Biocomputing with R

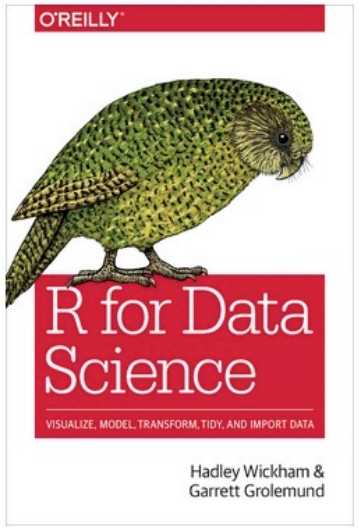
Niklaus Zemp

21 June 2022

Genetic Diversity Centre (GDC)
Bioinformatics
ETH Zurich

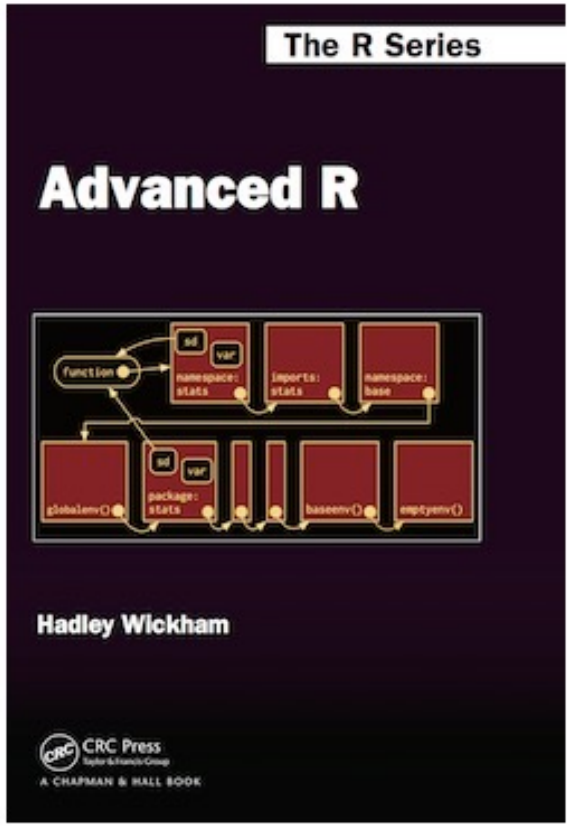
Resources

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



R-bloggers

R news and tutorials contributed by (750) R bloggers



Packages

Available Packages

Currently, the CRAN package repository features 18607 available packages.

Table of available packages, sorted by date of publication

Table of available packages, sorted by name

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Install »

- Discover 2140 software packages available in *Bioconductor* release 3.15.

Get started with *Bioconductor*

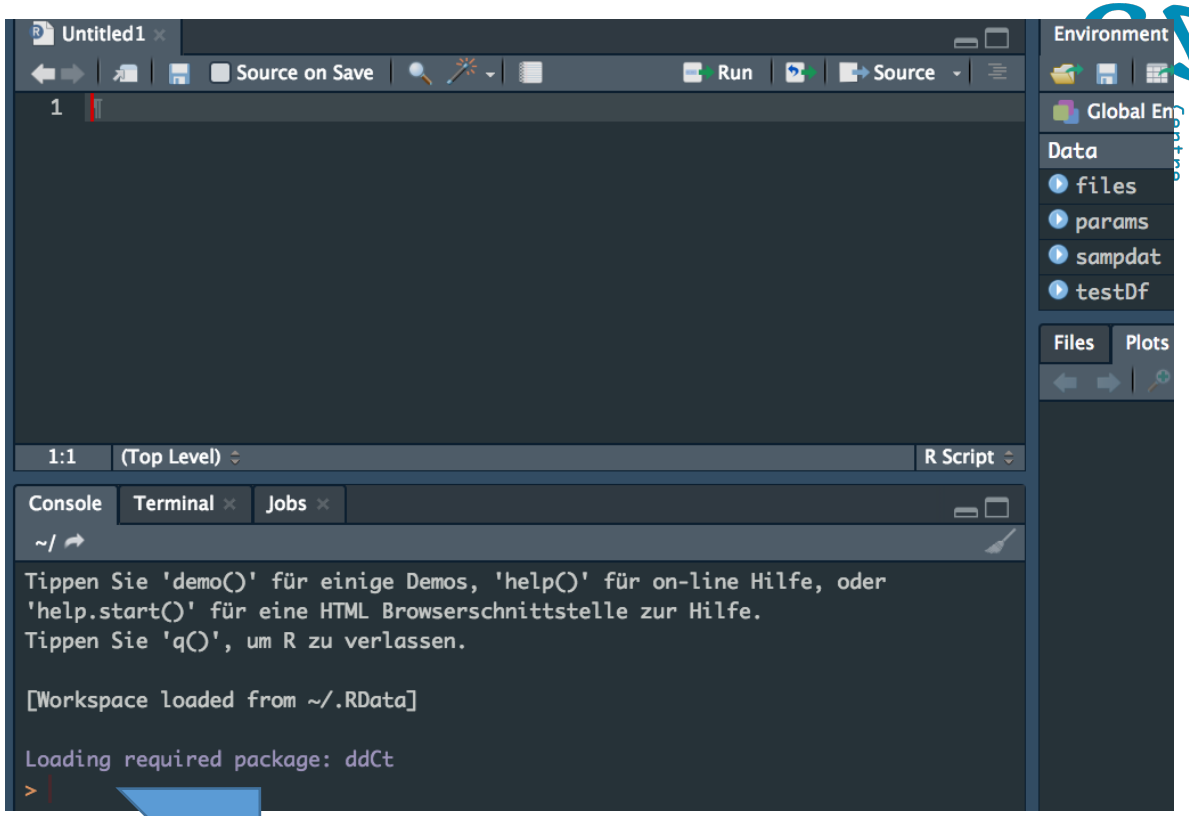
- Install *Bioconductor*
- Get support
- Latest newsletter
- Follow us on twitter
- Install R

