CSI 2300: Introduction to Data Science

Lecture 02: Basic Navigation

Today's Topics

Math Operations

Getting Help

Directories

Loading Data

Packages

Math Operations

At its core, R is a giant calculator, and it can perform all of the basic functions of a calculator.

Here are some of the mathematical functions to use in R:

- Addition: +
- Subtraction: -
- Multiplication: *
- Division: /
- Order of operations: Operations inside of parentheses are done first
- Power: The caret key, ^ , is used as follows: 5^2
- Square root: `sqrt()`
- Exponential: `exp()`
- Sine: `sin()`
- Cosine: `cos()`
- Log base e: `log()`

Example, Distance Formula: Find the distance between the points (3.0, 5.4) and (7.2, 8.1) using the distance formula,

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

```
x1 <- 3.0
y1 <- 5.4
x2 <- 7.2
y2 <- 8.1

D <- sqrt((x1 - x2)^2 + (y1 - y2)^2)
```

```
sqrt((x1-x2)^2 + (y1-y2)^2)
# [1] 4.992995

D
# [1] 4.992995
```

Example, Compound Interest: Imagine that you have 10, 000*toinvest*, *andyouare*20*yearsold*.*Youplantoi*
$P\left(1+\frac{r}{n}\right)^{nt}$ $

- P is the principal amount that you invest
- r is the annual interest rate in decimal form
- n is the number of times the interest is compounded per unit of t
- t is the time that the money is invested

Here, $P = 10,000$, $r = 0.09$, $n = 365$, and $t = 45$.

```
P <- 10000
r <- 0.09
n <- 365
t <- 25

A <- P * (1 + r / n)^(n * t)
A
# [1] 94851.05
```

Example, Daily High Temperature: The daily temperatures (min, max, average) for Waco, Texas can be downloaded from the National Weather Service[1]. We have the daily high temperatures for July and December, measured in Fahrenheit. Convert them to Celsius.

```
weather_Jul <- read.table(file = "dat/waco_july_daily_temps.txt", header=T)
weather_Dec <- read.table(file = "dat/waco_december_daily_temps.txt", header=T)

#Shows the first six rows of this file
head(weather_Jul)
#   DY MAX MIN AVG DEP HDD CDD  WTR SNW DPTH  SPD SPD.1 DIR MIN.1 PSBL S.S WX
# 1  1  95  80  88   4   0  23 0.00   0    0 13.8    23 180     M    M   4 NA
# 2  2  98  79  89   5   0  24 0.00   0    0  9.3    17 180     M    M   2  8
# 3  3 100  75  88   4   0  23 0.00   0    0  8.9    23 120     M    M   0 NA
# 4  4 100  69  85   1   0  20 0.00   0    0  7.8    14 170     M    M   0 NA
# 5  5  96  78  87   3   0  22 0.00   0    0  7.1    13 170     M    M   1 NA
```

