

Packages & reading/writing data

Justin Baumann

Table of contents

1	Learning Objectives	1
2	R Basics	1
2.1	Installing and Loading libraries	1
2.2	Looking at data!	2
2.3	Write data to file (saving data)	7
2.4	Read a file in (import data into R)	8
2.5	Read a file from an online source	8

1 Learning Objectives

- 1.) How to install and load libraries
- 2.) How to view and inspect data
- 3.) Read in data files & output data (write to file)

2 R Basics

2.1 Installing and Loading libraries

Libraries are packages of functions (and sometimes data) that we use to execute tasks in R. Packages are what make R so versatile! We can do almost anything with R if we learn how to utilize the right packages.

If we do not have a package already installed (for example, if you have only just downloaded R/ RStudio), we will need to use `install.packages('packagename')` to install each package that we need.

```
install.packages(tidyverse)
```

OR - We can use the ‘Packages’ tab in the bottom right quadrant to install packages. Simply navigate to ‘Packages’, select ‘install packages’ and enter the package names you need (separate each package by commas). **NOTE** for users for rstudio.mtholyoke.edu – You cannot install packages to the Mt Holyoke cloud instance of R. If we need something that isn’t installed we will need to contact IT!

In order for a *package to work*, we must first load it! We do this as with the code `library(packagename)`

```
library(tidyverse) #for data manipulation
library(palmerpenguins) #for some fun data!
```

It is best practice to load all of the packages you will need at the top of your script

In this course we will be following a best practices guide that utilizes a library called ‘Tidyverse’ for data manipulation and analysis. Tidyverse contains many packages all in one, including the very functional ‘dplyr’ and ‘ggplot2’ packages. You will almost always use Tidyverse, so make sure to load it in :)

Note the ‘#’ with notes after them in the code chunk above. These are called comments. You can comment out any line of code in R by using a ‘#’. This is strongly recommended when you are programming. We will discuss more later!

2.2 Looking at data!

R has integrated data sets that we can use to play around with code and learn.

examples: `mtcars` (a dataframe all about cars, this is available in R without loading a package), and `iris` (in the ‘vegan’ package, great for testing out ecology related functions and code)

Load a dataset R has some test datasets built into it. Let’s load one and look at it!

```
mtcars
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

Using head() and tail() Now let's look at the data frame (df) using head() and tail() These tell us the column names, and let us see the top or bottom 6 rows of data.

```
head(mtcars)
```

```

      mpg  cyl  disp  hp  drat    wt  qsec vs  am  gear  carb
Mazda RX4           21.0   6  160  110  3.90  2.620 16.46  0   1    4    4
Mazda RX4 Wag       21.0   6  160  110  3.90  2.875 17.02  0   1    4    4
Datsun 710           22.8   4  108   93  3.85  2.320 18.61  1   1    4    1
Hornet 4 Drive       21.4   6  258  110  3.08  3.215 19.44  1   0    3    1
Hornet Sportabout   18.7   8  360  175  3.15  3.440 17.02  0   0    3    2
Valiant              18.1   6  225  105  2.76  3.460 20.22  1   0    3    1

```

```
tail(mtcars) #tail shows the header and the last 6 rows
```

```
      mpg cyl  disp  hp drat   wt  qsec vs am gear carb
Porsche 914-2 26.0  4 120.3  91 4.43 2.140 16.7 0  1   5   2
Lotus Europa 30.4  4  95.1 113 3.77 1.513 16.9 1  1   5   2
Ford Pantera L 15.8  8 351.0 264 4.22 3.170 14.5 0  1   5   4
Ferrari Dino  19.7  6 145.0 175 3.62 2.770 15.5 0  1   5   6
Maserati Bora 15.0  8 301.0 335 3.54 3.570 14.6 0  1   5   8
Volvo 142E    21.4  4 121.0 109 4.11 2.780 18.6 1  1   4   2
```

column attributes If we want to see the attributes of each column we can use the `str()` function

```
str(mtcars) #str shows attributes of each column
```

```
'data.frame':  32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

`str()` is very important because it allows you to see the type of data in each column. Types include: integer, numeric, factor, date, and more. If the data in a column are factors instead of numbers you may have an issue in your data (your spreadsheet)

