**p-hacking**

**R Objects and Attributes**

* 5 objects – “atomic” classes
  + - Character
    - Numeric
    - Integer
    - Complex
    - Logical – TRUE/FALSE
* Vectors
  + - Everything in a vector must be of **the same class**
    - Empty vector: vector()
* Lists
  + - Vector of objects of **different** classes
* To find class:
  + - Class()
* To find length:
  + - Length()

**Vectors and Lists**

* **Creating vector:**
  + - x <- c(0.5, 0.6)
  + or
    - x <- vector(“numeric”, length=10)
* Converting classes:
  + - As.numeric(x)
    - As.logical(x)
    - As.character(x)
* **Lists**
  + - A type of vector that can contain elements of different classes
    - Create a list
      * **X <- list(1, “a”, TRUE, 1+4i)**

**Matrices**

* Vectors with a dimension attribute
  + - M <- matrix(nrow = 2, ncol = 3)
    - Attributes(m)
* M <- matrix(1:6, nrow = 2, ncol = 3)
* Or:
  + - M <- 1:10
    - Dim(m) <- c(2, 5)
* Cbind and rbind:
  + - X
    - Y
    - Cbind(x,y)
    - Rbind(x,y)

**Factors**

* Are used to represent unordered data
  + - Lm()
    - Glm()
* Create a factor
  + - X <- factor(c(“yes”, “yes”, “no”, “yes”, “no”))
    - Levels = c(“yes”, “no”)

**Missing values**

* NA: not available, NaN: not a number
* Is.na()
* Is.nan()

**Data Frames**

* Are used to store tabular data
* Row.names
* Created by:
  + - Read.table()
    - Read.csv()
  + Or
    - X <- data.frame(foo = 1:4, bar = c(T, T, F, F)
* Convert to matrix
  + - Data.matrix()

**Names attribute**

* R objects can also have names
* For vectors
  + - X <- 1:3
    - (x) <- c(“foo”, “bar”, “norf”)
* For lists
  + - X <- list(a = 1, b = 2, c = 3)
* For matrices
  + - Matrix m
    - Dimnames(m) <- list(c(“a”, “b”), c(“c”, “d”))

**Data types – Summary**

* Classes: numeric, logical, character, integer, complex
* Vectors, lists
* Factors
* Missing values
* Data frames
* Names

**Reading tabular data**

* **Read data**
  + - Data <- **Read.table(“D:\R\...”)**
      * + File
        + Header
        + Sep
        + colClasses
        + nrows
        + comment.char
        + skip
        + stringsAsFactors
    - **Read.csv**
    - readLines
    - source
    - dget
    - load
    - unserialize
* **Write data**
  + - write.table
    - writeLines
    - dump
    - dput
    - save
    - serialize

**Reading large tables?**

* Read the help page for read.table (PLEASE)
* Set comment.char = “”
* Set **nrows** small/enough

**Textual data formats**

* **Dump**
  + - X <- “foo”
    - Y <- data.frame(a = 1, b = “a”)
    - Dump(c(“x”, “y”), file = “data.R”)
    - Rm(x, y)
    - Source(“data.R”)
* **Dput** 
  + - Y <- data.frame(a = 1, b = “a”)
    - Dput(y, file = “y.R”)
    - New.y <- dget(“y.R”)
* **Dget**

**Connections: Interfaces to the Outside World**

* File
  + - Open a connection to a file
* Gzfile
  + - For gzip file
* Bzfile
  + - For bzip file
* url
  + - for URLs

**Subsetting – Basics**

* how to extract subsets of R objects?
  + - [
    - [[
    - $
* X <- c(“a”, “b”, “c”, “c”, “d”, “a”)
* X[1], x[2], x[1:4]
* X[x > “a”]
* Or
  + - U <- x > “a”
    - X[u] (same result)

**Subsetting – Lists**

* Example
  + - X <- list(foo = 1:4, bar = 0.6)
    - X[1]
    - X[[1]]
    - X$bar
    - X[[“bar”]]
      * Can be used with computed indexes
      * Can take an integer sequence
    - X[“bar”]
    - $bar
      * Only for literal names
* Or
  + - X[c(1,3)] can also do it

**Subsetting – Matrix**

* Matrices can be subsetted in the usual way
  + - X <- matrix(1:6, 2, 3)
    - X[1, 2]
    - X[1, ]
    - X[, 2]
* By default, a single element/single row extracted from a matrix is a vector of length 1. To turn this off?
  + - X[1, 2, drop = FALSE]

**Subsetting – Partial Matching**

* Partial matching of names is allowed with [[ and $
  + - X < list(aardvark = 1:5)
    - But you can use a instead of aardvark
    - X$a
  + Or
    - X[[“a”, exact = FALSE]] – to specify that the word doesn’t need to be like, exactly, hence the name “Partial” Matching

**Subsetting – Removing Missing Values**

* Example
  + - X <- c(1, 2, NA, 4, NA, 5)
    - Bad <- is.na(x)
    - X[!bad]
* For multiple things?
  + - X
    - Y <- c(“a”, “b”, NA, “d”, NA, “f”)
    - Good <- **complete.cases(x, y)**
    - X[good]
    - Y[good][1:6] is still possible

**Vectorized Operations**

* Vectorized 🡺 efficient, concise, easier to read
* Example
  + - X <- 1:4, y <- 6:9
    - The operations are going to be done in **parallel**
* **To REALLY multiply matrices:**
  + - X %\*% y