R Data Structures

R for Beginners

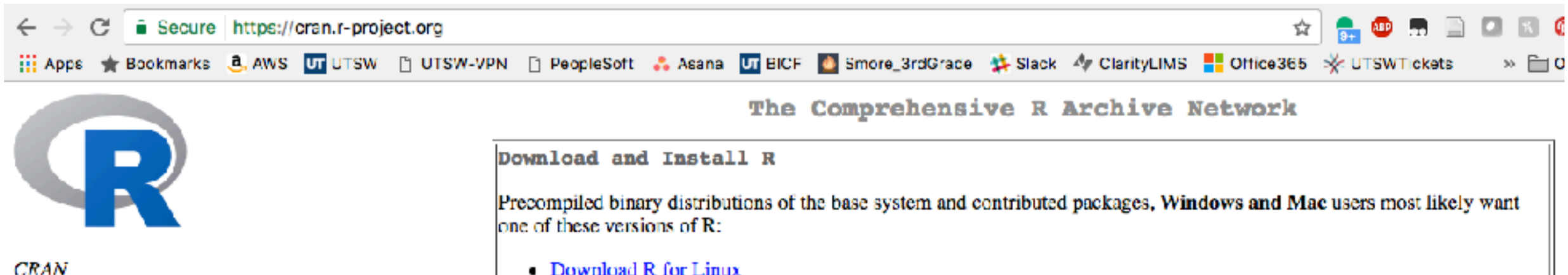
What is R?

- R is a free software environment for statistical computing and graphics
- Object oriented statistical language
- 2000: R version 1.0.0 was released.
- Quickly became popular for bioinformatics, microarray analysis
- New version released every 6 months
- Now -versions for Windows (32 and 64bit), UNIX/Linux, MacOS, and RStudio (GUI version)

What is R?

- Suite of operators for calculations on arrays and matrices
- Sophisticated graphical facilities for display or output files
- Active R community - R-help and R-devel mailing lists
- ~25 base, or standard, packages
- Thousands of contributed packages in repositories:
  - CRAN: http://CRAN.R-project.org
  - Bioconductor: www.bioconductor.org
  - Many more packages available on personal websites

Downloading R



Tools for Biologist

- Bioconductor — www.bioconductor.org
- A group of R packages aimed at high-throughput genomic data analysis and genomic annotations
- Open source and open development
- Each Bioconductor package usually has a “vignette” for documentation ie a tutorial for common usage
- Easy to download Bioconductor packages within R:
  - source(“http://www.bioconductor.org/biocLite.R")
  - biocLite()
  - biocLite(“package.name”)

bioconductor.org

Bookmarks AWS UTSW UTSW-VPN PeopleSoft Asana BICF Smore_3rdGrade Slack ClarityLIMS Office365 UTSWTickets

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Search:

Home Install Help Developers About

BioC 2017!

Please join us in Boston, July 26 (developer day), 27, and 28 for our annual conference. More information
Registration FULL.

About *Bioconductor*

Bioconductor provides tools for the analysis and comprehension of high-throughput genomic data. Bioconductor uses the R statistical programming language, and is open source and open development. It has two releases each year, 1383 software packages, and an active user community. Bioconductor is also available as an AMI (Amazon Machine Image) and a series of Docker images.

News

- Bioconductor 3.5 is available.
- Bioconductor F1000 Research Channel available.
- Orchestrating high-throughput genomic analysis with *Bioconductor* (abstract) and other recent literature.
- View recent course material.
- Use the support site to get help installing, learning and using Bioconductor.

Install »

Get started with *Bioconductor*

- Install *Bioconductor*
- Explore packages
- Get support
- Latest newsletter
- Follow us on twitter
- Install R

Learn »

Master *Bioconductor* tools

- Courses
- Support site
- Package vignettes
- Literature citations
- Common work flows
- FAQ
- Community resources
- Videos

Use »

Create bioinformatic solutions with *Bioconductor*

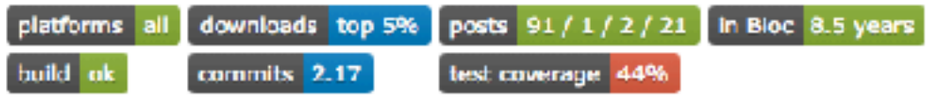
- Software, Annotation, and Experiment packages
- Amazon Machine Image
- Latest release annoucement
- Support site

Develop »

Contribute to *Bioconductor*

- Developer resources
- Use Bioc 'devel'
- 'Devel' Software, Annotation and Experiment packages
- Package guidelines
- New package submission
- Build reports

edgeR



Empirical Analysis of Digital Gene Expression Data in R

Bioconductor version: Release (3.5)

Differential expression analysis of RNA-seq expression profiles with biological replication. Implements a range of statistical methodology based on the negative binomial distributions, including empirical Bayes estimation, exact tests, generalized linear models and quasi-likelihood tests. As well as RNA-seq, it be applied to differential signal analysis of other types of genomic data that produce counts, including ChIP-seq, SAGE and CAGE.

Author: Yunshun Chen <yuchen at wehi.edu.au>, Aaron Lun <alun at wehi.edu.au>, Davis McCarthy <dmccarthy at wehi.edu.au>, Xiaobei Zhou <xiaobei.zhou at uzh.ch>, Mark Robinson <mark.robinson at imls.uzh.ch>, Gordon Smyth <smyth at wehi.edu.au>

Maintainer: Yunshun Chen <yuchen at wehi.edu.au>, Aaron Lun <alun at wehi.edu.au>, Mark Robinson <mark.robinson at imls.uzh.ch>, Davis McCarthy <dmccarthy at wehi.edu.au>, Gordon Smyth <smyth at wehi.edu.au>

Citation (from within R, enter `citation("edgeR")`):

Robinson MD, McCarthy DJ and Smyth GK (2010). "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data." *Bioinformatics*, **26**, pp. -1.

McCarthy, J. D, Chen, Yunshun, Smyth and K. G (2012). "Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation." *Nucleic Acids Research*, **40**(10), pp. -9.

Installation

To install this package, start R and enter:

```
## try http:// if https:// URLs are not supported
source("https://bioconductor.org/biocLite.R")
biocLite("edgeR")
```

Documentation »

Bioconductor

- Package vignettes and manuals.
- Workflows for learning and use.
- Course and conference material.
- Videos.
- Community resources and tutorials.

