Genetic
Diversity
Centre
Zurich

Biocomputing with R

Niklaus Zemp
17 June 2025

Genetic Diversity Centre (GDC)
Bioinformatics
ETH Zurich

Dice example

Write an R script (R code) that creates a text file containing the results of 10 rolls of a pair of dice.

Rprojects

Use AI Tools

R-packages

Comments

set.seed()

Use variables

pip

loops

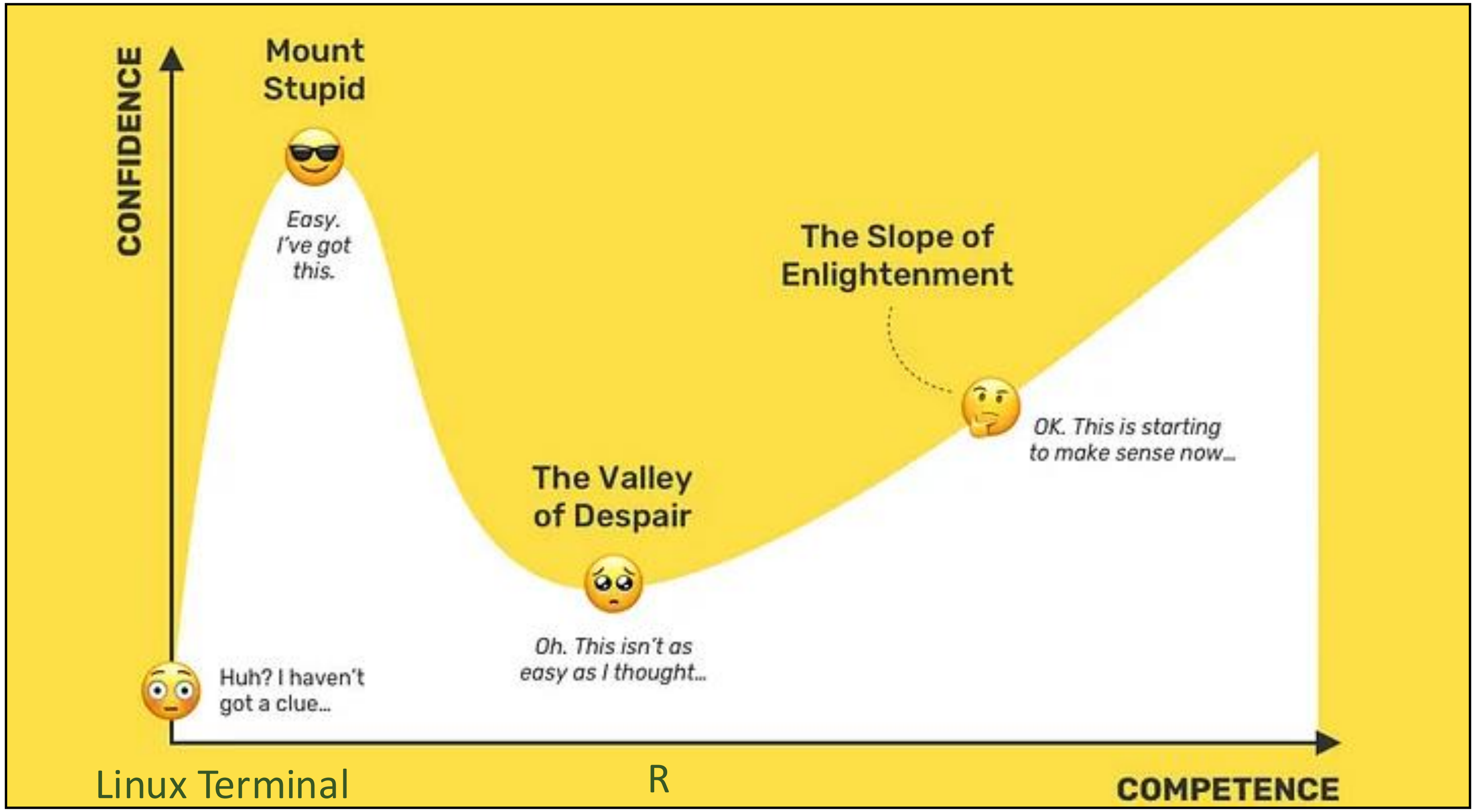
own functions

tidyverse

rMarkdown

Good scripting involves writing code that is not only functional but also maintainable, readable, efficient and flexible.

