Introduction to R

วิชา การค้นพบองค์ความรู้และการทำเหมืองข้อมูลชั้นสูง

Veerasak Kritsanapraphan

Chulalongkorn University

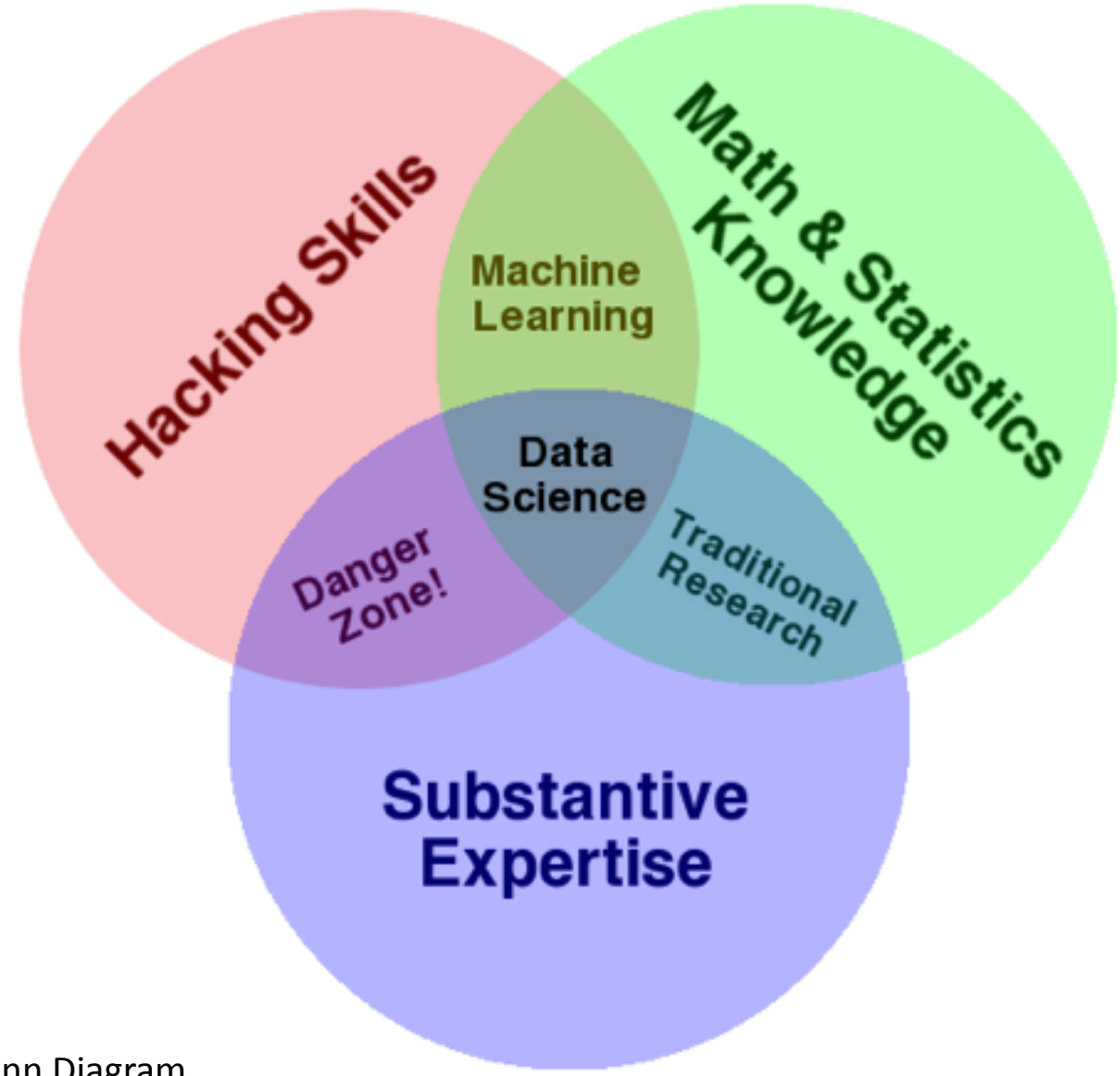
Email : veerasak.kr568@cbs.chula.ac.th

 @veerasakk

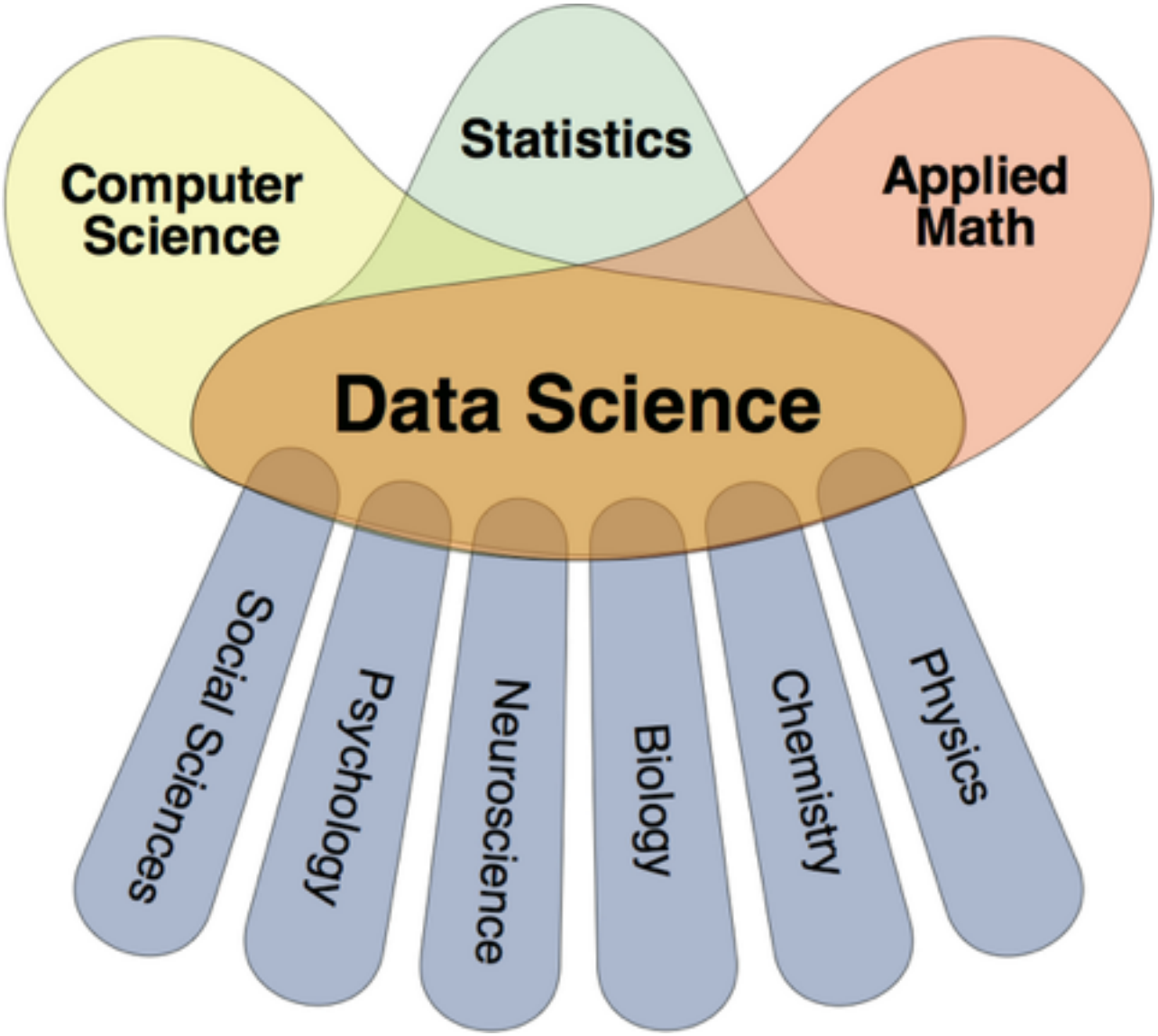
Slide and Sample Data

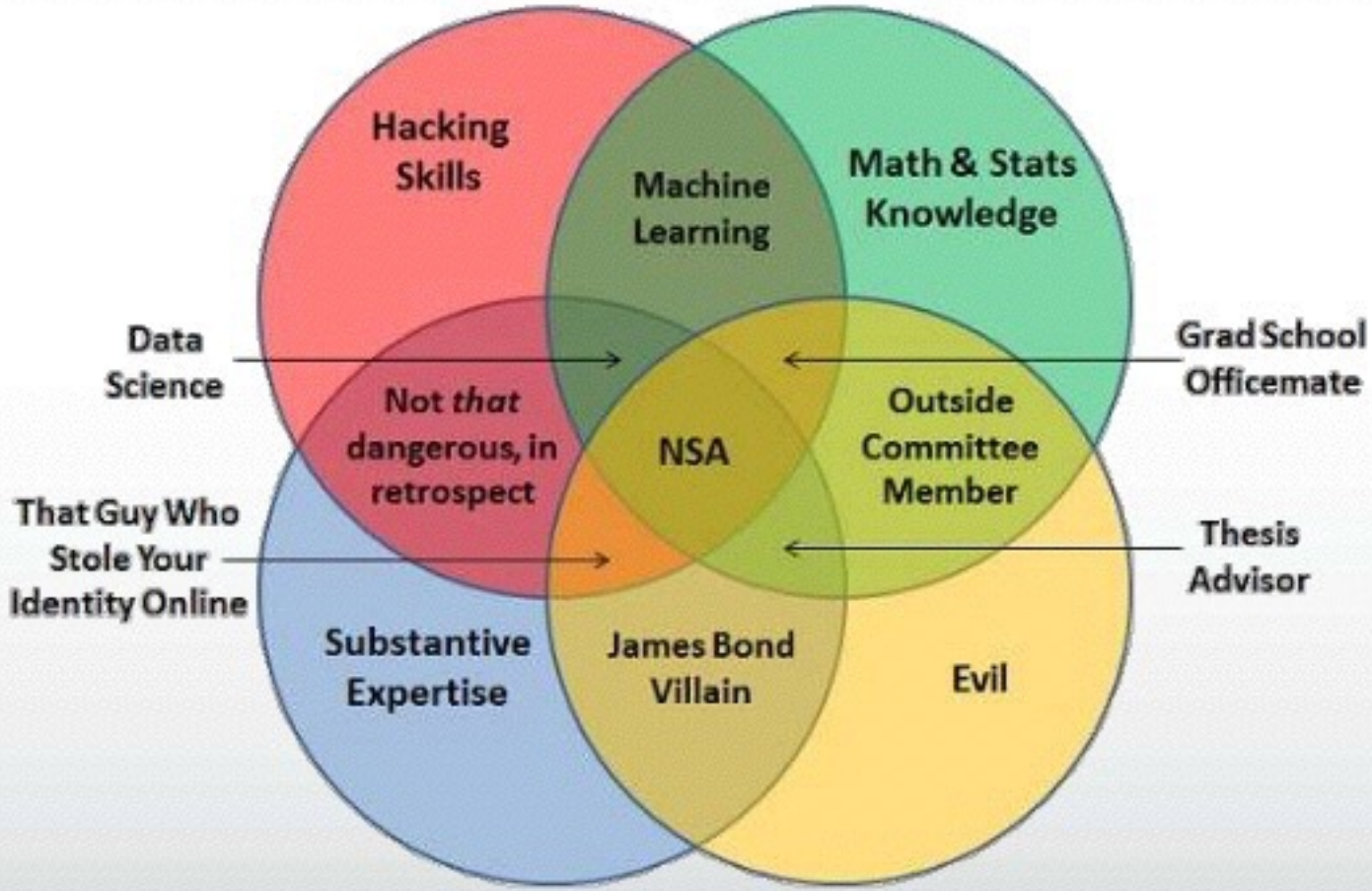
https://github.com/vkrit/chula_datamining

The beginner's perspective



Drew Conway's Venn Diagram


Computer Science
Statistics
Applied Math
Data Science
Social Sciences
Psychology
Neuroscience
Biology
Chemistry
Physics



http://joelgrus.com/2013/06/09/post-prism-data-science-venn-diagram

What is R?

- R is a system for statistical computation and graphics.
- It is heavily influenced by the **S** language
- **R** was initially written by **R**oss Ihaka and Robert Gentleman at the Department of Statistics of the University of Auckland in Auckland, New Zealand.