R / CRAN packages and documentation

Support »

Please read the posting guide. Post questions about Bioconductor to one of the following locations:

- Support site - for questions about Bioconductor packages
- Bioc-devel mailing list - for package developers

Documentation

To view documentation for the version of this package installed in your system, start R and enter:

```
browseVignettes("edgeR")
```

PDF		edgeR Vignette
PDF		edgeRUsersGuide.pdf
PDF		Reference Manual
Text		NEWS

Details

biocViews	AlternativeSplicing, BatchEffect, Bayesian, ChIPSeq, Clustering, Coverage, DifferentialExpression, DifferentialSplicing, GeneExpression, GeneSetEnrichment, Genetics, MultipleComparison, Normalization, QualityControl, RNASeq, Regression, SAGE, Sequencing, Software, TimeCourse, Transcription
Version	3.18.1
In Bioconductor since	BioC 2.3 (R-2.8) (8.5 years)
License	GPL (>=2)
Depends	R (>= 2.15.0), limma
Imports	graphics, stats, utils, methods, locfit
LinkingTo	
Suggests	MASS, statmod, splines, KernSmooth
SystemRequirements	
Enhances	
URL	http://bioinf.wehi.edu.au/edgeR
Depends On Me	DBChIP, EDDA, IntEREst, manta, methylMnM, MLSeq, RnaSeqGeneEdgeRQL, RnaSeqSampleSizeData, RUVSeq, samExploreR, TCC, tRanslatome
Imports Me	affycoretools, ampliQueso, ArrayExpressHTS, ASpli, baySeq, compcodeR, coseq, csaw, debrowser, DEFormats, DEGreport, DEsubs, DiffBind, diffHic, diffloop, DRIMSeq, easyRNASeq, EBSEA, EDDA, eegc, EGSEA, EnrichmentBrowser, erccdashboard, Glimma, HTSFilter, MEDIPS, metaseqR, MIGSA, msgbsR, msmsTests, PathoStat, PROPER, PureCN, regsplice, Repitools, ReportingTools, rnaSeqMap, RnaSeqSampleSize, scater, scde, scone, scran, splatter, STATegRa, SVAPLSseq, systemPipeR, TCGAbiolinks, TCseq, ToPASeq, tweeDEseq, yarn
Suggests Me	ABSSeq, biobroom, BitSeq, ClassifyR, clonotypeR, cqn, cydar, EDASeq, gage, gCrisprTools, GenomicAlignments, GenomicRanges, goseq, groHMM, GSAR, GSVA, ideal, JctSeqData, leeBamViews, missMethyl, oneChannelGUI, regionReport, SSPA, subSeq, tximport, variancePartition

Build Report

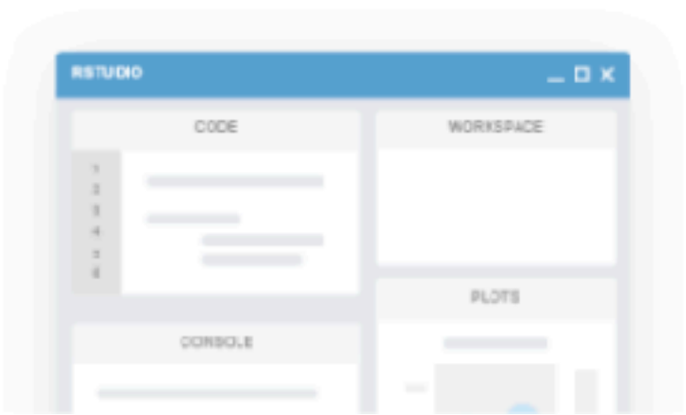
Manuals and Tutorials

- Under “Manuals” on R website – several in depth tutorials; some basic, some advanced
- Basic introductions to several specific topics in R
  - http://www.cyclismo.org/tutorial/R/
- Various forums available which discuss ranges of errors that users encounter – When in doubt, Just Google and get the syntax!
- Many R books available:
  - General purpose R: e.g., R Cookbook (2011)
  - R in a Nutshell (2010)
  - Specific topics: e.g., Introductory statistics in R
  - Applied Statistical Genetics with R
  - The art of R programming (software design)
  - R Graphics Cookbook
  - Data Mining with R: Learning with Case Studies

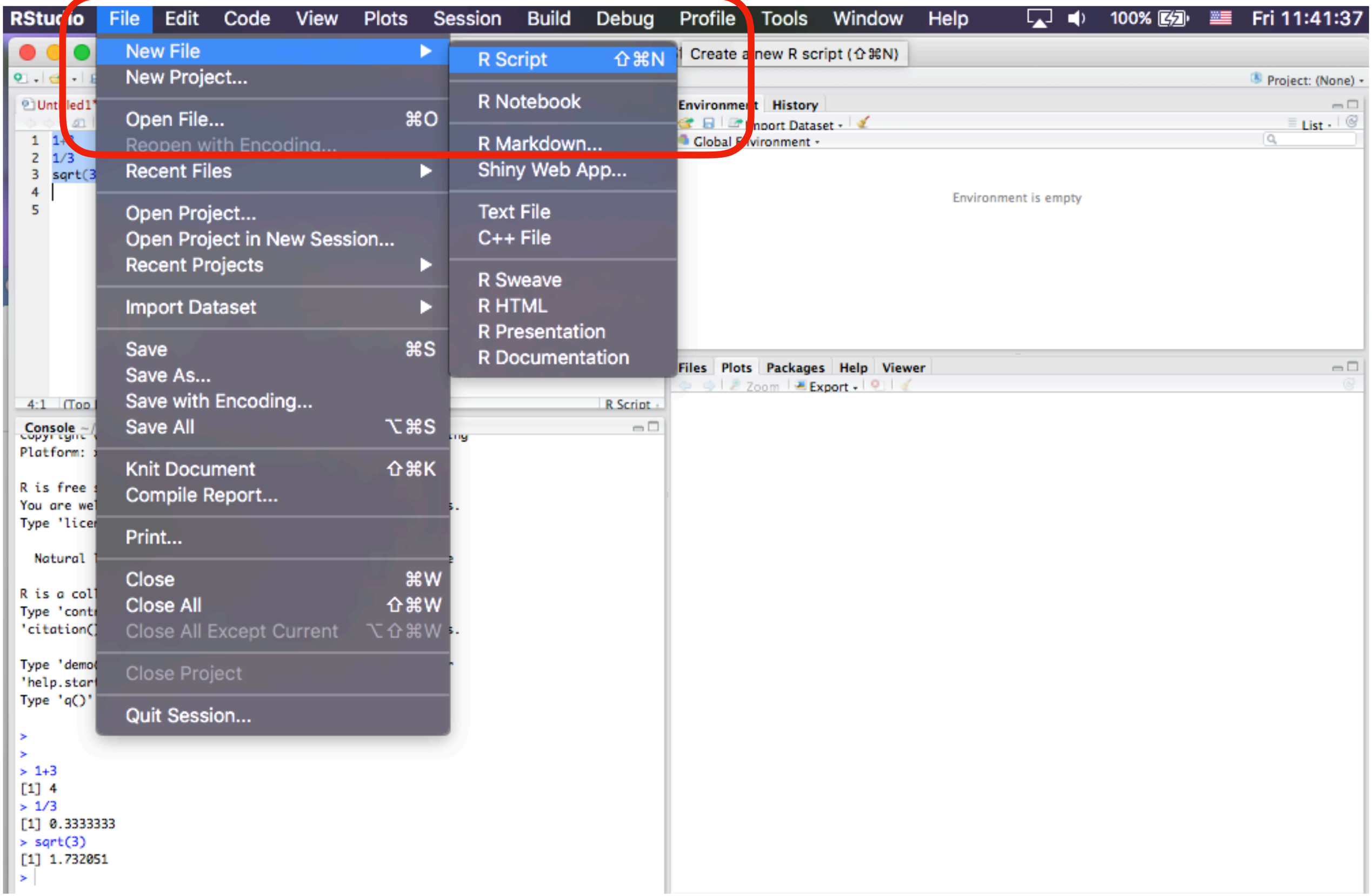
Working in R

- Can work interactively (line by line)
- In Batch mode (run a whole file with code at once)
- Linux Command Line: Rscript filename.r
- In linux, to run interactively type R in terminal window In Windows, Open the R program with interface

