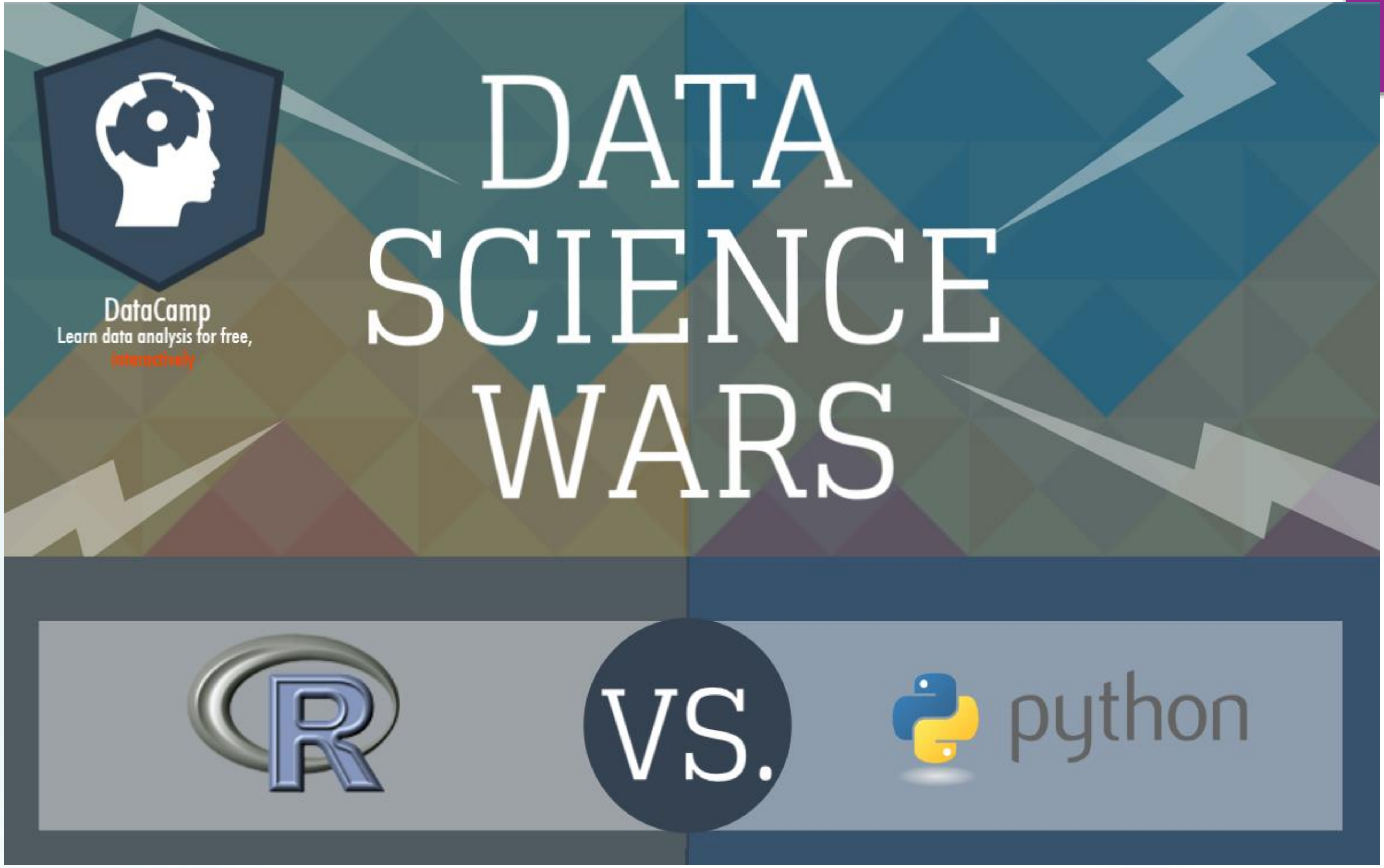
R for statistical computing

Yu Sheng

Center for Statistical Science

Fall 2018



https://www.datacamp.com/community/tutorials/r-or-python-for-data-analysis

Installing R

- CRAN (The Comprehensive R Archive Network)
- Mirror@TUNA
  https://mirrors.tuna.tsinghua.edu.cn/CRAN/

