R과 Shiny를 이용한 Web Application의 제작

문건웅

2018/10/26

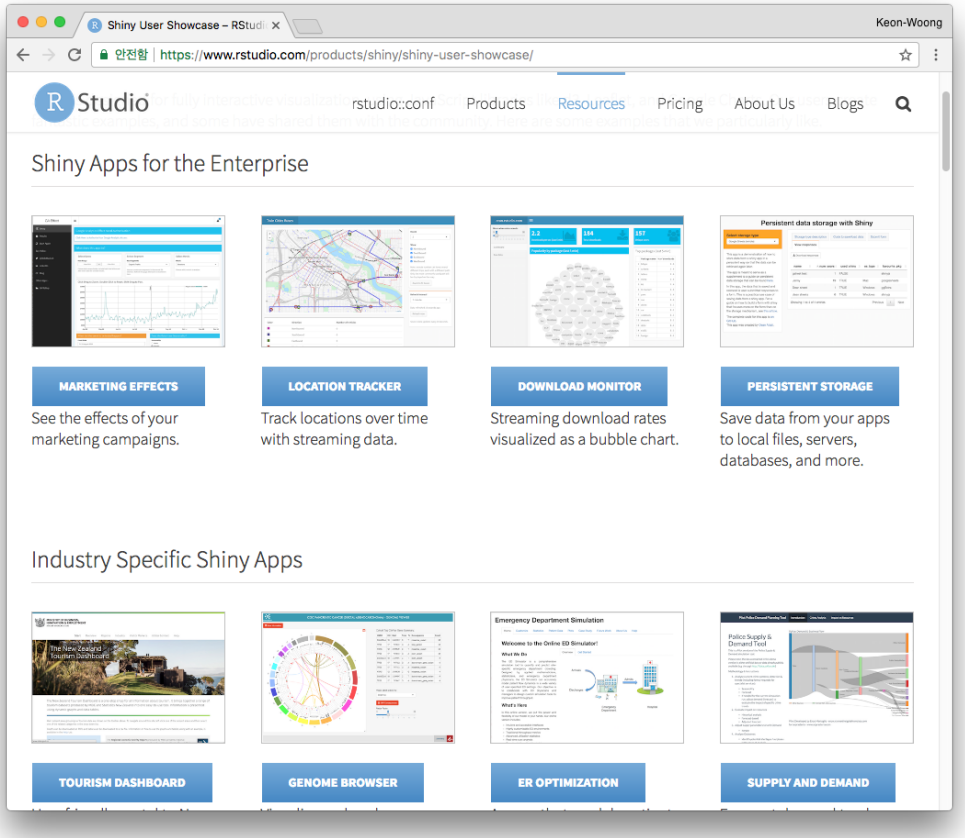
강사소개

문건웅

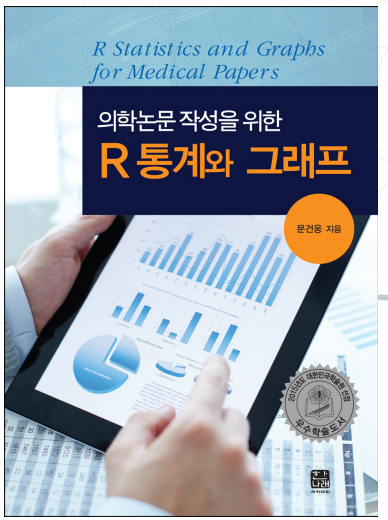
- 가톨릭대학교 의과대학 교수
- 성빈센트병원 순환기내과 재직
- R packages (CRAN)
    - mycor, moonBook, ztable(2015)
    - ggiraphExtra(2016)
    - dplyrAssist, editData, ggplotAssist(2017)
    - webr, rrtable(2018)
- Books
    - 의학논문 작성을 위한 R통계와 그래프(2015, 한나래)
        - 2015년 대한민국 학술원 우수학술도서
    - 웹에서 클릭만으로 하는 R 통계분석(2015, 한나래)
    - Learn ggplot2 Using Shiny App(2017, Springer)
- Web-R.org 운영

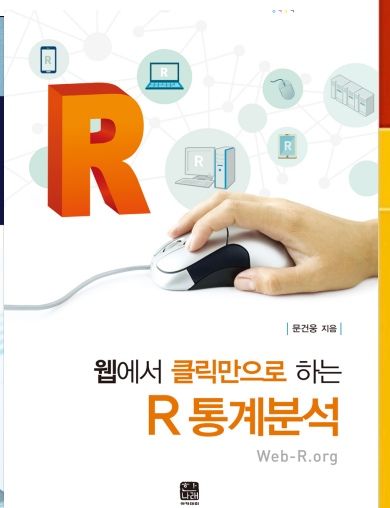
Shiny 로 어떤 앱을 만들 수 있나?

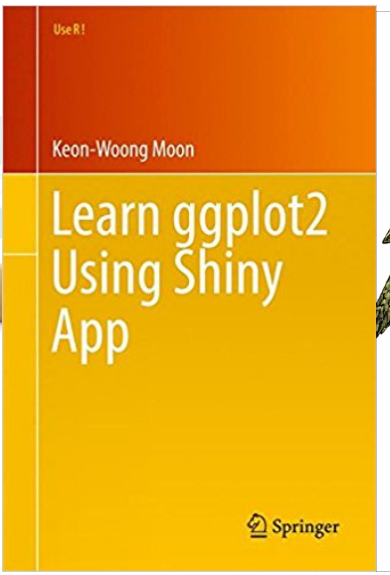
https://www.rstudio.com/products/shiny/shiny-user-showcase/

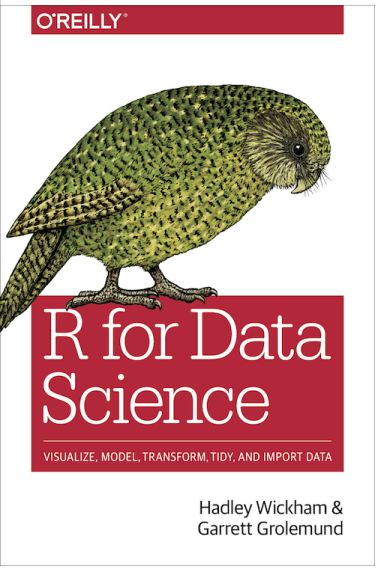


R을 배우자









Shiny를 배울 준비가 되어 있는가?

https://shiny.rstudio.com/tutorial/quiz/

필요사항

- R 설치 (https://cran.r-project.org/)
- RStudio 설치(https://www.rstudio.com/products/rstudio/)
- 필요한 R 패키지 : R console에서 다음 명령어 실행

```
install.packages(c("knitr","shiny","rmarkdown"))
install.packages(c("tidyverse","DT","moonBook"))
```

- 6번째 앱에서 knitr Reports 중 pdf 다운로드를 위하여는 LaTex 설치가 필요하다. (http://ktug.or.kr)

예제 파일 및 앱 소스파일

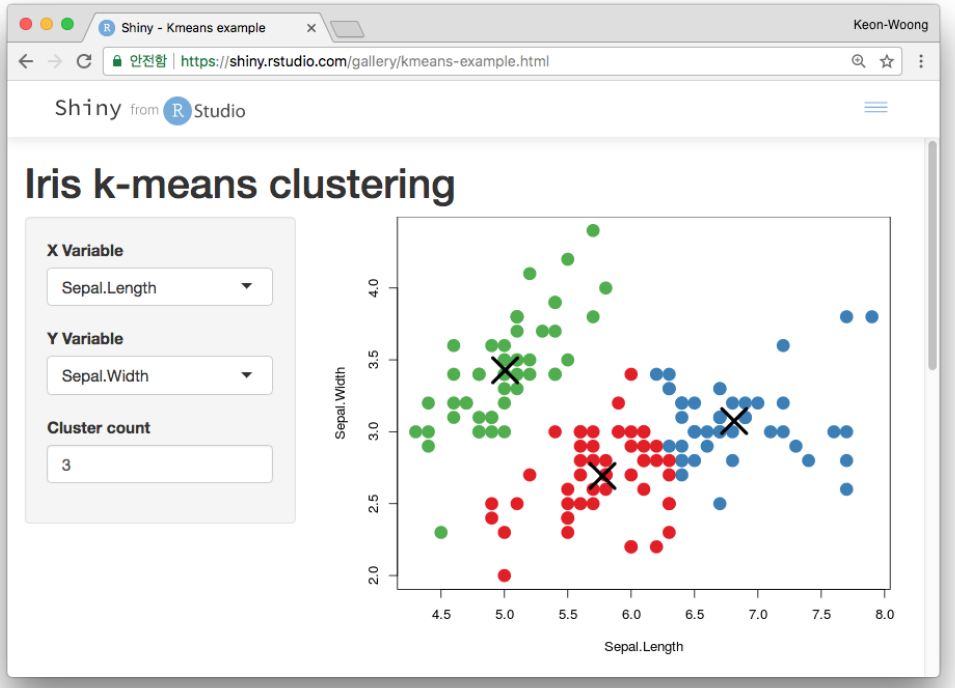
이번 강의에 사용되는 앱 및 소스 파일들은 다음 github에서 다운로드 받을수 있다.

