Introduction to R

Anindya Bhattacharya, PhD | abhatta3@ucla.edu
CBI fellow | Xia Yang lab

Anindya Bhattacharya, PhD

abhatta3@ucla.edu

Office hours: 2000 TLSB

Xia Yang lab

Department of Integrative Biology and Physiology

Center for Duchenne Muscular Dystrophy

Areas of interest: miRNA, genetic variation, gene expression data analyses, machine learning,

Collaboratory Website

http://collaboratory.lifesci.ucla.edu/

Workshop 3: Introduction to R

- **Day 1**
  - Introduction to R
    - Installation, configuration, basic concepts, syntax and usage
- **Day 2**
  - Data Analysis
    - Visualization methods
    - Statistical methods
- **Day 3**
  - Scripting, batch mode
  - Packages, Bioconductor

Workshop 3: Introduction to R

- **Day 1**
  - What is R? Why use it?
  - Getting up and running
    - Installation, configuration issues
    - overview of R interface
    - GUIs for working with R
  - Your first R session
  - Basic concepts
    - Starting and stopping an R session
    - finding help/documentation
    - Data structures
    - Getting data in and out of R
    - Basic analysis

Dropbox

- In the drop box folder are:
  - course powerpoint
  - sample scripts
  - exercises
- Please DO NOT make changes to these files because they will affect EVERYONE!
- INSTEAD, copy them to a directory on your computer
- Let me know if you need to be invited to the dropbox folder

What is R?

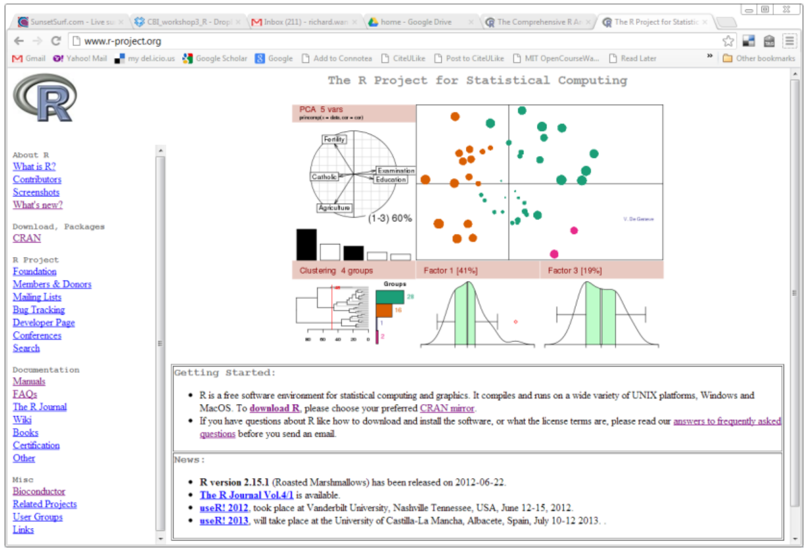
- A glorified calculator
- Statistical tables
- Visualization tool
- Programming language and environment

Why R?

- Many tools STATA, SPSS, Matlab, Excel…
- Advantages
  - powerful
  - up to date with latest algorithms (packages, Bioconductor)
  - strong community of users
  - Preferred by statistics community
  - it's free
- Disadvantages
  - steep learning curve, but can be useful quickly
  - not pretty
  - can be memory intensive

Where to download R?

- Use http://r-project.org or a CRAN mirror (http://cran.stat.ucla.edu)



www.r-project.org

The R Project for Statistical Computing

About R
What is R?
Contributors
Screenshots
What's new?

Download, Packages
CRAN

R Project
Foundation
Members & Donors
Mailing Lists
Bug Tracking
Developer Page
Conferences
Search

