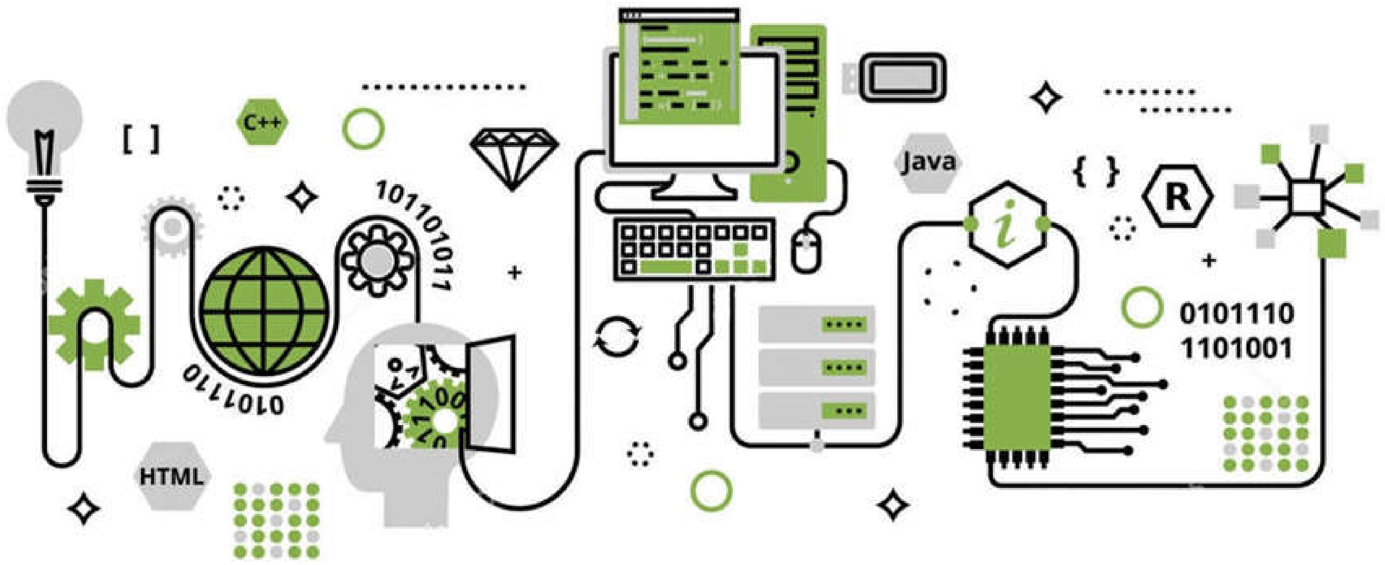


Introduction To R

Introduction to Bioinformatics

Roy Francis | 19-Sep-2018

Contents

Topics

- Familiarise with R & RStudio environment
- Running code, scripting, sourcing script
- Variables and operators
- Data types & datatype conversion
- Creating and running functions
- Base and grid graphics
- Input & output from R

What? Why R?

> R is a language and environment for statistical computing and graphics.

- Command line interface

Pros

- Data analysis
- Statistics
- High quality graphics
- Huge number of packages
- R is popular
- Reproducible research
- RStudio IDE
- FREE! Open source

