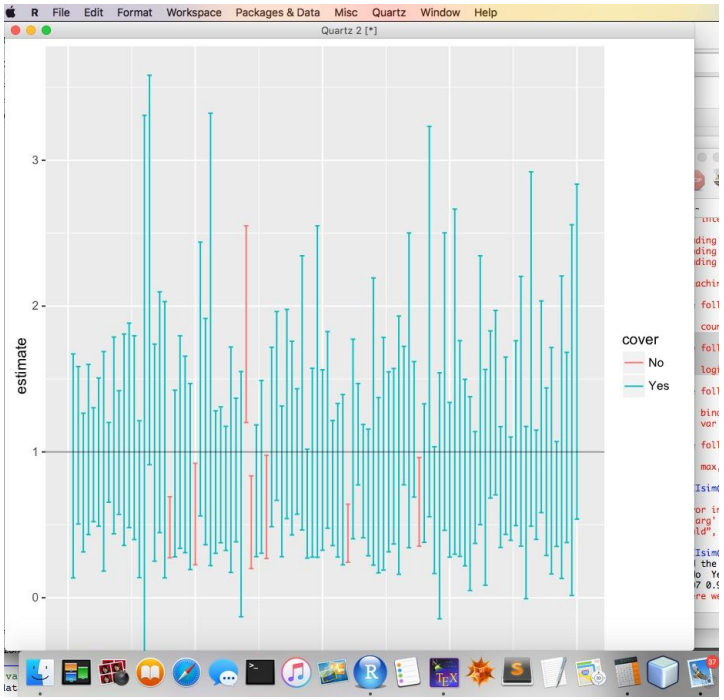
conf.level:confidence level for intervals.
Method- function used to compute intervals
Method.args: arguments required by method
Interval: function that computes a confidence interval from data, function should be a vector length of 2
Estimate: a function that computes an estimate from data

If we want to do a Binomial

 # Binomial treats 1 like success, 0 like failure

CIsim(n=30, samples=101, rdist=rbinom, args=list(size=1, prob=.3),

estimand = .3, method = binom.test, method.args=list(ci = "Wald"))



*Every time, you put this in the console, the console will give how much of the interval did the p value cover?

Decision Trees

Decision tree is a decision support tool that uses a tree like graph using a branching method to illustrate every possible outcome of a decision.

Examples of when Decision Trees are used:

Predicting will I pass the next Math 440B exam?
Predicting a cancer to be tumorous?
Predicting a loan to be good or bad?

Steps to develop a decision tree

install.packages("party")

The basic syntax for creating of decision tree

ctree(formula, data)

formula is a formula describing the predictor and response variables.

data name of the data set used

Predictor variable-is a variable used in regression, to predict another variable.

Response Variable- dependent variable

For our example we will use today:

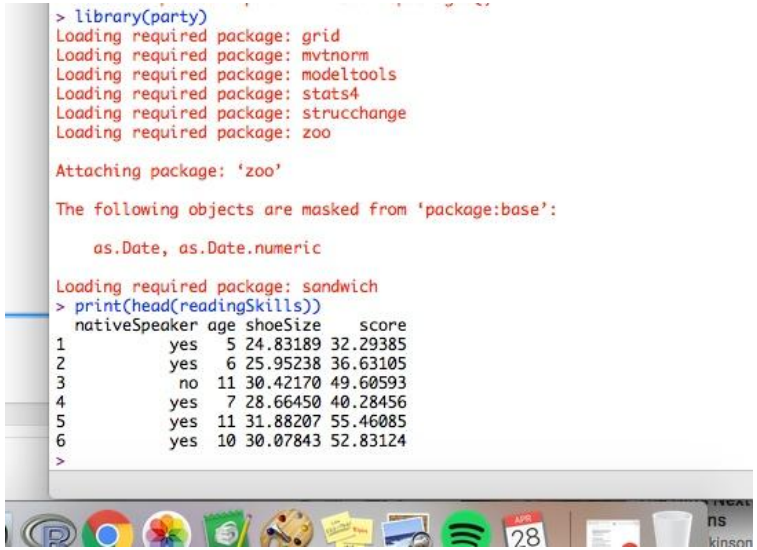
We will use the famous R built in data set named **readingSkills** to create a decision tree. It describes the score of someone's readingSkills if we know the variables "age","shoesize","score" and whether the person is a native speaker or not.