Documentation
Manuals
FAQs
The R Journal
Wiki
Books
Certification
Other

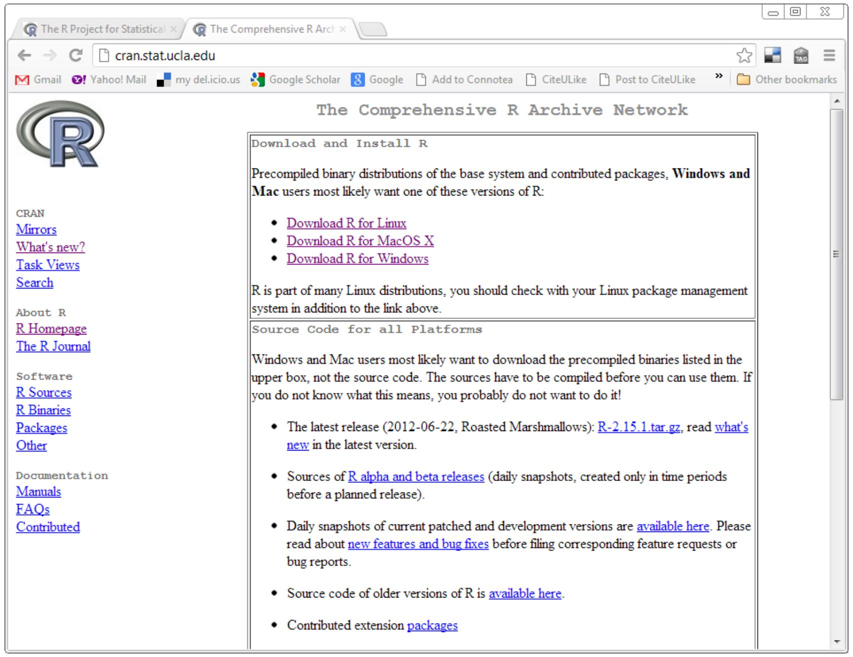
Misc
Bioconductor
Related Projects
User Groups
Links

Getting Started:

- R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.
- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

News:

- **R version 2.15.1** (Roasted Marshmallows) has been released on 2012-06-22.
- **The R Journal Vol.4/1** is available.
- **useR! 2012**, took place at Vanderbilt University, Nashville Tennessee, USA, June 12-15, 2012.
- **useR! 2013**, will take place at the University of Castilla-La Mancha, Albacete, Spain, July 10-12 2013. .



cran.stat.ucla.edu

The Comprehensive R Archive Network

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for MacOS X
- Download R for Windows

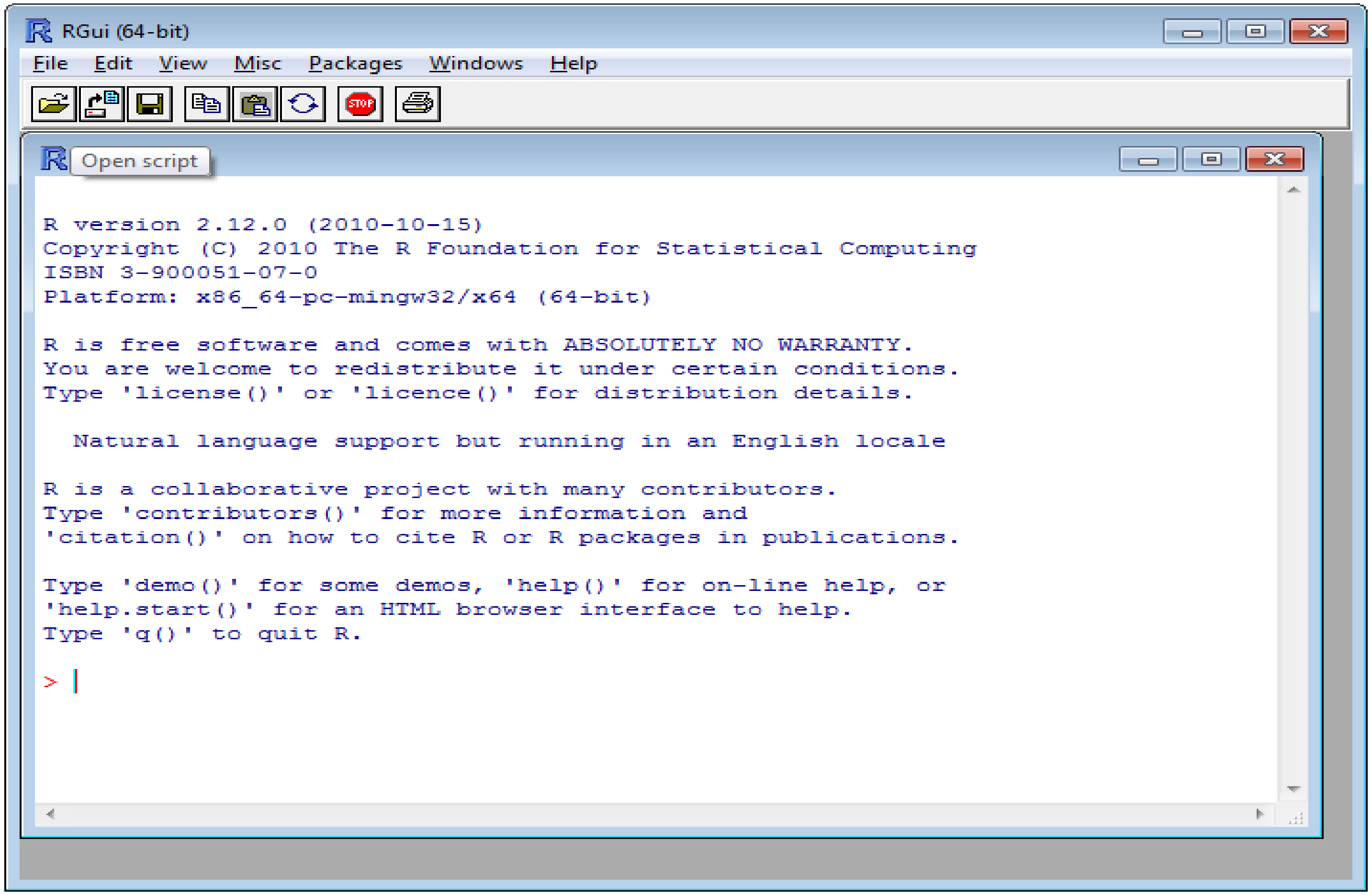
R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2012-06-22, Roasted Marshmallows): R-2.15.1.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

