Fun with

Bharat Bhushan Verma

https://www.linkedin.com/in/dearbharat/

dearbharat@gmail.com

+1 902 418 6649

Agenda

- Module 1. Getting Started
- What is R
- Install R and RStudio / Anaconda
- Explore Environment
- Variables

- Module 2. Data Types
- Numbers
- Text
- Vector
- Matrix
- Array
- Data Frame
- Factor
- List

Agenda

- Module 3. Packages & Data Sets
- Packages
- Data Sets

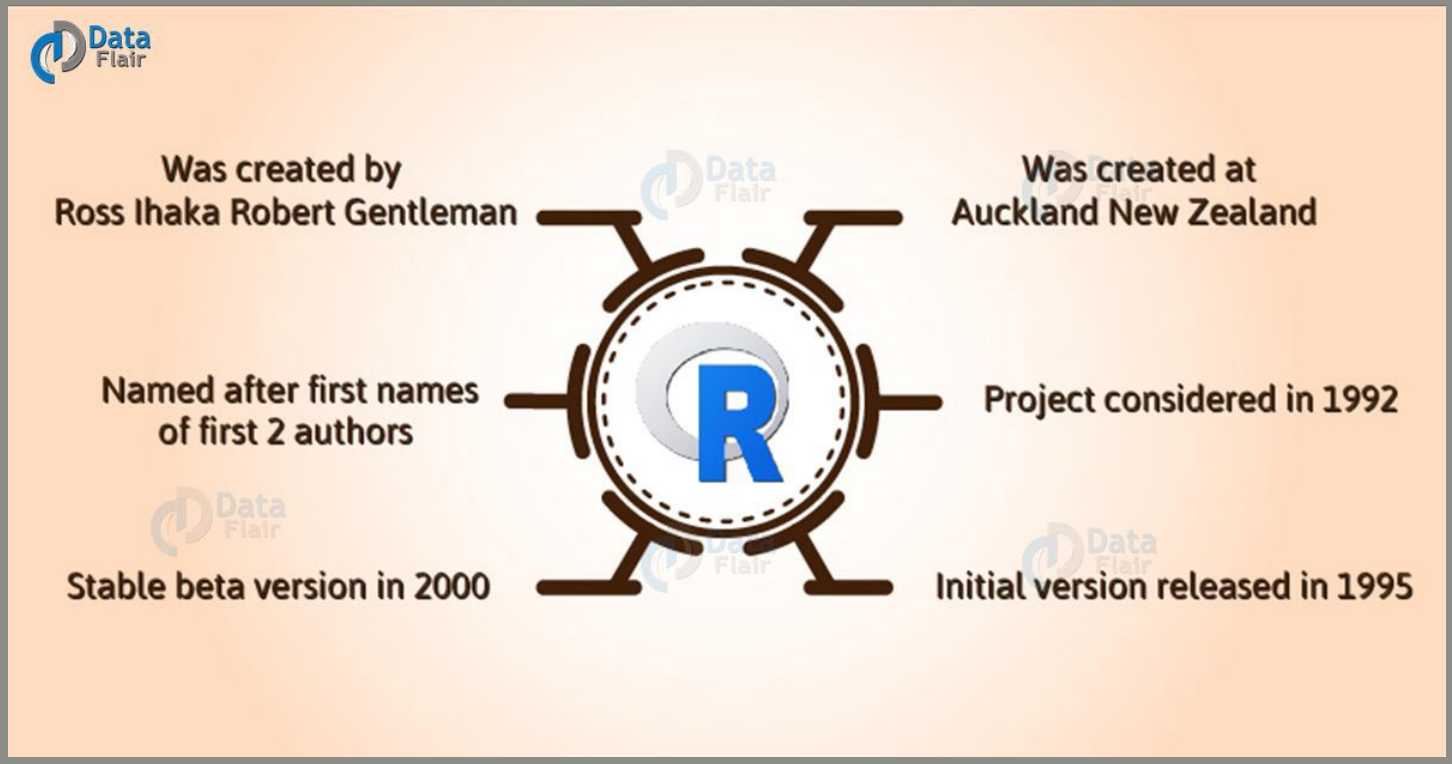
- Module 4. Control Structures (Self Study)
- Conditional
- Loop
- Break & Next
- Operators

- Module 5: Function (Self Study)
- Function Syntax
- Function Example
- Function With Default Arguments

GETTING STARTED

Module 1

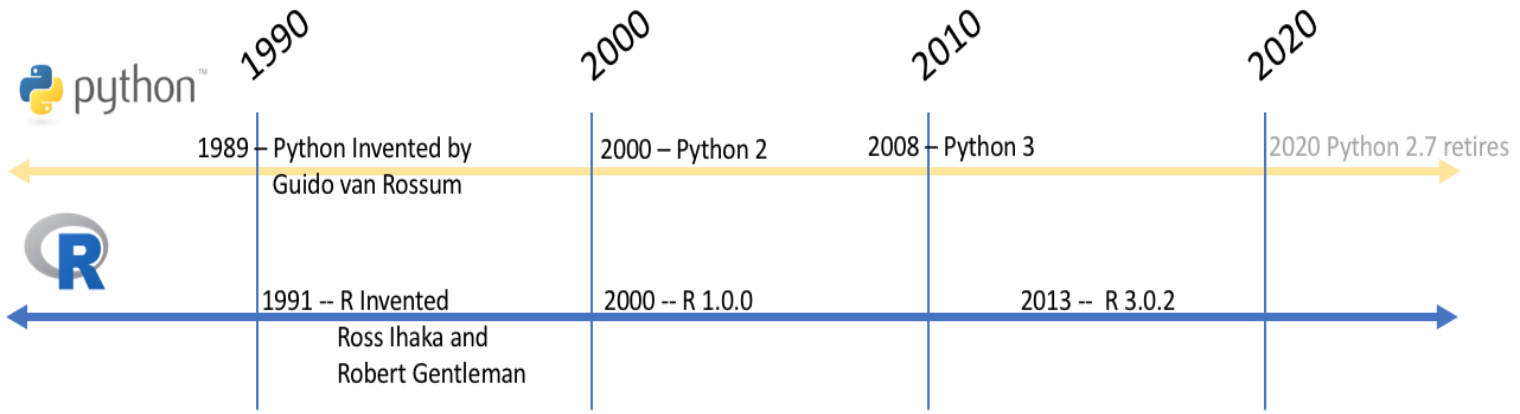
What is R?

