



# Introduction to R

วิชา การค้นพบองค์ความรู้และการทำเหมืองข้อมูลขั้นสูง

Veerasak Kritsanapraphan

Chulalongkorn University

Email : [veerasak.kr568@cbs.chula.ac.th](mailto:veerasak.kr568@cbs.chula.ac.th)



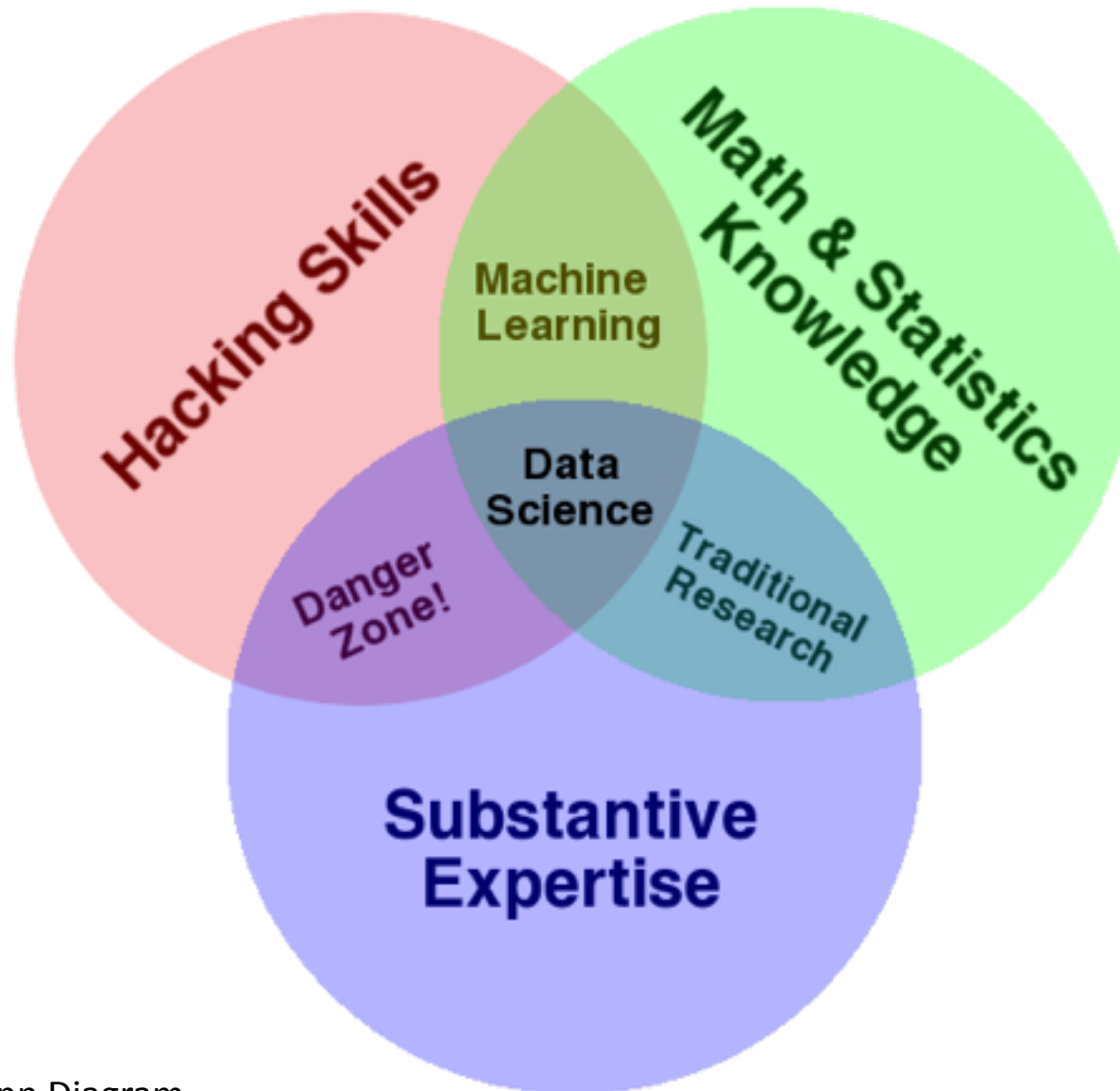
@veerasakk

# Slide and Sample Data

[https://github.com/vkrit/chula\\_datamining](https://github.com/vkrit/chula_datamining)



# The beginner's perspective



# 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 **Ross 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

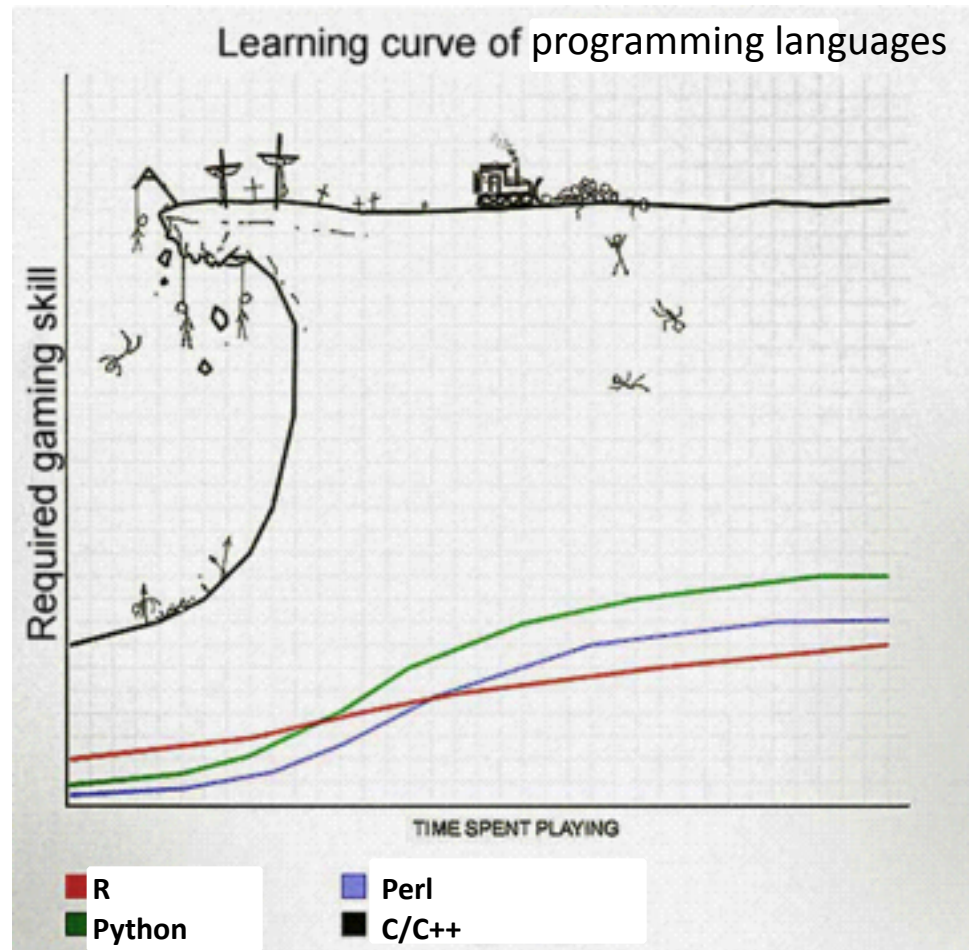
# What is R?

