1. Installing R and RStudio

This page is intended to provide a brief and high-level introduction to R. Additional educational material on using R/R Studio are available on many sites, including the RStudio education site or this R Spatial page.

Tips

If you have not installed R or RStudio, please see the 'getting started' section on the home page.

2. Orientation – how code works in RStudio

Comments and code:

1. Comments: any line in R code that starts with a '#' is considered a *comment*. These are not executed by RStudio as R code, but rather are used as notes to the user.
2. Code: a line or segment of code can be RUN by a) clicking on the line or b) highlighting the text, and press CMD + return (Mac) or CTRL + Enter (Windows). You can run a single line of code (clicking on or highlighting the line), or multiple lines of code (highlight multiple lines of code).

How it works in RStudio:

1. Create an R script using RStudio (instructions below)
2. Write code into the R file, or copy the code from our code blocks below. Make sure to SAVE.
3. Click on a single line and press CMD + return (on Mac) or CTRL + enter (on Windows) to run the command on that line. Alternatively, you can highlight any section of text over one or more lines and press CMD + return (on Mac) or CTRL + enter (on Windows) to run the contents. Our scripts are designed so 'modules' of code can be run at a time.
4. As above, nothing is returned if they are loaded successfully, or an error message is returned if they are not.

Example:

This is a comment:

Input

```
## Run the following line to set your working directory
```

And this is the code:

Input

```
setwd("/Users/Tom/Desktop")
getwd()
```

If you highlight both lines of code and press CMD + return (Mac) or CTRL + Enter (Windows) both lines of code will be run, which will 1) set the working directory to my desktop, and 2) check whether setting the working directory has worked. The result should look like this:

Output

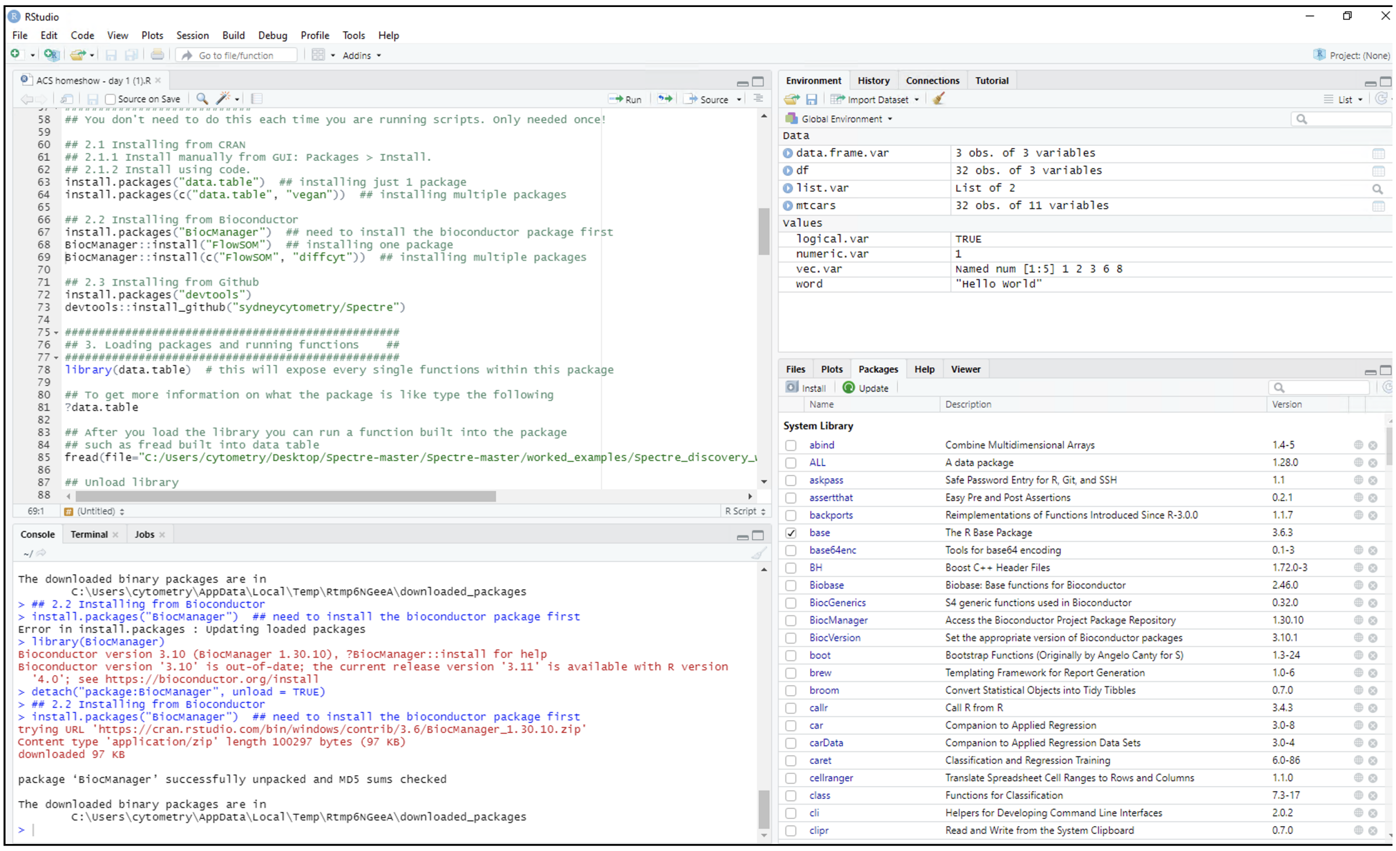
```
> setwd("/Users/Tom/Desktop/")
> getwd()
[1] "/Users/Tom/Desktop"
```