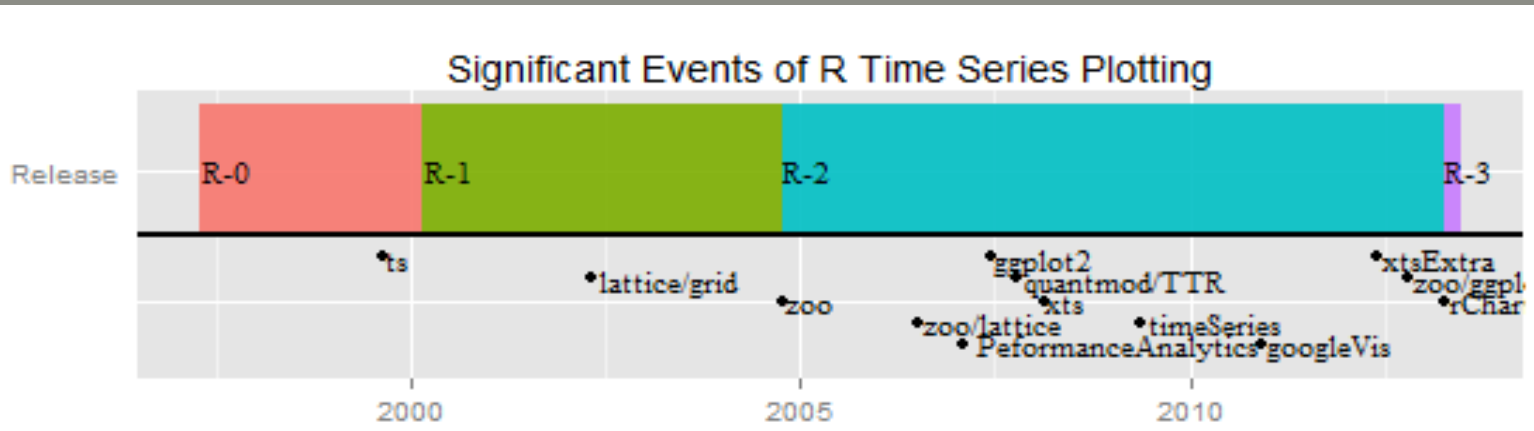

Data Flair
Was created by
Ross Ihaka Robert Gentleman
Was created at
Auckland New Zealand
Named after first names
of first 2 authors
R
Project considered in 1992
Stable beta version in 2000
Initial version released in 1995

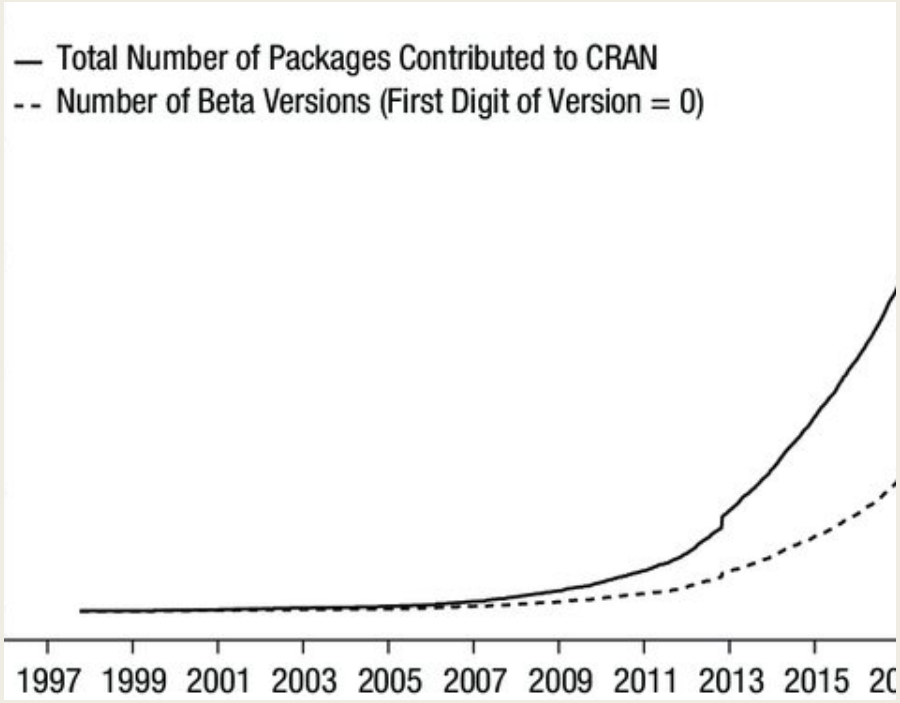
Why R?

- R is open source and free!
- R is vector-based
- R is a programming environment suited for statistics
- R can be integrated with other software