Rstudio

netic Diversity Centre (GDC) -
urse Webpage

art
equirements
nux 1 - Local Terminal
nux 2 - Remote Terminal
ocomputing
ocomputing with R
ocomputing on a HPC cluster
eproducible 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")
```

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")¬
¬
```

```
4  ¬
5  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

```
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

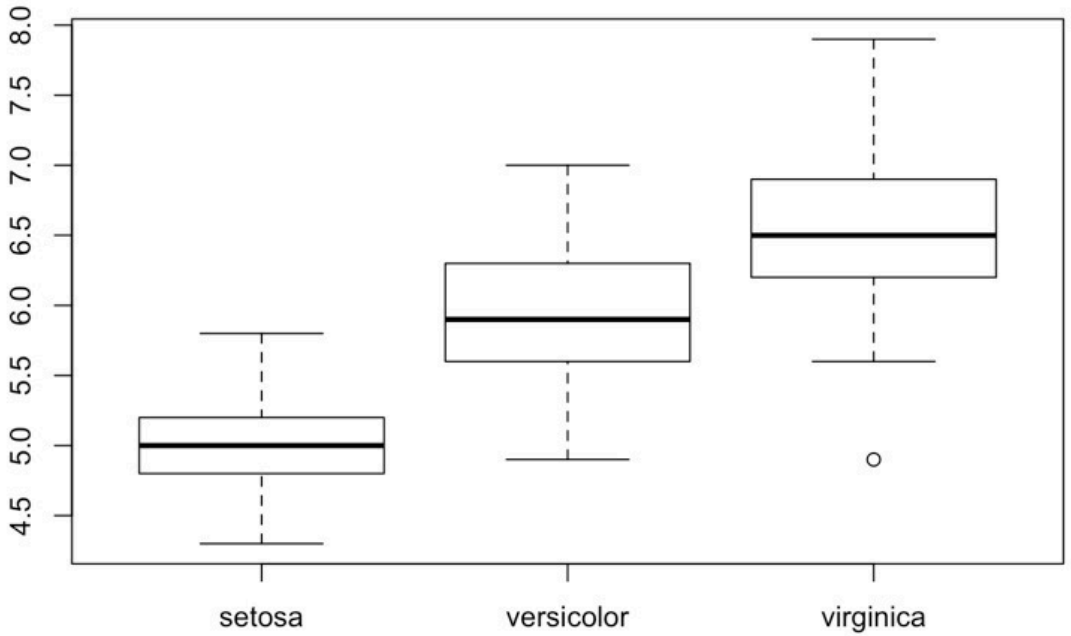
```
iris.df.orderd <- iris.df[order(iris.df$iris.Sepal.Length), ]
```

