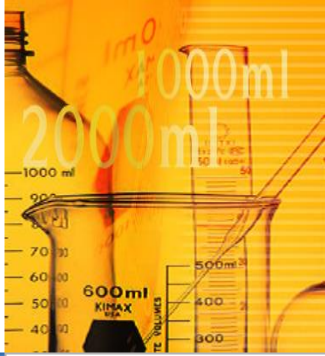
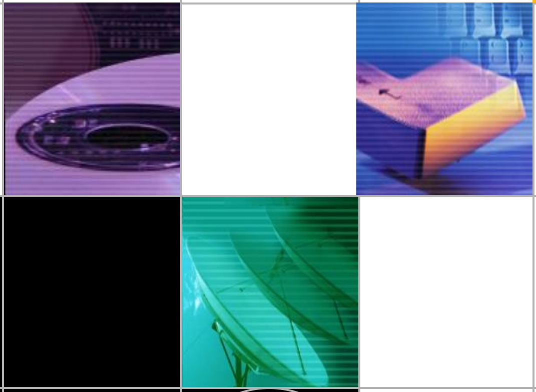
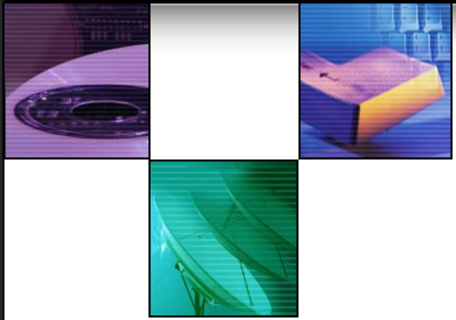
Machine learning





Chapter 12

Data visualization (2)

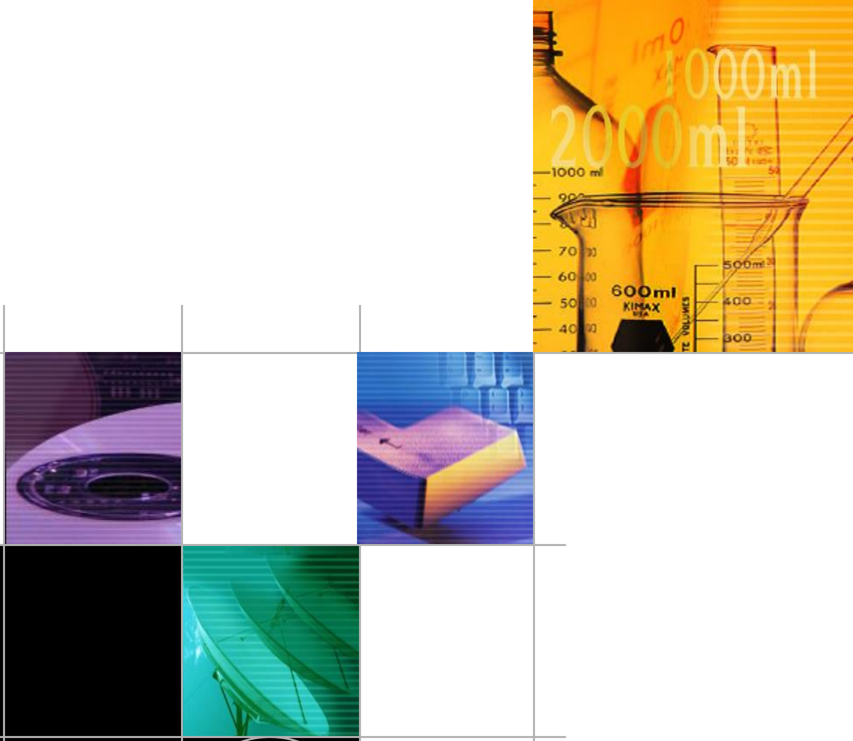


Sejong Oh

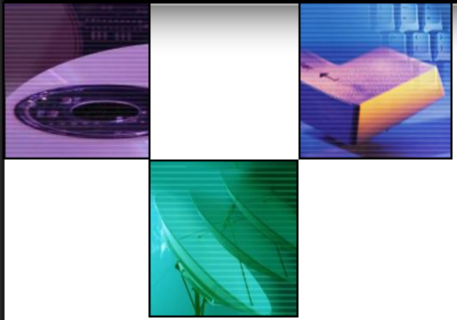
Bio Information Technology Lab.

Contents

- Data table
- ggplot
- Mosaic plot
- 지도상에 데이터 표현

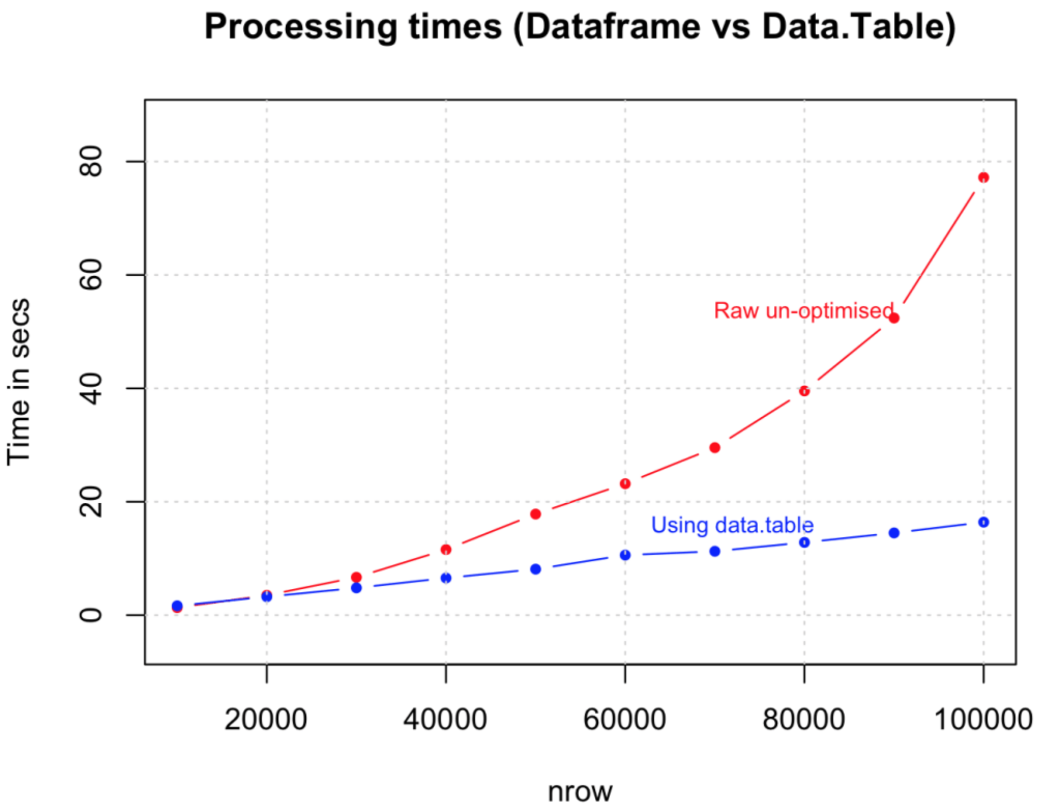


Data table



Data table

- Data frame 과 유사
- Data frame 보다 처리 속도가 더 빠르고 편리함
- Big size data 를 다루는 경우는 data table 추천

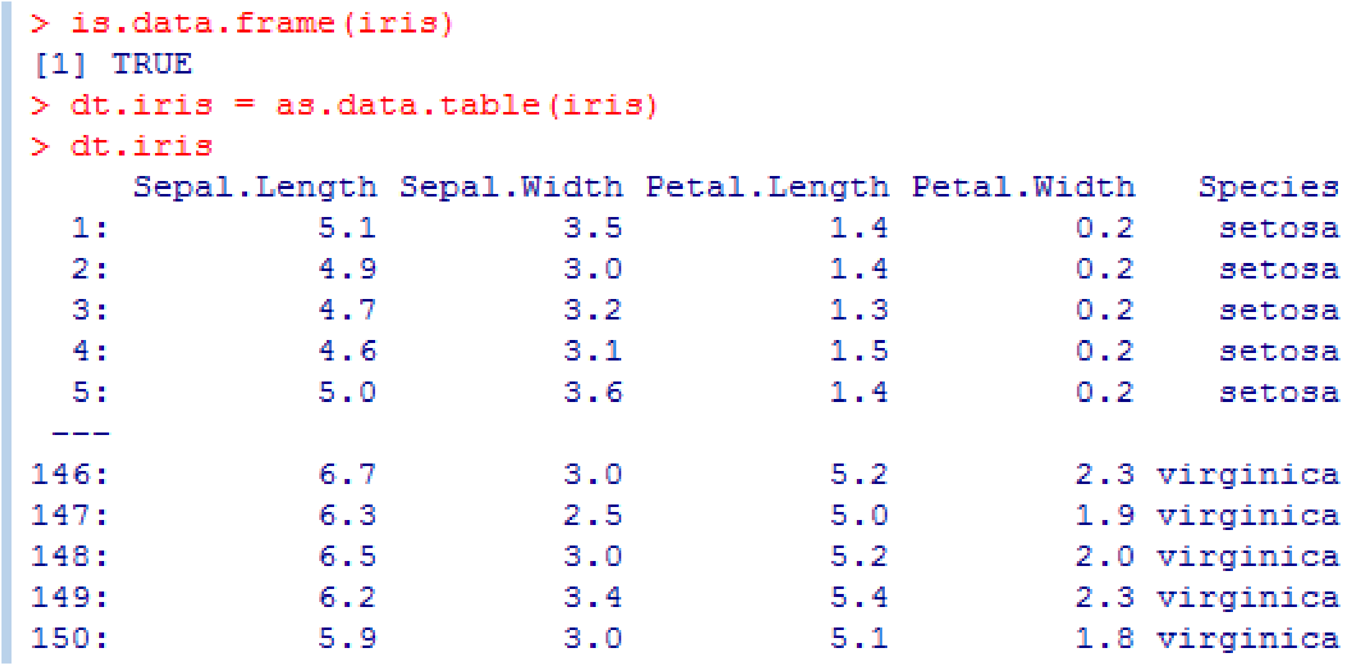


https://www.r-bloggers.com/strategies-to-speedup-r-code/

Data table

- 데이터 테이블 생성

```
library(data.table)
is.data.frame(iris)
dt.iris = as.data.table(iris)
dt.iris
```

