3: Coding Basics

Environmental Data Analytics | Kateri Salk

Spring 2020

Objectives

1. Discuss and navigate different data types in R
2. Create, manipulate, and explore datasets
3. Call packages in R

Data Types in R

R treats objects differently based on their characteristics. For more information, please see: https://www.statmethods.net/input/datatypes.html.

- **Vectors** 1 dimensional structure that contains elements of the same type.
- **Matrices** 2 dimensional structure that contains elements of the same type.
- **Arrays** Similar to matrices, but can have more than 2 dimensions. We will not delve into arrays in depth.
- **Lists** Ordered collection of elements that can have different modes.
- **Data Frames** 2 dimensional structure that is more general than a matrix. Columns can have different modes (e.g., numeric and factor). When we import csv files into the R workspace, they will enter as data frames.

Define what each new piece of syntax does below (i.e., fill in blank comments). Note that the R chunk has been divided into sections (# at beginning of line, —- at end)

```
# Vectors ----
vector1 <- c(1,2,5.3,6,-2,4) # numeric vector
vector1
```

```
## [1]  1.0  2.0  5.3  6.0 -2.0  4.0
```

```
vector2 <- c("one","two","three") # character vector
vector2
```

```
## [1] "one"   "two"   "three"
```

```
vector3 <- c(TRUE,TRUE,TRUE,FALSE,TRUE,FALSE) #logical vector
vector3
```

```
## [1]  TRUE  TRUE  TRUE FALSE  TRUE FALSE
```

```
vector1[3] # the 3rd item in vector 1
```

```
## [1] 5.3
```

```
# Matrices ----
matrix1 <- matrix(1:20, nrow = 5,ncol = 4) # 1-20 range starts in columns
matrix1
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    6   11   16
## [2,]    2    7   12   17
## [3,]    3    8   13   18
```

```
## [4,]    4    9   14   19
## [5,]    5   10   15   20
```

```
matrix2 <- matrix(1:20, nrow = 5, ncol = 4, byrow = TRUE) # 1-20 are ranged by rows
matrix2
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
## [4,]   13   14   15   16
## [5,]   17   18   19   20
```

```
matrix3 <- matrix(1:20, nrow = 5, ncol = 4, byrow = TRUE, # return after comma continues the line
                  dimnames = list(c("uno", "dos", "tres", "cuatro", "cinco"),
                                  c("un", "deux", "trois", "cat"))) # dimension names (row, column)

matrix1[4, ] # row 4
```

```
## [1]  4  9 14 19
```

```
matrix1[ , 3] # column 3
```

```
## [1] 11 12 13 14 15
```

```
matrix1[c(12, 14)] # 12th and 14th objective
```

```
## [1] 12 14
```

```
matrix1[c(12:14)] #  12th-14th objectives
```

```
## [1] 12 13 14
```

```
matrix1[2:4, 1:3] # row 2-4 and column 1-3
```

```
##      [,1] [,2] [,3]
## [1,]    2    7   12
## [2,]    3    8   13
## [3,]    4    9   14
```

```
matrix3[c(12,14)] # 12th objective and 14th objective in matrix 3 (counting vertically by default)
```

```
## [1]  7 15
```

```
cells <- c(1, 26, 24, 68)
rnames <- c("R1", "R2")
cnames <- c("C1", "C2")
matrix4 <- matrix(cells, nrow = 2, ncol = 2, byrow = TRUE,
  dimnames = list(rnames, cnames)) # name row and columns - order: row, column
matrix4
```

```
##    C1 C2
## R1  1 26
## R2 24 68
```

```
# Lists ----
list1 <- list(name = "Maria", mynumbers = vector1, mymatrix = matrix1, age = 5.3); list1
```

```
## $name
## [1] "Maria"
##
```

```
## $mynumbers
## [1]  1.0  2.0  5.3  6.0 -2.0  4.0
##
## $mymatrix
##      [,1] [,2] [,3] [,4]
## [1,]    1    6   11   16
## [2,]    2    7   12   17
## [3,]    3    8   13   18
## [4,]    4    9   14   19
## [5,]    5   10   15   20
##
## $age
## [1] 5.3
```

```
list1[[2]]
```

```
## [1]  1.0  2.0  5.3  6.0 -2.0  4.0
```

```
# Data Frames ----
d <- c(1, 2, 3, 4) # What type of vector? numeric
e <- c("red", "white", "red", NA) # What type of vector? character
f <- c(TRUE, TRUE, TRUE, FALSE) # What type of vector? logic
dataframe1 <- data.frame(d,e,f) # combine all above in matrix
names(dataframe1) <- c("ID","Color","Passed"); View(dataframe1) # "names"" is a function for columns
```

```
## Warning in system2("/usr/bin/otool", c("-L", shQuote(DSO)), stdout = TRUE):
## running command ''/usr/bin/otool' -L '/Library/Frameworks/R.framework/
## Resources/modules/R_de.so'' had status 1
```

```
dataframe1[1:2,] # row, column -- the exact first row
```

```
##   ID Color Passed
## 1  1   red   TRUE
## 2  2 white   TRUE
```

```
dataframe1[c("ID","Passed")] #
```

```
##   ID Passed
## 1  1   TRUE
## 2  2   TRUE
## 3  3   TRUE
## 4  4  FALSE
```

```
dataframe1$ID
```

```
## [1] 1 2 3 4
```

Question: How do the different types of data appear in the Environment tab?