Data Manipulation

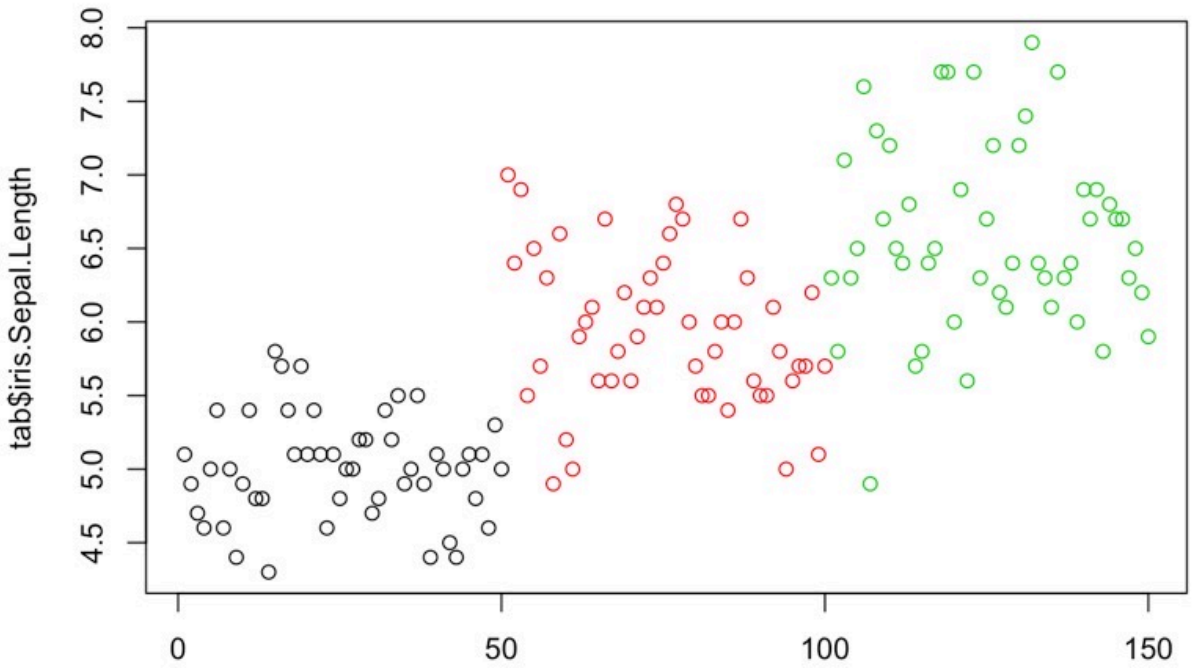
```
iris.tb <- as_tibble(iris) %>%
           select(Species, Sepal.Length, Sepal.Width, Petal.Length) %>%
           dplyr::filter(Species != "setosa") %>%
           mutate(Species2 = gsub("versicolor", "Versicolor", Species)) %>%
           arrange(., Sepal.Length)
```

Data visualization

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

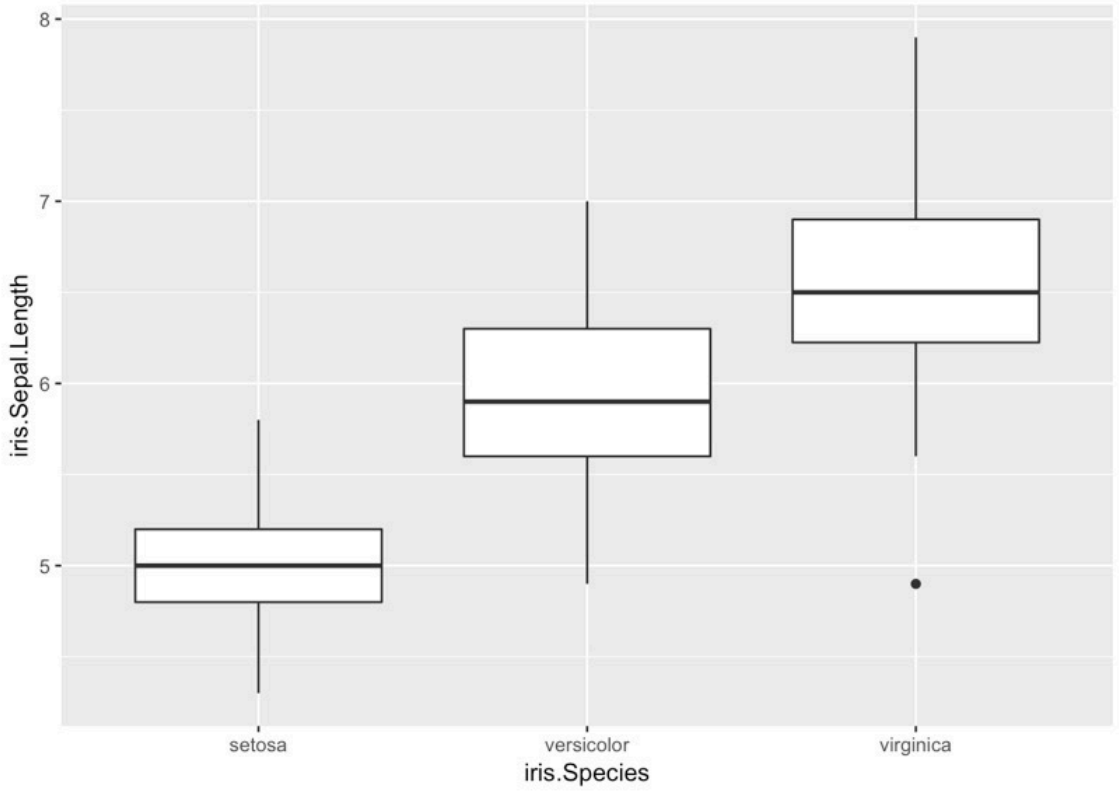



```
plot(tab$iris.Sepal.Length, col=tab$iris.Species)
```