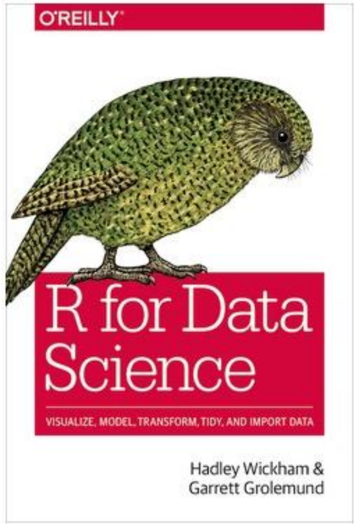
Dunning-Kruger effect



Resources

Claude

Many many tutorials, forum, YouTube videos posts and books available



O'REILLY

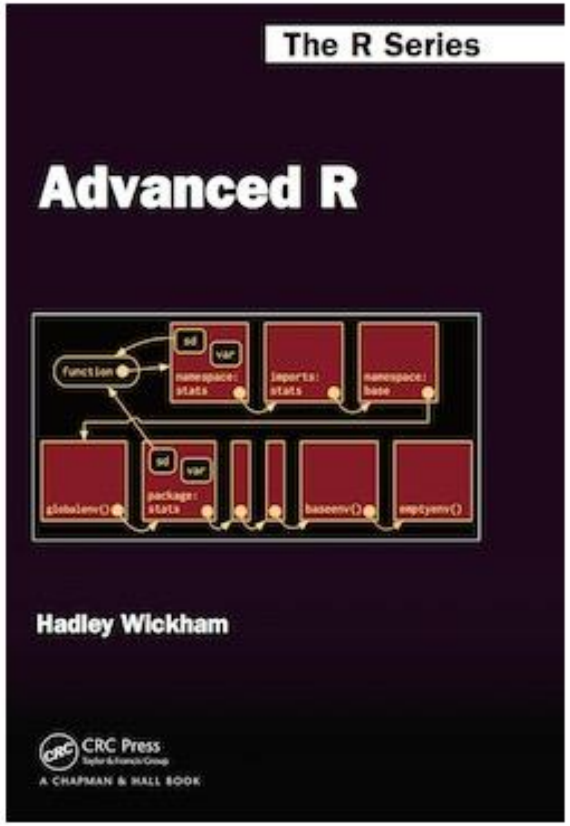
R for Data Science

VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA

Hadley Wickham &
Garrett Grolemund

R-bloggers

R news and tutorials contributed by (750) R bloggers



The R Series

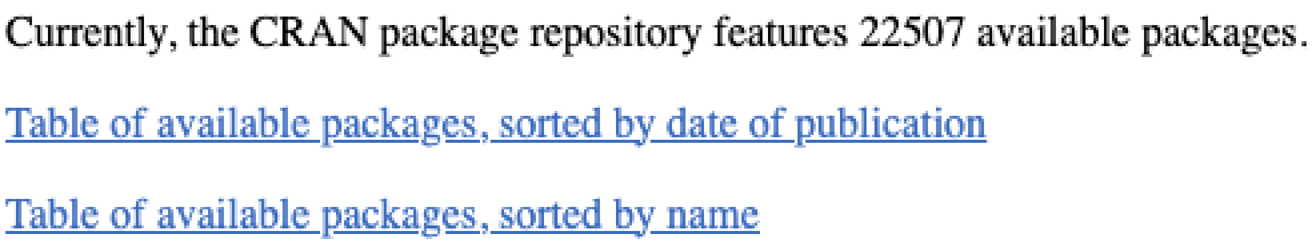
Advanced R

Hadley Wickham

CRC Press

A CHAPMAN & HALL BOOK

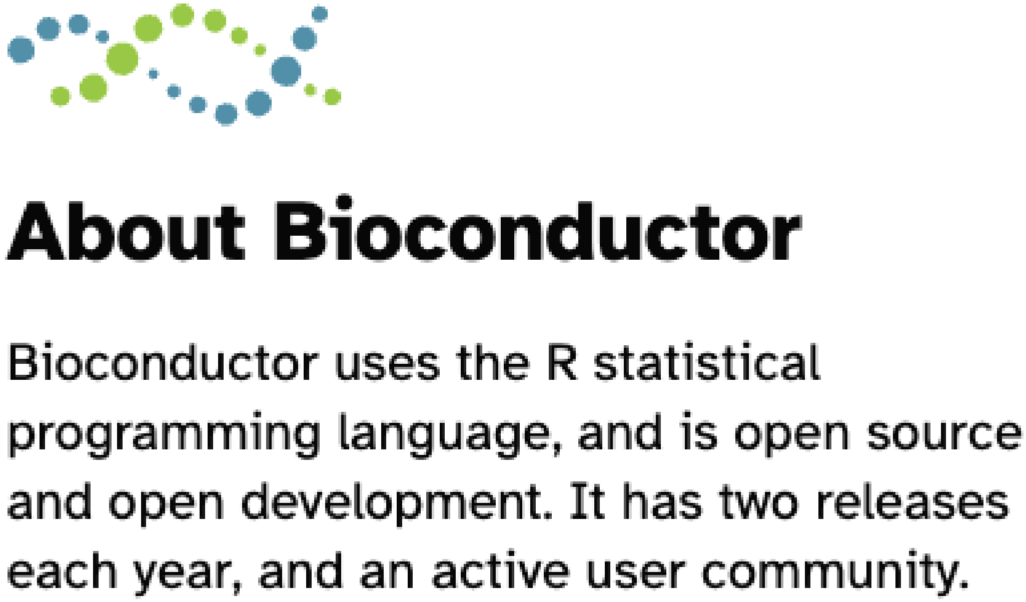
Packages



Currently, the CRAN package repository features 22507 available packages.

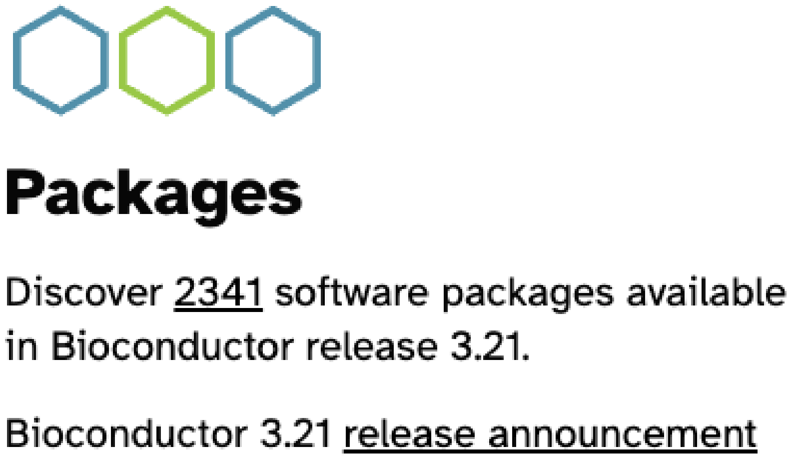
Table of available packages, sorted by date of publication

Table of available packages, sorted by name



About Bioconductor

Bioconductor uses the R statistical programming language, and is open source and open development. It has two releases each year, and an active user community.



Packages

Discover 2341 software packages available in Bioconductor release 3.21.

Bioconductor 3.21 release announcement

Tidyverse

Rstudio

netic Diversity Centre (GDC) -
urse Webpage

art
quirements
ux 1 - Local Terminal
ux 2 - Remote Terminal
ocomputing
ocomputing with R
ocomputing on a HPC cluster
producible Research
PS (NGS)

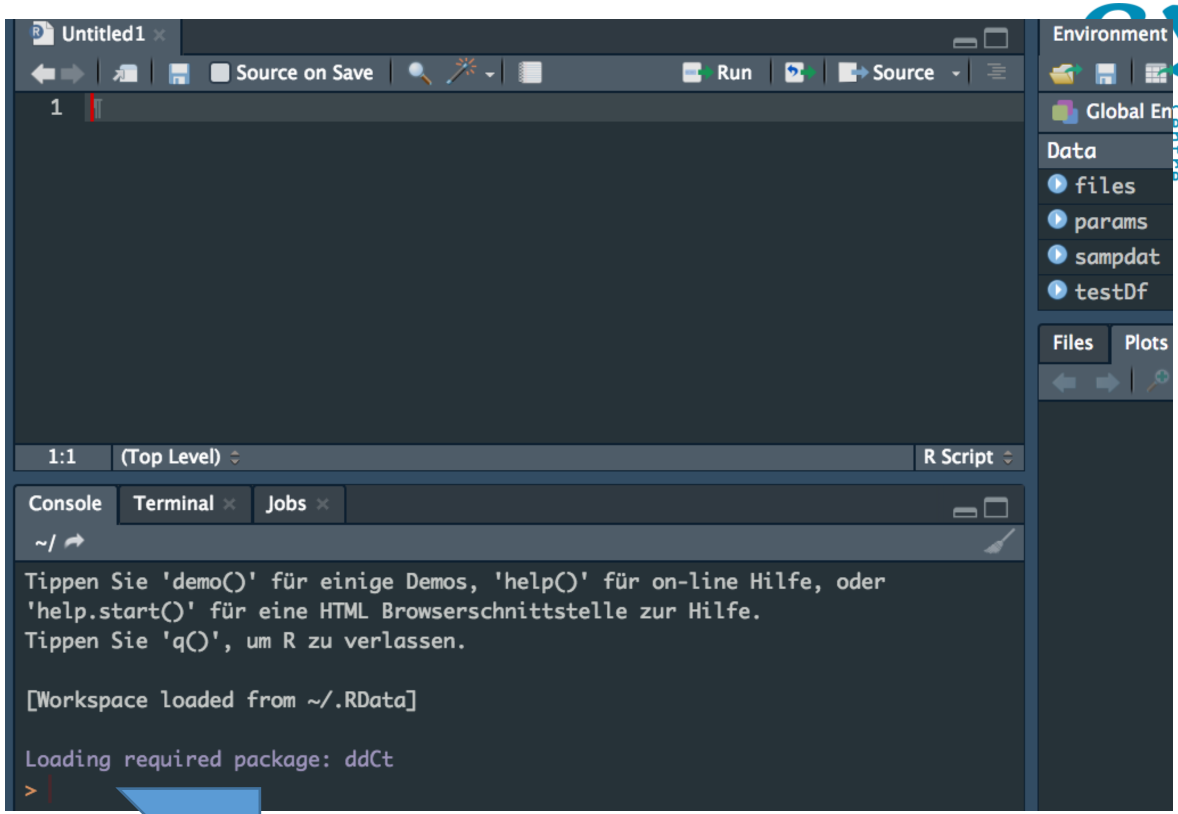
Biocomputing with R

```
### set working directory
setwd("~/Desktop")

### generate folder
dir.create("Demo")
setwd("Demo")

### download scripts for Demo
utils::download.file("https://www.gdc-docs.ethz.ch/GeneticDiversityAnalysis/GDA
utils::download.file("https://www.gdc-docs.ethz.ch/GeneticDiversityAnalysis/GDA

### open R scripts
file.edit("My_functions.R")
file.edit("Demo.R")
```