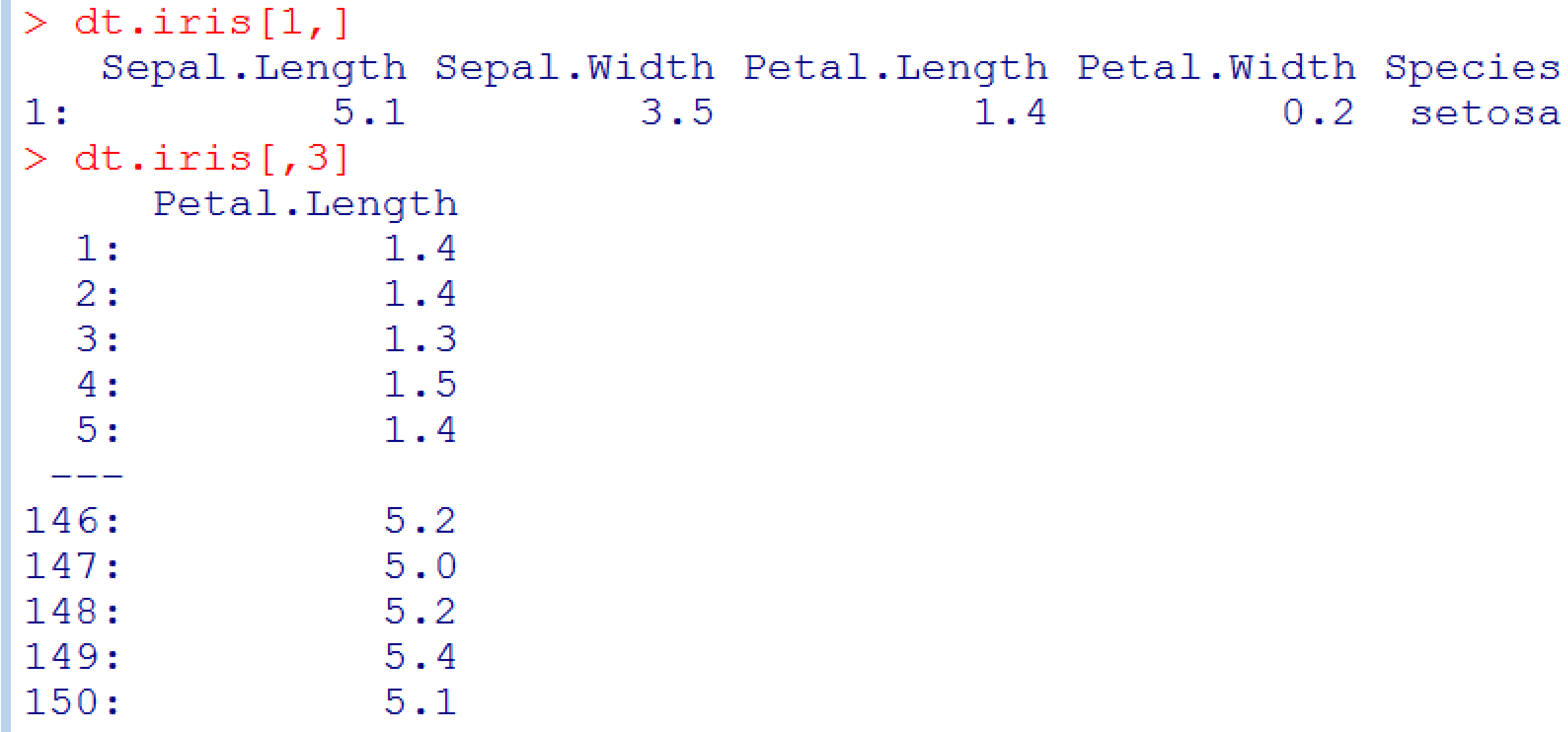


```
> is.data.frame(iris)
[1] TRUE
> dt.iris = as.data.table(iris)
> dt.iris
     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
  1:          5.1         3.5          1.4         0.2    setosa
  2:          4.9         3.0          1.4         0.2    setosa
  3:          4.7         3.2          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
 ---
146:          6.7         3.0          5.2         2.3 virginica
147:          6.3         2.5          5.0         1.9 virginica
148:          6.5         3.0          5.2         2.0 virginica
149:          6.2         3.4          5.4         2.3 virginica
150:          5.9         3.0          5.1         1.8 virginica
```

Data table

- 데이터 테이블 연산

```
dt.iris[1,]
dt.iris[,3]
dt.iris$Species
dt.iris[dt.iris$Species=="setosa", ]
```

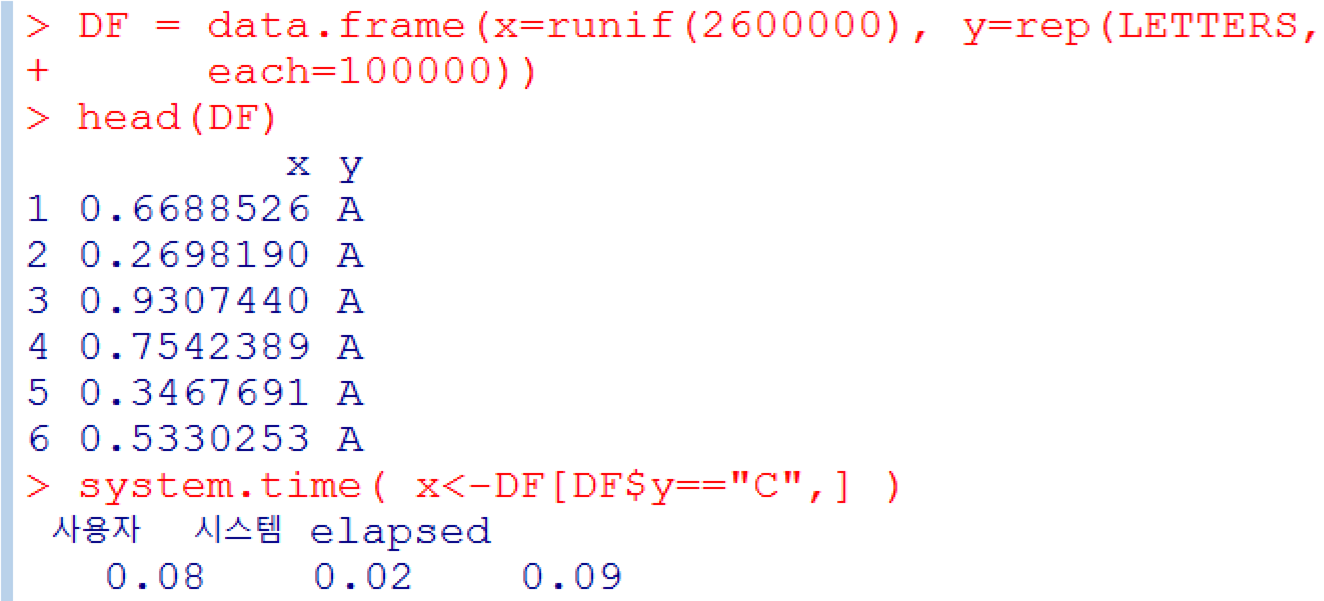


```
> dt.iris[1,]
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1:          5.1         3.5          1.4         0.2  setosa
> dt.iris[,3]
     Petal.Length
  1:          1.4
  2:          1.4
  3:          1.3
  4:          1.5
  5:          1.4
 ---
146:          5.2
147:          5.0
148:          5.2
149:          5.4
150:          5.1
```

Data table

- 검색을 위한 index의 설정
  - 데이터가 많고 검색이 빈번하게 일어나는 경우 key 를 설정해 두면 검색을 빠르게 할 수 있다.

```
DF = data.frame(x=runif(2600000), y=rep(LETTERS,
        each=100000))
head(DF)
system.time( x<-DF[DF$y=="C",] )
```

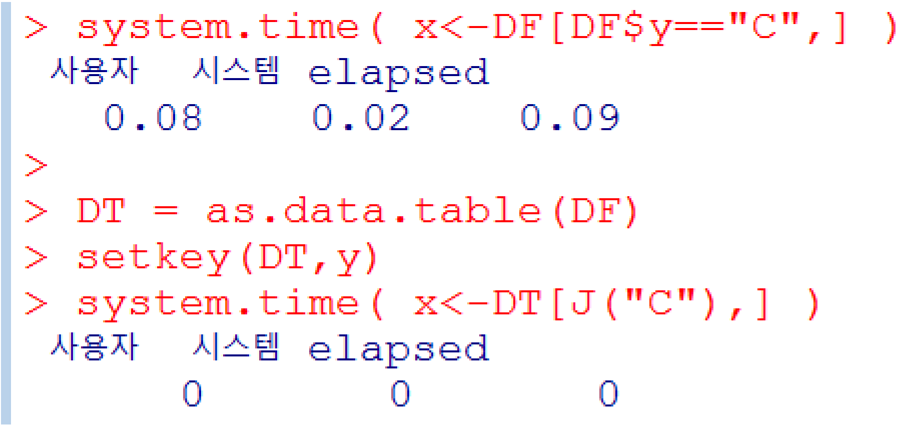


```
> DF = data.frame(x=runif(2600000), y=rep(LETTERS,
+          each=100000))
> head(DF)
          x y
1 0.6688526 A
2 0.2698190 A
3 0.9307440 A
4 0.7542389 A
5 0.3467691 A
6 0.5330253 A
> system.time( x<-DF[DF$y=="C",] )
 사용자   시스템 elapsed 
   0.08    0.02    0.09 
```

Data table

- 검색을 위한 index의 설정

```
DT = as.data.table(DF)
setkey(DT,y)
system.time( x<-DT[J("C"),] )
```



```
> system.time( x<-DF[DF$y=="C",] )
 사용자  시스템 elapsed 
   0.08    0.02    0.09 
> 
> DT = as.data.table(DF)
> setkey(DT,y)
> system.time( x<-DT[J("C"),] )
 사용자  시스템 elapsed 
      0       0       0 
```

Key 가 설정된 후 key 컬럼의 값을 이용하여 검색할때는 J() 함수를 이용한다

ggplot

ggplot

- The ggplot2 package, created by Hadley Wickham, offers a powerful graphics language for creating elegant and complex plots.
- Origianlly based on Leland Wilkinson's The Grammar of Graphics, ggplot2 allows you to create graphs that represent both univariate and multivariate numerical and categorical data in a straightforward manner.
- Grouping can be represented by color, symbol, size, and transparency.