Running R

- R GUI (the “double-click”)
  - Console
  - Code editor
  - Graphics

- R can be run from the command line (terminal)

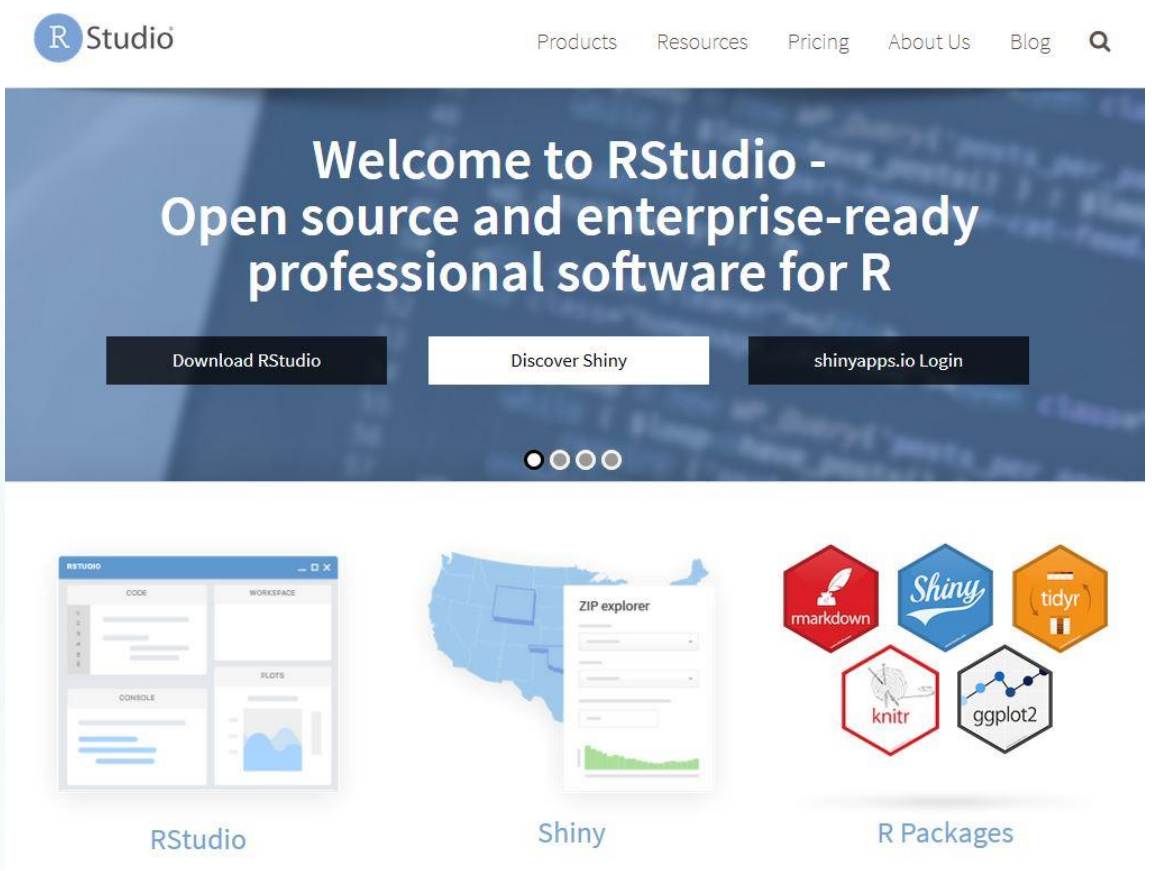
- `R CMD BATCH` (the old way)
- `Rscript` (the new way)
  - `Use '>' to redirect the output.`