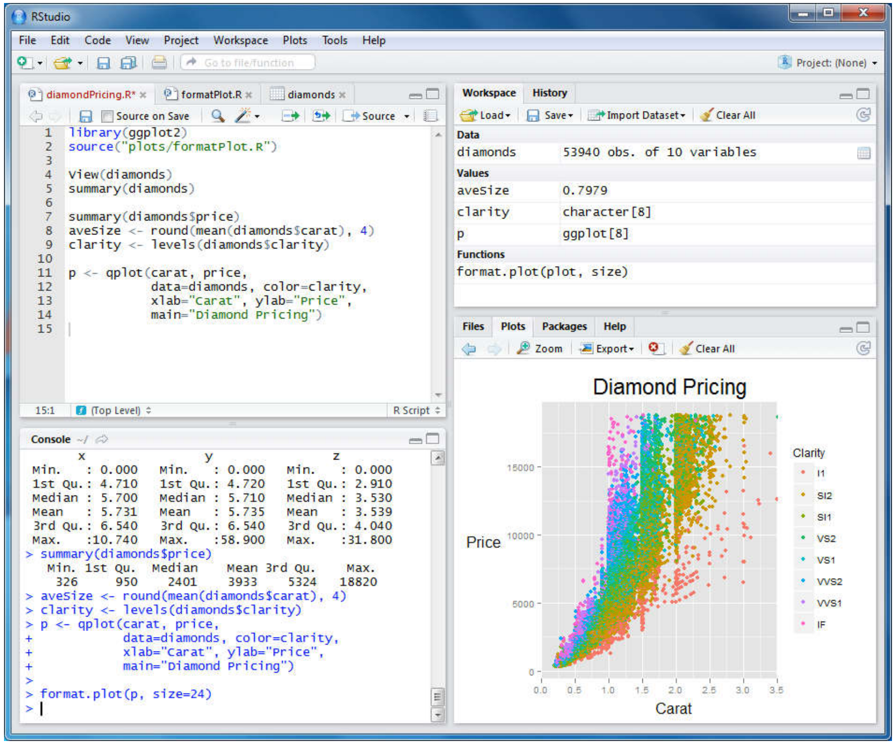
Cons

- Steep learning curve
- Not elegant/consistent
- Slow

Getting Started | Installation

NBIS SciLifeLab

- Install R from r-project.org.
- Install Rtools from https://cran.r-project.org/bin/
- Install RStudio IDE
- Code editor, highlighting, navigation, projects, version control, package building, debugger, profiler



```r
library(ggplot2)
source("plots/formatPlot.R")

View(diamonds)
summary(diamonds)

summary(diamonds$price)
aveSize <- round(mean(diamonds$carat), 4)
clarity <- levels(diamonds$clarity)

p <- qplot(carat, price,
           data=diamonds, color=clarity,
           xlab="Carat", ylab="Price",
           main="Diamond Pricing")
```

```
Data
diamonds        53940 obs. of 10 variables
Values
aveSize         0.7979
clarity         character[8]
p               ggplot[8]
Functions
format.plot(plot, size)
```

```
       x                y                z
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
 1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910
 Median : 5.700   Median : 5.710   Median : 3.530
 Mean   : 5.731   Mean   : 5.735   Mean   : 3.539
 3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040
 Max.   :10.740   Max.   :58.900   Max.   :31.800
> summary(diamonds$price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    326     950    2401    3933    5324   18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+            data=diamonds, color=clarity,
+            xlab="Carat", ylab="Price",
+            main="Diamond Pricing")
>
> format.plot(p, size=24)
> |
```

Getting Started | Interaction

- Execute commands directly in Console
- Ready console shows `>`
- Console shows `+` when waiting for information
- Press `Esc` to escape from `+` to `>`
- Save commands by writing scripts
- Run lines using `Ctrl`+`Enter`
- Run entire script using `Ctrl`+`Shift`+`Enter`

Variables & Operators

- Assign variables using `<-` or `=`

```
x <- 4
x = 4
x
```

```
## [1] 4
```

- Arithmetic operators

```
x <- 4; y <- 2;

x + y # add
x - y # subtract
x * y # multiply
x / y # divide
x %% y # modulus
x ^ y # power
```

```
## [1] 6
## [1] 2
## [1] 8
## [1] 2
## [1] 0
## [1] 16
```

- Logical operators return `TRUE` or `FALSE`

```
x == y # equal to?
x != y # not equal to?
x > y # greater than?
x < y # less than?
x >= y # greater than or equal to?
x <= y # less than or equal to?
```

```
## [1] FALSE
## [1] TRUE
## [1] TRUE
## [1] FALSE
## [1] TRUE
## [1] FALSE
```

```
T | F # OR
T & F # AND
```

```
## [1] TRUE
## [1] FALSE
```

- `any()`, `all` for logical vectors

Variables & Operators

- The function `paste()` concatenates strings

```
z <- "Look!"
z1 <- "Cat"
paste(z,z1)
```

```
## [1] "Look! Cat"
```

- Avoid conflicting variable names like `c`, `t` etc
- Variable names cannot start with a number
- `print()` function
- `\n` newline character

Data Type | Overview

```
##   dimension    homogenous heterogenous
## 1        1D Atomic vector         List
## 2        2D        Matrix    Dataframe
## 3        nD         Array
```

- Use `typeof()` to find type of a variable

```
x <- 4; typeof(x)
```

```
## [1] "double"
```

```
y <- "this"; typeof(y)
```

```
## [1] "character"
```

```
mode(x); class(x)
str(x); structure(x)
```

```
## [1] "numeric"
## [1] "numeric"
##  num 4
## [1] 4
```

- Explicit handling of missing data as `NA` and undefined data as `NULL` (NA vs NULL)

Data Type | Basic

NBIS SciLifeLab

- Mode

```
mode(1.0)
mode(1L)
mode("hello")
mode(factor(1))
mode(T)
```

```
## [1] "numeric"
## [1] "numeric"
## [1] "character"
## [1] "numeric"
## [1] "logical"
```

- Type

```
typeof(1.0)
typeof(1L)
typeof("hello")
typeof(factor(1))
typeof(T)
```

```
## [1] "double"
## [1] "integer"
## [1] "character"
## [1] "integer"
## [1] "logical"
```

Data Type | Vector | Create

- Vector stores multiple values
- Concatenate variables, values and vectors using the function `c()`

```
x <- c(2,3,4,5,6)
y <- c("a","c","d","e")
x
y
```

```
## [1] 2 3 4 5 6
## [1] "a" "c" "d" "e"
```

- Few different ways to create vectors.

```
c(2,3,5,6)
2:8
seq(2,5,by=0.5)
rep(1:3,times=2)
```

```
## [1] 2 3 5 6
## [1] 2 3 4 5 6 7 8
## [1] 2.0 2.5 3.0 3.5 4.0 4.5 5.0
## [1] 1 2 3 1 2 3
```

Data Type | Vector | Access

NBIS SciLifeLab

- Access vectors using the `[]` operator.

```
x[1]; y[3]
```

```
## [1] 2
## [1] "d"
```