- The "**R** Core Team" maintain the source code for the software and release regular updates

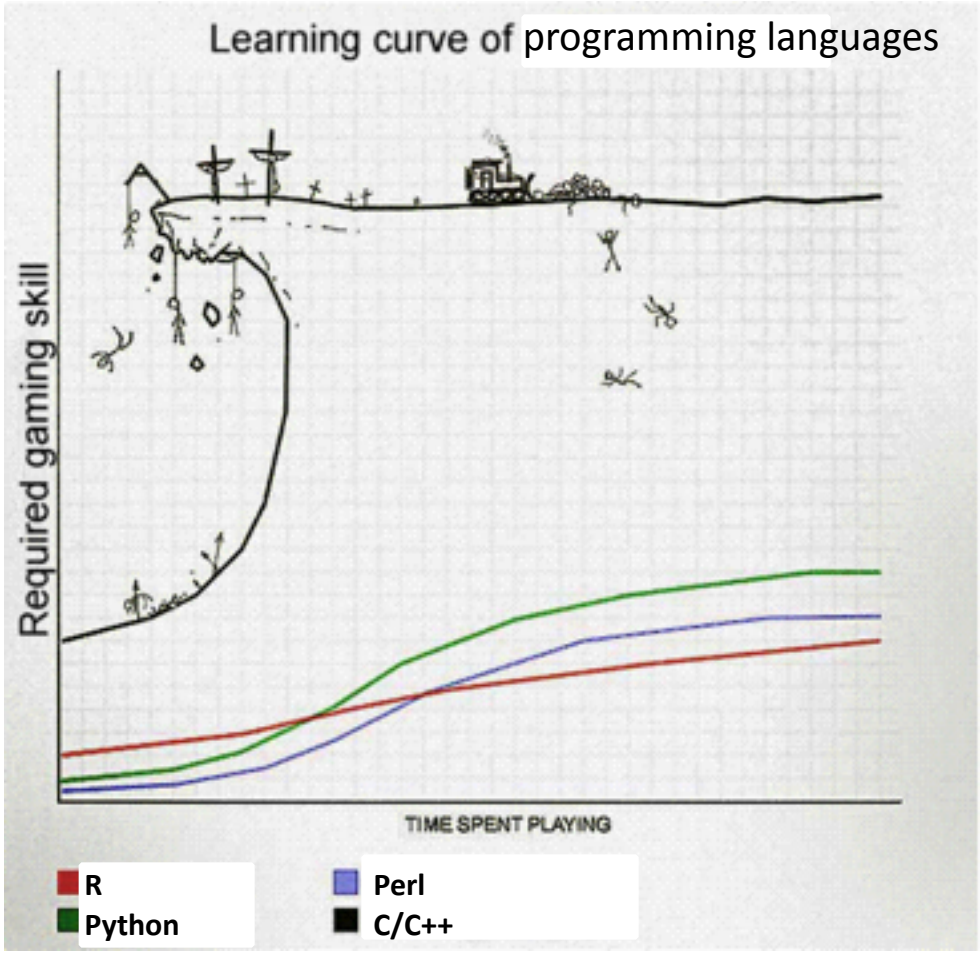
What is R?

- In addition, the R project is added to by many of its users, who write source code for many different types of analytical procedures
- Everything from analytical chemistry to epidemiology to linguistics
- Currently 4,045 different user--written libraries available

Why use R?

- R is Open-Source Software
- Many built-in functions and installable packages that will cover nearly every possible need
- R is an interpreted language
  - Code doesn’t have to be compiled
- Interactive console makes testing and debugging easy
- Cons to using R
  - Slower than compiled languages
  - Can have runtime errors

Why use R?