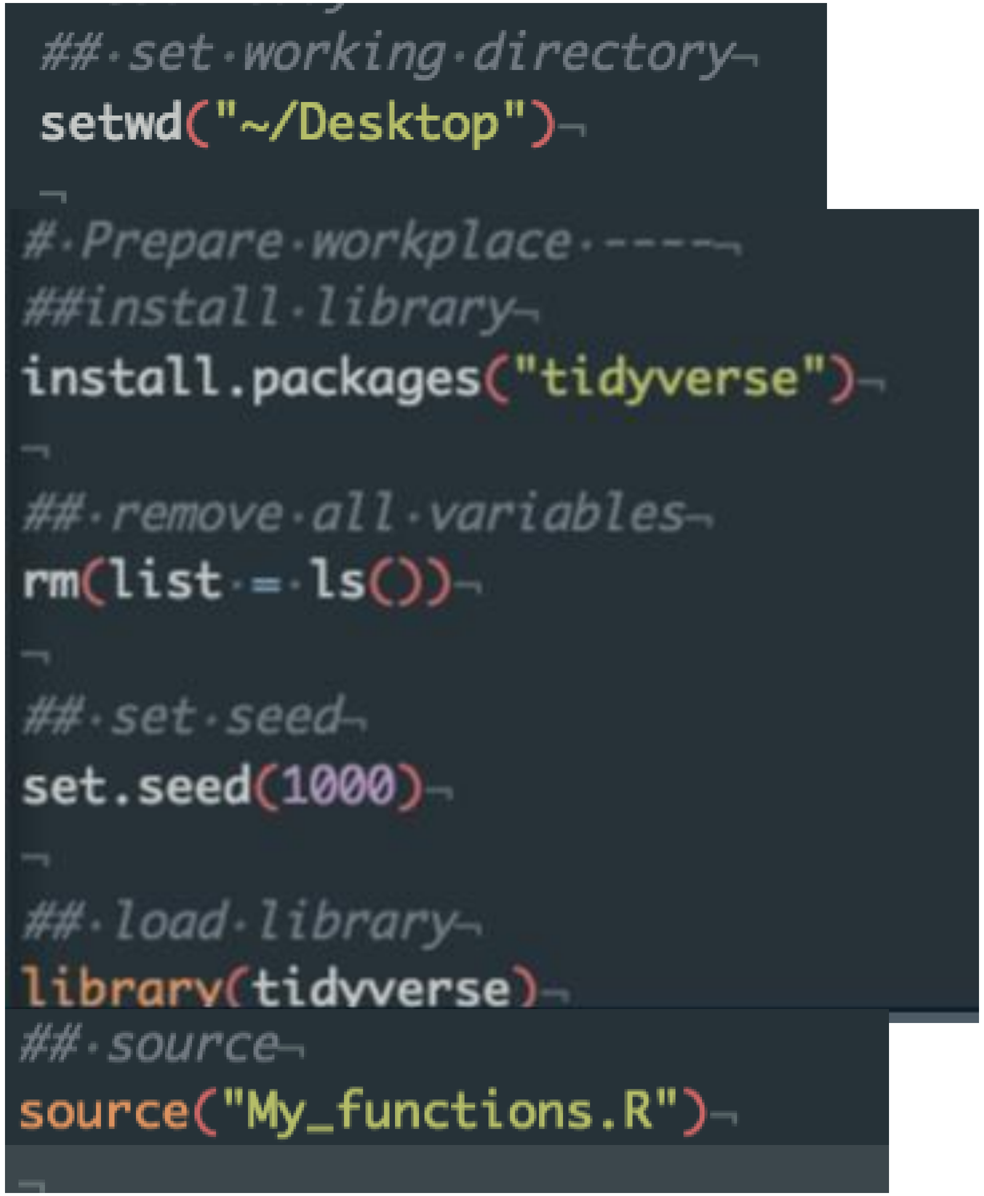
```
Tippen Sie 'demo()' für einige Demos, 'help()' für on-line Hilfe, oder
'help.start()' für eine HTML Browserschnittstelle zur Hilfe.
Tippen Sie 'q()', um R zu verlassen.

[Workspace loaded from ~/.RData]

Loading required package: ddCt
>
```

Copy-paste

R-scripts



```
## set working directory
setwd("~/Desktop")

# Prepare workplace ----
##install library
install.packages("tidyverse")

## remove all variables
rm(list = ls())

## set seed
set.seed(1000)

## load library
library(tidyverse)
## source
source("My_functions.R")

```

```

sessionInfo()
```

Import tables

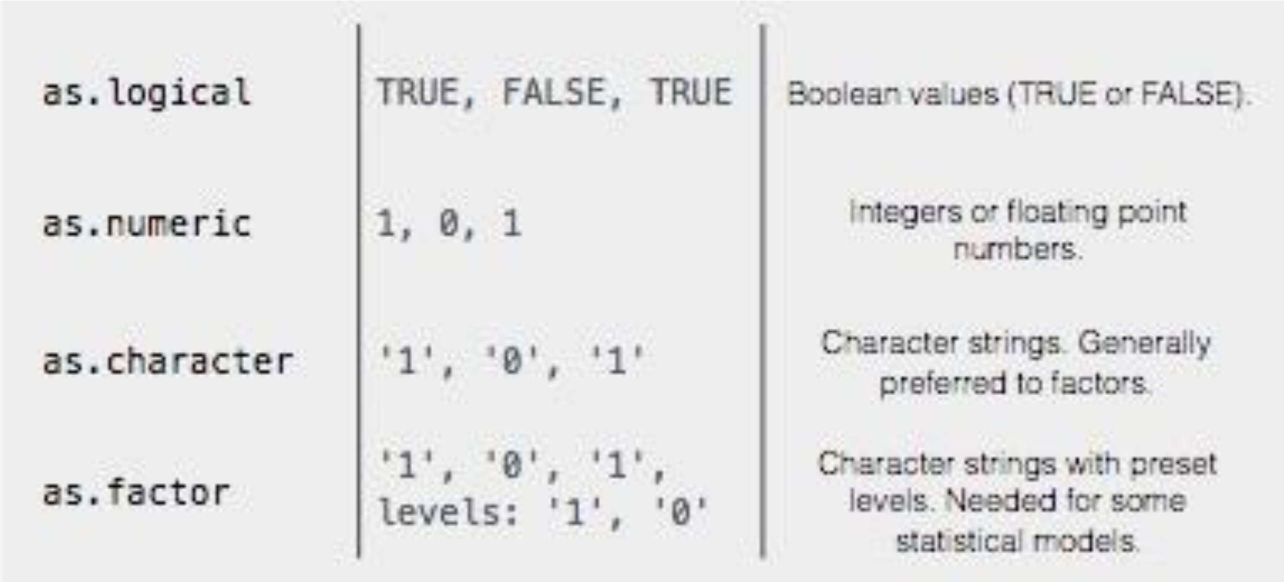
```
dat <- read.csv("http://gdc-web.ethz.ch/gdc-analysis-course/2019/data/Students19.txt", header = T, sep="\t")

dat <- read_tsv("http://gdc-web.ethz.ch/gdc-analysis-course/2019/data/Students19.txt")
```

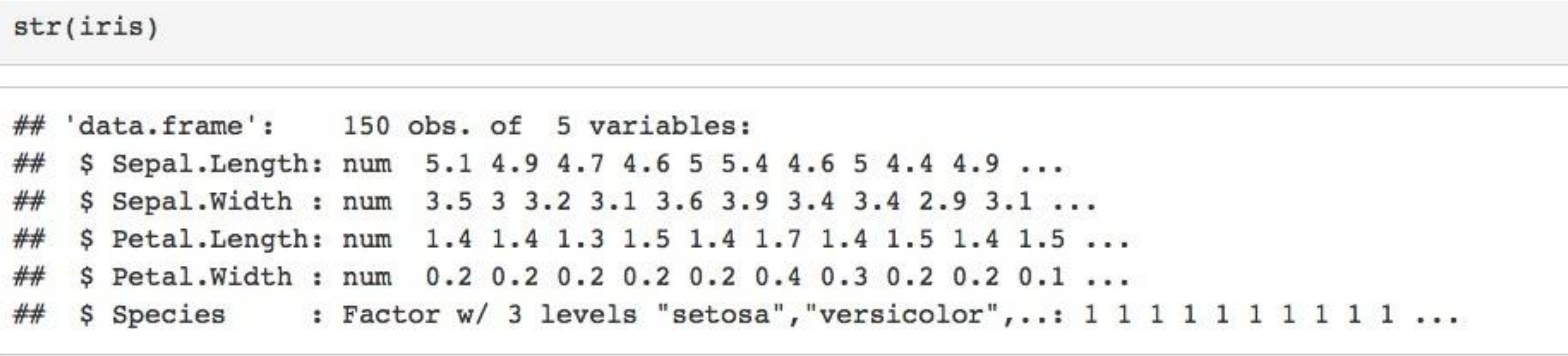
data.table::fread

R Objects

- Data frames
- Lists
- Vectors
- Matrixes



`as.logical`	`TRUE, FALSE, TRUE`	Boolean values (TRUE or FALSE).
`as.numeric`	`1, 0, 1`	Integers or floating point numbers.
`as.character`	`'1', '0', '1'`	Character strings. Generally preferred to factors.
`as.factor`	`'1', '0', '1',` `levels: '1', '0'`	Character strings with preset levels. Needed for some statistical models.