- Function `c()` to specify multiple positions.

```
x[c(1,3)]
```

```
## [1] 2 4
```

- Vectorised operation

```
x <- c(2,3,4,5); y <- c(9,8,7,6)
x+y

z <- c("a","an","a","a"); k <- c("boy","apple","girl","mess")
paste(z,k)
```

```
## [1] 11 11 11 11
## [1] "a boy"     "an apple" "a girl"    "a mess"
```

Data Type | Vector

NBIS SciLifeLab

- Verify data type

```
x <- c(2,3,4,5)
z <- c("a","an","a","a")

mode(x)
mode(z)
str(x)
str(z)
```

```
## [1] "numeric"
## [1] "character"
##  num [1:4] 2 3 4 5
##  chr [1:4] "a" "an" "a" "a"
```

```
is.atomic(x)
is.numeric(x)
is.character(z)
```

```
## [1] TRUE
## [1] TRUE
## [1] TRUE
```

Data Type | Factor

NBIS SciLifeLab

- Factors store categorical data

```
x <- factor(c("a","b","b","c","c"))
class(x)
str(x)
```

```
## [1] "factor"
##  Factor w/ 3 levels "a","b","c": 1 2 2 3 3
```

- Factor `x` has 3 categories (3 levels)

```
levels(x)
```

```
## [1] "a" "b" "c"
```

- Verify if an R object is a factor

```
is.factor(x)
```

```
## [1] TRUE
```

Data Type | Matrix | Create

NBIS SciLifeLab

- Create a matrix from vector

```
x <- matrix(c(2,3,4,5,6,7))
x
```

```
##      [,1]
## [1,]    2
## [2,]    3
## [3,]    4
## [4,]    5
## [5,]    6
## [6,]    7
```

- Matrix has rows and columns

```
dim(x) # dimensions
nrow(x) # number of rows
ncol(x) # number of columns
```

```
## [1] 6 1
## [1] 6
## [1] 1
```

- Specify rows and columns

```
x <- matrix(c(2,3,4,5,6,7),nrow=3,
            ncol=2,byrow=TRUE)
x
```

```
##      [,1] [,2]
## [1,]    2    3
## [2,]    4    5
## [3,]    6    7
```

```
str(x)
```

```
##  num [1:3, 1:2] 2 4 6 3 5 7
```

- Verify if an R object is a matrix

```
is.matrix(x)
```

```
## [1] TRUE
```

Data Type | Matrix | Access

NBIS SciLifeLab

- Access matrix using `[]` operator as `[row,col]`

```
x[2,2]
```

```
## [1] 5
```

- Get whole row/col using `[row,]` or `[,col]`

```
x[1,]
x[,2]
```

```
## [1] 2 3
## [1] 3 5 7
```

- Use `drop=FALSE` to retain a matrix as `[row,col,drop=FALSE]`

```
x[1,,drop=F]
x[,2,drop=F]
```

```
##      [,1] [,2]
## [1,]    2    3
##      [,1]
## [1,]    3
## [2,]    5
## [3,]    7
```

Data Type | Matrix | Label

NBIS SciLifeLab

- Add row/column names

```
rownames(x) <- c("a","b","c")
colnames(x) <- c("k","p")
x
```

```
##   k p
## a 2 3
## b 4 5
## c 6 7
```

- Access using labels

```
x["b",]
x[,"p"]
```

```
## k p
## 4 5
## a b c
## 3 5 7
```

Data Type | List

NBIS SciLifeLab

- Create using `list()`.

```
x <- list(c(2,3,4,5),c("a","b","c","d"),
          factor(c("a","a","b")),
          matrix(c(3,2,3,5,6,7),ncol=2))
x
```

```
## [[1]]
## [1] 2 3 4 5
##
## [[2]]
## [1] "a" "b" "c" "d"
##
## [[3]]
## [1] a a b
## Levels: a b
##
## [[4]]
##      [,1] [,2]
## [1,]    3    5
## [2,]    2    6
## [3,]    3    7
```

```
typeof(x); class(x);
```

```
## [1] "list"
## [1] "list"
```

- Access lists using `[]` and `[[]]`

```
x[1]
```

```
## [[1]]
## [1] 2 3 4 5
```

- Lists are recursive

```
x <- list(list(list(list())))
str(x)
```

```
## List of 1
##  $ :List of 1
##   ..$ :List of 1
##   .. ..$ : list()
```

Data Type | `data.frame` | Create

NBIS SciLifeLab

```
dfr <- data.frame(x = 1:3, y = c("a", "b", "c"))
dfr
```

```
##   x y
## 1 1 a
## 2 2 b
## 3 3 c
```

```
str(dfr)
```

```
## 'data.frame':    3 obs. of  2 variables:
##  $ x: int  1 2 3
##  $ y: Factor w/ 3 levels "a","b","c": 1 2 3
```

- Use `stringsAsFactors=FALSE` to avoid auto factor conversion

```
dfr <- data.frame(x = 1:3, y = c("a", "b", "c"), stringsAsFactors = F)
str(dfr)
is.data.frame(dfr)
```

```
## 'data.frame':    3 obs. of  2 variables:
##  $ x: int  1 2 3
##  $ y: chr  "a" "b" "c"
## [1] TRUE
```