To see the data:

install.packages("party")

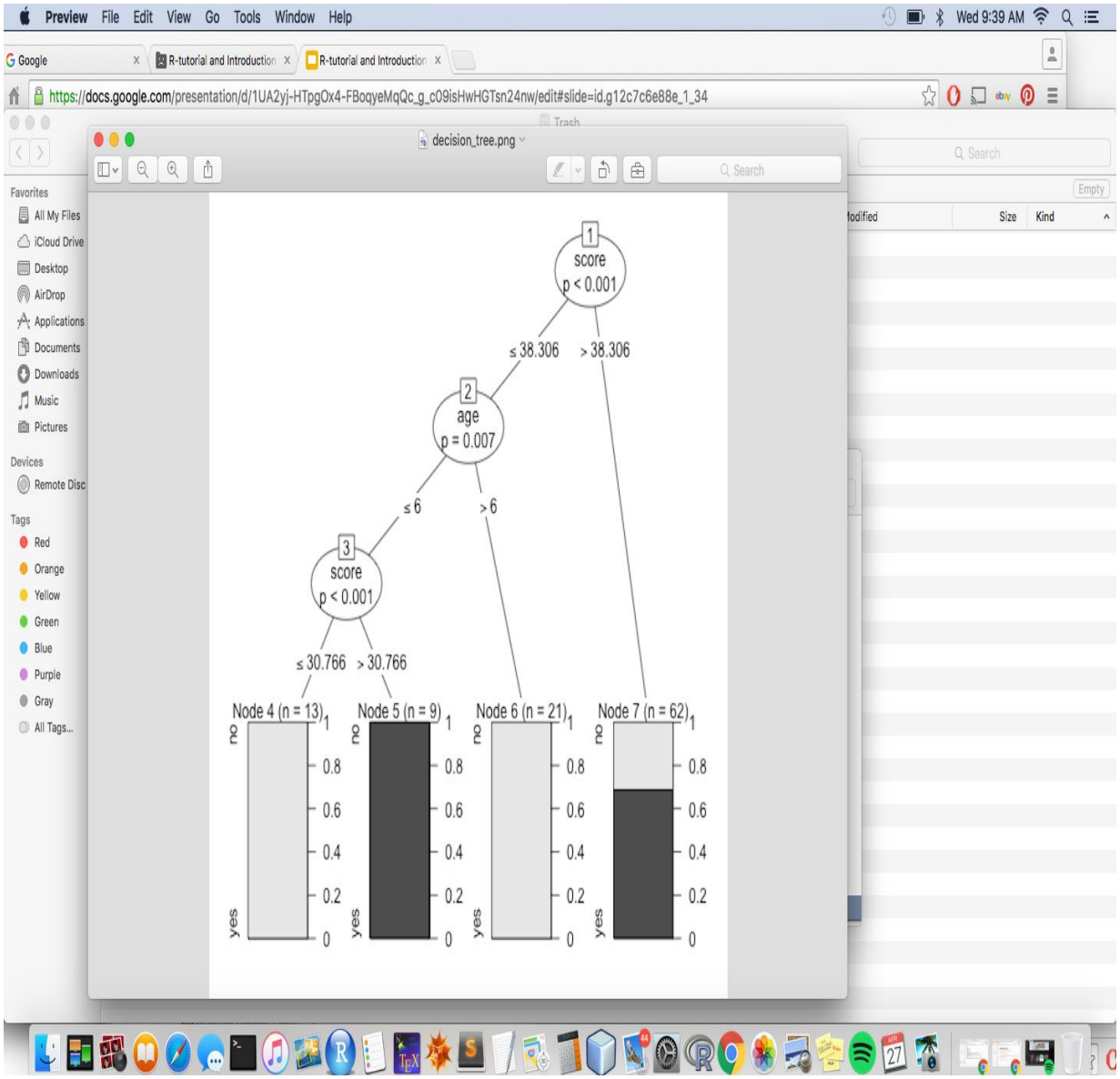
library(party)

print(head(readingSkills))



Example of how to make a decision tree.

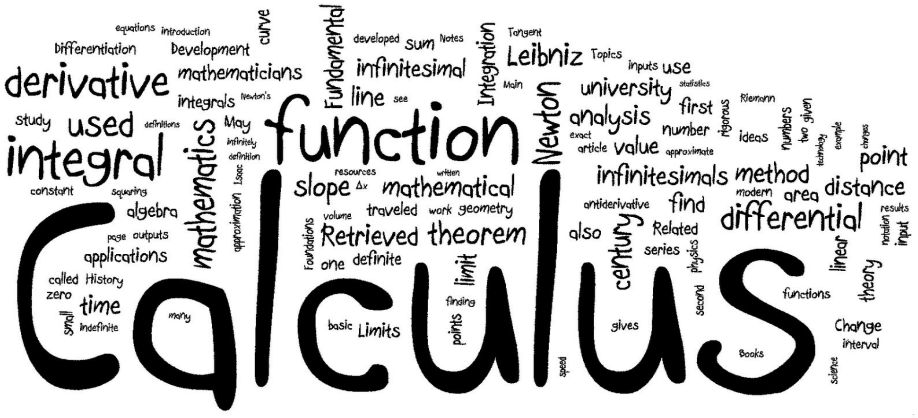
```
# Load the party package. It will
automatically load other dependent packages.
library(party)

# Create the input data frame.
input.dat <- readingSkills[c(1:105),]

# Give the chart file a name.
png(file = "decision_tree.png")

# Create the tree.
  output.tree <- ctree(
  nativeSpeaker ~ age + shoeSize + score,
  data = input.dat)

# Plot the tree.
plot(output.tree)

# Save the file.
dev.off()
```