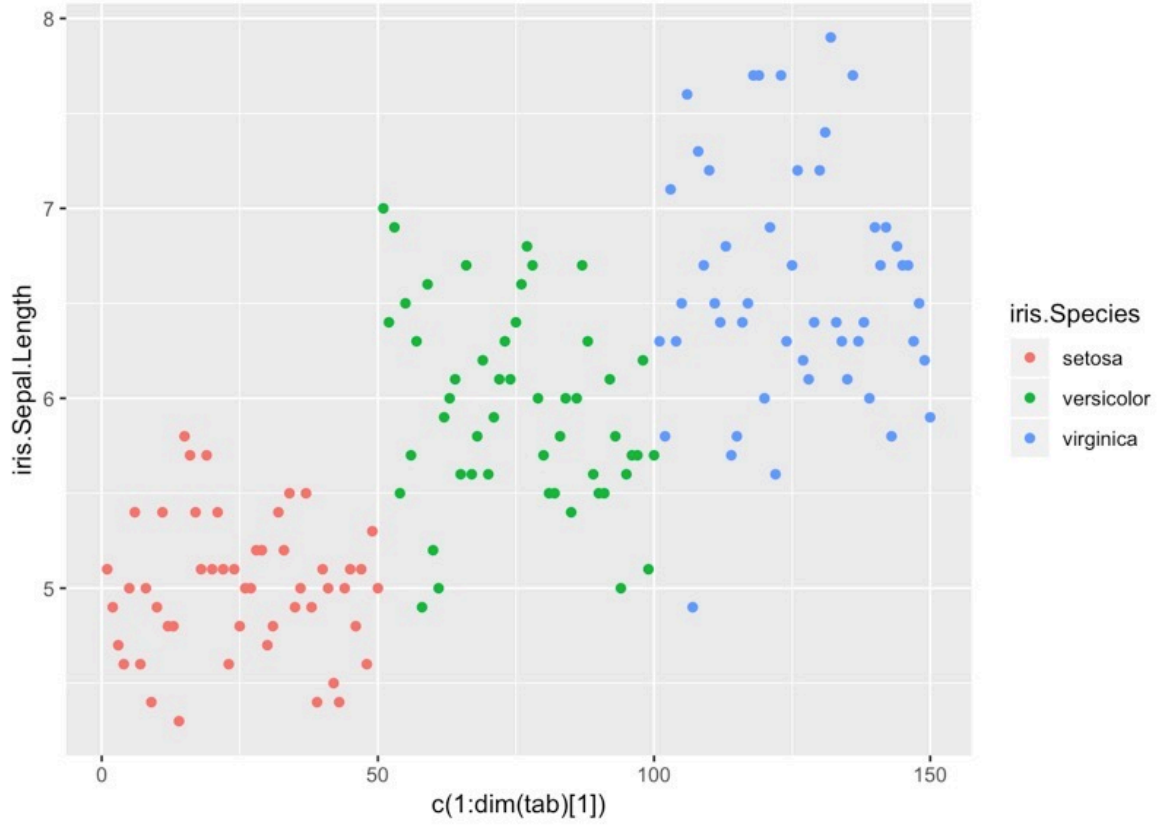


Visualization-ggplot2

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

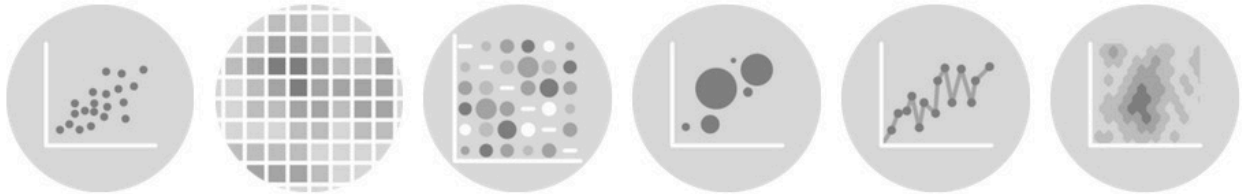



```
ggplot(tab, aes(x=c(1:dim(tab)[1]),y=iris.Sepal.Length, colour=iris.Species))+
  geom_point()
```