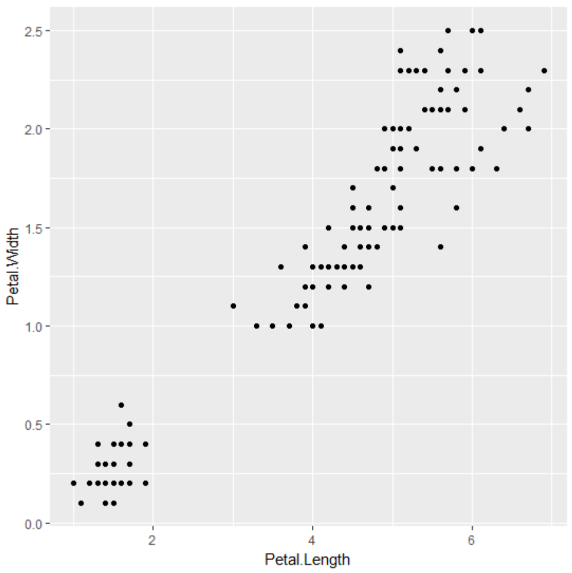
참조 : http://blog.daum.net/huh420/19

ggplot

- Simple example

```
library(ggplot2)
ggplot(data=iris, aes(x = Petal.Length,
        y = Petal.Width))+ geom_point()
```

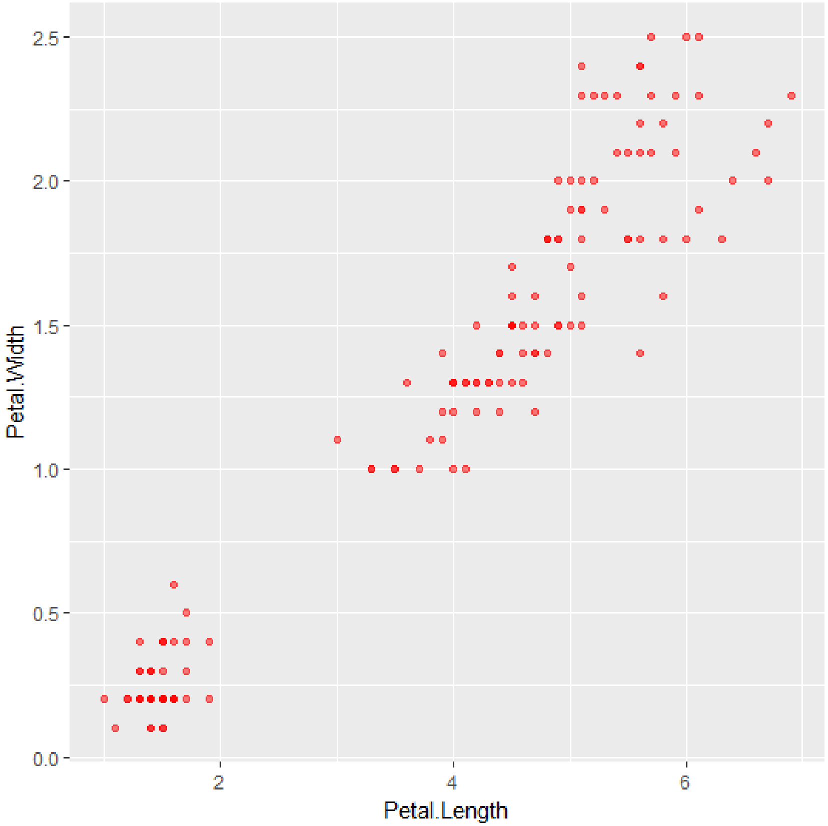
- ggplot() : 그래프를 그릴 대상 데이터 정의
- geom_xx() : 그래프의 형태 정의



ggplot

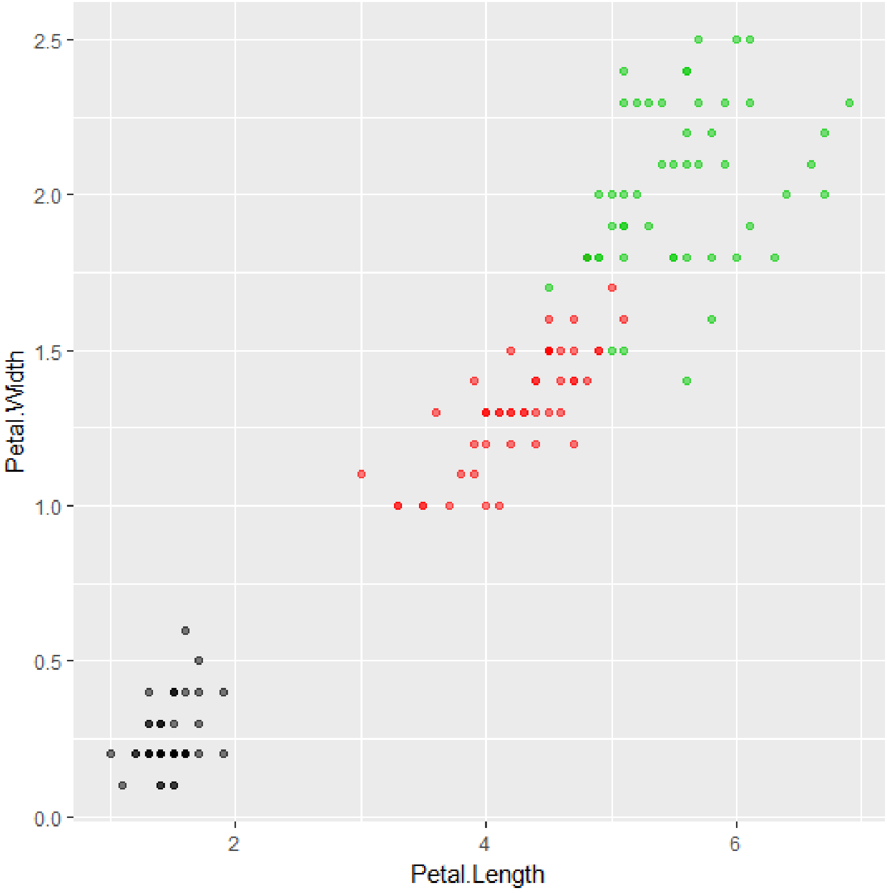
- Simple example

```
ggplot(data=iris, aes(x = Petal.Length,
        y = Petal.Width))+
geom_point(alpha=0.5, color="red")
```

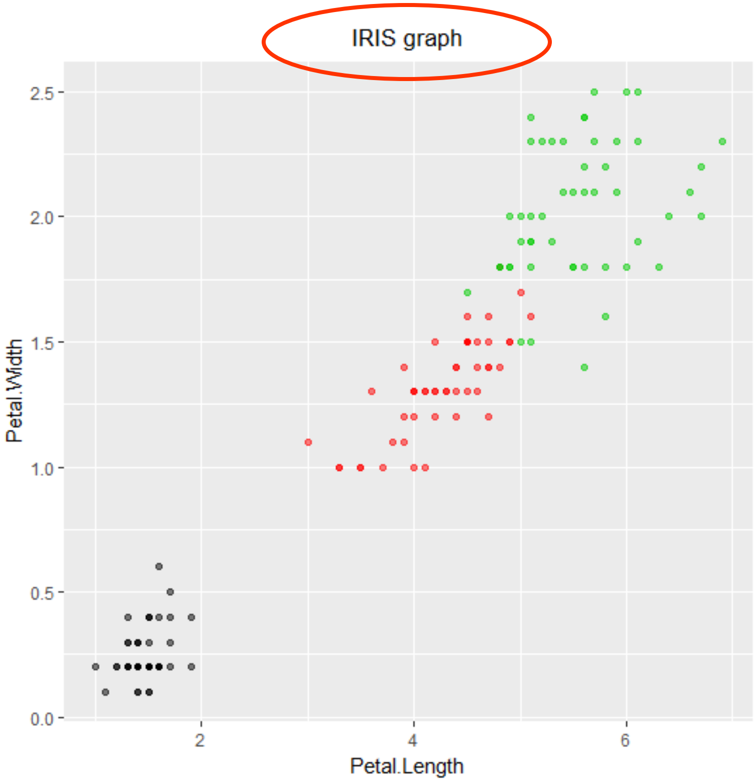


ggplot

```
mycolor = as.numeric(iris$Species)
ggplot(data=iris, aes(x = Petal.Length,
           y = Petal.Width))+
geom_point(alpha=0.5, color=mycolor)
```