[1] https://www.weather.gov/fwd/actclimo

```
# 6  6  95  75  85   1   0  20 0.11   0    0  6.8    24 340      M    M   4 13
#   SPD.2  DR
# 1    30 180
# 2    24 190
# 3    28 130
# 4    18 100
# 5    17 180
# 6    32 340
#head(weather_Dec)

#Shows the last six rows of this file
#tail(weather_Jul)
#tail(weather_Dec)

dim(weather_Dec)
# [1] 31 19
nDec <- nrow(weather_Dec)

#Shows the column names
colnames(weather_Jul)
#  [1] "DY"    "MAX"   "MIN"   "AVG"   "DEP"   "HDD"   "CDD"   "WTR"   "SNW"
# [10] "DPTH"  "SPD"   "SPD.1" "DIR"   "MIN.1" "PSBL"  "S.S"   "WX"    "SPD.2"
# [19] "DR"

#Shows the data structure
str(weather_Jul)
# 'data.frame': 31 obs. of  19 variables:
#  $ DY   : int  1 2 3 4 5 6 7 8 9 10 ...
#  $ MAX  : int  95 98 100 100 96 95 83 96 97 99 ...
#  $ MIN  : int  80 79 75 69 78 75 72 75 79 77 ...
#  $ AVG  : int  88 89 88 85 87 85 78 86 88 88 ...
#  $ DEP  : int  4 5 4 1 3 1 -6 1 3 3 ...
#  $ HDD  : int  0 0 0 0 0 0 0 0 0 0 ...
#  $ CDD  : int  23 24 23 20 22 20 13 21 23 23 ...
#  $ WTR  : chr  "0.00" "0.00" "0.00" "0.00" ...
#  $ SNW  : num  0 0 0 0 0 0 0 0 0 0 ...
#  $ DPTH : int  0 0 0 0 0 0 0 0 0 0 ...
#  $ SPD  : num  13.8 9.3 8.9 7.8 7.1 6.8 4.6 10.5 13.4 10.7 ...
#  $ SPD.1: int  23 17 23 14 13 24 24 18 22 21 ...
#  $ DIR  : int  180 180 120 170 170 340 60 180 180 180 ...
#  $ MIN.1: chr  "M" "M" "M" "M" ...
#  $ PSBL : chr  "M" "M" "M" "M" ...
#  $ S.S  : int  4 2 0 0 1 4 6 3 2 2 ...
#  $ WX   : int  NA 8 NA NA NA 13 13 NA NA NA ...
#  $ SPD.2: int  30 24 28 18 17 32 32 23 29 26 ...
#  $ DR   : int  180 190 130 100 180 340 60 190 170 170 ...

#Shows the type of object, more on this in the next lecture.
class(weather_Jul)
# [1] "data.frame"

#The dollar sign can be used to extract a particular column from the dataset.
#More on that in the next lecture.
#General principle:  always use the data from the source...no need to create replicate
max_daily_temp_jul <- weather_Jul$MAX
max_daily_temp_dec <- weather_Dec$MAX

sort(max_daily_temp_jul)
#  [1]  83  90  91  93  93  95  95  95  95  96  96  96  96  96  96  97  97  97  97
# [20]  97  97  98  99  99  99 100 100 100 101 103 103
sort(max_daily_temp_dec)
#  [1] 42 47 49 49 51 51 56 59 59 60 61 61 61 62 63 63 64 64 64 66 67 67 71 71 72
# [26] 73 77 77 78 78 80

length(max_daily_temp_jul)
# [1] 31
length(max_daily_temp_dec)
# [1] 31

summary(max_daily_temp_dec)
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
#   42.00   59.00   63.00   63.32   71.00   80.00 
summary(weather_Jul$WX)
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
#    1.00    3.00    8.00    7.75   13.00   13.00      23 
#summary(weather_Jul)

mean(weather_Jul$WX)
# [1] NA

max_daily_temp_jul_C <- (max_daily_temp_jul - 32) * 5 / 9; #sort(max_daily_temp_jul_C)
max_daily_temp_dec_C <- (max_daily_temp_dec - 32) * 5 / 9; #sort(max_daily_temp_dec_C)
```

There are also many basic summary functions in R that can be used for simple operations, such as

- Average: `mean()`

- Sum: `sum()`
- Minimum: `min()`
- Maximum: `max()`
- Median: `median()`

Example, Daily High Temperature: Using the temperature data from before, use the sum function and the mean function to compute the average daily temperature in July and December. Find the minimum and maximum high temperatures in each month.

Getting Help

There are multiple ways to get help in R. Learning to read the documentation can have a steep learning curve, but it should be your first point of call after receiving an error message at the console!