https://github.com/cardiomoon/shinyLecture2

Introduction of Shiny

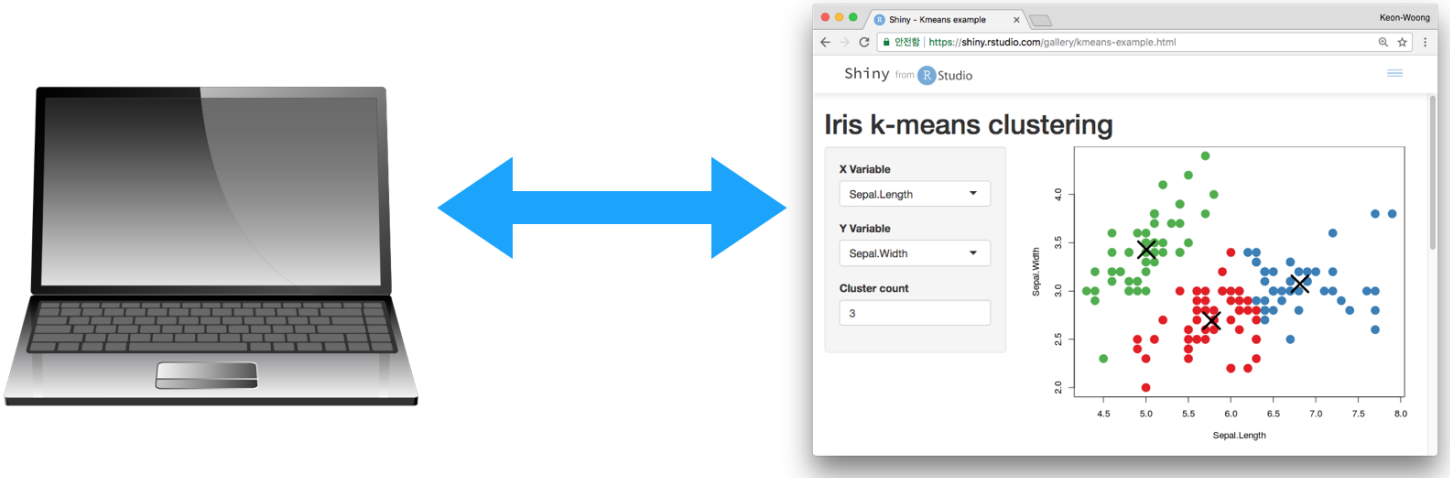
1. The First Shiny App
2. The 2nd App : Reactivity
3. The 3rd App : Reactivity(2)
4. Stop reactions with isolate()
5. One input, two outputs
6. Download knitr reports
7. Basic DataTable
8. Advanced App - Multiple Reactive Outputs

1. The First Shiny App




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

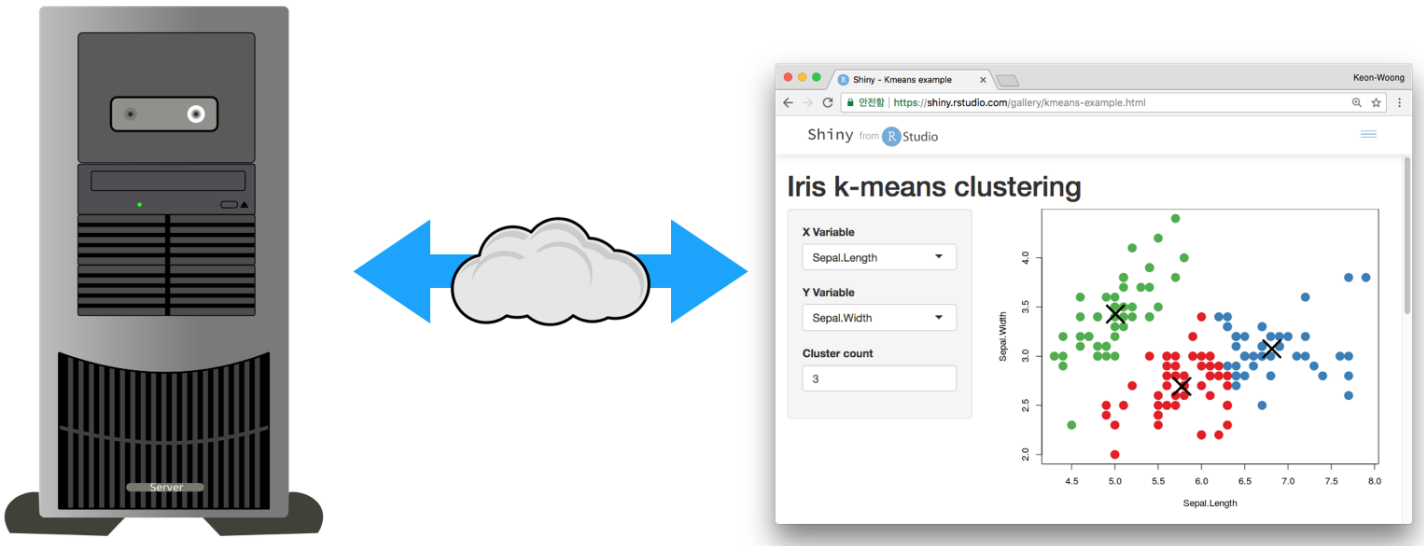
Shiny App은 R을 운영하는 컴퓨터에 의해 유지된다.



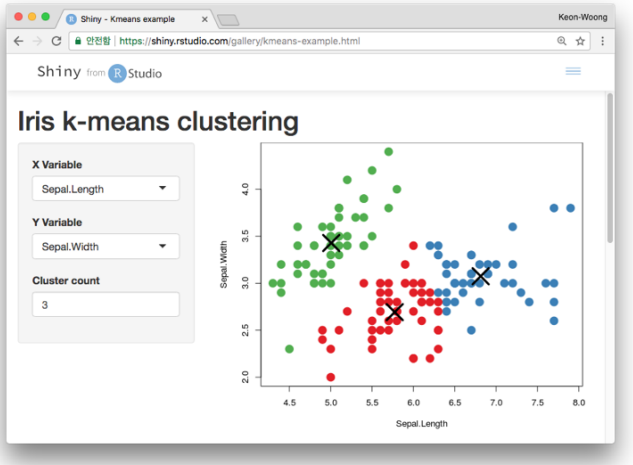
R console에서 다음 R 명령어를 실행시켜 보자

```
shiny::runGitHub('shinyLecture2', 'cardiomoon',subdir='inst/app0')
shiny::runApp("~/Documents/ownCloud/Documents/shinyLecture2/inst/app0
```

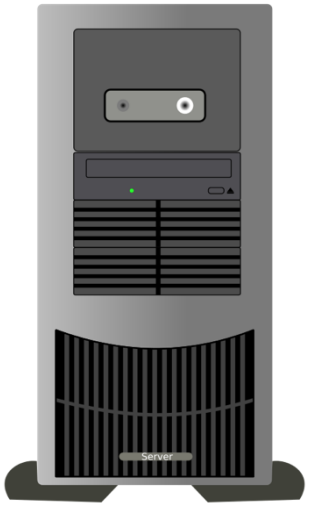
어떤 Shiny App은 R을 운영하는 서버에 의해 유지된다.

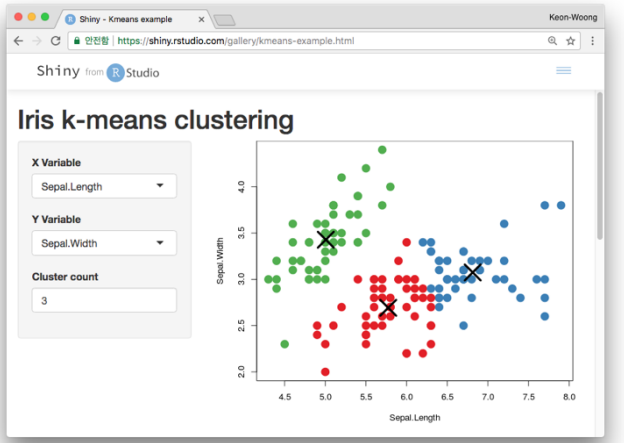


https://shiny.rstudio.com/gallery/kmeans-example.html



User Interface





Server Instructions

User Interface

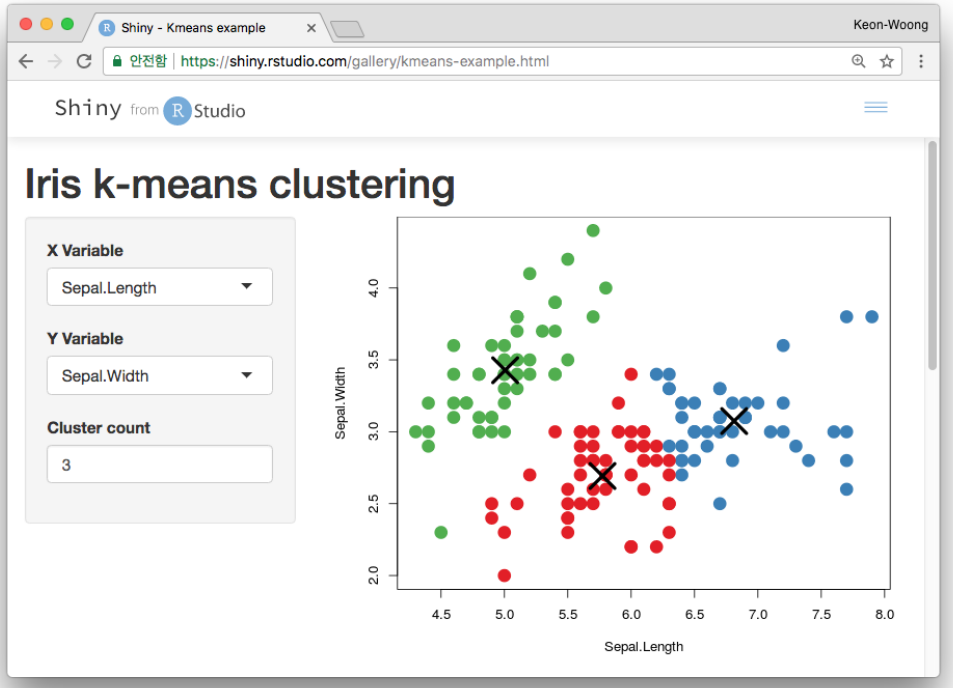
app.R

https://github.com/cardiomoon/shinyLecture2/tree/master/inst/app0

```
library(shiny)

ui<-pageWithSidebar(
  headerPanel('Iris k-means clustering'),
  sidebarPanel(
    selectInput('xcol', 'X Variable', names(iris)),
    selectInput('ycol', 'Y Variable', names(iris),
                selected=names(iris)[[2]]),
    numericInput('clusters', 'Cluster count', 3,
                 min = 1, max = 9)
  ),
  mainPanel(
    plotOutput('plot1')
  )
)
server<-function(input, output, session) {
```