Setting the working directory

- `getwd()`
- `setwd()`

Installing RStudio

- https://www.rstudio.com/



R packages

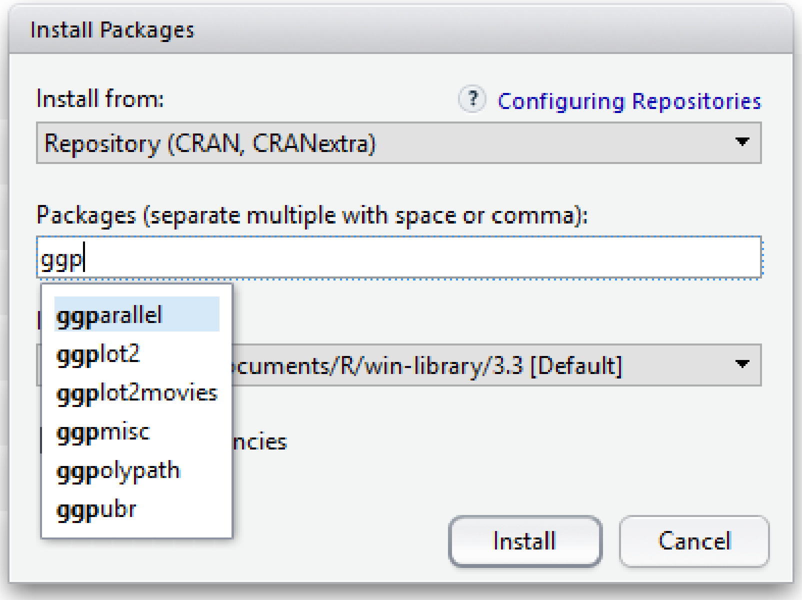
- *Packages* are collections of R functions, data, and compiled code in a well-defined format.

- The directory where packages are stored on your computer is called the *library*.

- `.libPaths()` – shows the library location
  `library()` – shows the installed packages
  `library(package_name)` – loads packages

Install R packages

- Use the functions `install.packages()`, or
- RStudio Tools -> Install Packages


Install Packages
Install from:
? Configuring Repositories
Repository (CRAN, CRANextra)
Packages (separate multiple with space or comma):
ggp
ggparallel
ggplot2
ggplot2movies
ggpmisc
ggpolypath
ggpubr
cuments/R/win-library/3.3 [Default]
ncies
Install
Cancel

Let’s get started…

- Try the following in R
  - `x = 1`
  - `x`
  - `x = “data science”`
  - `x`

动态类型，不同于C/C++的int double这样的静态类型

[1]表示结果的第一个元素， R 都是用向量来表示结果的

- What do you see?

默认浮点型数
如果真的想用整型需要“x=1L”

- Btw, naming conventions and coding style: https://google.github.io/styleguide/Rguide.xml