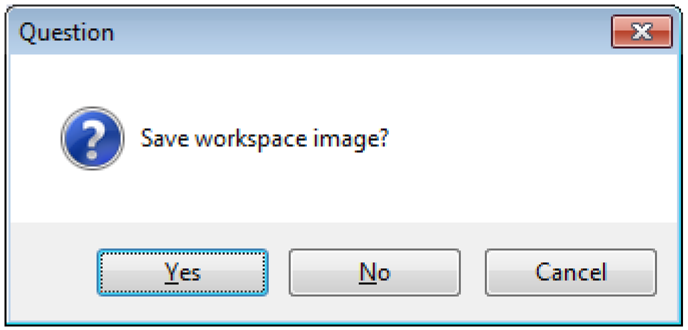
Starting Up…



```
R version 2.12.0 (2010-10-15)
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-pc-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

Stopping

- To end your session
  - Close window or q()
- What does this mean?



Question

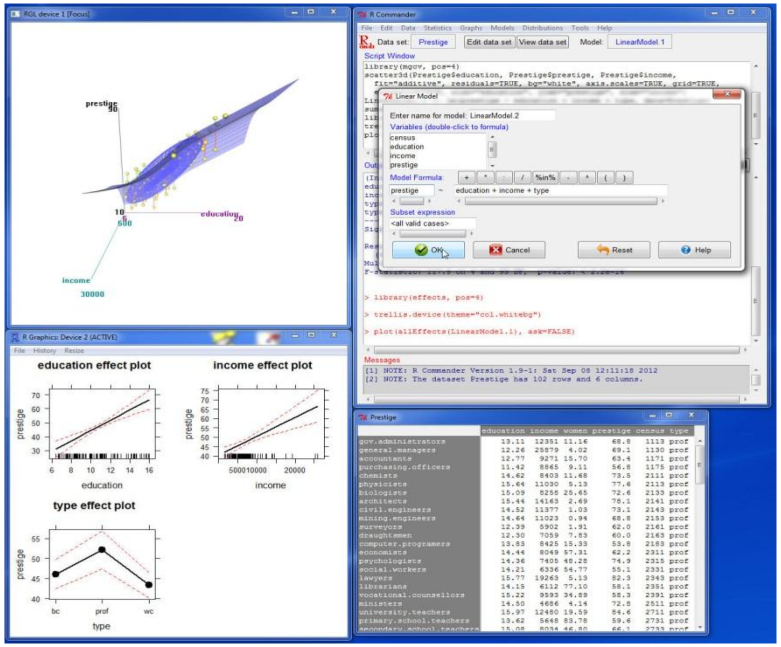
Save workspace image?

  - It saves everything you created (variables, etc) into a (hidden) file in the current directory usually named .Rdata

GUIs and IDEs

- R interface is pretty sparse
- Some alternatives:

	Win	Mac	Linux
RStudio Desktop	x	x	x
RStudio Server			x
Rcommander	x	x	x
Tinn-R	x		
JGR	x	x	x
RKWard	x		x
Emacs/ESS	x	x	x
Vi	x	x	x
notepad	x	x	x

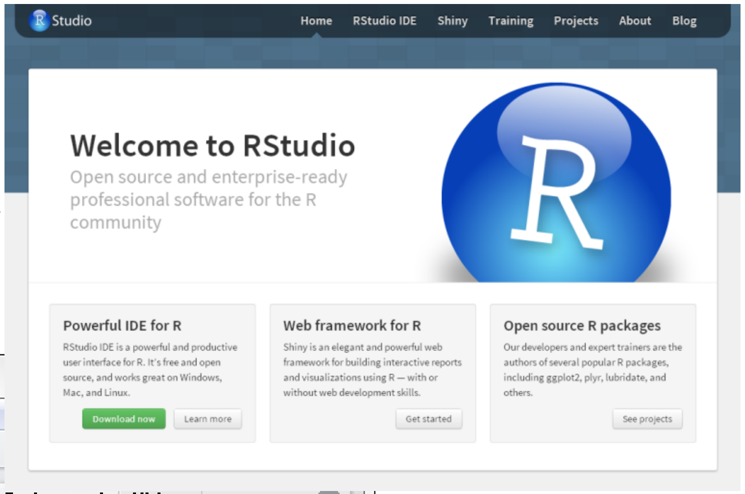