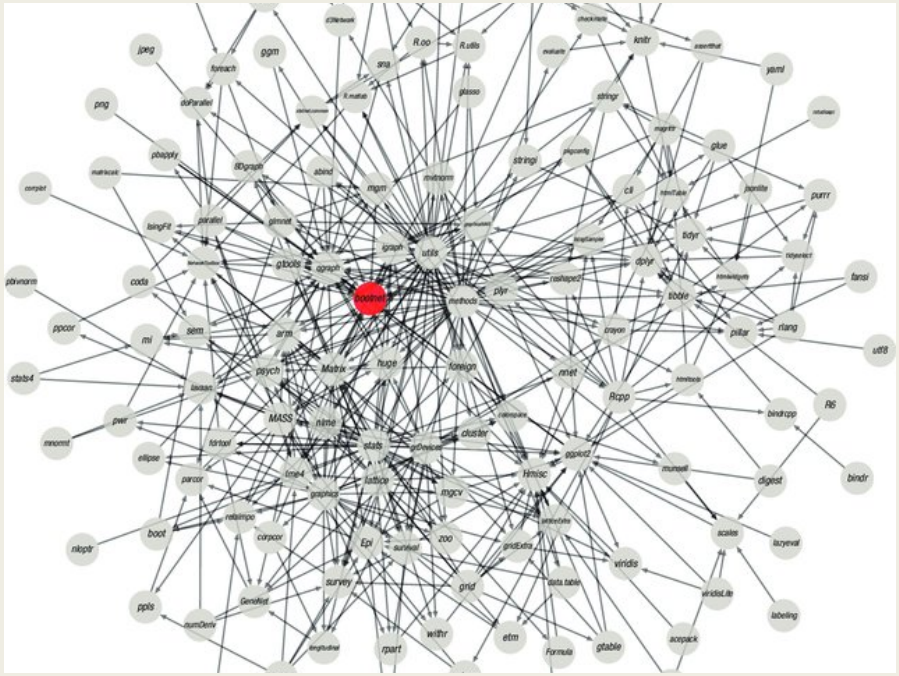
R Development History







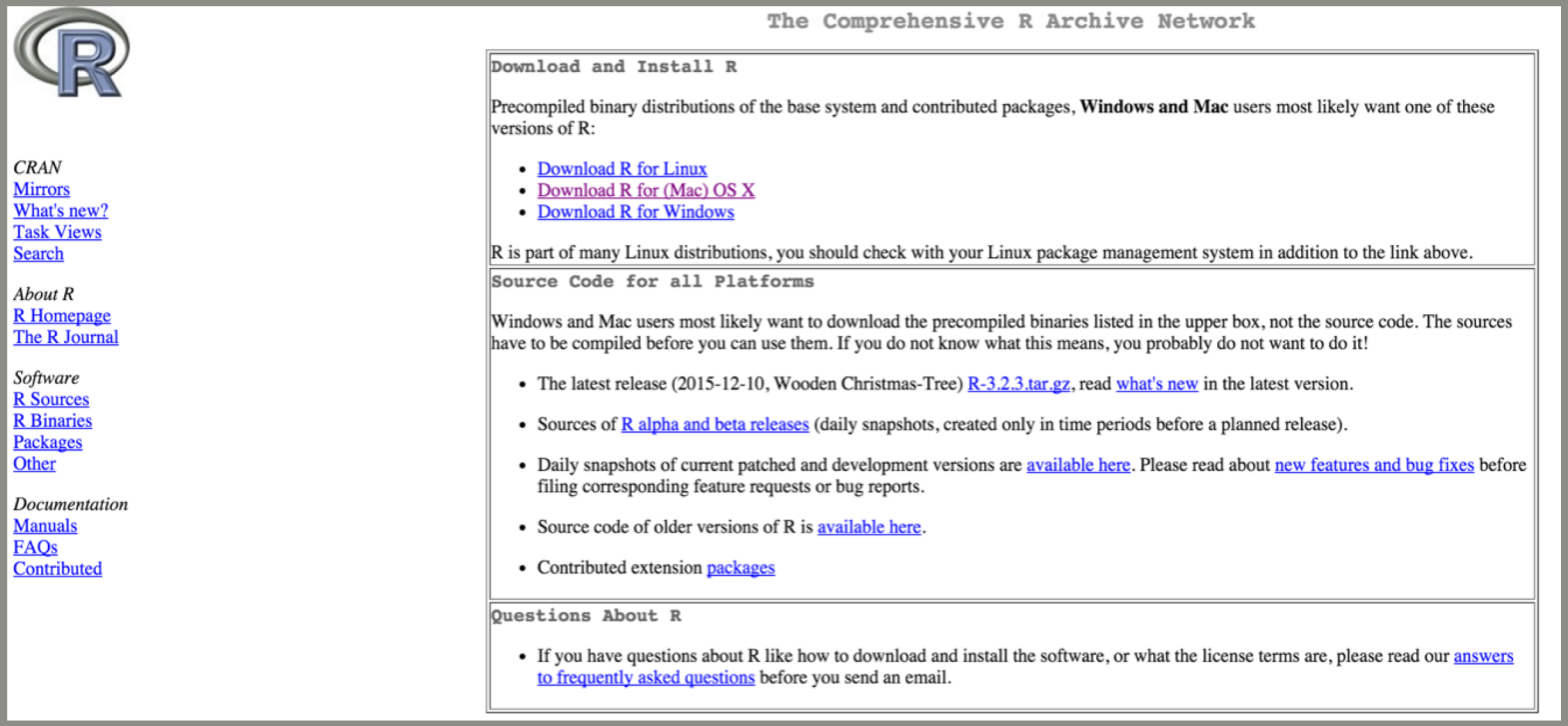
NUMBER OF PACKAGES IN R



PACKAGE DEPENDENCY IN R

Where to get R from?

http://cran.stat.nus.edu.sg/



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 (Mac) OS 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!

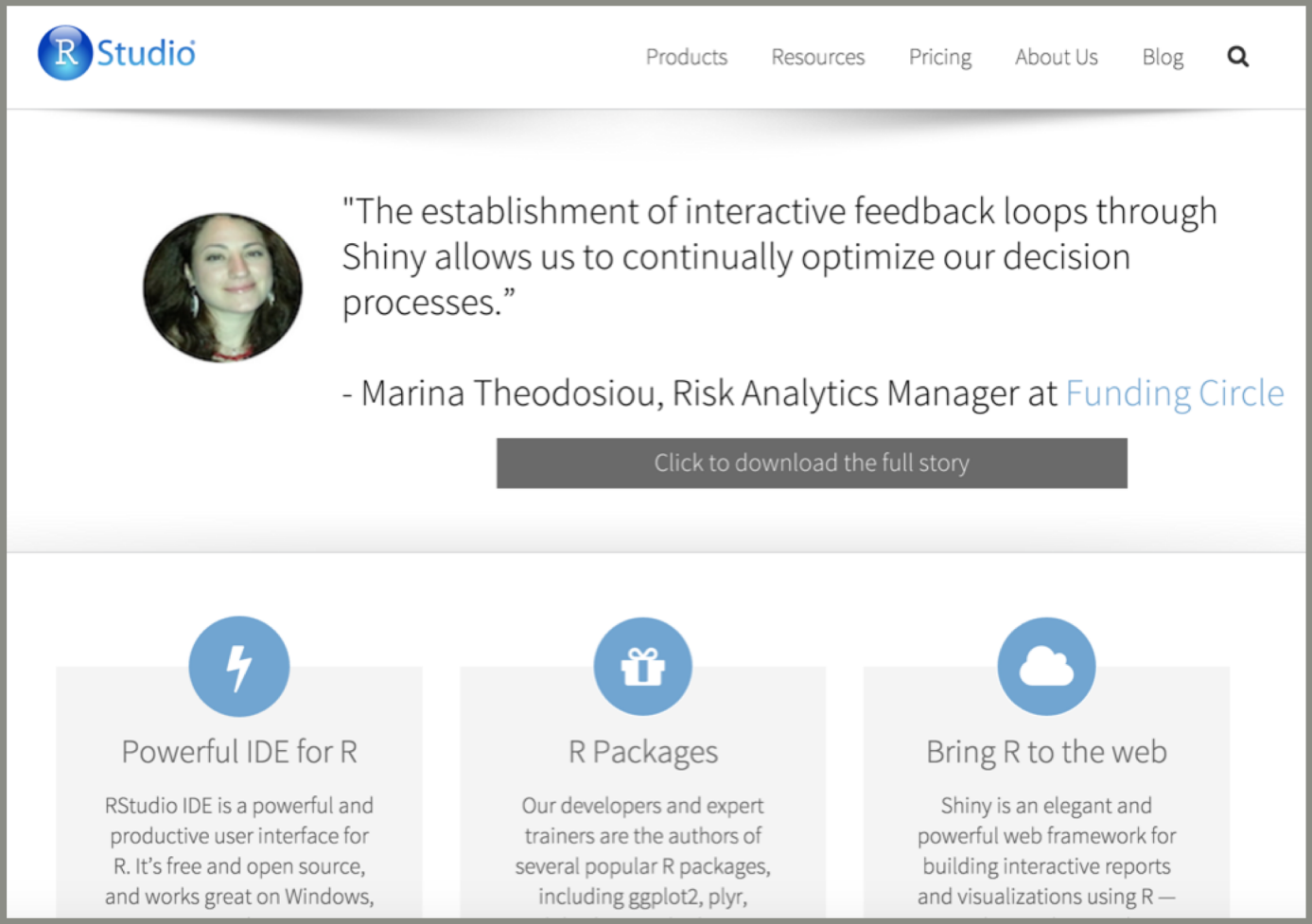
- The latest release (2015-12-10, Wooden Christmas-Tree) R-3.2.3.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available 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