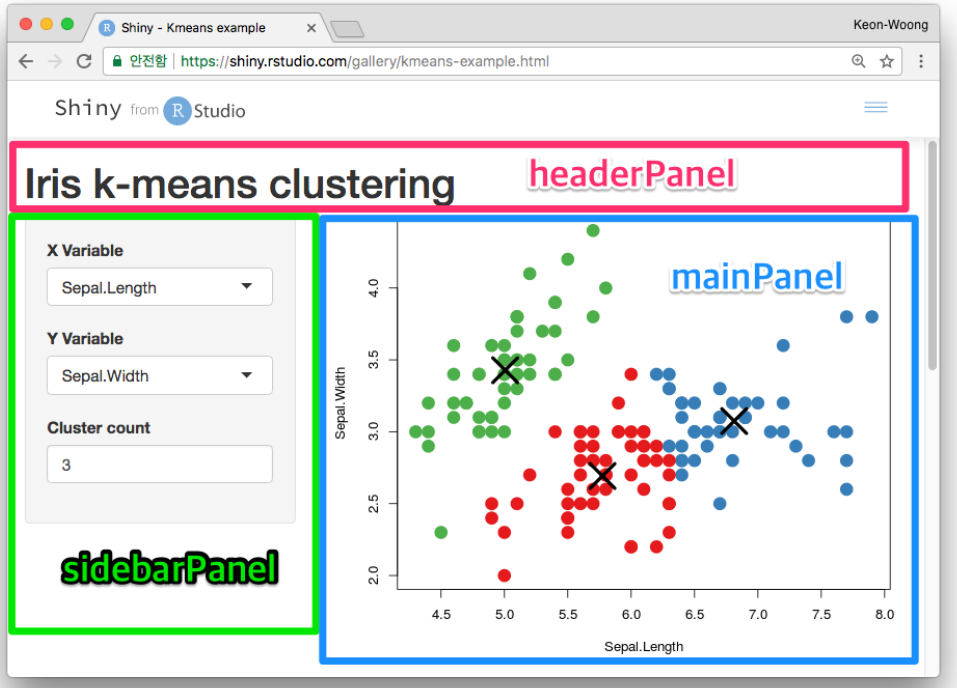
Shiny App


Shiny - Kmeans example
Keon-Woong
안전함 | https://shiny.rstudio.com/gallery/kmeans-example.html
Shiny from RStudio
Iris k-means clustering
X Variable
Sepal.Length
Y Variable
Sepal.Width
Cluster count
3
Sepal.Width
Sepal.Length
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

User interface

```
ui<-pageWithSidebar(
   headerPanel('Iris k-means clustering'),
   sidebarPanel(
    selectInput('xcol', 'X Variable', names(iris)),
    selectInput('ycol', 'Y Variable', names(iris),
                selected=names(iris)[[2]]),
    numericInput('clusters', 'Cluster count', 3,
                 min = 1, max = 9)
  ),
   mainPanel(
    plotOutput('plot1')
  )
)
```

Panels

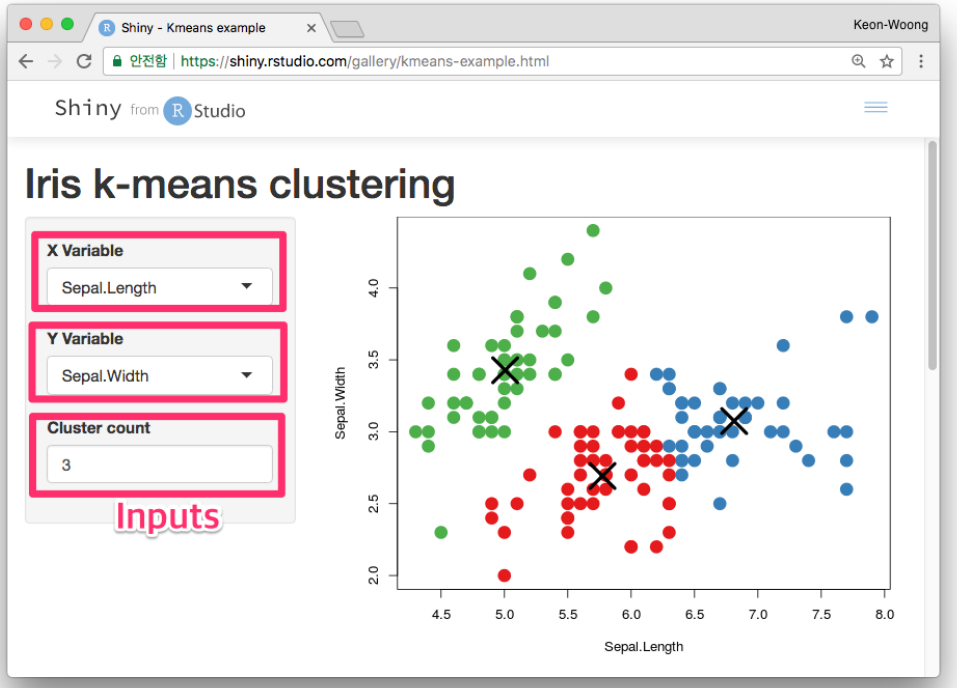

Shiny - Kmeans example
Keon-Woong
안전함 | https://shiny.rstudio.com/gallery/kmeans-example.html
Shiny from Studio
Iris k-means clustering
headerPanel
X Variable
Sepal.Length
Y Variable
Sepal.Width
Cluster count
3
sidebarPanel
mainPanel
Sepal.Width
Sepal.Length
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

Inputs

```
library(shiny)

ui<-pageWithSidebar(
  headerPanel('Iris k-means clustering'),
  sidebarPanel(
     selectInput('xcol', 'X Variable', names(iris)),
     selectInput('ycol', 'Y Variable', names(iris),
                 selected=names(iris)[[2]]),
     numericInput('clusters', 'Cluster count', 3,
                  min = 1, max = 9)
  ),
  mainPanel(
    plotOutput('plot1')
  )
)
```

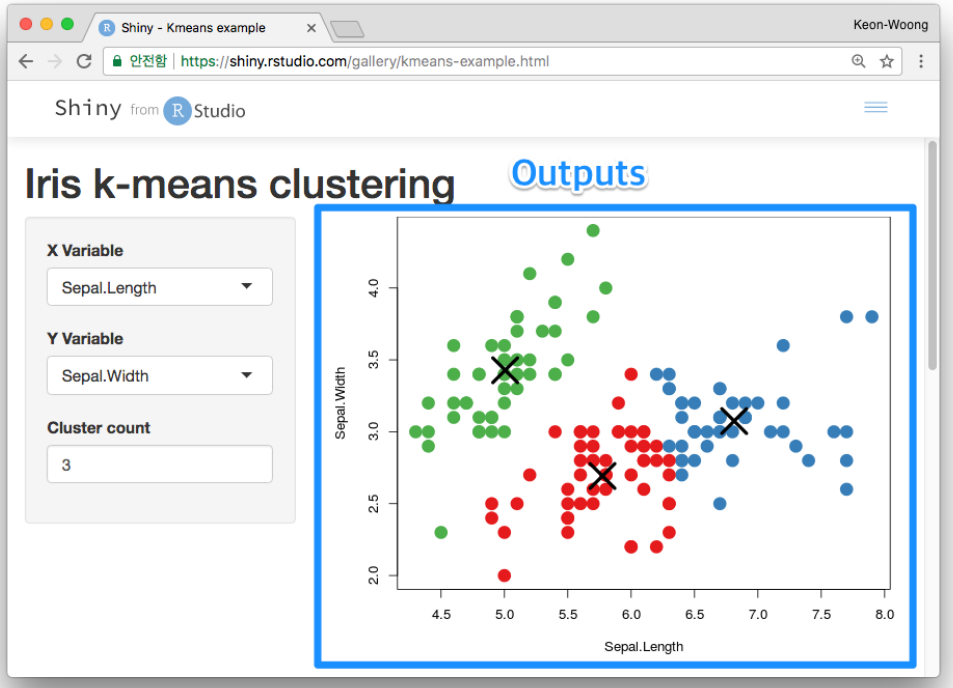
Inputs


Shiny - Kmeans example
Keon-Woong
안전함 | https://shiny.rstudio.com/gallery/kmeans-example.html
Shiny from RStudio
Iris k-means clustering
X Variable
Sepal.Length
Y Variable
Sepal.Width
Cluster count
3
Inputs
Sepal.Width
Sepal.Length
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

Outputs

```
ui<-pageWithSidebar(
  headerPanel('Iris k-means clustering'),
  sidebarPanel(
    selectInput('xcol', 'X Variable', names(iris)),
    selectInput('ycol', 'Y Variable', names(iris),
                selected=names(iris)[[2]]),
    numericInput('clusters', 'Cluster count', 3,
                 min = 1, max = 9)
  ),
  mainPanel(
     plotOutput('plot1')
  )
)
```

Outputs


Shiny - Kmeans example
Keon-Woong
안전함 | https://shiny.rstudio.com/gallery/kmeans-example.html
Shiny from RStudio
Iris k-means clustering
Outputs
X Variable
Sepal.Length
Y Variable
Sepal.Width
Cluster count
3
Sepal.Width
Sepal.Length
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

Shiny App Template 사용

Minimal Valid Shiny App

https://github.com/cardiomoon/shinyLecture2/blob/master/app.R

```
library(shiny)

ui <- fluidPage()

server <- function(input,output){}

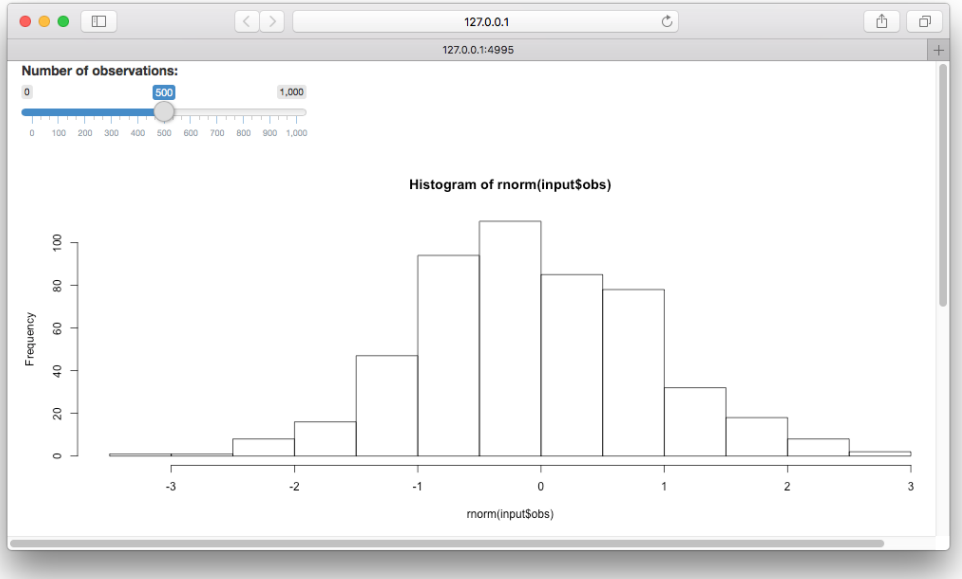
shinyApp(ui=ui,server=server)
```