-solving indefinite

- definite integrals

-taking a derivative



Determining a Derivative

*This requires the mosaic library

library(mosaic)

symbolicD(function~(what you want to take the derivative with respect to),. order=which derivative?)

```
Error: object 'd' not found
> d
Error: object 'd' not found
> d
Error: object 'd' not found
> d
Error: object 'd' not found
> library(mosaic)
> symbolicD(x^3~x, .order=1)
function (x)
3 * x^2
> symbolicD(x^3~x,.order=2)
function (x)
3 * (2 * x)
> symbolicD(x^3~x,.order=3)
function (x)
3 * 2
>
```

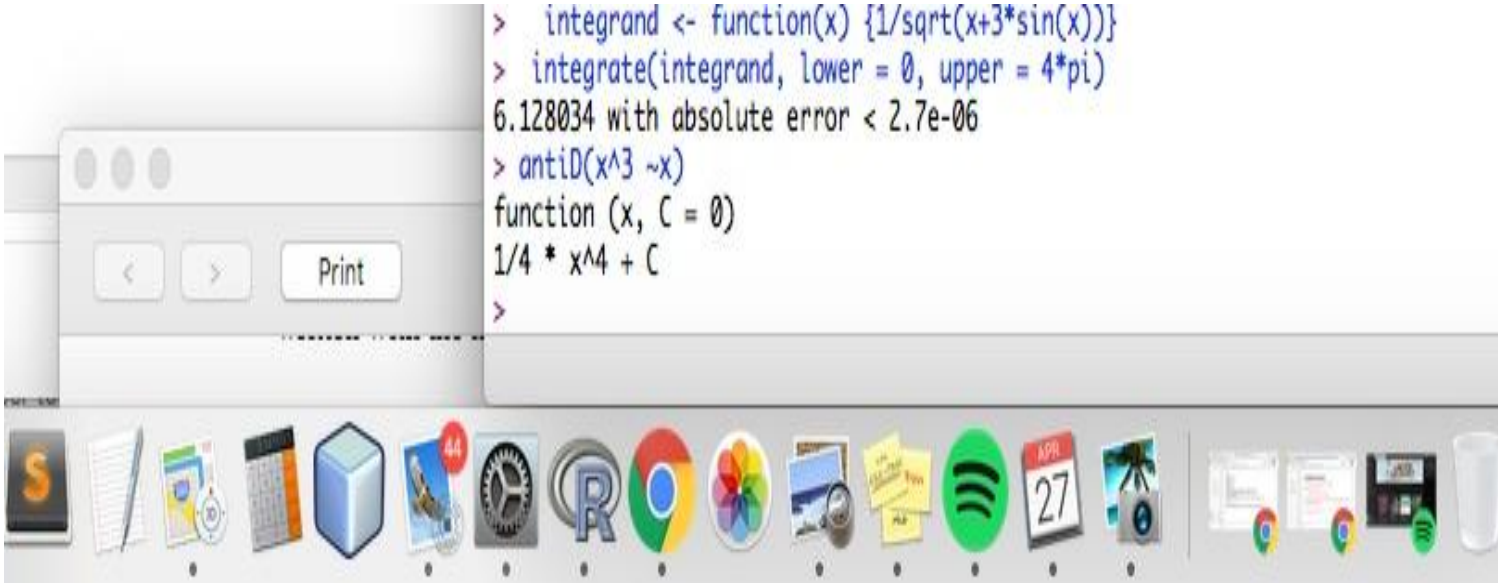
Integrals

#definite & indefinte integrals

library(stats)

integrand <- function(integrating with respect to) {function}
integrate(integrand, lower = ?, upper = ?)

antiD(function ~ integrating with respect to)



Disease Modeling in R

Applying the Simple SIR model in R

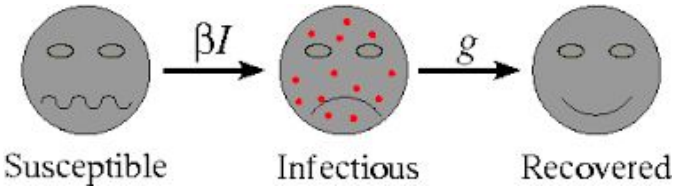
SIR model- computes the *"theoretical # of people infected with a contagious illness in a closed population over time"*

#The SIR Model is a simple model of infectious disease in a population which is broken into 3 groups:

1) Susceptible

2) Infected

3) Recovered



```
install.packages("deSolve")

library(deSolve)

#differential equation library

#first we need to create a function, that calls our differential equations.
```

We can do the following.