Distribution

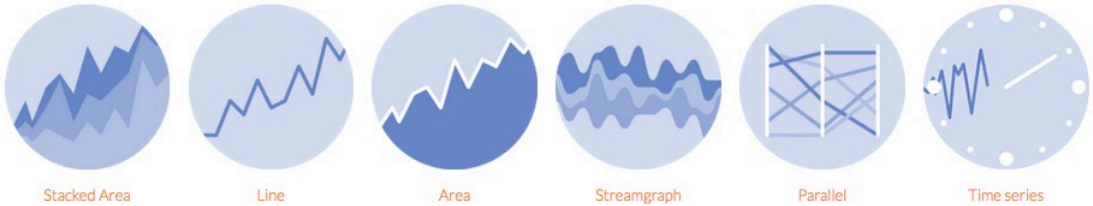
Correlation



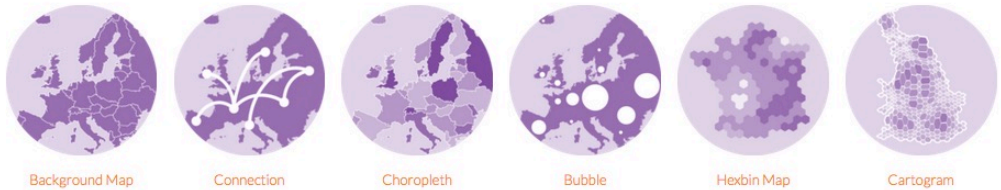
Rankings

Part of a whole

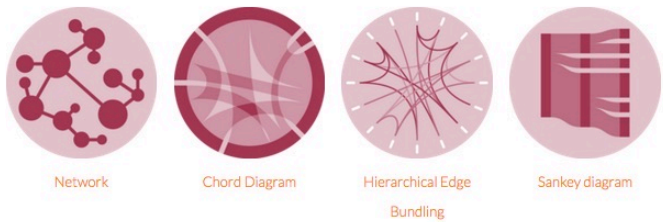
Evolution



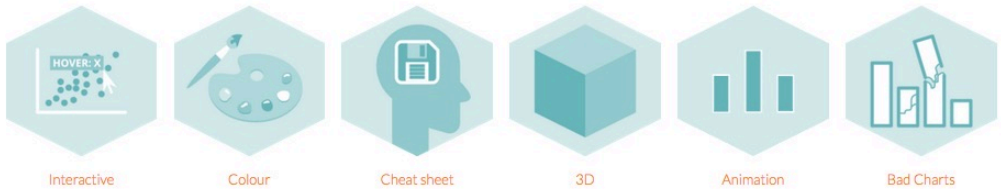
Maps



Flow

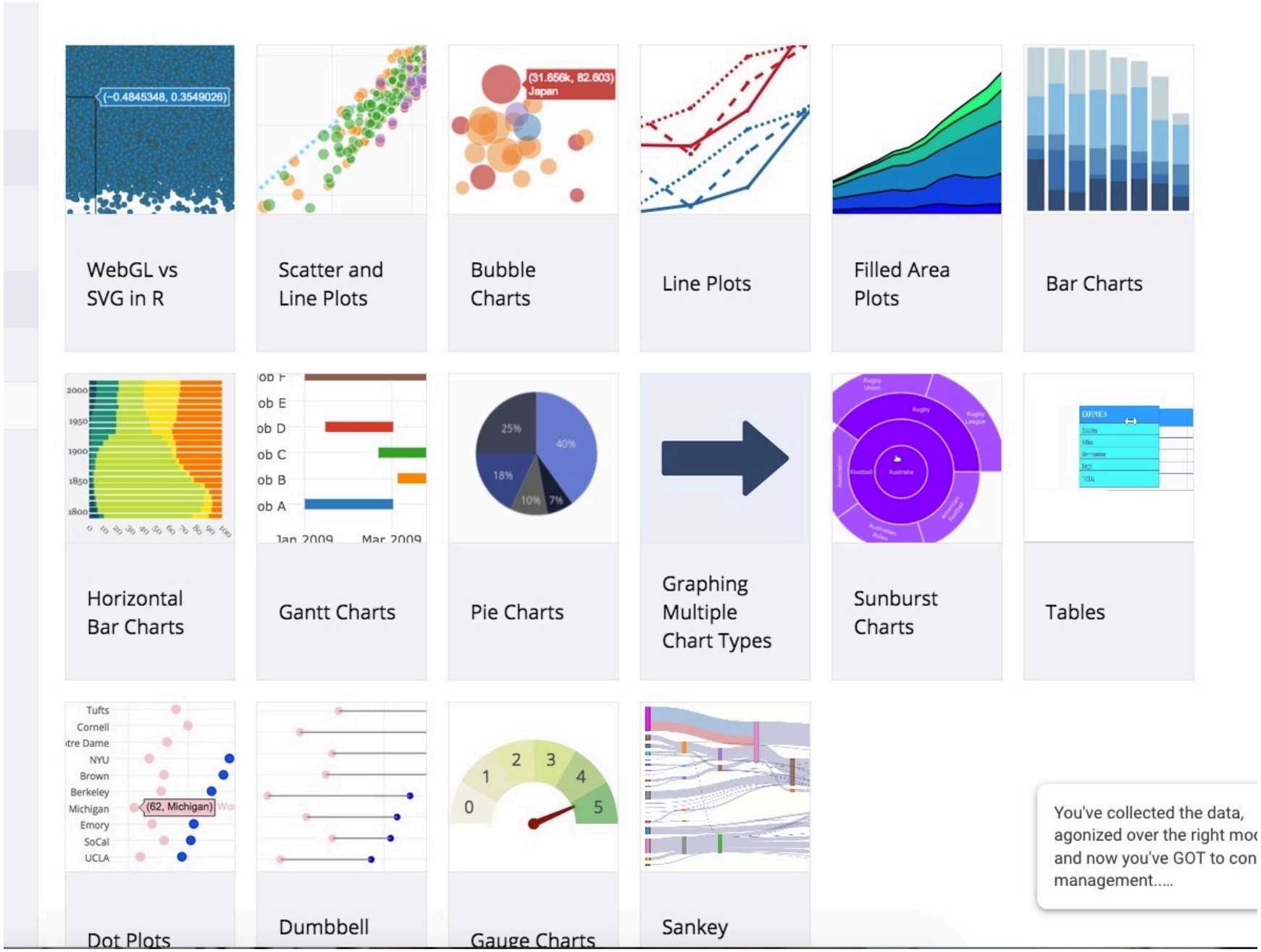


Other



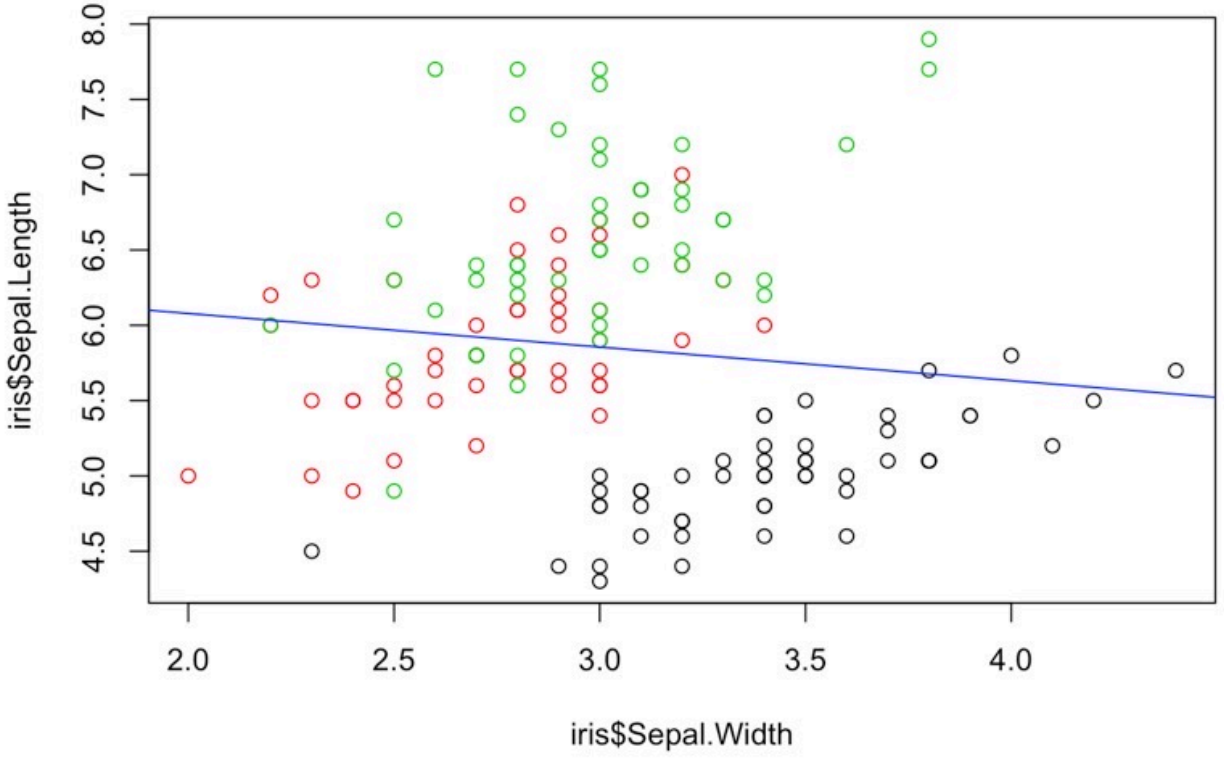
https://www.r-graph-gallery.com/

Plotly

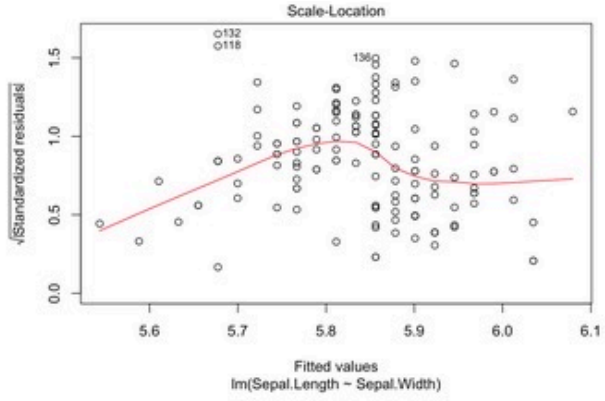

(−0.4845348, 0.3549026)
WebGL vs SVG in R
Scatter and Line Plots
(31.656k, 82.603)
Japan
Bubble Charts
Line Plots
Filled Area Plots
Bar Charts
Horizontal Bar Charts
ob E
ob D
ob C
ob B
ob A
Jan 2009
Mar 2009
Gantt Charts
25%
40%
18%
10%
7%
Pie Charts
Graphing Multiple Chart Types
Sunburst Charts
Tables
Tufts
Cornell
NYU
Brown
Berkeley
Michigan
Emory
SoCal
UCLA
(62, Michigan)
Dot Plots
Dumbbell
0
1
2
3
4
5
Gauge Charts
Sankey
You've collected the data,
agonized over the right mod
and now you've GOT to con
management.....