Changing column attributes Importantly, you can change the type of the column. Here is an example

```
mtcars$mpg=as.factor(mtcars$mpg) # Makes mpg a factor instead of a number
str(mtcars)
```

```
'data.frame':  32 obs. of  11 variables:
 $ mpg : Factor w/ 25 levels "10.4","13.3",...: 16 16 19 17 13 12 3 20 19 14 ...
```

```

$ cyl : num 6 6 4 6 8 6 8 4 4 6 ...
$ disp: num 160 160 108 258 360 ...
$ hp : num 110 110 93 110 175 105 245 62 95 123 ...
$ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
$ wt : num 2.62 2.88 2.32 3.21 3.44 ...
$ qsec: num 16.5 17 18.6 19.4 17 ...
$ vs : num 0 0 1 1 0 1 0 1 1 1 ...
$ am : num 1 1 1 0 0 0 0 0 0 0 ...
$ gear: num 4 4 4 3 3 3 3 4 4 4 ...
$ carb: num 4 4 1 1 2 1 4 2 2 4 ...

```

```

mtcars$mpg=as.numeric(mtcars$mpg) #Changes mpg back to a number
str(mtcars)

```

```

'data.frame': 32 obs. of 11 variables:
 $ mpg : num 16 16 19 17 13 12 3 20 19 14 ...
 $ cyl : num 6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num 160 160 108 258 360 ...
 $ hp : num 110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt : num 2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num 16.5 17 18.6 19.4 17 ...
 $ vs : num 0 0 1 1 0 1 0 1 1 1 ...
 $ am : num 1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num 4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num 4 4 1 1 2 1 4 2 2 4 ...

```

Summary statistics To see summary statistics on each column (mean, median, min, max, range), we can use `summary()`

```

summary(mtcars) #summarizes each column

```

mpg	cyl	disp	hp
Min. : 1.00	Min. :4.000	Min. : 71.1	Min. : 52.0
1st Qu.: 6.75	1st Qu.:4.000	1st Qu.:120.8	1st Qu.: 96.5
Median :14.00	Median :6.000	Median :196.3	Median :123.0
Mean :13.16	Mean :6.188	Mean :230.7	Mean :146.7
3rd Qu.:19.00	3rd Qu.:8.000	3rd Qu.:326.0	3rd Qu.:180.0
Max. :25.00	Max. :8.000	Max. :472.0	Max. :335.0
drat	wt	qsec	vs

```

Min.    :2.760   Min.    :1.513   Min.    :14.50   Min.    :0.0000
1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000
Median :3.695   Median :3.325   Median :17.71   Median :0.0000
Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000

      am          gear          carb
Min.   :0.0000   Min.   :3.000   Min.   :1.000
1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
Median :0.0000   Median :4.000   Median :2.000
Mean   :0.4062   Mean   :3.688   Mean   :2.812
3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
Max.   :1.0000   Max.   :5.000   Max.   :8.000

```

Counting rows and columns To see the number of rows and columns we can use `nrow()` and `ncol()`

```
nrow(mtcars) #gives number of rows
```

```
[1] 32
```

```
ncol(mtcars) #gives number of columns
```

```
[1] 11
```

Naming dataframes Rename `mtcars` and view in Environment tab in Rstudio

```
a<-mtcars
a
```

```

      mpg cyl  disp  hp drat   wt  qsec vs am gear carb
Mazda RX4           16   6 160.0 110 3.90 2.620 16.46 0  1   4   4
Mazda RX4 Wag       16   6 160.0 110 3.90 2.875 17.02 0  1   4   4
Datsun 710          19   4 108.0  93 3.85 2.320 18.61 1  1   4   1
Hornet 4 Drive      17   6 258.0 110 3.08 3.215 19.44 1  0   3   1
Hornet Sportabout  13   8 360.0 175 3.15 3.440 17.02 0  0   3   2
Valiant             12   6 225.0 105 2.76 3.460 20.22 1  0   3   1
Duster 360          3   8 360.0 245 3.21 3.570 15.84 0  0   3   4
Merc 240D           20   4 146.7  62 3.69 3.190 20.00 1  0   4   2