Graphical Interface to Command Line R



https://www.rstudio.com/


RStudio
File
Edit
Code
View
Plots
Session
Build
Debug
Profile
Tools
Window
Help
100%
Fri 11:41:37
New File
New Project...
Open File... ⌘O
Reopen with Encoding...
Recent Files
Open Project...
Open Project in New Session...
Recent Projects
Import Dataset
Save ⌘S
Save As...
Save with Encoding...
Save All ⌥⌘S
Knit Document ⇧⌘K
Compile Report...
Print...
Close ⌘W
Close All ⇧⌘W
Close All Except Current ⌥⇧⌘W
Close Project
Quit Session...
R Script ⇧⌘N
R Notebook
R Markdown...
Shiny Web App...
Text File
C++ File
R Sweave
R HTML
R Presentation
R Documentation
Create a new R script (⇧⌘N)
Project: (None)
Environment
History
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Zoom
Export
Console
Platform:
R is free
You are wel
Type 'licer
Natural
R is a coll
Type 'contr
'citation()
Type 'demo
'help.star
Type 'q()'
> 1+3
[1] 4
> 1/3
[1] 0.3333333
> sqrt(3)
[1] 1.732051
>

RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help 100% Fri 11:41:06

RStudio

Addins

Project: (None)

Untitled1*

Source on Save

Run

Source

```
1 1+3
2 1/3
3 sqrt(3)
4
5
```

4:1 (Top Level) R Script

Environment History

Import Dataset

List

Global Environment

Environment is empty

Files Plots Packages Help Viewer

Zoom Export

Console ~/

```
Platform: x86_64-apple-darwin13.4.0 (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.

>
>
> 1+3
[1] 4
> 1/3
[1] 0.3333333
> sqrt(3)
[1] 1.732051
>
```

R Functions

- Built-in functions are operations that one can “perform” on object that are available in R
- User-defined functions are functions that are written by the user
- Packages are R functions that are written by the R community that need to be loaded before using them

Basic Arithmetic

- `20 + 3`
  - `[1] 23`
- `20 - 3`
  - `[1] 17`
- `20 * 3`
  - `[1] 60`
- `20 / 3`
  - `[1] 6.666667`
- `20 ^ 3`
  - `[1] 8000`
- `20 %% 3`
  - `[1] 2`
- `20 %/% 3`
  - `[1] 6`

$(6 * 3) + 2 = 20$

Built-In Math Functions

Function	Description
abs(*x***)**	absolute value
sqrt(*x***)**	square root
ceiling(*x***)**	ceiling(3.475) is 4
floor(*x***)**	floor(3.475) is 3
trunc(*x***)**	trunc(5.99) is 5
round(*x***, digits=***n***)**	round(3.475, digits=2) is 3.48
signif(*x***, digits=***n***)**	signif(3.475, digits=2) is 3.5
cos(*x***), sin(***x***), tan(***x***)**	also acos(*x*), cosh(*x*), acosh(*x*), etc.
log(*x***)**	natural logarithm
log10(*x***)**	common logarithm
exp(*x***)**	e^*x*

Built-In String Functions

Function	Description
substr(*x*, **start=***n1*, **stop=***n2***)**	Extract or replace substrings in a character vector. x <- "abcdef" substr(x, 2, 4) is "bcd" substr(x, 2, 4) <- "22222" is "a222ef"
grep(*pattern*, *x* , **ignore.case=**FALSE, **fixed=**FALSE**)**	Search for *pattern* in *x*. If fixed =FALSE then *pattern* is a regular expression. If fixed=TRUE then *pattern* is a text string. Returns matching indices. grep("A", c("b","A","c"), fixed=TRUE) returns 2
sub(*pattern*, *replacement*, *x*, **ignore.case =**FALSE, **fixed=**FALSE**)**	Find *pattern* in *x* and replace with *replacement* text. If fixed=FALSE then *pattern* is a regular expression. If fixed = T then *pattern* is a text string. sub("\\s",".","Hello There") returns "Hello.There"
strsplit(*x*, *split***)**	Split the elements of character vector *x* at *split*. strsplit("abc", "") returns 3 element vector "a","b","c"
paste(..., sep="")	Concatenate strings after using *sep* string to seperate them. paste("x",1:3,sep="") returns c("x1","x2" "x3") paste("x",1:3,sep="M") returns c("xM1","xM2" "xM3") paste("Today is", date())
toupper(*x***)**	Uppercase
tolower(*x***)**	Lowercase

Built-In Stat Functions

Function	Description

dnorm(*x***)**	normal density function (by default m=0 sd=1)
	# plot standard normal curve
	x <- pretty(c(-3,3), 30)
	y <- dnorm(x)
	plot(x, y, type='l', xlab="Normal Deviate", ylab="Density", yaxs="i")

pnorm(*q***)**	cumulative normal probability for q (area under the normal curve to the left of q) pnorm(1.96) is 0.975
qnorm(*p***)**	normal quantile. value at the p percentile of normal distribution qnorm(.9) is 1.28 # 90th percentile
rnorm(*n***, m=**0**,sd=**1**)**	n random normal deviates with mean m and standard deviation sd. #50 random normal variates with mean=50, sd=10 x <- rnorm(50, m=50, sd=10)

dbinom(*x***, *size*, *prob*)**
pbinom(*q***, *size*, *prob*)**
qbinom(*p***, *size*, *prob*)**
rbinom(*n***, *size*, *prob*)**
	binomial distribution where size is the sample size
	and prob is the probability of a heads (pi)
	# prob of 0 to 5 heads of fair coin out of 10 flips
	dbinom(0:5, 10, .5)
	# prob of 5 or less heads of fair coin out of 10 flips
	pbinom(5, 10, .5)

dpois(*x***, *lamda*)**
ppois(*q***, *lamda*)**
qpois(*p***, *lamda*)**
rpois(*n***, *lamda*)**
	poisson distribution with m=std=lamda
	#probability of 0,1, or 2 events with lamda=4
	dpois(0:2, 4)
	# probability of at least 3 events with lamda=4
	1- ppois(2,4)

dunif(*x***, min=**0**, max=**1**)** **punif(***q***, min=0, max=**1**)** **qunif(***p***, min=**0**, max=**1**)** **runif(***n***, min=**0**, max=**1**)**	uniform distribution, follows the same pattern as the normal distribution above. #10 uniform random variates x <- runif(10)