```
library(mgcv, pos=4)
scatter3d(Prestige$education, Prestige$prestige, Prestige$income,
  fit="additive", residuals=TRUE, bg="white", axis.scales=TRUE, grid=TRUE,
```

```
> library(effects, pos=4)

> trellis.device(theme="col.whitebg")

> plot(allEffects(LinearModel.1), ask=FALSE)
```

```
[1] NOTE: R Commander Version 1.9-1: Sat Sep 08 12:11:18 2012
[2] NOTE: The dataset Prestige has 102 rows and 6 columns.
```

```
                           education income women prestige census type
gov.administrators             13.11  12351 11.16     68.8   1113 prof
general.managers               12.26  25879  4.02     69.1   1130 prof
accountants                    12.77   9271 15.70     63.4   1171 prof
purchasing.officers            11.42   8865  9.11     56.8   1175 prof
chemists                       14.62   8403 11.68     73.5   2111 prof
physicists                     15.64  11030  5.13     77.6   2113 prof
biologists                     15.09   8258 25.65     72.6   2133 prof
architects                     15.44  14163  2.69     78.1   2141 prof
civil.engineers                14.52  11377  1.03     73.1   2143 prof
mining.engineers               14.64  11023  0.94     68.8   2153 prof
surveyors                      12.39   5902  1.91     62.0   2161 prof
draughtsmen                    12.30   7059  7.83     60.0   2163 prof
computer.programers            13.83   8425 15.33     53.8   2183 prof
economists                     14.44   8049 57.31     62.2   2311 prof
psychologists                  14.36   7405 48.28     74.9   2315 prof
social.workers                 14.21   6336 54.77     55.1   2331 prof
lawyers                        15.77  19263  5.13     82.3   2343 prof
librarians                     14.15   6112 77.10     58.1   2351 prof
vocational.counsellors         15.22   9593 34.89     58.3   2391 prof
ministers                      14.50   4686  4.14     72.8   2511 prof
university.teachers            15.97  12480 19.59     84.6   2711 prof
primary.school.teachers        13.62   5648 83.78     59.6   2731 prof
secondary.school.teachers      15.08   8034 46.80     66.1   2733 prof
```



Welcome to RStudio

Open source and enterprise-ready professional software for the R community

Powerful IDE for R

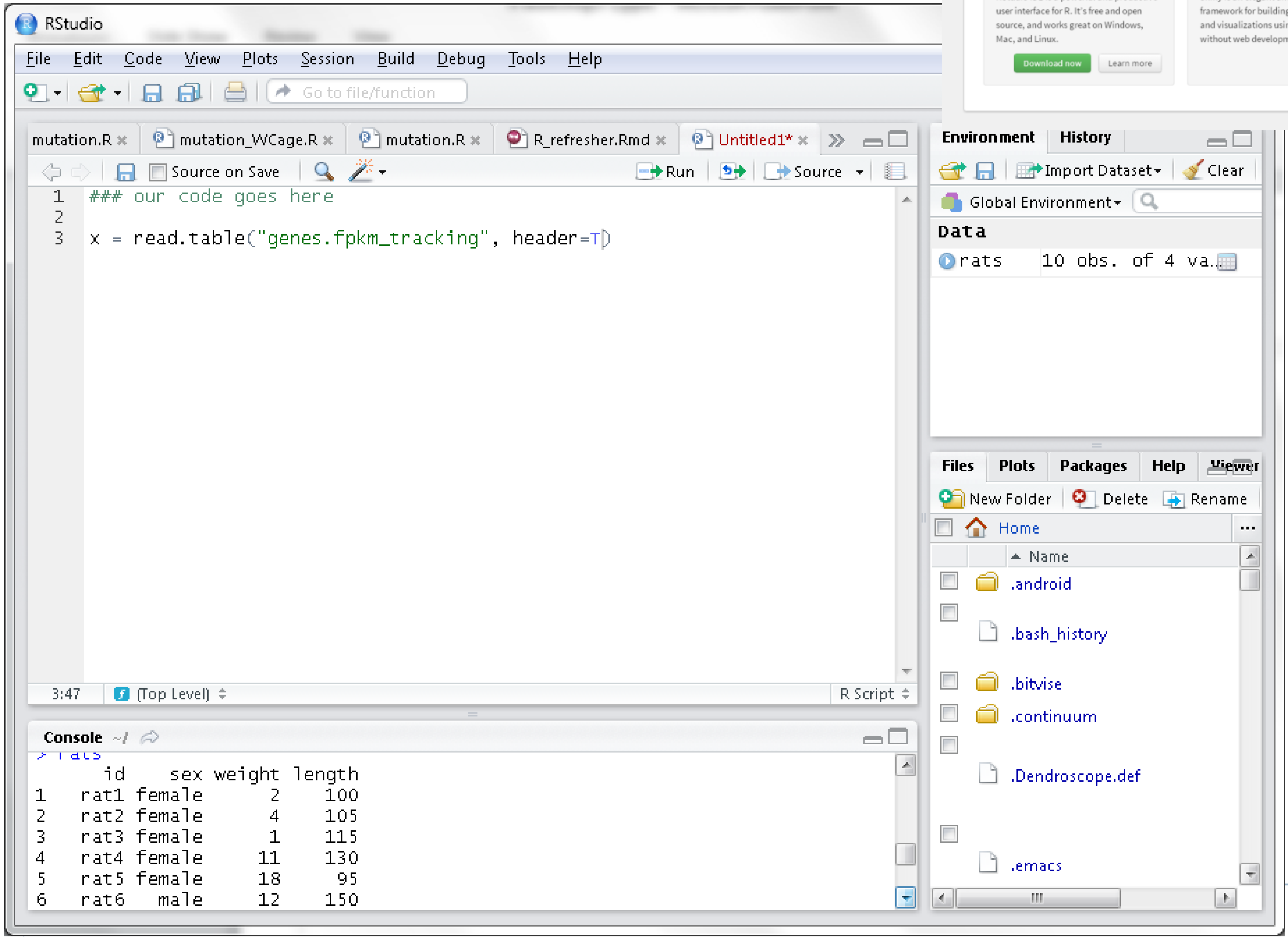
RStudio IDE is a powerful and productive user interface for R. It's free and open source, and works great on Windows, Mac, and Linux.

Web framework for R

Shiny is an elegant and powerful web framework for building interactive reports and visualizations using R — with or without web development skills.

Open source R packages

Our developers and expert trainers are the authors of several popular R packages, including ggplot2, plyr, lubridate, and others.