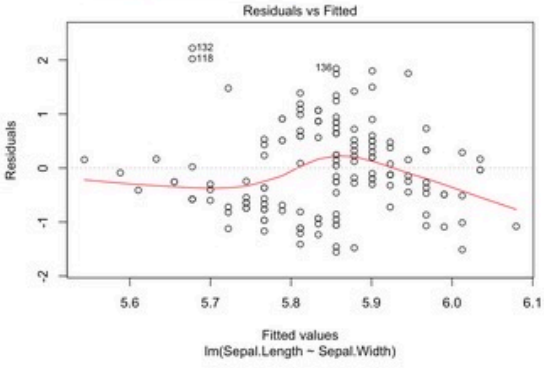
Statistics using R-Linear models

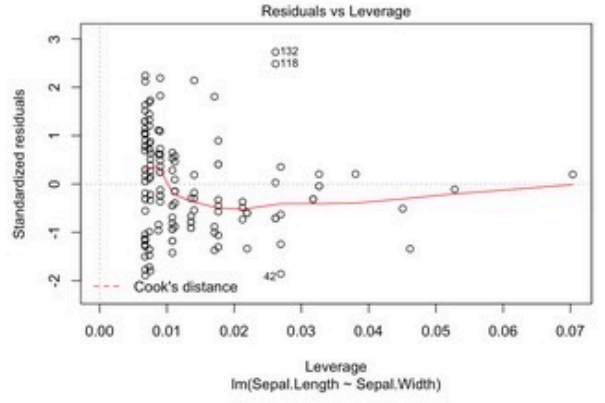
```
m1<-lm(Sepal.Length~Sepal.Width,data=iris)

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

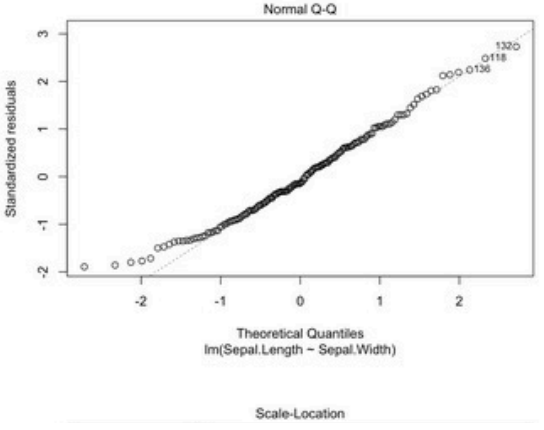