3. Create a new R script using RStudio

Open R Studio

To interact with the Spectre package in R, we will use **RStudio**. Open RStudio, and you should see something similar to the following:

- *Top left = R script. This is a text editor where lines or segments of code can be 'run', which will send commands to R.*
- *Bottom left = console. When commands are sent to R, the console will show the progress/output/result.*
- *Top right = workspace. Whenever you create an object in R (such as saving a set of data) it will show up here.*
- *Bottom right = various. This is mainly used for displaying plots (under 'Plots'), investigating the packages ('Packages'), or using the help section ('Help').*



```
## You don't need to do this each time you are running scripts. Only needed once!

## 2.1 Installing from CRAN
## 2.1.1 Install manually from GUI: Packages > Install.
## 2.1.2 Install using code.
install.packages("data.table")  ## installing just 1 package
install.packages(c("data.table", "vegan"))  ## installing multiple packages

## 2.2 Installing from Bioconductor
install.packages("BiocManager")  ## need to install the bioconductor package first
BiocManager::install("FlowSOM")  ## installing one package
BiocManager::install(c("FlowSOM", "diffcyt"))  ## installing multiple packages

## 2.3 Installing from Github
install.packages("devtools")
devtools::install_github("sydneycytometry/Spectre")

##############################################
## 3. Loading packages and running functions    ##
##############################################
library(data.table)  # this will expose every single functions within this package

## To get more information on what the package is like type the following
?data.table

## After you load the library you can run a function built into the package
## such as fread built into data table
fread(file="C:/Users/cytometry/Desktop/Spectre-master/Spectre-master/worked_examples/Spectre_discovery_

## Unload library
```