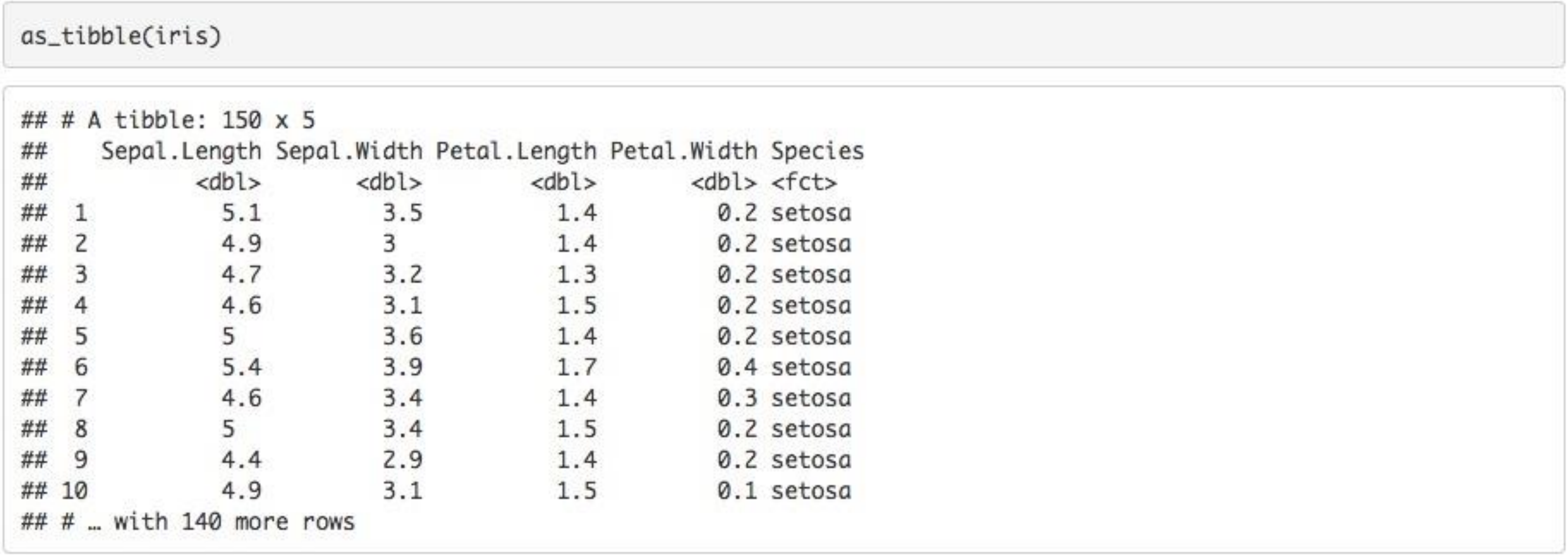


```
str(iris)
```

```
## '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     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

R Objects

tibble



```r
as_tibble(iris)
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            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           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # … with 140 more rows
```

Data Manipulation

```
iris.df <- data.frame(iris$Sepal.Length, iris$Sepal.Width, iris$Species)
```

Replace patterns

```
iris.df$Species2 <- gsub("setosa", "Setosa", iris.df$iris.Species)
```

subset

```
iris.df.sub <- subset(iris.df, iris.df$iris.Species == "setosa")
```

order table

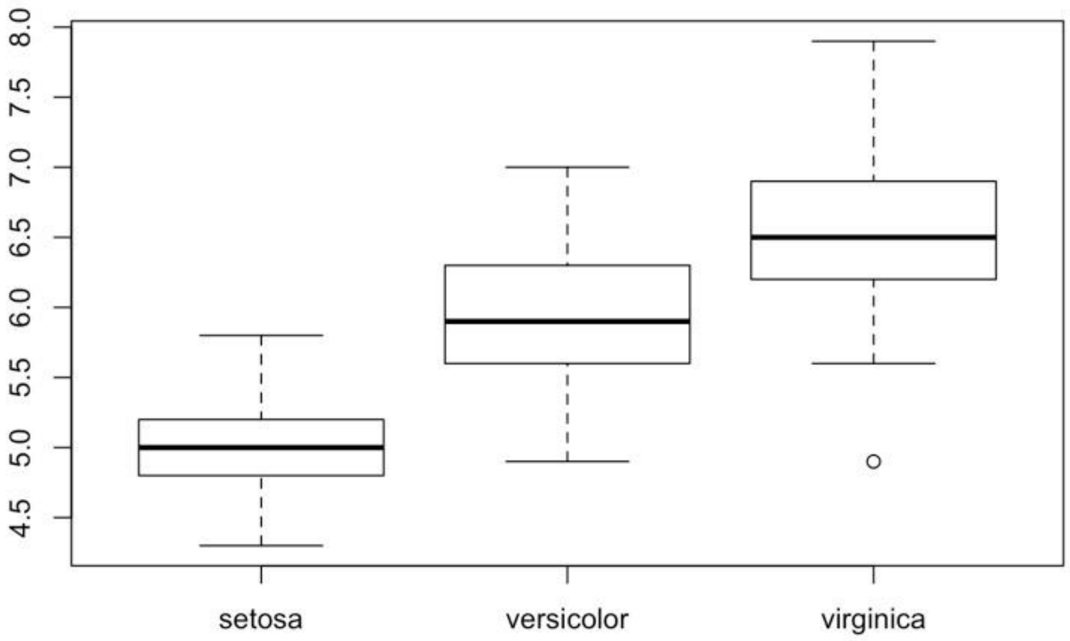
```
iris.df.orderd <- iris.df[order(iris.df$iris.Sepal.Length), ]
```

Data Manipulation

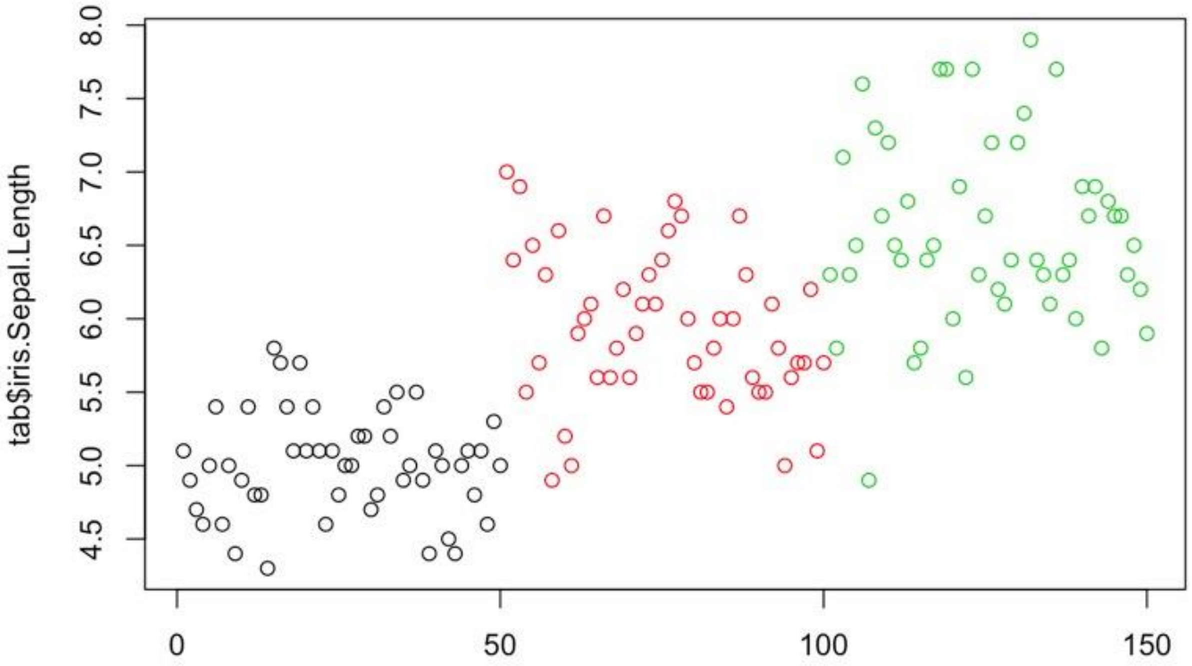
```
iris.tb <- as_tibble(iris) %>%
           select(Species, Sepal.Length, Sepal.Width, Petal.Length) %>%
           dplyr::filter(Species != "setosa") %>%
           mutate(Species2 = gsub("versicolor", "Versicolor", Species)) %>%
           arrange(., Sepal.Length)
```

Data visualization