Built-In Stat Functions

Function	Description
mean(*x***, trim=**0**,** **na.rm=**FALSE**)**	mean of object x # trimmed mean, removing any missing values and # 5 percent of highest and lowest scores mx <- mean(x,trim=.05,na.rm=TRUE)
sd(*x***)**	standard deviation of object(x). also look at var(x) for variance and mad(x) for median absolute deviation.
median(*x***)**	median
quantile(*x***,** *probs***)**	quantiles where x is the numeric vector whose quantiles are desired and probs is a numeric vector with probabilities in [0,1]. # 30th and 84th percentiles of x y <- quantile(x, c(.3,.84))
range(*x***)**	range
sum(*x***)**	sum
diff(*x***, lag=***1***)**	lagged differences, with lag indicating which lag to use
min(*x***)**	minimum
max(*x***)**	maximum
scale(*x***, center=**TRUE**, scale=**TRUE**)**	column center or standardize a matrix.

Built-In Functions

Function	Description
seq(*from* **,** *to***,** *by***)**	generate a sequence indices <- seq(1,10,2) #indices is c(1, 3, 5, 7, 9)
rep(*x***,** *ntimes***)**	repeat *x n* times y <- rep(1:3, 2) # y is c(1, 2, 3, 1, 2, 3)
cut(*x***,** *n***)**	divide continuous variable in factor with *n* levels y <- cut(x, 5)

Getting Help with Functions

- R Help: help() and ?
- The help() function and ? help operator in R provide access to the documentation pages for R functions, data sets, and other objects, both for packages in the standard R distribution and for contributed packages.
- To access documentation for the standard lm (linear model)
    - help(lm)
    - help(“lm")
    - ?lm
    - ?"lm" (i.e., the quotes are optional).
- To access help for a function in a package that’s not currently loaded, specify in addition the name of the package: for the rlm() (robust linear model) function in the MASS package:
    - help(rlm, package="MASS")

R Data Types

- R objects have data types
  - Character
    - `"a", "abc","hello world"`
  - Numeric (double or integer)
    - `1, 10.3, -199`
  - Logical
    - `TRUE, FALSE`
  - Complex
    - `1+4i`

R Data Types

- These data types allow you to know what sort of functions can be performed
- For example numbers can be added but characters can't be
  - `1 + 2`
    - `[1] 3`
  - `'a' + 'b'`
    - `Error in "a" + "b" : non-numeric argument to binary operator`

R Variables

- The variables in R are technically known as objects
- Objects should have meaningful names
    - Try to avoid common function names or else it gets confusing
        - mean, sqrt
- Object names CANNOT start with a number
- Object names CAN have “.” and numbers within them
    - Avoid “_”

Creating R Objects

- Variables (objects) can be created by = or <-
  - `x <- “abc"`
  - `x = “abc"`
- Determine data type of a variable
  - `typeof(x)`
    - [1] "character"
- Determine the number of variables in a variable
  - `length(x)`
    - [1] 1
- Determine the number of characters in a variable
  - `nchar(x)`
    - [1] 3

R Data Structures

- Vectors
  - Atomic vector — a collection of values
  - Factors — special vectors that represent categorical data
- Matrix — a special vector with rows and columns
- Data frame — a special data structure of rows and columns, the default structure for reading in “excel-like” files
- List — a vector of different data types (including other vectors)

Scalars are Vectors

- A vector can be a “collection” of values or a single value

```
20 + 3
```

```
[1] 23
```

R Arithmetic using variables

- `x <- 20`
- `y <- 3`
- `x + y`
  - [1] 23
- `x - y`
  - [1] 17
- `x * y`
  - [1] 60
- `x / y`
  - [1] 6.666667
- `x ^ y`
  - [1] 8000
- `x %% y (remainder of the division)`
  - `[1] 2`
- `x %/% y (integer of the division)`
  - [1] 6
- sqrt(9)
  - [1] 3

$(6 * 3) + 2 = 20$

Manipulation of Strings

- `rep(3,'a')`
  - `[1] "a" "a" "a"`
- `a <- "Hello"`
- `b <- 'How'`
- `c <- “are you? "`
- `print(paste(a,b,c))`
  - `[1] "Hello How are you? "`
- `print(paste(a,b,c, sep = "-"))`
  - `[1] "Hello-How-are you? "`
- `print(paste(a,b,c, sep = "", collapse = ""))`
  - `[1] "HelloHoware you? “`
- `toupper(a)`
  - `[1] "HELLO"`
- `tolower(a)`
  - `[1] “hello"`
- `substring(a,1,2)`
  - `[1] "He"`
- `substring(a,2,5)`
  - `[1] "ello"`

R Vectors

- A vector can be a “collection” of values or a single value
- An example of a numeric vector
  - `x <- c(1:10)`
  - `x`
    - [1] 1 2 3 4 5 6 7 8 9 10
  - `length(x)`
    - [1] 10
  - `typeof(x)`
    - [1] “integer
- Vectors can be any datatype (character, logical, complex)
  - `x <- c(‘a','b','c','d')`
  - `x`
    - [1] "a" "b" "c" "d"
  - `typeof(x)`
    - [1] "character"

Vectors

- `x <- c(1:10)`
- `x > 3`
  - [1] FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
- `(x > 3) & (x < 8)`
  - `[1] FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE`
- `x[x > 3]`
  - [1] 4 5 6 7 8 9 10
- `typeof((x > 3) & (x < 8))`
  - [1] “logical"

Vectors

- `x <- c(1:10)`
- `x <- c(x,11:20)`
- `x`
  - `[1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20`

- `x <- c(1:10)`
- `x*3`
  - `[1]  3  6  9 12 15 18 21 24 27 30`
- `mean(x)`
  - `[1] 5.5`
- `x <- x*3`
- `x[4]`
  - `[1] 12`

Factors

- Tell R that a variable is **nominal** by making it a factor.
- The factor stores the nominal values as a vector of integers in the range [1... k] and an internal vector of character strings (the original values) mapped to these integers.
- `genotype <- c(rep("WT",5),rep("KO",5))`
- `factor(genotype)`
  - `[1] WT WT WT WT WT KO KO KO KO KO`
  - `Levels: KO WT`
- `genotype <- factor(genotype,levels=c("WT","KO"))`
- `genotype`
  - `[1] WT WT WT WT WT KO KO KO KO KO`
  - `Levels: WT KO`
- `summary(genotype)`
  - `WT KO`
  - `5 5`
- `genotype <- factor(genotype,levels=c("WT","KO"),ordered=TRUE)`
- `min(genotype)`
  - `[1] WT`
  - `Levels: WT < KO`