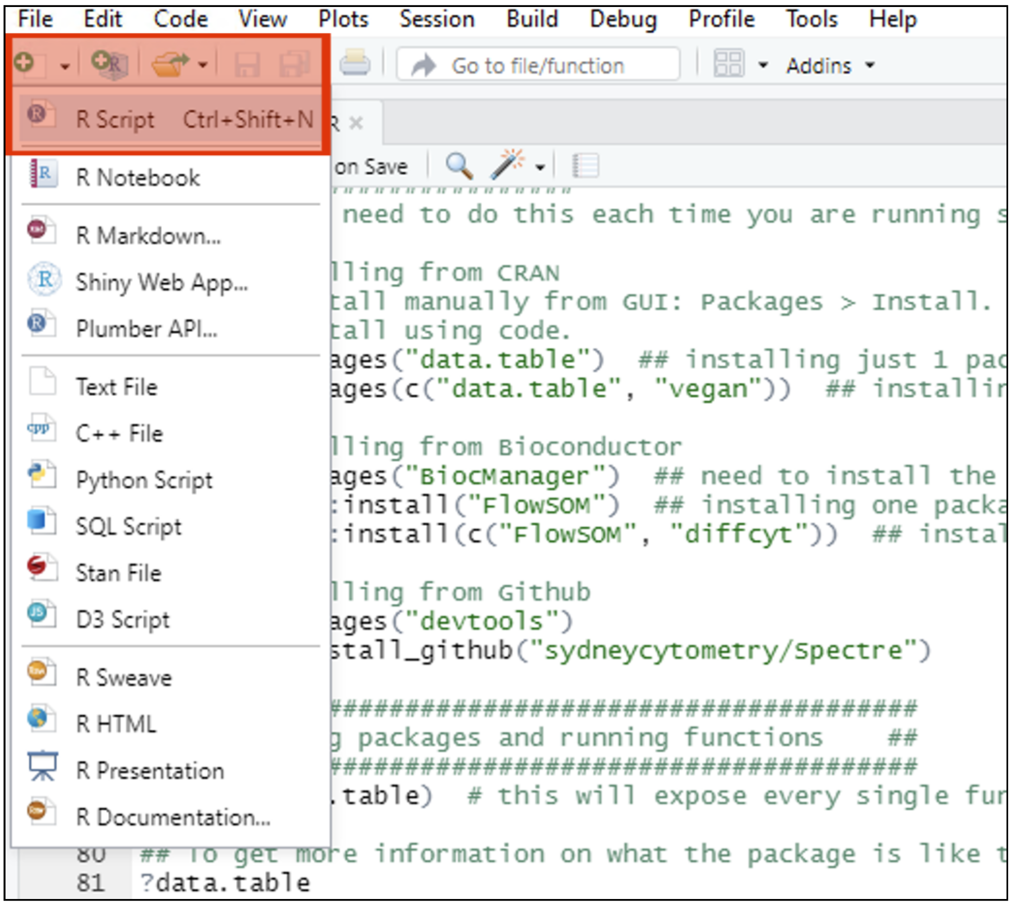
```
The downloaded binary packages are in
        C:\Users\cytometry\AppData\Local\Temp\Rtmp6NGeeA\downloaded_packages
> ## 2.2 Installing from Bioconductor
> install.packages("BiocManager")  ## need to install the bioconductor package first
Error in install.packages : Updating loaded packages
> library(BiocManager)
Bioconductor version 3.10 (BiocManager 1.30.10), ?BiocManager::install for help
Bioconductor version '3.10' is out-of-date; the current release version '3.11' is available with R version
  '4.0'; see https://bioconductor.org/install
> detach("package:BiocManager", unload = TRUE)
> ## 2.2 Installing from Bioconductor
> install.packages("BiocManager")  ## need to install the bioconductor package first
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.6/BiocManager_1.30.10.zip'
Content type 'application/zip' length 100297 bytes (97 KB)
downloaded 97 KB

package 'BiocManager' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\cytometry\AppData\Local\Temp\Rtmp6NGeeA\downloaded_packages
> |
```

Data	
data.frame.var	3 obs. of 3 variables
df	32 obs. of 3 variables
list.var	List of 2
mtcars	32 obs. of 11 variables
Values	
logical.var	TRUE
numeric.var	1
vec.var	Named num [1:5] 1 2 3 6 8
word	"Hello World"