http://www.vayapotra.es/wordpress/wp-content/uploads/2009/02/2rmqi6o.gif

Tools

R
www.r-project.org

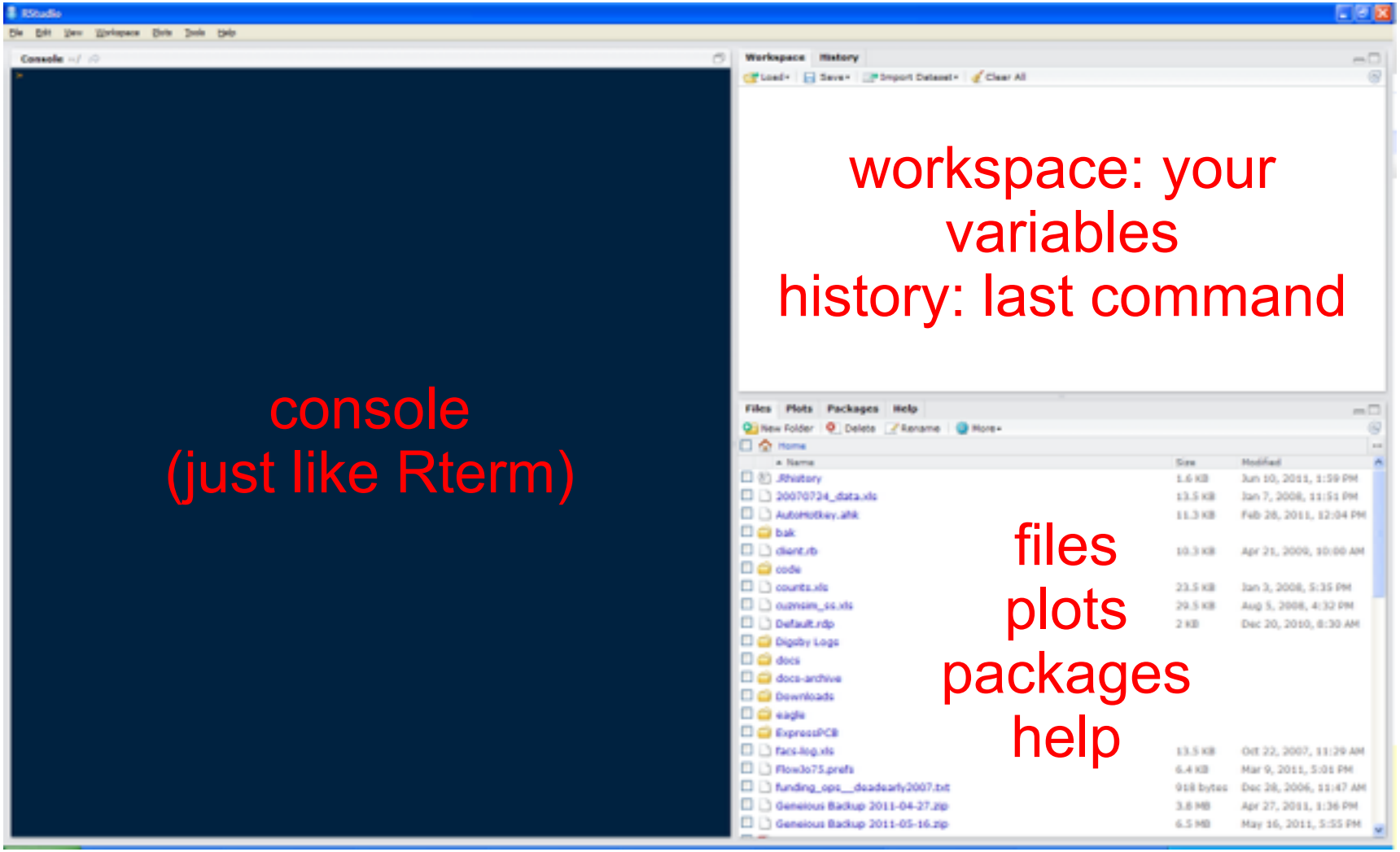
The engine*

RStudio
www.rstudio.org

The pretty face**

** Many alternatives exist. Smallest learning curve.*
*** A few alternatives exist. This happens to be the easiest at the moment.*

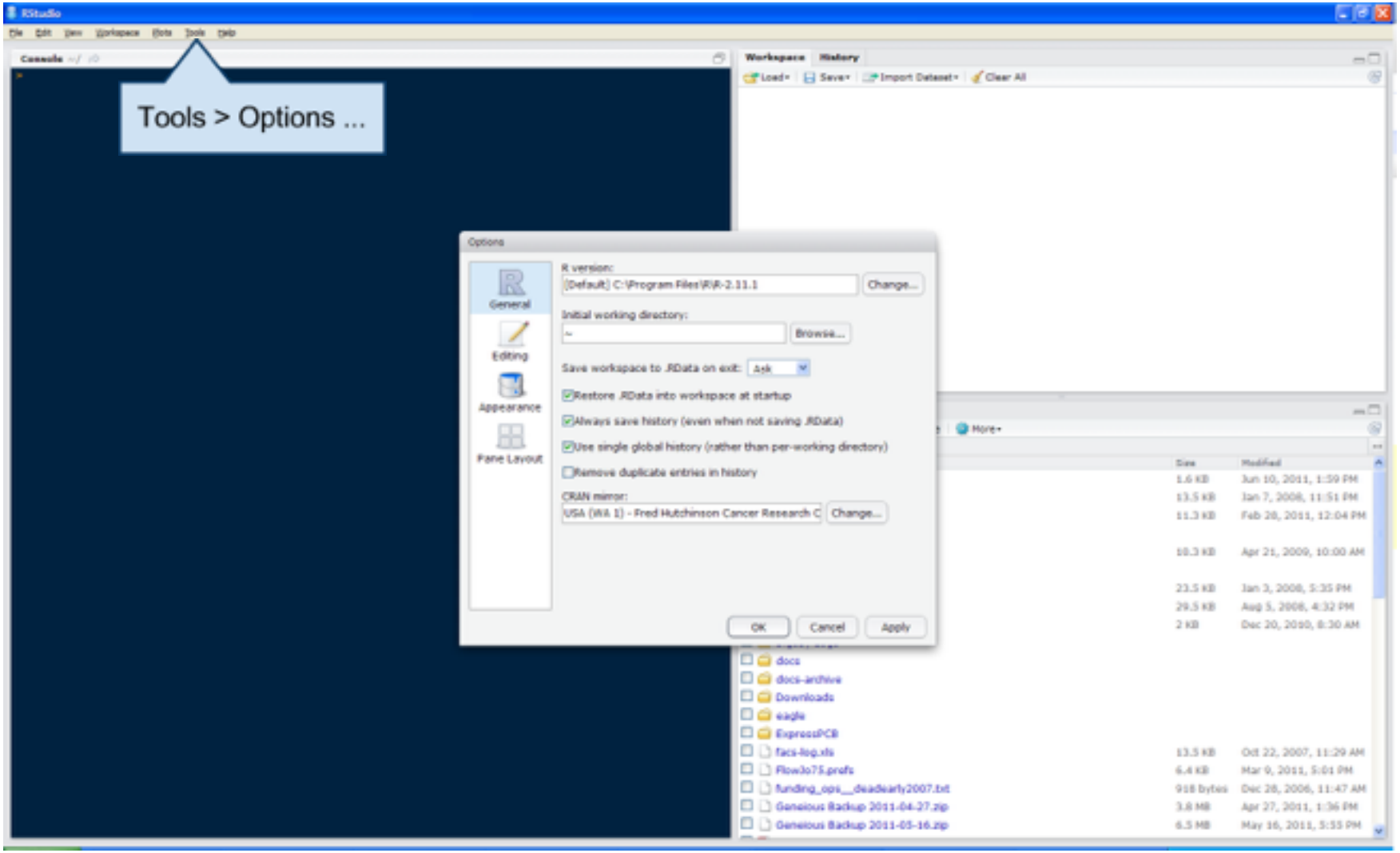
Introducing RStudio



RStudio Features

- Code completion
- Command history search
- Command history to R script / file
- Function extraction from Rscript
- Sweave support

Configuring RStudio



Choosing a CRAN Mirror

- CRAN mirrors contain the R packages that can extend the functionality of R

- Choose a mirror located close to you as that will most likely give you the fastest downloads

Choosing Repositories

- Repositories host the packages
  - CRAN, CRANextra, BioCsoft, BioCann, BioCexp, BioCext, Omegahat, R-Forge and rforge.net