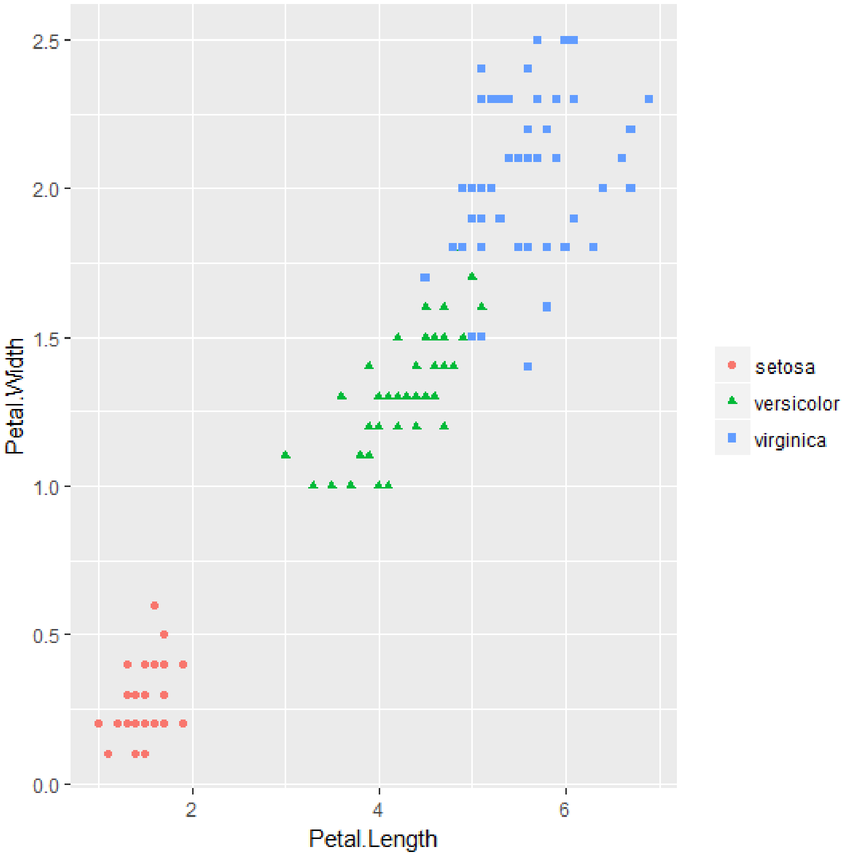


```
mycolor = as.numeric(iris$Species)
ggplot(data=iris, aes(x = Petal.Length,
        y = Petal.Width))+
geom_point(alpha=0.5, color=mycolor)+
ggtitle("IRIS graph") +
theme(plot.title = element_text(hjust = 0.5))
```



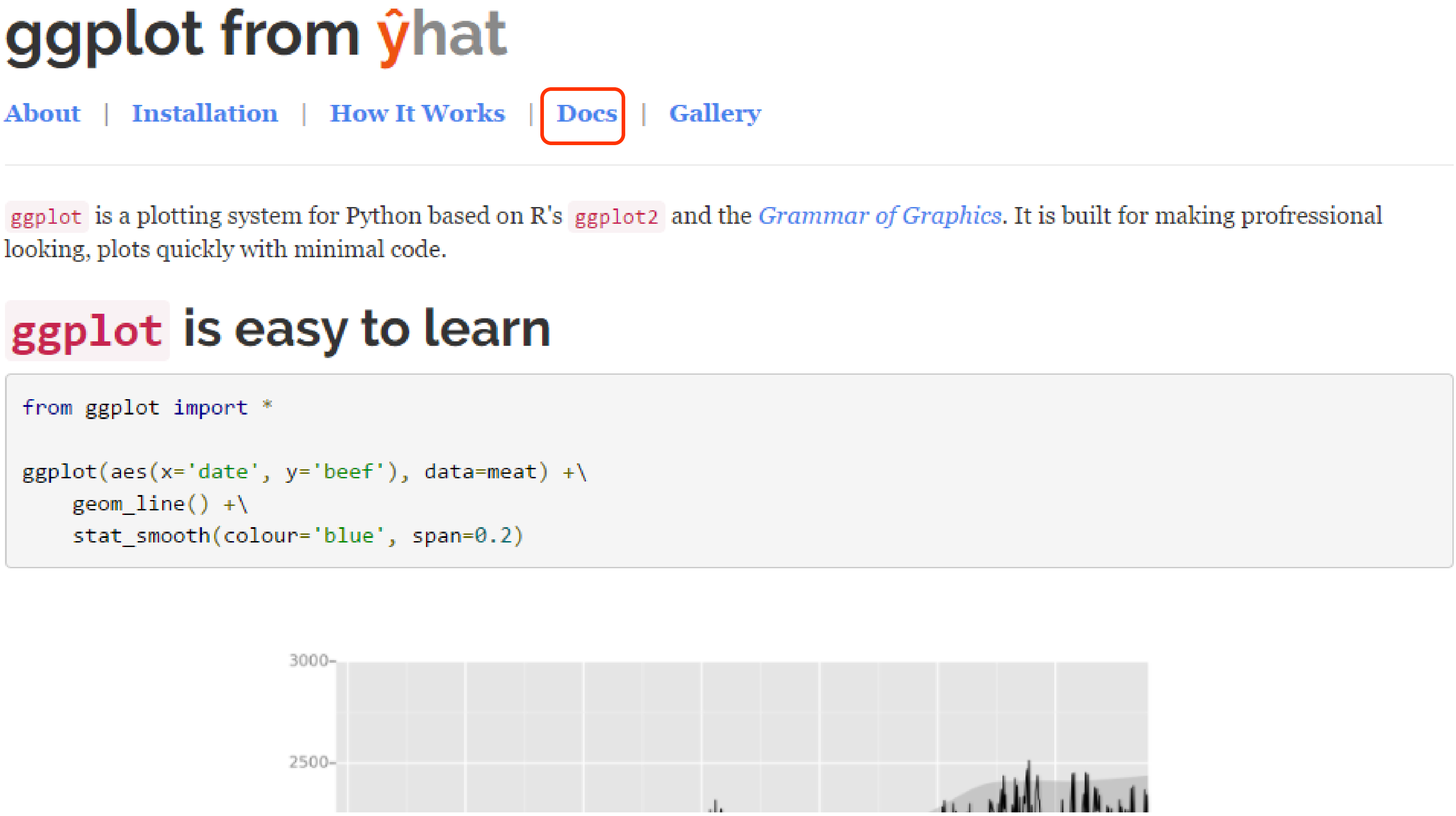
ggplot

```
ggplot(data=iris, aes(x = Petal.Length,
        y = Petal.Width, shape=factor(Species)))+
geom_point(aes(color=factor(Species)))+
theme(legend.title=element_blank())      # legend title 제거
```



ggplot

- Examples of ggplot
  - http://ggplot.yhathq.com/



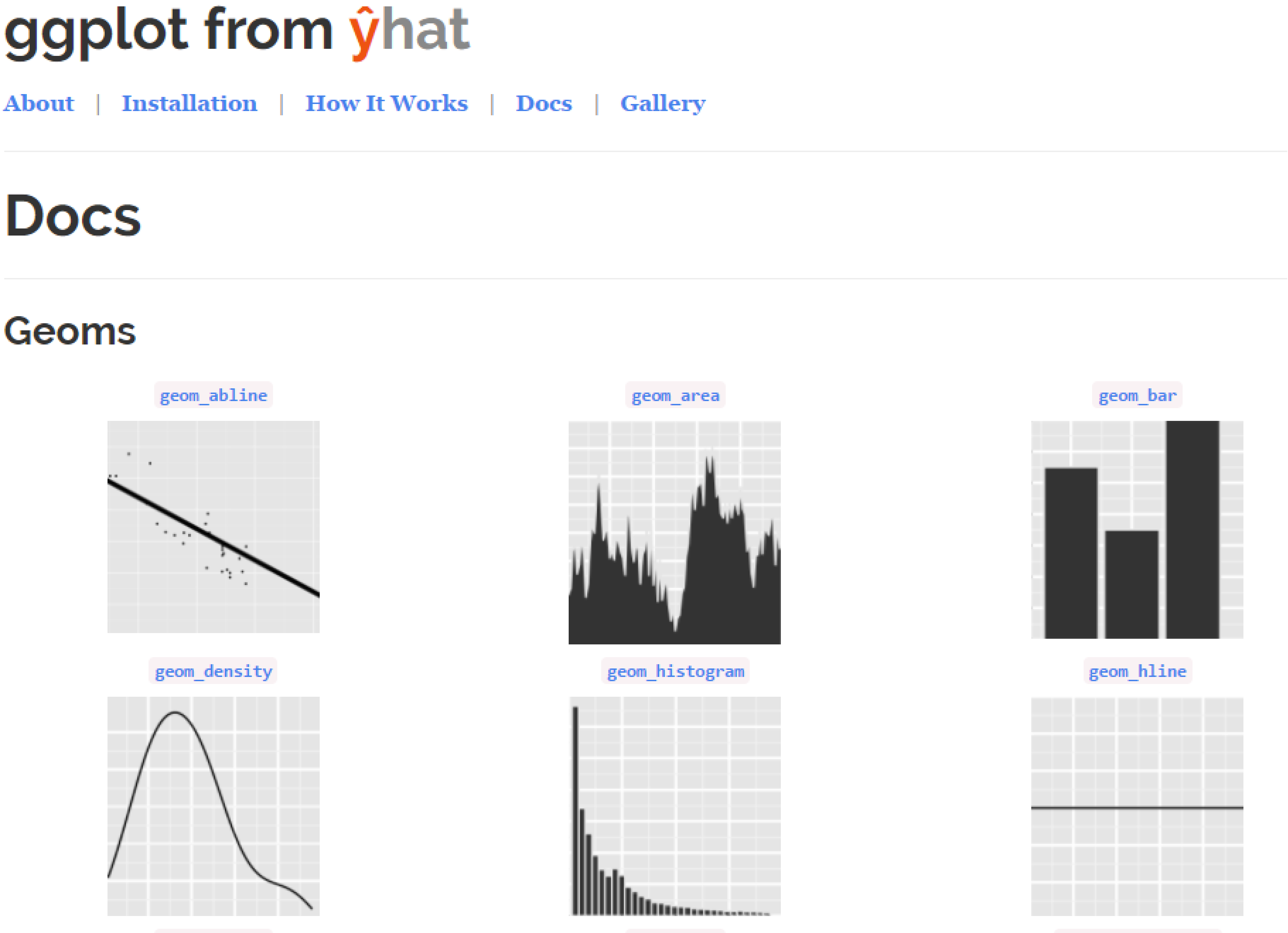
ggplot from ŷhat

About | Installation | How It Works | Docs | Gallery

ggplot is a plotting system for Python based on R's ggplot2 and the *Grammar of Graphics*. It is built for making profressional looking, plots quickly with minimal code.