Data Type | `data.frame` | Access

- Access using `[]` or `$` operator

```
dfr$x
dfr$y
```

```
## [1] 1 2 3
## [1] "a" "b" "c"
```

- `head()` / `tail()` functions show first/last six lines
- Subset a `data.frame` using `subset()`

```
subset(dfr,dfr$y=="a")
```

```
##   x y
## 1 1 a
```

Data Type | Conversion

NBIS SciLifeLab

```
x <- c(1,2,3); str(x)
```

```
##  num [1:3] 1 2 3
```

- Convert to character

```
y <- as.character(x); str(y)
```

```
##  chr [1:3] "1" "2" "3"
```

- Character coerced (if possible) to number

```
x <- c("1","2","hello"); str(x)
```

```
##  chr [1:3] "1" "2" "hello"
```

```
str(as.numeric(x))
```

```
##  num [1:3] 1 2 NA
```

- Few other conversion functions

`as.matrix()`, `as.data.frame()`, `as.integer()`, `as.Date()`

Functions | Built-In | Math

NBIS SciLifeLab

```r
# generate 10 random numbers between 1 and 200
x <- sample(x=1:200,10); x;
```

```
##  [1] 189  21 117 182  90  47 165 164 150 124
```

```r
sum(x) # sum
mean(x) # mean
median(x) # median
min(x) # min
log(x) # log
exp(x) # exponent
sqrt(x) # square-root
round(x) # round
sort(x) # sort
```

```
## [1] 1249
## [1] 124.9
## [1] 137
## [1] 21
##  [1] 5.241747 3.044522 4.762174 5.204007 4.499810 3.850148 5.105945
##  [8] 5.099866 5.010635 4.820282
##  [1] 1.206861e+82 1.318816e+09 6.493134e+50 1.100514e+79 1.220403e+39
##  [6] 2.581313e+20 4.556061e+71 1.676081e+71 1.393710e+65 7.120586e+53
##  [1] 13.747727  4.582576 10.816654 13.490738  9.486833  6.855655 12.845233
##  [8] 12.806248 12.247449 11.135529
##  [1] 189  21 117 182  90  47 165 164 150 124
##  [1]  21  47  90 117 124 150 164 165 182 189
```

Functions | Built-In | String

```r
a <- "sunny"; b <- "day"

paste(a,b) # join
grep("sun",a) # find a pattern
nchar("sunny") # number of characters
toupper("sunny") # to uppercase
tolower("SUNNY") # to lowercase
sub("sun","fun","sunny") # replace pattern
substr("sunny",start=1,stop=3) # substring
```

```
## [1] "sunny day"
## [1] 1
## [1] 5
## [1] "SUNNY"
## [1] "sunny"
## [1] "funny"
## [1] "sun"
```

Functions | Custom

```
a <- 1:6; b <- 8:10

d <- a*b
e <- log(d)
f <- sqrt(e)
f
```

```
## [1] 1.442027 1.700109 1.844234 1.861649 1.951067 2.023449
```

- Custom function definition

```
my_function <- function(a,b){
  d <- a*b
  e <- log(d)
  f <- sqrt(e)
  return(f)
}
```

- Re-use function

```
my_function(a=2:4,b=6:8)
```

```
## [1] 1.576359 1.744856 1.861649
```

- Function names must not start with number

Control Structure | `if`

NBIS SciLifeLab

- Conditional statements using `if()`

```
a <- 2; b <- 5;
if(a < b) print(paste(a,"is smaller than",b))
```

```
## [1] "2 is smaller than 5"
```

- Use `else` for alternative output

```
if(a < b) {
  print(paste(a,"is smaller than",b))
}else{
  print(paste(b,"is smaller than",a))
}
```

```
## [1] "2 is smaller than 5"
```

- Chain `if else` statements

```
grade <- "B"

if(grade == "A"){
  print("Grade is Excellent!")
}else if(grade == "B"){
  print("Grade is Good.")
} else if (grade == "C") {
  print("Grade is Alright.")
}
```

```
## [1] "Grade is Good."
```

Control Structure | `for`

NBIS SciLifeLab

- Use `for()` loop for known iterations

```
for (i in 1:5){
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

- Use `while()` loop for unknown number of iterations

```
i <- 1
while(i < 5){
  print(i)
  i <- i+1
}
```

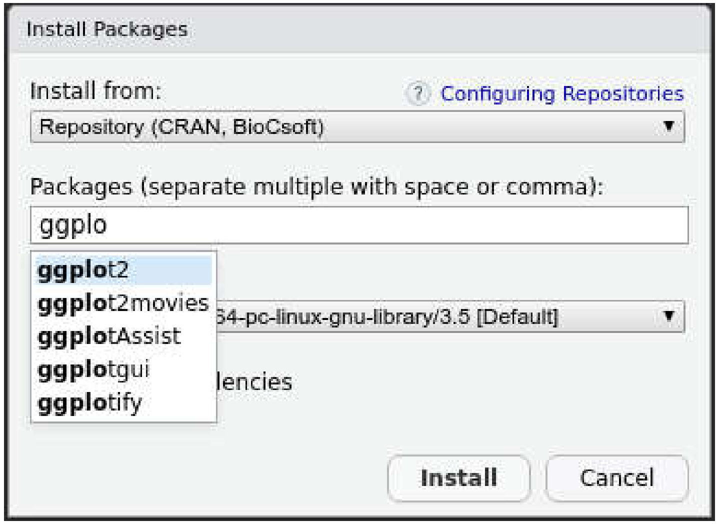
```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