R Data Structures

- Vectors
  - Atomic vector — a collection of values
  - Factors — special vectors that represent categorical data
- Matrix — a special vector with rows and columns
- Data frame — a special data structure of rows and columns, the default structure for reading in “excel-like” files
- List — a vector of different data types (including other vectors)

Matrices

- All columns in a matrix must have the same mode(numeric, character, etc.) and the same length. The general format is
  - mymatrix <- **matrix(**vector, **nrow=**r, **ncol=**c, **byrow=**FALSE)
  - **byrow=TRUE** indicates that the matrix should be filled by rows.
  - **byrow=FALSE** indicates that the matrix should be filled by columns (the default)
- y<-matrix(1:20, nrow=5,ncol=4)

Matrices

- `cells <- c(1,26,24,68)`
- `rnames <- c("R1", “R2")`
- `cnames <- c("C1", "C2")`
- `x <- matrix(cells, nrow=2, ncol=2, byrow=TRUE, dimnames=list(rnames, cnames))`
- `x[,1] # 1st column of matrix`
  - `R1 R2`
  - `  1 24`
- `x[2,] # 2nd row of matrix`
  - `C1 C2`
  - `24 68`
- `x[2,1] # row 2, column 1`
  - `[1] 24`
- `x[1,2]`
  - `Guess from the class?`

Matrices

```
> y<-matrix(1:20, nrow=5,ncol=4,byrow = FALSE)
     [,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
> y<-matrix(1:20, nrow=5,ncol=4,byrow = TRUE)
     [,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
> t(y) #transpose
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
```

Matrices

- diag(y)
- [1] 1 6 11 16
- diag(4)
- [,1] [,2] [,3] [,4]
- [1,] 1 0 0 0
- [2,] 0 1 0 0
- [3,] 0 0 1 0
- [4,] 0 0 0 1
- y*4
- [,1] [,2] [,3] [,4]
- [1,] 4 8 12 16
- [2,] 20 24 28 32
- [3,] 36 40 44 48
- [4,] 52 56 60 64
- [5,] 68 72 76 80

- y*y
- [,1] [,2] [,3] [,4]
- [1,] 1 4 9 16
- [2,] 25 36 49 64
- [3,] 81 100 121 144
- [4,] 169 196 225 256
- [5,] 289 324 361 400
- > y/y
- [,1] [,2] [,3] [,4]
- [1,] 1 1 1 1
- [2,] 1 1 1 1
- [3,] 1 1 1 1
- [4,] 1 1 1 1
- [5,] 1 1 1 1

Matrix Functions

cbind(A,B,...)	Combine matrices(vectors) horizontally. Returns a matrix.
rbind(A,B,...)	Combine matrices(vectors) vertically. Returns a matrix.
rowMeans(A)	Returns vector of row means.
rowSums(A)	Returns vector of row sums.
colMeans(A)	Returns vector of column means.
colSums(A)	Returns vector of column sums.
t(A)	Transpose

Matrix Functions

Operator or Function	Description
A * B	Element-wise multiplication
A %*% B	Matrix multiplication
A %o% B	Outer product. **AB'**
crossprod(A,B) **crossprod(A)**	**A'B** and **A'A** respectively.
diag(x)	Creates diagonal matrix with elements of **x** in the principal diagonal
diag(A)	Returns a vector containing the elements of the principal diagonal
diag(k)	If k is a scalar, this creates a k x k identity matrix. Go figure.
solve(A, b)	Returns vector **x** in the equation **b = Ax** (i.e., $\mathbf{A^{-1}b}$)
solve(A)	Inverse of **A** where A is a square matrix.
y<-eigen(A)	**y$val** are the eigenvalues of **A** **y$vec** are the eigenvectors of **A**
y<-svd(A)	Single value decomposition of **A**. **y$d** = vector containing the singular values of **A** **y$u** = matrix with columns contain the left singular vectors of **A** **y$v** = matrix with columns contain the right singular vectors of **A**
y <- qr(A)	QR decomposition of **A**. **y$qr** has an upper triangle that contains the decomposition and a lower triangle that contains information on the Q decomposition. **y$rank** is the rank of A. **y$qraux** a vector which contains additional information on Q. **y$pivot** contains information on the pivoting strategy used.

R Data Structures

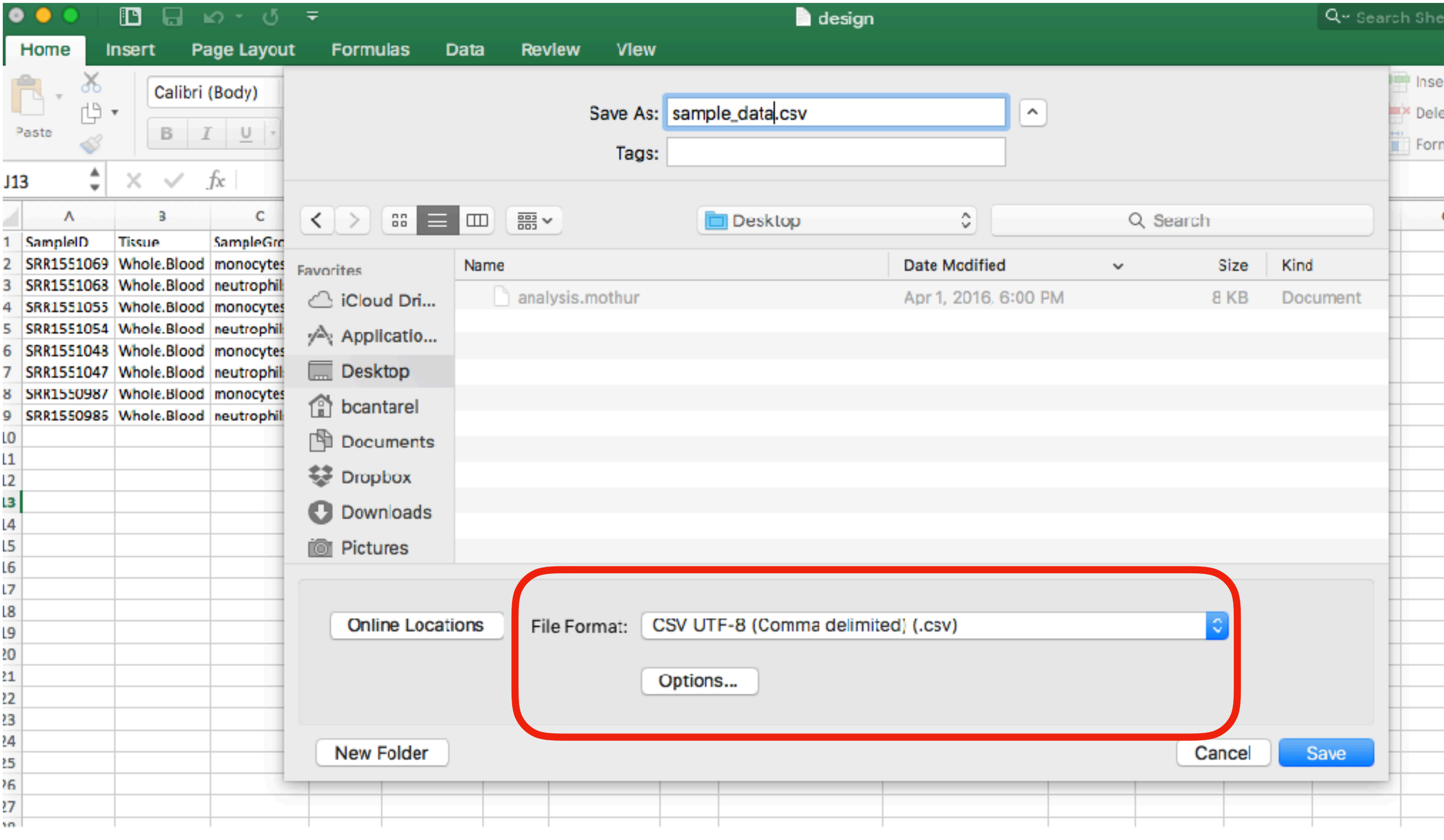
- Vectors
  - Atomic vector — a collection of values
  - Factors — special vectors that represent categorical data
- Matrix — a special vector with rows and columns
- Data frame — a special data structure of rows and columns, the default structure for reading in “excel-like” files
- List — a vector of different data types (including other vectors)

