Introduction to R

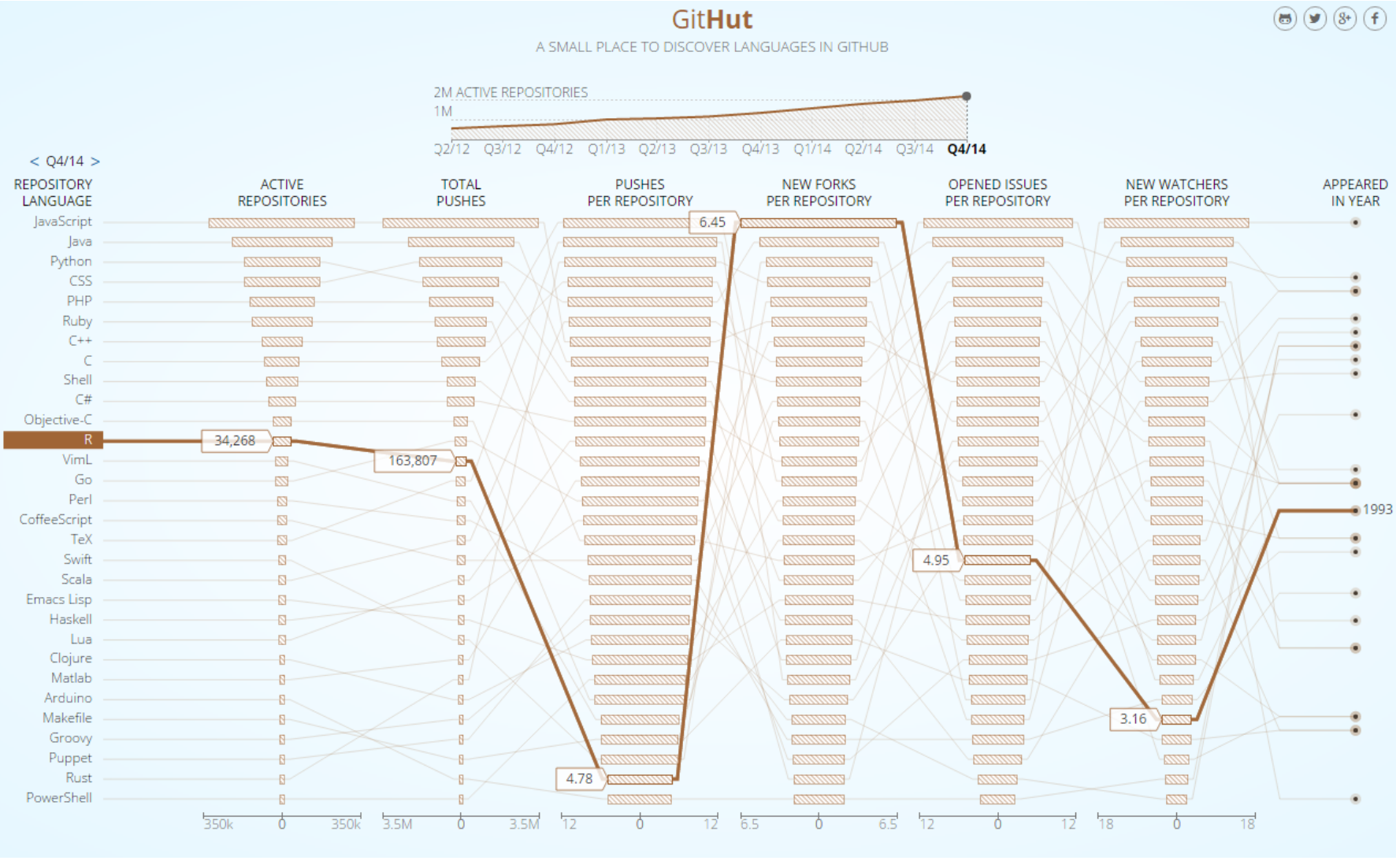
Computational Genomics

Weiguang (Wayne) Mao

Significant content courtesy by Silvia Liu

GitHut

A SMALL PLACE TO DISCOVER LANGUAGES IN GITHUB



Why or Why not

Statistical models can be written with only a few lines.