- 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

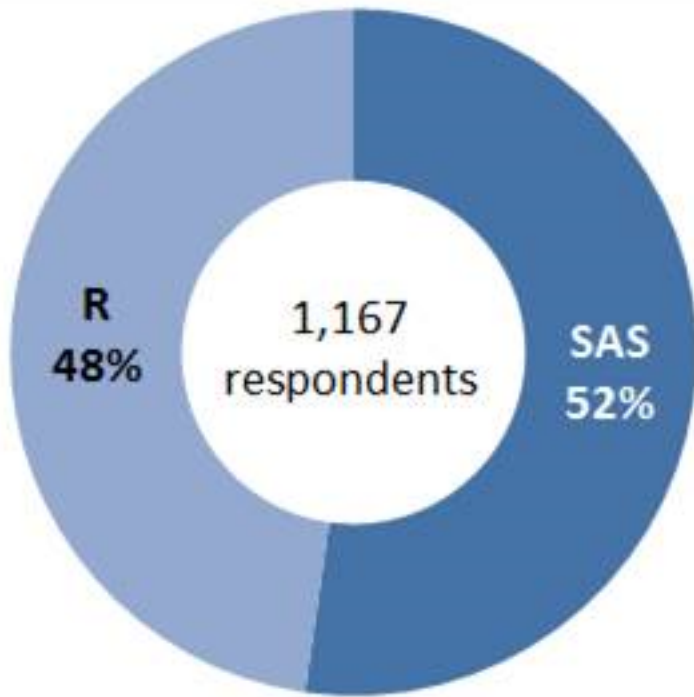
# Why use R?

- 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

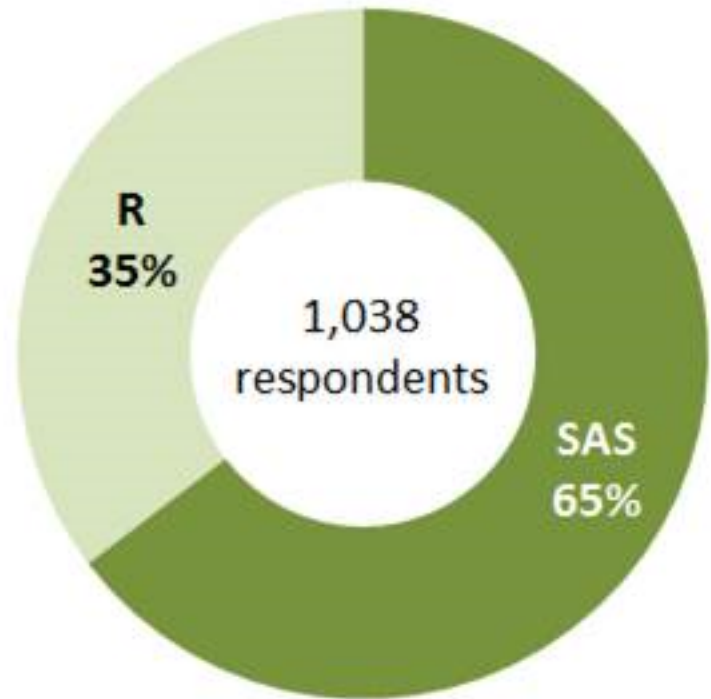
# Why use R?



# R vs SAS



**2015**



**2014**



# Tools



**R**

[www.r-project.org](http://www.r-project.org)

The engine\*



**RStudio**

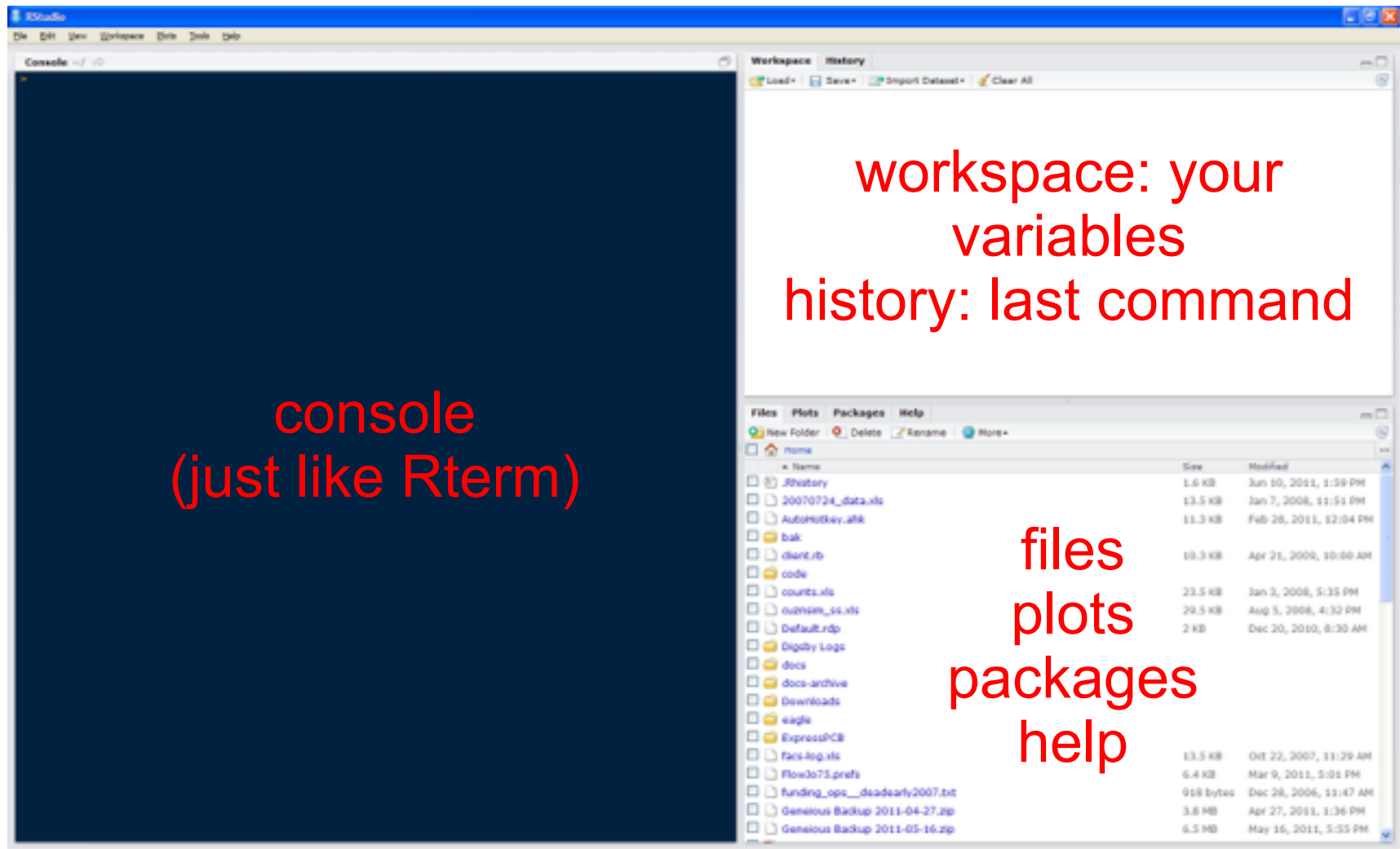
[www.rstudio.org](http://www.rstudio.org)