Producing the basic SIR model

```
#call your  SIR differential equations

sir <- function(time, state, parameters) {

  with(as.list(c(state, parameters)), {

    dS <- -beta * S * I

    dI <- beta * S * I - gamma * I

    dR <- gamma * I


    return(list(c(dS, dI, dR)))

  })
```

**To understand

Between S and I, the transition rate is β, where β is the contact rate, which takes into account the probability of getting the disease in a contact between a susceptible and an infectious subject.

Between I and R, the transition rate is v (simply the rate of recovery or death).

#**give values to S and I and your parameters**

init ‹« c(S = 1-1e-6, I = 1e-6, 0.0)

parameters ‹« c(beta = 1.4247, gamma = 0.14286)

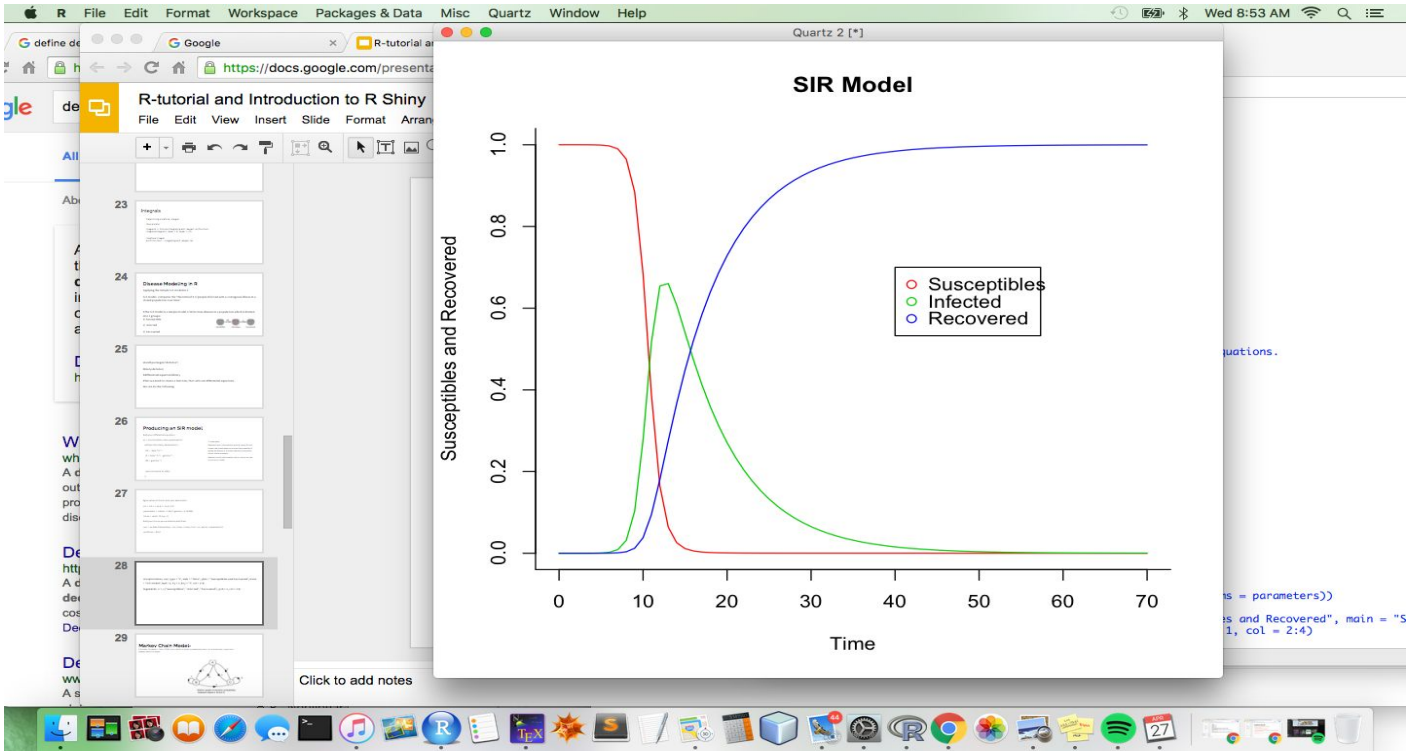
times ‹« seq(0, 70, by = **1)**

#**call your d.es in a dataframe so you are able to plot them**

out ‹« as.data.frame(ode(y = init, times = times, func = sir, parms = parameters))