R Packages

NBIS SciLifeLab

- CRAN (The Comprehensive R Archive Network)
- Bioconductor for Biology/Bioinformatics/NGS packages
- Use `install.packages()`

```
install.packages("ggplot2",dependencies=TRUE)
```



Install Packages

Install from: Repository (CRAN, BioCsoft)

Packages (separate multiple with space or comma): ggplo

ggplot2
ggplot2movies
ggplotAssist
ggplotgui
ggplotify

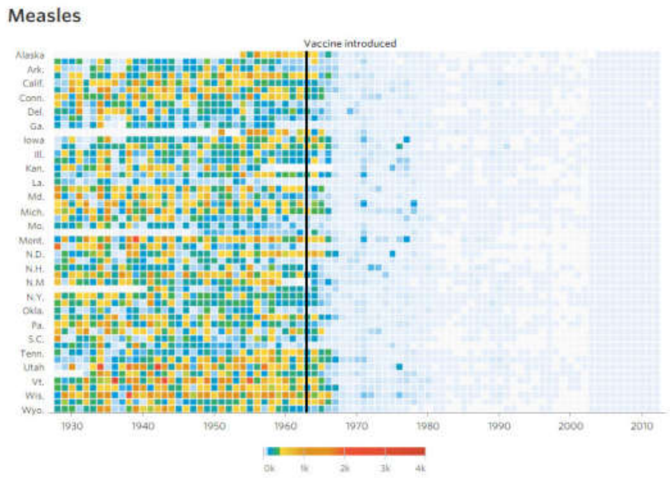
? Configuring Repositories
54-pc-linux-gnu-library/3.5 [Default]
lencies
Install
Cancel

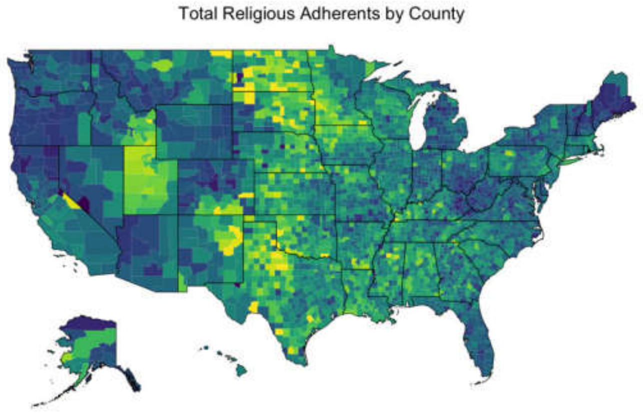
- For local packages, use `type="source"`

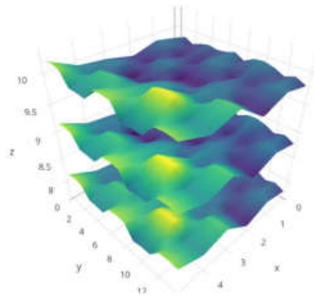
```
install.packages(path="./dir/package.zip",type="source")
```

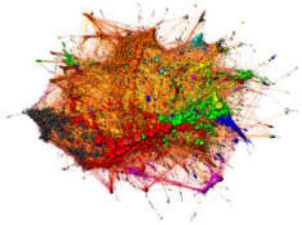
- Use `install_github()` from package `devtools` to install from GitHub

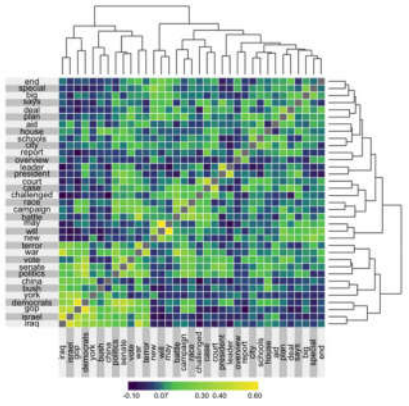
Graphics


Measles
Vaccine introduced
Alaska
Ark.
Calif.
Conn.
Del.
Ga.
Iowa
Ill.
Kan.
La.
Md.
Mich.
Mo.
Mont.
N.D.
N.H.
N.M
N.Y.
Okla.
Pa.
S.C.
Tenn.
Utah
Vt.
Wis.
Wyo.
1930
1940
1950
1960
1970
1980
1990
2000
2010
0k
1k
2k
3k
4k