- Use this code to set your repositories

```
setRepositories()
```

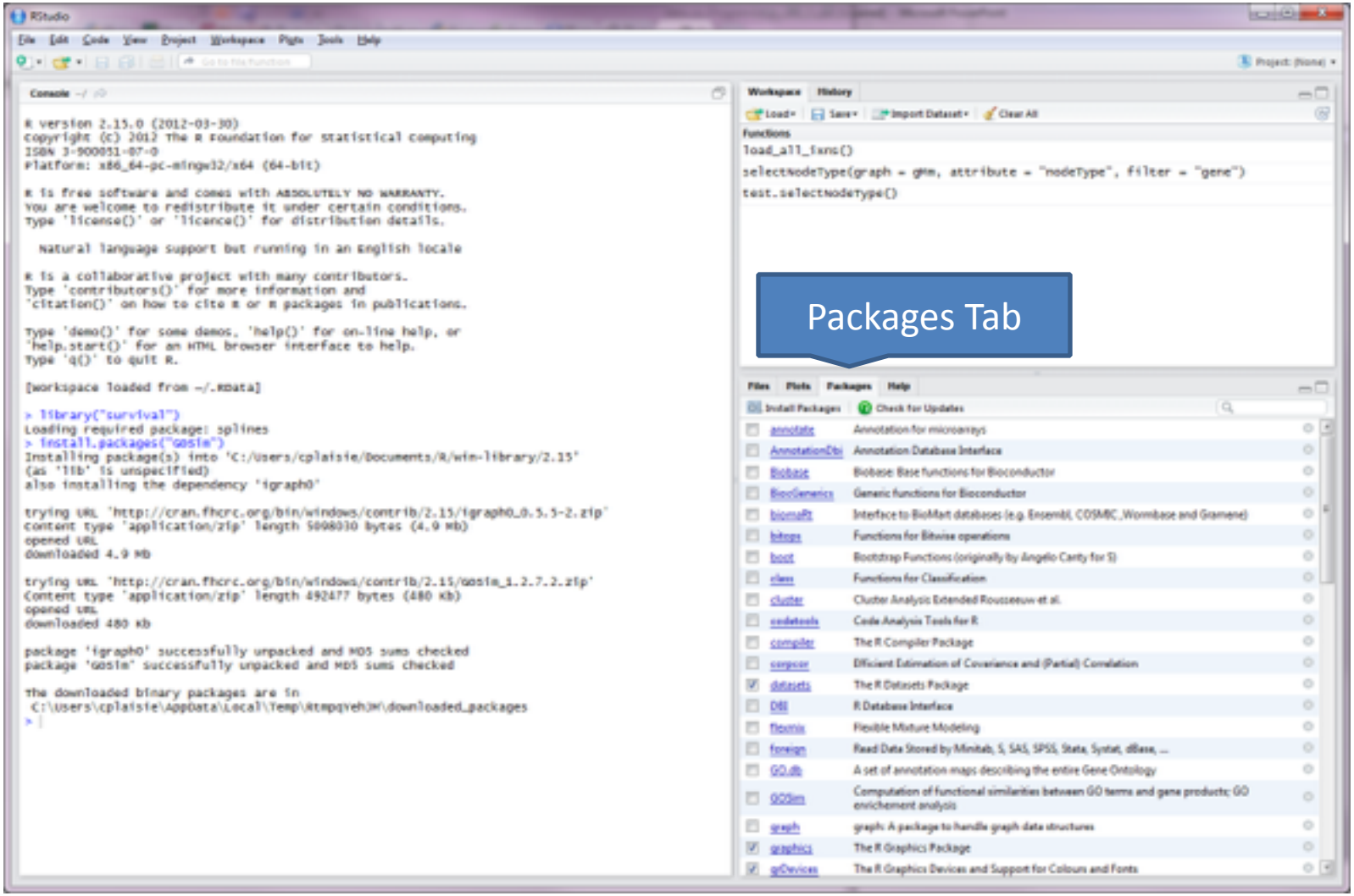
setRepositories()

```
> setRepositories()
--- Please select repositories for use in this session ---


1: + CRAN
2: + CRAN (extras)
3:   BioC software
4:   BioC annotation
5:   BioC experiment
6:   BioC extra
7:   Omegahat
8:   R-Forge
9:   rforge.net

Enter one or more numbers separated by spaces, or an empty line to cancel
1:
```

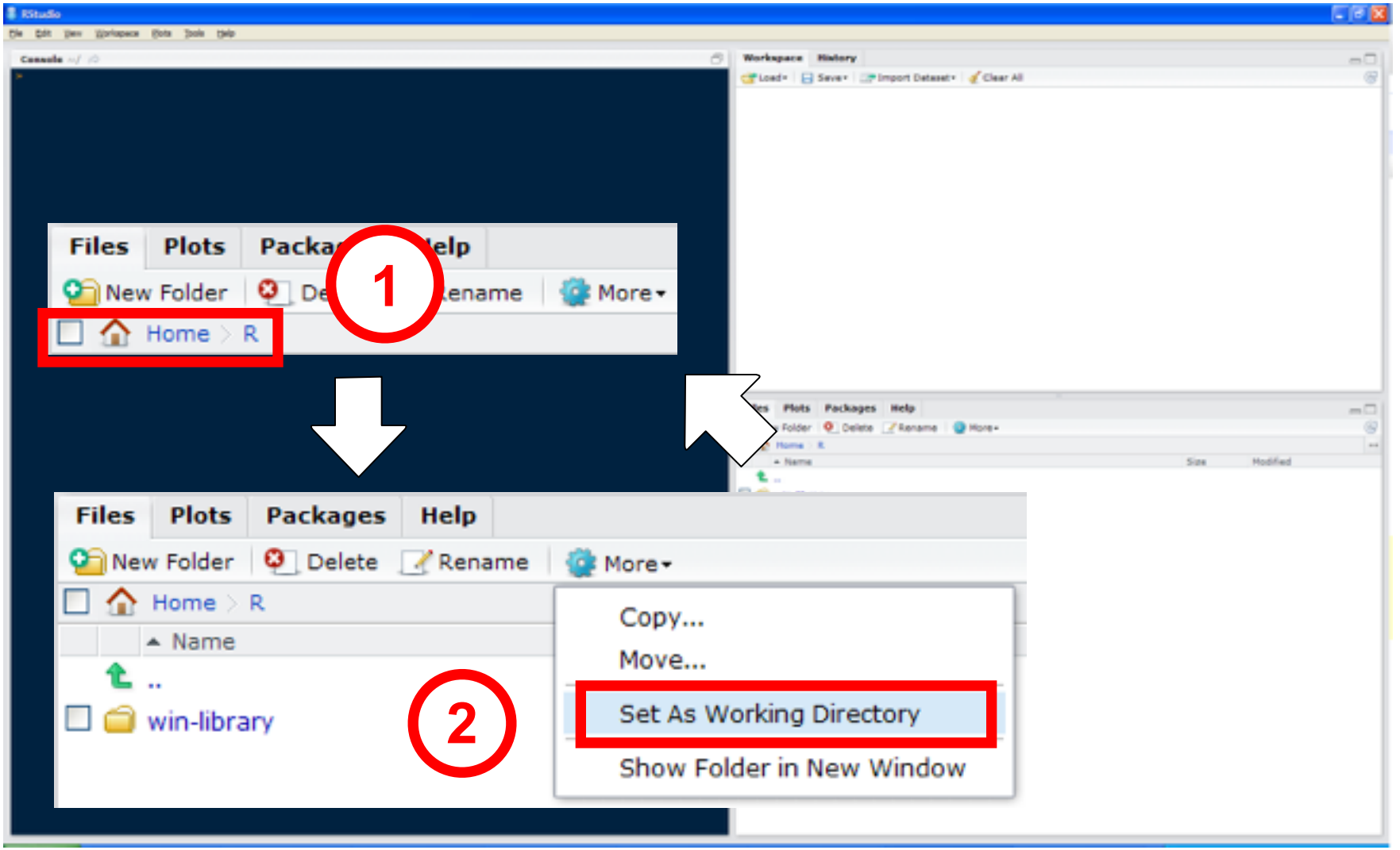
Packages Tab



How do Packages Work?

- Packages tab lets you see what packages are installed
- A package must be loaded before you can use it
  - In Rstudio this is accomplished by clicking the checkbox next to the package name in the package tab
- We will be using different packages throughout this tutorial

Working Directory



Ready to Code!

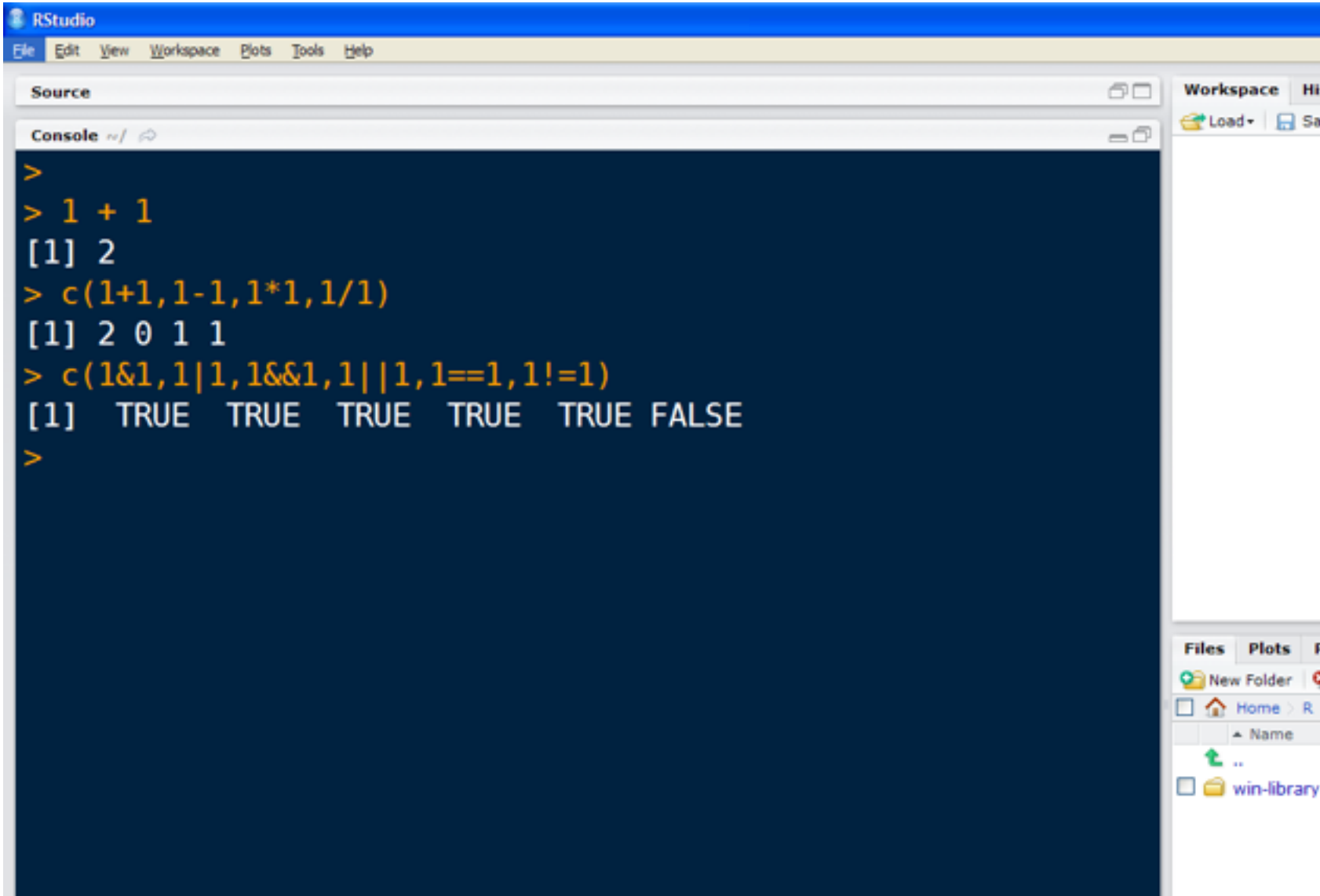
- The working directory is where RStudio will look first for scripts

