Intro to R Week 2 Lab 1

R Studio Interface

The R interface has 4 panels which include:
- scripts (markdown documents, notebooks)
- console (command line)
- enviornment (lists of created/availabe objects)
- files/plots/viewer

Packages

You use packages to run your code. For any package, you install just once, then use `library` to load the package when needed.
For example:

`install.packages("packagename")` You only need to do this once. You can also install packages by using the interface:
- Toos > Install Packages

To envoke a package (which you need to do everytime you start working in R)
`library(packagename)` when you start up RStudio.

Working Directory

- where you keep your files, data, etc
- always know where you are!
- make a seperate workign directory for each project

Working Directory Basics

- get working directory
- set working directory
- or use the File panel in R studio

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

Computational Thinking

Algorithms and Data Structures

- formal steps to solve a problem

Data Structures

- how to store information in a computer
- everything is an object

Types of Data Structures

- data types
  `character`: for text e.g., "statistics"
  `numeric`: e.g., 219.2
  `integer`: e.g., 44
  `logical`: TRUE , FALSE
  `factor`: for categories

What is the Data Types

- what is an object?
  `is.character`
  `is.numeric`
  `is.logical`

```r
is.character("2")
```

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

```r
is.character(2)
```

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

```r
is.numeric(2)
```

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

```r
is.numeric("2")
```

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

```r
is.logical(1)
```

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

```r
is.logical(TRUE)
```

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

Converting the Data type

- convert one type into another

```r
as.numeric("2")
```

```
## [1] 2
```

```r
as.character(2)
```

```
## [1] "2"
```

Assigning Variables

- everything in R is an object
- create objects by assigning values to them
- assign the value of 5 to x
- lets see what it is assigns the value to

```r
x<-5
x
```

```
## [1] 5
```

```r
mode(x)
```

```
## [1] "numeric"
```

```r
is.character(x)
```

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

```r
is.integer(x)
```

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

```r
is.numeric(x)
```

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

Mathematical Operations

- – ∗ /
  %% modulus or remainder
  %/% integer division
  ˆ power

```r
5+5
```

```
## [1] 10
```

```r
10-5
```

```
## [1] 5
```

```r
5*5
```

```
## [1] 25
```

```r
25/5
```

```
## [1] 5
```

```r
7%%2
```

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

```r
11 %/% 2
```

```
## [1] 5
```

```r
3^2
```

```
## [1] 9
```

```r
a<-2
b<-3
a+b
```

```
## [1] 5
```

```r
a*b
```

```
## [1] 6
```

```r
b^a
```

```
## [1] 9
```

```r
c<-a+b
c
```

```
## [1] 5
```

```
d<-a*b
d
```

```
## [1] 6
```

```
e<-c+d
e
```

```
## [1] 11
```

```
is.numeric(e)
```

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

```
d>c
```

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

```
f<-c==d
f
```

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

```
is.logical(f)
```

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

```
g<-as.numeric(f)
g
```

```
## [1] 0
```

Operators

==
all.equal()
>
<
>=
<=

```
5>4
```

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

```
5<4
```

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

```r
5<=5
```

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

```r
5>=5
```

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

Program Flow

1) get data and assign to data structures

2) move through steps in the algorithm
   logical branches: if statments
   repeated operations: for loops

3) combine operations into functions

Common Mistakes in R

common mistakes in R
- using wrong case
- forgetting “”
- forgetting ()
- using function from a package that is not loaded
- typos

Basic Data Structures

Vector

- collection of values of the same type

- concatentate

- c(elements)

```r
x <- c(3,7,4)
x
```

```
## [1] 3 7 4
```

```r
length (x)
```

```
## [1] 3
```

```r
z <- c("a","b","c")
z
```

```
## [1] "a" "b" "c"
```

```r
length(z)
```

```
## [1] 3
```

- sequences
- seq(n1,n2,increment)
- n1:n2

```r
y <- seq(1,10)
y
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
y2 <- seq(1,10,2)
y2
```

```
## [1] 1 3 5 7 9
```

Extracting Vector Elements

- specific element
  `[index]`

```r
x<- seq(1,10)
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
x[4]
```

```
## [1] 4
```

```r
x[1:5]
```

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

Matrices

- regular, two dimension tables
- convert a vector to matrix (x, nrow=n, ncol=m)
- dimension of matrics
  `dim`

```r
v1<-1:16
v1
```

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

```r
v2 <- matrix(v1, nrow=4,ncol=4)
v2
```

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

```r
v3 <- matrix(v1, nrow=4,ncol=4,byrow=TRUE)
v3
```

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

```r
dim(v3)
```

```
## [1] 4 4
```

Extracting Matrix Elements

- finding a point in the matrix [rowselection, columnselection]

```r
v3
```

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

```r
v3[2,2]
```

```
## [1] 6
```

```r
v3[1:2,3:4]
```

```
##      [,1] [,2]
## [1,]    3    4
## [2,]    7    8
```

```r
v3[,2]
```

```
## [1]  2  6 10 14
```

```r
v3[,3:4]
```

```
##      [,1] [,2]
## [1,]    3    4
## [2,]    7    8
## [3,]   11   12
## [4,]   15   16
```

```r
v3[c(1,3),]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    9   10   11   12
```

Lists

- collection of elements of different types

```r
z <- list (name="point", x=3.5, y=2)
z
```

```
## $name
## [1] "point"
##
## $x
## [1] 3.5
##
## $y
## [1] 2
```

Data Frames

- columns= variables
- rows= observations

```r
v1 <- 1:16
m1 <- matrix(v1, nrow=4,ncol=4)
m1
```

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

```
t1 <-data.frame(m1)
t1
```

```
##   X1 X2 X3 X4
## 1  1  5  9 13
## 2  2  6 10 14
## 3  3  7 11 15
## 4  4  8 12 16
```

```
t2 <-as.data.frame(m1)
t2
```

```
##   V1 V2 V3 V4
## 1  1  5  9 13
## 2  2  6 10 14
## 3  3  7 11 15
## 4  4  8 12 16
```

what is the difference between these two??

```
names(t1)
```

```
## [1] "X1" "X2" "X3" "X4"
```

```
row.names(t1)
```

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

```
names(t2)
```

```
## [1] "V1" "V2" "V3" "V4"
```

```
row.names(t2)
```

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

Lab 2 Week 3

Reading a Data Frame from a Text File

`read.table`
`read.csv`
### things to watch out for
- make sure file is in working director
- header (sets the variable names)
- set stringsAsFactors=FALSE

Reading a data file

data <- read.csv(“NSYL.csv”)

some useful commands

view the first 6 rows
head (data)
structure, gives types of columns
str
dimension, number of rows and columns
dim
statistical summary of variables: min, 1st quartile, median, mean, 3rd quartils, max
summary

Extracting variables

sometimes you will want to extract a vector to look at just a subset of data

useful commands

extracting variables columns

data_frame$variable_name
[[“variable,name”]] [,column_number]

extracting a data frame

[column_number]

Writing a Data Frame to a file

write.csf(df,filename)
set row.names=FALSE
will be written to workign directory