```
boxplot(tab$iris.Sepal.Length~tab$iris.Species)
```

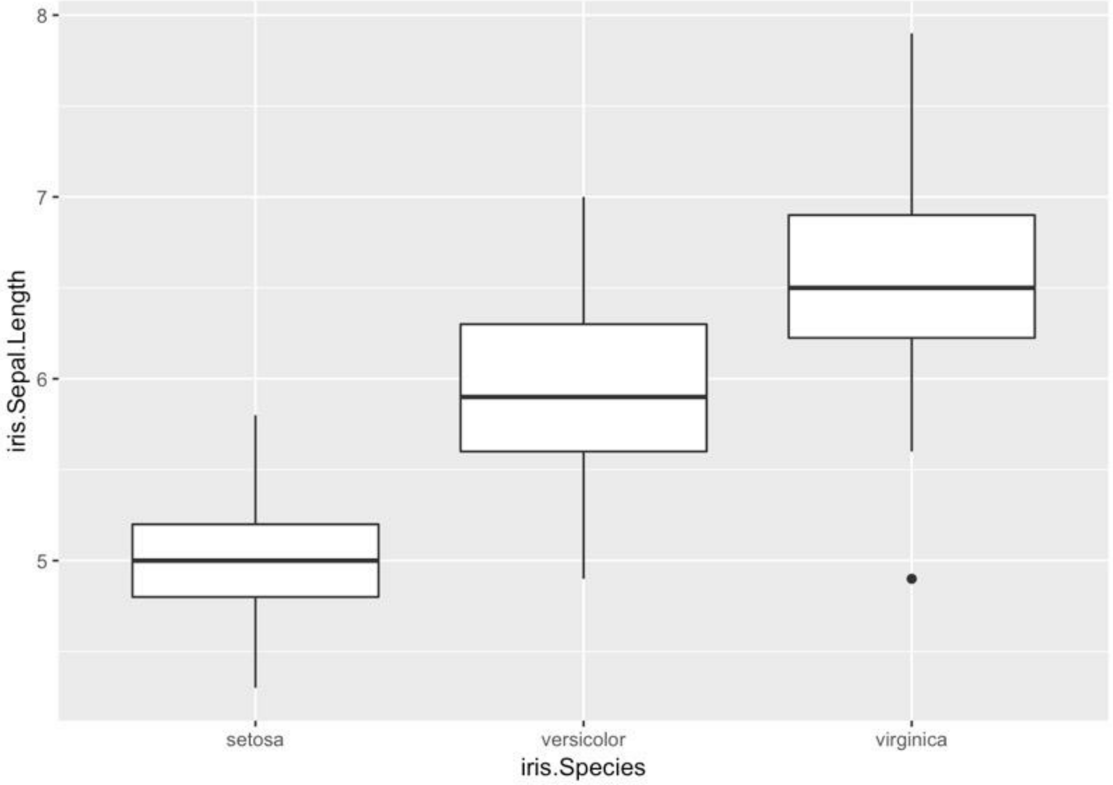


```
plot(tab$iris.Sepal.Length, col=tab$iris.Species)
```

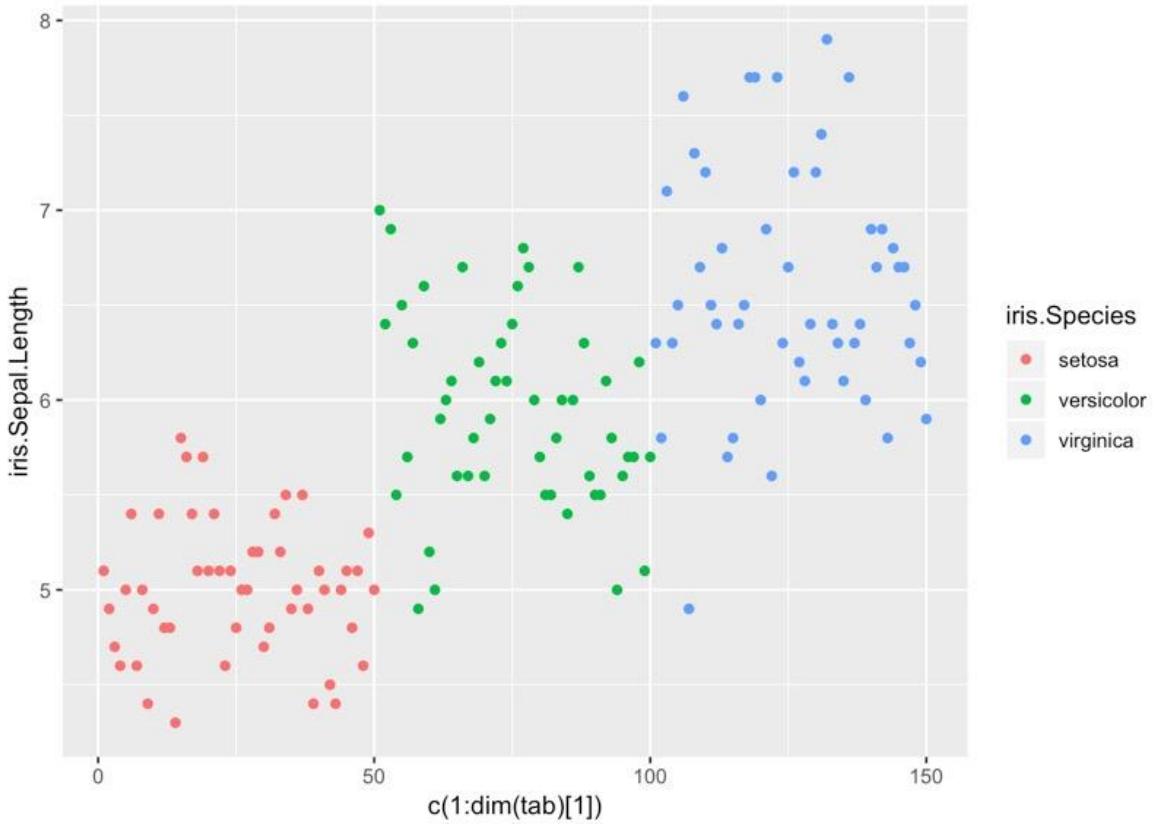


Visualization-ggplot2

