Introduction to R

Main objective <- Review the main features of R (remember there is a lot of documentation packaged in R itself!)

Berenica Vejvoda, Data Librarian
bvejvoda@uwindsor.ca
libdata@uwindsor.ca
October 25, 2024
Leddy Library, University of Windsor

Introduction to what is R?

R is an **open-source programming language** and software environment **for statistical computing** and graphics that is supported by the R Foundation for Statistical Computing.

R programming was created for statistics and data analysis and is used in academic and research fields.

Over 25 years old now!

Similar to the S language developed by Bell Labs in 1976 (48 years ago!). R is considered a different "implementation" of S. Unlike S (MathSoft), R is open source!!!

Along with Python, R is used widely in the field of data science, which consists of statistics, machine learning, and domain expertise or knowledge. In contrast, Python is a general application programming language.

R was developed with statistics in mind.

R packages give R its functionality

- 10,000+ packages available for installation (packages are similar to extensions or apps for other products like Google or Apple).
- Each package serves a purpose and has specific commands you can use.
- Packages are user-created programs which can be used to run a specific task or set of tasks.

R supports a very wide variety of statistical techniques:

- Linear modelling techniques such as multiple linear regression (MLR) (predictive modelling)
- Non-linear modelling techniques such as non-linear regression
- Classical statistical tests such as:
- Nominal: Chi-Square (e.g. # of correct responses)
- Ordinal: Factor Analysis
- Interval or Ratio: t test, ANOVA (linear relationship between two continuous variables)
- Pearson's r (linear relationship between two variables)
- Time series analysis
- Clustering (partitioning data into the same class e.g. text data mining)
- Graphical techniques – especially great if you need to create a large number of plots (very efficient compared to Excel)

Why R?

- Free
- Works on all operating systems
- Excellent and efficient graphical capacities
- Large and active community
- Many methods not available in other commercial software
- Good integration between programming language and statistical functions
- Good integration with a number of database systems and other languages
- Scripts written in R can be used on different operating systems,

including Linux, Apple, and Windows, as long as the R interpreter is installed.

What is RStudio?

RStudio is an IDE (integrated development environment) for R

Open source

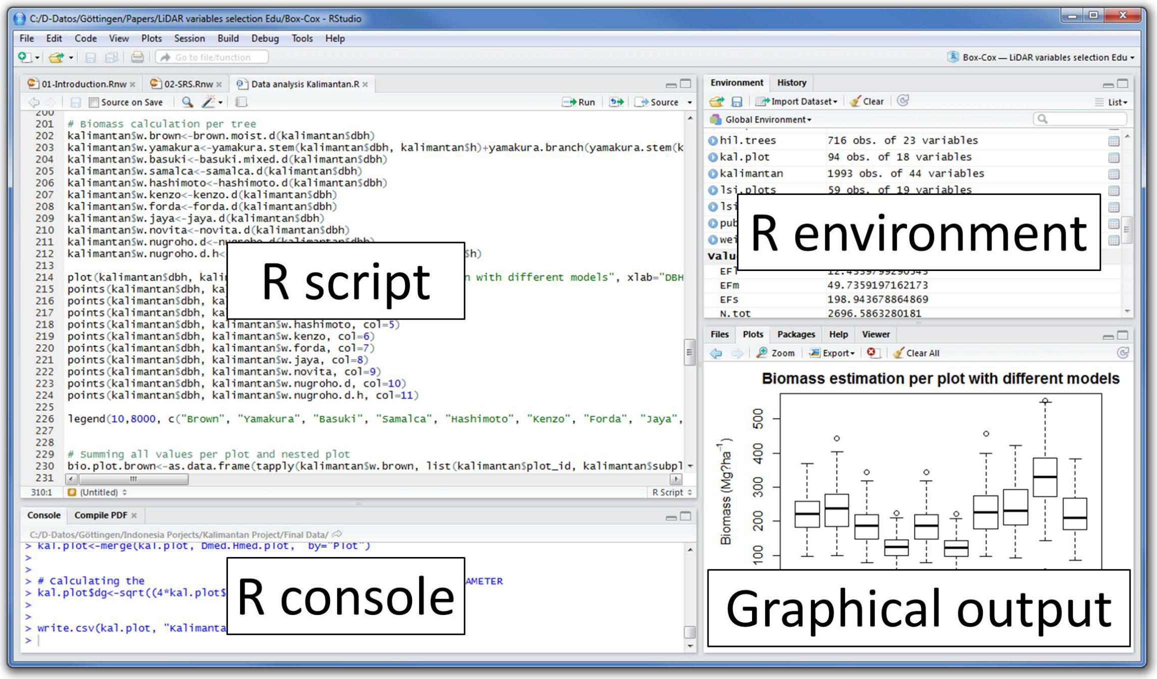
Need to install R first

Can use R without R Studio but you can't use R Studio without R

Most generally, it can help you work more efficiently by providing further functionality

RStudio makes it easier to:

- view datasets as spreadsheets
- see the results of your script (e.g. plot)
- change your plot in the plot display without re-running your code (e.g. size)
- install packages (collections of R functions, data and compiled code)
- set your working directory and access your files
- can help you work more efficiently by providing further functionality


C:/D-Datos/Göttingen/Papers/LiDAR variables selection Edu/Box-Cox - RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Box-Cox — LiDAR variables selection Edu
01-Introduction.Rnw
02-SRS.Rnw
Data analysis Kalimantan.R
Source on Save
Run
Source
Environment
History
Import Dataset
Clear
List
Global Environment
Files
Plots
Packages
Help
Viewer
Zoom
Export
Clear All
Biomass estimation per plot with different models
Biomass (Mg?ha^{-1})
100
200
300
400
500
310:1
(Untitled)
R Script
Console
Compile PDF
C:/D-Datos/Göttingen/Indonesia Porjects/Kalimantan Project/Final Data/
R script
R environment
R console
Graphical output

```
# Biomass calculation per tree
kalimantan$w.brown<-brown.moist.d(kalimantan$dbh)
kalimantan$w.yamakura<-yamakura.stem(kalimantan$dbh, kalimantan$h)+yamakura.branch(yamakura.stem(k
kalimantan$w.basuki<-basuki.mixed.d(kalimantan$dbh)
kalimantan$w.samalca<-samalca.d(kalimantan$dbh)
kalimantan$w.hashimoto<-hashimoto.d(kalimantan$dbh)
kalimantan$w.kenzo<-kenzo.d(kalimantan$dbh)
kalimantan$w.forda<-forda.d(kalimantan$dbh)
kalimantan$w.jaya<-jaya.d(kalimantan$dbh)
kalimantan$w.novita<-novita.d(kalimantan$dbh)
kalimantan$w.nugroho.d<-nugroho.d(kalimantan$dbh)
kalimantan$w.nugroho.d.h<                                  $h)

plot(kalimantan$dbh, kali                                  n with different models", xlab="DBH
points(kalimantan$dbh, ka
points(kalimantan$dbh, ka
points(kalimantan$dbh, ka
points(kalimantan$dbh, kalimantan$w.hashimoto, col=5)
points(kalimantan$dbh, kalimantan$w.kenzo, col=6)
points(kalimantan$dbh, kalimantan$w.forda, col=7)
points(kalimantan$dbh, kalimantan$w.jaya, col=8)
points(kalimantan$dbh, kalimantan$w.novita, col=9)
points(kalimantan$dbh, kalimantan$w.nugroho.d, col=10)
points(kalimantan$dbh, kalimantan$w.nugroho.d.h, col=11)

legend(10,8000, c("Brown", "Yamakura", "Basuki", "Samalca", "Hashimoto", "Kenzo", "Forda", "Jaya",


# Summing all values per plot and nested plot
bio.plot.brown<-as.data.frame(tapply(kalimantan$w.brown, list(kalimantan$plot_id, kalimantan$subpl
```

```
hil.trees      716 obs. of 23 variables
kal.plot       94 obs. of 18 variables
kalimantan     1993 obs. of 44 variables
lsi.plots      59 obs. of 19 variables
lsi
pub
wei
Valu
  EF
  EFm          49.7359197162173
  EFs          198.943678864869
  N.tot        2696.5863280181
```

```
> kal.plot<-merge(kal.plot, Dmed.Hmed.plot, by="Plot")
>
>
> # Calculating the                              AMETER
> kal.plot$dg<-sqrt((4*kal.plot$
>
>
> write.csv(kal.plot, "Kalimanta
>
```

