Getting Started and R Nuts and Bolts

This week is all about getting started with R and learning some of the basic details of the language. If you haven't already installed R, you should go to the [R web site](http://www.r-project.org/) and download R for your platform (Windows, Mac, or Unix/Linux). Also, if you want, you can download [RStudio](http://www.rstudio.com/), which is a free interactive development environment designed for R that is very useful and we use quite a bit in the Data Science Specialization. I've made some videos to help you along with the installation process:

* Installing R on Windows
* Installing R on a Mac
* Installing R on RStudio (on a Mac)

Before you start using R, one key concept is the **working directory**. This is the directory/folder on your computer where you will store project files, data, and code. It's important that you tell R where the working directory is that you will be using so that it knows where to find the appropriate file (the working directory can be any directory on your computer). These videos tell you how to set your working directory:

* Setting your working directory (Windows)
* Setting your working directory (Mac)

Learning Objectives

By the end of week 1 you should be able to:

* Install the R and RStudio software packages
* Download and install the swirl package for R
* Describe the history of the S and R programming lectures
* Describe the differences between atomic data types
* Execute basic arithmetic operations
* Subset R objects using the "[", "[[", and "$" operators and logical vectors
* Describe the explicit coercion feature of R
* Remove missing (NA) values from a vector

Assessments

* Quiz 1 - 80% or better required to pass
* There is **no official graded programming assignment for this week**. However, we have developed a series of practice exercises to get you started with R. These exercises are implemented using the swirl package for R. **The swirl programming assignment is NOT required**.
* Hey everyone welcome to R Programming.
* This is the second course in the Data Science Specialization and, as
* the title suggests, we will be focusing on R as a programming language.
* So in this course, we'll kind of start with the basic
* building blocks of R kind of different data types in the
* very, kind of low level details and then we'll kind of
* move on to writing and formulating our programs or our scripts.
* This involves things like control structures and writing r
* functions and kind of doing some basic operations on data.
* And then, and then we'll talk about kind of profiling your code, some of the
* tools for debugging and kind of how to work,
* how to work through longer pieces of code.
* And then so after this course is done, I think you'll
* have a pretty solid grasp of R as a programming language.
* We're not going to cover every last feature of the R packages because of
* their and some of the other courses in the specialization you'll learn more about,
* for example, how to use the graphics system, how to make plots, how
* to use some of the packages for
* things like statistical inference and machine learning.
* And so that will be covered in other classes.
* So, the pur, purpose of this class is to really kind of
* get you into R programming, in
* particularly or if you're not very familiar
* with the language and to make sure you kind of kind of get a
* hand, get a sense of the on the of the nuts and bolts.
* So I hope you enjoy it.
* And I think after this course is done you'll be ready to move
* on to a bunch of, to mo, to the other courses in the specialization.

And then in this lecture, I'm going to give a little overview

and a very brief history of the R statistical programing environment.

Play video starting at 7 seconds and follow transcript0:07

So the very first question, I think is most obvious, is which is, what is R?

And the answer is actually quite simple.

It's basically R is a dialect of S.

Okay, so that leads to the next logical question, which is what is S?

Play video starting at 21 seconds and follow transcript0:21

So S was a language, or is a language that was developed by John Chambers and

at the now-defunct Bell Labs.

And it was initiated in 1976

as an internal statistical analysis environment, so

the, an environment that people at Bell Labs could use to analyze data.

And initially it was implemented as a series of

FORTRAN libraries to kind of implement routines that were

tedious to have to do over and over again,

so there were FORTRAN libraries to repeat these statistical routines.

Play video starting at 50 seconds and follow transcript0:50

Early versions of the language did

not contain functions for statistical modelling.

That did not come until roughly version three of the language.

So in 1988, the system was rewritten in the C language and to make it

more portable across systems and it began to resemble the system that we have today.

So this was version three.

And there was a seminal book the, called the Statistical

Models in S written by John Chambers and Trevor Hastie.

Sometimes referred to as the white book.

And that documents,

all the statistical analysis functionality that came

into the version, that version of the language.

Version four of the S language was released in 1998.

And its version, it's the version we more or less use today.

Play video starting at 1 minute 29 seconds and follow transcript1:29

The book Programming with Data, which is a reference for this course, is written by

John Chambers sometimes called the green book and

it documents version four of the S language.

So, R is an implementation of the S language,

that was originally del, developed in Bell Labs.

So, just a little bit more history here, in 1993 Bell Labs gave a corporation

called StatSci which became Insightful Corporation, an exclusive

license to develop and sell the S language.

In 2004, Insightful purchased the S language completely from Lucent.

So Bell Labs became Lucent Technology for $2 million, and became the current owner.

In 2006, Alcatel purchased Lucent

Technologies and it's now called Alcatel-Lucent.

Play video starting at 2 minutes 14 seconds and follow transcript2:14

So Insightful developed a product which was a implementation

of the S language under the product name S-PLUS.

And they built a number of fancy features into it for

example graphical user interfaces and all kinds of a nice tools.

that, so that's where the plus comes from in S-PLUS.

In 2008 the Insightful Corporation

was acquired a company called TIBCO for $25 million

dollars and that's more or less where it stands.

TIBCO still develops as PLUS, although in a

variety of different types of business analytic type products.

And it continues to this day.

So you can see the history of the language is a little bit

tortured because of the various corporate acquisitions

but it still survives to this day.

The basic fundamentals of the S language have

not really changed since 1998 and the language that existed in 1998 looks

more or less like we, like what we use today at least superficially.

And it's worth nothing that in 1998 the S

language won the association for repeating machinery software system award.

A very pretigious honor.

Play video starting at 3 minutes 16 seconds and follow transcript3:16

So in a document called the stages and the evolution of S, John

Chambers who was the original writer of the S language the, the original creator

kind of laid out his key principal with designing the S language.

And it's very important I think to to see this which is that basically.

Play video starting at 3 minutes 32 seconds and follow transcript3:32

They wanted to create an interactive environment where you

didn't have to think of themselves as programming, right.

Then he says then as the needs became

clearer and their sophistication increased, they should be

able to slide gradually into programming, when the

language and system aspects would become more important.

So the basic idea is behind the S language and

then later the R language is that people

would enter the language in an interactive environment.

Where they could use the lang, the environment, without knowing about any

sort of programming, or having to know very detailed aspects of the language.

So, they could use the environment to look at data, and do basic analyses.

And then when the environment, when they kind of

outgrew their environment, then they can get into programming.

They could get into learning the language aspects and

learning to develop their own tools and, and the system would very

kind of, would promote the kind of transition from user to programmer.

And so that was the basic philosopy of the S language.

Play video starting at 4 minutes 28 seconds and follow transcript4:28

So that's enough about S. we, let's go back to R.

So what is R about?

So basically, R is a relatively recent development.

In 1991, it was created in New Zealand by two gentleman named Ross Ihaka and

Robert Gentleman.

Play video starting at 4 minutes 42 seconds and follow transcript4:42

So, and they talked about their experience developing R in a paper

writ-, published in 1996 in the Journal of Computation and Graphical Statistics.

In 1993 the first announcement of R was made to the public.

1995, Martin Michler convinced Ross and Robert to use,

to license R under the GNU General Public License.

And we'll talk a little bit about, more about that in a second.

And that made R what

we call free software.

1996 a mailing list was developed, so there's two main mailing lists.

One called R-help, which is a general mailing list for questions.

And R-devel, which is a more specific mailing list

for people who are doing development work in R.

Play video starting at 5 minutes 22 seconds and follow transcript5:22

1997, what's called the R core group was formed.

And these contained a lot of, this contained a lot of the same people.

From the S-PLUS who developed S-PLUS.

And the core group, basically controls the source code for R.

So this, so the primary source code for R.

Can only be modified by members of the R core group.

However, a number of, people who are not in the core group have

suggested changes to R, and they have been accepted by the core group.

Play video starting at 5 minutes 49 seconds and follow transcript5:49

So, some of the features of R the first one, which was important

back in the old days, when people were still using S+ but the

syntax is very similar to S, which made it easy for S+ users to switch over.

This feature isn't quite so relevant today,

where most people generally go to R directly.

Play video starting at 6 minutes 6 seconds and follow transcript6:06

The semantics are superficially similar to S, in that it looks like it's S, but

in reality are quite different, but we'll

talk more about this in the future lecture.

One of the main benefits of R is that

it runs on any standard computing platform or operating system.

Mac, Windows, Linux whatever you

want even on your PlayStation 3 and there are frequent releases, so there

are annual major releases and often there are bug fixes releases in between.

There is a very active development going on and so things are happening.

Play video starting at 6 minutes 34 seconds and follow transcript6:34

The software the core software of R is actually quite lean.

Its functionality is divided into modular packages, so you don't

have to download and install a massive piece of software.

Whereas you can download

a very small piece of fundamental core, kind of

functions, and then add things on as you need them.

So it's graphics capabilities are very sophisticated and

give the user a lot of control over

how graphics are, are, are created, and in

my opinion are better than most stat packages.

It might even be the best for the mo- kind of a general purpose statistical package.

It's very useful for interactive work as I

said before, but it contains this powerful programming language.

For developing new tools,

so, it eases the transition from the user to the program.

And fundamentally, actually, for a language like this, is

that there is a very active and vibrant user community.

So the mailing lists at R-help and R-devel are very active.

There's many, posts per day, and there's also a

series on stack overflow where questions can be answered.

So, the user community is, is one of the most interesting aspects of R.

It's where all the R packages come from and it creates a lot

of kind of interesting features.

Play video starting at 7 minutes 40 seconds and follow transcript7:40

Of course one of the, probably the most critical feature of R is that it's free.

Both in the sense of free beer and the sense of speech.

So what I mean by that, is that it doesn't cost any

money so you can download the entire software from from the web.

And also it's free software, so I'm going to divert for

a second to talk a little bit about free software.

So, with free software there are four basic principles, right?

You have four basic freedoms that you have.

The freedom

zero is the freedom to run the program for any purpose, so you don't need.

There's no restrictions on how you can run the program or when you

can run the program or what you can or cannot do with it.

Play video starting at 8 minutes 15 seconds and follow transcript8:15

Freedom one is the freedom to study how

the program works and adapt it to your needs.

So this happens almost every day which is that

you can look at the source code for R itself.

You can make changes to it if you want.