Questions About R

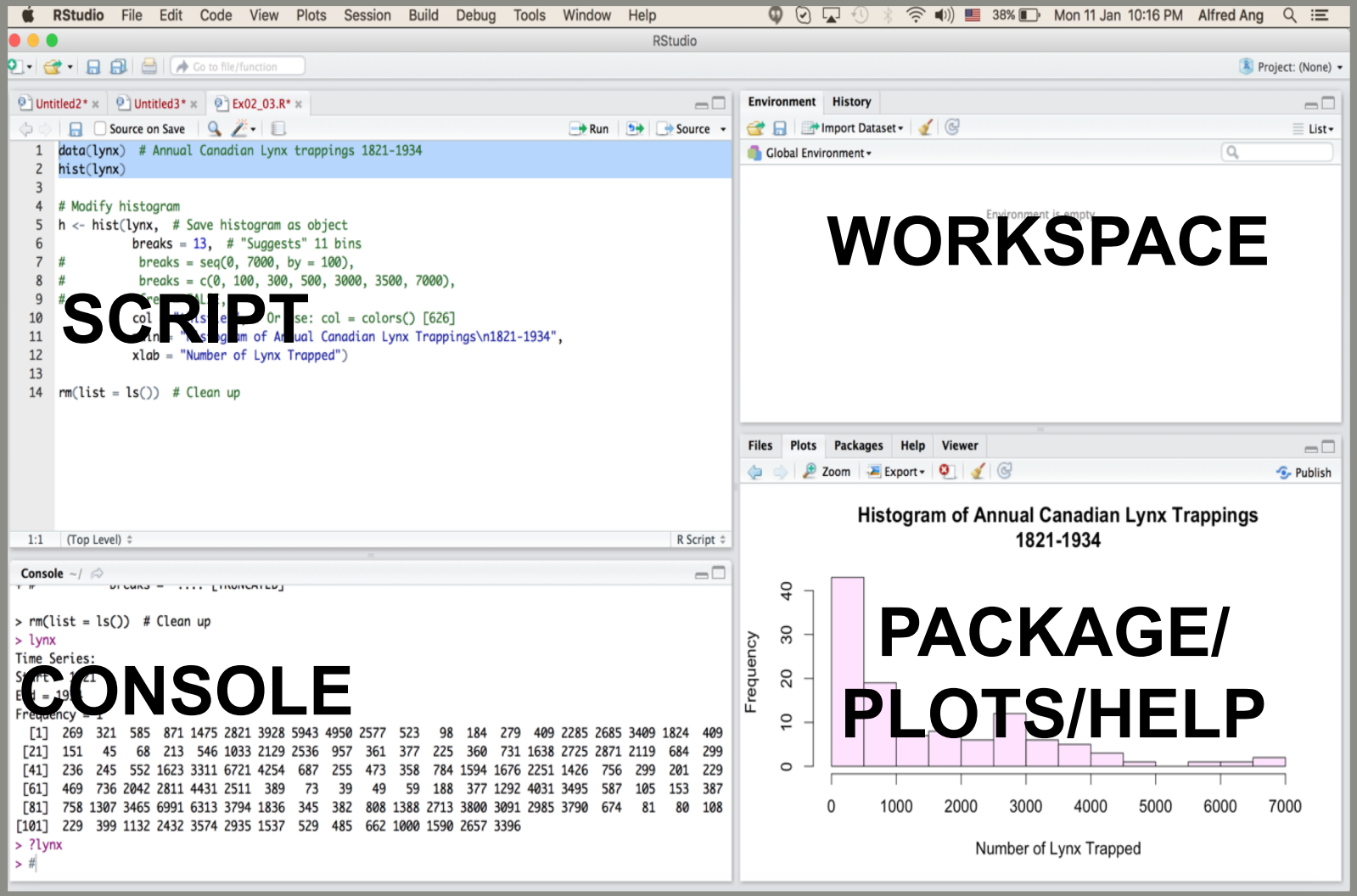
- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

R Studio IDE

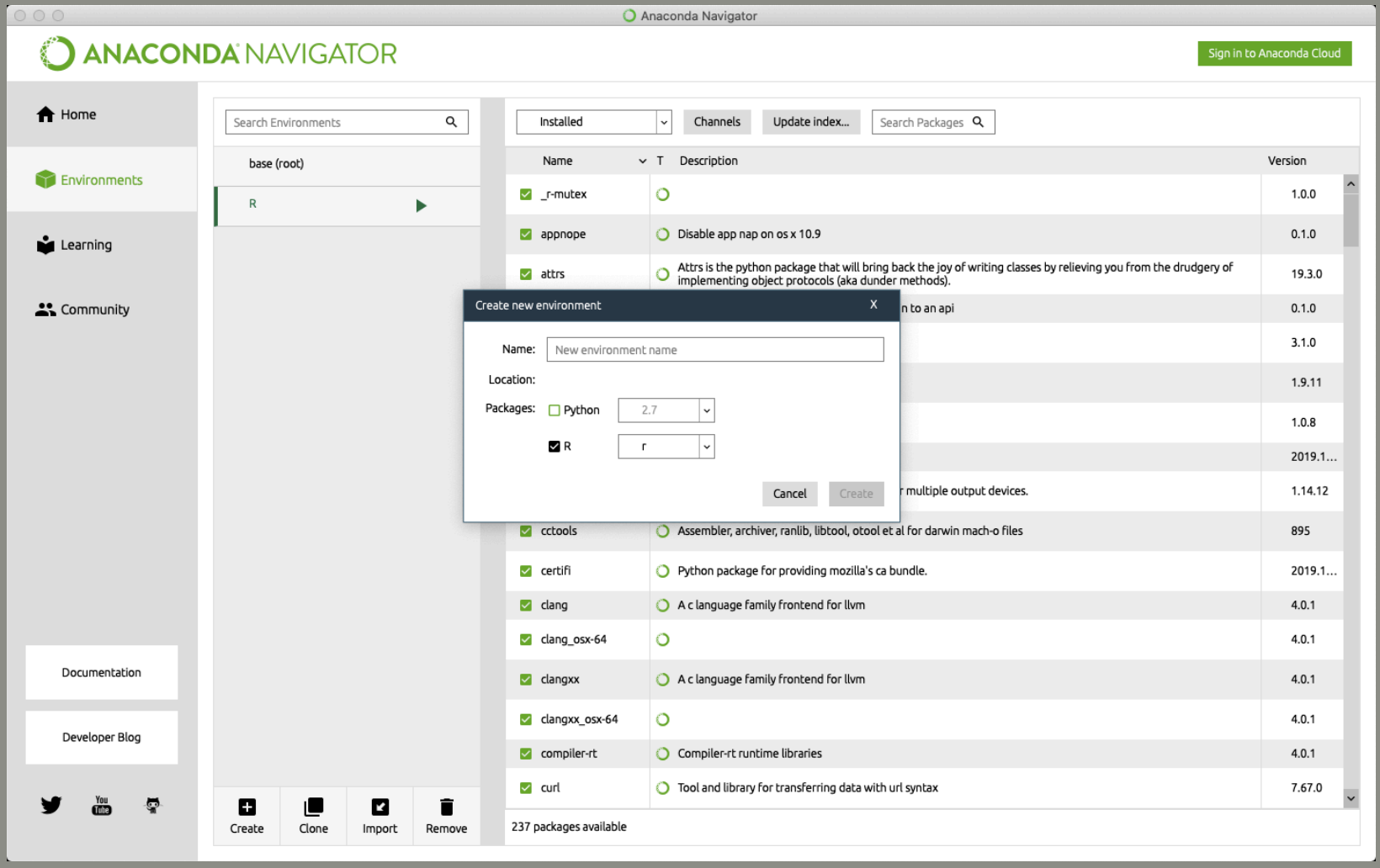
https://www.rstudio.com/



Explore RStudio Interface



Interactive R Environment in Anaconda



Interactive R Environment in Google Colab

- http://bit.ly/SMURProgGoogle
- Using R in python environment on Colab
  - *!jupyter-kernelspec list*
  - *%load_ext rpy2.ipython*
  - *%%R*
  - *x <- 42*
  - *print(x)*
- https://github.com/IRkernel/IRkernel/tree/master/example-notebooks

Quiz

1. What are the three properties of a vector, other than its contents?

2. What are the four common types of atomic vectors? What are the two rare types?

3. What are attributes? How do you get them and set them?

4. How is a list different from an atomic vector? How is a matrix different from a data frame?

5. Can you have a list that is a matrix? Can a data frame have a column that is a matrix?

DATA TYPES

Module 2

Comments and Help

Single line comment
#

Help
?....
help(...)