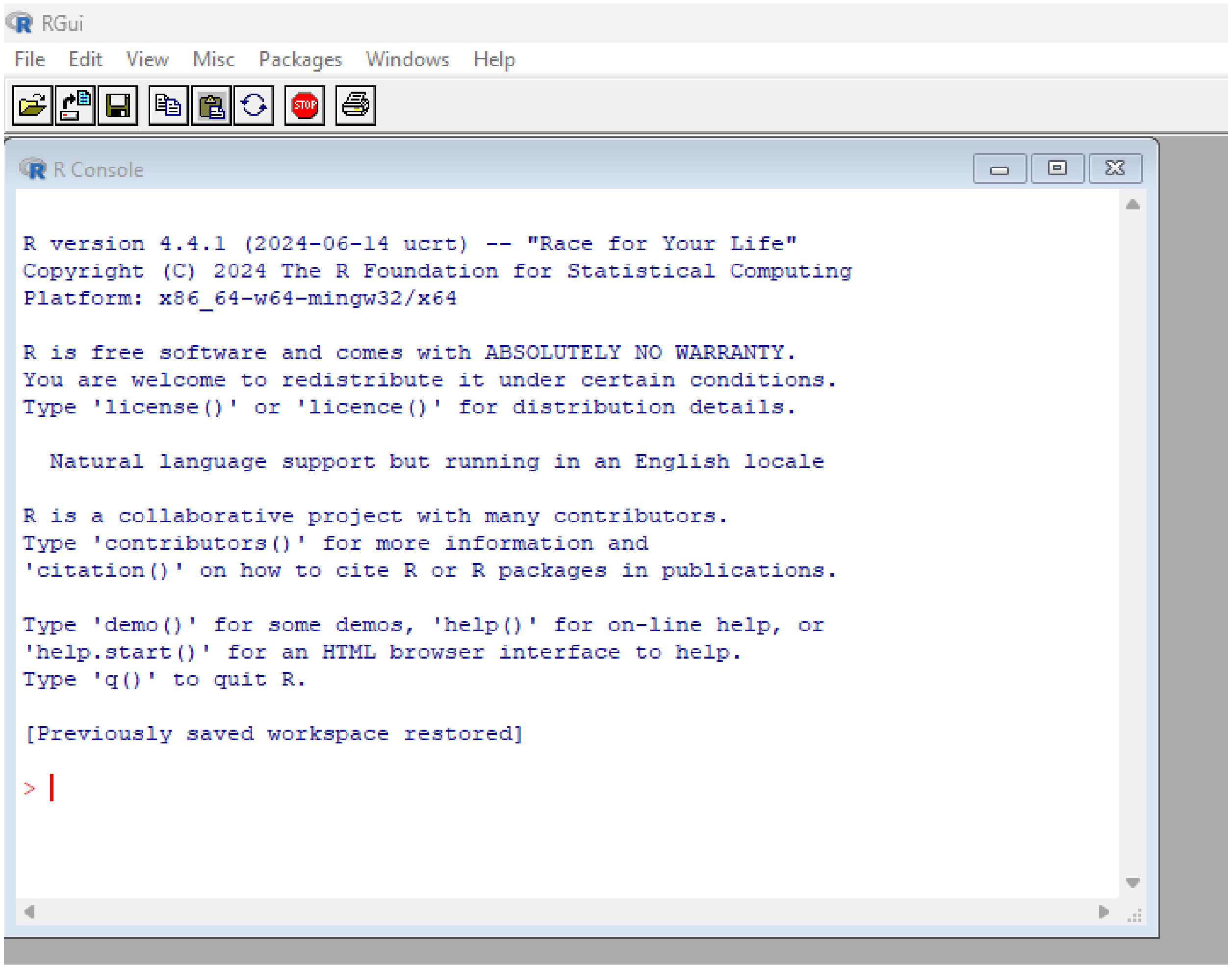
Installing R and RStudio

I assume that you have a PC or an Apple Mac, and that you want to install R on the hard disc. If you have access to the internet, then this could hardly be simpler. First go to the site called CRAN (this stands for Comprehensive R Archive Network). You can type its full address, http://cran.r-project.org/ or simply type

CRAN into Google and be transported effortlessly to the site. Once there, you need to 'Download and Install R' by running the appropriate precompiled binary distributions.

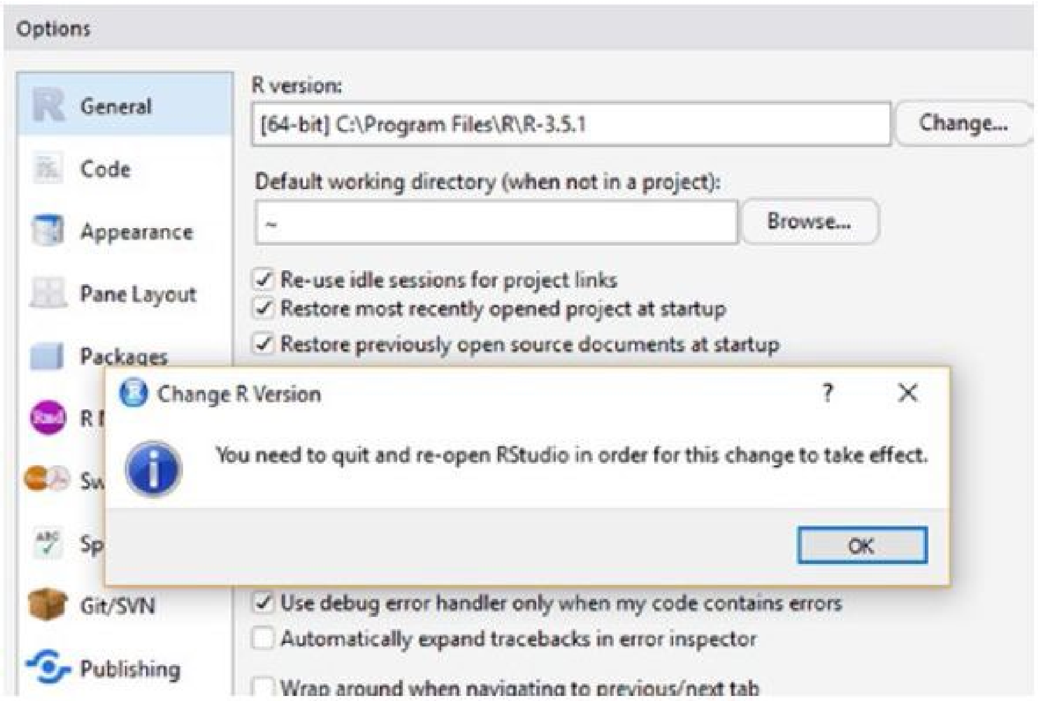
https://cran.r-project.org/bin/windows/base/

After the R programming command line application is installed, you can start it:



```
R version 4.4.1 (2024-06-14 ucrt) -- "Race for Your Life"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> 
```

To download and install RStudio, visit www.rstudio.com/ . After downloading the RStudio setup file, double-click the file to install RStudio. After installing RStudio, you can run the RStudio software. Before running the script, you need to select the R programming command line application version to use. Click Tools ➤ Global Option and Click the Change button to select the R version. After clicking OK and choosing the R version, you must restart the RStudio IDE. After restarting RStudio, the Console tab should show the selected R version.