R + visualization = perfect match 

Cran "Task Views" page lists a wide range of tasks for which R packages are available

6789 packages, 6/18/2015

Bioconductor Open source software for bioinformatics

1104 packages

Statisticians, engineers and scientists without computer programming skills find it easy to use.

- Your mission
- Free, open source

REVOLUTION ANALYTICS

http://dataconomy.com/r-vs-python-the-data-science-wars/
http://blog.revolutionanalytics.com/2015/06/how-many-packages-are-there-really-on-cran.html

R VS. python

As a full-fledged programming language, Python is a good tool to implement algorithms for production use.

R is slow, on purpose

Average Annual Salaries In The Range Of:

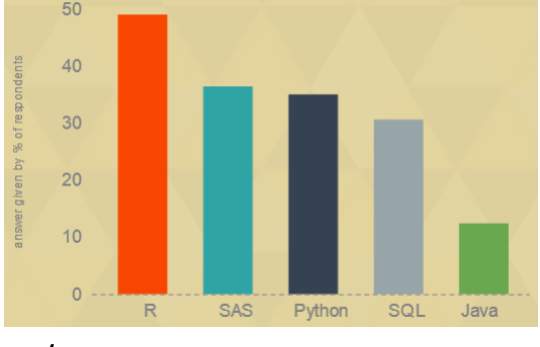
Python = US$110,000 to US$125,000 = R

Statisticians, engineers and scientists without computer programming skills find it easy to use.

Developed by statisticians, for statisticians

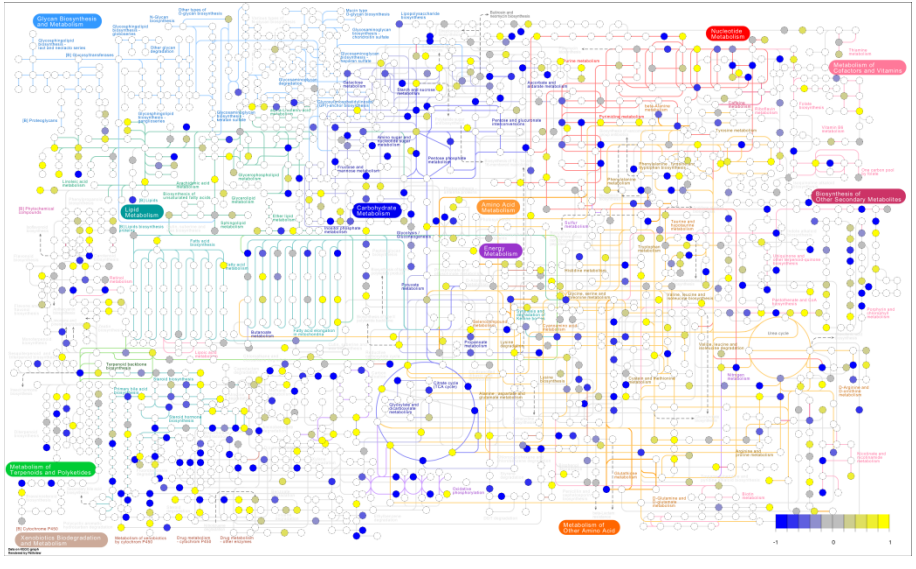
Python is used by programmers that want to delve into data analysis or apply statistical techniques, and by developers that turn to data science.

R has been used primarily in academics and research. However, R is rapidly expanding into the enterprise market.