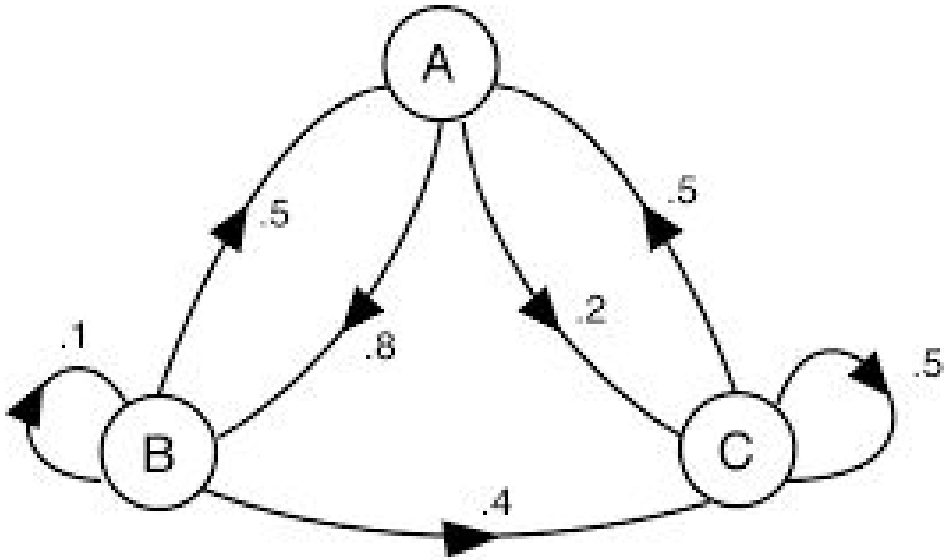
out$time ‹« NULL

```
matplot(times, out, type = "l", xlab = "Time", ylab = "Susceptibles and Recovered",
main = "SIR Model", lwd = 1, lty = 1, bty = "l", col = 2:4)

legend(40, 0.7, c("Susceptibles", "Infected", "Recovered"), pch = 1, col = 2:4)
```



Markov Chain Model-

A Discrete Time Markov Chain (DTMC) is a model for a random process where one or more entities can *change state* between distinct timesteps.



Markov graph of transiton probabilites between states A, B and C

Developing a Markov Chain Model in R should be Simple

To Create a Markov Chain in R, we need to know the following:

1) **Transition probabilities or the chance that an entity will move from one state to another. (Chance of getting reinfected?)**
2) **The initial state (that is, how many entities are in each of the states at time t=0 to t=3)**
3) **The markov chain package in R. Be sure to install markov chain before moving forward.**

Today, we will just Create a Model and model the following:

So, let’s imagine the situation for our Markov Chain Model:

There is a 10% infection rate of the Zika Virus.

There is a 20% recovery rate of the Zika Virus.

This implies that 90% of the Susceptible people will remain in the Susceptible state.

80% of those who are Infected will move to the Recovered Category, between successive timesteps.

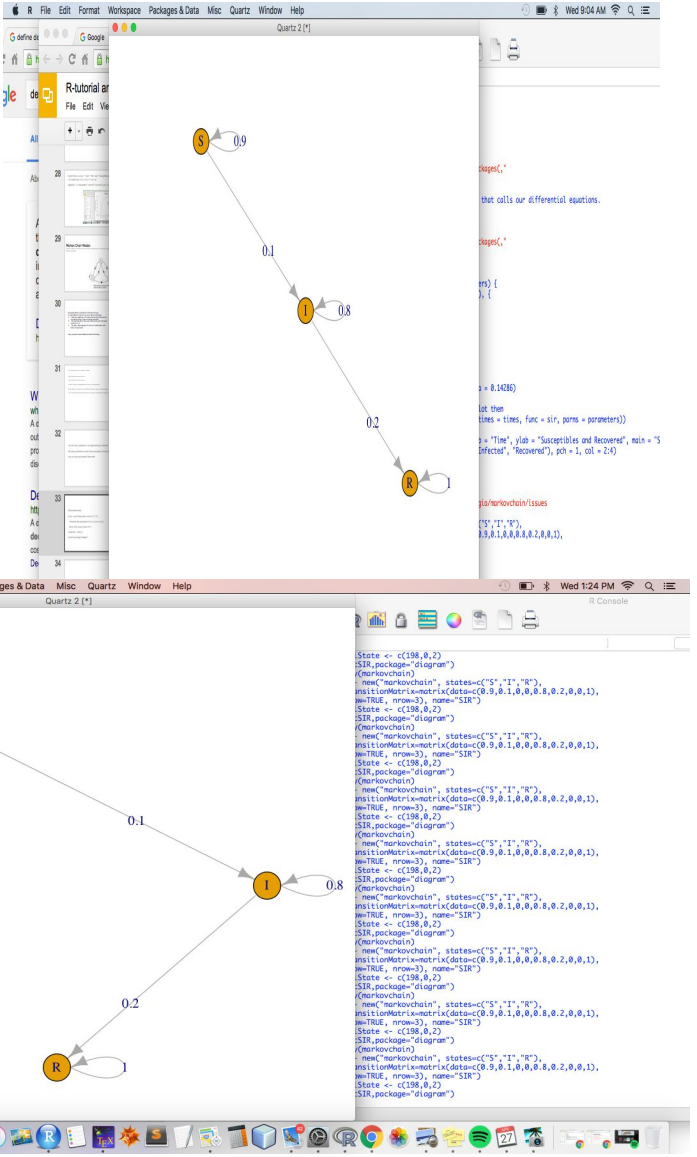
100% of those Recovered will stay recovered. None of the people who are Recovered will become Susceptible.