```
### our code goes here

x = read.table("genes.fpkm_tracking", header=T)
```

Data

```
rats    10 obs. of 4 va...
```

```
> rats
    id    sex weight length
1 rat1 female      2    100
2 rat2 female      4    105
3 rat3 female      1    115
4 rat4 female     11    130
5 rat5 female     18     95
6 rat6   male     12    150
```

.android
.bash_history
.bitvise
.continuum
.Dendroscope.def
.emacs

It’s like unix

- Many of the commands are similar to command line unix tools (cat, ls, grep, …)
- but in R, everything is a function so **()** is important!
  - cat()
  - ls()
  - rm()

- What happens if you forget the parentheses?
  - try it!

R packages

- List installed packages

```
> installed.packages()
> a<- installed.packages()
```

- Available packages

```
> p<-available.packages()
> dim(p)
```

- Is the package “boot” already installed?

```
> is.element(“boot”, a[,1])
>[1] TRUE
```

Finding help

```
> help("data.frame")
> ?data.frame
> help.search("chisquare")
> apropos("mean")
 [1] "colMeans"        "kmeans"          "mean"             "mean.data.frame"
 [5] "mean.Date"       "mean.default"    "mean.difftime"     "mean.POSIXct"
 [9] "mean.POSIXlt"    "rowMeans"        "weighted.mean"

> library(help=utils)
> help(BATCH)
```

Advice for learning R

- There are no easy ways to learn R except by
    1. Typing examples in yourself! It makes a difference (muscle memory, syntax)
    2. Persisting!
    3. Iterating!
- Steep learning curve, but with a few commands it can be useful quickly
- Try things out – you will not break your data!

Getting around

- R starts in a particular directory, usually the one where R is located