ggplot is easy to learn

```
from ggplot import *

ggplot(aes(x='date', y='beef'), data=meat) +\
    geom_line() +\
    stat_smooth(colour='blue', span=0.2)
```



ggplot from ŷhat

About | Installation | How It Works | Docs | Gallery

Docs

Geoms

geom_abline

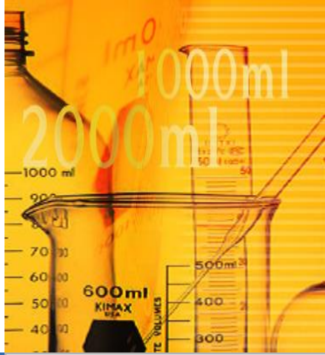
geom_area

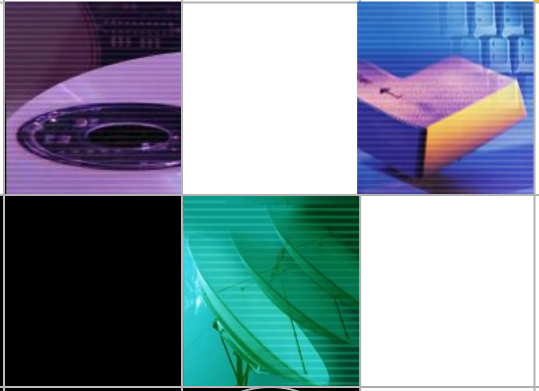
geom_bar

geom_density

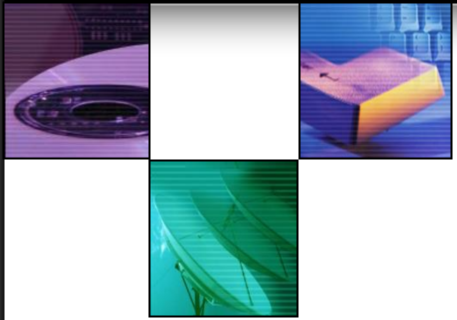
geom_histogram

geom_hline



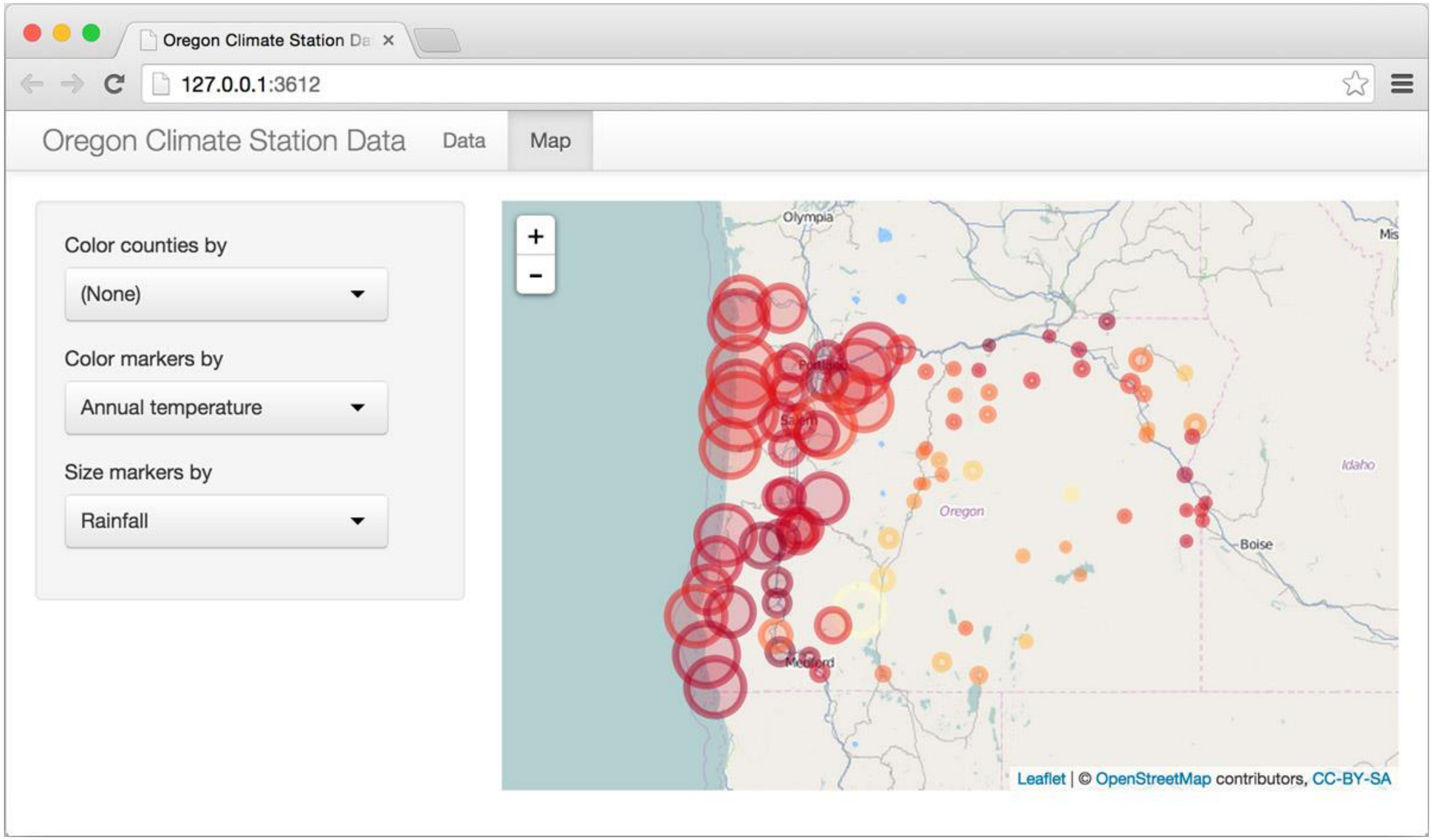


Shiny



Summary

- 그래프를 포함 R에서 작업한 결과를 Web page 에 보일수 있는 환경을 제공
- 웹프로그래밍 지식이 없어도 상당한 수준의 웹그래픽 가능


Oregon Climate Station Da ×
127.0.0.1:3612
Oregon Climate Station Data
Data
Map
+
-
Olympia
Mis
Portland
Salem
Oregon
Idaho
Boise
Medford
Leaflet | © OpenStreetMap contributors, CC-BY-SA

Color counties by
(None)

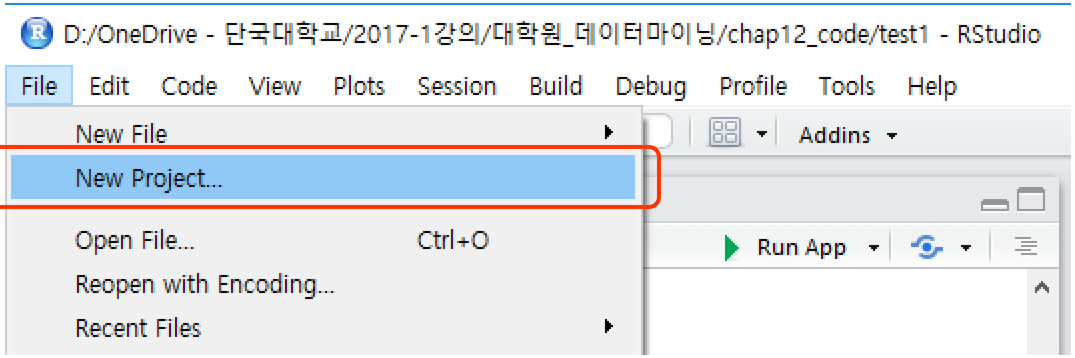
Color markers by
Annual temperature

Size markers by
Rainfall

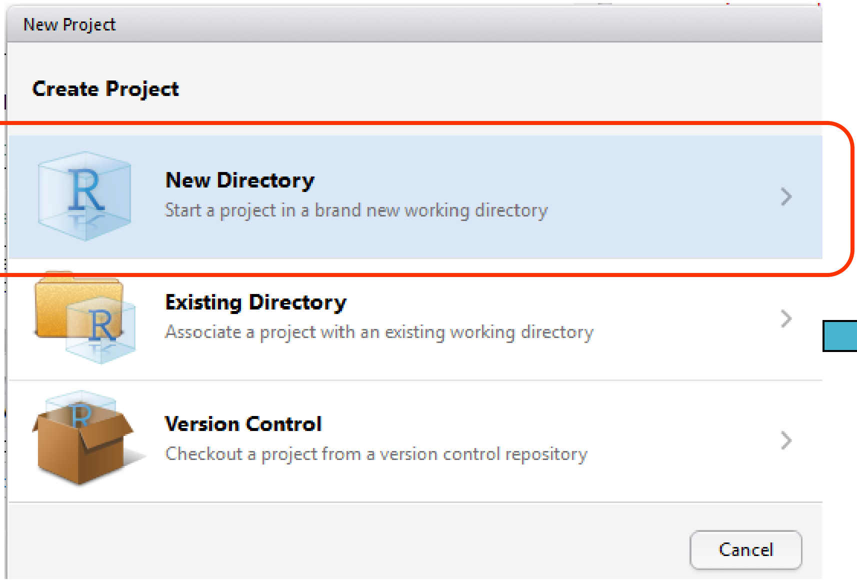
Summary

- Build useful web applications with only a few lines of code—no JavaScript required.
- Shiny applications are automatically “live” in the same way that spreadsheets are live. Outputs change instantly as users modify inputs, without requiring a reload of the browser.
- Shiny user interfaces can be built entirely using R, or can be written directly in HTML, CSS, and JavaScript for more flexibility.
- Works in any R environment (Console R, Rgui for Windows or Mac, ESS, StatET, RStudio, etc.)
- Attractive default UI theme based on Twitter Bootstrap.
- A highly customizable slider widget with built-in support for animation.
- Pre-built output widgets for displaying plots, tables, and printed output of R objects.
- Fast bidirectional communication between the web browser and R using the websocket package.
- Uses a reactive programming model that eliminates messy event handling code, so you can focus on the code that really matters.
- Develop and redistribute your own Shiny widgets that other developers can easily drop into their own applications (coming soon!).