Input과 Output으로 shiny app 만들기

- fluidPage() 함수의 인수로 Input()과 Output()추가

```
ui <- fluidPage(
   # *Input() functions,
   # *Output() functions
)
```

The 2nd App: Reactivity




```
shiny::runGitHub('shinyLecture2', 'cardiomoon',subdir='inst/app1')
```

Input()

*Input() 함수를 사용하여 input 만들기

```
sliderInput("obs", "Number of observations:",
                min = 0, max = 1000, value = 500)
```

*Input() 함수를 사용하여 input 만들기

```
  sliderInput("obs", "Number of observations:",
                 min = 0, max = 1000, value = 500)
```

```
<div class="form-group shiny-input-container">
  <label class="control-label" for="obs">Number of observations:</label>
  <input class="js-range-slider" id="obs" data-min="0" data-max="1000"
  data-from="500" data-step="1" data-grid="true" data-grid-num="10"
  data-grid-snap="false" data-prettify-separator="," data-prettify-enabled="t
  data-keyboard="true" data-keyboard-step="0.1" data-data-type="number"/>
</div>
```

*Input() 함수를 사용하여 input 만들기

```
library(shiny)

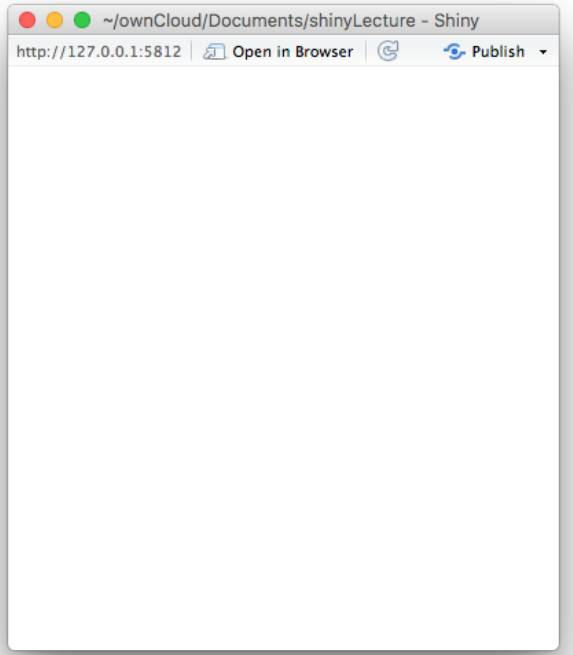
ui <- fluidPage(



)

server <- function(input,output)

shinyApp(ui=ui,server=server)
```



*Input() 함수를 사용하여 input 만들기

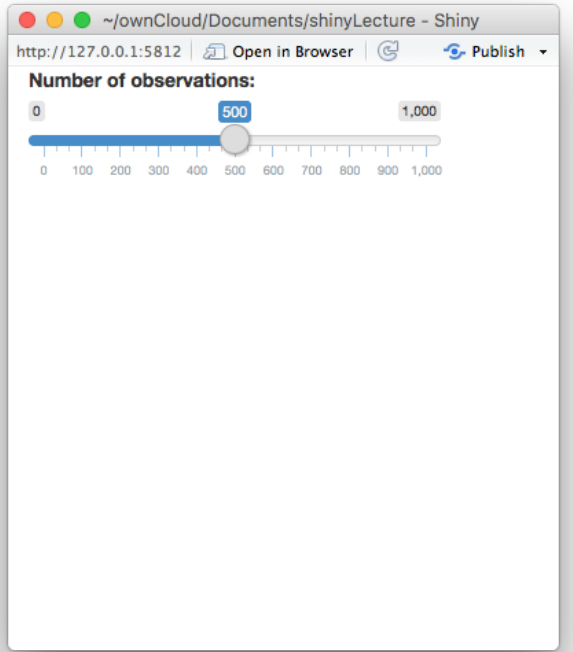
```
library(shiny)

ui <- fluidPage(
  sliderInput(inputId = "obs",
    label = "Number of observati
    min = 0, max = 1000, value =

)

server <- function(input,output)

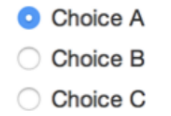
shinyApp(ui=ui,server=server)
```



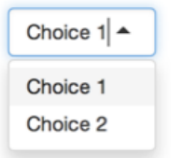
*Input functions

numericInput(inputId, label, value, min, max, step)

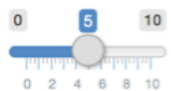
passwordInput(inputId, label, value)



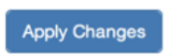
radioButtons(inputId, label, choices, selected, inline)



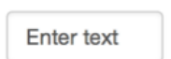
selectInput(inputId, label, choices, selected, multiple, selectize, width, size) (also **selectizeInput()**)



sliderInput(inputId, label, min, max, **value,** step, round, format, locale, ticks, animate, width, sep, pre, post)



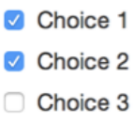
submitButton(text, icon)
(Prevents reactions across entire app)



textInput(inputId, label, value)

actionButton(inputId, label, icon, …)

actionLink(inputId, label, icon, …)

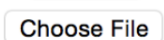


checkboxGroupInput(inputId, label, choices, selected, inline)

checkboxInput(inputId, label, value)

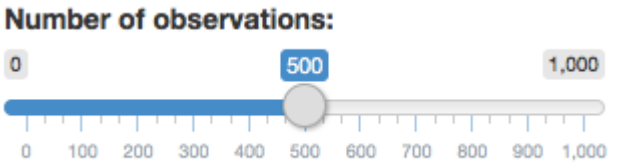
dateInput(inputId, label, value, min, max, format, startview, weekstart, language)

dateRangeInput(inputId, label, start, end, min, max, format, startview, weekstart, language, separator)



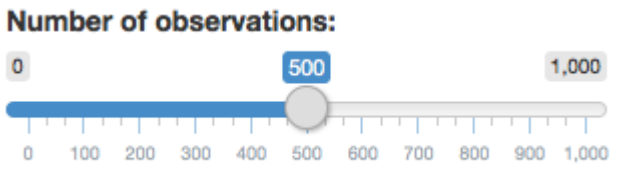
fileInput(inputId, label, multiple, accept)

구문(Syntax)




```
sliderInput(inputId = "obs", label = "Number of observations:",...)
```

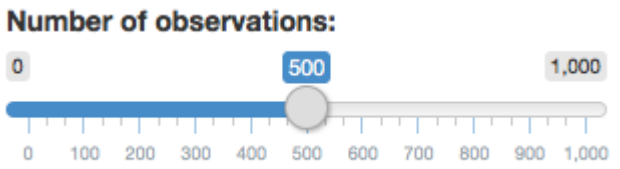
구문(Syntax)




```
sliderInput(inputId = "obs", label = "Number of observations:",...)
```

input name
(for internal use)

구문(Syntax)

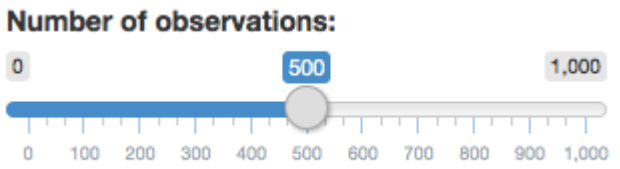



```
sliderInput(inputId = "obs", label = "Number of observations:",...)
```

input name
(for internal use)

label to display

구문(Syntax)




```
sliderInput(inputId = "obs", label = "Number of observations:",...)
```

input name (for internal use)

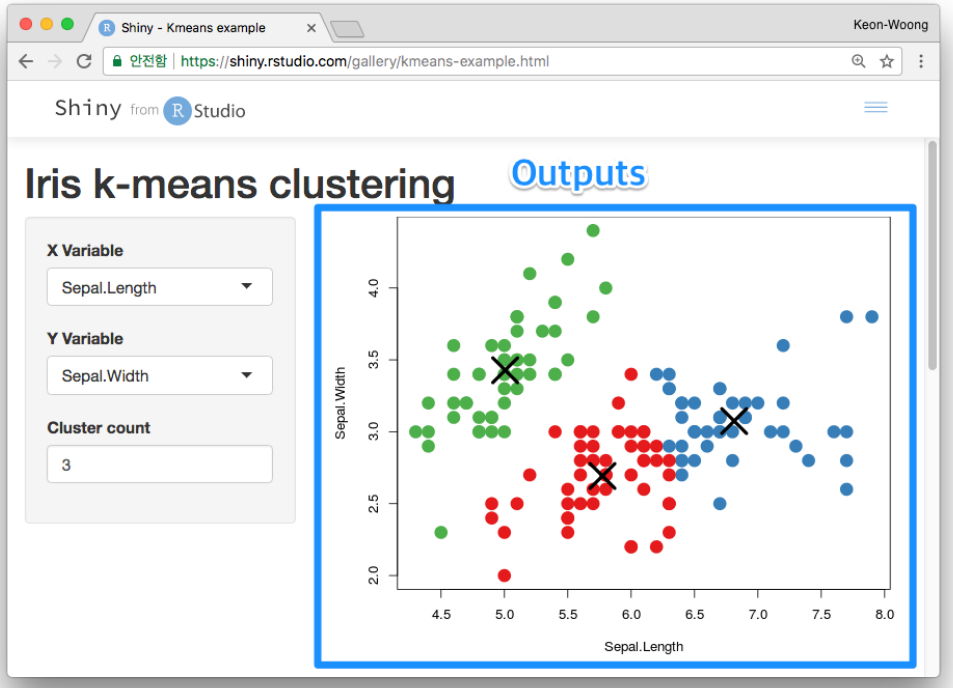
label to display

input-specific argument

?sliderInput

Output()

Outputs


Shiny - Kmeans example
Keon-Woong
안전함 | https://shiny.rstudio.com/gallery/kmeans-example.html
Shiny from Studio
Iris k-means clustering
Outputs
X Variable
Sepal.Length
Y Variable
Sepal.Width
Cluster count
3
Sepal.Width
Sepal.Length
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

Outputs

Function	Inserts
dataTableOutput()	an interactive table
htmlOutput()	raw HTML
imageOutput()	image
plotOutput	plot
tableOutput	table
textOutput	text
uiOutput	a Shiny UI element
verbatimTextOutput	text