The pretty face\*\*

*\* Many alternatives exist. Smallest learning curve.*

*\*\* A few alternatives exist. This happens to be the easiest at the moment.*

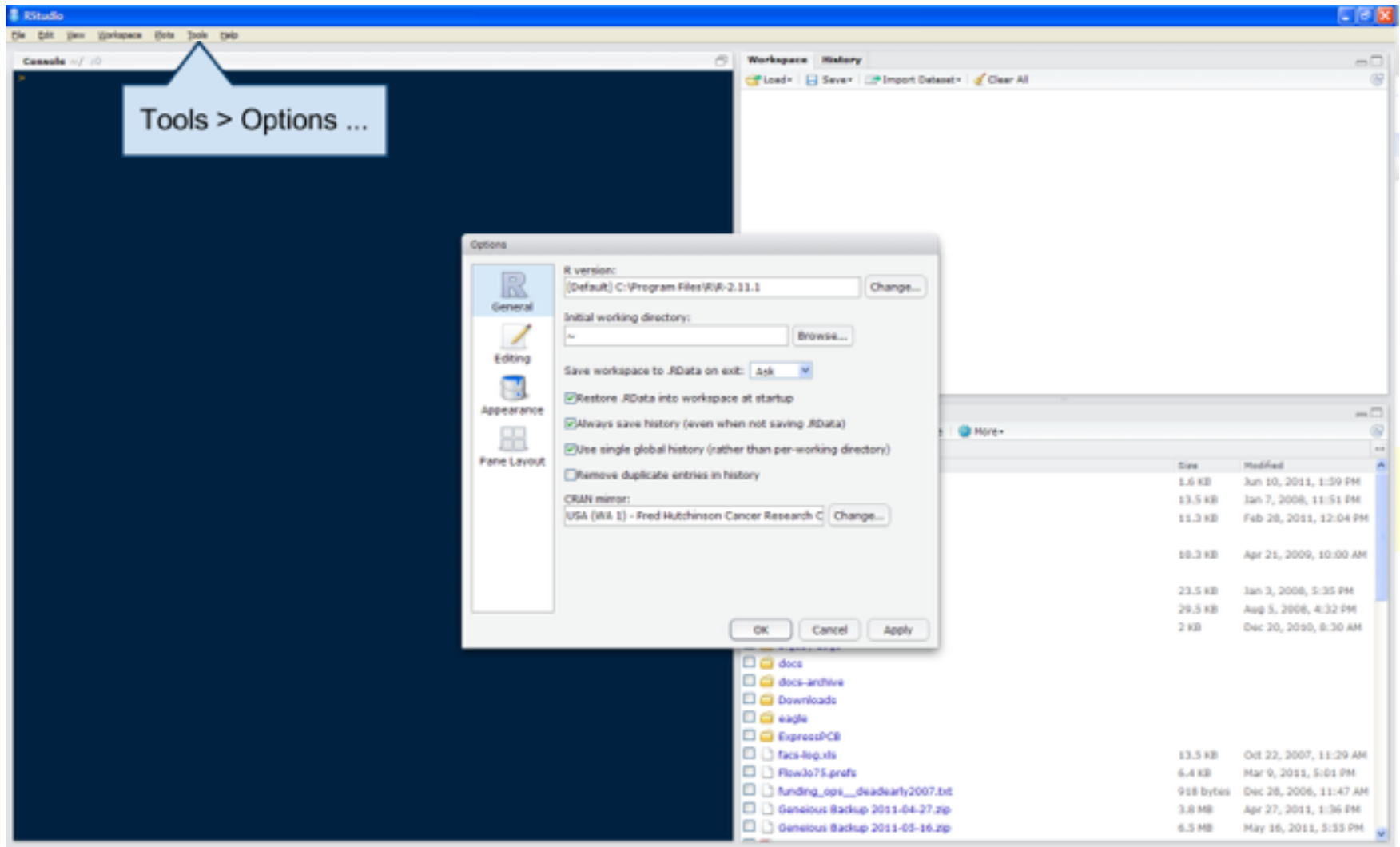
# Introducing RStudio



# RStudio Features

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

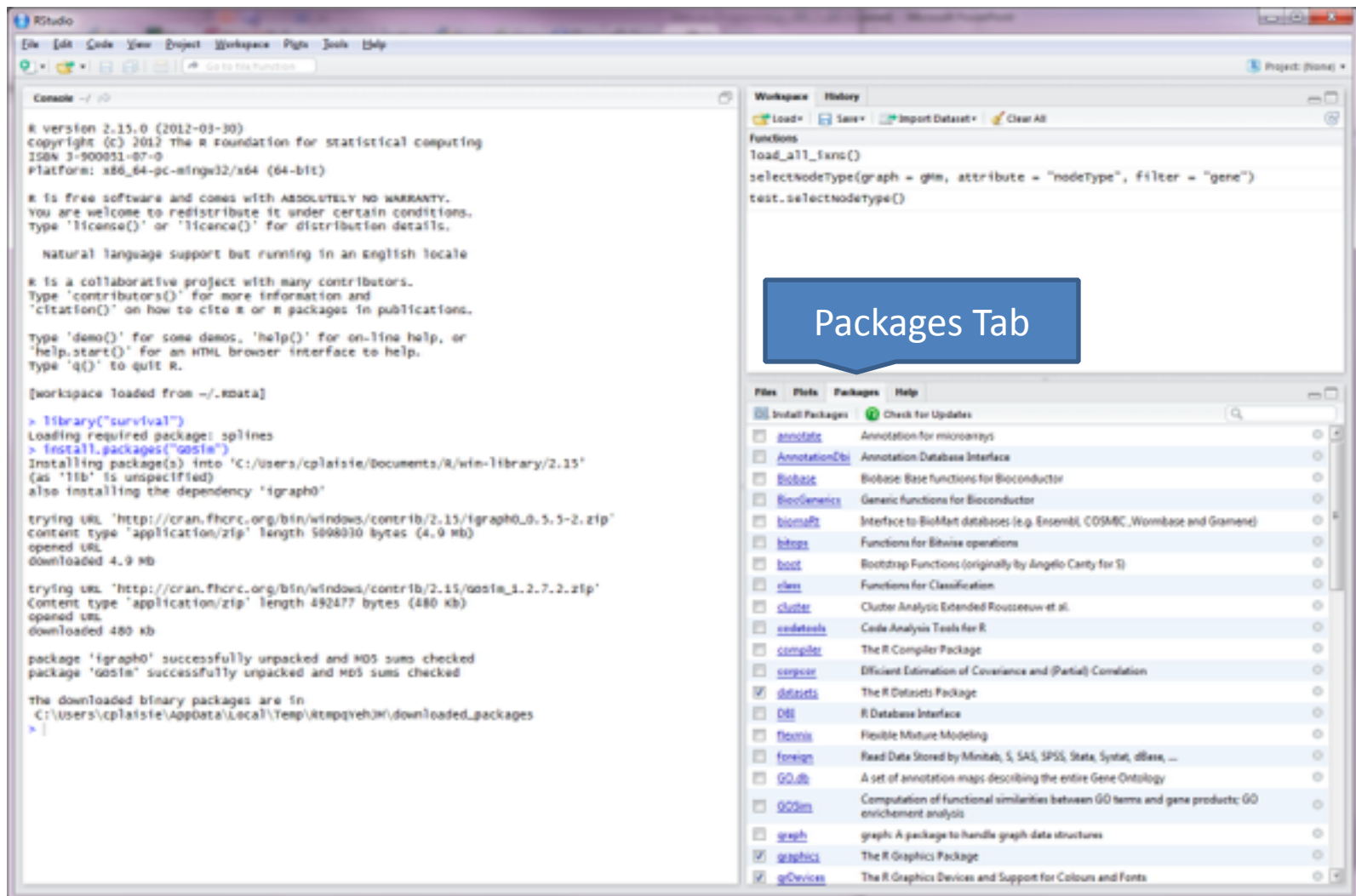
- Repositories host the packages
  - CRAN, CRANextra, BioCsoft, BioCann, BioCexp, BioCext, Omegahat, R-Forge and rforge.net
- Use this code to set your repositories

```
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



The screenshot displays the RStudio interface with the Packages Tab selected. The console on the left shows the following output:

```
R version 2.15.0 (2012-03-30)
copyright (c) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-87-9
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.