Create shiny project



D:/OneDrive - 단국대학교/2017-1강의/대학원_데이터마이닝/chap12_code/test1 - RStudio



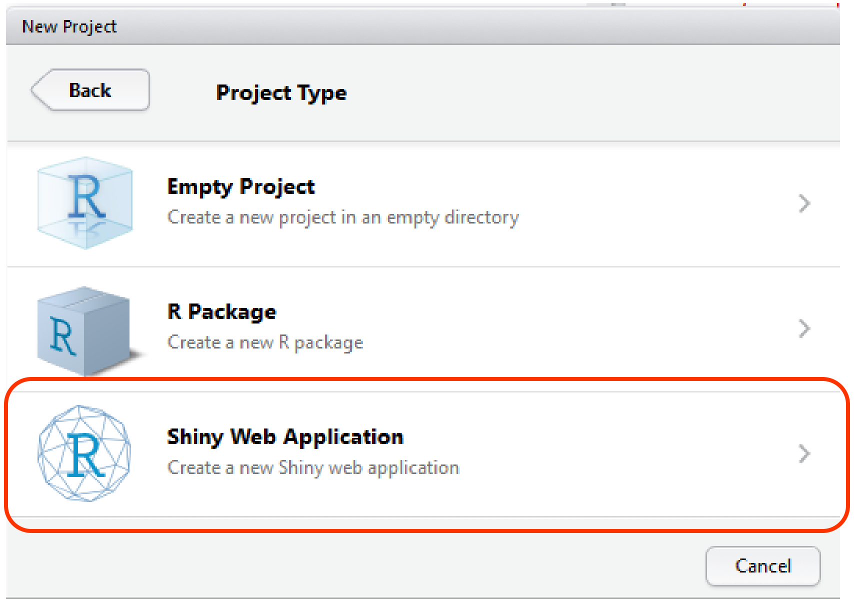
New Project

Create Project

New Directory
Start a project in a brand new working directory

Existing Directory
Associate a project with an existing working directory

Version Control
Checkout a project from a version control repository



New Project

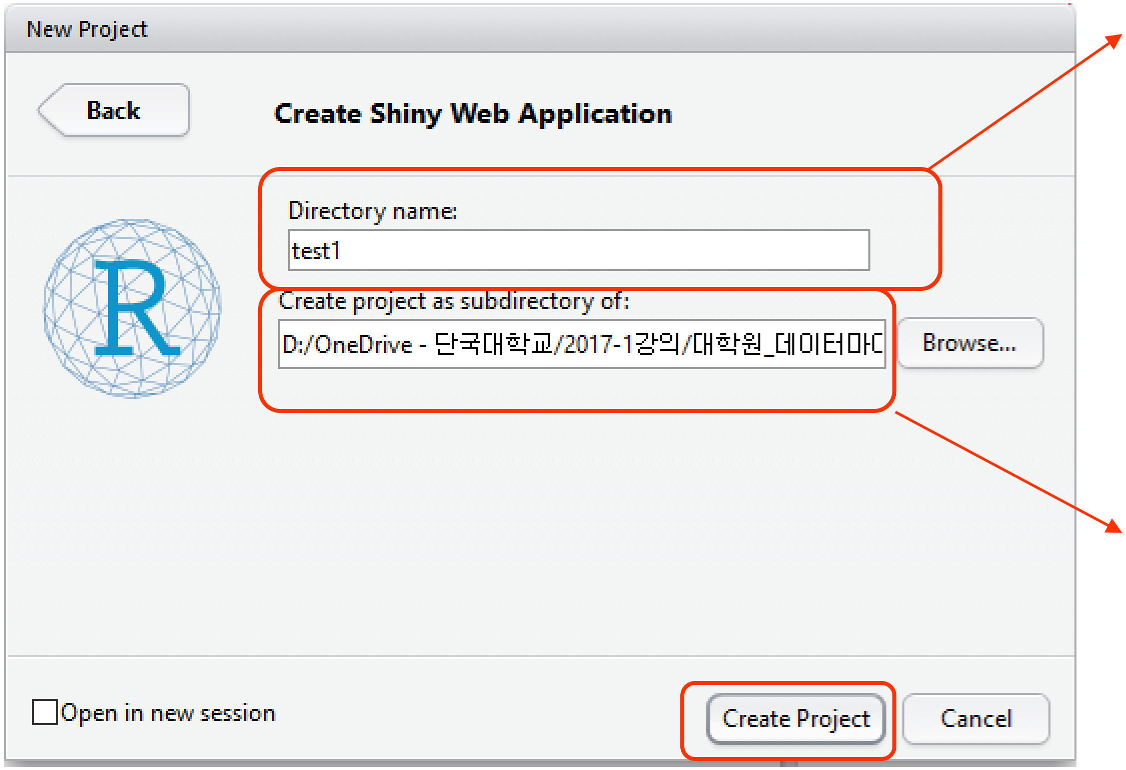
Project Type

Empty Project
Create a new project in an empty directory

R Package
Create a new R package

Shiny Web Application
Create a new Shiny web application

Create shiny project



New Project

Create Shiny Web Application

Directory name: test1

Create project as subdirectory of: D:/OneDrive - 단국대학교/2017-1강의/대학원_데이터마

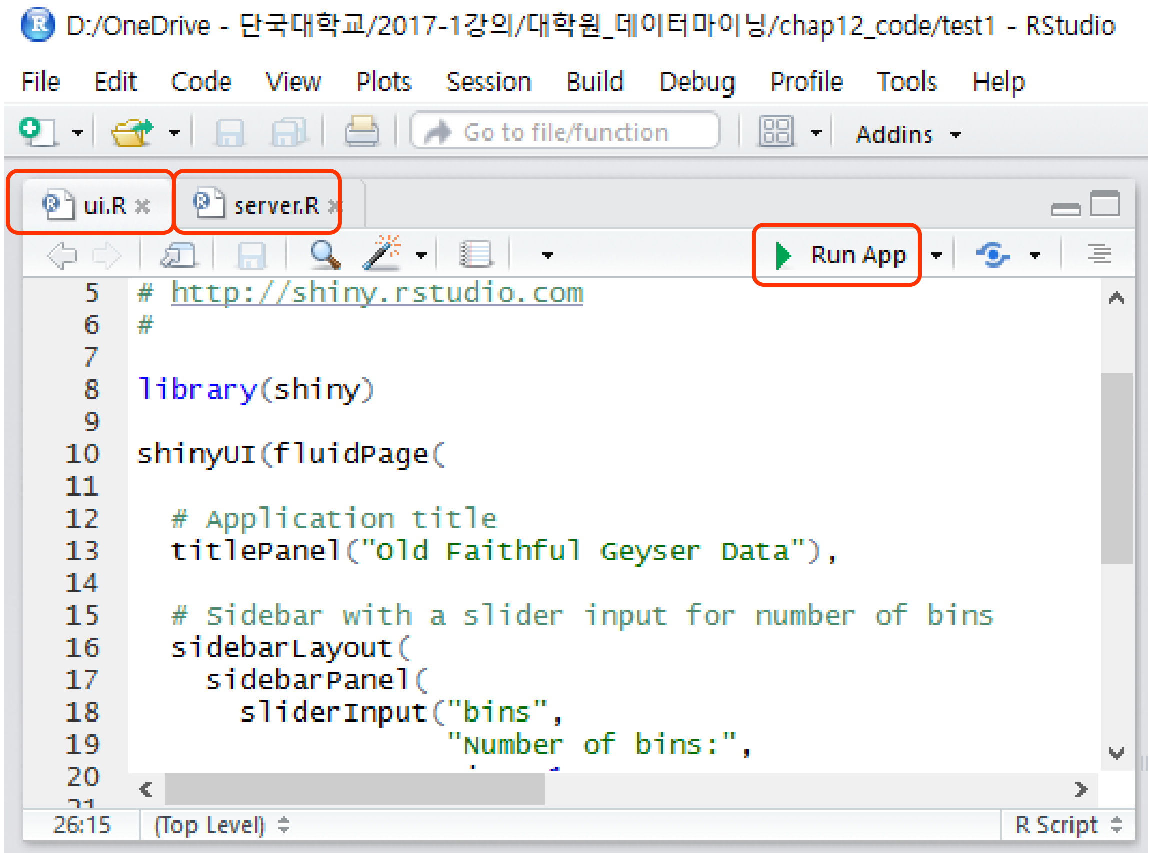
□ Open in new session

Project folder (sub folder of Work folder)