	Name	Description	Version
	System Library		
☐	abind	Combine Multidimensional Arrays	1.4-5
☐	ALL	A data package	1.28.0
☐	askpass	Safe Password Entry for R, Git, and SSH	1.1
☐	assertthat	Easy Pre and Post Assertions	0.2.1
☐	backports	Reimplementations of Functions Introduced Since R-3.0.0	1.1.7
✓	base	The R Base Package	3.6.3
☐	base64enc	Tools for base64 encoding	0.1-3
☐	BH	Boost C++ Header Files	1.72.0-3
☐	Biobase	Biobase: Base functions for Bioconductor	2.46.0
☐	BiocGenerics	S4 generic functions used in Bioconductor	0.32.0
☐	BiocManager	Access the Bioconductor Project Package Repository	1.30.10
☐	BiocVersion	Set the appropriate version of Bioconductor packages	3.10.1
☐	boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-24
☐	brew	Templating Framework for Report Generation	1.0-6
☐	broom	Convert Statistical Objects into Tidy Tibbles	0.7.0
☐	callr	Call R from R	3.4.3
☐	car	Companion to Applied Regression	3.0-8
☐	carData	Companion to Applied Regression Data Sets	3.0-4
☐	caret	Classification and Regression Training	6.0-86
☐	cellranger	Translate Spreadsheet Cell Ranges to Rows and Columns	1.1.0
☐	class	Functions for Classification	7.3-17
☐	cli	Helpers for Developing Command Line Interfaces	2.0.2
☐	clipr	Read and Write from the System Clipboard	0.7.0

Create a new .R file and save it

- Make a folder on your desktop called "**Spectre demo**"
- In RStudio, create a new R Script file (.R) called "**MyScript**" and save it in the folder you just created (Spectre demo).



```
 need to do this each time you are running s

lling from CRAN
tall manually from GUI: Packages > Install.
tall using code.
ages("data.table")  ## installing just 1 pac
ages(c("data.table", "vegan"))  ## installin

lling from Bioconductor
ages("BiocManager")  ## need to install the
:install("FlowSOM")  ## installing one packa
:install(c("FlowSOM", "diffcyt"))  ## instal

lling from Github
ages("devtools")
stall_github("sydneycytometry/Spectre")

#####################################
g packages and running functions    ##
#####################################
.table)  # this will expose every single fur

## To get more information on what the package is like t
?data.table
```

4. Exploring the 'iris' dataset using RStudio

For each of the code-blocks below, copy the code into your new script, press save, and then highlight and press CMD/CTRL return to execute the code

For this demo we will use the 'iris' dataset, which consists of measurements of 150 flowers. Each row represents one flower, and each column represents a different measurement of that flower.

PART 1. READ THE DATASET

Copy the following into your script, save, then highlight the code and press CMD/CTRL return.

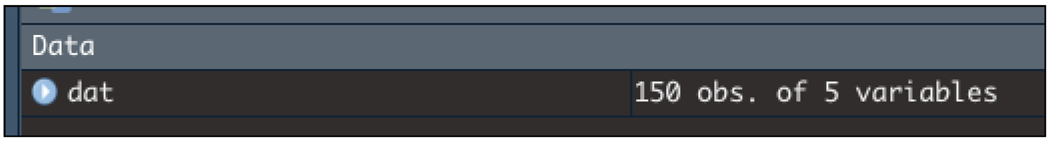
The first command we will run is to load the 'iris' dataset and save it as the object 'dat'. The lines starting with '#' are only comments, and will not excute as commands (even if you select them and press CMD + return).

Input

```
## Part 1: read the dataset

    # Use the 'iris' dataset (150 flowers one per row) with various measurement (each column is a different measurement)
    dat <- iris
```

Result:

After executing, you should should see a new object in the workspace (top right). This will be called 'dat', containing 150 observations, and 5 variables.



```
Data
dat                    150 obs. of 5 variables
```

Next we will review the dimensions of 'dat' (how many rows and columns) and preview data from the first 6 rows of dat.

Copy the following into your script, save, then highlight the code and press CMD/CTRL return.

Input

```
# Determine the number of rows and columns in the dataset
dim(dat)
```

Result:

You should now see the following in the console. Lines starting with '>' denote the commands that were executed. Lines without '>' are the output. As you can see below the request to show the dimensions of our dataset using dim(dat) has given us 150 rows and 5 columns.