- Keeping everything in a self contained directory helps organize code and analyses

- Check you current working directory with

```
getwd()
```

R as a Calculator



Basic Math / Basic Logic

- + addition
- - subtraction
- * multiplication
- / division
- % modulus (remainder)
- ^ to the power

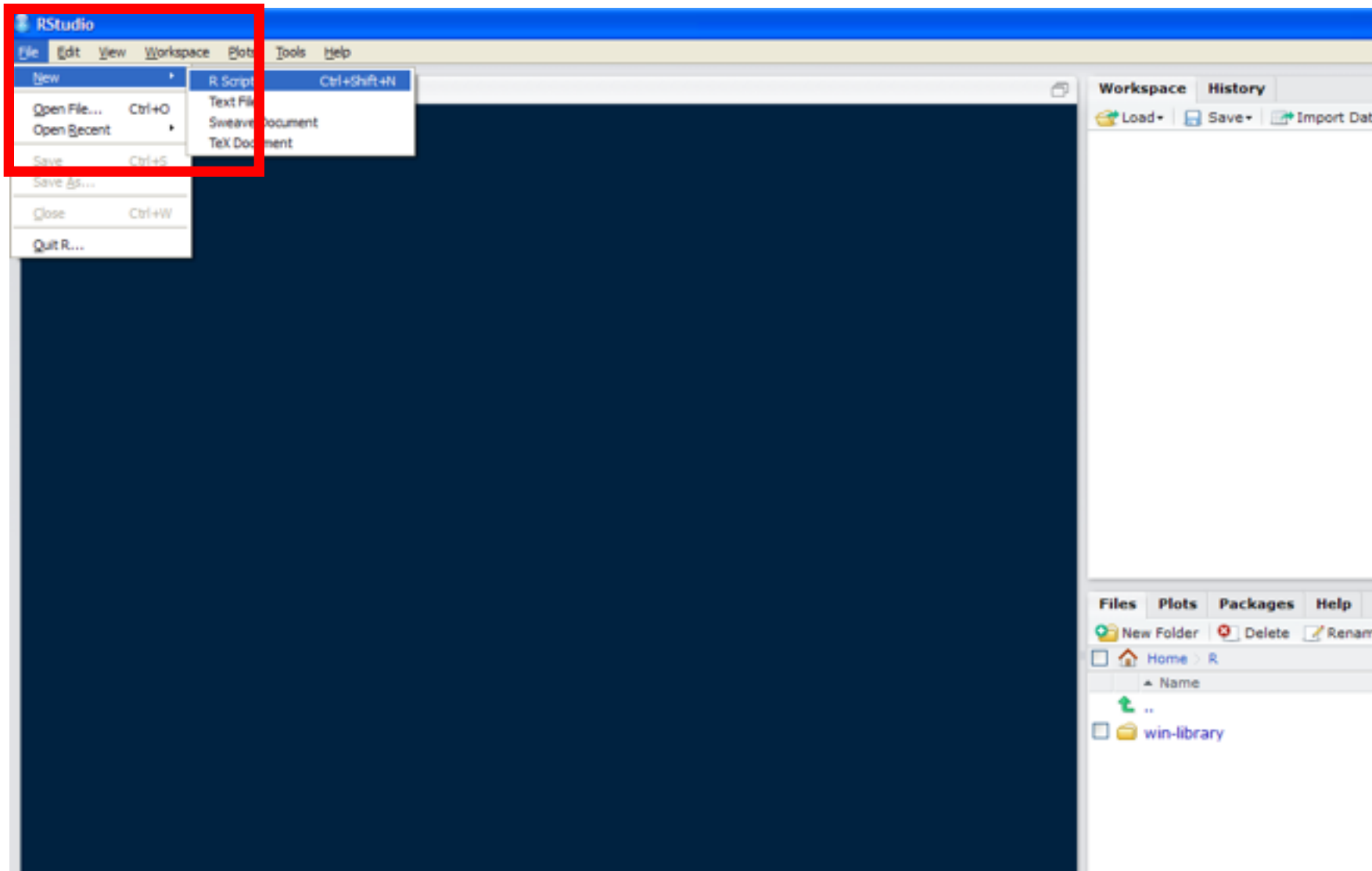
```
?Arithmetic
```

- ! NOT
- & bitwise AND
- | bitwise OR
- && short circuit AND
- || short circuit OR
- == equality
- != NOT equality