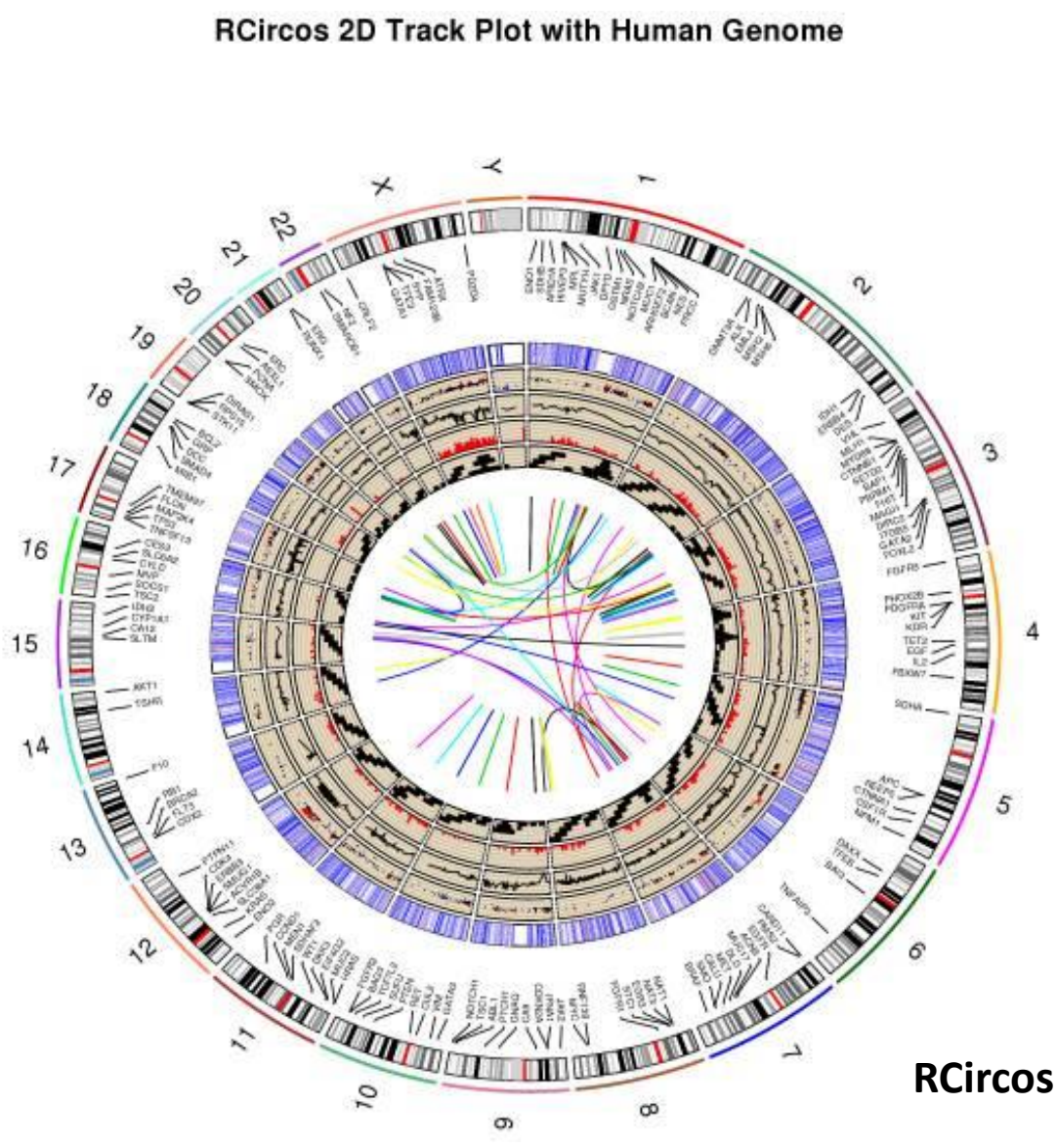
http://dataconomy.com/r-vs-python-the-data-science-wars/

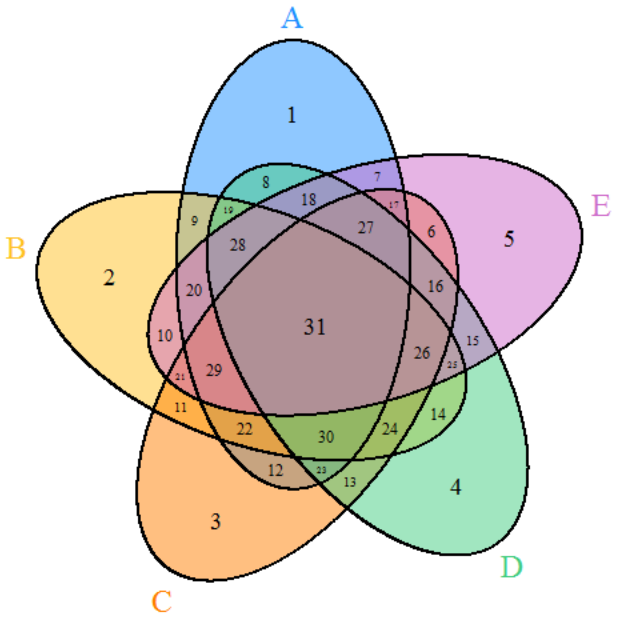
- ggplot2
- API (cytoscape)