R version: [64-bit] C:\Program Files\R\R-3.5.1

Default working directory (when not in a project): ~

- ✓ Re-use idle sessions for project links
- ✓ Restore most recently opened project at startup
- ✓ Restore previously open source documents at startup
- ✓ Use debug error handler only when my code contains errors
- Automatically expand tracebacks in error inspector
- Wrap around when navigating to previous/next tab

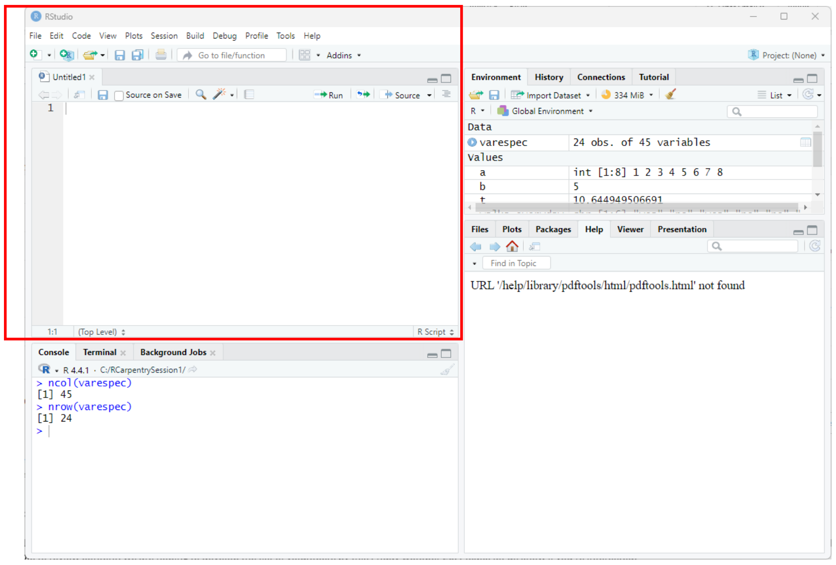
Change R Version

You need to quit and re-open RStudio in order for this change to take effect.

Creating R Script for your Work (very important!)

With RStudio, you can write code into the code editor and run the script. As you type the code, RStudio shows the intelligent code completion. You must select all the R code in the code editor and click Run or Ctrl + Enter to run the script. The RStudio IDE offers syntax highlighting features in the code editor. When you run the R script, you can view the results in the Console tab and see the scatterplot in the Plots tab.

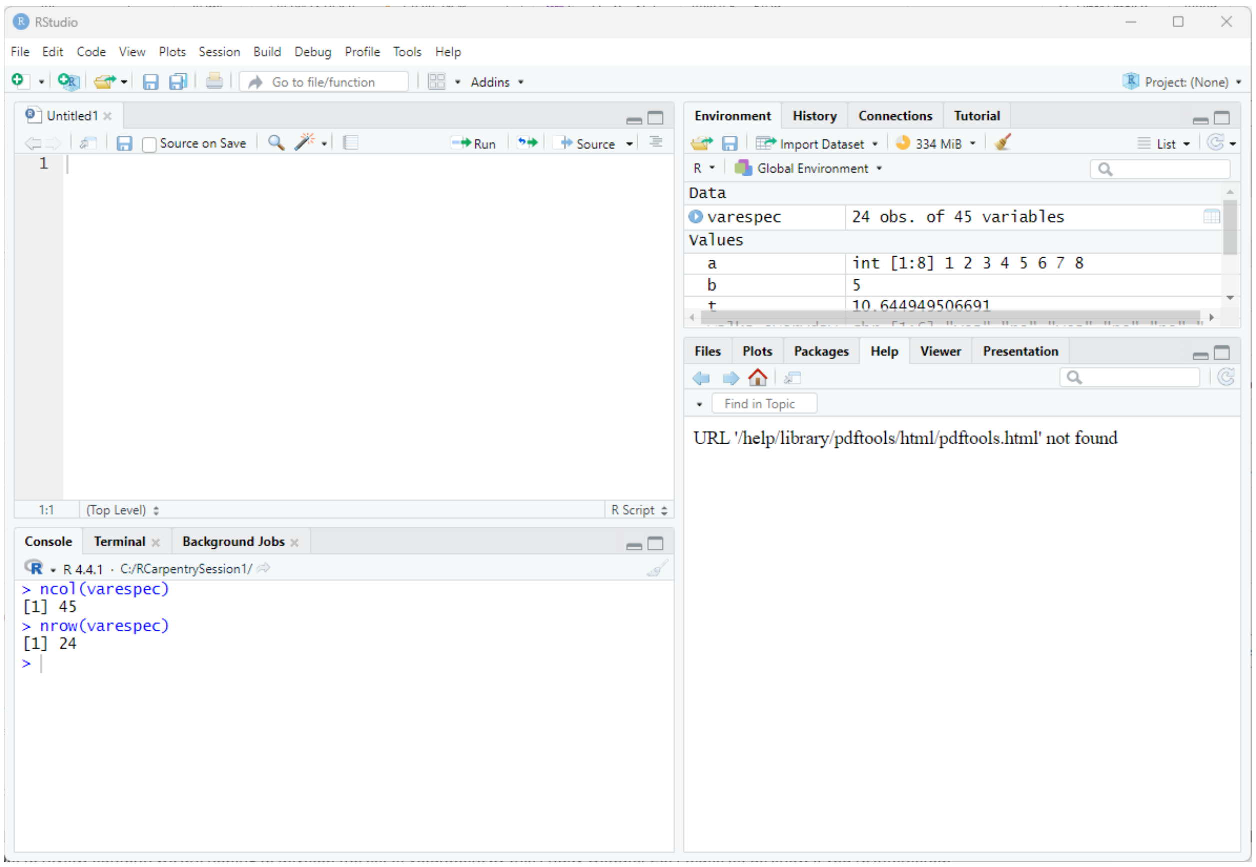
To create a new script in RStudio, click File ➤ New File ➤ R Script


```
> ncol(varespec)
[1] 45
> nrow(varespec)
[1] 24
>
```

Data	
varespec	24 obs. of 45 variables
Values	
a	int [1:8] 1 2 3 4 5 6 7 8
b	5
t	10.644949506691

URL '/help/library/pdftools/html/pdftools.html' not found

You can also see the option to create a new R script under Files -> Blank File -> R Script. When you go to save the file, it saves as an R file. This is a very important way to work so you can easily reproduce your code in future and document what you have run.



```
Data
varespec    24 obs. of 45 variables
Values
a           int [1:8] 1 2 3 4 5 6 7 8
b           5
t           10.644949506691
```

```
> ncol(varespec)
[1] 45
> nrow(varespec)
[1] 24
>
```

URL '/help/library/pdftools/html/pdftools.html' not found

Creating a Backup of your Dataset (very important!)

Create a back up of your work before making changes to the data structure.

# Create a backup before making changes

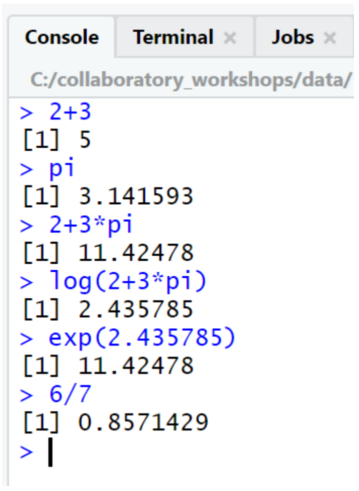
varespec_backup <- varespec

# To revert back to the original dataframe

varespec <- varespec_backup

Basics of R

R as a Calculator



```
> 2+3
[1] 5
> pi
[1] 3.141593
> 2+3*pi
[1] 11.42478
> log(2+3*pi)
[1] 2.435785
> exp(2.435785)
[1] 11.42478
> 6/7
[1] 0.8571429
> 
```

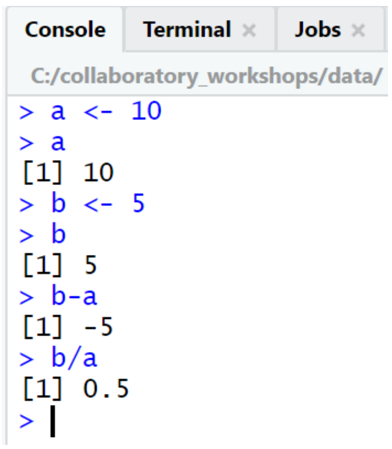
Entering and Manipulating Data in R

In R you have variables, functions and arguments

An **assignment operator** is used to assign the value on the right to the variable on the left

can use <- or = (also: ALT + for shortcut keys; Option + for Mac)

Scaler (simple single numeric value such as 1, 2/3, 3.14)



```
> a <- 10
> a
[1] 10
> b <- 5
> b
[1] 5
> b-a
[1] -5
> b/a
[1] 0.5
> 
```

Vector (is simply a series of variables)

A vector is used when you want to store and modify a set of values. The data types can be logical, integer, double, and character.