Let’s start with a population of 200 people and only 2 is infected.

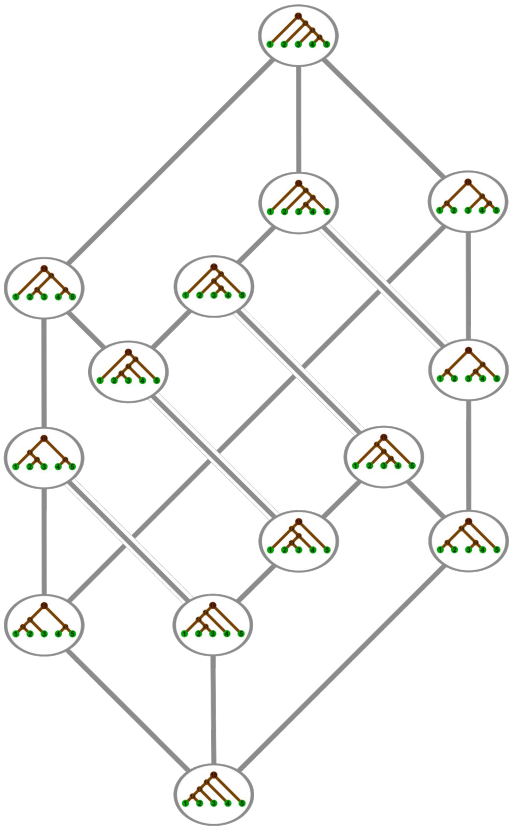
That mean you initial state is that 198 are Susceptible, 2 Infected, and 2 are recovered.

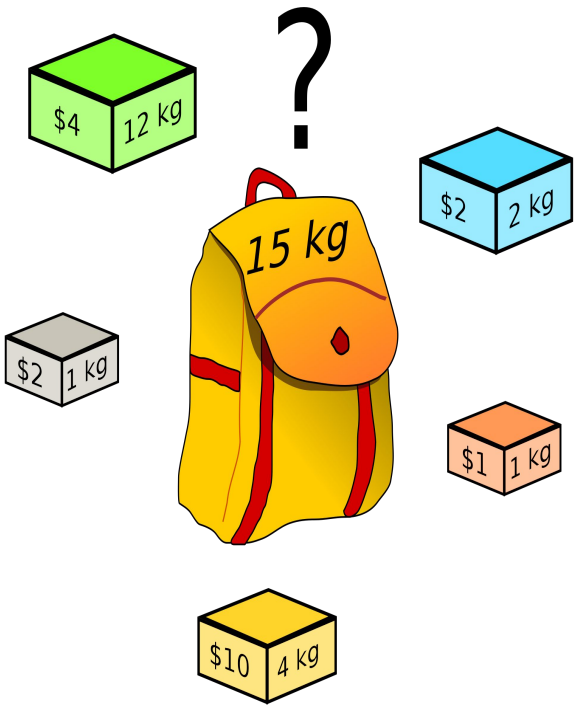
Now, we can set up our Markov Chain Model.

```
library(markovchain)

mcSIR ‹‹ new("markovchain", states=c
("S","I","R"),

    transitionMatrix=matrix(data=c
(0.9,0.1,0,0,0.8,0.2,0,0,1),

   byrow=TRUE, nrow=3), name="SIR")

initialState ‹‹ c(198,0,2)

plot(mcSIR,package="diagram")
```



Dynamic Programming/Combinatorial Algorithms





Longest Common Subsequence,Dynamic Programming

- We are given two sequences, and we must find the length of the longest subsequence present in both

- A subsequence is a sequence that appears in the same relative order, but not necessarily contiguous.

- “abc” & “abg” etc are subsequences of “abcdefg”.