```r
library(ggplot2)
ggplot(tab, aes(x=iris.Species,y=iris.Sepal.Length))+
  geom_boxplot()
```

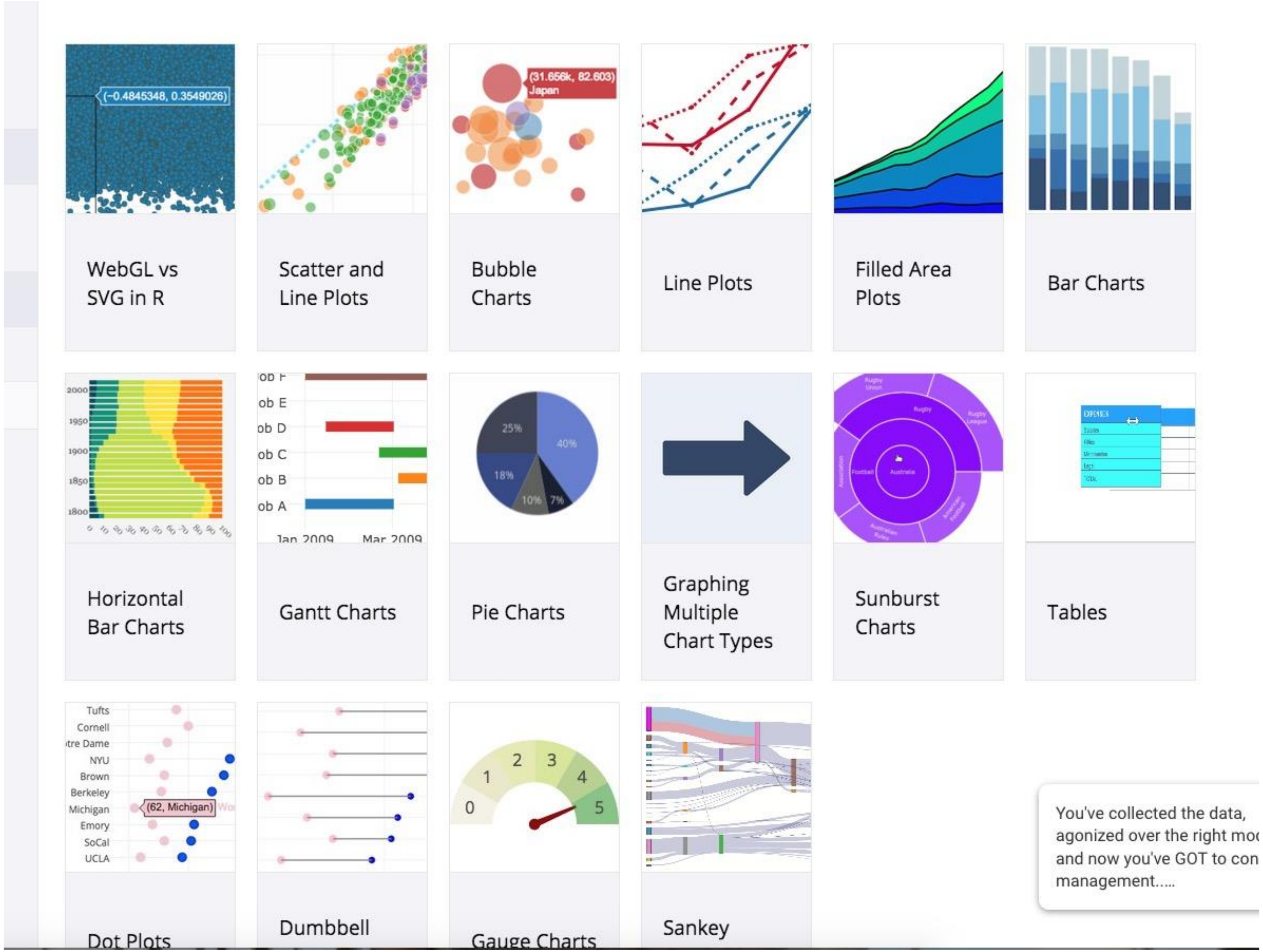


```r
ggplot(tab, aes(x=c(1:dim(tab)[1]),y=iris.Sepal.Length, colour=iris.Species))+
  geom_point()
```



https://www.r-graph-gallery.com/

Plotly

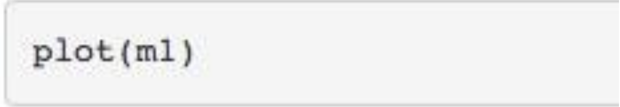


You've collected the data, agonized over the right mo
and now you've GOT to con
management.....

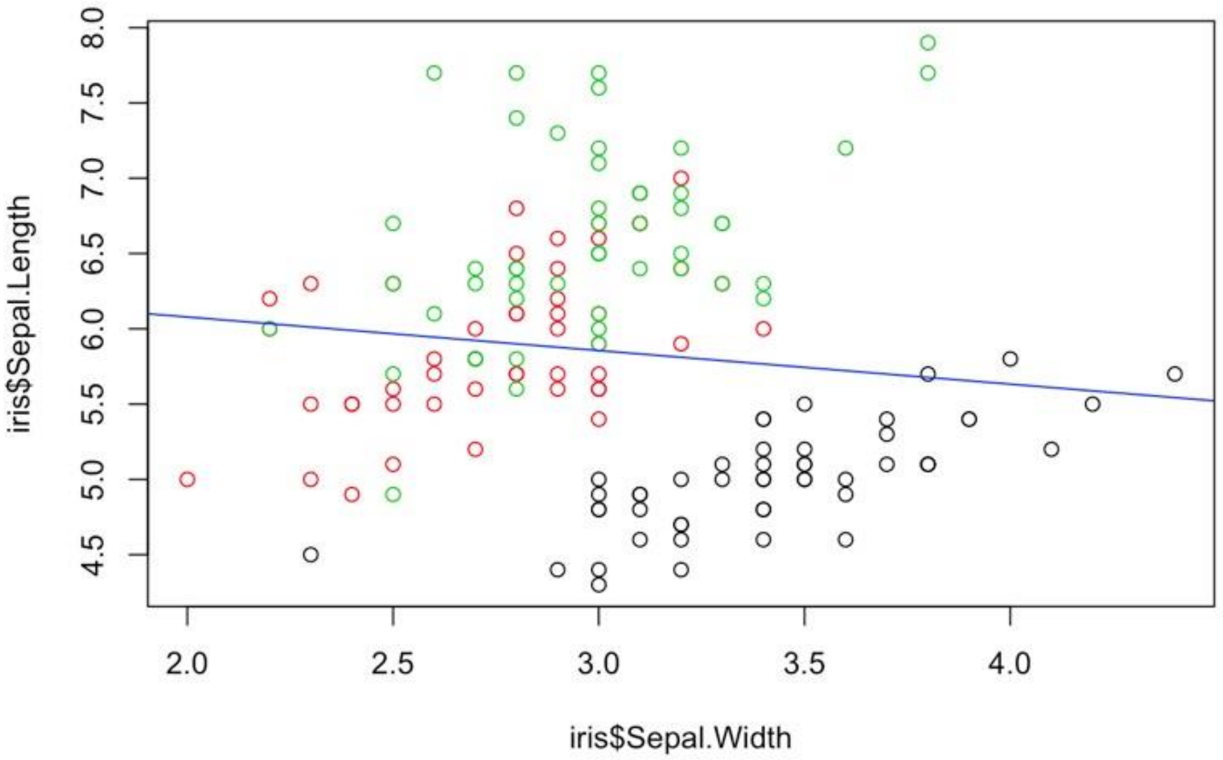
Statistics using R-Linear models

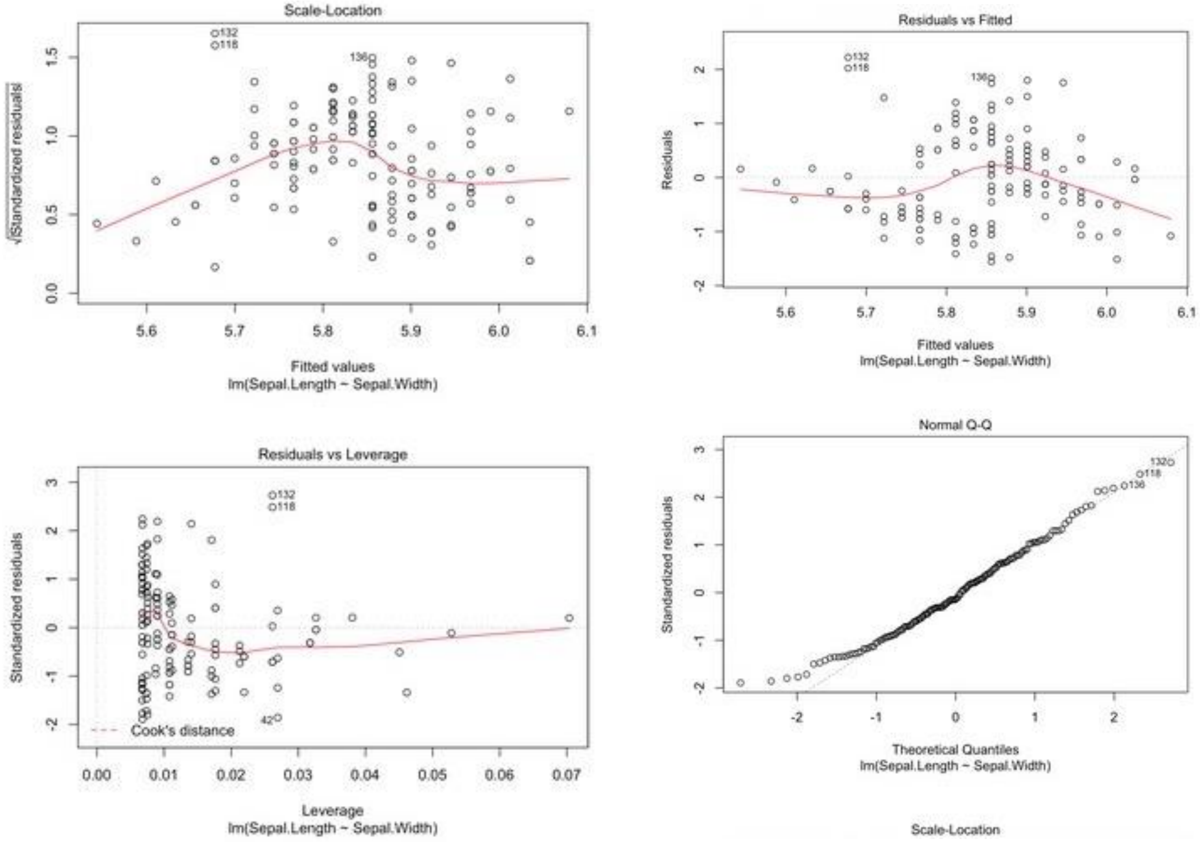
```
m1<-lm(Sepal.Length~Sepal.Width,data=iris)