Atomic classes

- Character
- Integer
- Numeric
- Complex
- Logical

类似于string是字符串，不是字符

Basic Numerical Operators

- +
- -
- *
- /   **正常的除法**
- %% **整数除法的商数**
- ^

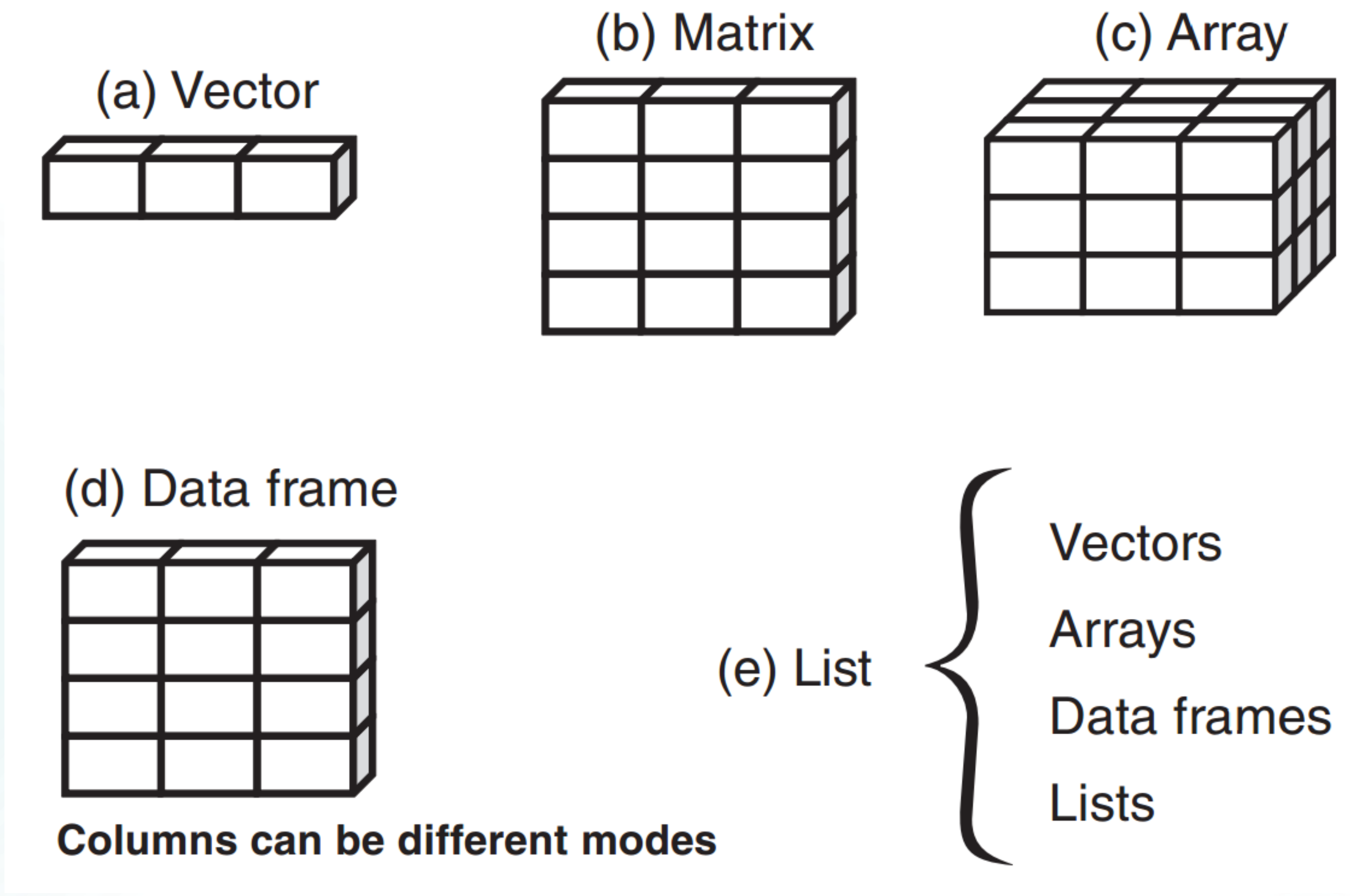
Basic Logical Operators

- ==, >, >=, <, <=, !=, !

- &, &&, |, ||, xor

单&或者单|是两个向量元素按对应元素位做逻辑与或者逻辑或返回一个向量
而双&&或者双||是只对两个向量的第一个元素进行，返回一维向量

Basic data structures



Vector

- Vectors are one-dimensional arrays 里面所有内容都是相同类型的元素
- Use `c()` or `vector(mode, length)` to initialize a vector
- Use `c()` to combine vectors 用于拼接两个相同类型的向量
- Holds data of the same class

Vector

- `:`, `seq()` and `rep()` are also useful for initializing vectors
- `seq(from, to, by, length.out)`
- `rep(…, times, length.out, each)`

Type Checking and Conversion

- Use `is.xxx()` to check the data type
  - `is.numeric()`
  - `is.vector()`

- Use `as.xxx()` to convert explicitly

Vector Operations

- Vector-vector: elementwise
  - The short vector will be looped over
  - Throws a warning when the longer object length is not a multiple of the shorter object length

不要求两向量长度相同

Quiz

Write down the answer to the following calculations (Don’t use your computer):

- `(1:8)*(1:2)`
- `(1:2)*(1:8)`
- `(1:8)*(1:3)`
- `(1:8)^(1:2)`
- `(1:2)^(1:8)`

Vector Operations

用R做loop非常慢，而向量运算则是并行的，因此尽量用向量来代替loop

- Vector-vector: elementwise
  - The short vector will be looped over
  - Throws a warning when the longer object length is not a multiple of the shorter object length
- Vector-scalar: a special case of the vector-vector operation
- “Vectorized operation”

Factor

- Factors are used to represent categorical data
  - Like *labels*

- Can be unordered or ordered
  - Unordered – nominal
  - Ordered – ordinal

- The benefit of using factors?
  - Saves space
  - Faster lookup (integer vs. string comparison)
  - Subsetting data

用来储存的是一个整数，整型的，下面是与它所代表的字符串相比

Factor

- The function `factor()` encodes a vector to factor, and the levels are unordered

- `factor(, order=T, levels=c(…))` gives a factor with ordered levels
  - Without “levels”, alphabetic order will be used.