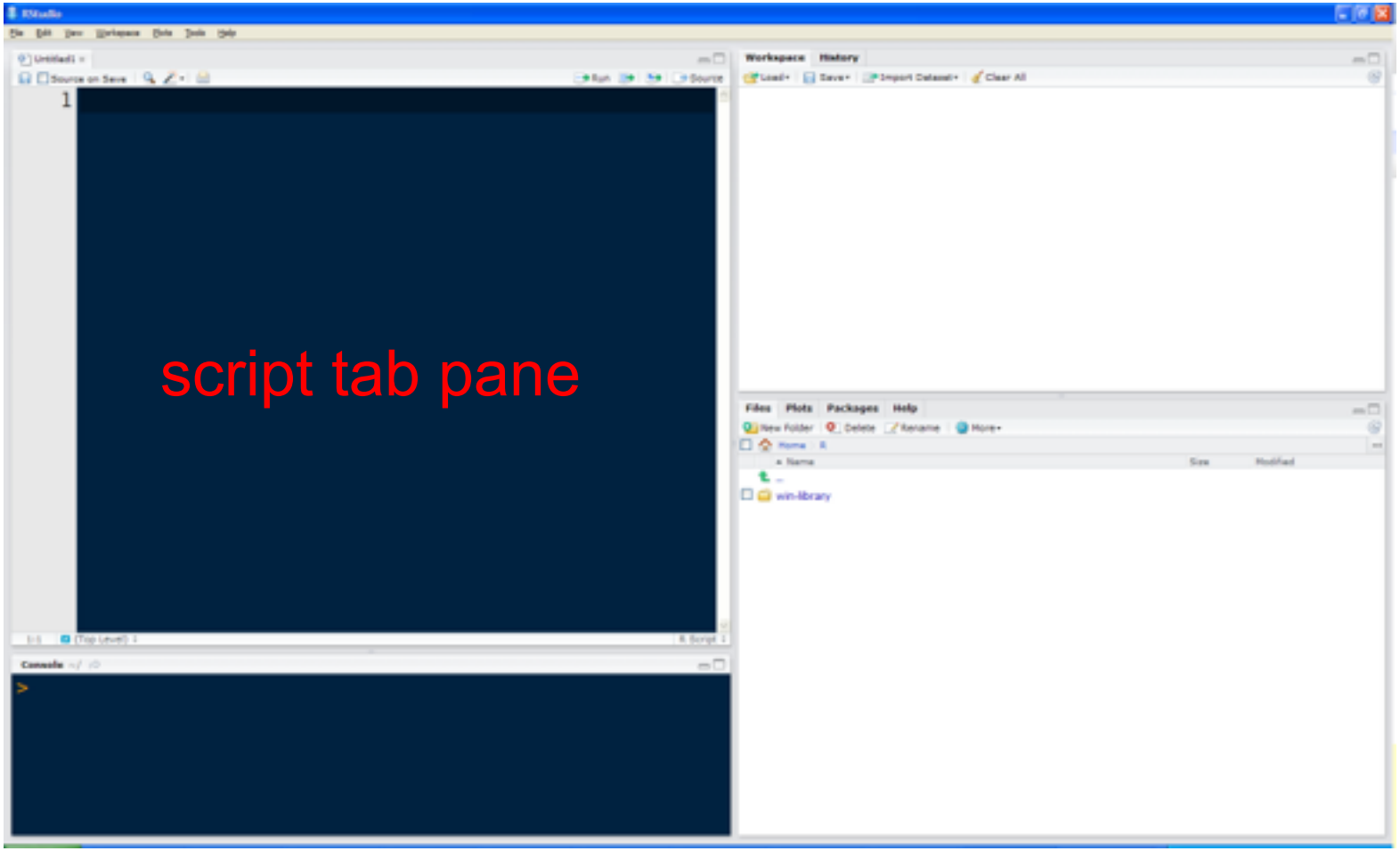
```
?Logic
```

Also try `?Syntax`, `?Comparison`

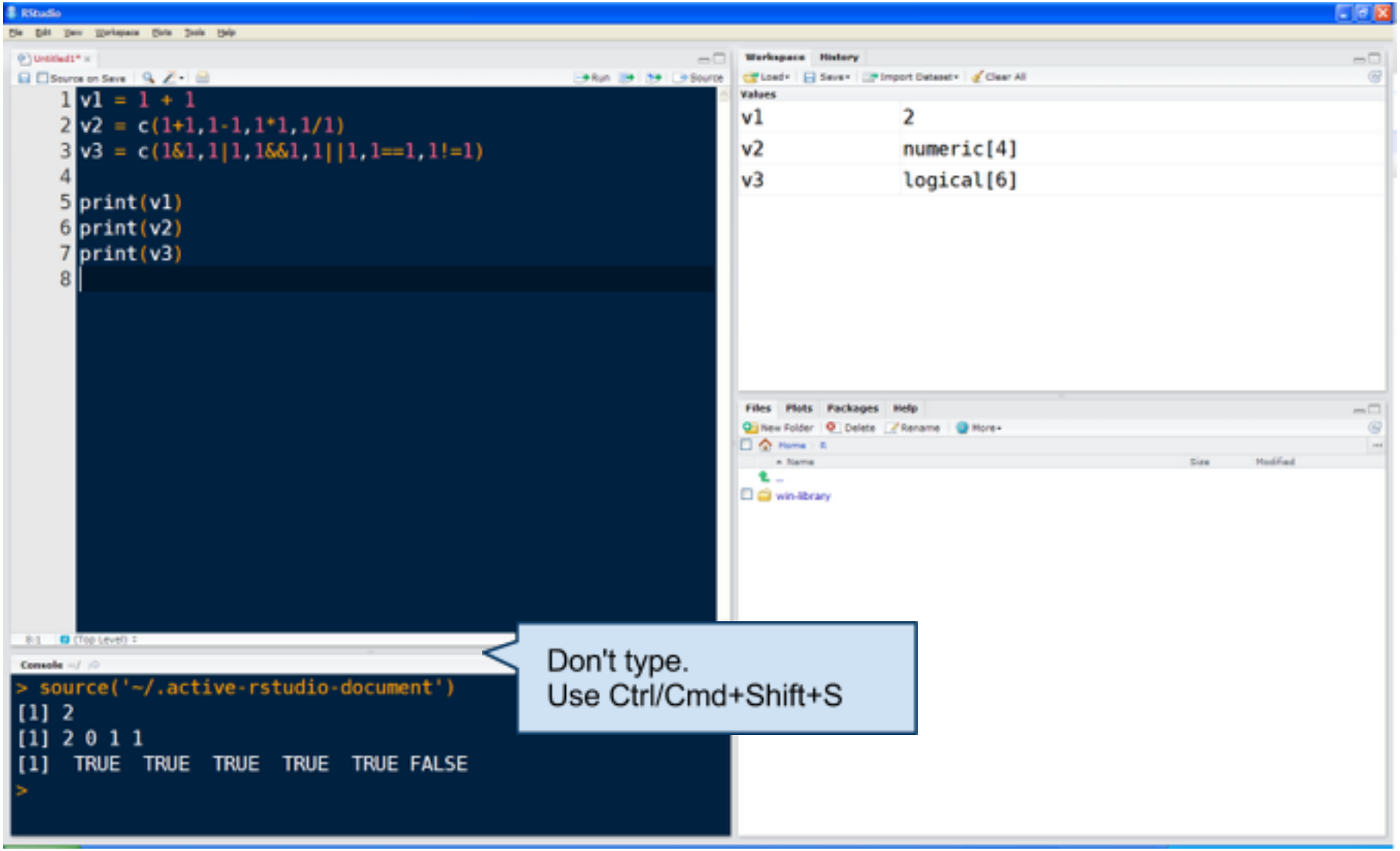
Make New R Script

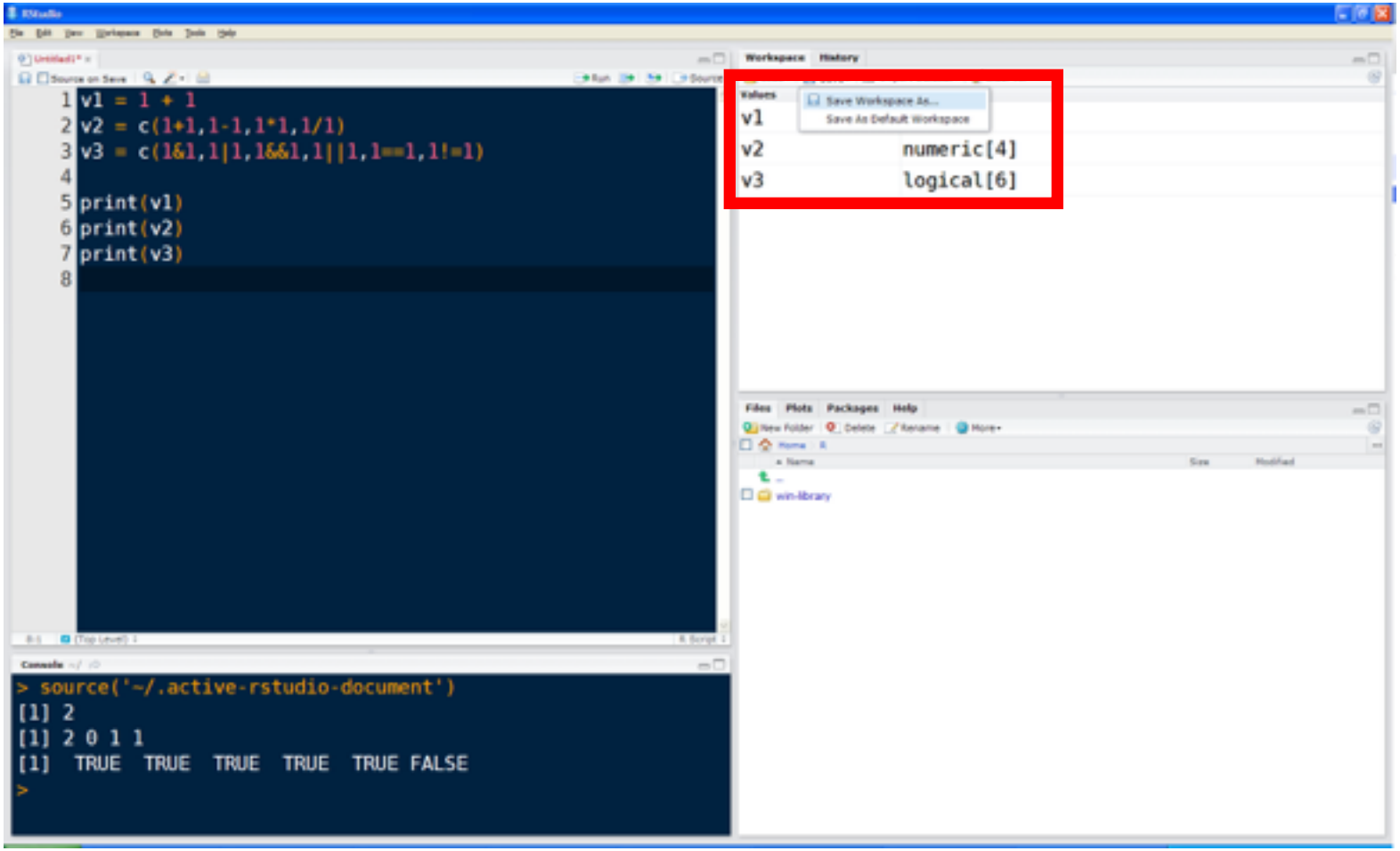


Make New R Script



Run Script Loaded in RStudio




v1 = 1 + 1
v2 = c(1+1,1-1,1*1,1/1)
v3 = c(1&1,1|1,1&&1,1||1,1==1,1!=1)
print(v1)
print(v2)
print(v3)
Workspace
History
Save Workspace As...
Save As Default Workspace
v1
v2 numeric[4]
v3 logical[6]
Files
Plots
Packages
Help
win-library
> source('~/.active-rstudio-document')
[1] 2
[1] 2 0 1 1
[1] TRUE TRUE TRUE TRUE TRUE FALSE
>