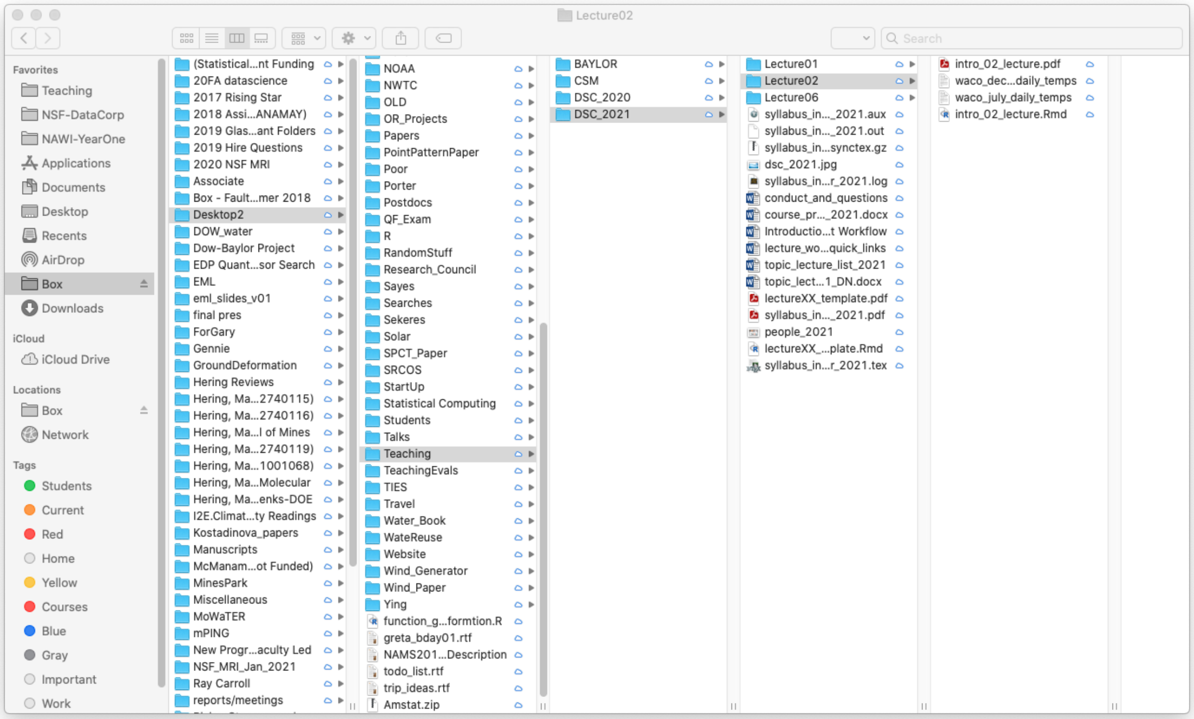
1. **The question mark** If you already know the name of the function, for example, type ?mean at the command line. Or, if you type the name of the function in RStudio and pause for a moment, a pop-up will tell you what the function does and give some of its arguments.
2. **Type the function name.** If you already know the name of the function, you can just type the function name at the command line with no parentheses or anything after it. The code called by the function will be printed to the screen. It isn't always helpful.
3. **help.search()** If you are not sure of a function name, type the following at the command line and see what you get:

```
help.search("uniform numbers")
help.search("uniform distribution")
```

4. **Google it.** Pose your question and type "in R" " or use "[R]" in your search. This will often lead you to a question with answers at stackoverflow.com, a website to ask and answer questions about coding.
5. **https://rseek.org/** is a special search engine site that searches through R resources like R bloggers and R Consortium links

Directories

A directory is oftentimes known as a folder in your computer. Many times, we work without giving much thought to these folders or to their specific locations, but if you are loading data into R or saving figures or R scripts, then you want to know what folder you are in. The following is a visual representation of the folder structure on a Mac, and Windows operating systems use something similar.



Another way to think of the directory structure is with a hierarchy or as a tree. Finally, a third way to represent the working directory is with a file path. The corresponding file path to the visual above is:

```r
getwd()
# [1] "/Users/caldera/Library/CloudStorage/Box-Box/CSI 2300 RW/Lecture02_02_AH_GH"
```

The names of the folders are separated by the slashes. Microsoft uses backslashes instead of forward slashes. Having this file path or knowing what it is will be important because any new datasets that you may create will be saved here unless otherwise specified, as well as any figures. Oftentimes, if you are struggling to load a dataset into R, you are pointed to a directory where that file does not live.