Data Frames

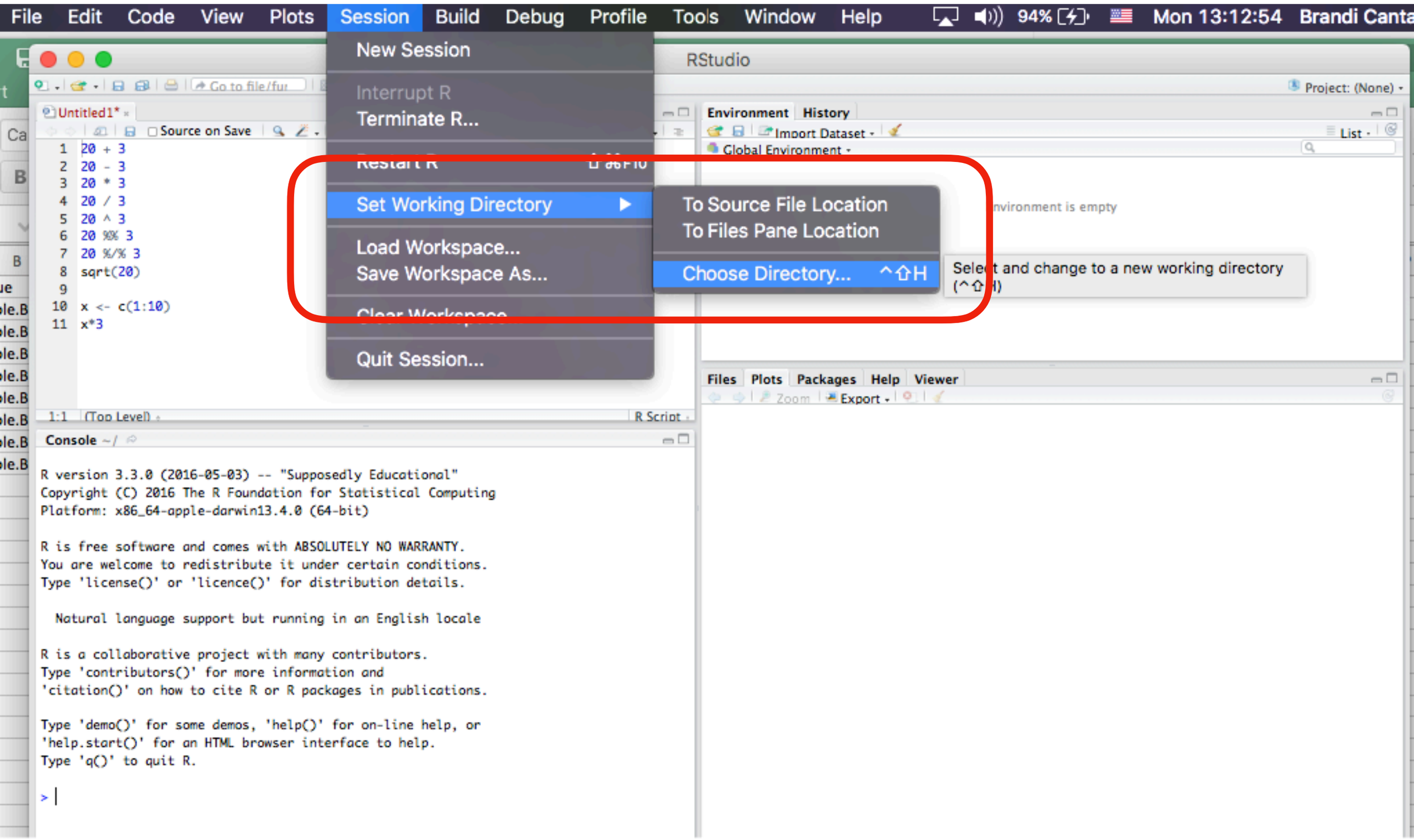
- A data frame is more general than a matrix, in that different columns can have different modes (numeric, character, factor, etc.). This is similar to SAS and SPSS datasets.
- `d <- c(1,2,3,4)`
- `e <- c("red", "white", "red", NA)`
- `f <- c(TRUE,TRUE,TRUE,FALSE)`
- `x <- data.frame(d,e,f)`
- `names(x) <- c(“ID","Color","Passed")`

```
 ID Color Passed
1  1   red   TRUE
2  2 white   TRUE
3  3   red   TRUE
4  4  <NA>  FALSE
```