Total Religious Adherents by County

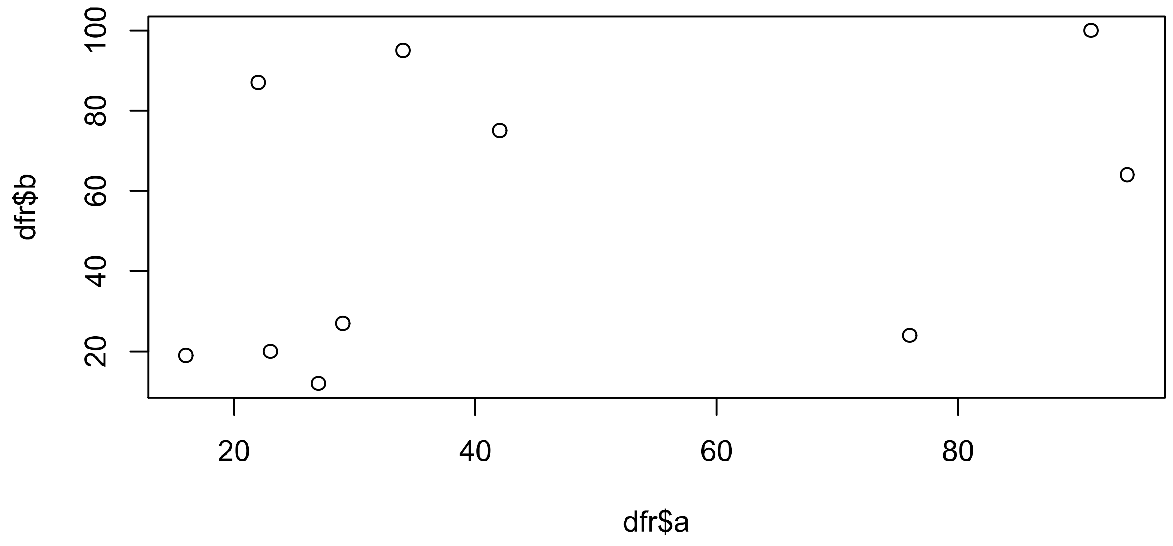

z
y
x




-0.10
0.07
0.30
0.40
0.60

Graphics | Base

```
dfr <- data.frame(a=sample(1:100,10),b=sample(1:100,10))
plot(dfr$a,dfr$b)
```

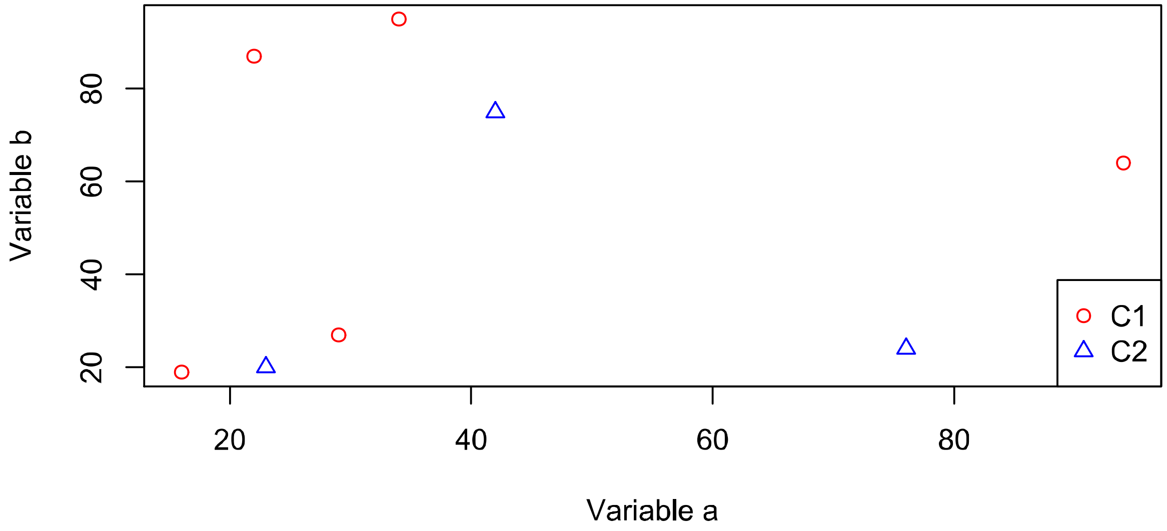


- Add axes labels etc

```
plot(dfr$a,dfr$b,xlab="Variable a",ylab="Variable b")
plot(dfr$a,dfr$b,xlab="Variable a",ylab="Variable b",type="b")
```