Pathview: An R package for pathway based data integration and visualization

http://dataconomy.com/r-vs-python-the-data-science-wars/





VennDiagram: Generate High-Resolution Venn and Euler Plots

RCircos: an R package for Circos 2D track plots.

IDE & Versions

- R

  3.2.3 (12/10/2015) Wooden Christmas-Tree

- Rstudio

  0.99.491 (desktop version/cluster version)

Installation

- R & Rstudio
  http://lib.stat.cmu.edu/R/CRAN/
  https://www.rstudio.com/products/rstudio/download/

- R package
  - CRAN
    install.package("packageName")

  - Bioconductor
    source("http://bioconductor.org/biocLite.R")
    biocLite("packageName")

Importing

- library(packageName)

How to run R scripts

- Command line

  Rscript myscript.R

```
print("Hello World")
```

```
D:\Software\R Language\bin>Rscript Recitation.R
[1] "Hello World"
```

- Within R/Rstudio

  Select the code blocks you want to run and press Ctrl+R/Enter

```
> print("Hello World")
[1] "Hello World"
```

-2. = and <-

- Long time ago

  = (equal) <- (assignment)

  == (equal)

```
> print(x=3)
[1] 3
> print(y<-3)
[1] 3
```

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

```
> f(a<-3)
[1] 3
> f(a=3)
Error in f(a = 3) : unused argument (a = 3)
> f(x=3)
[1] 3
```

http://blog.revolutionanalytics.com/2008/12/use-equals-or-arrow-for-assignment.html

-1. data.frame

```
> x <- data.frame("SN"=1:2,"Age"=c(21,15),"Name"=c("John","Dora"),stringsAsFactors=FALSE)
> x
  SN Age Name
1  1  21 John
2  2  15 Dora
```

```
> str(x)
'data.frame':   2 obs. of  3 variables:
 $ SN  : int  1 2
 $ Age : num  21 15
 $ Name: chr  "John" "Dora"
```

```
> x[x$Name=="John",]
  SN Age Name
1  1  21 John
```

- Treat it as matrices
- Take care of strings/texts (factor or character)

0. Start from 1

```
> m0
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> apply(m0,1,sum)
[1]  9 12
```

```
SumOfRow = [0,0]
for i in range(2):
    for j in range(3):
        SumOfRow[i] = SumOfRow[i]+m0[i][j]
print SumOfRow
```

```
[[1, 3, 5], [2, 4, 6]]
[9, 12]
```

R, like S, is designed ***around a true*** computer language, and it allows users to add additional functionality by defining new functions.

https://www.r-project.org/about.html

1. Types and Variables

- Use class(x) to check what type the variable is
- Basic types
  - Numeric
  - Integer
  - Logical: True(T)/ False (F)
  - ~~String~~ Character

```
> as.numeric()
numeric(0)
> as.integer()
integer(0)
> as.character()
character(0)
```

2. ~~String~~ Character

- nchar()

```
> str <- "HelloWorld"
> nchar(str)
[1] 10
```

- paste()

```
> paste("Hello","World",sep="-")
[1] "Hello-World"
```

- grep(), regexpr(), gregexpr()

```
> grep("a",c("a","b","c"))
[1] 1
```

```
> regexpr("o","HelloWorld")
[1] 5
attr(,"match.length")
[1] 1
attr(,"useBytes")
[1] TRUE
```

```
> gregexpr("o","HelloWorld")
[[1]]
[1] 5 7
attr(,"match.length")
[1] 1 1
attr(,"useBytes")
[1] TRUE
```