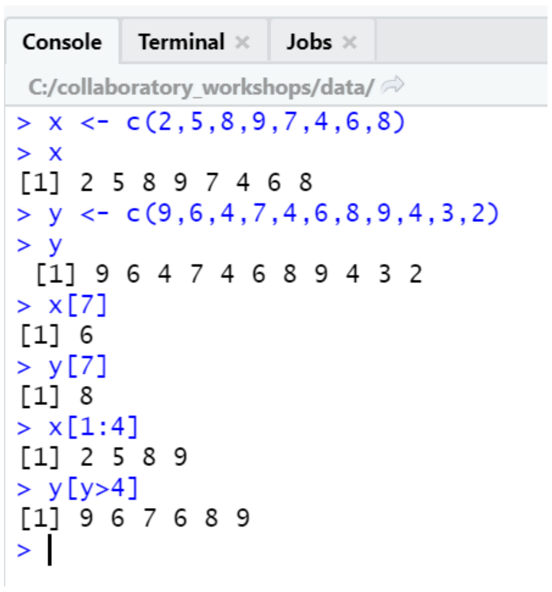
You can also use the operator ; to create a vector:

> a <- 1:8;

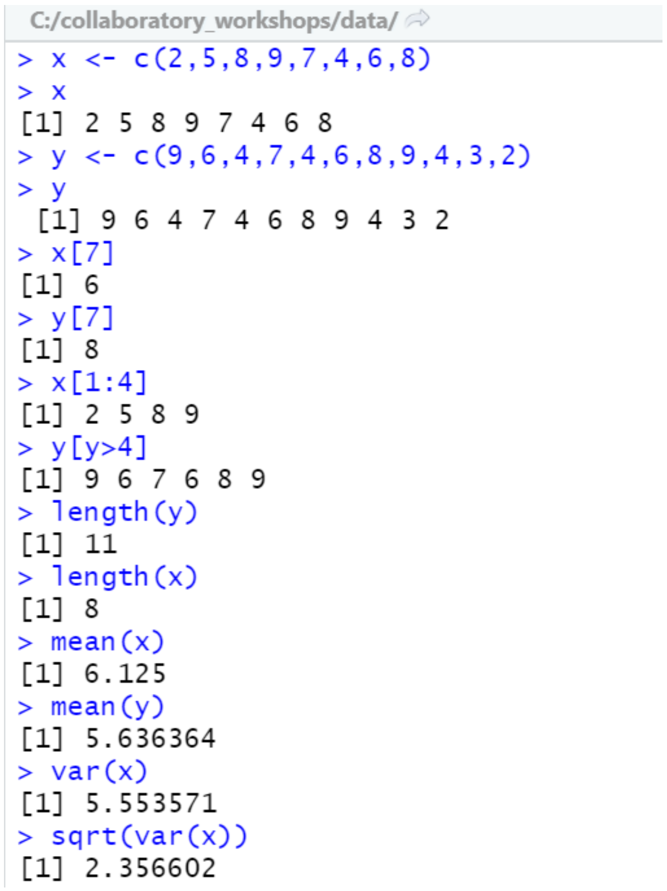
```
> a
[1] 1 2 3 4 5 6 7 8
```

To combine values into a vector use the “c ()” (concatenate function).

You can also apply functions to vectors.



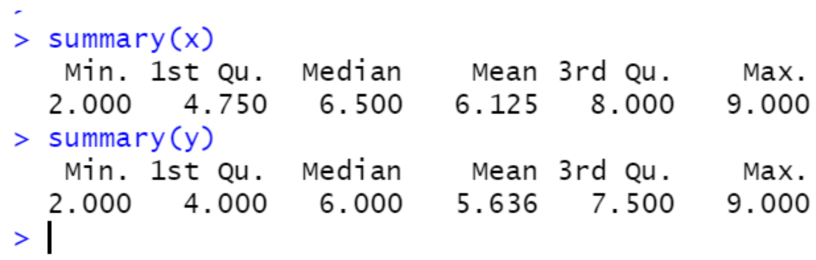
```
> x <- c(2,5,8,9,7,4,6,8)
> x
[1] 2 5 8 9 7 4 6 8
> y <- c(9,6,4,7,4,6,8,9,4,3,2)
> y
 [1] 9 6 4 7 4 6 8 9 4 3 2
> x[7]
[1] 6
> y[7]
[1] 8
> x[1:4]
[1] 2 5 8 9
> y[y>4]
[1] 9 6 7 6 8 9
> 
```



```
C:/collaboratory_workshops/data/
> x <- c(2,5,8,9,7,4,6,8)
> x
[1] 2 5 8 9 7 4 6 8
> y <- c(9,6,4,7,4,6,8,9,4,3,2)
> y
 [1] 9 6 4 7 4 6 8 9 4 3 2
> x[7]
[1] 6
> y[7]
[1] 8
> x[1:4]
[1] 2 5 8 9
> y[y>4]
[1] 9 6 7 6 8 9
> length(y)
[1] 11
> length(x)
[1] 8
> mean(x)
[1] 6.125
> mean(y)
[1] 5.636364
> var(x)
[1] 5.553571
> sqrt(var(x))
[1] 2.356602
```

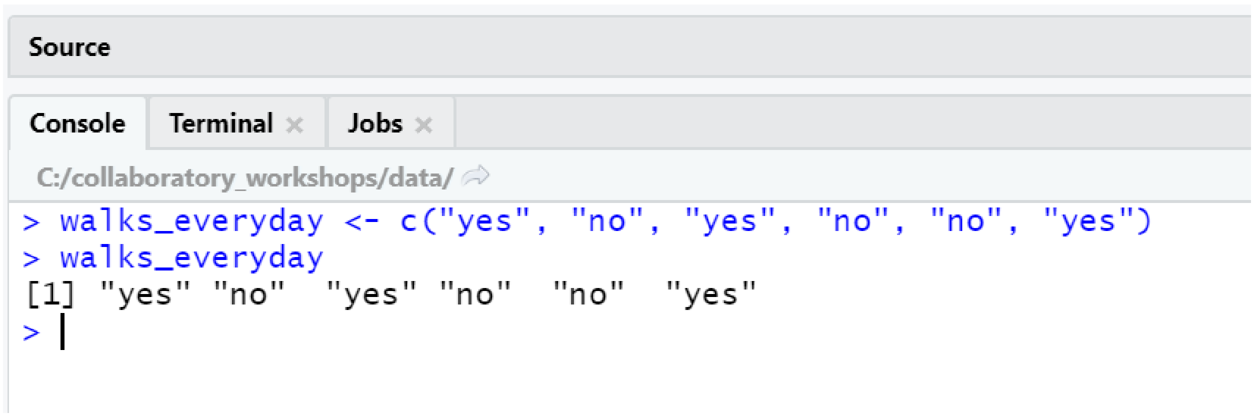
Summary Statistics

Or, you can also apply simple statistical calculations to vectors.
# summary () mean, median, 25th and 75th quartiles, min, max



```
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  2.000   4.750   6.500   6.125   8.000   9.000 
> summary(y)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  2.000   4.000   6.000   5.636   7.500   9.000 
> 
```

Character variables

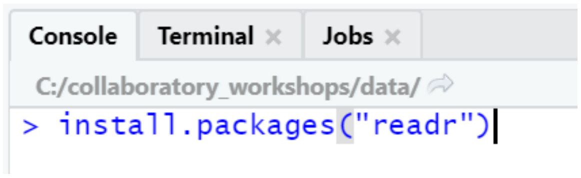


```
> walks_everyday <- c("yes", "no", "yes", "no", "no", "yes")
> walks_everyday
[1] "yes" "no"  "yes" "no"  "no"  "yes"
> 
```