You can, you may improve it or make a better version

of it. You can sell changes to it if you want.

You can do, you can modify the program any way you want and adapt it to your needs.

Of course, so you can look at the source code for this to get freedom one.

Freedom two is that you have the freedom to redistribute copies so you can help

your neighbor and so the idea is that you can give copies to other people.

You can sell copies.

You can do whatever you want with it.

Lastly you have the freedom to improve the program and release your improvements

to the public so the whole community benefits, so this is freedom three.

The idea is that when people make changes to the program they

can release them to the public so that everyone gets those changes.

And so these basic freedoms are outlined by the free software

foundation and you can see more about it at their website there.

Play video starting at 9 minutes 12 seconds and follow transcript9:12

So, there a couple drawbacks of R.

I won't go through all of them

and probably other people have many other complaints.

But there's some basic drawbacks which are one

that it's essentially based on 40 year old technology.

So the original S language developed in the 70s was based on

a couple of principles, and the basic ideas have not changed too much.

Play video starting at 9 minutes 34 seconds and follow transcript9:34

Since then and so as, one of the results of that for example

is that there is little built in support for dynamic or 3D graphics.

But things have improved, greatly and not on that front since the old days

and there's a lot of interesting tools now packages for doing dynamic or 3D graphics.

Play video starting at 9 minutes 49 seconds and follow transcript9:49

Another drawback of R that I, I hear a lot about is

that the functionality is based on

consumer demand and basically user contributions.

So if no one feels like implementing your

favorite message then that's your job to do.

And so you can't, there is no corporation,

there's no company that you can complain to.

There's no helpline that you can call to say

that, to demand a specific implementation or a specific feature.

If the feature's not there, then you have to build it.

Or at least

you can pay someone to build it.

Play video starting at 10 minutes 17 seconds and follow transcript10:17

Another drawback which is a little bit more technical is that the objects that

you manipulate in R have to be stored in the physical memory of the computer.

And so if the object is bigger, than the physical

memory of the computer, then you can't load it into memory.

And then therefore you can't do something in R with that object.

So there have been a lot of advancements to deal with this too.

Both in the R language and also

just in the hardware side there are computers now

that you can buy with tremendous amounts of memory.

And so some of those problems had been

resolved just by, kind of, improvements in technology.

But nevertheless, as we enter the, kind of, big data era where you have larger

and larger data sets, the model of loading

objects into physical memory can be a limitation.

And finally, I'll just say that R is not ideal for all possible situations.

And so many people,

I think, in ways is a good thing they have high expectations for R.

They expect it to be able to do everything.

Play video starting at 11 minutes 11 seconds and follow transcript11:11

But it doesn't do everything and so you should go into this knowing that fact.

Play video starting at 11 minutes 16 seconds and follow transcript11:16

So the basic R system is divided into

two, what you can think as two conceptual parts.

There is the base R system that you download

from a CRAN which is the comprehensive R archive network.

And that's kind of the go to place for all things R.

Then there's kind of everything else.

And so the base system contains what's called the base package which has all

the kind of low level fundamental functions

that you need to run the R system.

Play video starting at 11 minutes 42 seconds and follow transcript11:42

And then there are other packages contained in the

base system which includes for example util stats, data sets,

graphics and a bunch of other packages that are kind

of fundamental packages that more or less everyone might use.

And then there are a series

of recommended packages, so, boot for bootstrap, class for

classification, cluster, codetools, foreign, and a variety of other packages.

These are the commonly used packages, they may not

be critical packages, but they're commonly used by many people.

So all of these packages come with this,

the base R system that you download from CRAN.

Play video starting at 12 minutes 20 seconds and follow transcript12:20

Now, but there's much more than this obviously,

and on the, on CRAN, there are, right now

there are about 4,000 packages that have been

developed by users and programmers all around the world.

These packages are user contributed.

They're not controlled by the R core.

And they are uploaded to CRAN on a everyday on a periodic basis.

And the i-, and CRAN has a few, has a

number of restrictions and standards that have to be met in

order to get a package on to CRAN.

So, one of the nice things about CRAN is that there, that

the packages that you download have to meet a certain level of quality.

And so there have to be, for example there has

to be documentation for all the functions that are in

the package, and there has to be and they have

to make sure that they pass a certain number of tests.

So, so CRAN has, has a lot of different packages

written by users and the number is really increasing everyday.

So it's

very exciting to see all these packages on CRAN

and there, and to see new ones come up everyday.

Play video starting at 13 minutes 16 seconds and follow transcript13:16

There are also packages associated with the Bioconductor

project, which is a packaged, which is a

project designed to implement R software for, kind

of, genomic and, kind of, bio, biological data analysis.

and, of course, there are also all their packages

made that people make available on their personal websites.

And there's really no

reliable way to keep track of how many packages are available in this fashion.

So, there's really thousands of packages out there written by people.

That you can discover and use, to analyze data.

Play video starting at 13 minutes 47 seconds and follow transcript13:47

So there are a couple of documents that you can find on the R website.

As you're learning to use R, you then want to flip through some of these.

One is an introduction to R, which is a relatively long PDF

document now that kind of goes through the basics of how to use R,

how to use the language.

There's the Writing R Extensions manual which is really only

useful to read if you're thinking of developing R packages.

Which are these R extensions to the main system.

The R data import and export manual, which is useful

for getting R's data into R and the various different ways.

Play video starting at 14 minutes 19 seconds and follow transcript14:19

The R installation administration manual is, is most useful if you want to

build R from the source code, and I'll talk about that in another video.

And then the R internals manual.

Is is a really technical document for how R is designed.

How R is implemented at a very low level.

And it's not really for the faint of heart.

But if you're that kind of person, who wants to know how R

works at a very, very low level, this is the document for you.

Play video starting at 14 minutes 44 seconds and follow transcript14:44

So, I'm just going to end over here with a couple of texts

that are kind of standard or kind of classic texts in this area.

Of course

the books by John Chambers offers data analysis

and programming the data are both published by Springer.

Play video starting at 14 minutes 58 seconds and follow transcript14:58

And then there's two books by Bill Venables and Brian Ripley.

One is called Modern Applied Statistics with

S, and another one's called S Programing.

Although they have the, the, they talk about S in the title,

these books are all, are both very relevant for R programming too.

There's a book by Pinheiroand Base, which is Mixed Effects, Models in S and S-PLUS.

That's also

quite useful, for R programmers too.

And finally Paul Murrell who designed the R graphic system has written a

book called R Graphics and actually it's currently in its second edition right now.

So, a couple other resources, one is that

Springer, the publisher Springer has a series of books

called Use R, which is, which is a, a lot of very, kind of relatively short books.

How to use R for different types of topics, different application areas.

This is quite a nice

series of books that you may be interested in.

And there may be a book written for you particular area of application.

And there's a longer list of books on the R website.

So, that was a brief overview of R, and the history of how it kind of came to be.

and, starting with the next video, I'll start talking about the details of

the R programming language, and how we can use it to analyze data.

This video is about how to get help when learning to use R and taking this course.

Because of the size of the course and the number of students that are enrolled,

it's going to be difficult to be asking lots of questions on a one on one basis.

And so we're going to have to resort to a few other tools to get questions answered,

in particular, the discussion boards and generally through email.

And so with a certain type of way to ask questions that I hope to maximize

the chance of you getting the right answer or the answer that you're looking for.

And so, the main thing to remember is that asking questions via email is

a little bit different from asking questions in person.

Play video starting at 38 seconds and follow transcript0:38

You don't necessarily know that the people on the other side, the people that you're

asking, for example, on a discussion board or on a mailing list, you

don't necessarily know that they have the same background information that you have.

Furthermore, they may not know you personally and

so may not know kind of what you mean when you say certain types of things.

So that's kind of important to keep in mind when you're emailing questions

as opposed to when you're talking to someone in person.

Play video starting at 1 minute 2 seconds and follow transcript1:02

Keep in mind, of course, that elder people are very busy, and their time is limited,

and although they may be willing to help you by answering a question,

they may only have a certain amount of time to devote to answering that question.

Now I am here, of course, as the instructor, to help you in all

circumstances, but furthermore, I may not be able to answer all possible questions.

And so you're going to want to use the resources that you have available to you

in this course.

So, in your search for answers, there are a variety of things that you can do

on your own before venturing off to ask other people for the answer.

So, if you're going to be emailing a question to a forum or to a mailing list,

it's important that you search the archives of that forum for the answers.

So it's possible, and depending on the size of the forum,

almost very likely that someone has asked the same question that you're asking.

And if someone else has asked that question, and it has been answered,

then the answer is going to be in the archive of that forum.

So if the answer's already there, you've saved everyone a lot of time, including

yourself, if you search the archives for the forum and just find the answer.

Of course, the web is very large and has many answers, and

your first reaction when you have a question is to search the web.

For given the type of program you're using, for example, here we're using R,

there are many manuals that are available, and many answers may exist in the manual.

There's a frequently asked question,

or a FAQ that's on the R website that you can look for that contains many

questions that are commonly come up on the mailing lists and on the forums.

Another thing that you might want to try to do before go

venturing out to ask people for the answer is to play around with the problem and

try to find the answer by inspecting or experimentations.

Maybe if you have a function that's not working right, maybe change the inputs and

see if the outputs change or if the error message changes.

If you're lucky enough to have a skilled friend who knows something about R,

you can ask them personally, and it's usually easier to ask a person

on a one on one basis than to email a group of people in a forum.

Furthermore, and lastly, if you're a programmer,

you maybe able to find the answers you're looking for by reading the source code.

Play video starting at 3 minutes 8 seconds and follow transcript3:08

So all the things in the previous slide are useful things to do

on your own before venturing out to ask people questions.

However, if you don't find the answer, it's important to let

other people know that you did all those other things on the previous slide.

Because if, for example, the answer in the documentation for the programs, for

example, it's in one of the R manuals that someone who knows

the answer will usually respond by saying read the documentation or read the manual.

And then you just wasted one round of email.

because, you'll have to respond saying, I did read the manual, and

I didn't find the answer there.

And so, letting people know that you've done your homework, and

you've looked in a variety of places is very useful, and it saves a lot of time.

So, here's a very simple example of what might happen as you're using R.

So, here I'm loading the data source package, and

then I'm going to load the air quality data source from that package.

And then I'm going to want to run the correlation function

on this air quality data frame.