- strsplit()

```
> strsplit("Hello,World",",")
[[1]]
[1] "Hello" "World"
```

- substring()

```
> substring("HelloWorld",1,5)
[1] "Hello"
```

- sub(), gsub()

```
> sub("o","*","HelloWorld")
[1] "Hell*World"
```

```
> gsub("o","*","HelloWorld")
[1] "Hell*W*rld"
```

3. Data Structures

- Vector

```
> v1 <- 2:5
> v1
[1] 2 3 4 5

> v3 <- c(3, 2, 7.2, 0.9, 100)
> v3
[1]   3.0   2.0   7.2   0.9 100.0


> v4 <- seq(4, 7, 0.5)
> v4
[1] 4.0 4.5 5.0 5.5 6.0 6.5 7.0



> v2 <- c(2,"Hello")
> v2
[1] "2"     "Hello"
```

- which(), sort(), length(), rev()

```
> v0 <- 5:2
> v0
[1] 5 4 3 2
> which(v0>2)
[1] 1 2 3
```

```
> sort(v0)
[1] 2 3 4 5

> length(v0)
[1] 4

> rev(v0)
[1] 2 3 4 5
```

3. Data Structures

- Lists

```
> l1 <- list(name=c("Peter","Lily","Emma"),c("yes","no"),
+ age=c(20,40,33,rep(18,times=3)),
+ value=matrix(1:6,2,3))
> l1
$name
[1] "Peter" "Lily"  "Emma"

[[2]]
[1] "yes" "no"

$age
[1] 20 40 33 18 18 18

$value
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```
> l1$name
[1] "Peter" "Lily"  "Emma"
> l1[[1]]
[1] "Peter" "Lily"  "Emma"
```

3. Data Structures

- Linked lists

```
> tree <- list(list(1, 2), list(3, list(4, 5)))
> # left child: list(1, 2)
> tree[[1]]
[[1]]
[1] 1

[[2]]
[1] 2
```

```
> # right child
> tree[[2]]
[[1]]
[1] 3

[[2]]
[[2]][[1]]
[1] 4

[[2]][[2]]
[1] 5
```

```
> # right child of right child:list(4, 5)
> tree[[2]][[2]]
[[1]]
[1] 4

[[2]]
[1] 5
```

4. Matrices

- 2-dimension

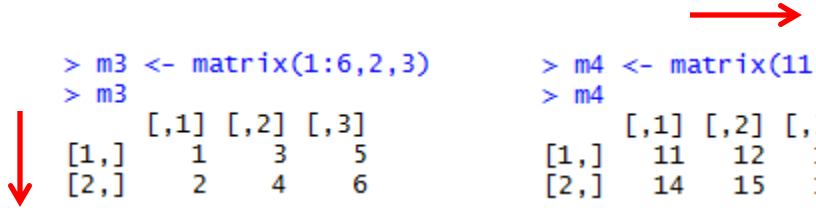
```
> d <- sample(1:20)
> d
 [1] 14 20  4  3 17 10 18 19  1  9 12  2 16 13 15  7  6  8 11  5

> m1 <- matrix(data=d,nrow=4,ncol=5,byrow=TRUE,
+ dimnames=list(rows=c("cat","dog","rat","fish"),
+ cols=c("a","b","c","d","e")))
> m1
      cols
rows    a  b  c  d  e
  cat  14 20  4  3 17
  dog  10 18 19  1  9
  rat  12  2 16 13 15
  fish  7  6  8 11  5
```

```
> m1["cat",]
 a  b  c  d  e
14 20  4  3 17
> m1[,1]
 cat  dog  rat fish
  14   10   12    7
```

```
> dim(m1)
[1] 4 5
> nrow(m1)
[1] 4
> ncol(m1)
[1] 5
> rownames(m1)
[1] "cat"  "dog"  "rat"  "fish"
> colnames(m1)
[1] "a" "b" "c" "d" "e"
```