Install R Packages

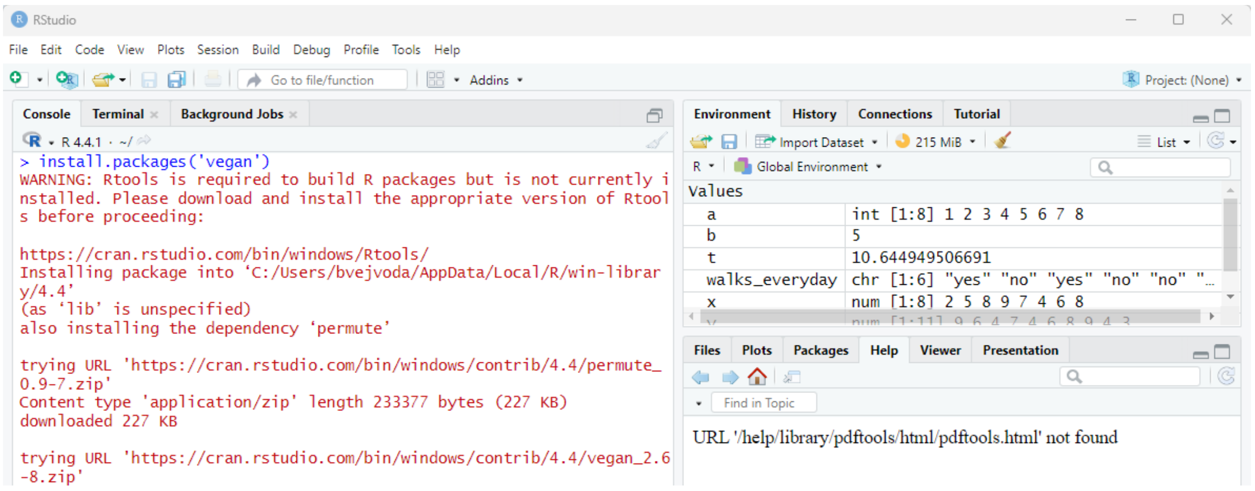
R packages are a collection of R functions, complied code and sample data

We need either the readr or rcpp package installed to load a csv (if done by command) so let's install "readr".



```
> install.packages("readr")
```

Now let's install 'vegan' an R package for community ecologists.



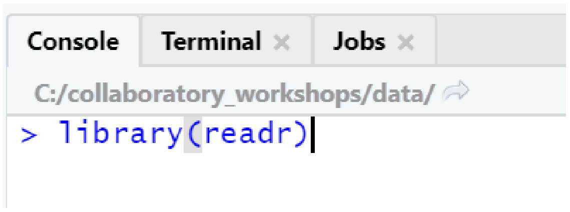
```
> install.packages('vegan')
WARNING: Rtools is required to build R packages but is not currently i
nstalled. Please download and install the appropriate version of Rtool
s before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/bvejvoda/AppData/Local/R/win-librar
y/4.4'
(as 'lib' is unspecified)
also installing the dependency 'permute'

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.4/permute_
0.9-7.zip'
Content type 'application/zip' length 233377 bytes (227 KB)
downloaded 227 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.4/vegan_2.6
-8.zip'
```

```
Values
a              int [1:8] 1 2 3 4 5 6 7 8
b              5
t              10.644949506691
walks_everyday chr [1:6] "yes" "no" "yes" "no" "no" "...
x              num [1:8] 2 5 8 9 7 4 6 8
```

URL '/help/library/pdftools/html/pdftools.html' not found

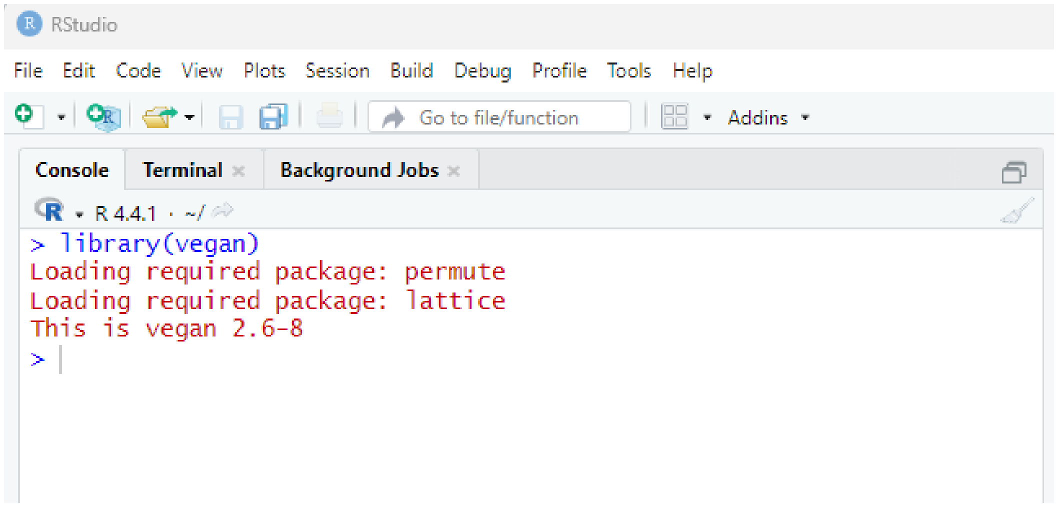
Load Installed R Packages

Now we need to load the installed package into R from the /library location.



```
C:/collaboratory_workshops/data/
> library(readr)
```

Now we can load ‘vegan’ too.

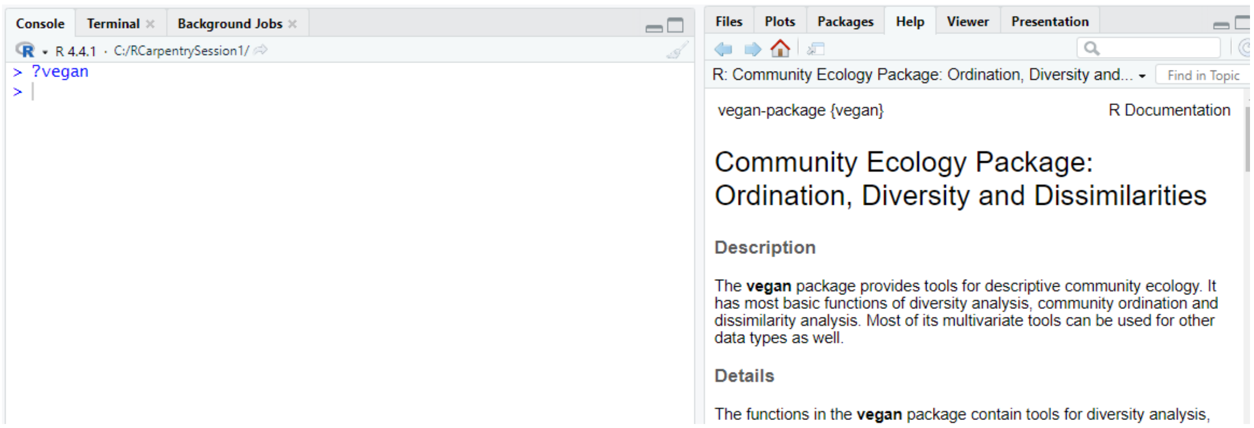


```
> library(vegan)
Loading required package: permute
Loading required package: lattice
This is vegan 2.6-8
>
```

Using the Help Function

You can use the ? operator or the help() function to access documentation for specific functions or datasets.

# Accessing documentation for the iris dataset ?vegan?


Console
Terminal
Background Jobs
R 4.4.1 · C:/RCarpentrySession1/
Files
Plots
Packages
Help
Viewer
Presentation
R: Community Ecology Package: Ordination, Diversity and...
Find in Topic

```
> ?vegan
>
```

vegan-package {vegan} R Documentation

Community Ecology Package: Ordination, Diversity and Dissimilarities

Description

The **vegan** package provides tools for descriptive community ecology. It has most basic functions of diversity analysis, community ordination and dissimilarity analysis. Most of its multivariate tools can be used for other data types as well.