So immediately, I get an error,

it says error in cor(airquality) : missing observations in cov/cor.

So you might be wondering well, why am I getting this error, what does it mean?

So the first thing you can do is go to Google.

And in many circumstances, Google's going to be your friend.

And this is no different when you're learning a new programming language or

when you're learning R.

And so, the easiest thing to do is to take the error message that you get and

literally cut and paste it into the search box for Google.

When I search on that, I get a number of results.

And particularly, this third result looks very promising.

It looks like someone asked this question on the R help mailing list, and

Play video starting at 4 minutes 44 seconds and follow transcript4:44

so maybe we're clicking on that to see if someone replied with the answer.

Play video starting at 4 minutes 52 seconds and follow transcript4:52

So, when you ask a questions on a mailing list, so

assuming that Google wasn't able to help you out,

if you're going to ask a question on the discussion board or on the mailing list,

there are a couple things that you need to think about before you ask that question.

First, is it possible to reproduce your problem?

So, if someone else can reproduce your problem, it makes it a lot easier for

that other person to figure out what the solution is going to be.

And so if you can provide some code or a very simple example that will reproduce

your problem, this will be enormously useful to everyone else involved.

And if you don't do this, typically, the first response that you'll get will be,

can you please provide a reproducible example?

Second, it's important to understand what you expect the output to be,

because if your expectation is wrong, then of course, it may or

may not be an error, depending on what your expectation should be.

So what you expect the output to be will indicate kind of what the nature of

the error and what needs to be solved.

And then given your expectation, you need to say, well, what do you see instead?

So what was the thing that was unexpected that gave you the question?

Other information that's important to specify when you're asking a question

is the version of the product you're using, so for example,

what version of R are you using?

What version of the R packages you're using,

if it's specific to a given package?

Because often there may be legitimate bugs in older versions of R or R packages,

and your problem might be solvable if you just upgrade to the latest version.

So if you're using the latest version of R, it's important to mention that.

Play video starting at 6 minutes 24 seconds and follow transcript6:24

Sometimes it's important to know what operating system you're using, so

whether you're using a Mac, or Windows, or Linux, or some other UNIX machine.

Some problems can be traced to the type of operating system that you're using.

And depending on the question that you have,

there may be additional information that you need to provide.

Play video starting at 6 minutes 41 seconds and follow transcript6:41

So when you send an email to a forum or to a discussion board,

it's important to get as much useful information in there as possible.

And this includes the subject line for the email.

So there's a couple of examples of subject lines that could range in usefulness.

So the first one is probably the least useful, it just says Help!

Can't fit linear model!

So here, there's very little information here.

All I know is that there's some problem with linear models.

I don't know anything else about what the user's problem is.

So the second version is much better, it tells me that on R version 2.15.0,

the lm() function produces a seg fault,

which meaning that R crashes, with a large data frame.

And furthermore, it says I'm using Mac OS 10.6.3.

So here, I've got the operating system,

the version of the operating system, I've got the version of R,

I've got what function I'm using, and I've got a summary of what actually happened.

So just a little bit smarter than that would just be to reformat the message, so

that I specify what version of R I'm using, the function,

and then the version of the operating system.

And then so that gives me the context, and then after that, I can say

what the problem was, which in this case, was a seg fault on a large data frame.

So here, the important details are right away put in

the subject header before I even get to the body of the message.

Play video starting at 8 minutes 6 seconds and follow transcript8:06

So, a couple things that you definitely want to do when you're asking a question

on a forum or a mailing list, first is to describe the goal, not the steps.

So you may have many, many steps that you're going through, and

maybe one of those steps is causing a problem.

It's useful for other people to know what the bigger picture is in terms of what

you're trying to do, because for example, they might have a better idea about how to

go about achieving that goal, which may be faster or simpler, and

may work around whatever problem you're having.

So, describe the ultimate goal, and then talk about what the problems are.

And don't just narrow it down to the one little step that you're having

a problem with.

Be explicit about your questions.

So remember, provide details about what you're trying to do.

And you have to provide the minimum amount of information necessary, so

not the maximum amount of information, the minimum amount of information.

And so it's common to see on some mailing lists posts that lots of

output is produced, and that's not very helpful.

because volume doesn't really help you in terms of diagnosing the problem.

You need to know exactly, to narrow down kind of where the problem is going to be.

So, and of course a couple of things, being courteous never hurts anyone.

And promoting civility on mailing lists is always a nice thing.

And if you find the solution later on, it's useful for

everyone else in the community if you follow up with the solution and

explain kind of what the problem was and how you solved it.

Play video starting at 9 minutes 31 seconds and follow transcript9:31

A couple of things you definitely do not want to do when posting to a forum,

you don't want to claim that you've found a bug.

This happens all the time, and usually, I'd say 99 times out of 100,

it's not a bug, and it's just a misunderstanding about what

should have happened, so a mistake in the expectations of the user.

Play video starting at 9 minutes 49 seconds and follow transcript9:49

Grovelling is a substitute for doing your homework,

that's not usually looked well upon, and you definitely shouldn't do that.

Definitely don't want to post homework questions on the mailing list or forums,

and the reason is because people who write the homework questions are reading those

mailing lists and will be able to identify all homework questions without a doubt.

So we've seen them all,

don't bother trying to get the answers to your homework on mailing lists.

Don't ever email multiple mailing lists at once.

So this is a little bit annoying, because people will subscribe to different

mailing lists and will be getting your message more than one time.

It's important to figure out which mailing list is the most appropriate mailing list

for your question and then send the message to that mailing list or forum.

And then lastly, don't ask others to debug your code without giving some sort of hint

as to what the problem might be.

So it's very difficult when a person posts a long listing of code and

says there's a problem in here somewhere, I don't know where, please help.

It's better to kind of specify where you think the problem is and

what you're trying to do, so that everyone can save some time.

Play video starting at 10 minutes 56 seconds and follow transcript10:56

So this is just a very brief case study on using a recent post to

the R-devel mailing list.

So, the R-devel mailing list is an email list for

people who are doing development work in R, so either they're developing packages,

or they're making modifications to the R source code itself.

And so

the subject was, so you can see from the subject that it's going to be a problem.

The subject is large dataset- confused.

Play video starting at 11 minutes 20 seconds and follow transcript11:20

So right away, you know that there's not that much information here to go on.

It's not clear what the problem is.

In the message, it says, I'm trying to load a dataset into R, but

I'm completely lost.

This is probably due mostly to the fact that I'm a complete R noob, but

it's got me stuck in a research project.

So you have to ask yourself,

what do I know about this person's problem from reading this message?

And the truth is very little, and so the response was somewhat predictable.

Basically, the first person response said yes, you are lost.

And then there is a pointer to the posting guide, which everyone should read before

sending an email to the mailing list, and then also a list of manuals.

And so, here, you can see that one round of email is immediately wasted,

because probably the answer to the question was in one of the manuals,

and so, the user didn't specify whether he or she had already done that.

So, in terms of what went wrong with this little exchange, first, of course,

the question was sent to simply the wrong mailing list.

The R-devel mailing list is for development questions and for

more sophisticated programmers.

It's not really a mailing list for questions for,

as this person stated, noobs.

So that question really should have gone to the R-help mailing list,

where it would have been better received.

However, in addition to that, the email subject was very vague,

it was not clear what the problem was.

The question itself was very vague.

There was no reproducible example there.

There was not possible for

other people who are reading the email to reproduce what that person's problem was.

And there was no evidence that any effort was made to solve the problem.

So there is no evidence that they searched the web or checked the manuals or

experimented with the problem, or anything like that, or even looked at the forums.

And so, the end result was a complete recipe for disaster, and

it's likely that this person did not get the answer to their question.

Play video starting at 13 minutes 11 seconds and follow transcript13:11

So a couple places to turn for this course, first of all,

the class discussion board I think will be the most useful because your fellow

students can help you out there, and I can respond to you on the discussion board.

Outside of class, there's the R help mailing list, which I just described.

And you can post to this mailing list as well, and it's useful to, of course,

follow all the rules that we talked about just now.

And depending on what other products you might be working on,

there are other project-specific mailing lists for other types of software.

And so this talk was inspired by Eric Raymond's posting called,

How to ask questions the smart way, and I encourage you to read that.

It's much longer, it has a lot of other useful tips.

#### R Console Input and Evaluation

So once we start typing things into the R prompt, they we're going to be start,

we're going to start coding and doing calculation.

So the things that we type into the R prompt are called expressions.

So for example, the symbol,

which looks like a left-hand arrow and is actually the less than symbol,

followed by a hyphen this is what's called the assignment operator.

The assignment operator is what assigns a value to a symbol.

So, for example, in this first expression here the symbol that I'm creating is

called x, and the value that I'm assigning it is call, is 1.

And thi, so, and I used the assignment operator to create that.

So x is 1, is a, is an R expression.

And the next expression I'm going to print that value so print is a function.

And I'm passing it the symbol x so

that when I print out x I get its value which, in this case is 1.

So another thing to think about x is also considered a,

is an R object that is a numeric object that has one element.

So it's really a numeric vector where the first element is the number one.

Play video starting at 1 minute 6 seconds and follow transcript1:06

In the third expression here, you notice, I'm just typing X at the prompt and, and,

and when you hit enter what happens is it prints out the value of X.

So this is called, this is another way to

print out an object without explicitly calling the print function.

Play video starting at 1 minute 22 seconds and follow transcript1:22

So in the,

in this expression over here, I'm creating a new symbol called message, MS, MSG.

And I'm assigning it a value of the string hello.

All right? So now, this is a character vector.

Play video starting at 1 minute 34 seconds and follow transcript1:34

And the first element of this character vector is the string hello.

I could add other elements to this vector if I wanted to,

but they would all have to be character.

Play video starting at 1 minute 43 seconds and follow transcript1:43

So the grammar of the language determines whether an expression is

syntactically correct or not.

Or whether it's complete.

So for example by this type x followed by the assignment operator and

I don't have anything else, that's not a, that's not a complete expression and so

when I hit Enter nothing will happen because it's waiting for

the expression to be completed.

The other thing I've got here is this hash symbol here.

So this hash symbol here it indicates that everything to the right of

that is a comment.

And so the, the, the, the,

Play video starting at 2 minutes 13 seconds and follow transcript2:13

the R engine will ignore anything that happens to the right of that symbol.

So you can put things like comments or

notes to yourself in code and R will just ignore those comments.