Variables

```
a <- 1
a = 1
1 -> a
```

(Most common)

Data Types

- Numbers
- Text
- Vector
- Matrix
- Array
- Data Frame
- Factor
- List

NUMBERS

Working With Numbers

2^4
abs(-3.2)
round(3.4)
round(5.24, digits = 1)
round(5.24, 1)
sqrt(4)
cos(pi/2), factorial(3)
3/0
Inf/Inf
NaN : Not a Number

TEXT

Splitting & Joining Text

```
a <- "Today is a good day"
strsplit(a," ")
strsplit(a,"is ")

a<-”bharat.verma"
b<-”smu.ca"
paste(a,b,sep="@")
```

Sorting Text

```
v <- c("Red","Blue","yellow","violet")
sort(v)
sort(v, decreasing=TRUE)
```

VECTOR

Create Vector Using : Operator

```
a <- 0:10
a <- 5:13
a <- 10:-4
```

```
class(a) - check the type of a
str(a)   - check the structure of a
```

Create Vector Using c Operator

```
c(1,2,4)
c(1,7:9)
c('red','green',"yellow")
c(1:5, 10.5, "red")
```

Create Vector Using seq

```
seq(10)
seq(3,20)
seq(3,20,3)
seq(1, 9, by = 2)
seq(from=4.5, to=2.5, by=-0.2)
seq(0, 1, length.out = 11)
```

Application of seq

```
x<-seq(0,4*pi,length.out = 200);
y <- sin(x);
plot(x,y)
```

Creating Vectors Using rep -1

```
rep(3,10)
rep(1:3, 3)
rep(c(1,2,3),3)
rep(seq(3),3)
rep(seq(3),length.out=5)
rep(seq(3),len=5)
rep(1:4, each = 2)
rep(1:4, c(2,1,2,1))
```

Creating Vectors Using rep -2

rep(1:4, each = 2, len = 4)
rep(1:4, each = 2, len = 10)
rep(1:4, each = 2, times = 3)

Accessing Vector Elements

a[5]
a[5:8]
a[5:3]
a[-2] - take away the 2nd element
a[-1:-4]
a[c(5, 6, 7, 8)]

Tail and Head

tail(a)
tail(a,n=3)
tail(a,3)

head(a)
head(a,n=3)
head(a,3)

Dropping NA values

```
a<-c(3,-2,4,NA,-1,8,-4,9,NA, 11,3,8,NA);
a[!is.na(a)]
```

NA: Missing data

Manipulating Vectors

```
x <- c(1,2,1)
y <- c(3,2,4)
v <- x + y
v <- x - y
```

Logical Indexing

c(1,2,3) = c(3,2,1)

1 %in% c(3,4,5)
c(1,2) %in% c(1,2,3,4)

Challenge: Logical Indexing

a <-c(2,3,-1,3,5,2,-3,1)

find the sum of the positive elements of a

Hint: a>0

Vector Arithmetic

a <-c(2,3,4,2,5,6)
mean(a)
median(a)
sum(a), prod(a)
min(a). max(a)
cummin(a), cummax(a)
cumsum(a), cumprod(a)
diff(a)

Sorting Vector Elements

```
v <- c(3,8,4,5,0,11, -9, 304)
sort(v)
sort(v, decreasing=TRUE)
```

MATRIX

Matrix

Matrices are the R objects in which the elements are arranged in a two-dimensional rectangular layout.

Syntax:

matrix(data, nrow, ncol, byrow = FALSE, dimnames)

Creating Matrix

Usage: matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)

```
matrix(1:12, ncol=4)
matrix(1:12, nrow=4)
matrix(c(3, 9, -1, 4, 2, 6), nrow=2)
matrix(1:12, ncol=4,byrow=TRUE)
matrix(1:12, nrow=4,byrow=TRUE)
```

Convert Vector to Matrix

```
a <- c(1,1,1,1)

dim(a)<-c(2,2)
```

Combining Vectors to Matrix

```
a1 <- c(1,1,1,1)
a2 <- c(2,2,2,2)
rbind(a1,a2)
cbind(a1,a2)
```

Accessing Matrix Elements

M[1,3]	- row 1, col 3
M[2,]	- row 2
M[,3]	- col 3
M[1:2,3:4]	- row 1 to 2, col 3 to 4

Challenge: Matrix Elements

a <- matrix(1:20,ncol=4). Extract the elements in red

```
     [,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
```

Time: 5 mins

Manipulating Matrix Elements

```
M1 <- matrix(c(3, 9, -1, 4, 2, 6), nrow=2)
M2 <- matrix(c(5, 2, 0, 9, 3, 4), nrow=2)
M3 <- M1 + M2
M4 <- M1 - M2
```

Row and Column Names

```
matrix(c(3, 9, -1, 4, 2, 6), nrow=2)
rownames(M)<-c("row1","row2")
colnames(M)<-c("col1","col2","col3")

M["row2",]
M[,"col3"]
```

Matrix Arithmetic

rowSums(M)	- sum of each row
colSums(M)	- sum of each col
colMeans(M)	- mean of each col
t(M1)	- transpose
solve(M1)	- invert
*	- element-wise multiplication
%*%	- dot product

ARRAY

WHAT IS ARRAY?

Arrays are the R data objects which can store data in more than two dimensions

CREATING ARRAY

```
a <-
array(c(11:14,21:24,31:34),dim=c(2,2,3))
```

Accessing Array Elements

a[,,1] - 1st matrix

a[1,,1] - 1st row, 1st matrix

DATA FRAME

Data Frame

A data frame is a table or a two-dimensional array-like structure in which each column contains values of one variable and each row contains one set of values from each column.

- The column names should be non-empty.
- The row names should be unique.
- The data stored in a data frame can be of numeric, factor or character type.
- Each column should contain same number of data items.

Creating Data Frame

```
a <- data.frame(
        gender = c("Male", "Male","Female"),
        height = c(152, 171.5, 165),
        weight = c(81,93, 78),
        age =c(42,38,26),
        row.names=c('Ally','Belinda','Alfred')
        )
```

Columns & Rows Names

rownames(x)
colnames(x) or names(x)

Rows & Column Numbers

nrow(x) : Number of Rows
ncol(x) : Number of Columns

Subset Operators

$: Select a single component from the data

[[: Select single component by name/position

[: Select multiple components

Accessing Single Component

Using $ operator

x$gender
x$height

Using [[operators]]

x[["gender"]]
x[[1]]

Accessing Multiple Components

Using [operator

```
a[1]
a["gender"]
a[-2]
a[1:2,]
a[c(1,4)]
a[,2]
a[c(2,3),c(1,2)]
```

FILTER DATA FRAME

```
a[a$gender=="Male",]
```

Filter Data using subset

```
subset(a, select = c("gender", "age"))
subset(a, subset = height > 163, select =
  c("gender", "height", "age"))
```

ADD COLUMN TO DATA FRAME

a$name =c('Ally','Belinda','Jane')

Add Rows to Data Frame

```
b <- data.frame(
                    gender = c("Female",
"Male","Female"),
                    height = c(155, 171.5, 155),
                    weight = c(71,93, 68),
                    age =c(42,38,26),
                    row.names =c('SC','Alfred','TC')
                    )

c <- rbind(a,b)
```

SUMMARY OF DATA FRAME

summary(a)

FACTOR

Factor

Factors are discrete (categorical) data such as colors, countries, digits, letters

```
a <- factor(c("east","east","west","south"))
a <- factor(rep(letters[1:3],each=4))
```

LIST

List

Lists are the R objects which contain elements of different types like − numbers, strings, vectors and another list inside it.