plot(iris$Sepal.Length~iris$Sepal.Width, col=iris$Species)
abline(m1$coefficients, col="blue")
```

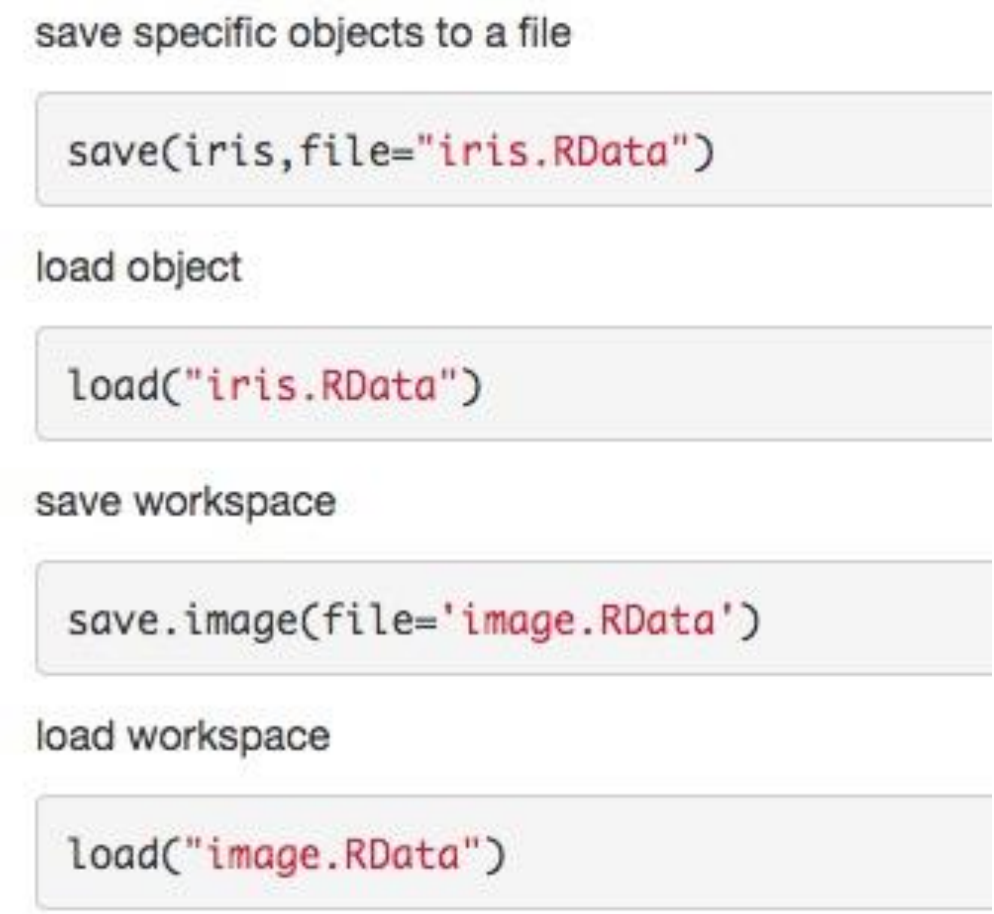