Play video starting at 2 minutes 28 seconds and follow transcript2:28

So once you've typed in a syntactically valid and

complete expression at the prompt when you hit enter what happens is

that the expression is evaluated by the R engine.

And the result of that evaluation expression is then returned.

And so, so sometimes when you evaluate an expression,

nothing happens because there's nothing to really show.

And so, for example,

in the first expression here when I say x is assigned to be five.

So I'm creating an object called x.

It's a numeric vector and the first element's going to be five.

Now when I hit enter nothing happens because there's really nothing to show.

Play video starting at 3 minutes 2 seconds and follow transcript3:02

And so but now when I hit x and

I hit enter it prints out the value five, so it prints out the value of x.

So when I hit x, when, when I type in x and

I hit enter, that, and it prints out five, that's called autoprinting, and

so when you just type an object's name and hit enter.

R will by default autoprint the value of that object.

This is the same as calling the print function on that object which will

just print out the value of that object.

So you can explicitly print an object or you can auto print an object.

So this is, this sounds a little complicated but

it's really just the natural thing to do and it is what most people would expect.

Play video starting at 3 minutes 40 seconds and follow transcript3:40

You'll notice that when I print out the object x,

there's a little one in brackets here.

And you might be wondering what that is.

So, all that indicates is that, it,

it's telling you what element of the vector is being shown.

And this will make more sense when we have longer vectors to look at.

But all this is shame, saying is that the number five that you're seeing there

is the first element of the vector.

Play video starting at 4 minutes 6 seconds and follow transcript4:06

So for printing you'll see that here I'm creating an x an object called x and

it's the sequence one to 20, so

the colon operator here that I've used is what's used to create a sequence.

So, when I say one colon 20, that creates a sequence of one, two,

three, all the way up to 20.

So, now when I autoprint x in this case, you'll see I've got a long,

much longer vector here.

In this case, it's an integer vector.

And you'll see that the first line of the printout it has a one next to it,

because that's the first element.

And then the, the second line has a 16 in brackets because that's,

the first element of that line is the 16th element of this vector.

So it's all kind of straightforward but just that's how the printout works

#### Data Types - R Objects and Attributes

In this lecture we're going to start getting into the nitty gritty and

the details of R.

In particular I'm going to talk about different data types that are used in R

and some basic operations on those data types.

So first it's important to kind of get the language right correctly.

So all the things that you manipulate in R, all the things that we encounter in R,

are what might be called objects objects can be all different kinds,

can contain all different kinds of data.

But everything in R, is an object.

So the R has five basic atomic classes of objects.

So these are kind of the very low level or,

or basic classes of objects and they are character, numeric.

So these are like real numbers or decimal numbers.

integers, complex numbers, and logicals.

So logicals are just true a false type things.

Play video starting at 48 seconds and follow transcript0:48

And so the most basic object in R is called a vector.

And a vector conta-,

Can contain multiple copies of, for example, of a single type of object.

So you can have a vector of characters or a vector of integers, one thing you cannot

do with a standard vector is have mixed types of objects you cannot have a vector

of characters and numerics, or numerics and integers, or integers and logicals.

It, everything in a vector has to be the same class.

Play video starting at 1 minute 16 seconds and follow transcript1:16

Of course, with any great rule, there's always an exception, and

this, this one is no exception.

So, in this, with vectors, there's one type of vector that

can have multiple different types of classes, and that's called a list.

So a list is represent as a vector, so there's a se, it's a sequence of objects.

But each element of that vector can be a different,

can be an object of a different class.

So for example, you can have a list.

Play video starting at 1 minute 39 seconds and follow transcript1:39

That has a character, that has a numeric, it has a logical.

You can have a list that's inside the list and one element of

the list can be a data frame so, any element of the list can be anything.

And that's an, actually why what makes list so useful.

Play video starting at 1 minute 54 seconds and follow transcript1:54

So the list is the one exception to the ot to the.

General rule that a vectors can only contain elements of the same class.

Play video starting at 2 minutes 1 second and follow transcript2:01

So you can create an empty vector with the vector function.

And the vector function has two basic arguments.

The first argument is the class of the object, so

the type of object that you want to have in the vector.

And the second argument is the length of the vector itself.

Play video starting at 2 minutes 17 seconds and follow transcript2:17

Perhaps the most important type of object in R of course is the number.

So numbers in R are generally treated as what are called numeric objectsum, so

pretty much all numbers are treated as double number precision real numbers.

So, even if you are looking at a number that's like one or two, R thinks of those

numbers as numeric objects there is a way to explicitly say you want an integer and

you can specify the L subs, the L suf, the capital L suffix there.

So for example, if you just enter the number 1 in R,

that gives you a numeric object.

But entering 1 with a capital L next to it explicitly gives you an integer.

This distinction is not very important right now, but,

it will become important later.

There's also a special number called inf, which stands for

infinity and, and inf is like a real number it can be used in calculations and

you will get the expected result.

So, for example, if you take one, divide it by zero,

you'll get infinity and if you take 1 and divide it by infinity you'll get zero.

So, emphasis special number, and you can also have minus infinity, too.

Play video starting at 3 minutes 20 seconds and follow transcript3:20

There's another special value called NAN or Nan.

And this represents an undefined value so you can name it as not a number.

So, for example, if you take zero over zero that's not a number It's not defined

so you'll get a Nan back Nan can also be thought of as a missing value but

we'll talk a little bit more about missing values a little bit later so

another thing that, that comes with each object in R is an attribute.

So not every, object in R necessarily has attributes, but,

but they are, but attributes can be part of an object in R.

Some of the most common types of

attributes that we'll encounter are namesor dim names, or, or dimension names.

A dimension, so a matrix will have dimensions for

example it will have a number of rows and a number of columns if you

have a multidimensional array you'll have more than two dimensions.

Play video starting at 4 minutes 7 seconds and follow transcript4:07

The class of the object, so every object will have a class.

So for example, numeric objects their class is numeric and

integer objects, their class is integer.

Play video starting at 4 minutes 17 seconds and follow transcript4:17

Every object also has a length.

So for a vector it's quite simple the length of the object is

just the number of elements in the vector.

And then there may be other user-defined attributes or

metadatas which, so these are things that you can define separately, for

an object using various attribute functions.

There is a general function called attributes which allows you to set or

modify the attributes for an R object.

#### Data Types - Vectors and Lists

So the c function is another function that can be used to create vectors of objects,

and you can think of c as standing for

concatenate because it can be used to kind of concatenate things together.

So, for example, I can create an object called x by concatenating 0.5 and 0.6 and

that will give me a numeric vector of lenght 2 for the first element is .5 and

the second element is .6.

In the second example here, I've got a logical vector, we are concatenating

through true and false, so shorthand for true and false, you can use t and

f, capital T and capital F, so these 2 lines give you the same objectum,

you can create a character vector by concatenating a bunch of characters.

You can create integer vector by creating a sequence with colon operator, and

you can also create a vector of complex numbers where the i is a special symbol,

which indicates the imaginary part of the complex number.

Play video starting at 56 seconds and follow transcript0:56

So using the vector function you can also create,

a vector of a certain type and a certain length.

So here, I'm going to create a numeric vector of length 10.

Play video starting at 1 minute 7 seconds and follow transcript1:07

And by default it will initialize the vector, with a default value for

numeric vectors the default value is zero.

Play video starting at 1 minute 17 seconds and follow transcript1:17

So what happens if you take a vect you create a vector and

you mix two different types of objects and so the general it that is that r.

Will kind of create the least common denominator vector so, will not give you

an error but what will happen is that it will coerce the vector to be the,

the class that's kind of the least common denominator.

So here, in the first example, I've got in trouble concatenating number 1.7 and

letter a, so clearly these are not in the same class one is numeric, and

the other is character.

So the least common denominator here, is going to be character.

And so we're, so what you're going to get is that y is going to be

a character vector, where the first element is going to be the string 1.7 and

the second element's going to be the, the letter A so in the second example here,

I've got concatenating true, which is a logical, and a two, which is numeric.

Play video starting at 2 minutes 10 seconds and follow transcript2:10

And so what's going to happen here is that you're going to get a numeric vector and

the true is going to be converted into a number.

And so how's that happen, so and the, and by the convention in R

true is represented as the number one and false is represented as the number zero.

Play video starting at 2 minutes 27 seconds and follow transcript2:27

And so what you're going to get here, is a vector 1,2.

Lastly this last example here I am calculating the letter A, and

the logical true and so

here the least common denominator is again going to be character.

And so the vector that you end up with is a vector where the first element is A and

the second element is the string true, so T R U E.

It's not going to be illogical so

you need to be a little bit aware, of the types of coercion that can occur in our,

when you mix different types of elements in a vector.

And because you won't get an error, but, but

the coercion will happen behind the scenes.

Play video starting at 3 minutes 7 seconds and follow transcript3:07

that, in the previous slide we talked about kind of a implicit coercion that

occurs behind the scenes, but you can explicitly coerce objects from one

class to another using functions that usually start with the word as.

So for example, if you want to convert something to a numeric you can use

the function called as.numeric.

If you want to convert something into character you can use

the function as.character now if you apply these functions, so if you apply

as.numeric to a numeric vector nothing will happen so, here in this example I'm

starting off by creating an object called x which is a sequence of zero to six.

So this is going to, this is an integer sequence as you could see when I

call class on the object but I convert it into a numeric sequence.

Play video starting at 3 minutes 50 seconds and follow transcript3:50

And so I can call as.numeric on x, and you can see that it prints out 0,

1, through 6, which look like an integer object but

it's actually going to be numeric or I can convert it into a logical and so

I can say as.logical on it, and what happens?

Well, as you can see, the convention is that 0 is false.

Play video starting at 4 minutes 8 seconds and follow transcript4:08

And any number that's greater than zero is going to be true so here I've got a,

when I convert to logical I get false and then everything else is true when I call

as.character on X it takes all the numbers and, and converts them into characters.

So now I've got the string zero, the string one, two ect and finally when,

if I call as.complex this is fairly straightforward because you can all it

does is says that you have a complex number where all the imaginary components

Play video starting at 4 minutes 39 seconds and follow transcript4:39

are zero, now coercion we'll notice always doesn't work.

And when it doesn't work you get what are called NA values.

So non sensical coercion will result in NAs.

So for example if I take the vector ABC.