UNNAMED LIST

```
a  <-list(1:10, matrix(1:4,ncol = 2),"r" )
```

Named List

List groups different objects

a <-list('A'=1:10, 'B'=matrix(1:4,ncol = 2),'C'="r")

Accessing List Element

Using [[ ]] operator
a[[1]]
a[['A']]
Using [ ] operator
a[1]
a['A']
Using $ operator
a$A

Modifying List Element

```
list1[[1]] = c("d","e","f")
list1$'letters' - c("aa","bb","cc")
```

Merging Lists

```
list1 <- list(1,2,3)
list2 <- list("Sun","Mon","Tue")

merged.list <- c(list1,list2)
```

CONVERTING LIST TO VECTOR

v1 <- unlist(list1)

DATE

Create a Date

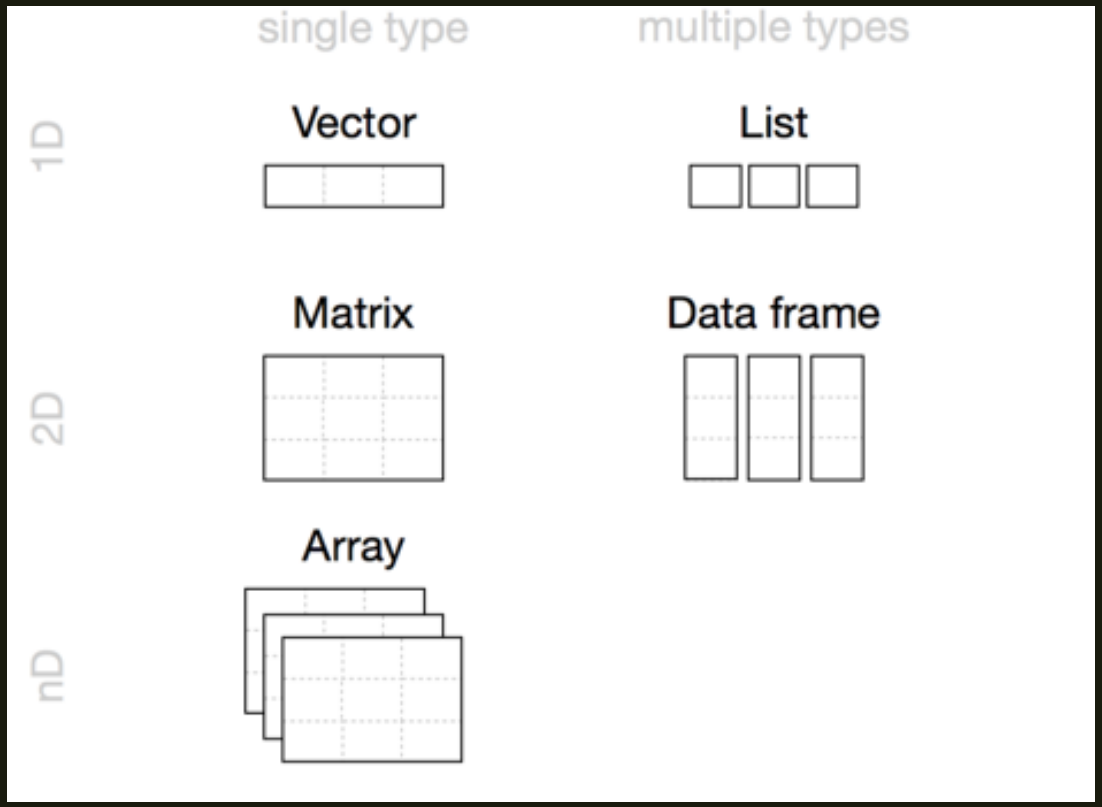
```
xd <-as.Date("2016-03-13")
xd <-as.Date("5 Aug 2016",format="%d %b %Y")
```

%Y : Year with century
%y : Year without century
%m : Month in decimal
%B : Full month name
%b : Abbreviated month name
%d : Day in decimal

Working with Date

weekdays(xd)
months(xd)
xd+7

DATA TYPES SUMMARY



PACKAGES & DATA SETS

Module 3

PACKAGES

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
Finance	Empirical Finance
Genetics	Statistical Genetics
Graphics	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
HighPerformanceComputing	High-Performance and Parallel Computing with R
MachineLearning	Machine Learning & Statistical Learning
MedicalImaging	Medical Image Analysis
MetaAnalysis	Meta-Analysis
Multivariate	Multivariate Statistics
NaturalLanguageProcessing	Natural Language Processing
NumericalMathematics	Numerical Mathematics
OfficialStatistics	Official Statistics & Survey Methodology
Optimization	Optimization and Mathematical Programming
Pharmacokinetics	Analysis of Pharmacokinetic Data
Phylogenetics	Phylogenetics, Especially Comparative Methods
Psychometrics	Psychometric Models and Methods
ReproducibleResearch	Reproducible Research
Robust	Robust Statistical Methods
SocialSciences	Statistics for the Social Sciences
Spatial	Analysis of Spatial Data
SpatioTemporal	Handling and Analyzing Spatio-Temporal Data
Survival	Survival Analysis