```

Merc 230	19	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	14	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	11	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	9	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	10	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	6	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	1	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	1	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	4	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	24	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	23	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	25	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	18	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	7	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	6	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	2	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Pontiac Firebird	14	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Fiat X1-9	22	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	21	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	23	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Ford Pantera L	8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
Ferrari Dino	15	6	145.0	175	3.62	2.770	15.50	0	1	5	6
Maserati Bora	5	8	301.0	335	3.54	3.570	14.60	0	1	5	8
Volvo 142E	17	4	121.0	109	4.11	2.780	18.60	1	1	4	2

```
head(a)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	16	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	16	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	19	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	17	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	13	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	12	6	225	105	2.76	3.460	20.22	1	0	3	1

2.3 Write data to file (saving data)

We use the write.csv function here. a= the name of the dataframe and the name we want to give the file goes after 'file=' The file name must be in quotes and must include an extension. Since we are using write.csv we MUST use .csv

```
write.csv(a, file='mtcars.csv')
```

2.4 Read a file in (import data into R)

NOTE: if you have a .xls file make sure you convert to .csv. Ensure the file is clean and orderly (rows x columns). Only 1 excel tab can be in each .csv, so plan accordingly. Note that in order to read a file in to R from your computer (or cloud server), that file **MUST** be located within your working directory (or you must know and enter the file path).

IF your file is in your working directory, you can read it in like this:

```
b<-read.csv('mtcars.csv')
head(b)
```

		X	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	
1		Mazda	RX4	16	6	160	110	3.90	2.620	16.46	0	1	4	4
2		Mazda	RX4 Wag	16	6	160	110	3.90	2.875	17.02	0	1	4	4
3		Datsun	710	19	4	108	93	3.85	2.320	18.61	1	1	4	1
4		Hornet	4 Drive	17	6	258	110	3.08	3.215	19.44	1	0	3	1
5		Hornet	Sportabout	13	8	360	175	3.15	3.440	17.02	0	0	3	2
6		Valiant		12	6	225	105	2.76	3.460	20.22	1	0	3	1

You are welcome to use other functions to read in data (including `read_csv` or `read.xls`). Especially for beginners, I strongly encourage you to use .csv format. Other file formats can get complicated (often unnecessarily complicated). That said, R can also handle .txt, .xls, images, shapefiles (for spatial analysis or GIS style work), etc. It is very versatile! Feel free to explore :)

A note on `read_csv` -> I consider this to be the “best” option for reading in .csv files. It is a ‘smarter’ version of `read.csv` and can automatically figure out what kind of data (numeric, factor, date, etc) each column is. If you use `read.csv`, you have often have to manually change these options.

2.5 Read a file from an online source

In some cases you may be using data you’ve found online. Perhaps you can download, save, and then read your file into R. Sometimes that is more work than we want to do. You can just call a file directly from it’s URL. Here is an example:

I have a dataframe on coral cover from Belize that I want to read in. It is located on my github [coral cover data](#). Let’s read it directly into R! The URL you see above is **NOT** what

we use in R. If you find a file on Github you want to locate the ‘raw’ version of the file. To do this:

- 1.) Click the link above (or find a data file on github)
- 2.) Navigate to the top right menu and look for the box that says “Raw” in it. You can click on that and open the raw file and then copy the URL. OR, you can click the box next to the “Raw” box to copy the link to the raw file. We use this link to read our data into R. This will work for any .csv you find on github. I like to get practice data from the TidyTuesday project on Github. You can find their data at the following link:

[Tidy Tuesday Data](#)

```
coralcover<- read_csv('https://raw.githubusercontent.com/jbaumann3/BIOL234_Biostats_MHC/ma
```

```
Rows: 77 Columns: 6
```

```
-- Column specification -----
```

```
Delimiter: ","
```

```
chr (1): type
```

```
dbl (5): site, lat, transect, diver, cc_percent
```

```
i Use `spec()` to retrieve the full column specification for this data.
```

```
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(coralcover)
```

```
# A tibble: 6 x 6
```

	site	type	lat	transect	diver	cc_percent
	<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>
1	1	Back Reef	3	1	4	5.84
2	1	Back Reef	3	2	4	0.951
3	1	Back Reef	3	3	4	5.24
4	1	Back Reef	3	4	5	5.00
5	1	Back Reef	3	5	5	5.90
6	2	Patch Reef	3	1	4	5.28