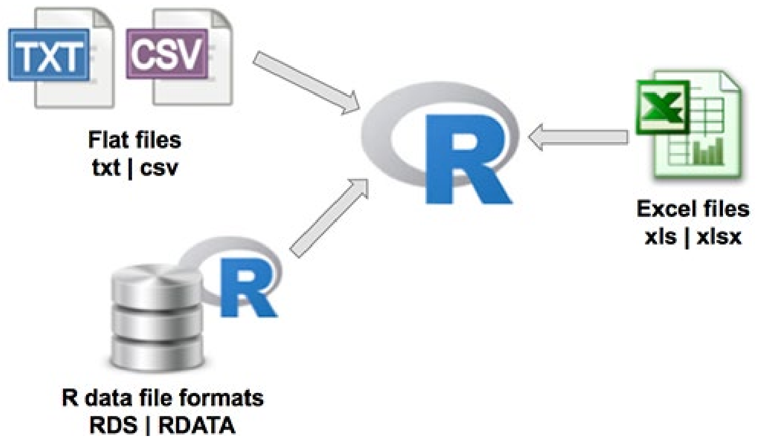
Details

The functions in the **vegan** package contain tools for diversity analysis,

Loading Data in R

R programming allow you to import a dataset, which can be a comma-separated values (CSV) file, Excel file, tab-separated file, JSON file, or others like SPSS, Stata or SAS.

Reading data into the R console or R Studio is important, since you must have some data before you can do statistical computing and understand the data.

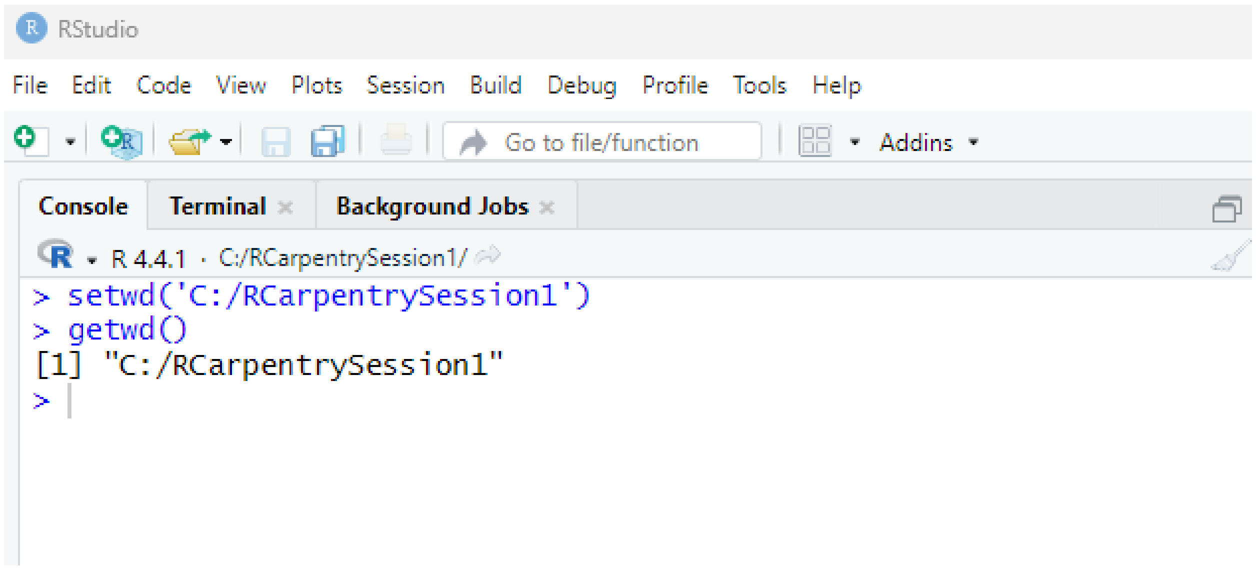


Set Working Directory

Before you import data into R Studio, you must determine your workplace or work directory first.

To print the current work directory, you use the getwd() function.

You can set the work directory using the setwd() function.


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Console Terminal Background Jobs
R 4.4.1 · C:/RCarpentrySession1/

```
> setwd('C:/RCarpentrySession1')
> getwd()
[1] "C:/RCarpentrySession1"
> 
```

Let’s Import a real dataset!

You can read a comma-separated values (CSV) file using the **read.csv()** function in the R programming language.

Download varespec to your laptop from here and store it somewhere you can see it.

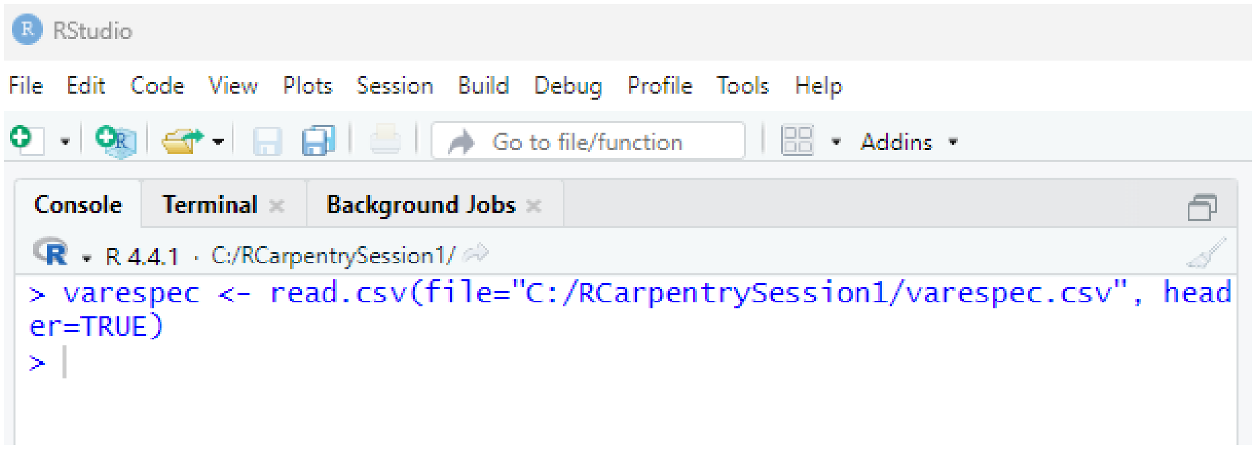
https://gist.github.com/essicolo/cd5c8b77c91e14b9fe648d63f9afaed9

Note that there are different functions for different data formats.

varespec <- read.csv(file="C:/RCarpentrySession1/varespec.csv", header=TRUE)

Note also the argument “header = TRUE” which holds if the file contains the names of the variables as its first line. Note also, “sep=”,” specifies the comma as a delimiter in the file.

Will use a .csv file found here:

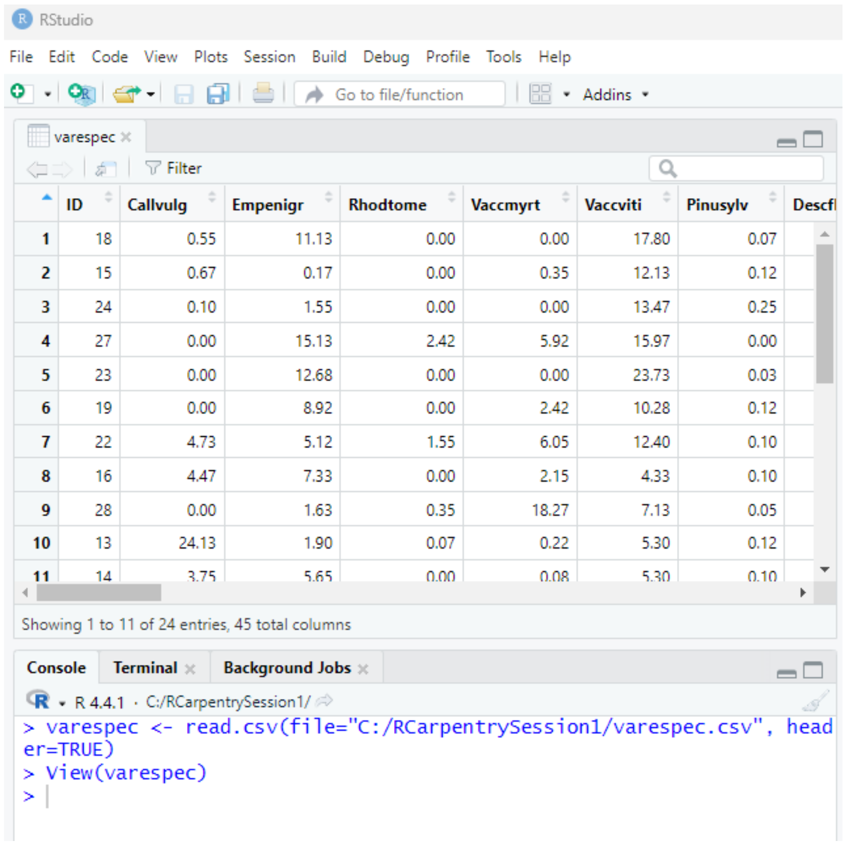

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Console Terminal Background Jobs
R 4.4.1 · C:/RCarpentrySession1/

```
> varespec <- read.csv(file="C:/RCarpentrySession1/varespec.csv", head
er=TRUE)
> 
```