Graphics | Base

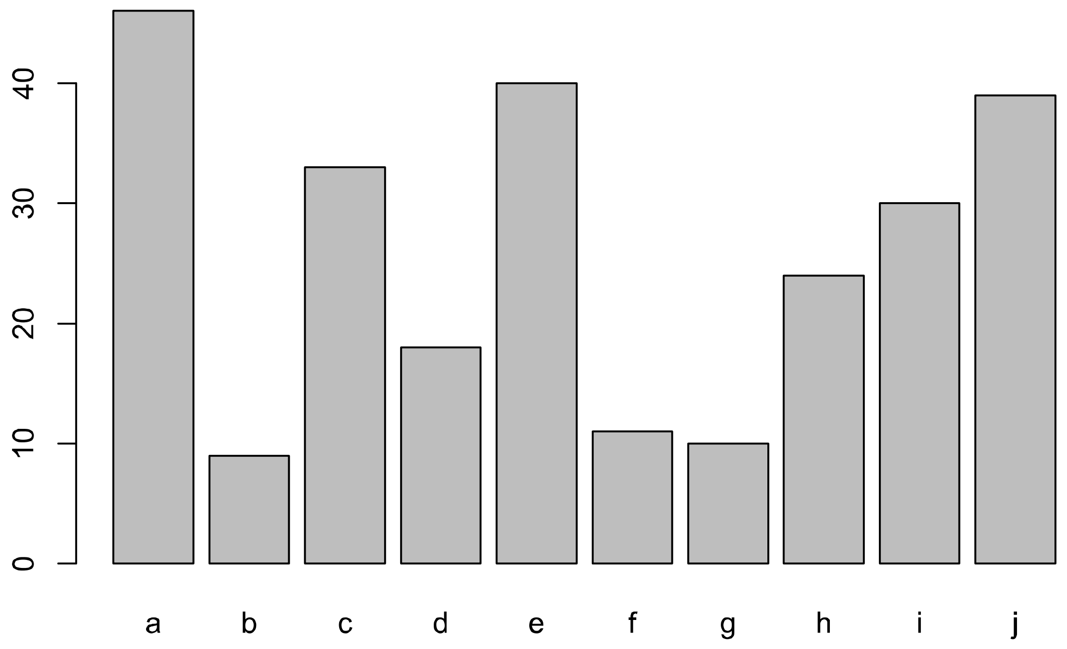
```
dfr$cat <- rep(c("C1","C2"),each=5) # add category

# subset data
dfr_c1 <- subset(dfr,dfr$cat == "C1")
dfr_c2 <- subset(dfr,dfr$cat == "C2")

plot(dfr_c1$a,dfr_c1$b,xlab="Variable a",ylab="Variable b",col="red",pch=1)
points(dfr_c2$a,dfr_c2$b,col="blue",pch=2)
legend(x="bottomright",legend=c("C1","C2"),
       col=c("red","blue"),pch=c(1,2))
```



Graphics | Base

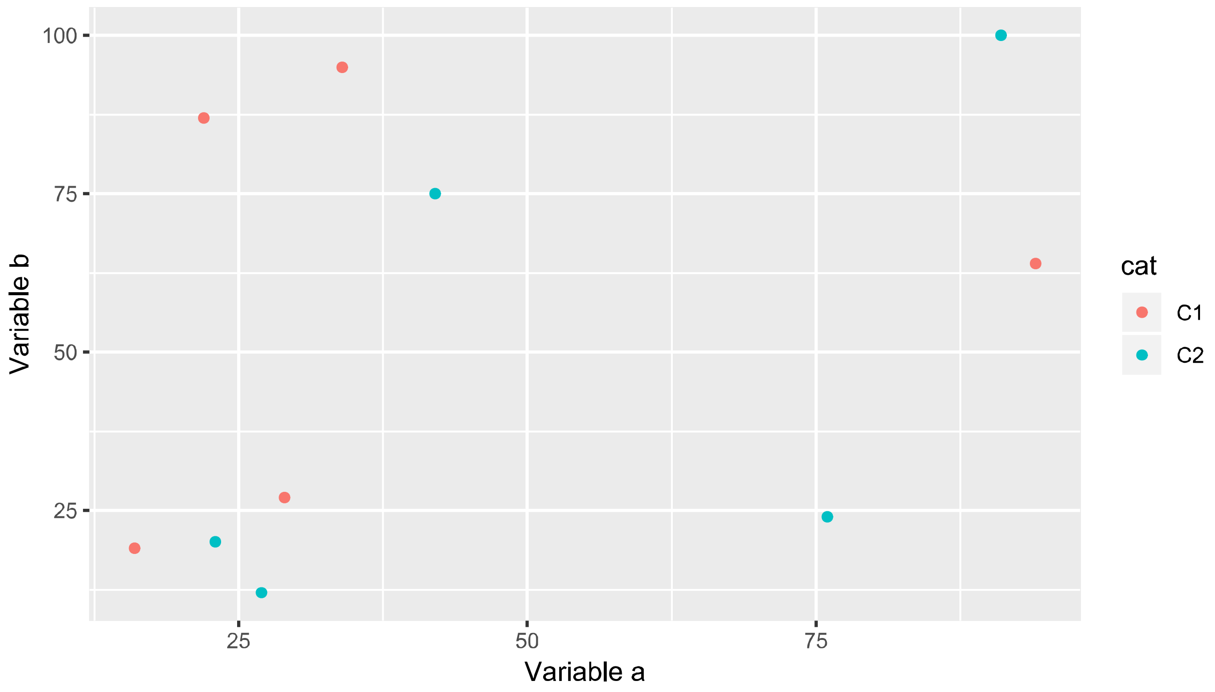
- Barplot

```
ldr <- data.frame(a=letters[1:10],b=sample(1:50,10))
barplot(ldr$b,names.arg=ldr$a)
```