Answer: the abbraviation of the type of data is illustrated in each variable

Question: In the R chunk below, write "dataframe1$". Press `tab` after you type the dollar sign. What happens?

Answer: a list of all columns in the dataframe was shown

Coding challenge

Find a ten-day forecast of temperatures (Fahrenheit) for Durham, North Carolina. Create two vectors, one representing the high temperature on each of the ten days and one representing the low.

```
highf <- c(65,45,50,50,40,40,45,52,54,58)
lowf <- c(31,29,43,24,23,23,26,30,42,41)
```

Now, create two additional vectors that include the ten-day forecast for the high and low temperatures in Celsius.

```
highc <- round((highf-32)*5/9, 0)  #c(17,7,10,10,4,4,7,11,12,14)
highc
```

```
##  [1] 18  7 10 10  4  4  7 11 12 14
```

```
lowc <- round((lowf-32)*5/9,0) #c(-1,-1,6,-4,-5,-5,-3,-1,5,5)
lowc
```

```
##  [1] -1 -2  6 -4 -5 -5 -3 -1  6  5
```

Combine your four vectors into a data frame and add informative column names.

```
dataframetemp <- data.frame(highf, highc, lowf,lowc)
names(dataframetemp) <- c("High (F)", "High (C)", "Low (F)", "Low (C)")
dataframetemp
```

```
##    High (F) High (C) Low (F) Low (C)
## 1        65       18      31      -1
## 2        45        7      29      -2
## 3        50       10      43       6
## 4        50       10      24      -4
## 5        40        4      23      -5
## 6        40        4      23      -5
## 7        45        7      26      -3
## 8        52       11      30      -1
## 9        54       12      42       6
## 10       58       14      41       5
```

Use the common functions **summary** and **sd** to obtain basic data summaries of the ten-day forecast. How would you call these functions differently for the entire data frame vs. a single column? Attempt to demonstrate both options below.

```
summary(dataframetemp)
```

```
##     High (F)       High (C)        Low (F)        Low (C)
##  Min.   :40.0   Min.   : 4.00   Min.   :23.0   Min.   :-5.00
##  1st Qu.:45.0   1st Qu.: 7.00   1st Qu.:24.5   1st Qu.:-3.75
##  Median :50.0   Median :10.00   Median :29.5   Median :-1.50
##  Mean   :49.9   Mean   : 9.70   Mean   :31.2   Mean   :-0.40
##  3rd Qu.:53.5   3rd Qu.:11.75   3rd Qu.:38.5   3rd Qu.: 3.50
##  Max.   :65.0   Max.   :18.00   Max.   :43.0   Max.   : 6.00
```

```
sd(dataframetemp$`High (F)`)
```

```
## [1] 7.880355
```

```
sd(dataframetemp$`High (C)`)
```

```
## [1] 4.398232
```

Packages

The Packages tab in the notebook stores the packages that you have saved in your system. A checkmark next to each package indicates whether the package has been loaded into your current R session. Given that R is an open source software, users can create packages that have specific functionalities, with complicated code "packaged" into a simple commands.

If you want to use a specific package that is not in your libaray already, you need to install it. You can do this in two ways:

1. Click the install button in the packages tab. Type the package name, which should autocomplete below (case matters). Make sure to check "intall dependencies," which will also install packages that your new package uses.
2. Type `install.packages("packagename")` into your R chunk or console. It will then appear in your packages list. You only need to do this once.

If a package is already installed, you will need to load it every session. You can do this in two ways:

1. Click the box next to the package name in the Packages tab.
2. Type `library(packagename)` into your R chunk or console.

```
# We will use the packages dplyr and ggplot2 regularly.
#install.packages("dplyr")
#install.packages("ggplot2")
# comment out install commands, use only when needed and re-comment

library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.5.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

```
# Some packages are umbrellas under which other packages are loaded
#install.packages("tidyverse")
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.5.2
```

```
## -- Attaching packages ---------------------------------------------------- tidyverse 1.3.0 --
```

```
## v tibble  2.1.3     v purrr   0.3.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'tibble' was built under R version 3.5.2
```

```
## Warning: package 'tidyr' was built under R version 3.5.2
```

```
## Warning: package 'purrr' was built under R version 3.5.2
```

```
## Warning: package 'stringr' was built under R version 3.5.2

## Warning: package 'forcats' was built under R version 3.5.2

## -- Conflicts ---------------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Question: What happens in the console when you load a package?

Answer:

Tips and Tricks

- Sequential section headers can be created by using at least four -, =, and # characters.
- The command `require(packagename)` will also load a package, but it will not give any error or warning messages if there is an issue.
- You may be asked to restart R when installing or updating packages. Feel free to say no, as this will obviously slow your progress. However, if the functionality of your new package isn't working properly, try restarting R as a first step.
- If asked "Do you want to install from sources the packages which needs compilation?", type `yes` into the console.
- You should only install packages once on your machine. If you store `install.packages` in your R chunks/scripts, comment these lines out.
- Update your packages regularly!