Matrix

- Create a matrix with `matrix(data, nrow, ncol, byrow, dimnames=list(rownames, colnames))`
- The values in `data` will be used repeatedly.
- Fills by column by default. 按列储存
- Stores data by column – Implication?
- Use `cbind()` and `rbind()` to bind matrices and/or vectors.

Matrix Operations

- Transpose: `t()`

- Matrix multiplication: `%*%` **只有一个***时是对应位置进行乘法

- Inverse: `solve()` – why? solve是用来解线性方程组的

- Diagonal matrix: `diag(v)` `diag(X)`将目标矩阵的对角线取出来生成一个向量

  以给定向量生成一个
  其他地方为0的稀疏阵

Matrix Operations

- Elementwise operations: + `- * / ^`

- Matrix-matrix: requires same size

- Matrix-vector, vector-matrix
  - length(matrix) needs to be a multiple of length(vector)
  - The vector will be looped

- Scalar-matrix and matrix-scalar

Quiz

Write down the answer to the following calculations (Don’t use your computer):

- `A = matrix(1:4, 2, 2); b = c(1, 2)`
- `A%*%b`
- `b%*%A`

此时因为不能运算，所以默认将b转置后再计算

- `A*b`
- `b*A`
- `t(b)%*%A`
- `t(b)*A`

Data Frame

- Also for 2D data.

- Difference from matrix?
  - Can hold columns of different value types.
  - Arguably the most important data structure in R.

但是同一列要统一类型

- Pro: convenient data manipulation
- Con: slower computation and greater memory consumption
- See: https://stackoverflow.com/questions/5158790/should-i-use-a-data-frame-or-a-matrix

Data Frame

- Use `data.frame()` to combine vectors and matrices as a data frame.

- More commonly, data frames are created when reading data from file, using `read.table()`.

Array

- For *N*-D data
  `array(vector, dimensions, dimnames)`

- Not common in statistics…

List

- An R list is not the same “list” as in other programming languages
  - Python list: an array-like data structure
  - Java list: an “interface”
  - R list: a container of various objects

- Lists are used to freely combine variables of various types
  - `list([name1=]object1,[name2=]object2,…)`

List

- In fact, data frames are a special kind of lists, with a few restriction
  - you can't use the same name for two different variables
  - all elements of a data frame are vectors
  - all elements of a data frame have an equal length.

- `is.list(df) = TRUE`

- Therefore, functions for lists, such as `sapply()`, work on data frames, too.

Element Access

- Use [i] for vectors and factors
- Use [i, j] for matrices and data frames
- Use [[i]] for lists and data frames

Type Checking and Conversion

- `is.xxx()`, `as.xxx()` also work for the above data structures.

Attributes - size

- `length()` for vectors, factors, matrices, data frames, and lists
  - Beware of the difference of `length()` on matrices and data frames.
- `dim()`, `nrow()`, `ncol()` for matrices and data frames

Attribute - name

- `name()` for vectors, lists, and data frames

- `rownames()`, `colnames()` for matrices and data frames

- Access data via names: `["..."]`, `["...", "..."]`, `df$name`, `list$name`

Reading and Writing Data

- `read.table()` and `write.table()` for tabular data

- They are very important, because tabular data are the dominant format in statistical analysis.

`read.table()`

- `file` – file name and path
- `header` – if the file has a header line
- `sep` – a single character indicating how the columns are separated
- `colClasses` – a character vector indicating the class of each column
- `stringsAsFactors` – should character variables be coded as factors?

`read.table()`

- `read.csv()` and `read.delim()` are the same as `read.table()`, but with different default values.
- `read.csv()` for comma separated values
- `read.delim()` for tab separated values

`write.table()`

- `file` – file name and path
- `quote` – if characters should be surrounded by quotes
- `sep` – a single character indicating how the columns are separated
- `row/col.names` – if row or column names should be saved

Writing Text Data

- `sink(file, append)` for redirecting textual output to a file.
  - One can also use '>' with Rscript for redirection.

- `sink()` without argument exits the current redirecting

- `sink()` can be stacked

Read/Write in Binary Format

- `load()` and `save()` with the .RData format.
- .Rdata is fast to load.
- .Rdata saves space with data compression.