Graphics | `ggplot2`

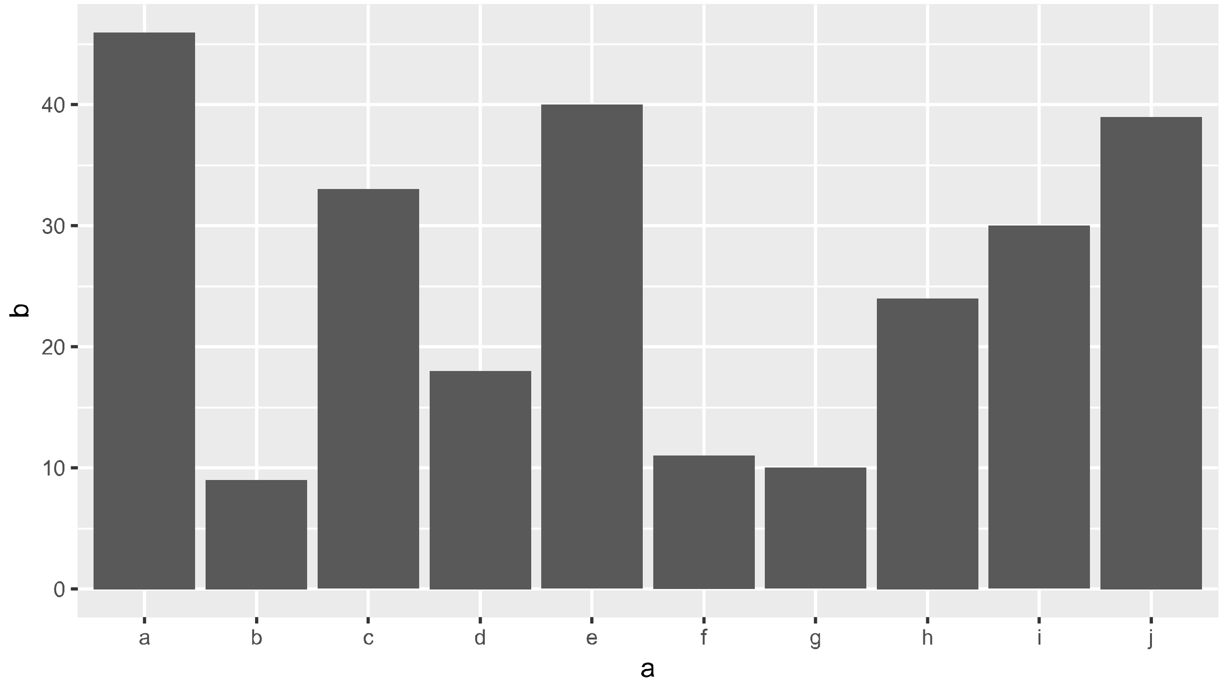
```r
library(ggplot2)

ggplot(dfr,aes(x=a,y=b,colour=cat))+
  geom_point()+
  labs(x="Variable a",y="Variable b")
```



Graphics | `ggplot2`

- Barplot

```r
ggplot(ldr,aes(x=a,y=b))+
  geom_bar(stat="identity")
```



Input/Output | Text

NBIS SciLifeLab

```
dfr <- read.table("iris.txt",header=TRUE,stringsAsFactors=F)
head(dfr)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
str(dfr)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : chr  "setosa" "setosa" "setosa" "setosa" ...
```

```
dfr1 <- dfr[dfr$Species == "setosa",]
write.table(dfr1,"iris-setosa.txt",sep="\t",row.names=F,quote=F)
```

`sep="\t"` sets tab delimiter, `row.names=F` avoids printing rownames, `quote=F` avoids quotes around strings.

Input/Output | Image

- Create data

```
dfr <- data.frame(a=sample(1:100,10),b=sample(1:100,10))
```

- Base plot

```
png(filename="plot-base.png")
plot(dfr$a,dfr$b)
dev.off()
```

- ggplot method 1

```
p <- ggplot(dfr,aes(a,b)) + geom_point()

png(filename="plot-ggplot-1.png")
print(p)
dev.off()
```

- ggplot method 2

```
ggsave(filename="plot-ggplot-2.png",plot=p)
```

Help

NBIS SciLifeLab

- Use `?function` to get function documentation
- Use `??name` to search for a function
- Use `args(function)` to get the arguments to a function
- Go to the package CRAN page/webpage for vignettes
- R bloggers: Great blog to follow to keep updated with the latest in the R world as well as tutorials.
- Stackoverflow: Online community to find solutions to your problems.

Learning R

There are lots of resources for getting help in R.

Tutorials

- Introduction to R: Tutorial by Datacamp with excellent tutorials.
- R programming tutorial: Youtube video tutorial by Derek Banas.
- R for data science Data science tutorial by Hadley wickham.
- Data carpentry Data carpentry R workshop (Medium-Advanced)

Reference

- R Cookbook: General purpose reference.
- Quick R: General purpose reference.
- Awesome R: Curated list of useful R packages.
- RStudio cheatsheets: Useful cheatsheets.
- Advanced R by Hadley Wickham (Medium-Advanced)

Links

- Tutorialspoint List: Good list of resources.

Useful packages

NBIS SciLifeLab

These are some useful packages when starting with data analysis in R.

- dplyr, tidyr: Data manipulation
- ggplot2: Data visualisation
- stringr: String manipulation
- lubridate: Date/time manipulation
- rmarkdown: Reproducible research and report generation

Thank you

Built on : 19-Sep-2018 at 11:09:42. Graphics from freepik.com