You can do the following:

- `getwd()`: to find what working directory you are currently in
- `setwd()`: to change the working directory
- Go to Session -> Set Working Directory -> Choose Directory
- Use `file.choose()` to open a dialogue box to navigate to the folder where the data is stored

Possible workflows for this course:

1. Download folder > Click file of interest

- I recommend you download the entire linked folder from the website. It will land in your Downloads. (Optional but **highly recommended** copy it over into the course folder where you'd like your files to live for this course.)
- If you click the file you want to work in **and** you do not have any previously open R windows, R will automatically open up the file in the working directory where the file lives. However, if you have any other open windows, the file will open up in the current working directory, which is most likely not where you want to be!

2. Download the folder > Open new Project in R Studio

- Project files (.Rproj) are bundles that have any R scripts, markdown files, figures, .Rdata and more all bundled together. They're great if you want to be able to close R Studio and pick up exactly where you left off! They're also a stepping stone to buildling package development and working in github.

3. Go rogue and download only the files of interest

- If you do this, you'll need to be a bit clever about making sure your working directory and the directory where your data lives match up when it comes time to read in your data. If you are lost `getwd()` will always tell you what directory you're in!

Loading Data

There are a lot of different file types, and they are each loaded in unique ways.

- .csv: One of the most common file types is the .csv file, which stands for "comma separated values." It is loaded with the command:

`read.csv()`

- .xls: These indicate an Excel spreadsheet. You need an R package to read in spreadsheets and to select the corresponding page in the spreadsheet (if there are multiple pages).

`library(xlsx)` is one that I have used for Excel files

`read.xlsx()`

- .txt: These are plain text files. They are loaded with the command:

`read.table()`

- .Rdata

`load()` is used for .Rdata files

These are great to use if you have a very large dataset because they load fast.

- .rds

`readRDS()` is used for .rds files

Also good for large datasets, and they tend have a more stable loading than .Rdata files.

You will see many of these as we go through the semester. Other file types are possible, such as files that use the structure of another major statistical software package like SAS (.xpt) and SPSS (.sav). There are also ways to import data directly from a webpage using the website address, as follows:

```r
website <- "https://www.waterdatafortexas.org/reservoirs/statewide.csv"
reservoirs <- read.csv(file=website, header=T, skip=29)
```

Example, Daily High Temperature: Create a folder for the material for this class. We suggest creating a subfolder inside of this main folder for every lecture. Download the two Waco temperature datasets, and save them in a folder for lecture 02. Use `file.choose()` to find the path to this dataset on your computer. Then, use `read.table()` to read the data into R.

Packages

Because R is free, many people create additional functions and helpful utilities. These do not come in the set of "base" R functions, but they can be installed and called into working memory. The list of available packages available to download is very long[2]. As of March 2023, there are over 19,000 of them! Don't worry… we will only use a few of these.

Installing packages can be done in several ways:

- The top tabs: Tools −> Install Packages
- At the command line: `install.packages()` with the package name in quotes inside
- With the packages window (will do a demo)

A package only needs to be installed once, but it must be loaded into working memory each time an R session is initiated. To load an installed package into working memory, use the command:

`library(package.name)`

To illustrate, we will install and load the `lubridate` package, which we will use frequently throughout the class.

```
#install.packages("lubridate")
#library(lubridate)
#?lubridate
```

We will also use our own `mowateR` package that contains many large datasets that we will use in the class. This package is not yet housed on R's official CRAN website and is still under development.

```
#install.packages("remotes")
#remotes::install_github("lukedurell/mowater")
#library(mowater)
#?mowater
```

[2] https://cran.r-project.org/web/packages/available_packages_by_name.html