Output

```
[1] 150   5
```

Copy the following into your script, save, then highlight the code and press CMD/CTRL return.

Input

```
# Examine the first few lines of dataset
head(dat)
```

Result:

You should now see the following in the console. Lines starting with '>' denote the commands that were executed. Lines without '>' are the output. The request to preview the first 6 rows of our data using head(dat) has shown us the contents of the first 6 rows.

Output

```
 Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

PART 2. PLOT THE DATASET

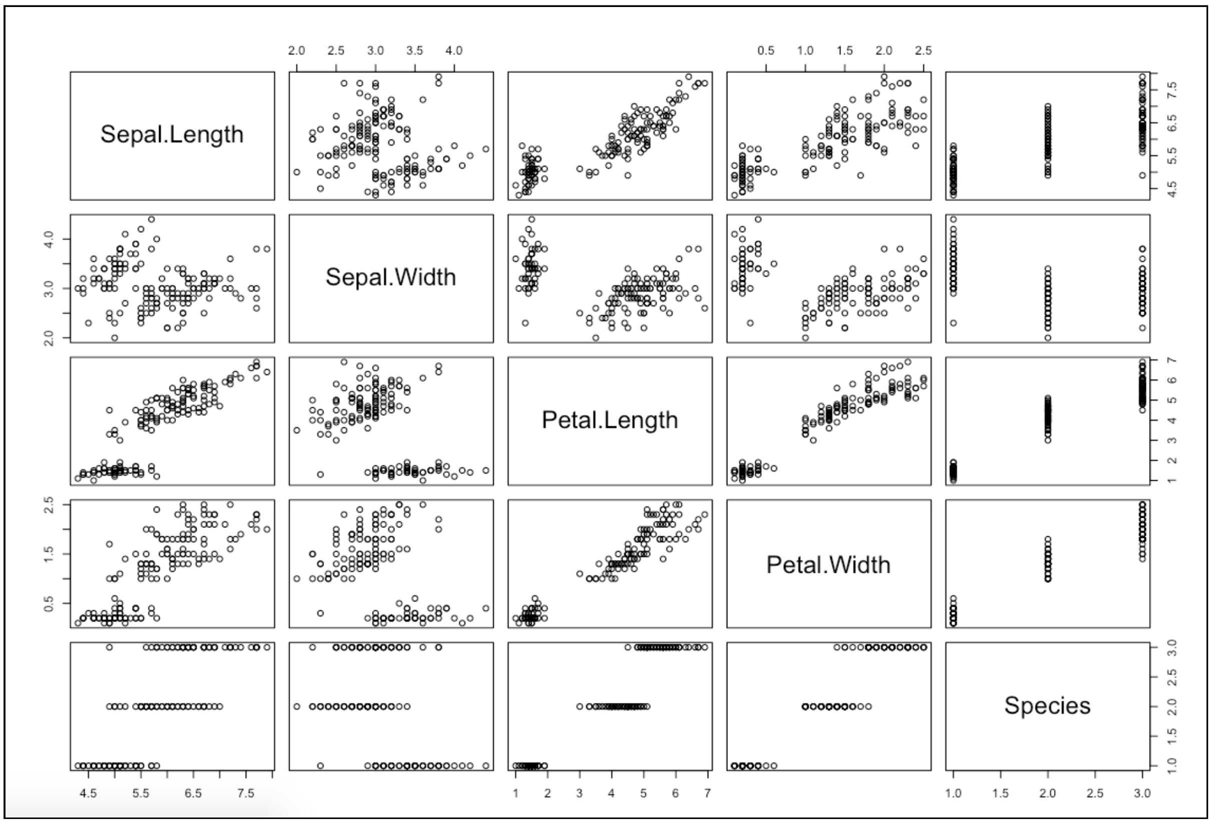
Next, we will plot some of the dataset. Select the following (in RStudio) and press return.

Copy the following into your script, save, then highlight the code and press CMD/CTRL return.

Input