Saving Your Workspace

- In R you can save your entire workspace, variables and all in its current state
- Then you can reload this at a later time or can provide this to collaborators
- Workspace data files are saved with the extension '.Rdata'

Getting Help

- Inside of R
  - Simplify make a fake dataset
  - help(<function name>)
  - Help.start()
  - ?<function name>
  - ??<search term>
- On the web
  - www.rseek.org and R only search engine
  - CRAN
  - Google topic with CRAN

Creating Variables

- Named containers for data
- Names can be anything you like except for 'special' words
    - if else repeat while function for in next break
    - TRUE FALSE NULL Inf NaN NA NA_integer_ NA_real_ NA_complex_ NA_character_
- Better if names describe what the stored data is
  Creating / Setting variables is a matter of equality:

```
var = somedata
var <- somedata
```

Difference between = and <-

- The operators <- and = assign into the environment in which they are evaluated.
- The operator <- can be used anywhere, whereas the operator = is only allowed at the top level (e.g., in the complete expression typed at the command prompt) or as one of the subexpressions in a braced list of expressions.

```
matrix(1,nrow=2)

matrix(1,nrow<-2)
```

Data Types in R

- R's atomic data type is the **vector**
  - numeric
    - floating point
    - integer
  - logical
  - character
- **functions**
- **lists**
  - let you combine other data types

```
> a <- 101
> length(a)
[1] 1
> a[1]
[1] 101
```

```
> a[2]
[1] NA
> a[2] <- 202
> length(a)
[1] 2
```

R as calculator

```
> 3 + 9 + 12 – 7
[1] 17

> pi * 2^3 – sqrt(4)

> factorial(4)
```

Reading and Getting Data into R

Combine Command

- c(1, 2, 3, 4)
- c(item1, item2, item3, item4)
- c(“item1”, “item2”, “item3”)

Scan Command

- our.data = scan()

- scan(what = 'character')
- data5 = scan(sep = ',', what = 'char')
- data6 = scan(file = 'data.txt')

Working Directory

- getwd()

- setwd('pathname')

Reading Bigger Data Files

- read.csv()
- read.csv(file, sep = ',', header = TRUE, row.names)
- fw = read.csv(file.choose())

- my.ssv = read.table(file.choose(), header = TRUE)
- my.tsv = read.delim(file.choose())
- my.tsv = read.csv(file.choose(), sep = '\t’)
- my.tsv = read.table(file.choose(), header = TRUE, sep = '\t')

Convert between number and text data

> cut2 = as.character(cut)

➢cut3 = as.factor(cut2)

➢data7i = as.integer(data7)

➢data7n = as.numeric(data7i)

Data Frames

- 2 Dimensional Objects, it has rows and columns. R treats the columns as separate samples or variables, rows represent the replicates or observations.

```
> grass
  species   cut
1      12   mow
2      15   mow
3      17   mow
4      11   mow
5      15   mow
6       8 unmow
7       9 unmow
8       7 unmow
9       9 unmow
```

Matrix Objects

- A matrix is a two-dimensional data object. At first glance a matrix looks just like a data frame:

```
> bird
             Garden Hedgerow Parkland Pasture Woodland
Blackbird        47       10       40       2        2
Chaffinch        19        3        5       0        2
Great Tit        50        0       10       7        0
House Sparrow    46       16        8       4        0
Robin             9        3        0       0        2
Song Thrush       4        0        6       0        0
```

Structure of Objects

- str()
- To Examine the structure of an object

- class()
- Tell you the class of object

Saving Data Files to Disk

- save()

➢save(list, file = ‘filename’)

➢save(bf, bf.lm, bf.beta, file = ‘desktop/ butterly.rdata’)

➢save(list = ls(pattern =‘^bf’), file = ‘desktop/ butterfly.rdata’)

➢save(list = ls(all=TRUE), file=‘filename’)

➢save.image(file=‘filename’)

Reading Data Files from Disk

- load(file='filename.Rdata')
- load(file=file.choose())

Save Data to disk as text files

- write(x, file="data", sep='.')

Save Data Frame or Matrix

- write.table(mydata, file='filename', row.names=TRUE, sep=' ', col.names=TRUE)

Selecting and displaying parts of a vector

COMMAND	RESULT
`data1[1]`	Shows the first item in the vector.
`data1[3]`	Shows the third item.
`data1[1:3]`	Shows the first to the third items.
`data1[-1]`	Shows all except the first item.
`data1[c(1, 3, 4, 8)]`	Shows the items listed in the `c()` part.
`data1[data1 > 3]`	Shows all items greater than 3.
`data1[data1  5 | data1 > 7]`	Shows items less than 5 or greater than 7.

Sorting and rearranging a vector

- sort()
- sort(unmow, decreasing =TRUE)

Get an Index using order()

- order(unmow)

Logical Values from a vector

- data1 = c(3, 5, 7, 3, 2, 6)
- which(data1 == 6)

- Quicklooks
- head(mf)
- head(mf, n = 3)

Give simple Stat

- summary(bird.m)

Rotating Data Tables

- fw.t = t(fw)

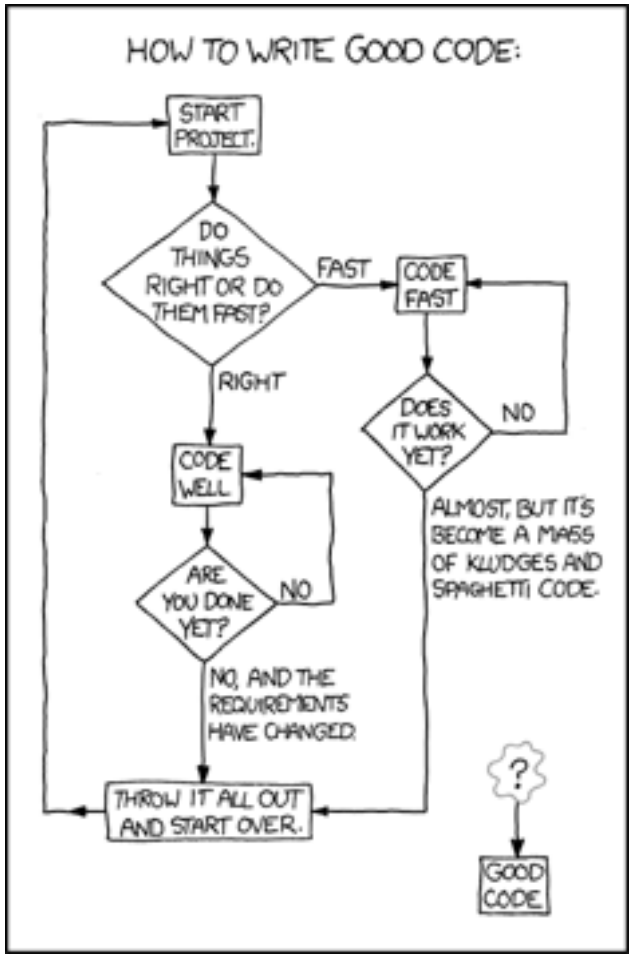
Making Data Frames

- My.frame = data.frame(item1, item2, item3)

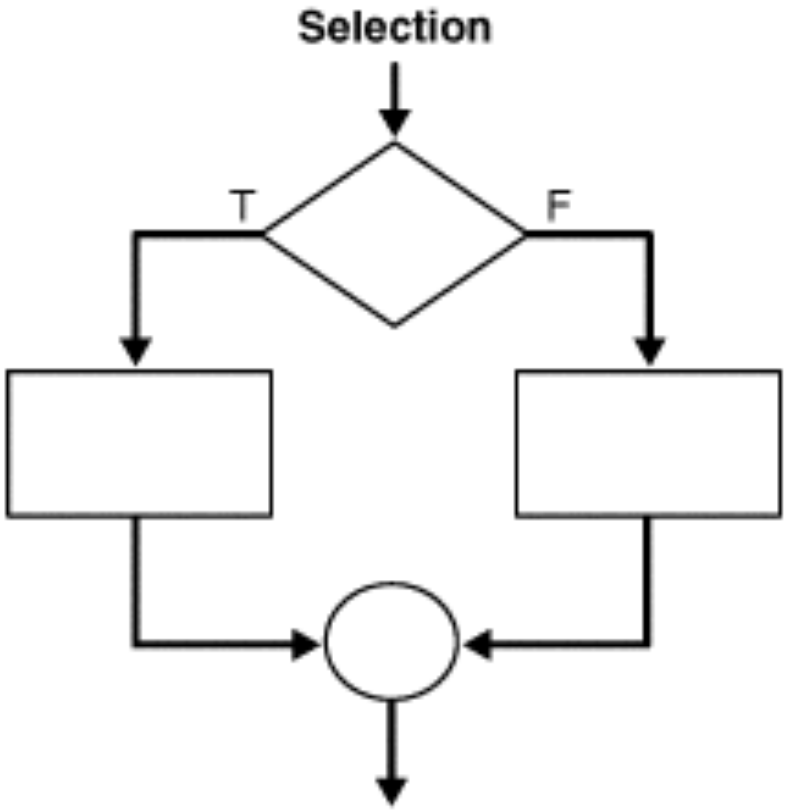
Making Matrix Objects

- cmat = cbind(sampl1, sampl2)