```
plot(m1)
```









Keep data

save specific objects to a file

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

load object

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

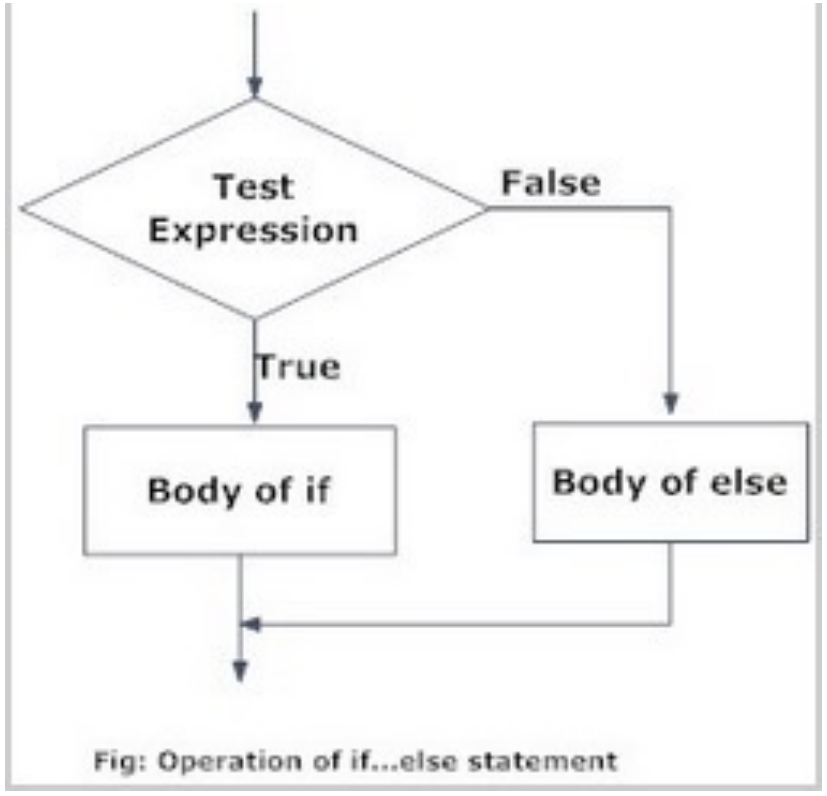
save workspace

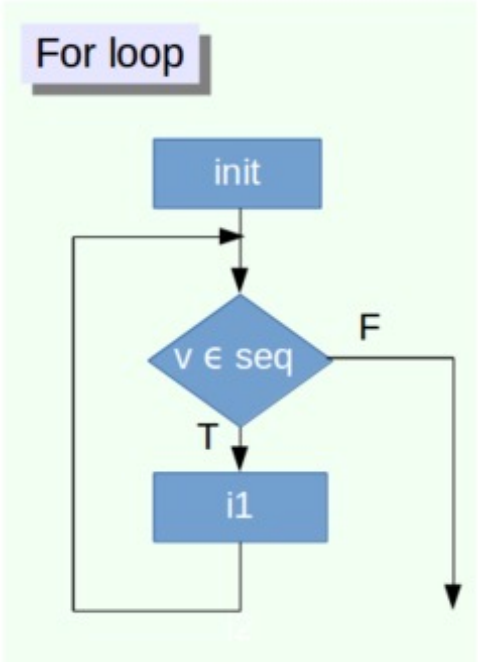
```
save.image(file='image.RData')
```

load workspace

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

Programming in R

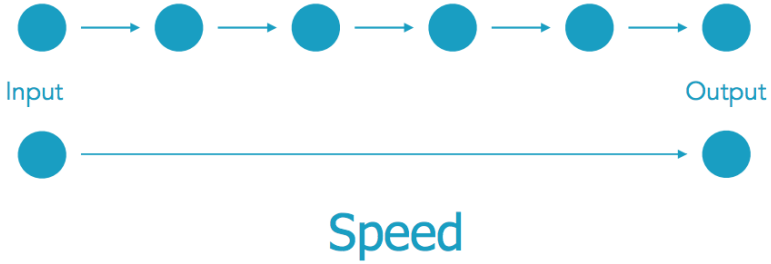




Hands-On Programming with R

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

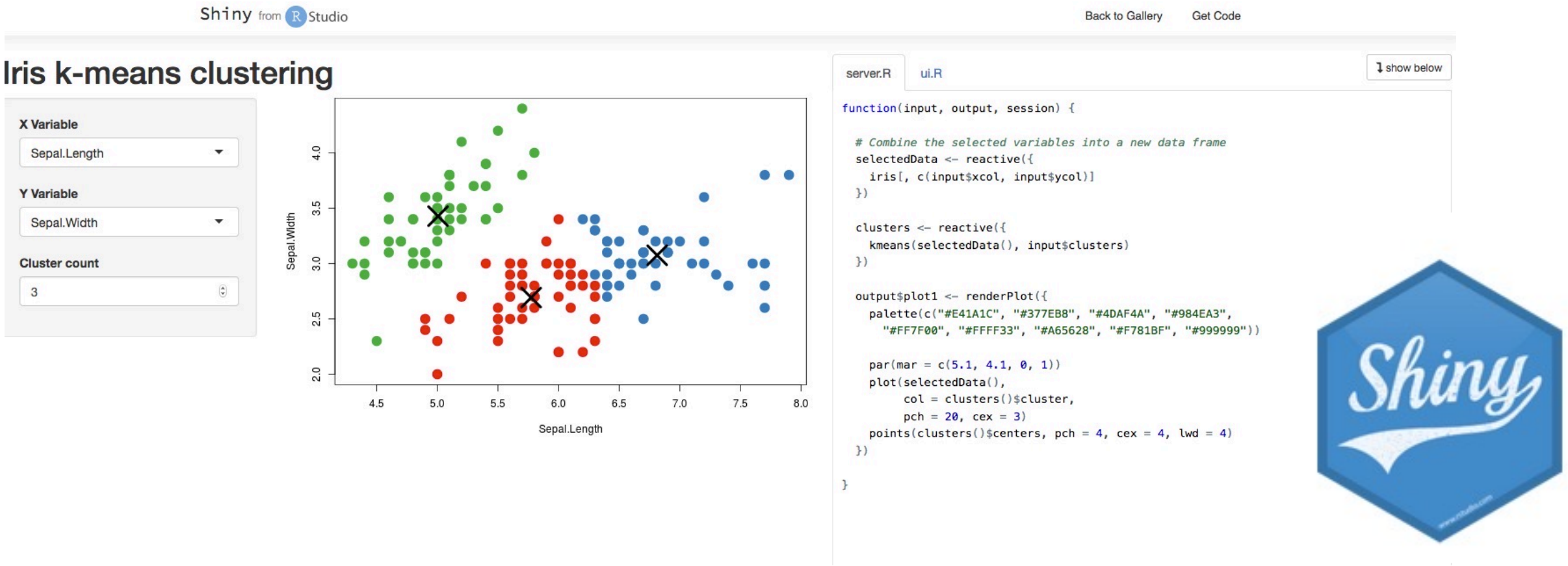
Customized functions



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

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

ploting(10000)
```

Shiny Apps



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

Customized packages

Take home massage

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