And call as.numeric.

Well there's really no way to convert the letters a, b, and

c to numerical variables so what you get is a vector of NAs and

plus a warning similarly if you call as.logical on x, you're going to get

a vector of NAs The next data type I'm going to talk about is the list.

Now I mentioned lists a little bit earlier in this lecture and the idea is

that they're, they're like a vector except that every element of a list could be a,

an object of a different class and so that makes lists very, very handy for

kind of carrying around different types of data.

Play video starting at 5 minutes 25 seconds and follow transcript5:25

And they're very useful in R and they become very common especially when in

conjunction with other types of functions that we're going to learn about.

So here I'm creating a list called x by using the list function which is a,

which can be used to construct the list.

And the first element is a numeric value, numeric object of one.

The second element is a character, letter a.

Third is illogical and the fourth is a complex number.

So this is not a problem with lists and

when I autoprint the list you'll see that it prints out a little bit differently It

doesn't print out like a vector because every element is different.

So you can see that in the double brackets here so the, the elements are indexed by

double brackets so the first element is the vector 1.

The second element is a vector with A.

The third element is a vector with true and the fourth element is a vector.

With the complex number 1 + (4i).

So lists are indexed you'll notice that el, elements of a list will have double

brackets around them elements of other vectors just have the single brackets, so

that's one way to separate a list from other types of vectors

#### Data Types - Matrices

Matrices are a special type of vector in R.

So there's, they're not a spe, a different, a separate class of objects.

But they're basically vectors that have a sp,

an attribute called the di, called the dimension.

And so the dimension attribute is an integer vector of length two.

Where the first number is the number of rows the matrix, and

the second number is the number of columns.

And so, if I can create a matrix empty matrix with the matrix function, and

I can explicitly say how many rows and how many columns there are.

This is not the only way to create a matrix, but it's one way.

And so when I auto print the matrix by typing the, the object m and

hitting Enter, you'll see that it'll show me that,

first of all the matrix is full, is initialized with NA values.

There's no values in here.

And you can see that there's two rows and

three columns and they're labeled by the numbers in the brackets.

If I call the dim function on m, it'll give me the dimension attribute,

which in this case it says there are two rows and three columns,

because the first rows are number rows and the second number, sorry excuse me.

The first number is the number of rows and

the second number is the number of columns.

If I call the attributes function on m, you'll see it

returns a list where the first element is the dim element and it has the vector 2,3.

So these are all aspects of a matrix which is

a vector that has a dimension attribute to it.

Play video starting at 1 minute 22 seconds and follow transcript1:22

Matrices are constructed column wise.

So you can think of the matrix taking a vector and

all the numbers are inserted into the matrix by, by column.

So the first column gets filled and then when you hit the number of maximum number

of rows, then the second column gets filled and the third column et cetera.

So if I create a matrix by taking the sequence 1 to 6 and then I say,

I specifies that it has two rows and three columns.

Well, how does this matrix get constructed?

So the way, it gets constructed column wise.

So the first thing it happens is it takes 1, 2, and

now there's only two rows, so it can only go to 2.

And then that makes the first column.

The second column is made up of 3, 4.

And then the third column is made up of 5 and 6.

Play video starting at 2 minutes 5 seconds and follow transcript2:05

You can also create a matrix by creating the dimension attribute on a vector.

So, for example, I can take, I can create a vector that's a sequence 1 to 10.

And then add the dimension attribute to it.

So here, I'm using the dim function, but

I'm assigning a value to the dim attribute of m.

So here I'm assigning the vector 2,5.

So what I'm saying is, I want to take this vector and

transform it into a matrix that is two rows and five columns.

Play video starting at 2 minutes 34 seconds and follow transcript2:34

And so now after I've done that, I've got a matrix m, which has two rows and

five columns and it's filled in the matrix column wise.

Play video starting at 2 minutes 44 seconds and follow transcript2:44

Another way to create a matrix, and

this is a common way, is by binding columns or binding rows.

Play video starting at 2 minutes 50 seconds and follow transcript2:50

And so you, this is column, column binding and row binding can be achieved by the fu,

the functions c-bind and r-bind.

So for example, suppose I have the two objects x,

which is sequenced from 1 to 3 and y, which is a sequence from 10 to 12.

If I cbind those two objects, then I'll get a, I'll get a matrix

Play video starting at 3 minutes 7 seconds and follow transcript3:07

where the first column is 1 to 3, and the second column is 10 to 12.

So this is kind of what you might expect would happen.

If I rbind those two objects,

then the first row will be 1 to 3, and the second row will be 10 through 12.

So cbind-ing and rbind--ing is another way to create a matrix.

#### Data Types - Factors

So factor is a special type of vector, which is used to create,

to represent categorical data.

So, and there's two types of factor, there is unordered or ordered, so

you can think of this as being, as storing data that are.

Have labels that are categorical but have no ordering, so for

example male and female.

Play video starting at 22 seconds and follow transcript0:22

Or you can have ordered factors which might represent things that are ranked.

So they have an order but they're not numerical for example you know,

in many universities you'll have assistant professors, associates professors, and

full professors.

Those are categorical but they're ordered.

Play video starting at 39 seconds and follow transcript0:39

So one, you can think of a factor as an integer vector where

each integer has a label.

So for example, you might, you can think of it as a vector as one two three,

where one represents you know, high, for example high value and

two represents a medium value and three represents a low value.

So you might have a, a variable that's called high, medium and low.

And underlying in R is represented by the numbers one, two, and three.

Play video starting at 1 minute 4 seconds and follow transcript1:04

so, factors are important because they're treated specially by modeling functions

like lm and glm which we'll talk about later.

But these are functions for, for, for fitting linear models.

Play video starting at 1 minute 15 seconds and follow transcript1:15

And factors are with labels generally speaking are better than using

simple integer vectors because the factors are, what are called self describing.

So having a variable that has values male and female is more

descriptive than having a variable that just, that just has ones and twos.

So for example, in many data sets you'll find that a var,

there will be a variable that's coded as one and two and it's, and it's not.

Easy to know whether that variable is really a numeric variable that only

takes values one and two, but the problem is that's not something that's coded in

the data set, so it's hard to tell.

If you use a factor variable then the coding for the labels is all,

is kind of built into the variable and it's much easier to understand.

Play video starting at 1 minute 57 seconds and follow transcript1:57

So factors can be created with the factor function, and

the input into the factor function is a character vector.

So here, I'm just creating a simple factor with the which has what, two levels, and

the levels are yes and no.

And so x is a factor, you can see what,

it prints out a little bit differently from a character vector,

in the sense that it prints up the value, yes, yes, no, yes, no.

And then it has a separate attribute which is called the levels.

And so the levels of this factor are no and yes, okay.

So there's only two levels.

I can, I can call table on this factor and

it will give me a frequency count of how many of each level there are.

So for example, it'll tell me there are two nodes.

And there's three yeses.

Now, the un-class function strips out the class for fa, for a vector.

So for example, I can, if I call un-class on x it'll,

it'll kind of bring it down to an integer vector, and you can see that underlying.

The factors represent as 22121 so, yes, it's coded as two and

no, it's coded as one.

Now it's not really essential for you to know this because you can

just treat the factor as being a vector of yeses and nos but it's used sometimes it's

it's useful just to know under, underneath kind of how factors are represented by R.

And so you see, it's really an integer vector with the attribute,

the levels attribute of no and yes.

Play video starting at 3 minutes 19 seconds and follow transcript3:19

The order of the levels in the factor,

can be set using the levels argument in factors.

So for ex, and sometimes this is important because in modeling functions and

when you include a factor variable this, this,

sometimes it's important to know what the baseline level is.

And so the baseline level is just the first level in the factor, and

the way this is determined by NR is critical.

It's determined using alphabetical order, so for

example, if I create a factor variable.

With the, with the elements yes and no, then the base line level with be

the first level that's encountered and because no comes before yes in

the alphabet then no will be the base line level and yes will be the second level.

Now this may not be something that you want you might want for

example a yes to be the base line level and

no to be the second level and then in that case you have explicitly tell r.

That yes is going to be the first level and

you can view that using the levels argument to the factor function.

So now when I print out the x object you see that the elements are still the same,

still yes yes no, yes no.

But the levels attribute is reversed.

because yes is the first level and no is the second level.

#### Data Types - Missing Values

So there's a special type of object that we haven't talked too much about yet.

And these are missing values.

Missing values in R are denoted by either NA or NAN which we talked about before.

NAN is used for undefined mathematical operations.

And NA is pretty much used for everything else.

Play video starting at 22 seconds and follow transcript0:22

And so, there's a function in R called is.na which is used to test objects to

see if they are NA.

To see if they are missing values in that object.

There's another function called is.nan which is used to test for NANs.

So, NA values can have a class, too.

So you can have missing integer val, values or

you can have missing character values or missing numeric values etc.

And so even though it looks like it's all NAs,

the NAs can have different classes potentially.

And then it's an NA, an NAN value is considered to be also NA, so for

example, an NAN value, a NAN value, is missing.

Play video starting at 59 seconds and follow transcript0:59

Is considered to be missing.

So, but the reverse is not true.

So an NA value is not necessarily, an NAN value.

I've got a few different types of missing values listed here.

So, here I created a vector x which is 1,2, NA, 10, and 3.

So, now, this is a numeric vector.

And the NA value in here's going to be a numeric missing value.

So when I call is.na on x, what it returns is a, is a logical vector.

And the logical vector indicates whether each element of the vector x

is missing or not.

And so, there's only one missing element in this vector, and so

that's the third element.

So you can see that the, that the logical vector that's returned.

The first two are false, the third is true, and the fourth and

the fifth are false.

So the, the, the element that's true indicated where the missing value is.

If I call is.NaN on this vector,

you'll see that vector that's returned is all false.

Because there aren't any NaN values, or

their aren't any MAN values in this vector so everything's false.

Of course, if I create a vector that has an end, a NAN value and an, and

an NA value in it.

You'll see that is.na returns true for both of them.

But is.nan only returns true for the for the value that's actually NAN.

#### Data Types - Data Frames

The last data type I'm going to talk about here is the data frame.

The data frame is a key data type used in R and it's used to store tabular data.

So of course, tabular data make up a lot of what we use in statistics.

Of course not all types of data are tabular.

But because so much data becomes a tabular form.

Data frames are very important in R.