*Output()

Output을 UI 에 나타내려면 fluidPage() 함수의 인수로 *Output() 함수를 추가

```
plotOutput(outputId = "distPlot")
```

*Output() 함수를 사용하여 Output 만들기

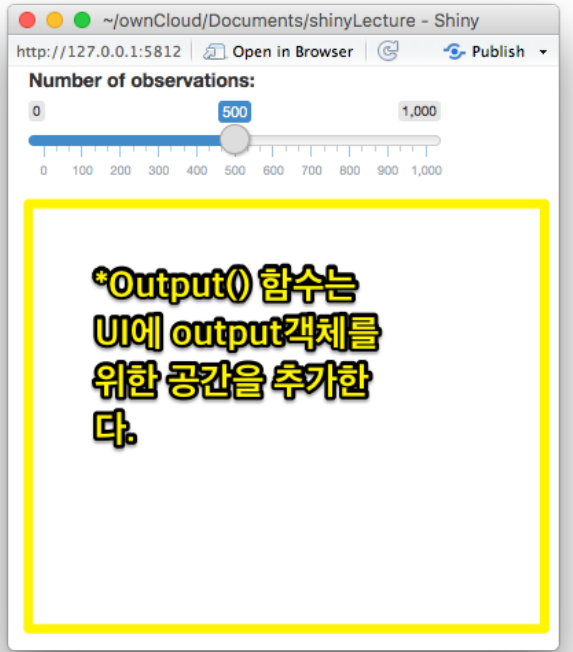
```
library(shiny)

ui <- fluidPage(
  sliderInput(inputId = "obs",
    label = "Number of observati
    min = 0, max = 1000, value =
   plotOutput("distPlot")

)

server <- function(input,output)

shinyApp(ui=ui,server=server)
```



Server function

Server() 함수의 3가지 규칙

```
server <- function(input,output){




}
```

1. UI에 표시할 객체를 output$ 에 저장한다.

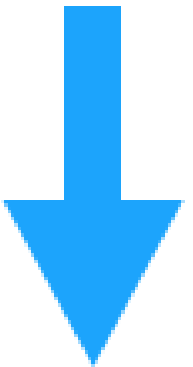
```
server <- function(input,output){

    output$distPlot <- #code


}
```

1. UI에 표시할 객체를 output$ 에 저장한다.

output$distPlot



plotOutput("distPlot")

2. 표시할 객체를 render*() 함수로 만든다.

```
server <- function(input,output){

    output$distPlot <- renderPlot({
          hist(rnorm(100))
    })
}
```

render*() functions

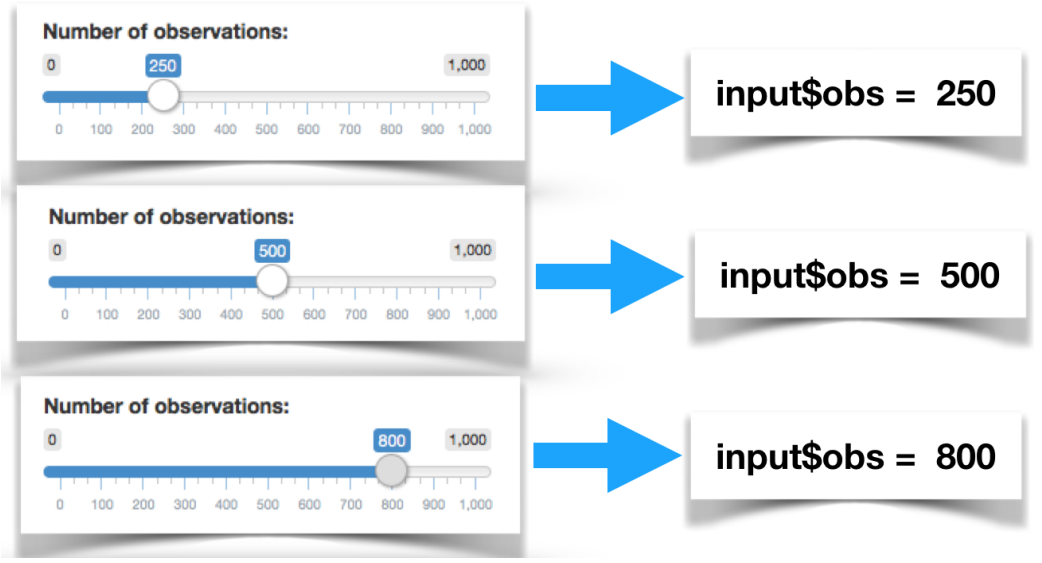
function	creates
renderDataTable()	An interactive table
renderImage	An image(save as a link to a source file)
renderPlot	A plot
renderPrint()	A code block of printed output
renderTable()	A table
renderText()	A characer string
renderUI()	a shiny UI element

3. Input의 값을 input$ 로 사용한다.

sliderInput(inputId="obs",...)

input$obs

input values



3. Input의 값을 input$ 로 사용한다.

```
server <- function(input,output){

    output$distPlot <- renderPlot({
          hist(rnorm(input$obs))
    })
}
```

Reactivity

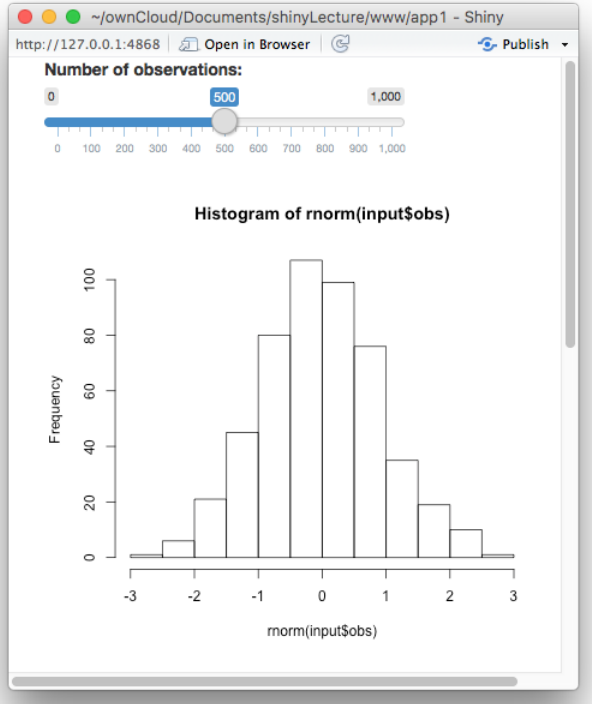
- output 객체를 rendering 하기 위해 input의 값을 사용할 때마다 reactivity가 자동으로 발생한다.

```
server <- function(input,output){

     output$distPlot <- renderPlot({
          hist(rnorm(input$obs))
    })
}
```

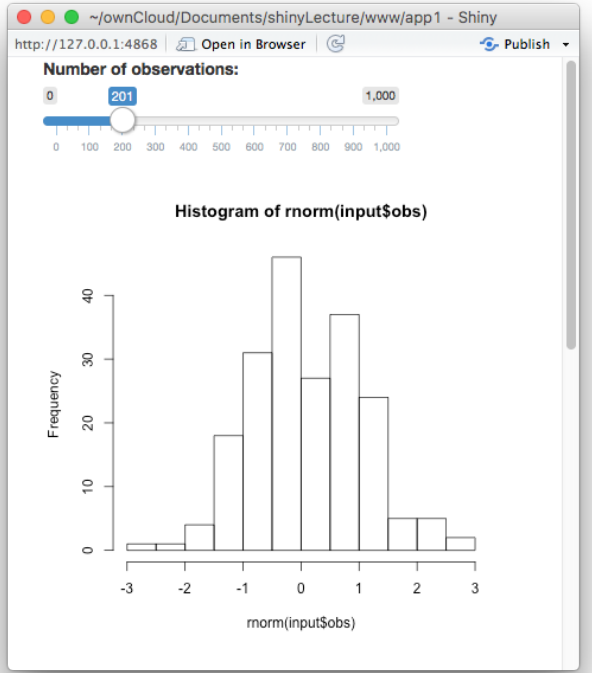
input$obs

```
renderPlot({
    hist(rnorm(input$obs))
})
```



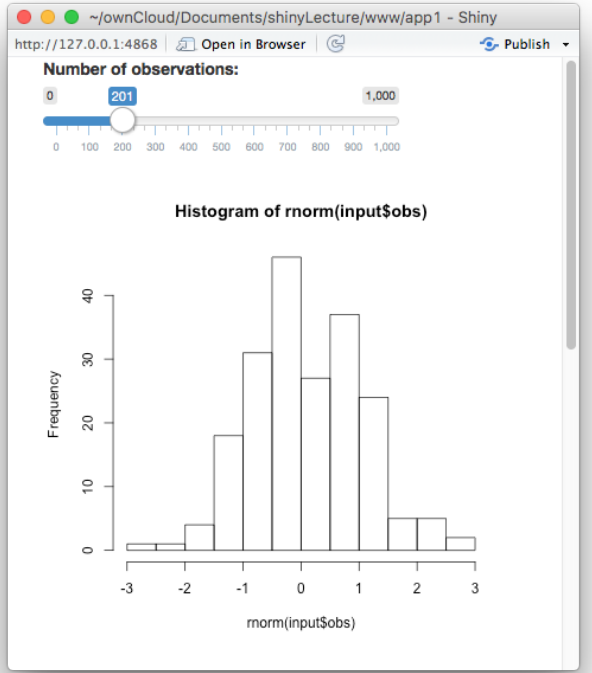
input$obs

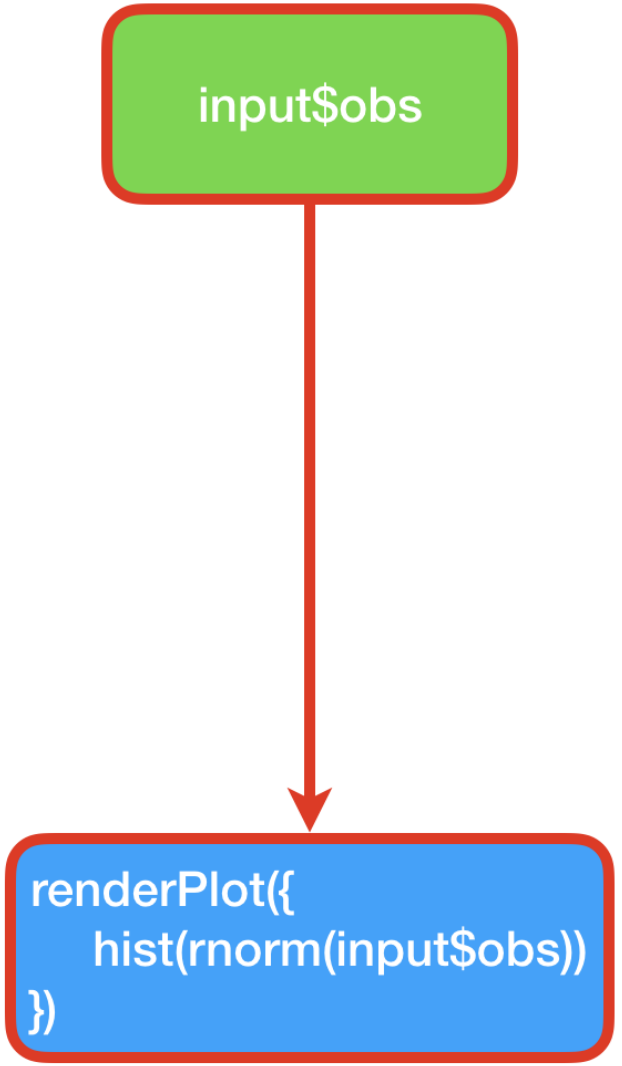
```
renderPlot({
    hist(rnorm(input$obs))
})
```