4. Matrices

- 2-dimension



```
> m3 <- matrix(1:6,2,3)
> m3
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```
> m4 <- matrix(11:16,2,3,byrow=TRUE)
> m4
     [,1] [,2] [,3]
[1,]   11   12   13
[2,]   14   15   16
```

```
> m5 <- matrix(21:26,3,2)
> m5
     [,1] [,2]
[1,]   21   24
[2,]   22   25
[3,]   23   26
```

```
> m3*m4
     [,1] [,2] [,3]
[1,]   11   36   65
[2,]   28   60   96
```

```
> m3%*%m5
     [,1] [,2]
[1,]  202  229
[2,]  268  304
```

```
> rbind(m3,m4)
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
[3,]   11   12   13
[4,]   14   15   16
```

```
> t(m3)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

```
> apply(m3,1,sum)
[1]  9 12
```

```
> as.vector(m3)
[1] 1 2 3 4 5 6
```

4. Matrices

- Multi-dimension (array)

```
> a <- array(1:24,dim=c(4,3,2),
+ dimnames=list(c("a","b","c","d"),c("x","y","z"),c("old","new") ))
> a
, , old

  x y  z
a 1 5  9
b 2 6 10
c 3 7 11
d 4 8 12

, , new

   x  y  z
a 13 17 21
b 14 18 22
c 15 19 23
d 16 20 24
```

5. Dictionaries

- names(), rownames(), colnames()

```
> v1
[1] 2 3 4 5
> names(v1) <- c("a","b","c","d")
> v1[1]
a
2
> v1["a"]
a
2
```

```
> m1 <- matrix(data=d,nrow=4,ncol=5,byrow=TRUE,
+ dimnames=list(rows=c("cat","dog","rat","fish"),
+ cols=c("a","b","c","d","e")))
> m1
      cols
rows    a  b  c  d  e
  cat  14 20  4  3 17
  dog  10 18 19  1  9
  rat  12  2 16 13 15
  fish  7  6  8 11  5
```

```
> m1["cat",]
 a  b  c  d  e
14 20  4  3 17
> m1[,1]
 cat  dog  rat fish
  14   10   12    7
```

6. Conditionals

- If, else if, else

```
> x<-3
> if (x < 0){
+     print("Negative")
+ }else if (x == 0){
+     print("Zero")
+ }else{
+     print("Positive")
+ }
[1] "Positive"
```

```
> x <- 1
> if (x < 0){
+     print("Negative")
+ }
> else if (x == 0){
Error: unexpected 'else' in "else"
>     print("Zero")
[1] "Zero"
> }else{
Error: unexpected '}' in "}"
>     print("Positive")
[1] "Positive"
> }
Error: unexpected '}' in "}"
```

7. Iteration

- for