So data frames are basically represented as a special type of list,

where every element of that list has the same length.

Right, so you can think of each column of the data frame as an element of the list,

and of course, in order to be a table, every column has to have the same length.

Play video starting at 38 seconds and follow transcript0:38

However, each column doesn't have to be the same type.

So the first column could be numbers, the second column could be factor,

the third column could be integers the fourth column could be logicals,

it doesn't matter what the different types are.

so, unlike matrices where, wh,

which have to store the same type of object in every single element of

the matrix, data frame can store your cla objects of different classes.

And so, data frames also have some special attributes.

First, the first special attribute is called a row name.

And so every row of a data frame has a name.

And this can be useful for kind of annotating the data.

So for example, each row re, might represent a subject enrolled in a study,

and then the row names would be the subject ID for example.

Play video starting at 1 minute 24 seconds and follow transcript1:24

however, sometimes the row names are not interesting, and, and, and

often you'll just use row names of 1, 2, 3, et cetera.

Play video starting at 1 minute 32 seconds and follow transcript1:32

Data frames can be created by calling most often calling the read.table, the read.csv

function and we'll get into that a little bit when I talk about reading data into R.

And you can also create a matrix from a data frame by

calling the data.matrix a function.

Now, you can't if you have a data frame that has many different types of objects,

and then if you coerce that into a matrix, it's going to force so

each object to be coerced so that they're all the same.

So you may get something that's not exactly expected.

Play video starting at 2 minutes 2 seconds and follow transcript2:02

So, data frames can be created besides using read.table or

read.csv, you can use the data.frame function and here I've

created a very simple data frame where the first the first column is called,

is the foo variable, and the second column is the bar variable.

The foo variable is an integer sequence from one to four, and

the bar variable is a logical vector with two trues and two falses.

So when I autoprint the data frame out you'll see the,

it prints out the two columns and here the row names since I didn't specify any

special row names, just defaults to 1, 2, 3, 4, because there's four rows.

And then when I call the nrow function on x,

I see that there's four rows in the ncall function, shows me that there are two rows

#### Data Types - Names Attribute

R objects can also have names.

So this not true for just data frames.

It's true for all r objects.

And this can be very useful for writing readable code and self describing objects.

So for example, I'm creating a vector that's an integer sequence 1, 2,

3 and by default, there's no name.

So when I call the names function on x, it gives me a null value.

However, I can, I can give a name to each element of the vector x.

So for example, if I, I can say the first element's called food,

the second element's called bar, and the third element's called norf.

So now when I print out my x vector, I get a vector 1, 2, 3 but

then each one has a name over it, which is the name I just specified.

Play video starting at 45 seconds and follow transcript0:45

And so when I call the names function I get the,

the names that are associated with each element of the vector foo, bar, and norf.

Play video starting at 55 seconds and follow transcript0:55

Lists can also have names.

And so for example here I'm creating a list with the list function

where the first element is called a, the second element is called b, and

the third element is called c.

And so when I print out the list, it prints out the names of each element and

the values associated with those names.

Play video starting at 1 minute 12 seconds and follow transcript1:12

Finally matrices can have names.

These are called dim names.

So here I created a matrix from the sequence 1 to 4.

It's a two by two matrix.

And so the, when, when I use the dim names function I pass it a list.

Excuse me, I assign it a list.

Where the first element of the list is the, is the vector of row names and

the second element of the list is a vector of column names.

So here I want to name the rows a and b, and I want to name the columns c and d.

So that's what I passed to the dim names function.

And now when I print out my matrix I can see that the row names and

the column names are labeled as I wanted.

#### Data Types - Summary

So, that's kind of a whirlwind tour of the different basic data types in R.

So far, we've talked about the atomic classes in numeric,

logical, character, integer, and complex vectors.

We talked about how vectors can only have elements of the same class and

the main exception to that is lists which can have elements of different classes.

There are factors which are used for, for coding categorical data, with ordered and

unordered data.

There are missing values that are represented by NAs, and NANs.

Data frames are used to store tabular data or each COM can be of a different class.

And finally, all our R objects can have names which mean usul,

which can be useful for creating self-describing data.

#### Reading Tabular Data

This lectures going to talk about reading and writing data in R.

So there's a few different types of ways you can do this and I want to talk

about some of the primary functions that use an R to read and write data.

So there are a few principle functions that we're going to talk about for

reading into R.

The first two are read.table and read.csv and these are for reading tabular data.

And they are probably the two most commonly used functions for

reading data into R.

These functions read text files that, that contain data that are stored in

kind of rows and columns type of format and return a data frame in R.

The function read lines is for reading lines of a text file so this, this can

be any type of file really, it just gives you text in a, as a character vector in R.

The source function is important for reading R code, so if you have R code, for

example functions or anything written get written to

a file the source function will read all that code into R.

Play video starting at 1 minute 0 seconds and follow transcript1:00

The dget function is also for reading R code files but it's for

reading R objects that have been dparsed into text files.

We'll talk a little more about this later.

The low and unserialized functions are for reading binary objects into R.

Play video starting at 1 minute 18 seconds and follow transcript1:18

So the analogous functions for writing data are write.table, writeLines, dump,

dput, save and serialize and those kind of pair up with their reading analog.

So, the read.table function is the most commonly used function for

reading data into R.

It's important that you know kind of, how the arguments work,

what the arguments are and understand what they mean.

So the first argument is pretty obvious, it's name of a file or

the name of a connection, which we'll get to a little bit later.

Usually you're going to give this a file name, it's going to be a string and

it's going to be a path to a certain file in your computer.

The header is a logical flag indicating whether the first line is a header line,

so if the first line for example it has all the variable names in it,

then that's not really a piece of data, that's just a line that has labels on it.

So you want to tell the read.table function whether the first line contains

the variable names or not or whether the line just, right away contains data.

Play video starting at 2 minutes 12 seconds and follow transcript2:12

The sep argument stands for separator.

it's, it's a string that indicates how the columns are separated.

So for example if you have a file that's separated by commas then the separator

has a comma.

You may sometimes files are separated by semicolons or by tabs or by spaces.

And so you want to tell read.table what the separator is going to be.

ColClasses is a character vector which indi wh, wh, which,

whose length is the same length as the number of columns the data set.

And the character vector indicates what,

what is the class of each column the data set.

So, for example, is the, if the first column is numeric and

the second column is logical, and the third column is a factor, et cetera.

And so the colClass is a vector, which is not required but it,

it tells the, it tells read.table what the class of the data is for each column.

End rows is the number of rows in the data set, this is not required but

it it can be used.

Comment.char is the character string that indicates what's the comments character So

the default, for example, is the pound symbol or the sharp symbol and anything

after, anything to the right of that symbol is ignored the comment character.

So you can specify other characters to be comment characters, and the lines,

lines of the file that begin with that comment character will be ignored.

Skip is the number of lines to skip from the beginning.

So sometimes there may be some header information or some non-data region at

the beginning of the file, and you want to skip right over that.

And so you can tell the read.table function to skip the say the first ten

lines of the first 100 lines and then only start reading data after that.

Play video starting at 3 minutes 42 seconds and follow transcript3:42

Last argument is strings as factors this defaults to true.

And the idea is that it,

the question is whether you want to encode character variables as factors.

So by default.

Anytime our read.table encounters a column of data that looks like

it's a character variable, it will call, it will assume that what,

what you mean to read in, is a factor variable.

If you don't me, mean to read this in as a factor variable,

then you can set strings as factors equal to false.

Play video starting at 4 minutes 11 seconds and follow transcript4:11

So for small and

kind of moderately sized data sets it has computers are going to get better and

better everyday, the definition of small and moderate is kind of growing.

But you can use read.table usually without specifying any of

the other arguments besides the file name.

So you can say read.table on, say foo dot text so this is just the name of the file,

and it will automatically take care of figuring out, you know, what the c,

classes of the different columns are, it'll figure out how many rows there are,

et cetera.

So you don't have to specify any of that information if you don't feel like it.

And, and then, and then, this will return an object here ca, that I call data, and

that would be a data frame.

So it'll automatically skip any lines and begin with the comment symbol.

It will figure out how many rows there are and, agai,

and again it'll figure out what type of variable is in each column of the table.

So, tell it you can, that you can tell R all these things and if you want to and

the reason you might do that is to make it run faster and more efficiently.

So with small and moderate size datasets its really not much advantage to doing

that because because it'll be pretty fast and pretty efficient as it is.

The read.csv function is identical to read.table, except for

the key differences that the, the default separator for the read.csv function is

the comma, whereas the default separator for read.table is the space.

So parti, so read.csv is useful for reading csv files,

this, this can usually, this stands for comma separated value.

It's usually something that you get from a spreadsheet program,

like Microsoft Excel or something similar to that.

So csv is a very common format that most spreadsheet types of

programs will understand.

The other thing that read.csv specifies is that it always specified header to

be equal to true.

#### Reading Large Tables

So with larger data sets of beyond the small to moderate,

then there are a couple of things you can do when reading in tabular data.

That will make your life a lot easier, and

more importantly it will prevent R from totally choking.

So first you should read the help page for read.table.

In fact, you should probably have it memorized.

There is a lot of key hints in that help page.

Lot of useful information.

And in my opinion not enough people read this help page carefully enough.

So that they can kind of recite in their sleep.

And if I, so there's a lot of so once you've read that you'll see

there's a lot of important information for kind of how to optimize read.table.

In particular for large data sets.

And so one of the things you're going to want to do

is to make a very rough calculation.

Of how much memory you need to store the data set you're about to read.

And so that way you can get a sense of well,

is there enough memory on my computer to store this data set?

Because if you recall correctly,.

Play video starting at 1 minute 0 seconds and follow transcript1:00

R will have to,

R is going to store your entire dataset in memory unless you do otherwise.

So when you call read.table or

read.csv, it's reading your entire dataset into the RAM of the computer.

And so you need to know,

roughly speaking, how much RAM this datasets going to require.

And we'll talk about how to calculate that in a second.

So another opti,

easy optimization you can say is if there's no comment lines in your file.

Then just set the comment char to be the comment.char is meant to be blank.

So just an empty quote there.

Play video starting at 1 minute 32 seconds and follow transcript1:32

The call classes argument is actually very important.

Because it whe, if you don't specify it then what R does by default is it

goes through every column of your dataset.

And tries to figure out what type of data it is.