```
plot(m1)
```





Keep data



save specific objects to a file

```
save(iris,file="iris.RData")
```

load object

```
load("iris.RData")
```

save workspace

```
save.image(file='image.RData')
```

load workspace

```
load("image.RData")
```

Programming in R

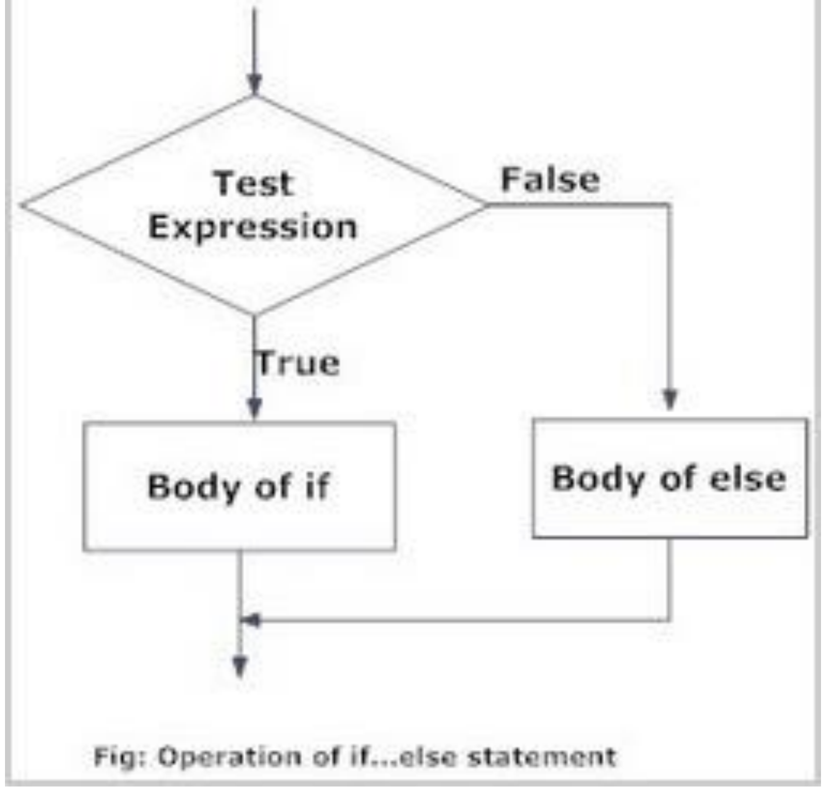
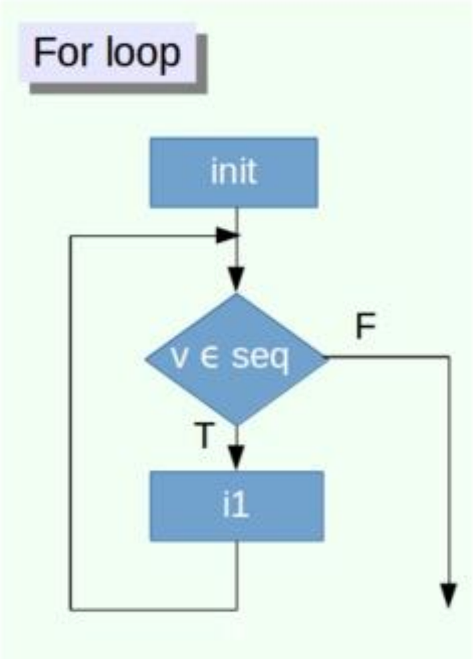


Fig: Operation of if...else statement

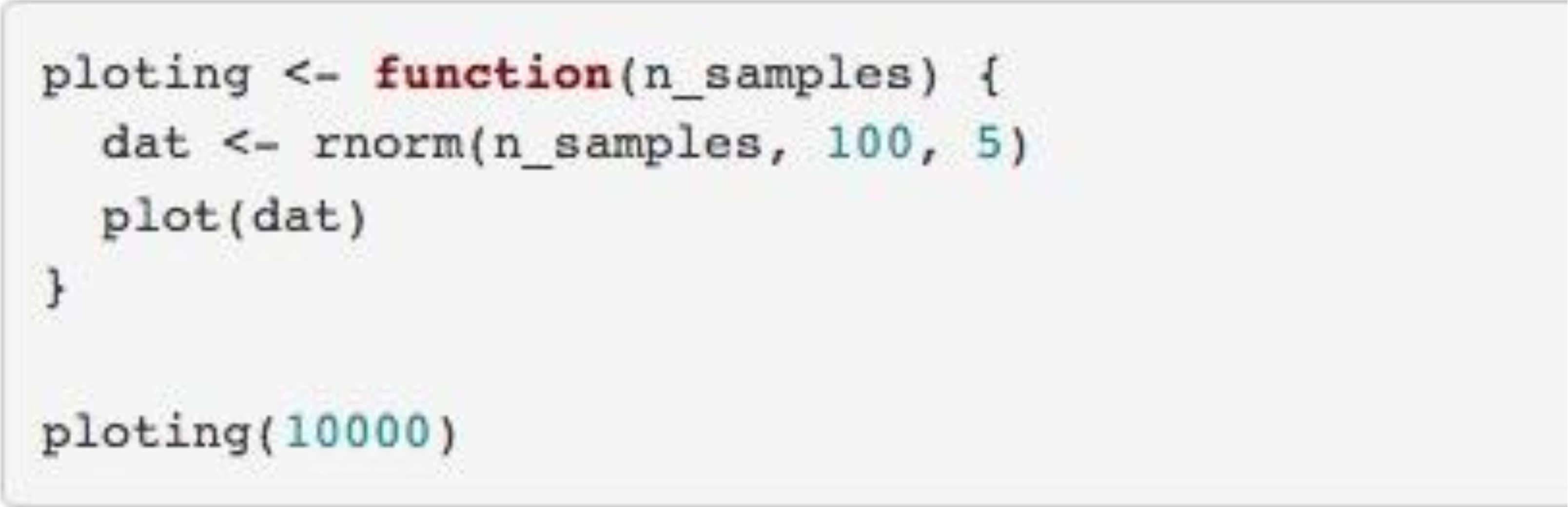


Hands-On Programming with R

https://rstudio-education.github.io/hopr/

Customized functions

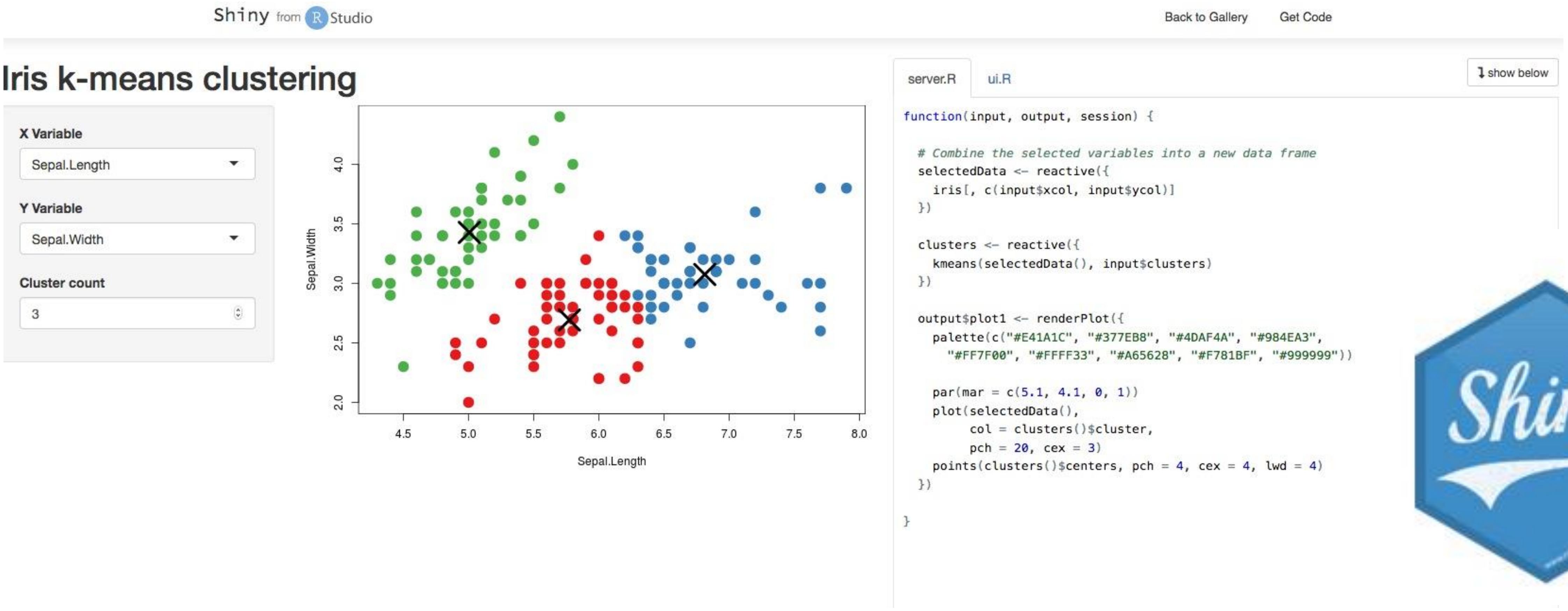
```
Myfunction <- function(variables) {
Function1
Function2
value/plot
}
```



```r
ploting <- function(n_samples) {
  dat <- rnorm(n_samples, 100, 5)
  plot(dat)
}

ploting(10000)
```

Shiny Apps



```
function(input, output, session) {

  # Combine the selected variables into a new data frame
  selectedData <- reactive({
    iris[, c(input$xcol, input$ycol)]
  })

  clusters <- reactive({
    kmeans(selectedData(), input$clusters)
  })

  output$plot1 <- renderPlot({
    palette(c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3",
      "#FF7F00", "#FFFF33", "#A65628", "#F781BF", "#999999"))

    par(mar = c(5.1, 4.1, 0, 1))
    plot(selectedData(),
         col = clusters()$cluster,
         pch = 20, cex = 3)
    points(clusters()$centers, pch = 4, cex = 4, lwd = 4)
  })

}
```

https://shiny.rstudio.com/gallery/kmeans-example.html

Customized packages

Dice example

Write an R script (R code) that creates a text file containing the results of 10 rolls of a pair of dice.

Take home message

- Fastly developing
- Many resources for biologists
- Also suitable for programming
- AI tools

Bioinformatics and computational biology, often referred to as "bioinformatics" or "biocomputing," is the application of computer science and technology to the management, analysis, and interpretation of biological data. This can include DNA sequences, protein structures, and genetic and metabolic pathways, as well as data from high-throughput experiments such as genome-wide association studies, transcriptomics, and proteomics.

R is a popular programming language and environment for data analysis that is widely used in bioinformatics and other fields. There are many libraries and packages available in R that are specifically designed for bioinformatics, such as Bioconductor, which provides tools for the analysis of genomic data, and rOpenSci, which provides tools for working with various bioinformatics data resources, such as the National Center for Biotechnology Information (NCBI) databases.

There are many R packages for bioinformatics and computational biology for tasks like sequence alignment, assembly, annotation, gene expression analysis, functional analysis, and more. Some popular bioinformatics R package include:

Biostrings: for working with DNA and RNA sequences

BSgenome: for working with genome assemblies

Bioconductor: It's a collection of R packages for the analysis of genomic data.

ShortRead: for quality control and preprocessing of high-throughput sequencing data

edgeR, DESeq2: for identifying differentially expressed genes from RNA-seq data

GAGE, GOstats, topGO for functional analysis of gene sets

R can also be used to visualize bioinformatics data using packages like ggplot2, heatmap3, and ComplexHeatmap for heatmap visualization, igraph for network visualization, and rgl for 3D visualization of molecular structures.

Overall, R provides a powerful and flexible platform for bioinformatics and computational biology research, with a wide range of tools and resources available for managing, analyzing, and interpreting large and complex biological data sets.