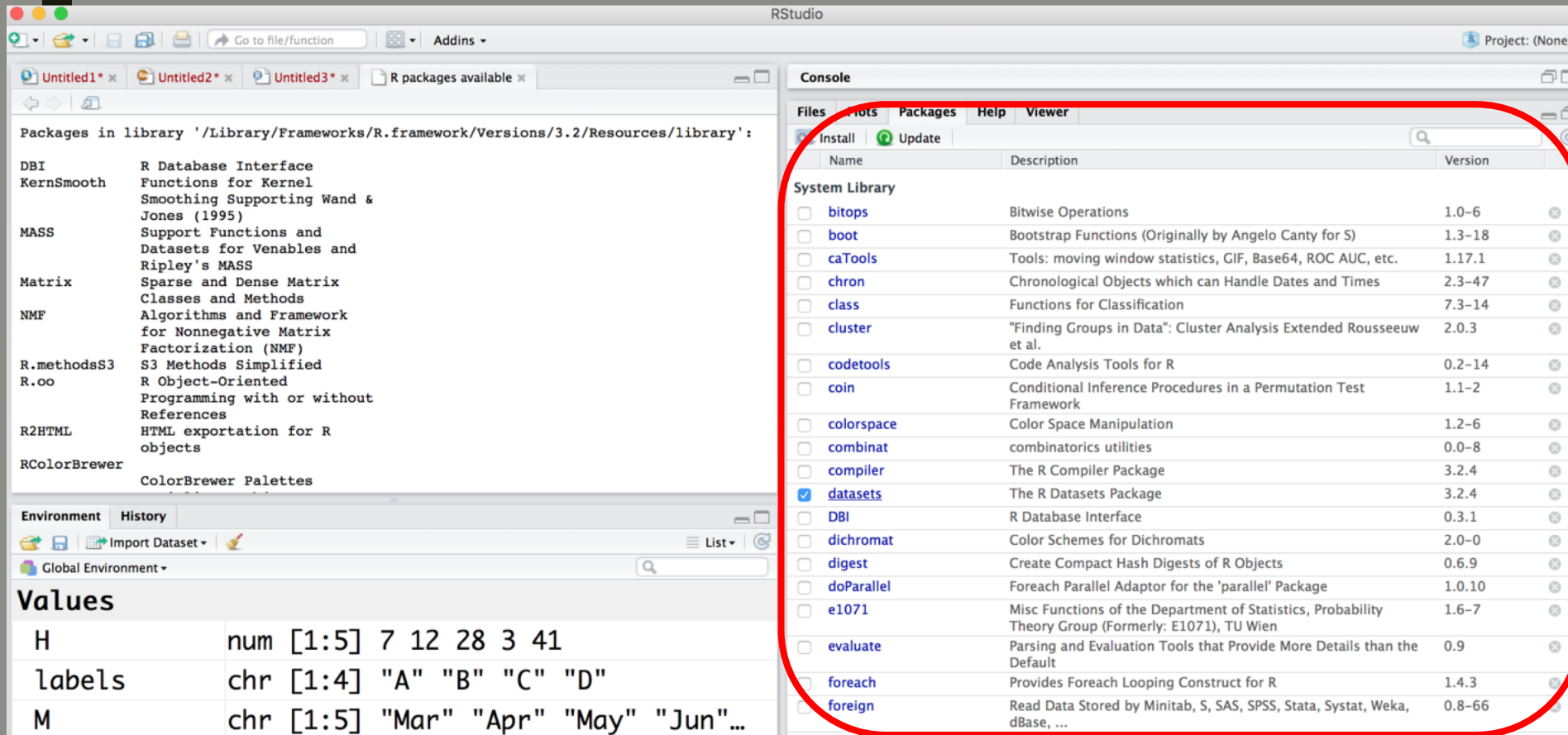
R PACKAGES

https://cran.r-project.org/web/views/

RStudio Package Management

nstall and load package in RStudio

unload and remove a package in RStudio



Install/Load Packages by Command Line

1. Install package
install.packages("ggplot2")

2. Load a package
library("ggplot2")

Unload and Remove Package

```
detach("package:ggplot2", unload = TRUE)

remove.packages("ggplot2")
```

DATA SETS

R DATASET PACKAGES

http://stat.ethz.ch/R-manual/R-devel/library/datasets/html/00Index.html

MTCar Dataset

- The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).
- A data frame with 32 observations on 11 variables.

data(mtcars)
str(mtcars)

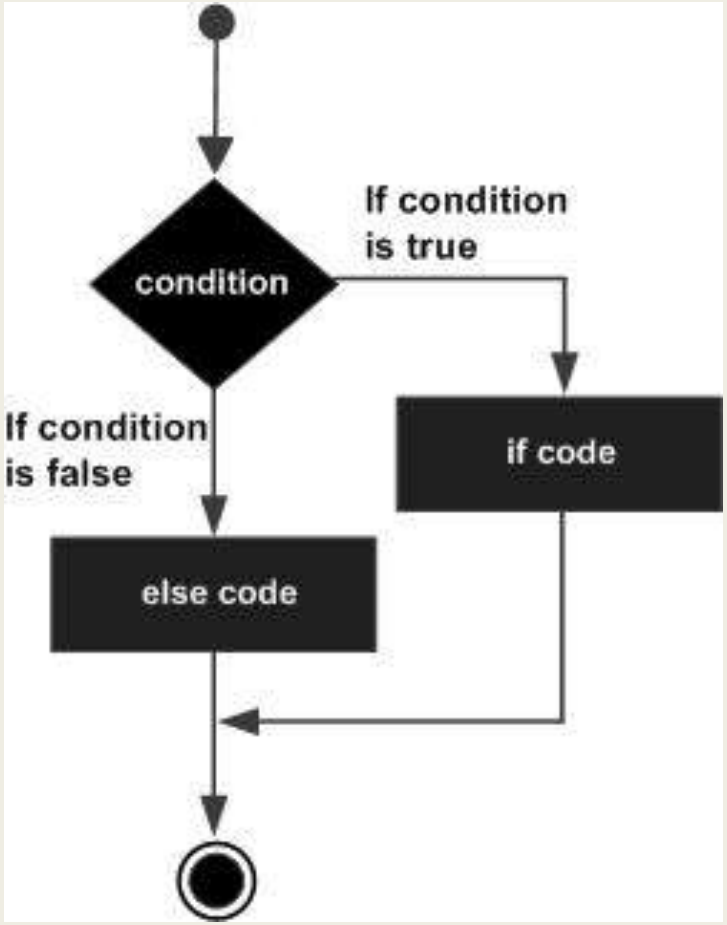
MTCar Dataset Variables

- mpg - Miles/(US) gallon
- cyl - Number of cylinders
- disp - Displacement (cu.in.)
- hp - Gross horsepower
- drat - Rear axle ratio
- wt - Weight (1000 lbs)
- qsec - 1/4 mile time
- vs - V/S
- am - Transmission
- gear - Number of forward gears
- carb - Number of carburetors

CONTROL STRUCTURES

Module 4

CONDITIONAL



IF-ELSE STATEMENT

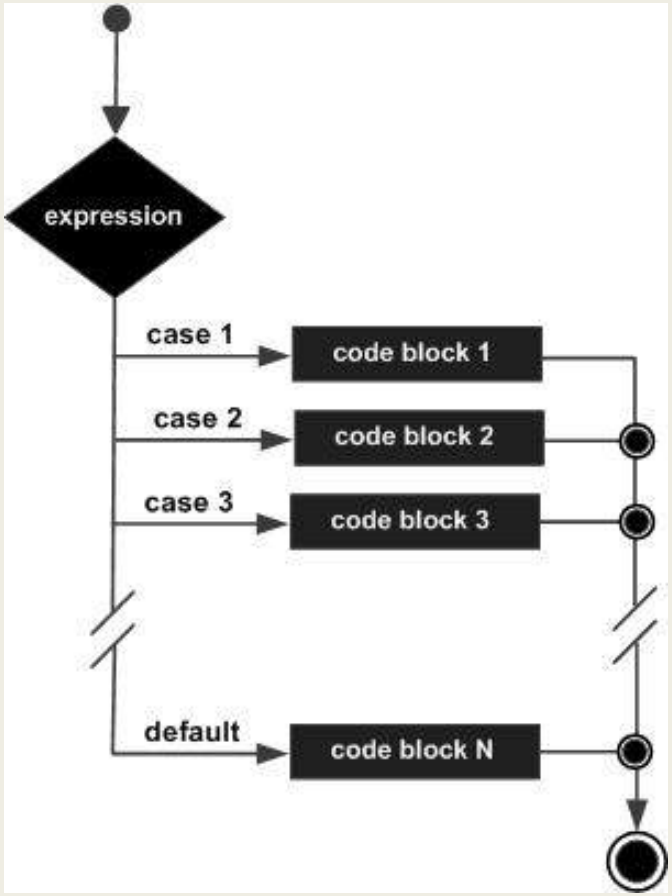
if-else Syntax

```
if (condition) {
do Something}
else {
do Something Else}
```