Now, that's all fine, well I'm fine when the dataset is small to moderate.

But reading each of these columns and trying to figure out what type of data it

is takes time, it takes memory, and it can generally slow things down.

If you can tell R, what type of data, is in each column,

then R doesn't have to spend the time to figure it out on its own.

And so, it'll, it'll generally make read.table run a lot faster.

So you can save yourself a lot of time.

So if you, if, if you have a few columns in your dataset,

then then you can usually just say what, what the classes are.

But if you have, or if they are all the same, so for

example if all the columns are numeric.

You can just say, you can just set call classes equal to numeric.

And if you only sent, you give it a single value,

it will just assume that every column has that same value.

So if you just say numeric it will assume that every column is numeric.

Otherwise what you can do if you have a huge data set,

you can read in maybe the first 100 or first 1,000 rows.

By specifying the nrows argument.

And then going through each of the looping over each of the columns using sapply and

calling the class function.

So the class function will give you,

will tell you what class of data is in each column.

And then you can use this, and then you can save, store this information.

And then read the entire data set after by specifying the call classes argument.

Play video starting at 3 minutes 2 seconds and follow transcript3:02

So the n rows argument is actually very useful too.

It doesn't necessarily make R run any faster, but

it does help with memory usage.

And so, if you can tell R how many rows are going to be read in to,

to the, in to, in to R.

Then it can calculate the memory that's going to be required.

And not have to kind of figure it out on the go.

So even if you mildly overestimate how many rows there are in the data set,

that's okay.

Because it won't make a difference, it'll still read the correct number of rows.

Play video starting at 3 minutes 34 seconds and follow transcript3:34

So in general, when you're using R with large data sets, and

there's lots of large data sets out there nowadays.

It's useful to have a few things, a few bits of information on hand.

So, for example, how much memory does your computer have?

How much physical RAM is there?

These days in most computers will have on the order of a few gigabytes up

to many gigabytes of physical RAM.

What other applications are in use?

So are there other applications that are running on your computer that are eating

up some processor time or memory?

If you're on a multi-use system, are there other users logged into the system.

Are they using up some of the resources on the computer?

What is the operating system for your computer?

So, is it a Mac?

Is it Windows?

Is it Unix?

Is it Linux?

Is it something like that?

And then, also it's useful to know whether the O,

the operating system that you're running is 32-bit or 64-bit.

On a 64-bit system there, there, you'll generally be

able to access more memory if the computer has a lot more memory.

Play video starting at 4 minutes 31 seconds and follow transcript4:31

So if you want to do a rough calculation before you read in a table into R,

using the read.table or the read.csv function.

You can just do a very quick calculation.

So here is,

suppose I have a data frame here, with 1.5 million rows and 120 columns.

So this is not a particularly big data set but it's reasonable.

so, suppose that all of the nu, all the columns are numeric.

So, I don't have to worry about different types of data.

They're all, all the columns are numeric.

Play video starting at 4 minutes 57 seconds and follow transcript4:57

The question is how much memory is required to store this data frame in

memory, okay?

So, I can do a simple calculation.

So, the num, the number of elements in this data, in this data

frame is going to be 1.5 million times 120, right, because it's a square object.

And so that's, so that's the number of elements in the data frame.

Now, if it's a numeric all the data are numeric then each

number requires eight bytes of memory to store.

Because the, because the numbers are stored using 64-bit numbers and

there's eight bits per byte.

So that's eight bytes of memory per numeric object.

So that's going to,

so here's the number of bytes, now there's two to the 20 bytes per megabyte.

So I can divide that, the number of bytes by 2 to the 20, and

that's how many megabytes I got.

So it's got, I've got 1,373.29 megabytes.

And I can divide that again by 2 to the 10 to get the number of gigabytes,

that's going to be roughly 1.34 gigabytes.

So the, the raw storage for this data frame, is roughly 1.34 gigabytes.

now, you're actually going to need a little bit more memory than that to

read the data in.

Because there's a little bit of, overhead required for reading the data in.

And so, and so the rule of thumb, is to, is that you're going to need

almost twice as much memory to read this dataset into R using read.table.

Then the, then the object itself requires.

So if your computer only has, let's say two gigabytes of RAM eh, and

you're trying to read in this 1.34 gigabyte table.

You might want to think twice about trying to do it.

Because it, you're going to be pushing the boundaries of of memory that,

that is required to read this dataset n.

Of course, if your computer has like four or eight or 16 gigabytes of RAM,

then you should have no problem in terms of the memory requirements.

It will still take some time just to read it in just because it takes time to

read in all the data, but you won't be running out of memory.

So doing this kind of calculation is enormously useful when you're reading in

large data sets.

Because it can give you a sense of you know do I have enough memory.

Is the reason, if you grunt any errors,

you'll know whether the error is because of memory, running out of memory or not.

So I encourage you to do

this kind of calculation when you're going to be reading in large data sets.

And you, and you, and you know in advance kind of how big it's going to be

#### Textual Data Formats

There are other types of formats that you can save data in

beyond the tabular format, beyond, or the CSV file or text file.

These are also textual formats, but they are a little bit different for,

from the tabular data that we've talked about before.

And the two main functions for writing out data and f, are dumping and dputing.

So, and, and the idea behind these types of formats is they're text formats, but

they're not really, they're not really formatted in a way that's,

in the same as like a table because they contain a little bit more meta-data.

So data about, for example,

the type of the data in, in each class, object for example.

So if you, if you dump or dput a data frame.

It will include in the output, that the class of each column, of the data frame,

so that you don't have to specify it when you read it in.

And so the advantage of, of doing, using this type of mechanism to store data or

to read, or to read data, is that you don't have,

it's still a textual format, which can be useful, but it also contains metadata, so

that you don't have to specify it every single time you read it in.

Because that, if you don't, ca,

if the metadata do not get carried with the data set itself, then it, they ca,

they can get lost if you, if they get transferred somewhere else and

if you don't remember what the metadata are, for example the classes of

the different columns, then you kind of have to reconstruct that from scratch.

Play video starting at 1 minute 25 seconds and follow transcript1:25

So that's one advantage of using, using a function like dump or

dput to, to output data from R.

And similarly the, the, the, the functions for reading data using, fr,

that haven't been outputted from dump or dput are source and, dget.

Play video starting at 1 minute 42 seconds and follow transcript1:42

So in general, the textual formats are very nice formats for storing data

because, there's a number of different types of, different advantages to them.

First of all, they're editable, so you can, if you want to you can edit them.

I wouldn't say this is something that I would advice, but

because of you wanted something that's reproducible.

But, for example if something gets corrupted then you can look at the file

to see if it's possible to recover it.

So textual formats can be a little longer lived, if you're going to be storing data

for a long time, sometimes it's useful to, if it's possible to use a type of textual

format so that you can avoid problem, potential problems with corruption.

Textual formats can also work better if you're using like a version control

program, like subversion or git, where you're tracking changes between documents.

and, and those types of programs tend to be much more useful with

textual data rather than binary data, so that you can track changes meaningfully.

Play video starting at 2 minutes 39 seconds and follow transcript2:39

Textual formats adhere to the general kind of Unix philosophy, which is to

store all kinds of data, which generally stores all kinds of data in text.

But the one downside of textual formats is that they tend not to be space efficient,

so they tend to, they tend to take up a lot of space, and so,

it often need to be compressed.

Play video starting at 2 minutes 58 seconds and follow transcript2:58

So, d, the dput function takes an arbitrary R object, and

it will, use, it will take most types of R objects except for some more exotic ones,

and it will create some R code that will essentially reconstruct the object in R.

So here's I'm creating a small data frame,

it's got two columns, the first column is called A the second column is called B,

and then I'm going to dput this data frame.

And you'll see the out,

if you discall dput it'll just output the results to the console.

And you can see that what I've done is that.

What it does, it, it's re, it's constructed some R code.

For example, it's creating this list that has these two elements in it.

And you can see that each element has has the data that's in it.

And it has the names embedded here, it's got the row names here.

And it has the class of the object which, in this case, is the data frame.

And so, all the metadata here like the row names and

the names and the class are all included in the output.

Now, of course, you generally don't want to print this to the console,

that's not particularly useful, you probably want to save it to a file.

So you can dput the file to a file and then later on, you can read it into R

using dget, and when you dget the object, you will get this object and

you will see that it's, you have kind of reconstructed the object from before.

So the dput function,

essentially writes R code, which can be used to reconstruct an R object.

Play video starting at 4 minutes 18 seconds and follow transcript4:18

The dump function is a lot like dget however,

the difference is that dget can only be used on a single R object.

Whereas dump can be used on multiple R objects and so what you do is what

you pass a dump is the character vector which contains the names of the objects.

So here I created two objects one called x,

the other called y and when I pass the dump.

Is are the names of those objects?

The names are X and Y.

And I give it a file, that I want to store the da, the objects in.

And then I can remove them if I want to, but to read those objects back into R,

I can call the source function on that file and you'll see that the Y object and

the X object have been reconstructed.

#### Connections: Interfaces to the Outside World

So there are a variety of ways that you can interface between R, R,

wi, with the outside world.

And generally speaking there are functions that, that are used to kind of open up

the what are called connections to the outside world.

Play video starting at 21 seconds and follow transcript0:21

Usually you want to, the most common type of connection is to me,

is to a file, so for example if you want to read a file then you can,

you can create a file connection, you might want to for example o,

or read a compressed file, or that's a slight variation on that.

And most functions will do this in the background without you having to

know what's going on.

So for example, when you call read.table with it and

you pass it the name of a file, what it does behind the scenes is it

opens up a file connection to that file, and then reads from that file connection.

The connection can be made to other types of objects too.

For example, you can open a connection to a webpage using the URL function.

And so, when you open a connection to a webpage,

you can read data from that webpage using the URL connection.

And so, the idea behind the connection interface is, is that it kind of,

that it abstracts out.

The mechanism for connecting to different types of objects that are external to R,

whether they be files, or webpages, or whatever.

So the file function is the function that opens a connection to

a standard uncompressed file.

So this, this can be useful for

text files, for, for reading in other types of text files.

Gzfile and bzfile, are used for opening connections to compressed data files.

So gz file i, are, is used for files that are compressed with the gzip algorithm and

bz files used for, is for

opening connections to files compressed with the bzip2 algorithm.

Files that are compressed with gzip usually have a gz extension and

files compressed with bzip2 usually have a bz2 extension.