Work folder

Create shiny project

- You can see two template files
  - ui.R
  - server.R

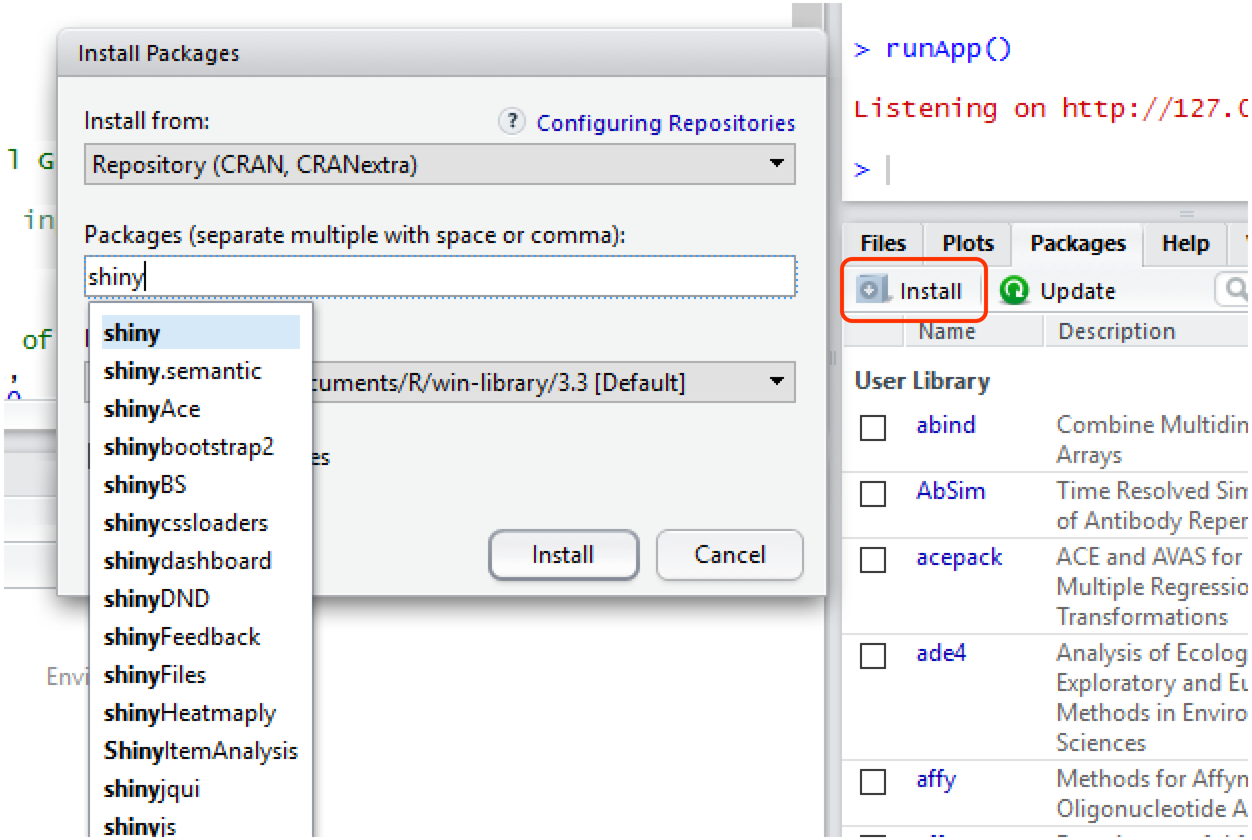


D:/OneDrive - 단국대학교/2017-1강의/대학원_데이터마이닝/chap12_code/test1 - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
ui.R
server.R
Run App
26:15
(Top Level)
R Script

```
# http://shiny.rstudio.com
#

library(shiny)

shinyUI(fluidPage(

  # Application title
  titlePanel("Old Faithful Geyser Data"),

  # Sidebar with a slider input for number of bins
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
                  "Number of bins:",
```

Install shiny package

```
install.packages("shiny")
```



```
> runApp()
Listening on http://127.0
> 
```

Install Packages

Install from: Repository (CRAN, CRANextra)

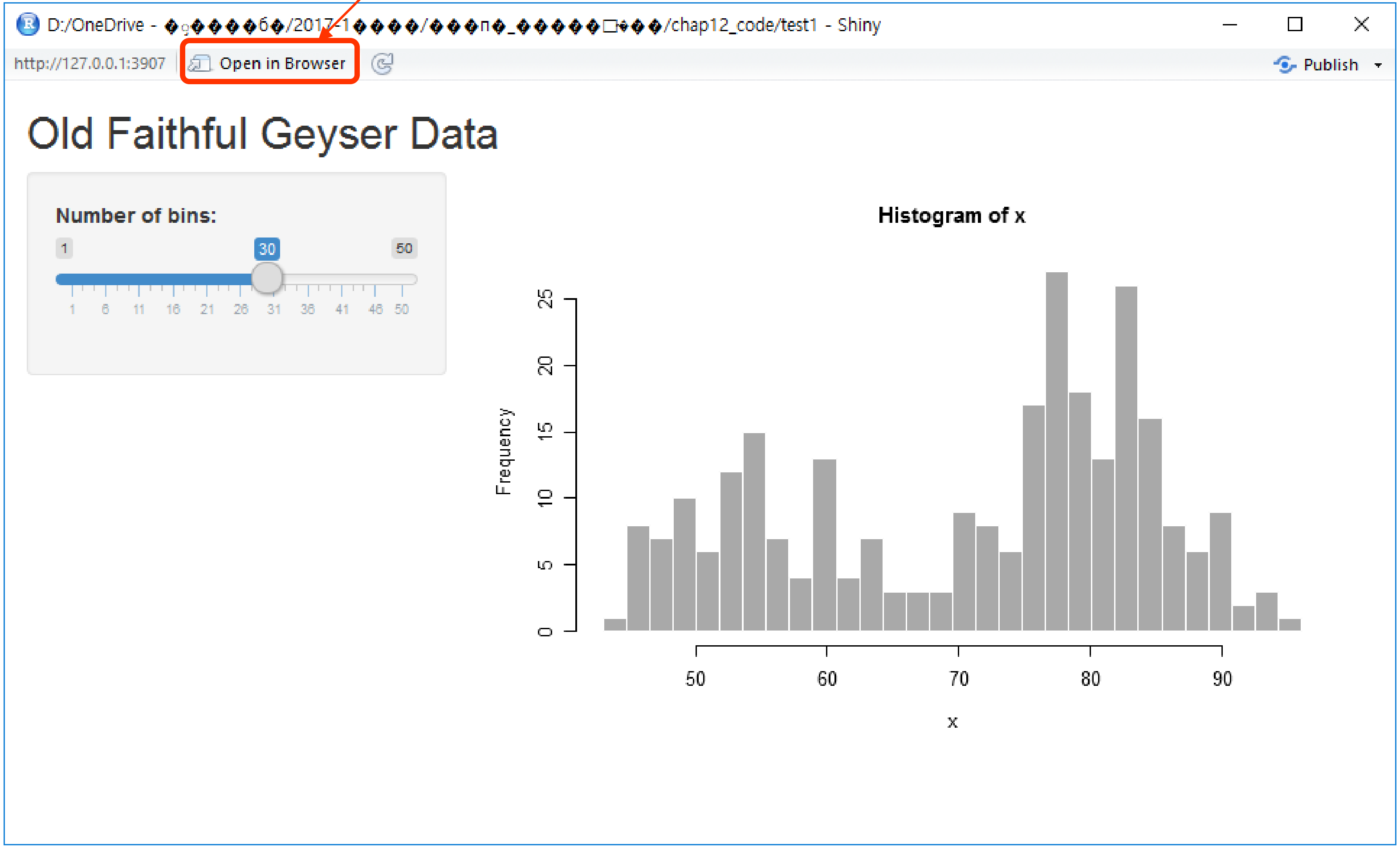
Packages (separate multiple with space or comma): shiny

	Name	Description
□	abind	Combine Multidin Arrays
□	AbSim	Time Resolved Sin of Antibody Reper
□	acepack	ACE and AVAS for Multiple Regressio Transformations
□	ade4	Analysis of Ecolog Exploratory and Eu Methods in Enviro Sciences
□	affy	Methods for Affyn Oligonucleotide A

Run shiny app

- Click “Run App” icon

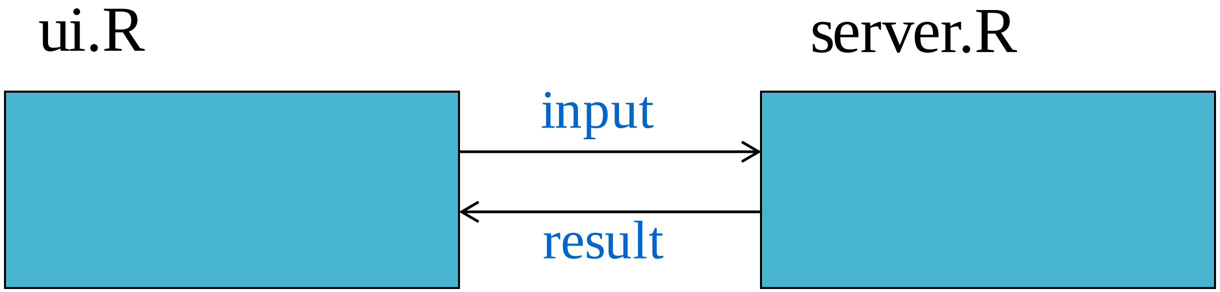
Open app on web browser



ui.R & server.R

- ui.R
  - User interface 를 정의

- server.R
  - ui.R로부터 input 값을 넘겨받아 그래프를 작성한 후 다시 ui.R 로 결과를 넘겨줌



ui.R

```
library(shiny)

# Define UI for application that plots random distributions
shinyUI(pageWithSidebar(

  # Application title
  headerPanel("Hello Shiny!"),

  # Sidebar with a slider input for number of observations
  sidebarPanel(
    sliderInput("obs",
                "Number of observations:",
                min = 1,
                max = 1000,
                value = 500)
  ),

  # Show a plot of the generated distribution
  mainPanel(
    plotOutput("distPlot")
  )
))
```

server.R

```
library(shiny)

# Define server logic required to generate and plot a random distribution
shinyServer(function(input, output) {

  # Expression that generates a plot of the distribution. The expression
  # is wrapped in a call to renderPlot to indicate that:
  #
  #  1) It is "reactive" and therefore should be automatically
  #     re-executed when inputs change
  #  2) Its output type is a plot
  #
  output$distPlot <- renderPlot({

    # generate an rnorm distribution and plot it
    dist <- rnorm(input$obs)
    hist(dist)
  })
})
```