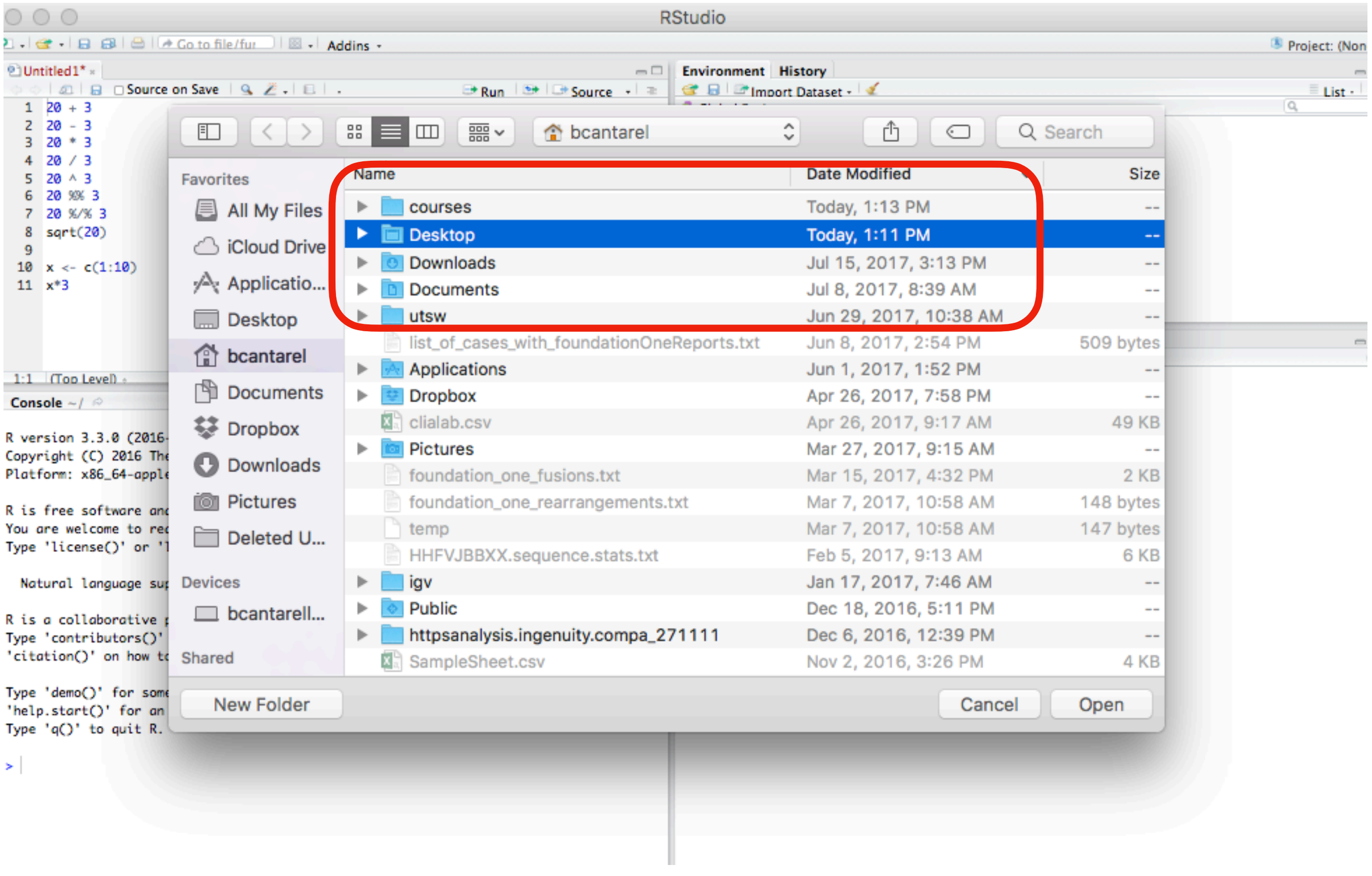
Excel To Data Frame



Excel To Data Frame



Excel To Data Frame



Excel To Data Frame

- `setwd("~/Desktop")`
- `tbl <- read.csv(file="sample_data.csv",header=TRUE)`

```
> head(tbl)
    SampleID      Tissue SampleGroup SubjectID     Organism  Race
1 SRR1551069 Whole.Blood   monocytes        53 Homo sapiens White
2 SRR1551068 Whole.Blood neutrophils        53 Homo sapiens White
3 SRR1551055 Whole.Blood   monocytes        21 Homo sapiens White
4 SRR1551054 Whole.Blood neutrophils        21 Homo sapiens White
5 SRR1551048 Whole.Blood   monocytes        20 Homo sapiens White
6 SRR1551047 Whole.Blood neutrophils        20 Homo sapiens White
      SampleName Gender        FullPathToFqR1
1   53_Monocytes female SRR1551069_1.fastq.gz
2 53_Neutrophils female SRR1551068_1.fastq.gz
3   21_Monocytes female SRR1551055_1.fastq.gz
4 21_Neutrophils female SRR1551054_1.fastq.gz
5   20_Monocytes female SRR1551048_1.fastq.gz
6 20_Neutrophils female SRR1551047_1.fastq.gz
        FullPathToFqR2
1 SRR1551069_2.fastq.gz
2 SRR1551068_2.fastq.gz
3 SRR1551055_2.fastq.gz
4 SRR1551054_2.fastq.gz
5 SRR1551048_2.fastq.gz
6 SRR1551047_2.fastq.gz
```

Data Frames

- `tbl[3:5]`
  - `columns 3,4,5 of data frame`
- `tbl[c(“SampleID”,”Tissue")]`
  - `columns SampleID and Tissue from data frame`
- `tbl$Gender`
  - `variable Gender in the data frame`
- `tbl[tbl$SampleGroup == ‘monocytes',]`
- `subset(x=tbl,SampleGroup == ‘monocytes',select=c('Tissue','SampleID'))`

Data Frames

Operator or Function	Description
str(df)	gives a very brief description of the data
names(df)	gives the name of each variables
summary(df)	gives some very basic summary statistics for each variable
head(df)	shows the first few rows
tail(df)	shows the last few rows.
duplicated()	looks at duplicated elements and returns a logical vector. You can use table() to summarize this vector.
unique()	keeps only the unique lines in a dataset.

Merge Data Frames

setwd(“~/Desktop")
tbl1 <- read.csv(file="sample_data.csv",header=TRUE)
tbl2 <- read.csv(file="table2.csv",header=TRUE)

```
> head(tbl1)
    SampleID      Tissue SampleGroup SubjectID     Organism  Race     SampleName
1 SRR1551069 Whole.Blood   monocytes        53 Homo sapiens White   53_Monocytes
2 SRR1551068 Whole.Blood neutrophils        53 Homo sapiens White 53_Neutrophils
3 SRR1551055 Whole.Blood   monocytes        21 Homo sapiens White   21_Monocytes
4 SRR1551054 Whole.Blood neutrophils        21 Homo sapiens White 21_Neutrophils
5 SRR1551048 Whole.Blood   monocytes        20 Homo sapiens White   20_Monocytes
6 SRR1551047 Whole.Blood neutrophils        20 Homo sapiens White 20_Neutrophils
  Gender        FullPathToFqR1        FullPathToFqR2
1 female SRR1551069_1.fastq.gz SRR1551069_2.fastq.gz
2 female SRR1551068_1.fastq.gz SRR1551068_2.fastq.gz
3 female SRR1551055_1.fastq.gz SRR1551055_2.fastq.gz
4 female SRR1551054_1.fastq.gz SRR1551054_2.fastq.gz
5 female SRR1551048_1.fastq.gz SRR1551048_2.fastq.gz
6 female SRR1551047_1.fastq.gz SRR1551047_2.fastq.gz
> head(tbl2)
    SampleID SubjectID BMI
1 SRR1551069        53  23
2 SRR1551068        53  23
3 SRR1551055        21  28
4 SRR1551054        21  28
5 SRR1551048        20  35
6 SRR1551047        20  35
```