```
## Part 2: plot the dataset

    # Plot iris dataset (all plots)
    plot(dat)
```

After executing, your should see the following under 'Plots'.



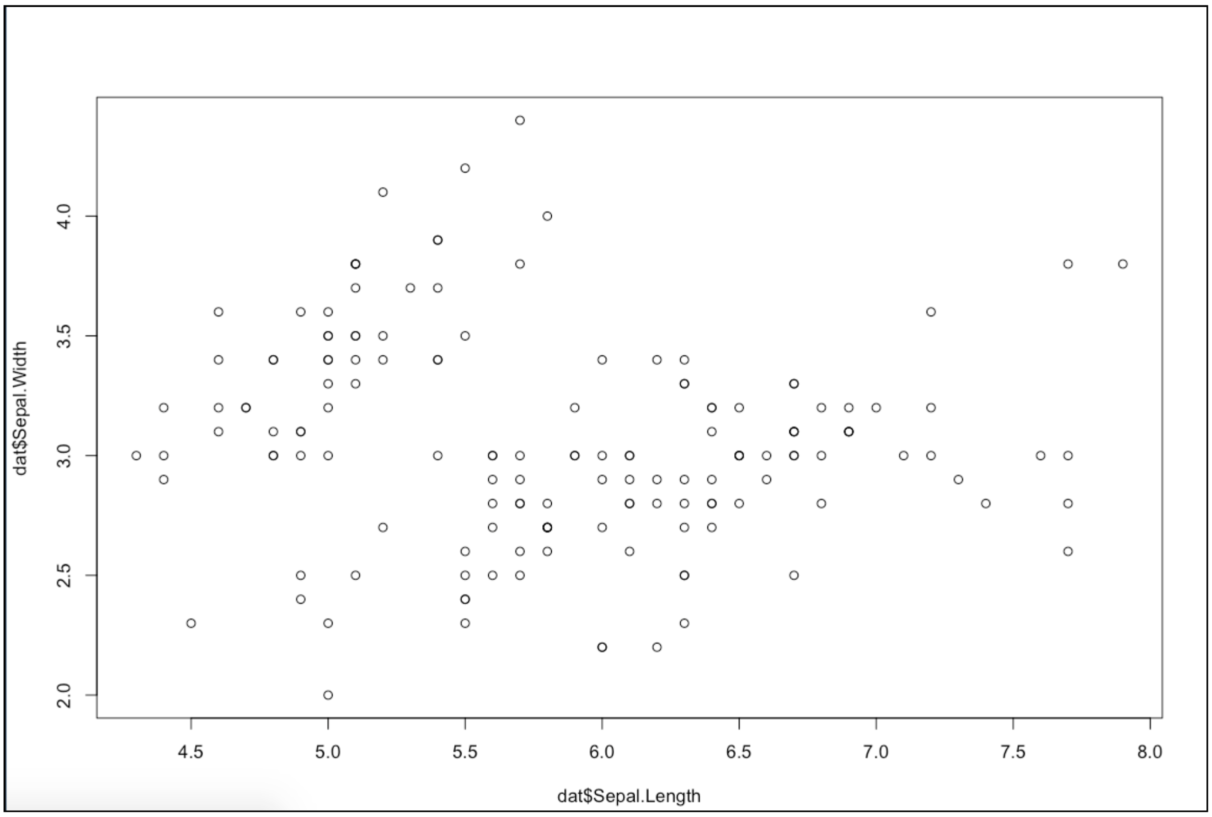
Copy the following into your script, save, then highlight the code and press CMD/CTRL return.

To be a little more specific, let's try plotting one column of the dataset against another.

Input

```
    # Plot iris dataset (chosen X and Y parameters)
    plot(x = dat$Sepal.Length, y = dat$Sepal.Width)
```

Now we should see a plot of the sepal width vs length.



PART 3. SAVE THE DATASET

Now, let's save the dataset as a .csv file. A .csv file is kind of like an .xlsx file, without the bells and whistles. Data in a table format is saved, using commas to indicate the separation of new columns. When this is read by excel or RStudio, it displays a table.

Copy the following into your script, save, then highlight the code and press CMD/CTRL return.

Run the following lines to determine the current working directory (where you will read files from and write files to):

Input