[workspace loaded from ~/.Rdata]

> library("survival")
Loading required package: splines
> install.packages("igraph")
Installing package(s) into 'C:/Users/cplaisie/Documents/R/win-library/2.15'
(as 'lib' is unspecified)
also installing the dependency 'igraph0'

trying URL 'http://cran.fhcrc.org/bin/windows/contrib/2.15/igraph0.0.5.5-2.zip'
content type 'application/zip' length 5098930 bytes (4.9 mb)
opened URL
downloaded 4.9 MB

trying URL 'http://cran.fhcrc.org/bin/windows/contrib/2.15/igraph0.0.5.5-2.zip'
Content type 'application/zip' length 492477 bytes (480 kb)
opened URL
downloaded 480 kb

package 'igraph0' successfully unpacked and MD5 sums checked
package 'igraph' successfully unpacked and MD5 sums checked

the downloaded binary packages are in
C:/Users/cplaisie/AppData/Local/Temp/Rtmpqveh0W/downloaded_packages
>
```

The Packages Tab on the right shows a list of installed and available packages. A blue callout box labeled "Packages Tab" points to this section. The list includes:

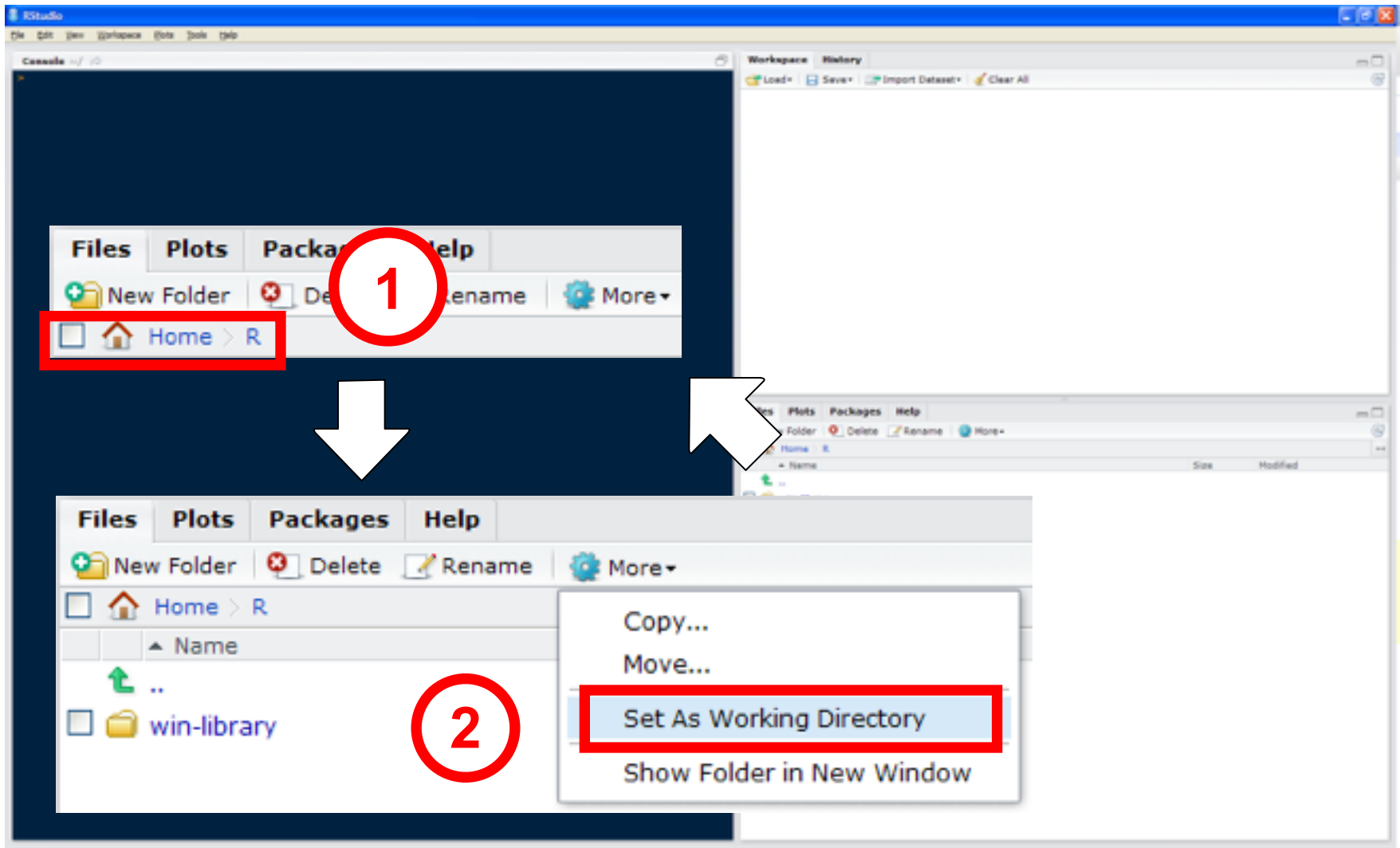
Package	Description
annotate	Annotation for microarrays
AnnotationDbi	Annotation Database Interface
Biobase	Biobase: Base functions for Bioconductor
BiocGenerics	Generic functions for Bioconductor
biomaRt	Interface to BioMart databases (e.g. Ensembl, COSMIC, Wormbase and Genemid)
bitops	Functions for bitwise operations
boot	Bootstrap Functions (originally by Angelo Canty for S)
class	Functions for Classification
cluster	Cluster Analysis Extended Rousseeuw et al.
codetools	Code Analysis Tools for R
compiler	The R Compiler Package
corpcor	Efficient Estimation of Covariance and (Partial) Correlation
datasets	The R Datasets Package
DBI	R Database Interface
flexmix	Flexible Mixture Modeling
foreign	Read Data Stored by Minitab, S, SAS, SPSS, Stata, Syntat, dBase, ...
GO.db	A set of annotation maps describing the entire Gene Ontology
GOsim	Computation of functional similarities between GO terms and gene products; GO enrichment analysis
igraph	igraph: A package to handle graph data structures
graphics	The R Graphics Package
grDevices	The R Graphics Devices and Support for Colours and Fonts



# 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

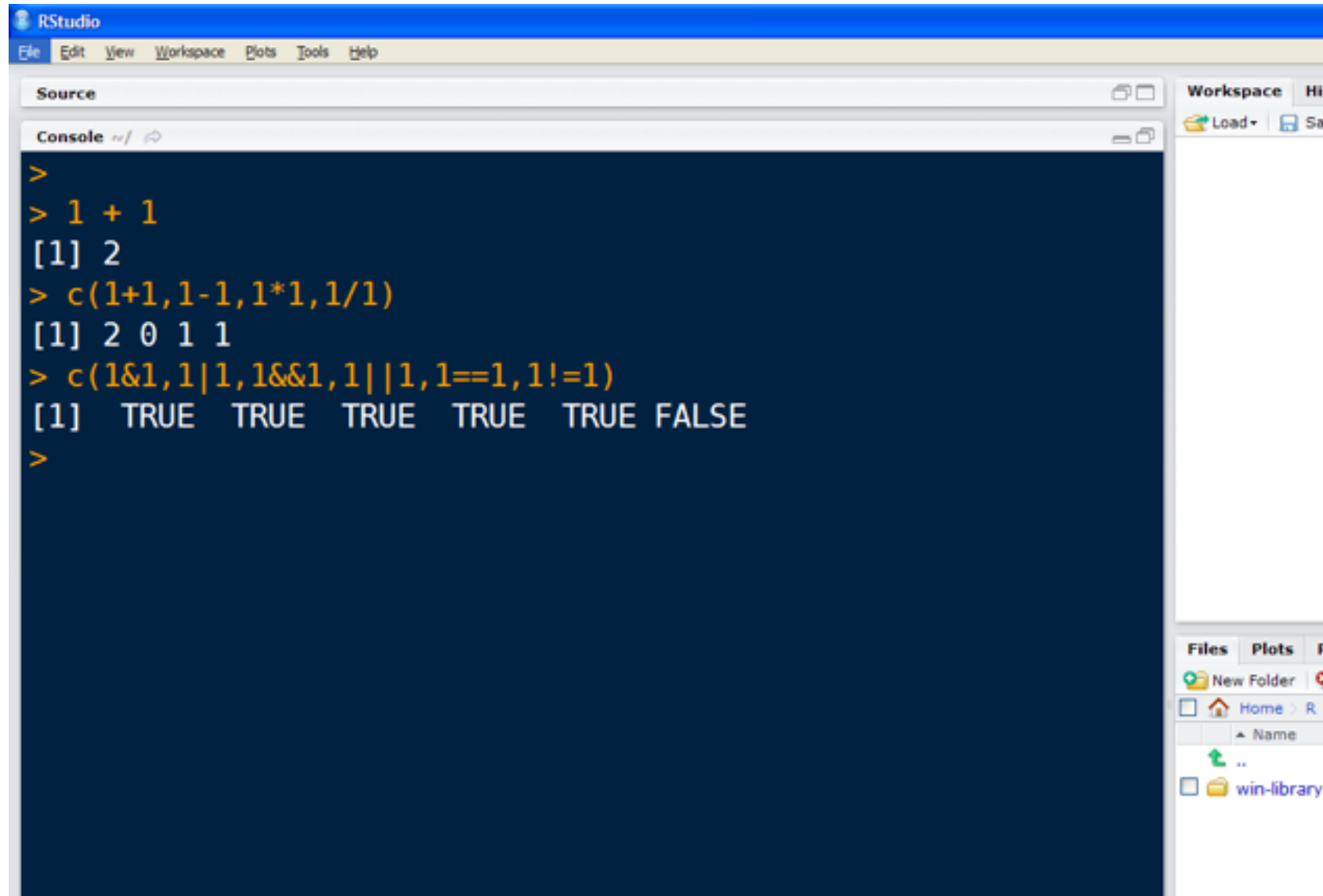


# Ready to Code!

- 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



The screenshot shows the RStudio application window. The top menu bar includes File, Edit, View, Workspace, Plots, Tools, and Help. The main window is divided into three panes: Source (top), Console (middle), and Workspace (right). The Console pane shows the following R code and output:

```
>  
> 1 + 1  
[1] 2  
> c(1+1, 1-1, 1*1, 1/1)  
[1] 2 0 1 1  
> c(1&1, 1|1, 1&&1, 1||1, 1==1, 1!=1)  
[1] TRUE TRUE TRUE TRUE TRUE FALSE  
>
```

The Workspace pane on the right shows a 'Load' button and a 'Save' button. The Files pane at the bottom right shows a 'New Folder' button and a list of files including 'Home', 'R', and 'win-library'.

# Basic Math / Basic Logic

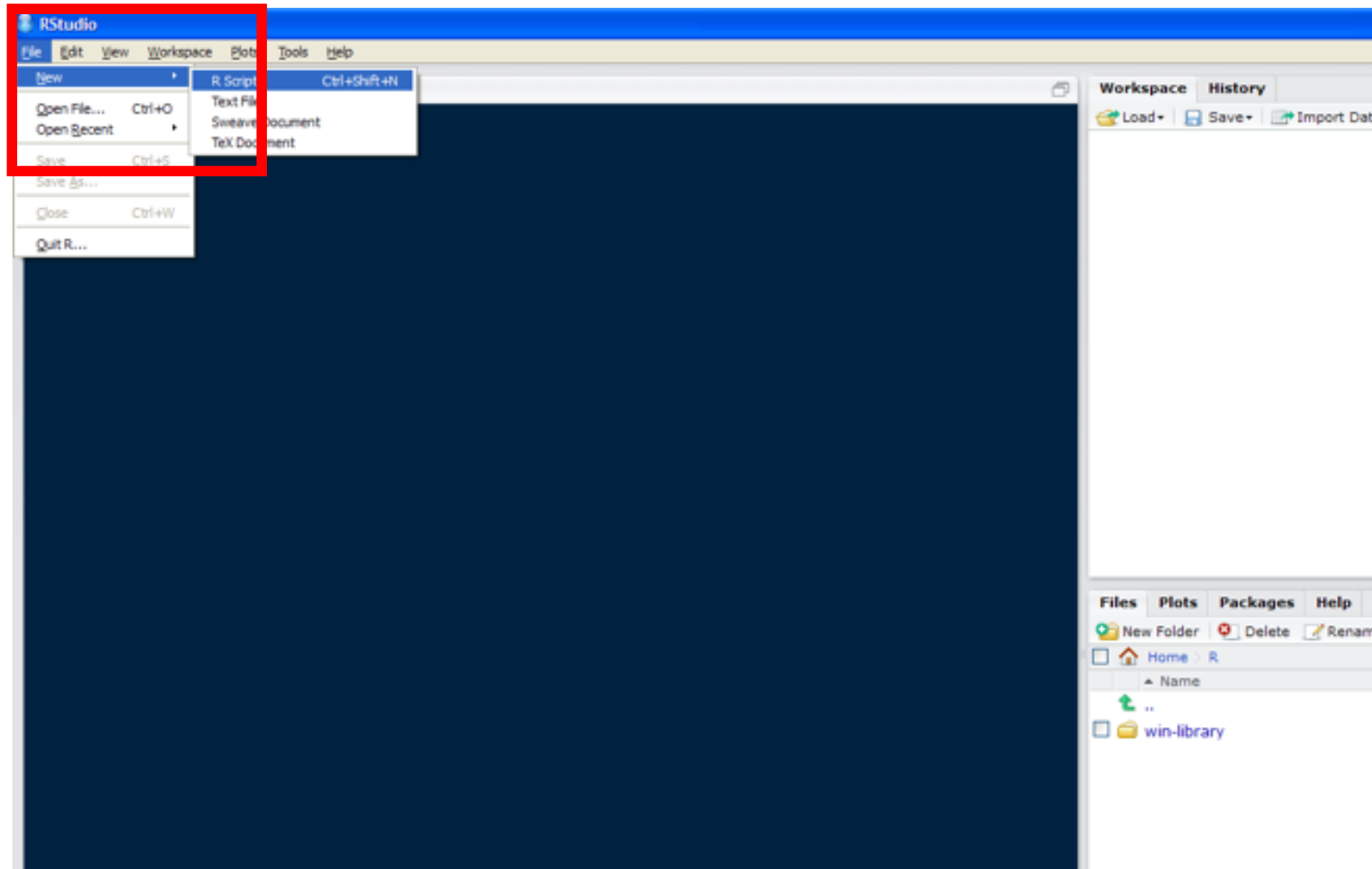
- + addition
- - subtraction
- \* multiplication
- / division
- % modulus (remainder)
- ^ to the power
- ! NOT