Play video starting at 1 minute 55 seconds and follow transcript1:55

So the file function here has a few arguments,

the description argument is the name of the file and there's a flag that's called,

that goes to the open argument and you have to know what the codes are, but

basically r is for reading, w is for writing, a is for appending, and then rb,

wb and ab are for reading, writing, and appending on binary files.

The other options for file are not particularly important at this time.

Play video starting at 2 minutes 17 seconds and follow transcript2:17

So, connections can be very powerful and they can let you navigate files and

other external objects in a more sophisticated way than just,

like, reading the whole thing, for example.

Play video starting at 2 minutes 27 seconds and follow transcript2:27

And generally you don't have to deal with the connect interface in many cases, but

sometimes it's useful.

So for example, so here I've got a simple example or opening a fi, a file

connection to some file called foo.text, I'm going to open it for reading.

I can call read.csv on the connection, and

that by default will just read the entire file then I can close the connection.

So that three line process is the same as just calling read.csv on the file.

So in this case there was no need to use the connection to read the file.

However, sometimes a connection can be useful if you want to read parts of

a file.

So for example,

here I've got the readLines function which just reads lines from a text file.

And I'm going to open up this words.gz file.

So, this is a file that has words in it for it's like a dictionary file.

And it's compressed using the gz, the gzip algorithm.

So I'm going to be using the gz file function to open a connection to that.

And I'm just going to read the first ten lines.

So now I'm going to re,

use this connection, and to read the first ten lines.

And here, the first ten lines are printed out here

Play video starting at 3 minutes 28 seconds and follow transcript3:28

as you can see these are the first top ten words in the file.

And similarly, write lines is a,

is a function that can be used to write out lines of text to a file.

And each, and what you do is pass write lines of character vector and

each element of the character vector becomes a line in the text file.

Play video starting at 3 minutes 46 seconds and follow transcript3:46

You can also use readLines to read elements from a web page, so for example,

you can use the URL function to create a connection to a website, so

this website here is the Johns Hopkins School of Public Health website.

And I'm going to open the connection there for

reading, and then I'm going to read lines from this connection.

Play video starting at 4 minutes 5 seconds and follow transcript4:05

And so and I'm, and then and so the lines of text that come

from the connection are going to be stored in this character vector x.

So when I look at the first couple of lines from x you can

see that it looks like HTML which is kind of what you would expect.

And so the URL function is useful for

creating a connection to a kind of a non file object.

and then using read.lines is useful to read the text from that connection.

So this is another way to read data beyond using functions like

read.table or read.csv

#### Subsetting - Basics

[NOISE] I'm going to continue to talk about data types, and

basic operations in R.

In particular in this video I'm going to talk about subsetting objects in R.

So there are a couple of different operators that you can use,

to extract subsets of diff, of R objects.

There's the single bracket.

Sorry, the single square bracket.

The double square bracket,

which we saw in the previous video, and there's the dollar sign.

So the sing, the basic kind of principles to remember here is that the single

square bracket always returns an object of the same class as the original.

So the subset a vector, you're going to get back a vector.

If you subset a list, you're going to get back a list.

Any time you used the single bracket operator to subset an object,

you'll get the same, an object of the same class back.

Play video starting at 48 seconds and follow transcript0:48

si, furthermore the single bracket operator can be used to

select more than one element of an object.

Play video starting at 56 seconds and follow transcript0:56

With one ex, exception that we'll get to later.

But double bracket operator is used to extract elements of a list or

a data frame.

It can only be used to exa, extract a single element and.

Of that object, either the list or the data frame.

And the class of the returned object will not necessarily be a list or a data frame.

So the idea with the double bracket operator is that, remember that lists can,

can, can hold things that are of many different classes.

They don't all have to be the same.

So, the first element might be a vec a numeric vector, the second element might

be a data frame, the third element might be a complex vector, et cetera.

And so when you use the double bracket operator to extract an element of a list,

the oh, the object that comes back maybe,

may not be a list, it may be an object of a totally different class.

So that's what the double brack operator is useful for.

The dollar sign is used to extract elements of a list, again of a list or

data frame that have a name.

Very similar objects can have names and the reason,

one of the reasons you've used names in an object is so

that you can reference elements of the object by the different names.

Play video starting at 1 minute 59 seconds and follow transcript1:59

Otherwise the, the semantics of the dollar sign are similar to the double bracket in

the sense that when you use the dollar sign to extract an element of

an object it may or may not be of the same class as the original object.

So, here is the first,

the first example, a very simple vector, a character vector called x.

And and

I'm going to use the single bracket operator to extract the first element.

So here, what I get back is a,

is another character vector with the single element a in it.

If I, if I use, if I try to extract the second element of x,

what I would get returned back to me is a character vector with the element b in it.

I could also extract a sequence of elements so

if I say, If I, If I want to get the first four elements of x I can cre,

construct the sequence one through four and then I get a, b, c, c.

So in these three examples here what I've done is I,

I, is I subset the vector x using a numeric index so the numeric index is one,

two or the sequence one through four.

The oth, another type of index that you can use is the, is a logical index.

So, in this next example here, I'm going to subset the vector x and I want,

I only want all the elements were, that are greater than or

equ, sorry, that are greater than the letter a, right?

So, you might, it might seem strange to you that

I'm using the greater than sign with letters instead of numbers but

there is a lexicographical ordering to the letters, and

all the letters that are greater than a are letters like b, c, d, e, et cetera.

So what I get returned to me is a character vector that only

contains the letters that are greater than a.

So, here I've got b, c, c, and d.

The other thing I can do, is I can create a logical vector,

which here I call u, which is just the it's a, it's, it tells it's a true or

false vector, which tells me,

which tells me which elements of the vector x are greater than a.

So, if I print out u here I can see that the, the first element is equal to a, so

it's not greater than a.

Then, the next four are greater than a,

but then the last element is equal to a, so again, that's false.

And so, I can subset the vector x with this u vector, and

then I get out all the elements that are greater than a.

So there are two types of indices that I use here,

one, the first type with the numeric index.

And the second type was the logical index.

#### Subsetting - Lists

So subsetting a list is a little bit different.

Because you can use the double bracket or the dollar sign operator.

You can also use the single bracket operator.

So here I've got a list, the first element is called, is a named element called foo.

That's an ind, and it's a sequence 1 through 4.

And the second element is named bar, and it's the number 0.6.

So this is a list of two elements in it.

I can extract the first element by using the single square bracket.

And I get, when I, remember the single square bracket always returns the element

that's the same class as the original.

So if x is a list, than x bracket 1 is going to be a list too.

So what I get back is a list that has element call foo,

which is a sequence 1 through 4.

Now if I use, so if I use the double bracket then if I said x double bracket 1,

what I get back is just a sequence, 1 through 4.

So, so the difference here is that in the first example, I got a list that contained

the sequence 1 through 4, and in the second example, I got just the sequence.

That's the difference between the single bracket and the double bracket operator.

Play video starting at 1 minute 3 seconds and follow transcript1:03

In the next example here I'm using a dollar sign.

And so I'm saying, x dollar bar.

And that what that mean is that,

that gives me that element that is associated with the name bar.

So in that case it's the, it's a single number 0.6.

I can also use the double bracket operator and pass in a string.

So x double bracket quote bar is the same as doing as x dollar bar and

it just gives me 0.6.

If I use the single bracket with the name,

I can say x bracket quote bar, that gives me a list with the element bar in it.

So remember, because the single bracket always returns a list if I'm

going to subset a list.

So the nice thing about being able to subset an element using its name,

is that you don't have to remember where it is in the list.

So if I couldn't remember whether bar was the first element or

was the second element, I don't have to remember whether, what,

where it is in order to use the numeric index.

I can just use its name, and then I don't have to,

then it will automatically extract that, extract that element from the list.

Play video starting at 2 minutes 1 second and follow transcript2:01

If you want to extract multiple elements of a list then you need to

use the single bracket operator.

So for example, if I want the third, the first and the third element here,

in which case, which is the foo and the baz element, I can pass a,

a vector, a 1, 3, the numeric vector 1, 3 to x using the single bracket operator.

And that returns to me a list with the elements foo and the elements baz.

Play video starting at 2 minutes 23 seconds and follow transcript2:23

So that's how I extract multiple elements of a list.

There's, you cannot use the double bracket or

the dollar sign operators when you only extract multiple elements of a list.

Play video starting at 2 minutes 34 seconds and follow transcript2:34

The nice thing about the double bracket operator, which is different from

the dollar sign, is that you can use the double bracket operator to, to,

to index it a list, where the index itself was computed.

So, notice that when I used the dollar sign before, I had to,

I actually typed out the word bar.

I had to type out the name of the object.

Sometimes the name of, sorry the name of the element.

But sometimes the name of the element is actually the result of some computation.

So for example here I've got a list with three elements, foo, bar, and baz.

And then I create a variable called name which is actually the string foo.

So if I use the double bracket operator on this variable here.

Play video starting at 3 minutes 13 seconds and follow transcript3:13

Notice that the there's no element in the list that has the name, name in it.

Play video starting at 3 minutes 19 seconds and follow transcript3:19

But there is an element in the list that has the name foo in it.

So now when I, when I pass this variable called name into the double

bracket operator, it returns me the, the element that was associated with foo.

because that's what the value of the name variable is.

So if I can, if I compute the index that I want to use,

then I have to use the double bracket operator.

If I use the dollar sign, then it's going to literally look for

an element of the list that's, that has the word name associated with it,

and that of course doesn't exist in this list.

Play video starting at 3 minutes 47 seconds and follow transcript3:47

So to use the dollar sign I need to use a literal symbol.

Play video starting at 3 minutes 54 seconds and follow transcript3:54

Now, the double bracket operator can take an integer sequence in as,

rather than a single number, and

the way you can think of this is that it kind of recurses into the list.

So if you look at this current list I've got here,

with the first element a is another list which has elements 10, 12 and 14.

So suppose I wanted to extract the number 14.

Well, that's really the third element of the first element, right?

So it's the third element of the list,

which happens to be the first element of the other list.

And so I can extract the 1, 3 element term by passing the vector 1,

3 to it to the x list using the double bracket operator.

And that's equivalent to kind of doing this double sub-setting of one and three.

I can also extract the first element of the second element by use,

by passing the integer vector 2,1 to get 3.14