- LCS for input Sequences “ABCD” and “ABED” is ABD, length of 3

So to start in your R console download the package

install.packages(“qualV”)

library(qualV)

This package has “qualitative methods for certain dynamic models”.

How can we do this in R?

Let's, say if we have the following subsequences, "abcbcdd" and "abcdefdd"

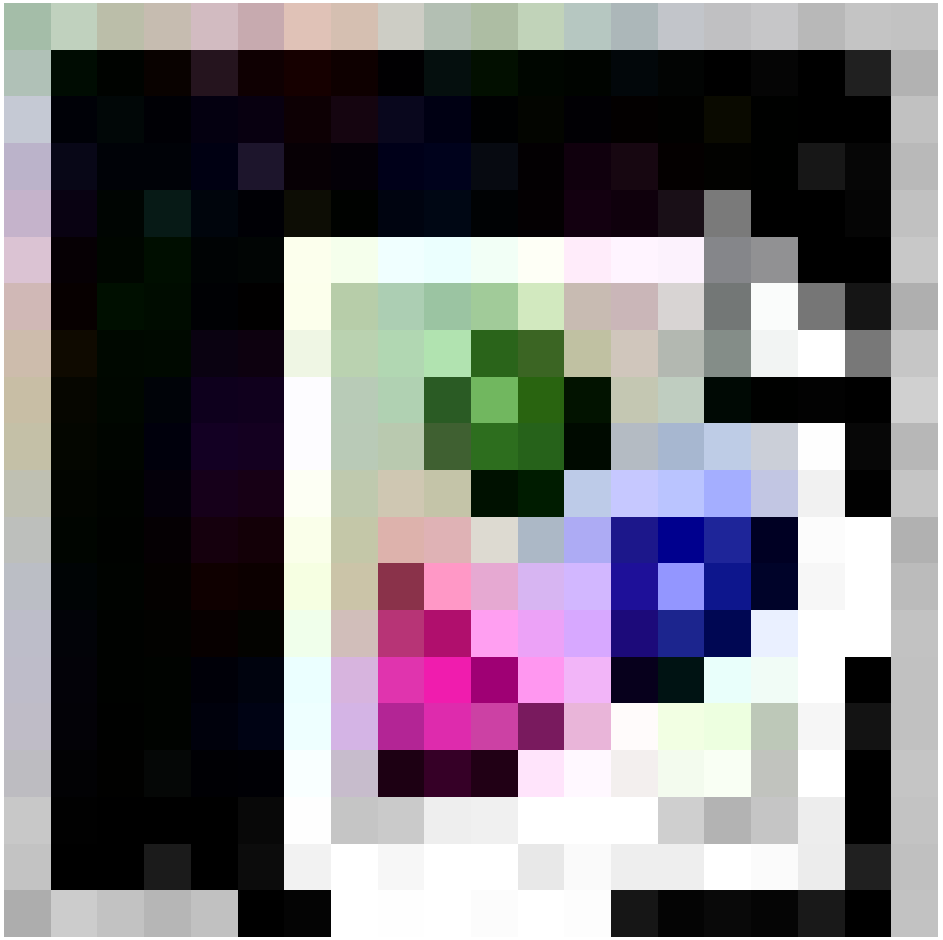
LLCS = length of LCS

LCS=longest common substring

QSI= quality similar index

va=One possible LCS vector of A

vb= One possible LCS of vector B



The Knapsack Algorithm

Description:We are given a set of items, each with a weight and value, and we must determine the number of each items to include in a collection, so that the total weight is less than or equal to given limit, and the total value is as large as possible.

This algorithm can be altered.

In the following example, we will be taking into consideration:

Retail Value & Price

The following scenario is taken from a Math 440A Lab:

A store is selling scientific equipment is having a going-out-of business sale. its merchandise is marked down. Its merchandise is marked down to varying degrees. You are employed as a buyer for another company that sells such goods. Your assignment is to acquire objects whose total retail value is as large as possible. You can spend up to $10,000. The available inventory is listed below. There is only one of each item available, what should you buy?