- & bitwise AND
- | bitwise OR
- && short circuit AND
- || short circuit OR
- == equality
- != NOT equality

?Arithmetic

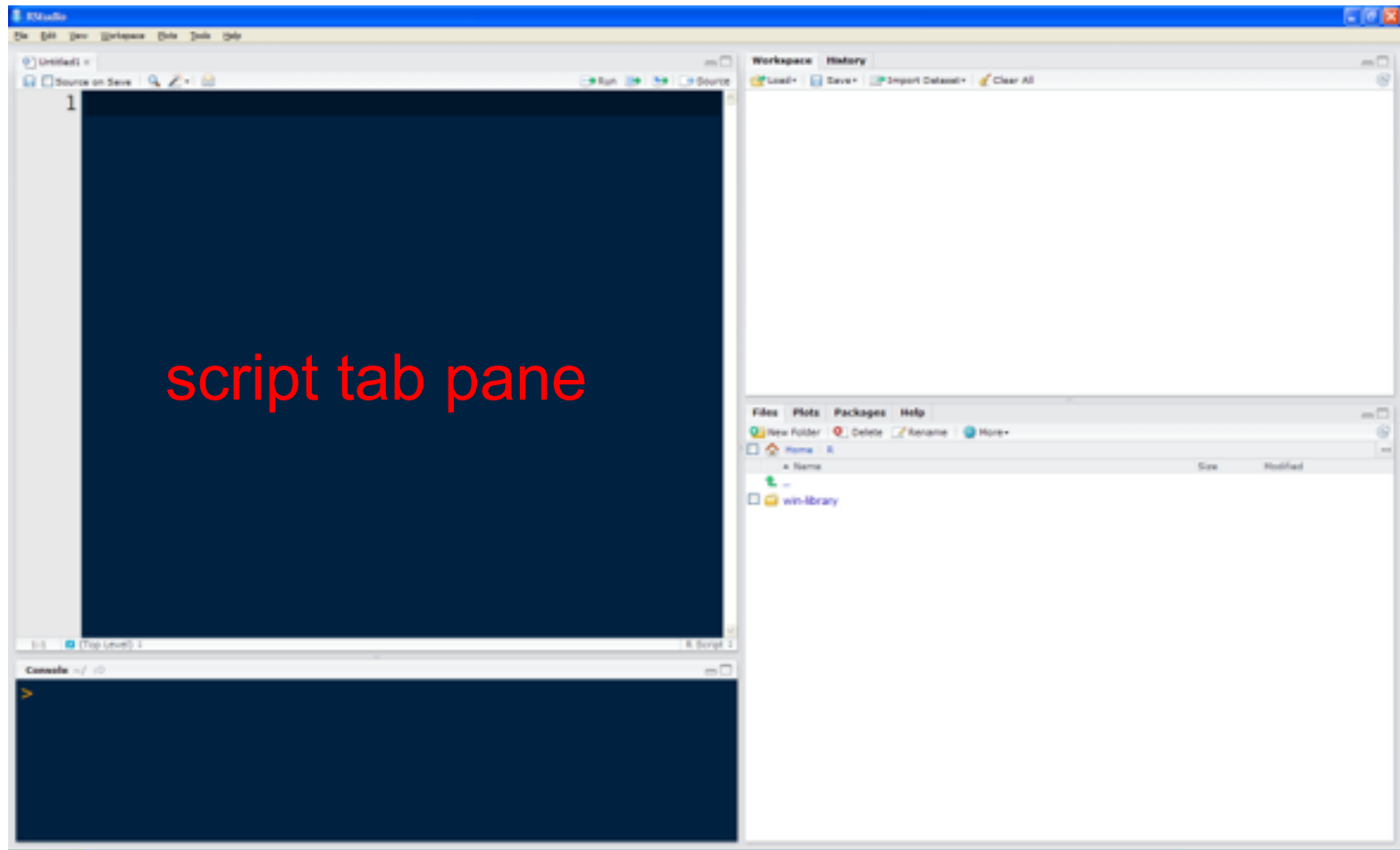
?Logic

Also try ?Syntax, ?Comparison

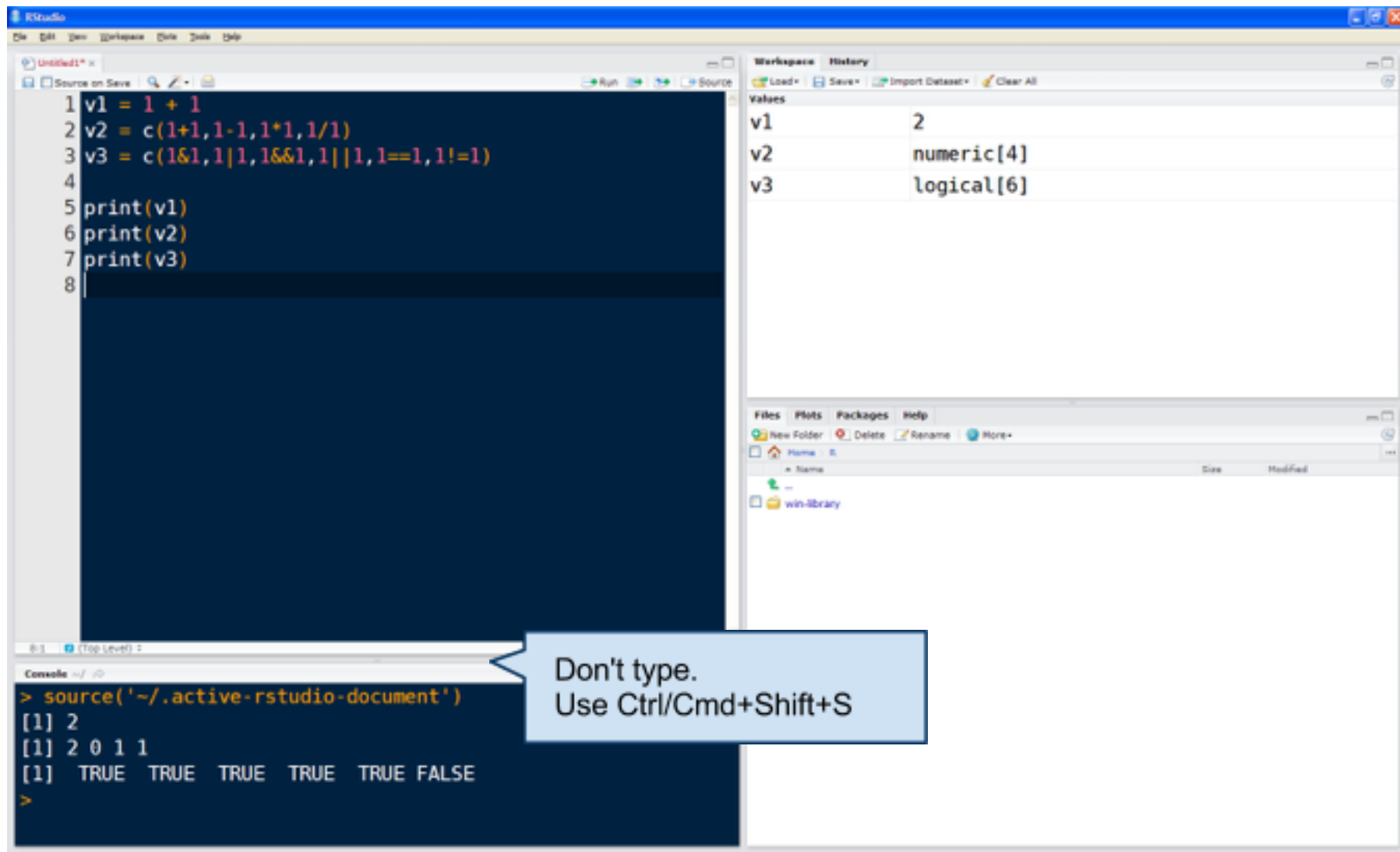
# Make New R Script



# Make New R Script



# Run Script Loaded in RStudio



The screenshot shows the RStudio interface with a script editor on the left, a workspace/history pane on the right, and a console at the bottom. The script in the editor contains the following R code:

```
1 v1 = 1 + 1
2 v2 = c(1+1, 1-1, 1*1, 1/1)
3 v3 = c(1&1, 1|1, 1&&1, 1||1, 1==1, 1!=1)
4
5 print(v1)
6 print(v2)
7 print(v3)
8
```

The workspace/history pane on the right shows the following values:

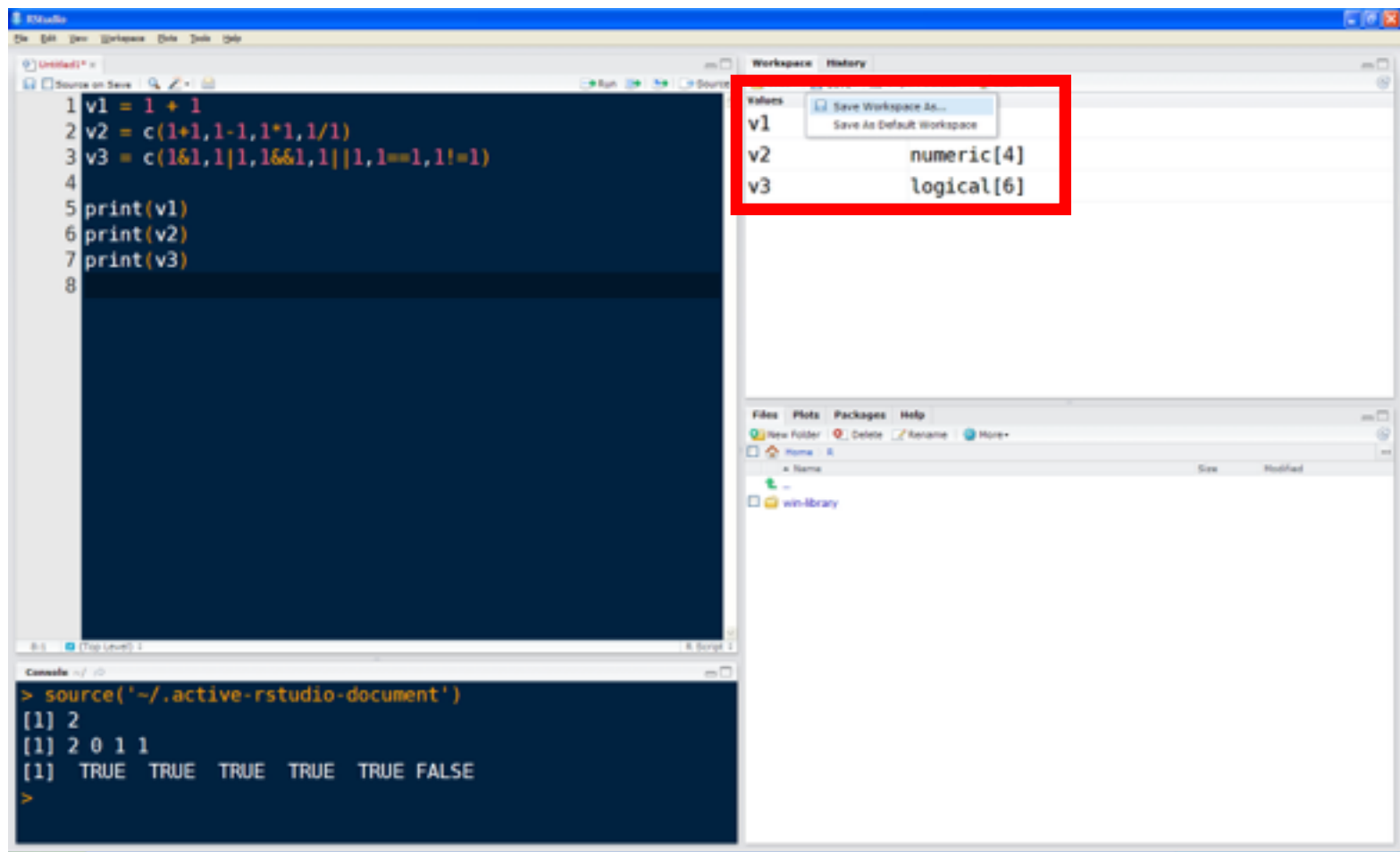
Variable	Value
v1	2
v2	numeric[4]
v3	logical[6]

The console at the bottom shows the output of the script after running it:

```
> source('~/.active-rstudio-document')
[1] 2
[1] 2 0 1 1
[1] TRUE TRUE TRUE TRUE TRUE FALSE
>
```

A callout box points to the console with the text: "Don't type. Use Ctrl/Cmd+Shift+S".





# 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](http://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/butterfly.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
<code>data1[1]</code>	Shows the first item in the vector.
<code>data1[3]</code>	Shows the third item.
<code>data1[1:3]</code>	Shows the first to the third items.
<code>data1[-1]</code>	Shows all except the first item.
<code>data1[c(1, 3, 4, 8)]</code>	Shows the items listed in the <code>c()</code> part.
<code>data1[data1 &gt; 3]</code>	Shows all items greater than 3.
<code>data1[data1 &lt; 5   data1 &gt; 7]</code>	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)`

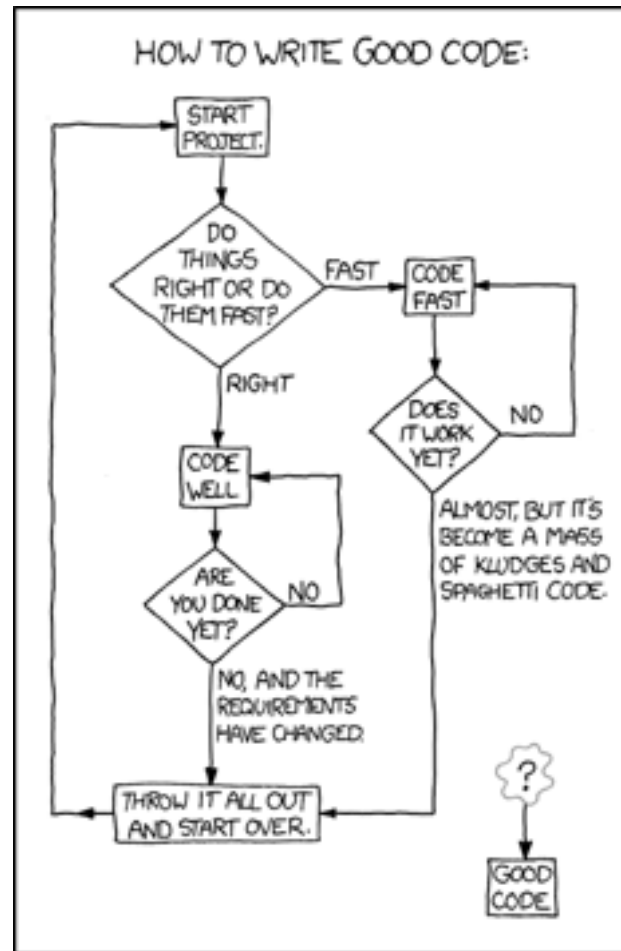
## Making Data Frames

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

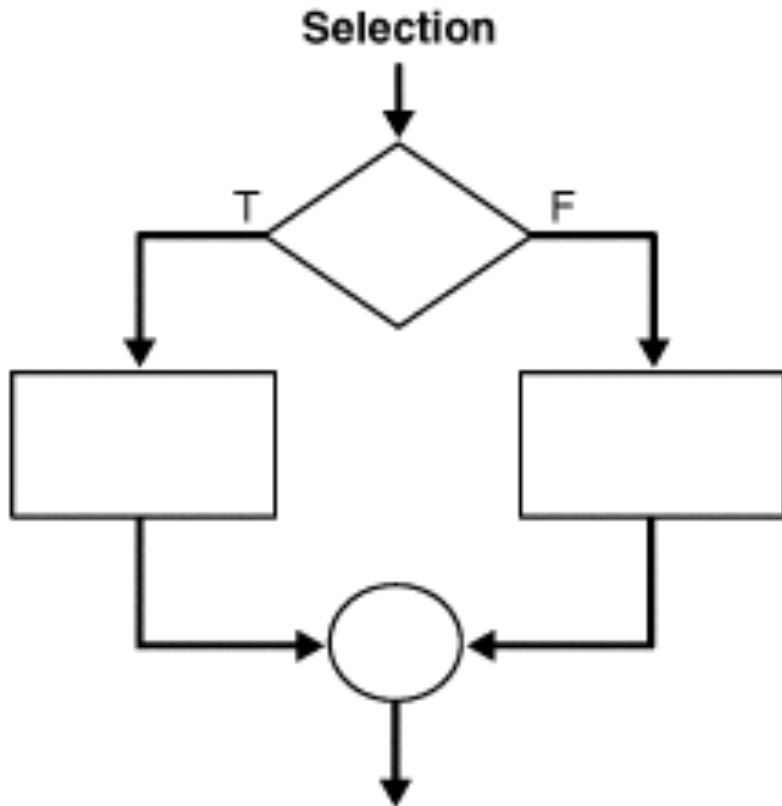
## Making Matrix Objects

- `cmat = cbind(sampl1, sampl2)`

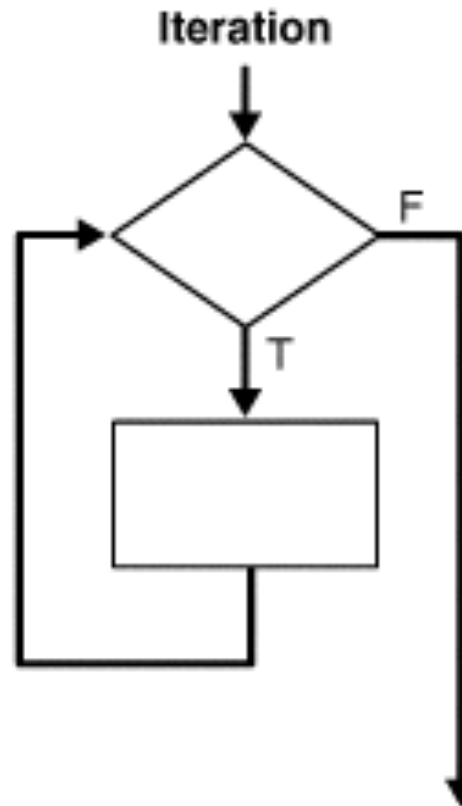
# Coding



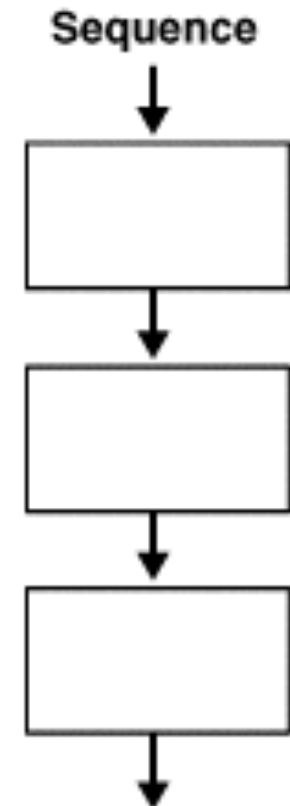
# Programmatic Structure



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 (preferably) 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

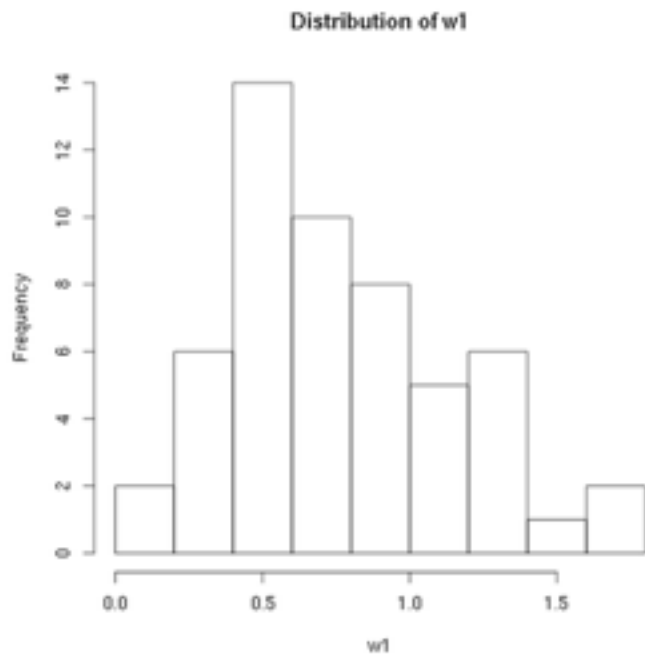
```
> w1 <- read.csv(file="w1.dat",sep=",",head=TRUE)
> names(w1)
[1] "vals"
> tree <- read.csv(file="trees91.csv",sep=",",head=TRUE)
> names(tree)
[1] "C"      "N"      "CHBR"   "REP"    "LFBM"   "STBM"   "RTBM"   "LFNCC"
[9] "STNCC"  "RTNCC"  "LFBCC"  "STBCC"  "RTBCC"  "LFCACC" "STCACC" "RTCACC"
[17] "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 vignette 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>