```
# print the current working directory
getwd()
# set the current working directory
setwd("/home/…/project1")
# show the files in your current directory
dir()
```

- Create a directory for your project
  - good way to organize

```
>dir.create("R_CODE")
```

A few things

- comments start with a hash tag (#) and are ignored by R. Great way to make notes!
- The prompt ">" is where you type commands, statements

```
> a = 1
```

- "+" is a continuation character; if your statement is not complete, R prompts you to finish it:

```
> a =
+ 3
> a
[1] 3
```

- Typing the variable name gives you its contents

```
> a
[1] 3
```

- Case matters! **A** is not the same as **a**

```
> A
Error: object 'A' not found
```

- Typing a function name without parens () gives the source code

Try it out

```
# genes.fpkm_tracking output of cufflinks
# it is tab delimited with a header line
# what happens if we say header=F?
x = read.table("genes.fpkm_tracking", header=T)
ls()
# what do the numbers in dim() mean?
dim(x)
names(x)
head(x)
class(x)
sum(x[,11] > 100)
sum(x$q1_FPKM > 100)
# try changing the axis limits "ylim" and printing character "pch"
plot(x[,11], x[,14], xlim=c(0, 10), ylim=c(0,20), xlab="exp", main="my
experiment", pch=2)
# try chaning the number of breaks
hist(x[,11], col="red", breaks=100)
cor(x[,11], x[,14])
newx = x[ x$status=="OK", ]
ls()
```

Basic data structures in R

	Homogeneous	Heterogeneous
1d	Vector	List
2d	Matrix	Data frame
nd	Array	

Note: R has no 0-dimensional, or scalar types. Individual numbers or strings, which you might think would be scalars, are actually vectors of length one.

Indexing

- let's create a vector

```
a = c(1,2,3,4)
a[2] # gives 2, indexing in R start with 1 not 0!
a[1:3]
```

- create a matrix

```
m = matrix(c(1,2,3,4,5,6),  nrow=2, ncol=3)
m[,1]
[1] 1 2
m[,2] = 30
```

- logical indexing

```
x = 1:10
x > 5
[1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
x[ x > 5]
[1]  6  7  8  9 10
```

Vectors

- Basically arrays

```
> v = 100
> v <- 100
> c(1,2,3)
> 1:3
> rep(1:2, 3)
> vector(mode="numeric", length=5)
```

- Vector Arithmetic

```
> 1 + 1:3
[1] 2 3 4
> 1:2 + 1:3 #what happens?
```

Vector Arithmetic

- Vector Arithmetic

```
> 1:2 + 1:3 #what happens? - ERROR!
> 1:2 + 1:10
[1]  2  4  4  6  6  8  8 10 10 12
```

- Vector indexing (1-based)

```
> b = 1:2 + 1:10
> b[1]
```

- Vector length

```
> length(b)
```

Name

- You can name a vector in three ways:

```
> x <- c(a = 1, b = 2, c = 3)
> x <- 1:3; names(x) <- c("a", "b", "c")
> x <- setNames(1:3, c("a", "b", "c"))
```

Factors : Categorical data

- Categorical variables
- not the same as character class

```
> col = c("red", "green", "red", "yellow", "red")
> class(col)
[1] "character"
> col.f = factor(col)
[1]  red       green    red       yellow  red
Levels:  green  red  yellow
> levels(col.f) = c("green", "red", "blue")
> col.f
[1] red   green red   blue  red
Levels: green red blue
> table(col.f)
col.f
green   red  blue
    1     3     1
```

Matrices

- Grid of numbers. Must be numeric!

```
>  x  <-  matrix(1:10,  nrow=2)
>  dim(x)
[1]  2  5
>  x
[,1]  [,2]  [,3]  [,4]  [,5]
[1,]       1        3        5        7        9
[2,]       2        4        6        8      10
>  as.vector(x)
[1]    1    2    3    4    5    6    7    8    9   10
> dim(x)
[1] 2 5

# hint: x[2,4] means get element from row 2, col 4
```

List

- **Lists are different from atomic vectors because their elements can be of any type, including lists. You construct lists by using list() instead of c().**

```
# define list x
>x <- list(1:3, "a", c(TRUE, FALSE, TRUE), c(2.3, 5.9))
# display structure of x
>str(x)
# vector from a list
> unlist(x, recursive = TRUE, use.names = TRUE)
```

Matrices and arrays

- Adding a dim() attribute to an atomic vector allows it to behave like a multi-dimensional array.