If-else Statements

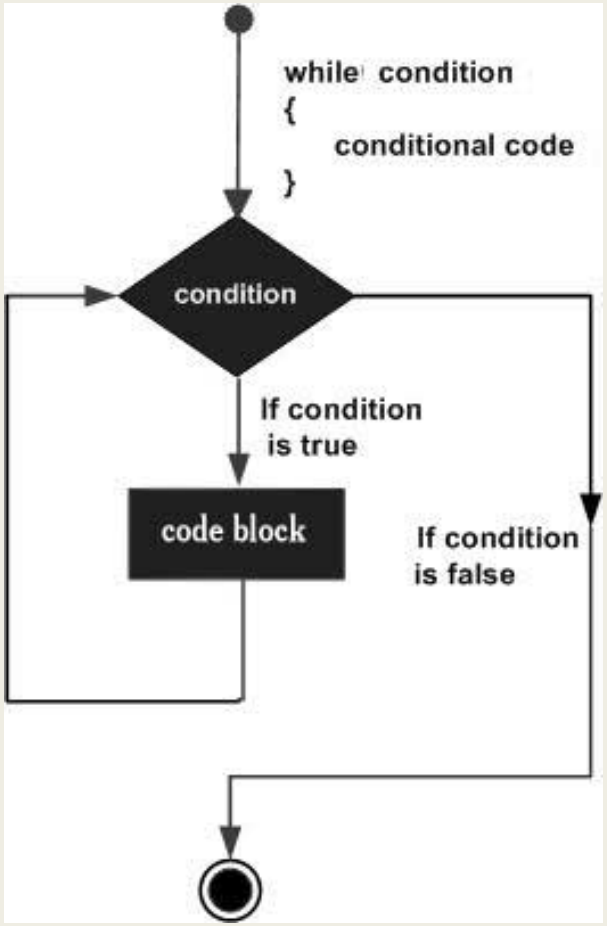
```
x <- 5
y <- 4

if (x<y) {
print("x is smaller than y")
} else {
print("x is larger than y")
}
```



SWITCH STATEMENT

LOOP



WHILE LOOP STATEMENT

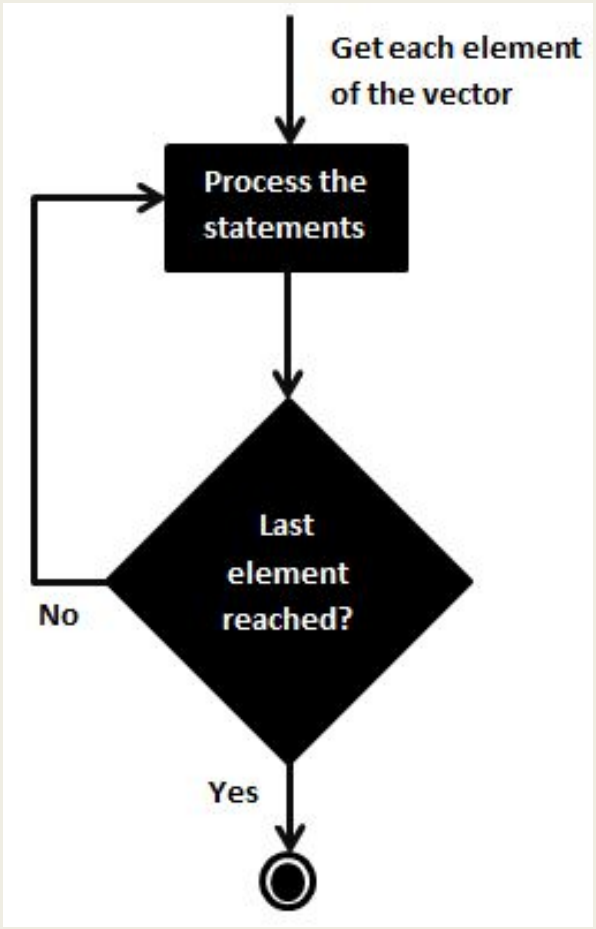
While Loops Syntax

```
while (condition) {
    do Something
}
```

While Loops Examples

```
x <- 0
while (x<10) {
        print(x)
        x <- x+1
}
```

While loop can potentially be infinite loop, be careful!



FOR LOOP STATEMENT

For Loops Syntax

```
for (variable in vector) {
    statements
}
```

For Loops Example

```
v <- c(1,2,3,4,5)
for ( i in v) {
   print(i)
}
```

Next

```
for ( i in 1:10) {
if (i == 7) {next}
print(i)
}
```

Break

```
for ( i in 1:10) {
if (i == 7) {break}
print(i)
}
```

Challenge

1. Read data1.csv
2. Do a subset of month of May and June
3. Count the number of days where temp is more than 65

Time: 10 mins

Hint to Challenge

```
data <- read.csv('data1.csv',header=TRUE)
data.subset <- subset(data, Month == 5 | Month ==
  6)
temp <- data.subset$Temp
count <- 0
for (i in temp) {
  if (i >65) { count <- count +1}
}
print(a)
```

OPERATORS

Arithmetic Operators

Addition	+
Substration	-
Multiplication	*
Division	/
Modulus	%%

Logical Operators

and	&&
or	||
not	!
elementwise	
and	&
or	|

FUNCTION

Module 5

Function syntax

```
variable_name  <- function(arg_1, arg_2, ...) {
    Function body
}
```

The last expression is the return value

Built In Functions in R

factorial(3)
mean(1:6)

Function Examples

```
f <- function(x) {
  x*x
}

filter <- function(x) {
  x[x>0]
}
```

Function with Default Args

```
f <- function(x=3,y=4) {
  x*2+y*3
}
```

Named Args

```
f <- function(x,y) {
  x*x+y*y*y
}
f(x=3,y=2)
f(x=2,y=3)
```

Argument Matching

The order of argument matching is

1. Check for exact match for a named argument
2. Check for a partial match
3. Check for a positional match

… Argument

The … argument indicate a variable number of arguments that are usually passed on to other functions.

```
f <- function(x,y,…) {
  plot(x,y,…)
}
```

Eg

```
f(x,y,col="red",main="sine")
```

Nested Function

```
make.power <- function(n) {
  pow <- function(x) {
    x^n
  }
  pow
}

cube = make.power(3)
cube(4)
```

Challenge: Function

Write a function to roll 2 dices, return the sum of the 2 dices

1 dice : 1, 2,3,4,5,6

Hint to Challenge

```
roll <- function() {
  ...
  dice <-sample(1:6,size=2)
  ....
}
```

Quiz Solution

1. The three properties of a vector are type, length, and attributes.

2. The four common types of atomic vector are logical, integer, double (sometimes called numeric), and character. The two rarer types are complex and raw.

3. Attributes allow you to associate arbitrary additional metadata to any object. You can get and set individual attributes with attr(x, "y") and attr(x, "y") <- value; or get and set all attributes at once with attributes().

4. The elements of a list can be any type (even a list); the elements of an atomic vector are all of the same type. Similarly, every element of a matrix must be the same type; in a data frame, the different columns can have different types.

5. You can make “list-array” by assuming dimensions to a list. You can make a matrix a column of a data frame with df$x <- matrix(), or using I() when creating a new data frame data.frame(x = I(matrix()).

Resources

- R Packages : https://cran.r-project.org/web/views/
- R Packages http://cran.stat.ucla.edu/web/packages/available_packages_by_name.html
- R Packages http://crantastic.org/