```
> sum <- 0
> for (i in 1:10){
+   sum <- sum+i
+ }
> sum
[1] 55
```

- while

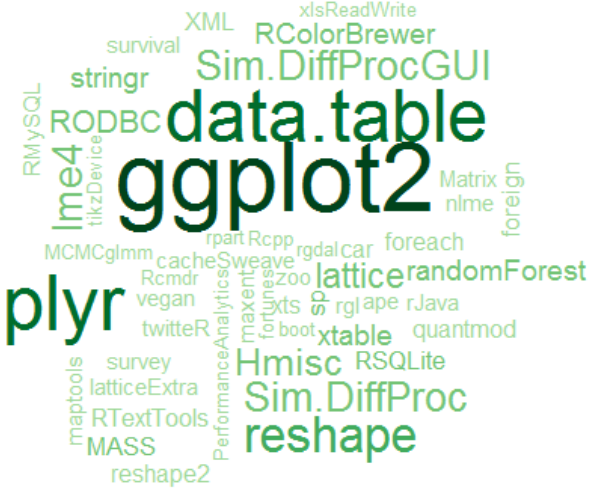
```
> sum <- 0
> i <- 1
> while (i <= 10){
+   sum <- sum+i
+   i <- i+1
+ }
> sum
[1] 55
```

- repeat

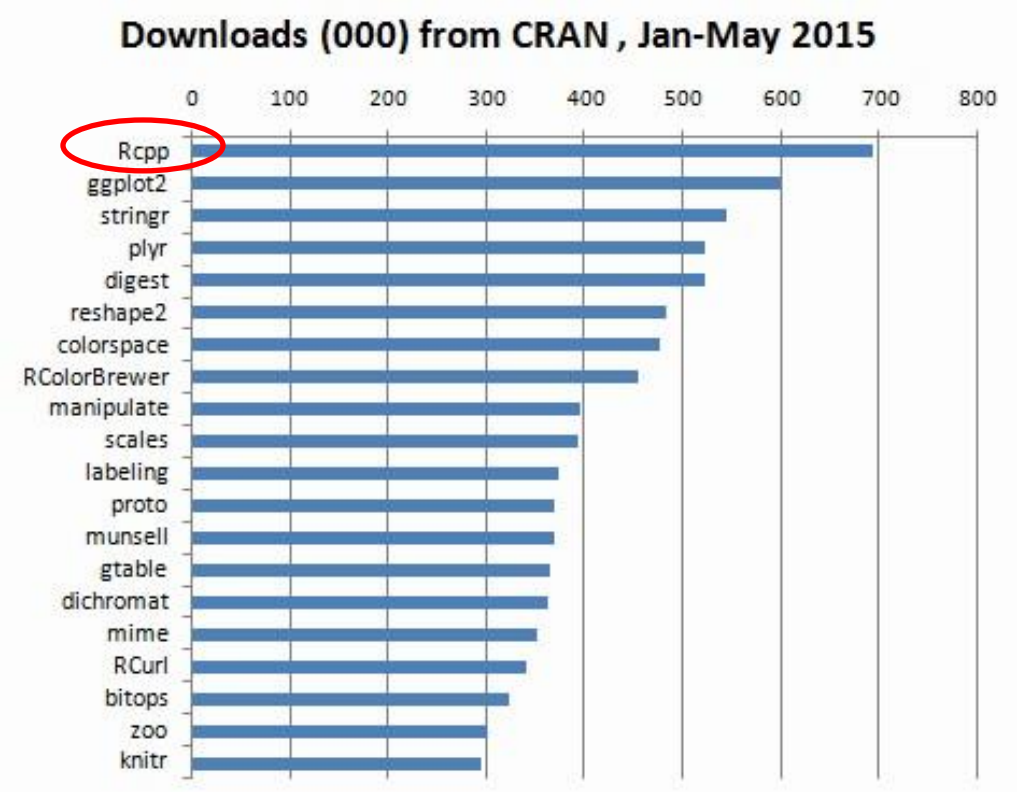
```
> sum <- 0
> i <- 0
> repeat{
+   i <- i+1
+   sum <- sum+i
+   if (i > 9) break
+ }
> sum
[1] 55
```

8. Functions