Merge Data Frames

setwd(“~/Desktop")
tbl1 <- read.csv(file="sample_data.csv",header=TRUE)
tbl2 <- read.csv(file=“table2.csv",header=TRUE)
merge(tbl1,tbl2,by='SampleID')

```
> merge(tbl1,tbl2,by='SampleID')
    SampleID      Tissue SampleGroup SubjectID.x     Organism     Race     SampleName
1 SRR1550986 Whole.Blood neutrophils          44 Homo sapiens Hispanic 44_Neutrophils
2 SRR1550987 Whole.Blood   monocytes          44 Homo sapiens Hispanic   44_Monocytes
3 SRR1551047 Whole.Blood neutrophils          20 Homo sapiens    White 20_Neutrophils
4 SRR1551048 Whole.Blood   monocytes          20 Homo sapiens    White   20_Monocytes
5 SRR1551054 Whole.Blood neutrophils          21 Homo sapiens    White 21_Neutrophils
6 SRR1551055 Whole.Blood   monocytes          21 Homo sapiens    White   21_Monocytes
7 SRR1551068 Whole.Blood neutrophils          53 Homo sapiens    White 53_Neutrophils
8 SRR1551069 Whole.Blood   monocytes          53 Homo sapiens    White   53_Monocytes
  Gender        FullPathToFqR1        FullPathToFqR2 SubjectID.y BMI
1 female SRR1550986_1.fastq.gz SRR1550986_2.fastq.gz          44  31
2 female SRR1550987_1.fastq.gz SRR1550987_2.fastq.gz          44  31
3 female SRR1551047_1.fastq.gz SRR1551047_2.fastq.gz          20  35
4 female SRR1551048_1.fastq.gz SRR1551048_2.fastq.gz          20  35
5 female SRR1551054_1.fastq.gz SRR1551054_2.fastq.gz          21  28
6 female SRR1551055_1.fastq.gz SRR1551055_2.fastq.gz          21  28
7 female SRR1551068_1.fastq.gz SRR1551068_2.fastq.gz          53  23
8 female SRR1551069_1.fastq.gz SRR1551069_2.fastq.gz          53  23
```

Merge Data Frames

Description

Merge two data frames by common columns or row names, or do other versions of database *join* operations.

Usage

```
merge(x, y, ...)

## Default S3 method:
merge(x, y, ...)

## S3 method for class 'data.frame'
merge(x, y, by = intersect(names(x), names(y)),
      by.x = by, by.y = by, all = FALSE, all.x = all, all.y = all,
      sort = TRUE, suffixes = c(".x",".y"),
      incomparables = NULL, ...)
```

Arguments

`x, y`	data frames, or objects to be coerced to one.
`by, by.x, by.y`	specifications of the columns used for merging. See 'Details'.
`all`	logical; `all = L` is shorthand for `all.x = L` and `all.y = L`, where `L` is either `TRUE` or `FALSE`.
`all.x`	logical; if `TRUE`, then extra rows will be added to the output, one for each row in `x` that has no matching row in `y`. These rows will have `NA`s in those columns that are usually filled with values from `y`. The default is `FALSE`, so that only rows with data from both `x` and `y` are included in the output.
`all.y`	logical; analogous to `all.x`.
`sort`	logical. Should the result be sorted on the `by` columns?
`suffixes`	a character vector of length 2 specifying the suffixes to be used for making unique the names of columns in the result which not used for merging (appearing in `by` etc).
`incomparables`	values which cannot be matched. See `match`. This is intended to be used for merging on one column, so these are incomparable values of that column.
`...`	arguments to be passed to or from methods.

Merge Data Frames

Examples

```
## use character columns of names to get sensible sort order
authors <- data.frame(
    surname = I(c("Tukey", "Venables", "Tierney", "Ripley", "McNeil")),
    nationality = c("US", "Australia", "US", "UK", "Australia"),
    deceased = c("yes", rep("no", 4)))
books <- data.frame(
    name = I(c("Tukey", "Venables", "Tierney",
               "Ripley", "Ripley", "McNeil", "R Core")),
    title = c("Exploratory Data Analysis",
              "Modern Applied Statistics ...",
              "LISP-STAT",
              "Spatial Statistics", "Stochastic Simulation",
              "Interactive Data Analysis",
              "An Introduction to R"),
    other.author = c(NA, "Ripley", NA, NA, NA, NA,
                     "Venables & Smith"))

(m1 <- merge(authors, books, by.x = "surname", by.y = "name"))
(m2 <- merge(books, authors, by.x = "name", by.y = "surname"))
stopifnot(as.character(m1[, 1]) == as.character(m2[, 1]),
          all.equal(m1[, -1], m2[, -1][ names(m1)[-1] ]),
          dim(merge(m1, m2, by = integer(0))) == c(36, 10))

## "R core" is missing from authors and appears only here :
merge(authors, books, by.x = "surname", by.y = "name", all = TRUE)

## example of using 'incomparables'
x <- data.frame(k1 = c(NA,NA,3,4,5), k2 = c(1,NA,NA,4,5), data = 1:5)
y <- data.frame(k1 = c(NA,2,NA,4,5), k2 = c(NA,NA,3,4,5), data = 1:5)
merge(x, y, by = c("k1","k2")) # NA's match
merge(x, y, by = "k1") # NA's match, so 6 rows
merge(x, y, by = "k2", incomparables = NA) # 2 rows
```

Lists

- An ordered collection of objects (components). A list allows you to gather a variety of (possibly unrelated) objects under one name.
- w <- list(name="Fred", mynumbers=a, mymatrix=y, age=5.3)

Workspace Functions

ls()	lists the objects in your workspace
rm(object1,object2)	removes an object in your workspace
rm(list=ls())	removes all objects in your workspace
save(object1,object2,file="file.RData")	saves R objects to a file
load("file.Rdata")	load an R object from a file

Workspace Functions

getwd()	find current working directory
setwd(‘C:/workingDirectory’) #	set working directory
quit()	quit
library()	list all packages available to load
library(package)	load package
require(package)	load package

Install Packages

- source(“http://bioconductor.org/biocLite.R")
- biocLite("DESeq2")
  - # Package for DE analysis of RNA-seq data
- install.packages(“ggplot2")
  - # Used to create plots in R

Question?