Viewing the data

To display the data as a spreadsheet in the data viewer window:

View(varespec)

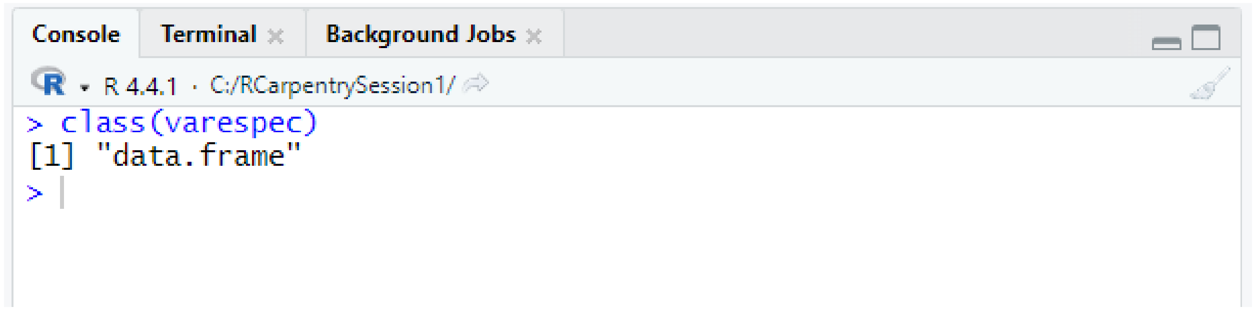

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
varespec
Filter
Console
Terminal
Background Jobs
R 4.4.1 · C:/RCarpentrySession1/

	ID	Callvulg	Empenigr	Rhodtome	Vaccmyrt	Vaccviti	Pinusylv	Descfl
1	18	0.55	11.13	0.00	0.00	17.80	0.07	
2	15	0.67	0.17	0.00	0.35	12.13	0.12	
3	24	0.10	1.55	0.00	0.00	13.47	0.25	
4	27	0.00	15.13	2.42	5.92	15.97	0.00	
5	23	0.00	12.68	0.00	0.00	23.73	0.03	
6	19	0.00	8.92	0.00	2.42	10.28	0.12	
7	22	4.73	5.12	1.55	6.05	12.40	0.10	
8	16	4.47	7.33	0.00	2.15	4.33	0.10	
9	28	0.00	1.63	0.35	18.27	7.13	0.05	
10	13	24.13	1.90	0.07	0.22	5.30	0.12	
11	14	3.75	5.65	0.00	0.08	5.30	0.10	

Showing 1 to 11 of 24 entries, 45 total columns

```
> varespec <- read.csv(file="C:/RCarpentrySession1/varespec.csv", head
er=TRUE)
> View(varespec)
> 
```

Display class of data (type): Most data are data frames



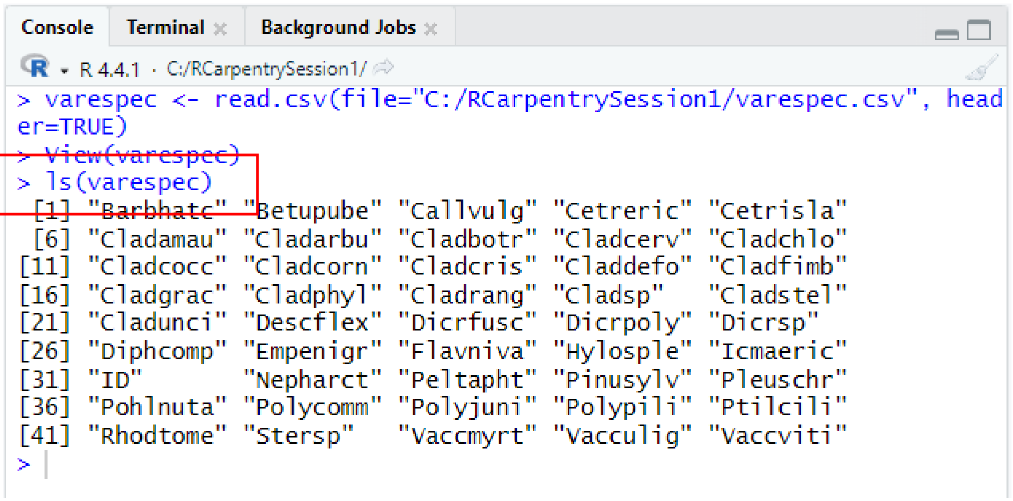
```
> class(varespec)
[1] "data.frame"
> 
```

List Variables

ls(varespec)

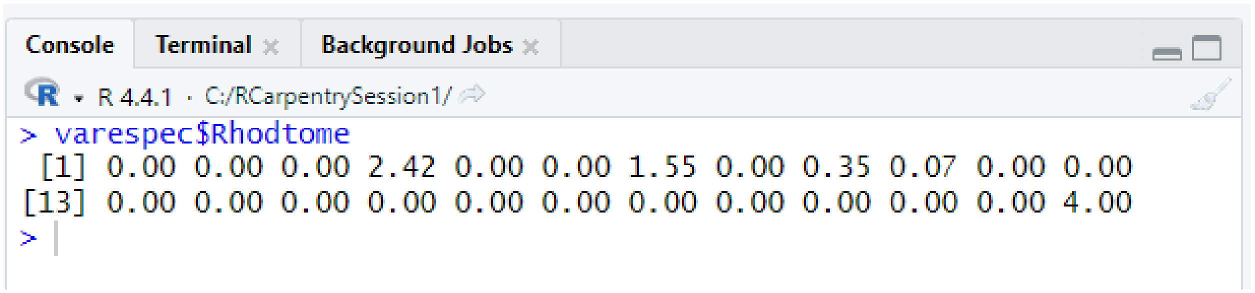
OR:

names(varespec)



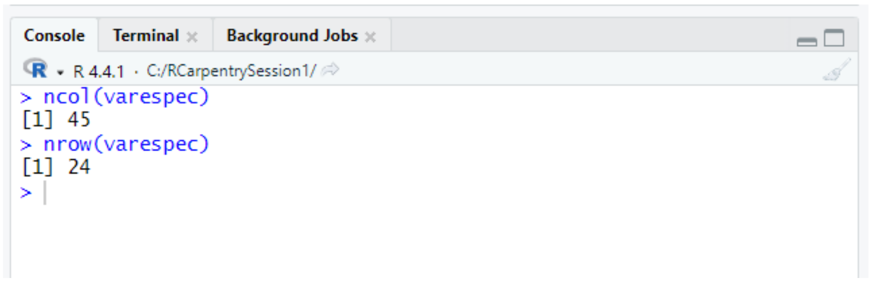
```
> varespec <- read.csv(file="C:/RCarpentrySession1/varespec.csv", head
er=TRUE)
> View(varespec)
> ls(varespec)
 [1] "Barbhatc" "Betupube" "Callvulg" "Cetreric" "Cetrisla"
 [6] "Cladamau" "Cladarbu" "Cladbotr" "Cladcerv" "Cladchlo"
[11] "Cladcocc" "Cladcorn" "Cladcris" "Claddefo" "Cladfimb"
[16] "Cladgrac" "Cladphyl" "Cladrang" "Cladsp"   "Cladstel"
[21] "Cladunci" "Descflex" "Dicrfusc" "Dicrpoly" "Dicrsp"
[26] "Diphcomp" "Empenigr" "Flavniva" "Hylosple" "Icmaeric"
[31] "ID"       "Nepharct" "Peltapht" "Pinusylv" "Pleuschr"
[36] "Pohlnuta" "Polycomm" "Polyjuni" "Polypili" "Ptilcili"
[41] "Rhodtome" "Stersp"   "Vaccmyrt" "Vacculig" "Vaccviti"
>
```