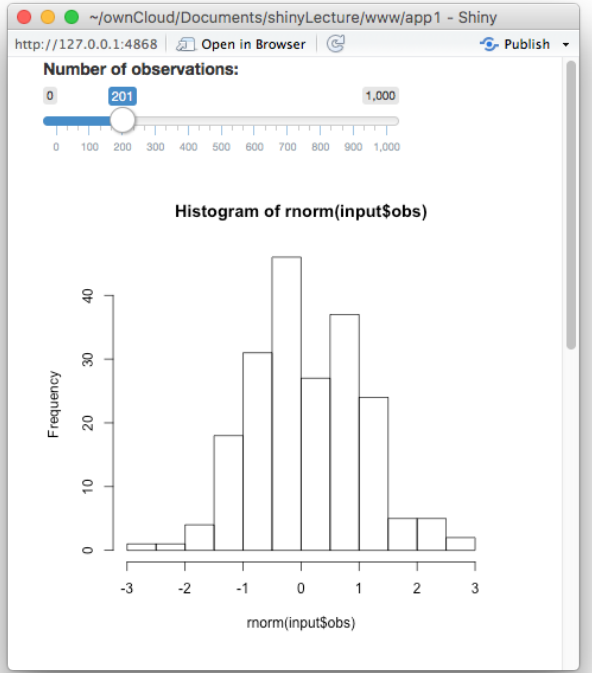


input$obs

```
renderPlot({
    hist(rnorm(input$obs))
})
```

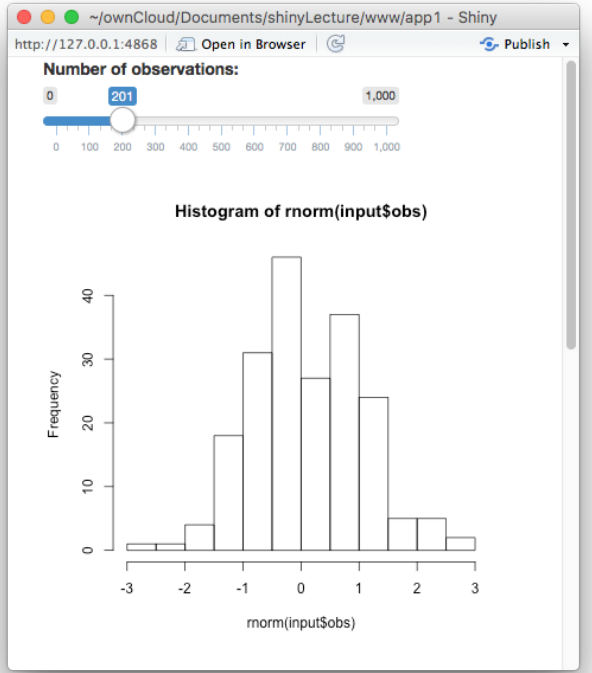



input$obs
renderPlot({
hist(rnorm(input$obs))
})


~/ownCloud/Documents/shinyLecture/www/app1 - Shiny
http://127.0.0.1:4868
Open in Browser
Publish
Number of observations:
0
201
1,000
0 100 200 300 400 500 600 700 800 900 1,000
Histogram of rnorm(input$obs)
Frequency
0 10 20 30 40
-3 -2 -1 0 1 2 3
rnorm(input$obs)

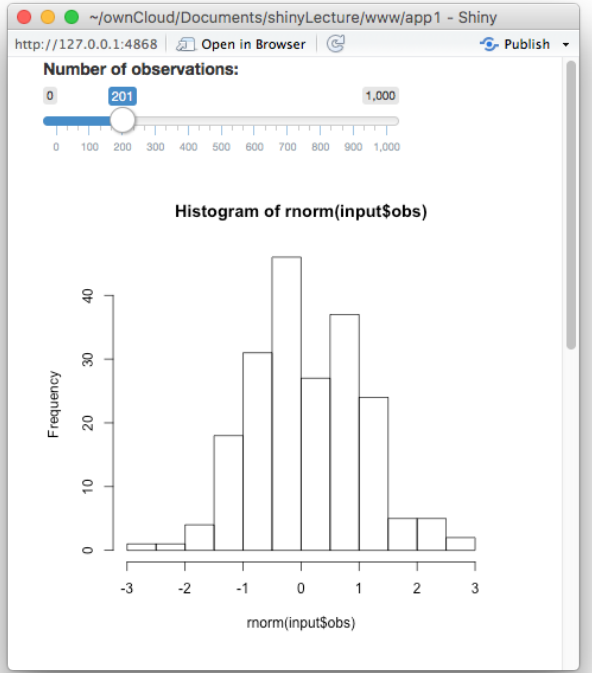
input$obs

```
renderPlot({
    hist(rnorm(input$obs))
})
```



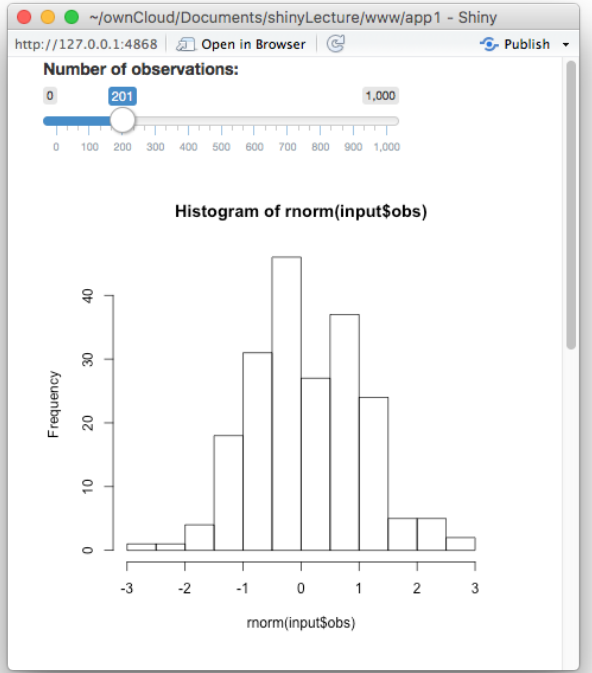
input$obs

```
renderPlot({
    hist(rnorm(input$obs))
})
```



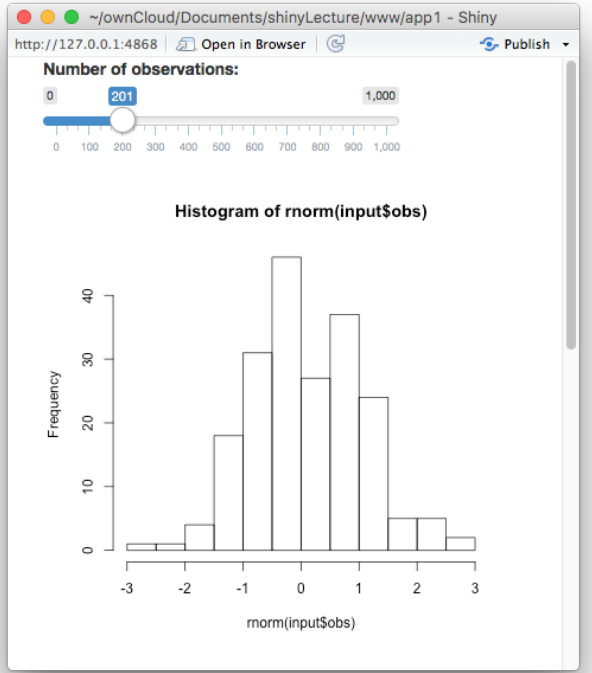
input$obs

```
renderPlot({
    hist(rnorm(input$obs))
})
```