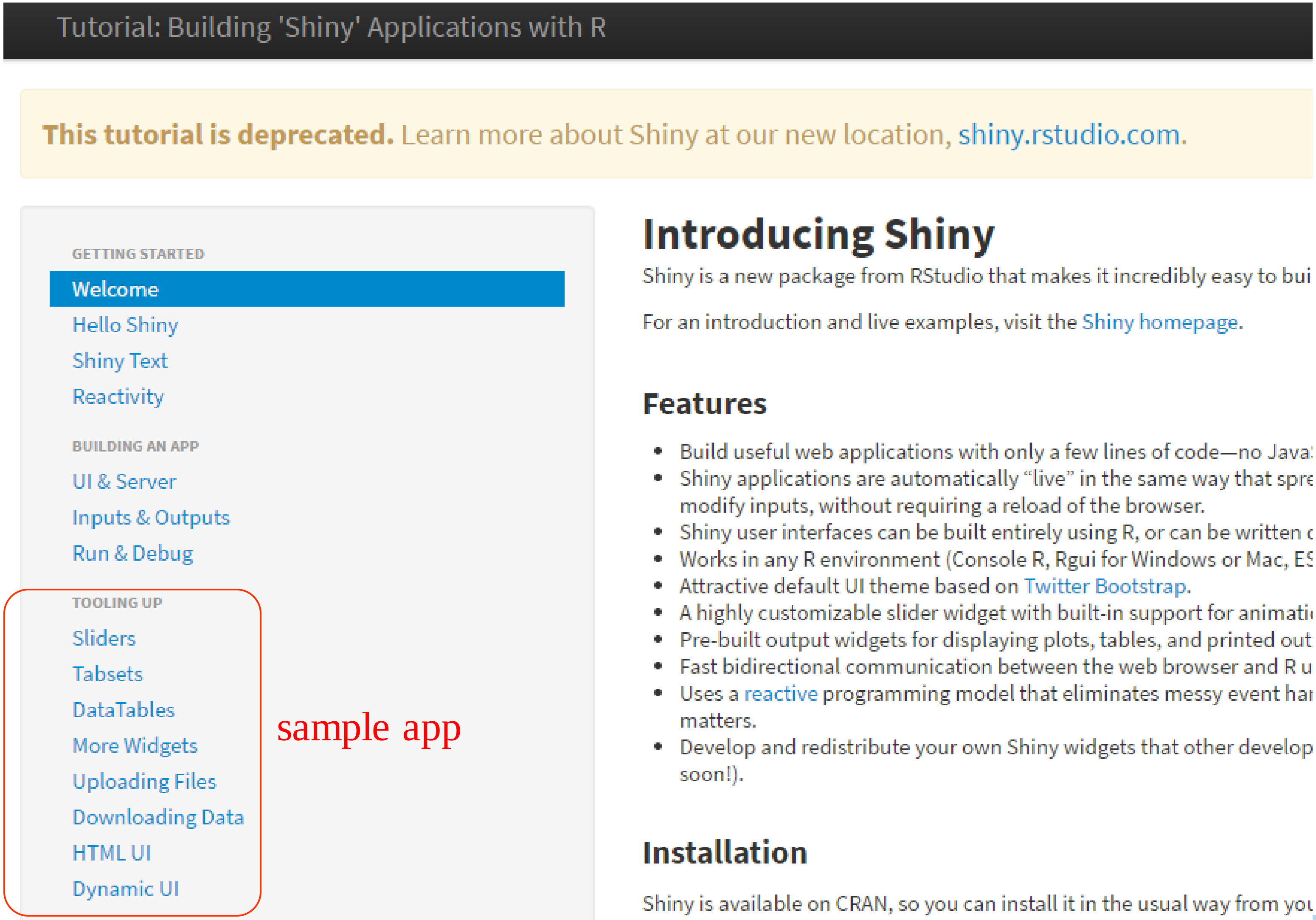
Install shiny server

- Small webserver
- Can be installed on Linux server (free)
- We can use Rstudio webserver after registration

- See registration guide
  - http://blog.naver.com/PostView.nhn?blogId=woohuyck111&logNo=221009223764

Shiny tutorial

- http://rstudio.github.io/shiny/tutorial/



This tutorial is deprecated. Learn more about Shiny at our new location, shiny.rstudio.com.

Introducing Shiny

Shiny is a new package from RStudio that makes it incredibly easy to bui

For an introduction and live examples, visit the Shiny homepage.

Features

- Build useful web applications with only a few lines of code—no Java
- Shiny applications are automatically "live" in the same way that spre modify inputs, without requiring a reload of the browser.
- Shiny user interfaces can be built entirely using R, or can be written c
- Works in any R environment (Console R, Rgui for Windows or Mac, ES
- Attractive default UI theme based on Twitter Bootstrap.
- A highly customizable slider widget with built-in support for animati
- Pre-built output widgets for displaying plots, tables, and printed out
- Fast bidirectional communication between the web browser and R u
- Uses a reactive programming model that eliminates messy event har matters.
- Develop and redistribute your own Shiny widgets that other develop soon!).

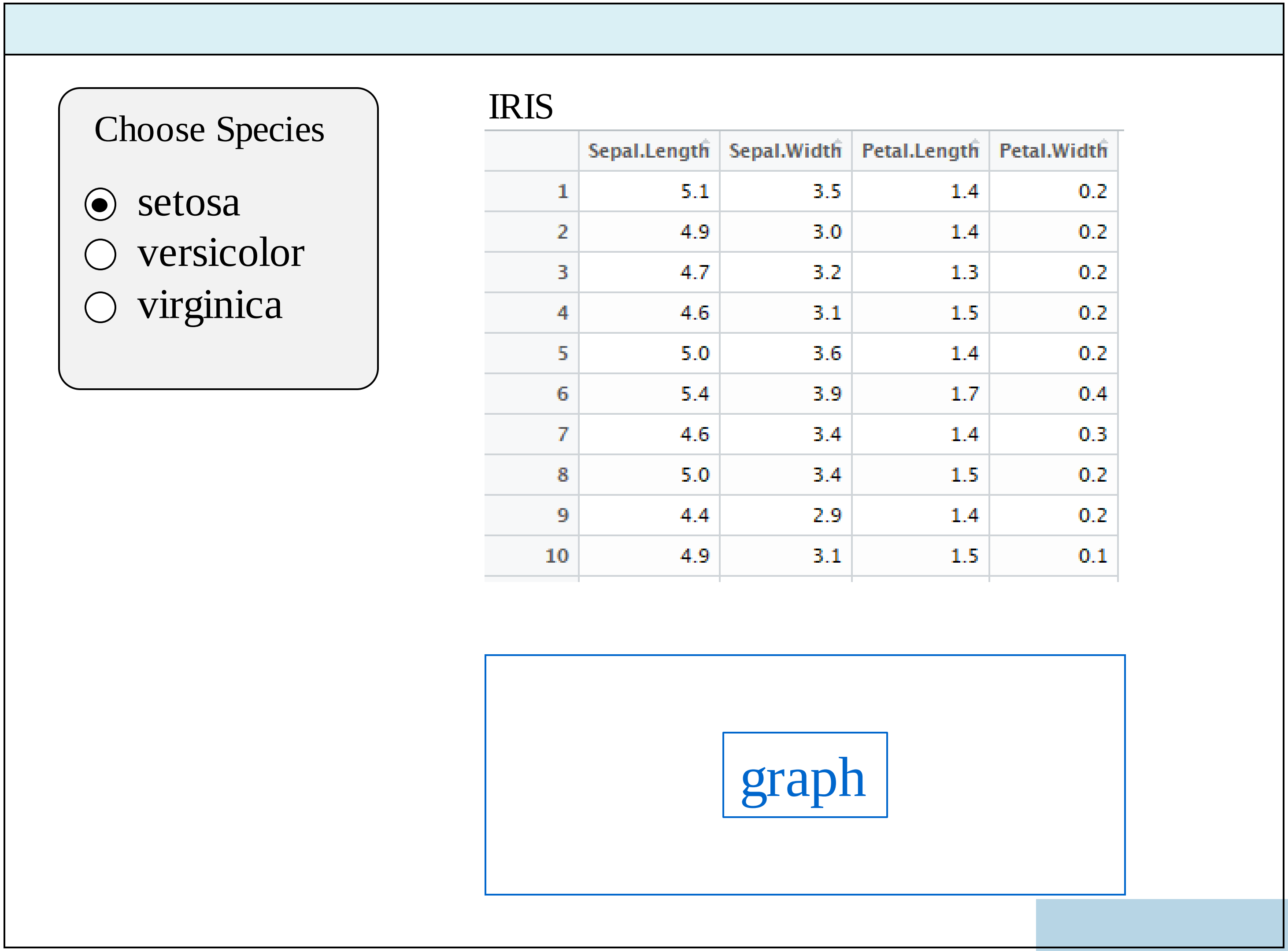
Installation

Shiny is available on CRAN, so you can install it in the usual way from you

sample app

[과제 1]

- Using shiny, implement following App



Choose Species

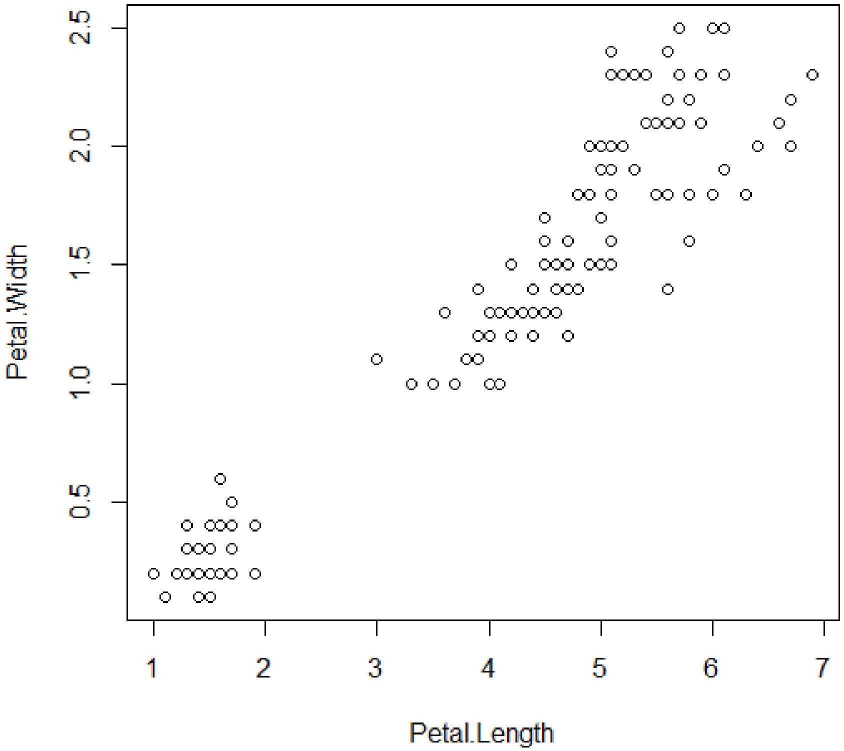
● setosa
○ versicolor
○ virginica

IRIS

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
5	5.0	3.6	1.4	0.2
6	5.4	3.9	1.7	0.4
7	4.6	3.4	1.4	0.3
8	5.0	3.4	1.5	0.2
9	4.4	2.9	1.4	0.2
10	4.9	3.1	1.5	0.1

[과제 1]

- graph


Petal.Width
Petal.Length