Generic Text Reader

- `readLines()`

Functions

```
f = function() {
    cat("Hello, world!\n")
}
```

Functions

```
f = function(name) {
    cat("Hello, ",name,"!\n",sep="")
}
```

Functions

```
f = function(name = "World") {
    cat("Hello, ",name,"!\n",sep="")
}
```

Functions

```
f = function(name = "World") {
    out.text = paste("Hello, ",name,"!",sep="")
    return(out.text)
}
```

Arguments are copied

```
f = function(word) {
    word = paste(word,word,"!",sep=" ")
    return(word)
}
word = "Run"
word2 = f(word)
print(word)
print(word2)
```

Multiple arguments

```
f = function(arg1, arg2, …) {
    …
    return(…)
}
```

- R uses optional arguments instead of overloading

Argument matching

- R function arguments can be matched by
  - position
  - name
- The two methods can be mixed
- Argument names can also be matched partially (I don’t recommend it)

Functional Programming

- Functions are “first-class objects” in R.

- Functions can be passed as arguments to other functions. This is very handy for the various ‘apply’ functions, like `lapply()` and `sapply()`.

- Functions can be nested, so that you can define a function inside of another function

If-else

```
if (cond) statement


if (cond)
    statement


if (cond) {
    statement
}
```

If-else

```
if (cond) statement1 else statement2
```

```
if (cond) {
    statement1
} else {
    statement2
}
```

correct

```
if (cond)
    statement1
else
    statement2
```

wrong

ifelse

- Vectorized operation

```
ifelse(test, yes, no)
```

- Values are recycled from `yes` and `no`

For

- Hidden index

```
for (var in seq) statement
```

```
for (var in seq) {
    statement
}
```

```
for (i in 1:5) {
    i = 5
    print(i)
}
```

While

```
while (cond) statement


while (cond) {
    statement
}
```

Loop is very slow!

- Try to avoid explicit loops.

- Use vectorized operations instead, such as built-in reduction functions, and `apply()`, etc.

apply

- `apply(X, MARGIN, FUN, ...)`

- `lapply(X, FUN, ...)`
- `sapply(X, FUN, ...)`

- `tapply(X, INDEX, FUN, ...)`

- BTW – `rowSums, colSums, rowMeans, colMeans`

Common useful functions

- Set functions: `c, intersect, union, setdiff, %in%, is.element, unique, duplicated`...
- Mathematical functions: `sum, abs, sqrt, sign, exp, log, round, ceiling, floor, trunc`...
- Statistical functions: `mean, median, sd, var, cor, max, min, range, quantile, table, [dpqr]{unif,norm,`...`}`...
- Character functions: saved for text mining

It is often said that 80% of data analysis is spent on the process of cleaning and preparing the data.

Dasu and Johnson, 2003

Subsetting

- Subsetting – selecting or excluding variables and observations

- By position
  - `x[4]`
  - `x[-4]`
  - `x[2:4]`
  - `x[-(2:4)] or x[-2:-4]`
  - `x[c(1,5)]`

Subsetting

- By condition (vector of T/F)
  - `x[x == 10]`
  - `x[x < 0]`
  - `x[x %in% c(1, 2, 5)]`
- By name (vector of column names)
  - `x[, "PM2.5"]`
  - `x[, c("PM2.5", "PM10")]`

Subsetting with Assignment

- Subsetting operators can be combined with assignment to modify selected values of the input vector.

Preserving subsetting

	Simplifying	Preserving
Vector	`x[[1]]`	`x[1]`
List	`x[[1]]`	`x[1]`
Factor	`x[1:4, drop = T]`	`x[1:4]`
Matrix	`x[,1] or x[1,]`	`x[, 1, drop = F] or x[1, ,drop = F]`
Data frame	`x[,1] or x[[1]]`	`x[, 1, drop = F] or x[1]`

Useful Helper Functions

- `%in%`
- `unique(), duplicated()`
- `which(), which.min(), which.max()`
- `grep(), grepl()`

Matching/joining

- `match(value, target, …)`

- Example: (left join) Find the semantic group for each term in "RA_concepts".

The subset() function

- `subset(airquality, Temp > 80, select = c(Ozone, Temp))`

- `subset(airquality, Day == 1, select = -Temp)`

- `subset(airquality, select = Ozone:Wind)`

Random sampling

- Without replacement: `dat[sample(1:nrow(dat), 100), ]`

- With replacement: `dat[sample(1:nrow(dat), replace=T), ]`

- `set.seed()`