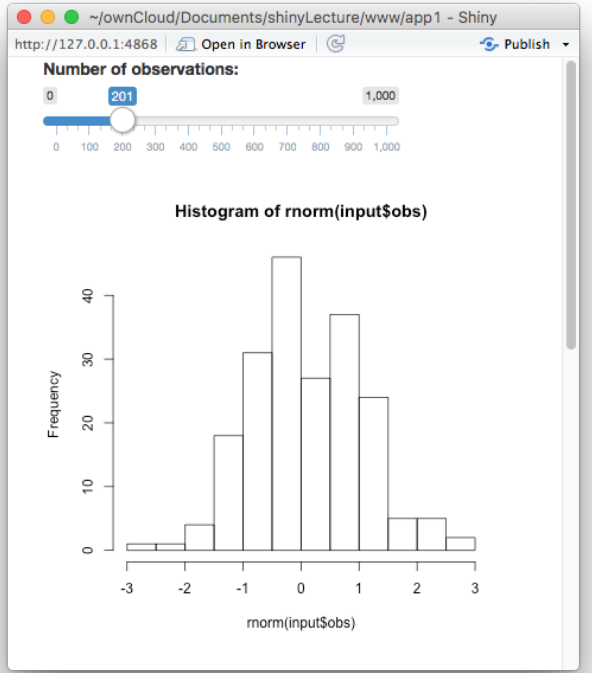
input$obs

```
renderPlot({
    hist(rnorm(input$obs))
})
```



input$obs

```
renderPlot({
    hist(rnorm(input$obs))
})
```



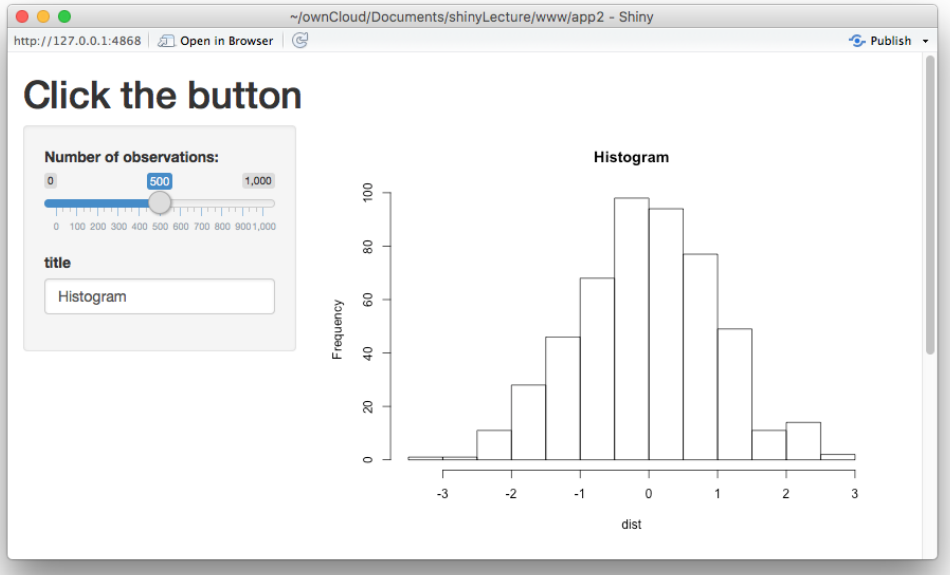
Server function

server 함수내에서 input의 값들을 output으로 전달하기 위해서는

1. output의 객체를 저장할 때 **output$** : output$distPlot
2. output 의 객체를 만들때 **render*()** : renderPlot({})
3. input의 값을 접근할 때는 **input$** : input$obs

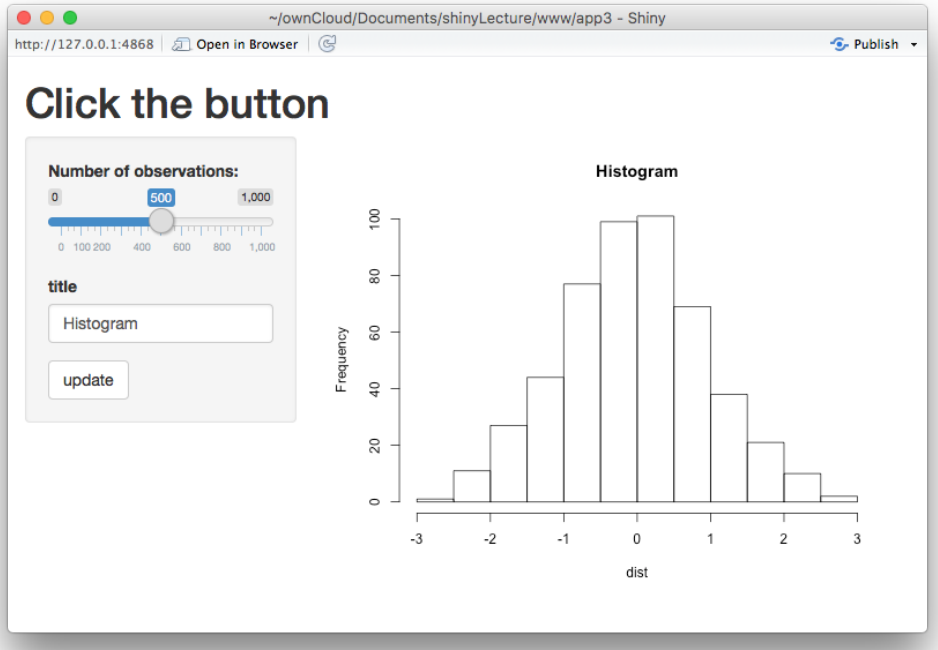
==> input의 값이 변할 때마다 reactivity 가 발생하여 output 객체를 rendering 한다

3. Reactivity(2)



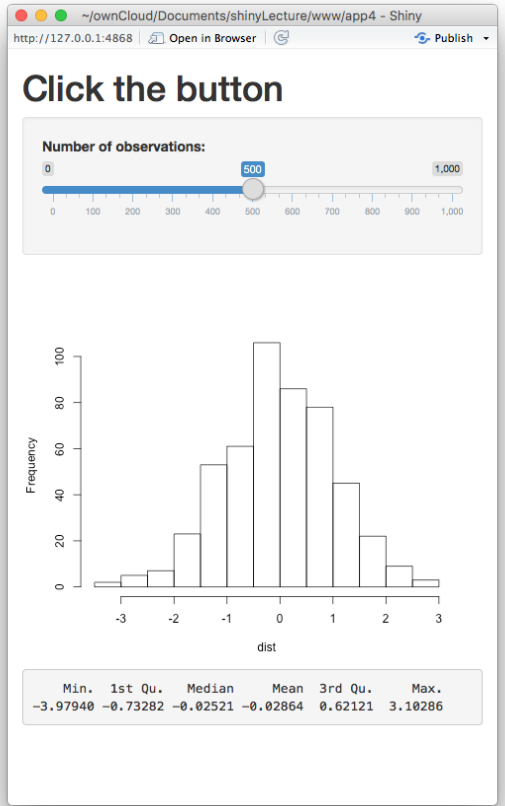

```
shiny::runGitHub('shinyLecture2', 'cardiomoon',subdir='inst/app2')
```

4. Stop reactions with isolate()

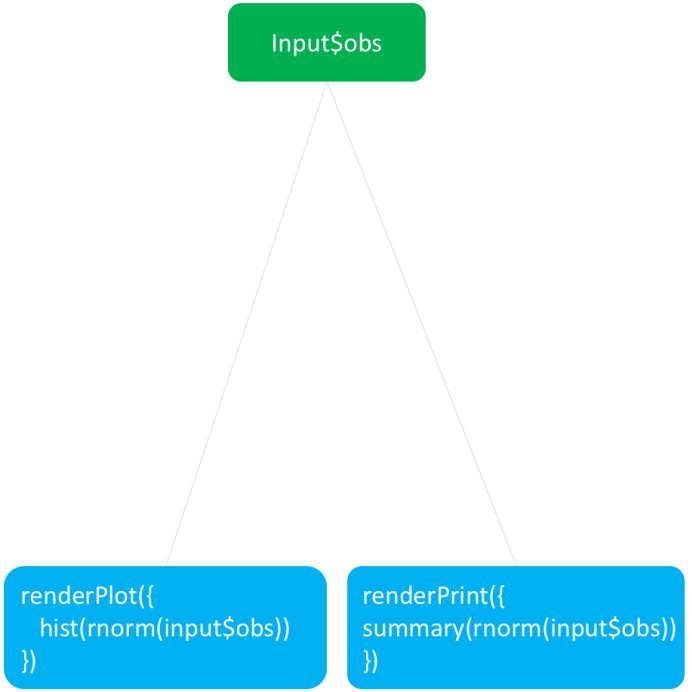


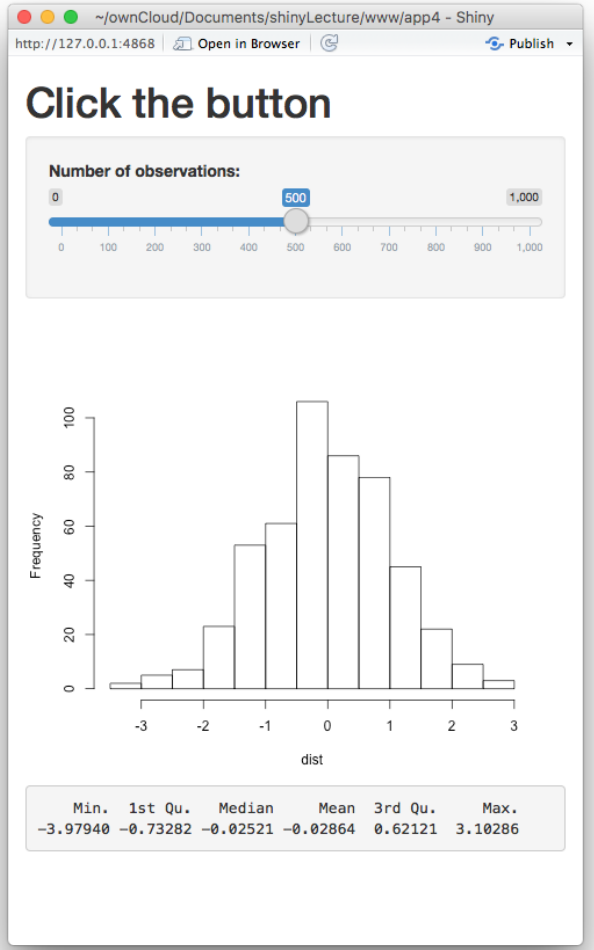

```
shiny::runGitHub('shinyLecture2', 'cardiomoon',subdir='inst/app3')
```

5. one input, two outputs




```
shiny::runGitHub('shinyLecture2', 'cardiomoon',subdir='inst/app4')
```


Input$obs
renderPlot({
hist(rnorm(input$obs))
})
renderPrint({
summary(rnorm(input$obs))
})


~/ownCloud/Documents/shinyLecture/www/app4 - Shiny
http://127.0.0.1:4868
Open in Browser
Publish
Click the button
Number of observations:
0
500
1,000
0
100
200
300
400
500
600
700
800
900
1,000
Frequency
0
20
40
60
80
100
-3
-2
-1
0
1
2
3
dist
Min. 1st Qu. Median Mean 3rd Qu. Max.
-3.97940 -0.73282 -0.02521 -0.02864 0.62121 3.10286

reactive()

- reactive 함수로 반응성 객체를 만든다.

```
data <-reactive({rnorm(input$obs)})
```

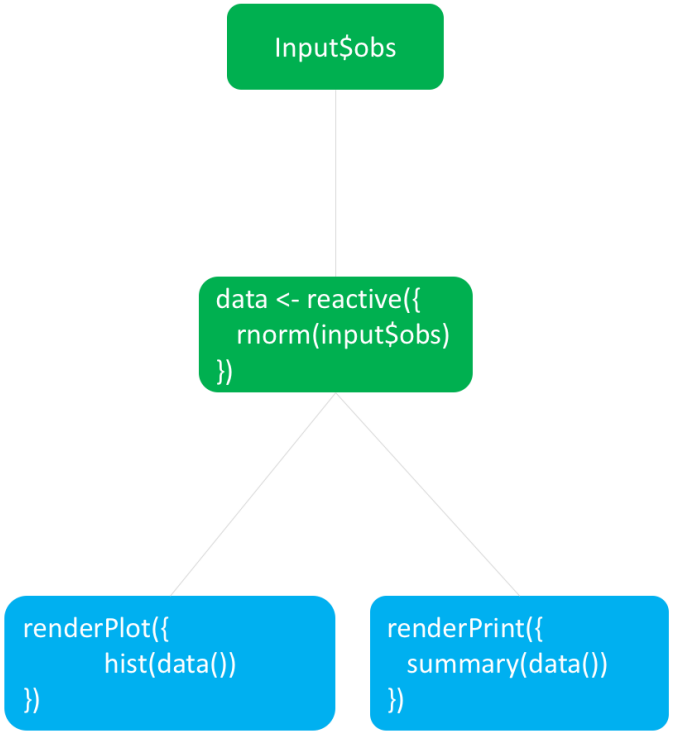
- 이 객체는 reactive value가 변할 때마다 반응한다.