Display a variable and its response value



```
> varespec$Rhodtome
 [1] 0.00 0.00 0.00 2.42 0.00 0.00 1.55 0.00 0.35 0.07 0.00 0.00
[13] 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 4.00
>
```

Display # of columns and rows



Console Terminal Background Jobs

R 4.4.1 · C:/RCarpentrySession1/

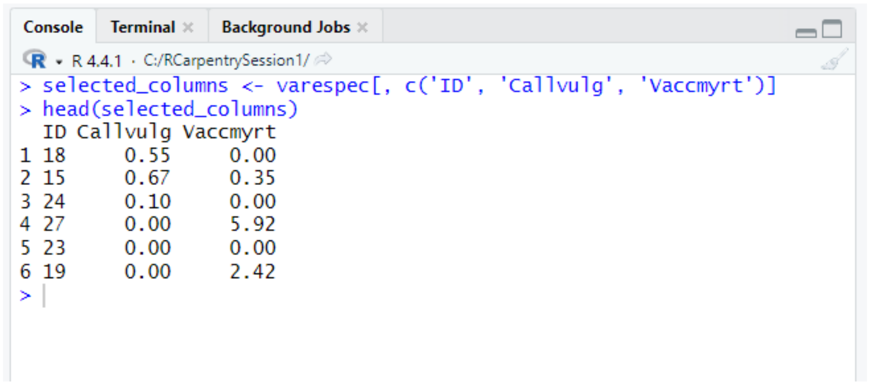
```
> ncol(varespec)
[1] 45
> nrow(varespec)
[1] 24
> 
```

Using Select function

selected_columns <- varespec[, c('ID', 'Callvulg', 'Vaccmyrt')]

Then to display the selected columns:

head(selected_columns)



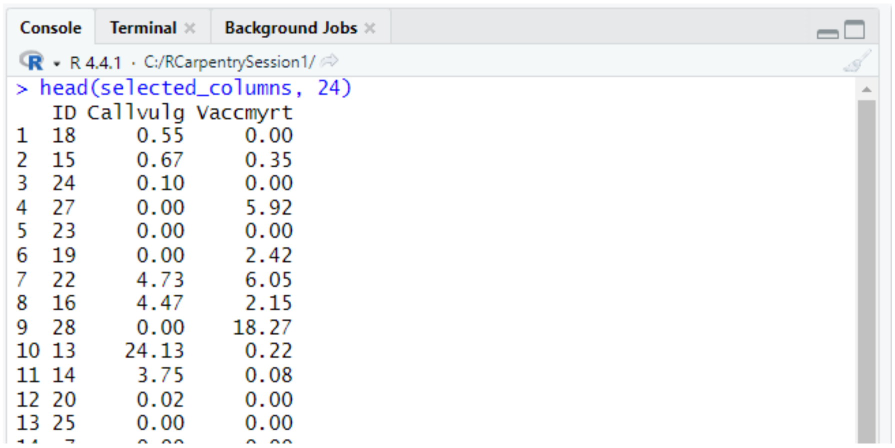
Console Terminal Background Jobs

R 4.4.1 · C:/RCarpentrySession1/

```
> selected_columns <- varespec[, c('ID', 'Callvulg', 'Vaccmyrt')]
> head(selected_columns)
  ID Callvulg Vaccmyrt
1 18     0.55     0.00
2 15     0.67     0.35
3 24     0.10     0.00
4 27     0.00     5.92
5 23     0.00     0.00
6 19     0.00     2.42
> 
```

By default, R displays the first 6 rows by default. To see more rows you can specify:

head(selected_columns, 24)



Console Terminal Background Jobs

R 4.4.1 · C:/RCarpentrySession1/

```
> head(selected_columns, 24)
   ID Callvulg Vaccmyrt
1  18     0.55     0.00
2  15     0.67     0.35
3  24     0.10     0.00
4  27     0.00     5.92
5  23     0.00     0.00
6  19     0.00     2.42
7  22     4.73     6.05
8  16     4.47     2.15
9  28     0.00    18.27
10 13    24.13     0.22
11 14     3.75     0.08
12 20     0.02     0.00
13 25     0.00     0.00
```

Renaming variables

# Rename the 'Callvulg' column to 'Callvulg2' using base R

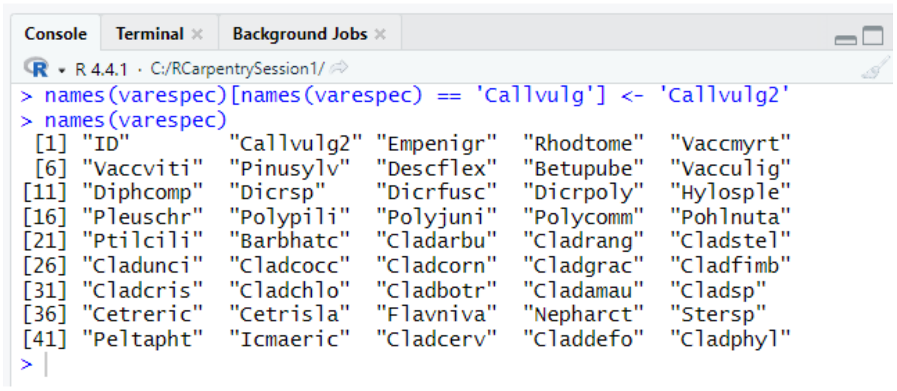
names(varespec)[names(varespec) == 'Callvulg'] <- 'Callvulg2'

# Verify the change by checking the column names

names(varespec)

1. names(varespec): This retrieves the column names of the varespec dataframe.
2. names(varespec) == 'Callvulg': This creates a logical vector that is TRUE where the column name matches 'Callvulg'.
3. names(varespec)[...] <- 'Callvulg2': This assigns the new column name 'Callvulg2' to the position where the logical vector is TRUE.

After running this code, the Callvulg column will be renamed to Callvulg2



```
> names(varespec)[names(varespec) == 'Callvulg'] <- 'Callvulg2'
> names(varespec)
 [1] "ID"        "Callvulg2" "Empenigr"  "Rhodtome"  "Vaccmyrt" 
 [6] "Vaccviti"  "Pinusylv"  "Descflex"  "Betupube"  "Vacculig" 
[11] "Diphcomp"  "Dicrsp"    "Dicrfusc"  "Dicrpoly"  "Hylosple" 
[16] "Pleuschr"  "Polypili"  "Polyjuni"  "Polycomm"  "Pohlnuta" 
[21] "Ptilcili"  "Barbhatc"  "Cladarbu"  "Cladrang"  "Cladstel" 
[26] "Cladunci"  "Cladcocc"  "Cladcorn"  "Cladgrac"  "Cladfimb" 
[31] "Cladcris"  "Cladchlo"  "Cladbotr"  "Cladamau"  "Cladsp"   
[36] "Cetreric"  "Cetrisla"  "Flavniva"  "Nepharct"  "Stersp"   
[41] "Peltapht"  "Icmaeric"  "Cladcerv"  "Claddefo"  "Cladphyl" 
> 
```

Missing values

Converting 0.00 to NA can be justified for several reasons:

1. **Data Quality**: 0.00 may represent a lack of data rather than a meaningful value, such as when a measurement was not taken or is unknown. Replacing it with NA helps to clarify that this entry should not be interpreted as valid data.
2. **Statistical Analysis**: Many statistical methods treat NA values differently from zeros. By converting 0.00 to NA, you can avoid misleading results in analyses that should exclude non-representative values, such as when calculating means, variances, or conducting regression analyses.
3. **Data Integrity**: It helps maintain the integrity of the dataset by distinguishing between actual zero values (which may be significant) and entries that are missing or undefined.

4. **Visualization Clarity**: When visualizing data, representing missing values as NA can lead to clearer and more informative plots, helping to better convey the data's story without obscuring it with misleading zeros.
5. **Consistency**: In datasets with multiple types of missing or incomplete information, using NA consistently can help streamline data handling and processing in subsequent analyses.

Now let's run code to convert the 0.00 values to NA.

```
# Convert 0.00 to NA in the entire dataframe

varespec[varespec == 0] <- NA

# Print the result

print(varespec)
```