```
> add <- function(a,b){
+   a+b
+ }
> add(1,2)
[1] 3
```



2013



9. Writing Files

#write data to a file

write(x, file = "data", ncolumns = if(is.character(x)) 1 else 5, append = FALSE, sep = " ")

#write data to a file in a table format

write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ", eol = "\n", na = "NA", dec = ".", row.names = TRUE, col.names = TRUE, qmethod = c("escape", "double"), fileEncoding = "")

#save R objects into a file

save(…, file="")

#save the current workspace

save.image(file = ".RData", version = NULL, ascii = FALSE, compress = !ascii, safe = TRUE)

10. Reading Files

#Reads a file in table format and creates a data frame from it

read.table(file, header = FALSE, sep = "", quote = "\"'", dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"), row.names, col.names, as.is = !stringsAsFactors, na.strings = "NA", colClasses = NA, nrows = -1, skip = 0, check.names = TRUE, fill = !blank.lines.skip, strip.white = FALSE, blank.lines.skip = TRUE, comment.char = "#", allowEscapes = FALSE, flush = FALSE, stringsAsFactors = default.stringsAsFactors(), fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)

#read a csv file

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

#load datasets written with function *save*

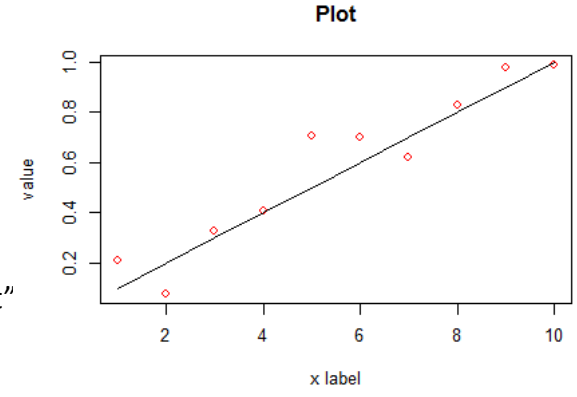
load(file, envir = parent.frame(), verbose = FALSE)

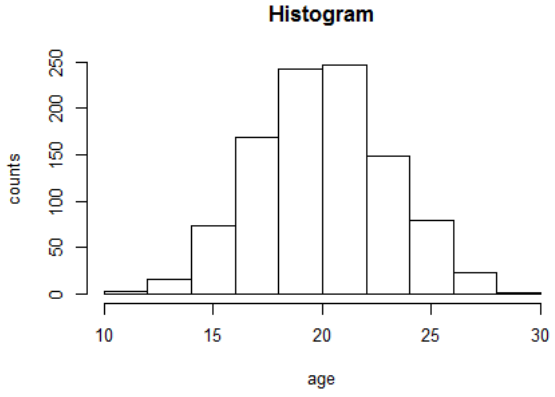
11. Plot

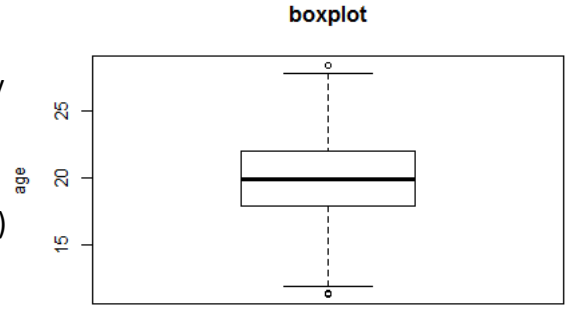
```
#2-by-2 sub graph
par(mfrow=c(2,2))

#graph 1: plot
x <- 1:10
y <- seq(0.1, 1, by=0.1)
z <- y+rnorm(10, mean=0, sd=0.1)
plot(x, z, type="p", col="red", main="Plot", xlab="x label", ylab="value")
lines(x,y)

#graph 2: histogram
age <- rnorm(1000, mean=20, sd=3)
hist(age, main="Histogram", xlab="age", ylab="counts")

#graph 3: boxplot
boxplot(age, main="boxplot", ylab="age")

#graph 4: qq plot
qqnorm(age, main="qq plot")
```

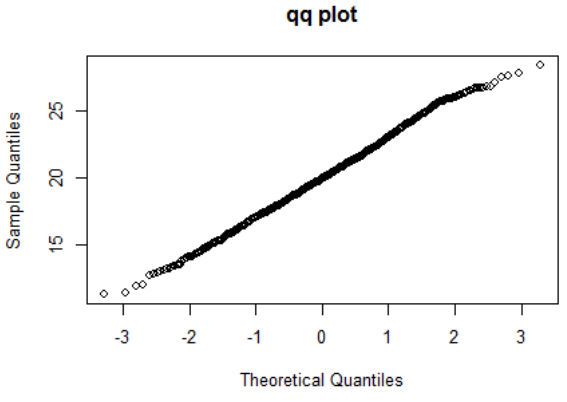
11. Plot

```
#2-by-2 sub graph
par(mfrow=c(2,2))

#graph 1: plot
x <- 1:10
y <- seq(0.1, 1, by=0.1)
z <- y+rnorm(10, mean=0, sd=0.1)
plot(x, z, type="p", col="red", main="Plot"
lines(x,y)

#graph 2: histogram
age <- rnorm(1000, mean=20, sd=3)
hist(age, main="Histogram", xlab="age", y

#graph 3: boxplot
boxplot(age, main="boxplot", ylab="age")

#graph 4: qq plot
qqnorm(age, main="qq plot")
```









Look it up!