```
# Two scalar arguments to specify rows and columns
>a <- matrix(1:6, ncol = 3, nrow = 2)
>rownames(a) <- c("A", "B")
>colnames(a) <- c("a", "b", "c")
# One vector argument to describe all dimensions
>b <- array(1:12, c(2, 3, 2))
> dimnames(b) <- list(c("one", "two"), c("a", "b", "c"),
  c("A", "B"))
```

Data frames

- the most useful data structure
- data frames hold data like a spreadsheet
  - observations are rows
  - covariates are columns
- like matrices, they can be subsetted and indexed with two subscripts

Data frames

```
> df = data.frame(type=rep(c("case", "control"), c(2, 3)),
time=rexp(5))
>  df
      type      time
1     case 1.1745712
2     case 1.1691266
3  control 0.8227643
4  control 0.1301390
5  control 1.0581316
>  df$time
[1]  1.1745712  1.1691266  0.8227643
[4]  0.1301390  1.0581316
> df["time"]
```

Data frames

```
# let’s update row names
>  names(df)
[1]  "type"  "time"
>  rn  <-  paste("id",  1:5,  sep="")
>  rownames(df)  <-  rn
>  df[1:2,  ]
     type  time
id1  case  1.174571
id2  case  1.169127
# update column names
> names(df) = c("var1", "var2")
> df[1:2,]
    var1 var2
id1 case 0.1653743
id2 case 0.4509330
```

Data type conversions

- Many problems are due to unexpected data types
  - eg. using a factor in place of a character string
- to convert from one data type to another, try

```
as.vector()
as.character()
as.data.frame()
as.numeric()
as.matrix()
```

- to determine what data type your variable is, try

```
class(variable)
```

Functions

- you can define your own functions

```
> square = function(x) {
+                   x*x
+                      }
> square(10)
[1] 100

> square(1:4)
[1]    1    4    9   16
```

- R functions take can take named and default parameters

```
say = function(word="hi", repeats=3) {
   cat( rep(word, repeats), "\n")
}
say()
say(word="hola", repeats=1)
say(repeats=2)
```

Reading in data

- A very common operation

```
> x = read.table("myfile.txt")
```

- Several variants: read.csv(), read.delim()
- this reads a delimited file into a data.frame object
- every line of the file must have the same number of elements!
  - sometimes you can fix things eg read.table("myfile.txt", sep="\t")
- entire file read into memory, so this can be slow but optimizations possible

Reading the help

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", row.names, col.names,
           as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           fileEncoding = "", encoding = "unknown")

read.csv(file, header = TRUE, sep = ",", quote="\"", dec=".",
         fill = TRUE, comment.char="", ...)

read.csv2(file, header = TRUE, sep = ";", quote="\"", dec=",",
          fill = TRUE, comment.char="", ...)

read.delim(file, header = TRUE, sep = "\t", quote="\"", dec=".",
           fill = TRUE, comment.char="", ...)

read.delim2(file, header = TRUE, sep = "\t", quote="\"", dec=",",
            fill = TRUE, comment.char="", ...)
```

Writing data

- Writing data to a file is pretty simple
- Some odd defaults need to be commonly overridden

```
write.table(x, file="file_name.txt", quote=F, sep="\t",
row.names=F, col.names=T)
```

- Other variants write.csv()

Saving and loading session

- all objects in your workspace can be saved to a file that can be read again on the next startup
- save() or save.image()
- the file is usually called .RData in your current directory
- load() will load an .RData file
- GUI might have these options

Some useful commands

- show a recent history of commands you typed
  - history()
- customizing your R
  - .Rprofile is a file found in your home directory where you can set defaults, define some commonly used functions, etc
- rm() – remove an object from workspace
- ls() – list the objects in your workspace
- cbind(array1, array2) – combine columns
- rbind(array1, array2) – combine rows
- t() – transpose a vector, matrix or data frame
- sessionInfo() – info about your settings

Match

- match returns a vector of the positions of (first) matches of its first argument in its second.

```
> df1 = read.table("df1.txt", header=T)
> df2 = read.table("df2.txt", header=T)
> match(df2$id, df1$id)
[1]  1  4  8  9 NA NA

# alternate syntax
> df2[,1] %in% df1[,1]
[1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE
```

Merge

- Merge two data frames that have a common identifier

```
df1 = read.table("df1.txt", header=T)
df2 = read.table("df2.txt", header=T)
both = merge(df1, df2, by.x=1, by.y=1)
new = merge(df1, df2, by.x=1, by.y=1, all.x=T)
```

which

- which helps you find the index of an element

```
> a = 1:10
> which(a < 5)
[1] 1 2 3 4
> a[which(a < 5)]
[1] 1 2 3 4
> ids = which(a < 5)
[1] 1 2 3 4
> a[ids]
```