```
## Part 3: save the dataset

    # Determine the current working directory
    getwd()
```

This will return the location of your current working directory. In my case:

Output

```
[1] "/Users/thomasa"
```

Let's aim to save the CSV to our desktop. To do this we would have to a) change the 'working directory' to the desktop (on a mac, it would look something like "/Users/Tom/Desktop"). When we set a working directory, we are telling R a) where to look for files when we ask it to, and b) where to create files when we ask it to.

To set the working directory, follow these instructions:

Use setwd()

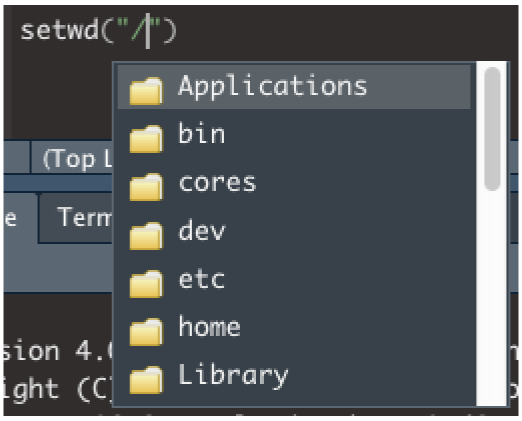
Type 'setwd()' into R

Input

```
setwd()
```

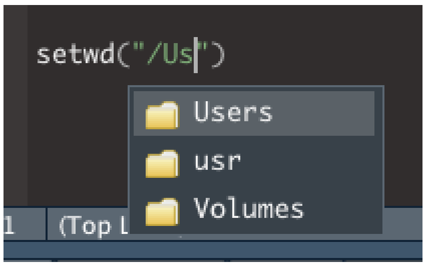
Finding a specific directory (absolute path)

- *On a mac, start by entering **"/"** between the (): **setwd("/")***
- *Click after the **/** and press TAB*
- *You should see a list of options from your root directory. You can press the UP or DOWN arrows, or use the mouse cursor, to select on of the options, and press ENTER to select in.*



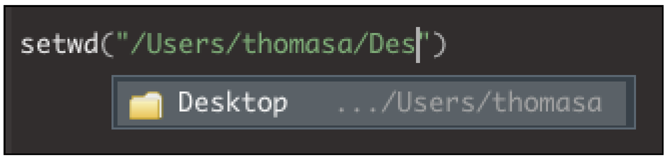
```
setwd("/")
```

- *If you start typing the name of a directory, the list of options will be filtered to options that match what you are typing.*



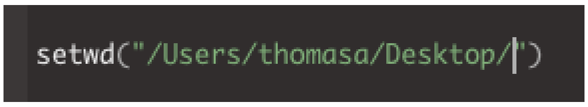
```
setwd("/Us")
```

- *In this case, I'll select 'Users' and press ENTER. I can then repeat the process to navigate down my folders*



```
setwd("/Users/thomasa/Des")
```

- *You can repeat this process to find your working directory until you reach your desired location.*



```
setwd("/Users/thomasa/Desktop/")
```

If I now select this line, or highlight the code and press ENTER, R will set the working directory:

Input

```
setwd("/Users/thomasa/Desktop/")
```

The following should be returned

Output

```
> setwd("/Users/thomasa/Desktop/")
```

Now I can check the working directory has been set correctly by running the following:

Input

```
getwd()
```

If everything has gone correctly, the following (or equivalent) should be returned:

Output

```
[1] "/Users/thomasa/Desktop/"
```

Now we will write the dataset to a .csv file (which will be saved in the working directory). We will use the function 'write.csv'. The input variables here are what dataset we want to write (x = dat) and what we want to call the file (file = "iris_dataset.csv").

Execute the following, and check the folder (set as your working directory) to see that the new file has been created.

Output

```
# Write a .csv file of the dataset
write.csv(x = dat, file = "iris_dataset.csv")
```