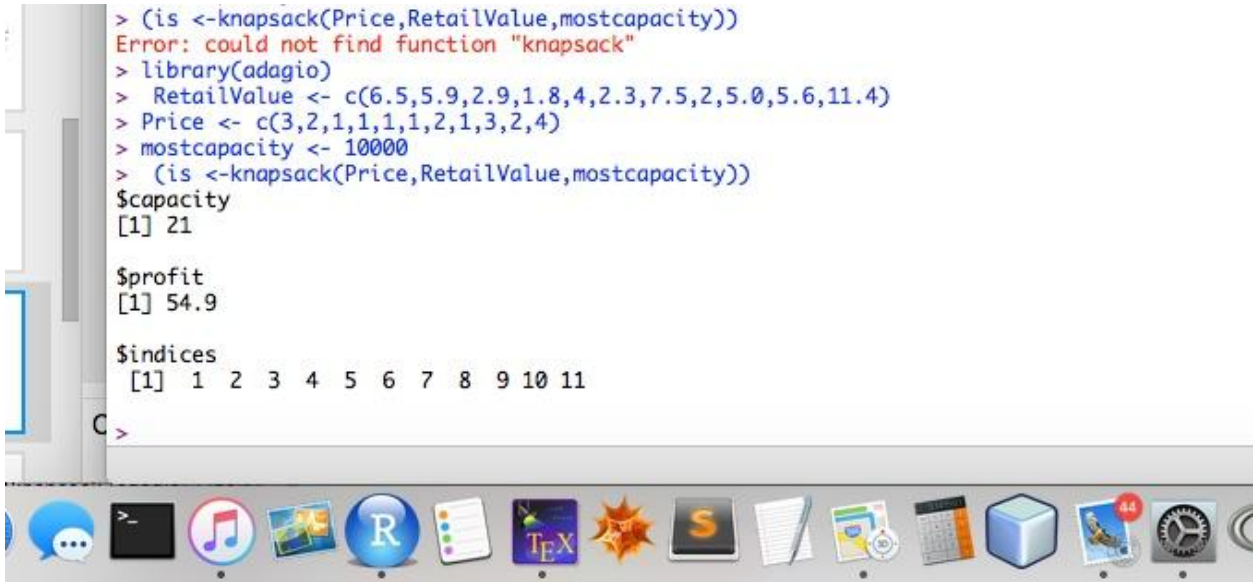
Item	1	2	3	4	5	6	7	8	9	10	11
RetVal	6.5	5.9	2.9	1.8	4	2.3	7.5	2	5	5.6	11.4
Price	3	2	1	1	1	1	2	1	3	2	4

So, this scenario can be applied to the idea of the KnapSack Question

most_capacity most amount of money to spend

#profit signifies Retail Price, most profit you can make out of the retail price

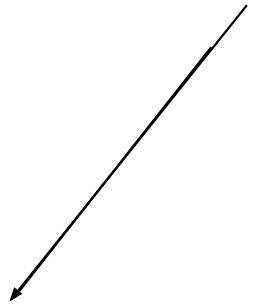
#indices ‹« define how many items we have



Stable Marriage Algorithm

It is a problem of finding a stable matching between two equally sized sets of elements given an ordering sequence. A matching is mapped from the elements of one set to the elements of the other

- The Stable Marriage Problem:
- Given *n* men and *n* women,
- where each person has ranked all members of the opposite sex in order of preference,
- marry the men and women together such that there are no two people of opposite sex who would both rather have each other than their current partners.
- When there are no such pairs of people, the set of marriages is deemed stable.



Propose/dispose alg

while(∃ man not engaged) {

man proposes to his highest ranked woman who has not rejected him

if(this woman is not engaged)

they get engaged

else

woman engages the preferred man, (and rejects him)

This is how the Gayle Shapley-Stable Marriage Algorithm Works:

Assume we have a set of given men and women and they have their preferences ranked from best to worst.

M_1	W_1	W_2	W_3	W_4
M_2	W_2	W_3	W_4	W_1
M_3	W_2	W_3	W_1	W_4
M_4	W_1	W_3	W_2	W_4

W_1	M_4	M_2	M_3	M_1
W_2	M_4	M_3	M_1	M_2
W_3	M_4	M_2	M_3	M_1
W_4	M_1	M_3	M_4	M_2

Example provided from
Professor Noga
Comp 496 ALG
Homework #1

There are a-lot of varieties of this algorithm

So, let's do the College Admissions Problem:

- In this scenario, we are taking into consideration College Admissions.
- So, we know colleges can of course accept more than one student.

#lets think of scenario, that 1000 students get matched to 400 colleges.

#Where each college has two slots. By construction 200 students will not get into any college.

#So, we draw the students and the colleges preferences.

install.packages(“matchingR”)

library(matchingR)

```
# set seed,random number generator, it can be saved & restored but not altered
set.seed(1)
# set number of students
nstudents = ?
# set number of colleges
ncolleges = ?
# generate preferences
uStudents = matrix(runif(ncolleges*nstudents), nrow = ncolleges, ncol = nstudents)
uColleges = matrix(runif(nstudents*ncolleges), nrow = nstudents, ncol = ncolleges)
# student-optimal matching
results = galeShapley.collegeAdmissions(studentUtils =  uStudents, collegeUtils =  uColleges, slots = 2)
str(results)

# check if matching is stable
galeShapley.checkStability(uStudents, uColleges, results$matched.students, results$matched.colleges)
```

End