Special values

- NA (Not available)
  - sometimes data shows up as 'NA'
  - usually due to
    - missing data
    - conversion
  - **not** the same as zero (0) !
  - test with is.na(x)
- NaN (not a number)
  - 0/0
  - is.nan()
- Inf (infinite) or -Inf
  - 10/0
  - test with is.finite(x) or is.infinite(x)

NA/NAN/INF

```
#Test if any NaN
  sum(is.nan(df1[,1]))
#Test if any INF
  sum(is.finite(df1$id))
#Test if any NA's
  sum(is.na(df2))
#Find the col/row with the NA
res = is.na(df2)
> colSums(res)
    id     age weight
     0       0      1
> rowSums(res)
[1] 0 0 0 1 0 0
```

Logical operators

- Useful for chaining operations
  - | (OR), & (AND), ! (NOT)
    - for instance, you want values greater than 10 or less than 5

```
df[ df[,1] > 10 | df[,1] < 5]
df[ !is.na(df[,1])]
df[ df[,2] == "gene" & df[,1] > 10]
```

- == is the equal operator (not =, which assigns)

Flow control

- If statements for branching

```
if (x == 3){
     doSomething()
  } else {
     doSomethingElse()
  }
```

- loops
  - slow in R

```
  a = seq(1,10)
  for(i in 1:length(a)){
  cat(a[i], "\n")
  }
```

Apply

- R operations are usually vectorized, so loops are avoided.
- Loops are quite slow in R

```
mat = matrix(seq(1,10), 2)
# how to get rowwise or columnwise mean?
# could use a loop, but you have to specify dimension
# margin 1 = row, 2 = cols
apply(mat, 1, mean)
apply(mat, 2, mean)
```

Sorting

- Sorting returns an array in sorted order

```
> b = c(1,9,8,2,4,6)
> sort(b, decreasing=F)
```

- Ordering returns a vector of array indicies that puts the vector in a particular order
  - more useful for sorting data frames

```
> b = c(1,9,8,2,4,6)
b.order = order(b, decreasing=F)
> b.order
[1] 1 4 5 6 3 2
> b[b.order]
[1] 1 2 4 6 8 9
```

Reading in a script

- Suppose you've written a R script that processes your data
- source() will read it in as if you typed it and execute all commands listed in the file
- source("R_script_file.R")

Additional resources

- Many resources but sometimes difficult to access
  - R-bloggers.com
  - R-project.org – FAQ and manuals
  - bioconductor.org
  - UCLA ATS (http://www.ats.ucla.edu/stat/r/)
  - Quick R http://www.statmethods.net/index.html
  - http://onertipaday.blogspot.com/
  - http://wiki.stdout.org/rcookbook/
  - http://gettinggeneticsdone.blogspot.com/
  - manuals.bioinformatics.ucr.edu/workshops
- Help sites
  - stackoverflow.com
  - nabble.com
  - google
- Good free books
  - An Introduction to R (on r-project.org)
  - Using R

Additional resources

- Courses
  - Coursera
    - https://class.coursera.org/rprog-002
  - EdX.org
    - https://www.edx.org/course/harvardx/harvardx-ph525x-data-analysis-genomics-1401#.U0xFnvldW_g
  - Stanford Class
    - https://class.stanford.edu/courses/HumanitiesScience/StatLearning/Winter2014/about
  - Google YouTube
    - https://www.youtube.com/watch?v=iffR3fWv4xw&list=PLOU2XLYxmsIK9qQfztXeybpHvru-TrqAP
  - Bioconductor
    - http://www.bioconductor.org/help/events/

Shortcuts

- To interrupt
  - MAC: <CMD> <Option> <.> or <ESC>
  - PC/Linux: Ctrl-C

- sessionInfo()
  - this displays details about your environment, what packages are loaded, etc… useful for debugging

Workshop 3: Introduction to R

- **Day 2**
  - Statistical methods
    - Probability distributions
    - Random number generation
    - Common tests
      - Fisher's exact test, chi square, qvalue, etc…
  - Visualization
    - Plotting functions
    - Customizing plots
    - Saving plots for publication

Workshop 3: Introduction to R

- **Day 3**
  - Automating or scripting R
    - Batch mode
  - Packages that extend R
    - Installation
    - Documentation
    - Useful packages
  - Bioconductor
    - Extending R for the life sciences
    - Useful bioconductor packages
    - Some basic bioC concepts
    - Bioconductor demo