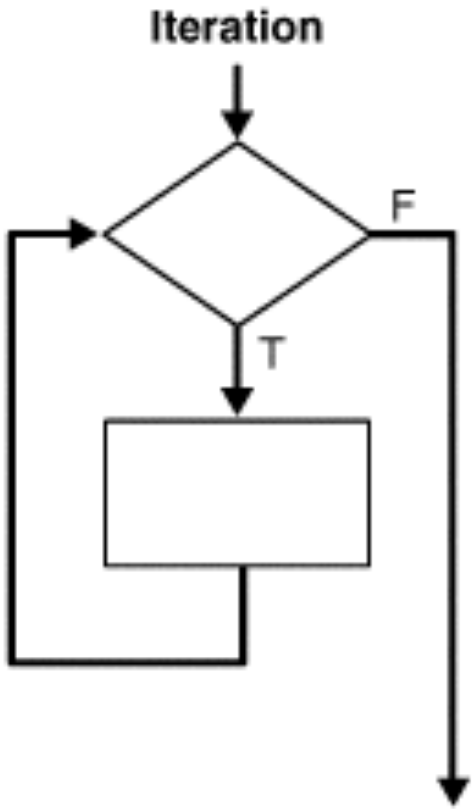
Coding


HOW TO WRITE GOOD CODE:
START PROJECT.
DO THINGS RIGHT OR DO THEM FAST?
FAST
CODE FAST
RIGHT
DOES IT WORK YET?
NO
CODE WELL
ALMOST, BUT IT'S BECOME A MASS OF KLUDGES AND SPAGHETTI CODE.
ARE YOU DONE YET?
NO
NO, AND THE REQUIREMENTS HAVE CHANGED.
?
THROW IT ALL OUT AND START OVER.
GOOD CODE

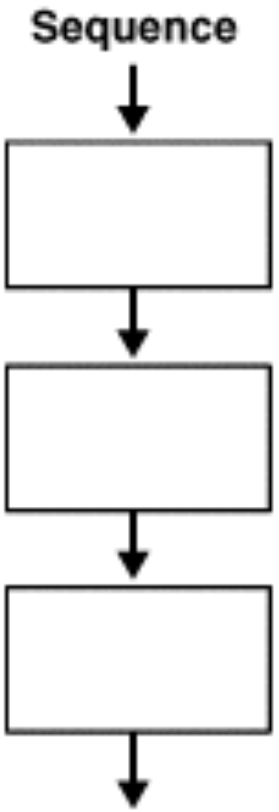
Programmatic Structure



if ... then ... else



for ...
while ...



For Loop

- Repeats a line or lines (known as a block) of code until:
  - (for loop) a count limit is reached

```
for (i in 1:10) {
  ... code ...
}
```

- the above loop runs 10 times
- '{' and '}' enclose code looped
- the variable i updated in the loop to values in the sequence 1:10

A Simple For Loop

```
# P-values from our analysis
p.values = c(0.1, 0.05, 0.003, 0.4, 0.9)

# A vector to store the negative log p-values
neglog10.p.values = 1:5

# Transform the p-values
for(p in 1:length(p.values)) {
  neglog10.p.values[p] = -log10(p.values[p])
}
```

While Loop

- Repeats a line or lines (known as a block) of code until:
  - (while loop) a logical condition is reached

```
while (stop != TRUE) {
   ... code ...
}
```

- the above loop runs until code sets
- stop = TRUE

- **warning:** if not properly written while loops can run infinitely

A Simple While Loop

```
# Numbers from our analysis
v1 = c(21, 22, 53, 74, 85, 96, 97, 58, 49, 30, 85)

# Iterator
i = 1

# Look for first instance of 85
while(v1[i] != 85) {
      i = i + 1
}

# Print out where we found it
print(paste('v1[',i,'] = 85',sep=''))
```

Functions

- Bits of code that do one thing and (preferrably) do it well
- Functions break up your code into more manageable and reusable parts
- Defining (e.g. in a script):

```
fun = function(arguments) {
    ... code ...
}
```

- Calling:

```
party = fun(food,beer,folks)
```

A Simple Function

```
fn1 <- function(N) {
    for(i in as.numeric(1:N)) {
        y <- i*i
    }
}

fn2 <- function(N) {
    i = 1
    while(i <= N) {
        y <- i*i
        i <- i + 1
    }
}

system.time(fn1(60000))
system.time(fn2(60000))
```

Basic Plots

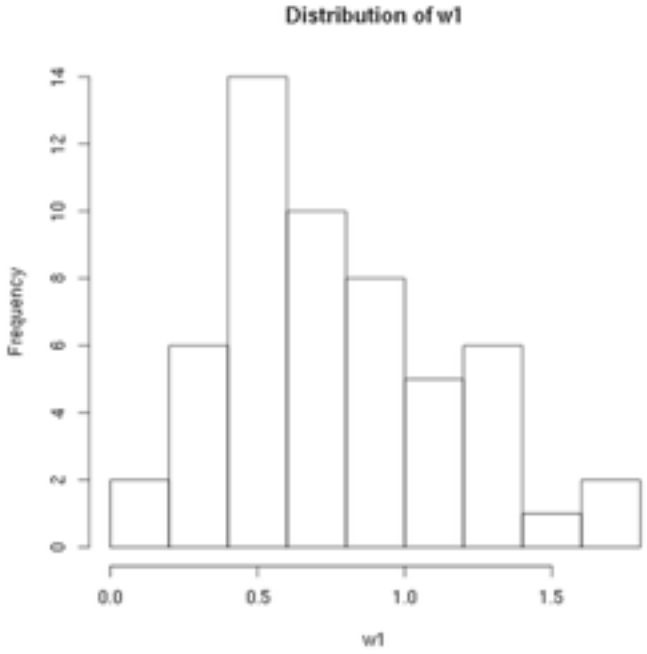
```
>  w1 <- read.csv(file="w1.dat",sep=",",head=TRUE)
> names(w1)
[1] "vals"
>  tree <- read.csv(file="trees91.csv",sep=",",head=TRUE)
> names(tree)
 [1] "C"      "N"      "CHBR"   "REP"    "LFBM"   "STBM"   "RTBM"   "LFNCC"
 [9] "STNCC"  "RTNCC"  "LFBCC"  "STBCC"  "RTBCC"  "LFCACC" "STCACC" "RTCACC"
[17] "LFKCC"  "STKCC"  "RTKCC"  "LFMGCC" "STMGCC" "RTMGCC" "LFPCC"  "STPCC"
[25] "RTPCC"  "LFSCC"  "STSCC"  "RTSCC"
```

```
> stripchart(w1$vals,vertical=TRUE)
> stripchart(w1$vals,vertical=TRUE,method="jitter")
```

```
> stripchart(w1$vals,method="stack",
             main='Leaf BioMass in High CO2 Environment',
             xlab='BioMass of Leaves')
```

Histogram

```
> hist(w1$vals)
> hist(w1$vals,main="Distribution of w1",xlab="w1")
```




```
> hist(w1$vals,
       main='Leaf BioMass in High CO2 Environment',
       xlab='BioMass of Leaves')
```

```
> title('Leaf BioMass in High CO2 Environment',xlab='BioMass of Leaves')
```

Boxplot

```
> boxplot(w1$vals)
```

```
> boxplot(w1$vals,
          main='Leaf BioMass in High CO2 Environment',
          ylab='BioMass of Leaves')
```

```
> boxplot(w1$vals,
          main='Leaf BioMass in High CO2 Environment',
          xlab='BioMass of Leaves',
          horizontal=TRUE)
```

Scatter Plot

```
> plot(tree$STBM,tree$LFBM)
```

```
> cor(tree$STBM,tree$LFBM)
[1] 0.911595
```

```
> plot(tree$STBM,tree$LFBM,
       main="Relationship Between Stem and Leaf Biomass",
       xlab="Stem Biomass",
       ylab="Leaf Biomass")
```

Vignettes

- Some packages have vignettes
  - A vignettes is an example of how to run the code and a lot of additional text explaining a lot more that you may want to know

- List all available vignettes:

```
vignette()
```

- Display the vignette as a pdf by executing

```
vignette('<topic>')
```

- To play with the vignette code

```
vig = vignette('<topic>')
edit(vig)
```

More tutorial

- http://swirlstats.com/
- http://cran.r-project.org/doc/manuals/R-intro.html