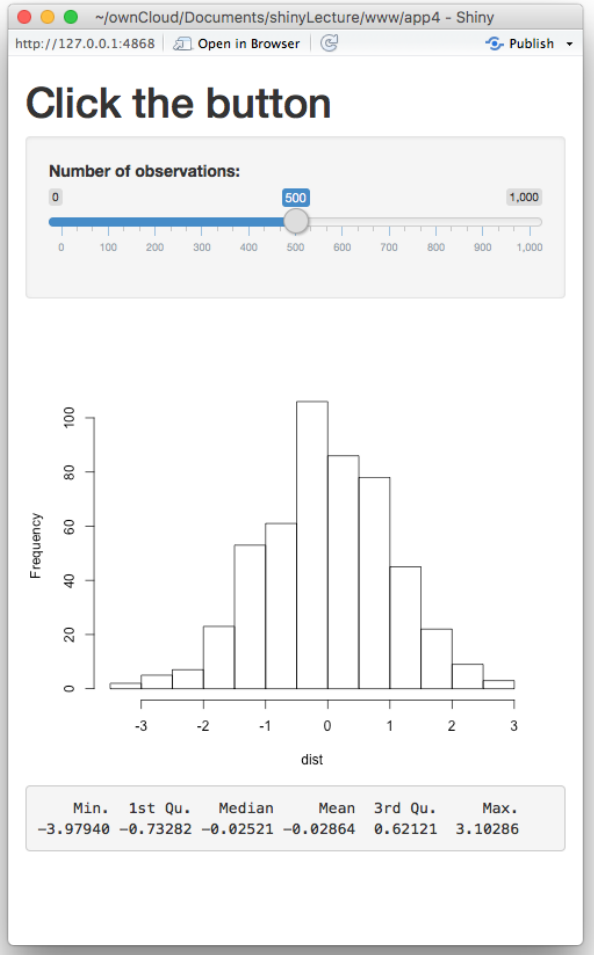
반응성 객체

반응성 객체의 두가지 면에서 특별하다.

```
data()
```

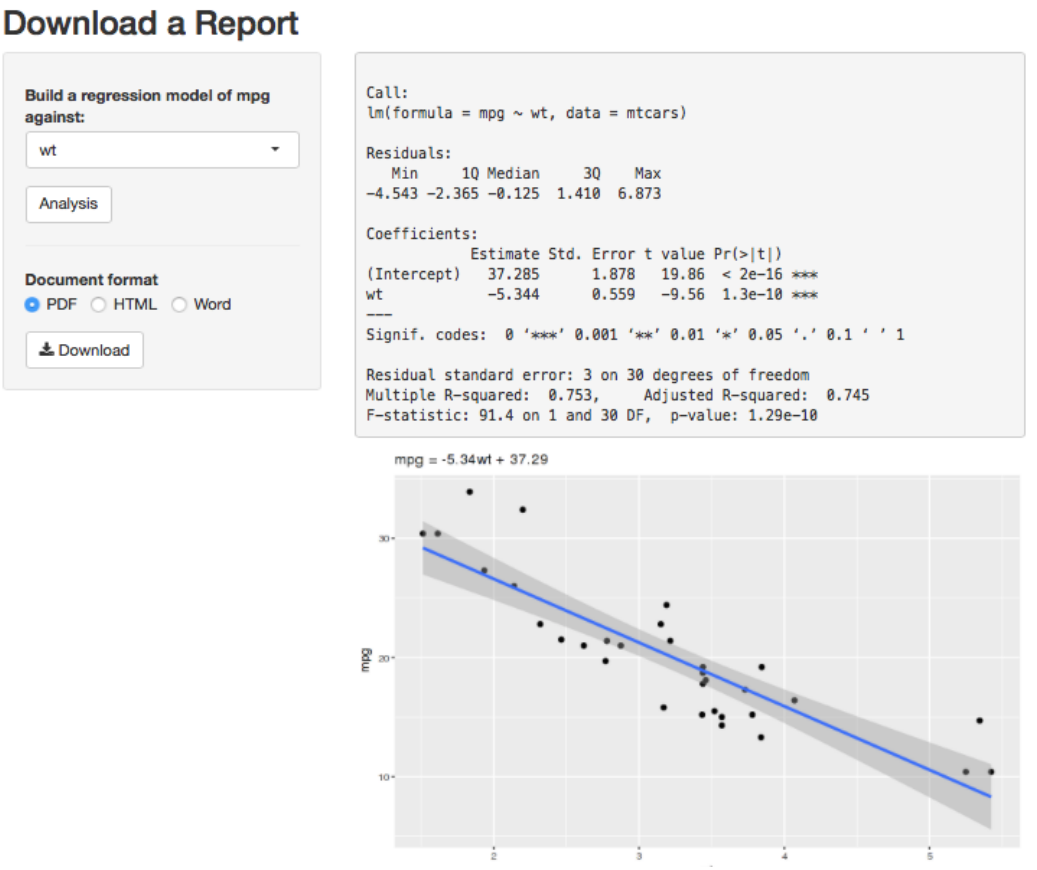
- 반응성 객체를 호출할 때는 함수처럼 호출한다.
- 반응성 객체는 그 값을 임시로 저장한다(cache).
    - 무효화되지 않을 경우 가장 최근의 값을 반환한다.






```
shiny::runGitHub('shinyLecture2', 'cardiomoon',subdir='inst/app5')
```

6. Download knitr Reports




```
shiny::runGitHub('shinyLecture2', 'cardiomoon',subdir='inst/app13')
```

Report.Rmd File

shinyLecture.Rmd | ShinyGadget.Rmd | test.Rmd | server.R | ui.R | Untitled1 | report.Rmd

Knit | Insert | Run

```
---
title: "Regression Analysis"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE,comment=NA)
```

Summary of Regression Model:

```{r model, echo=FALSE}
options(digits = 2)
fit <- eval(parse(text=paste0("lm( mpg ~",input$x,",data = mtcars)")))
b   <- coef(fit)
```

```{r}
summary(fit)
```

The fitting result is $mpg = `r b[2]`r input$x`+`r b[1]`$.
Below is a scatter plot with the regression line.

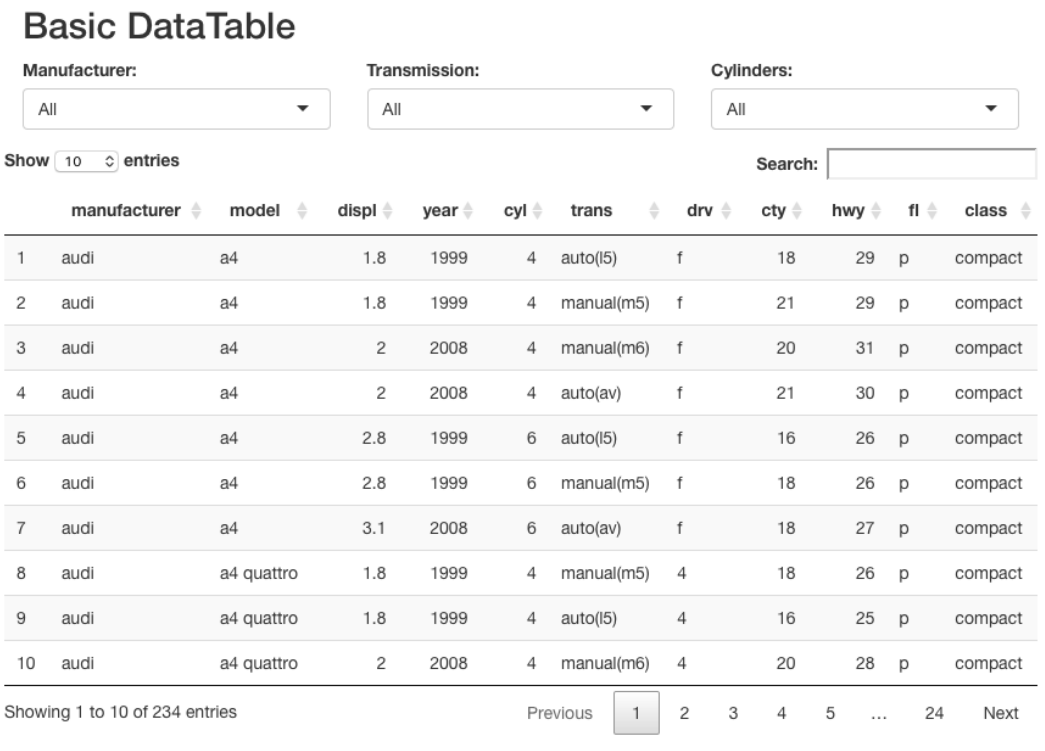
```{r plot, echo=FALSE,fig.height=4}
ggplot(data=mtcars,aes_string(req(input$x),"mpg"))+
          geom_point()+
          geom_smooth(method="lm")+
          ggtitle(regEquation())
```
```

https://github.com/cardiomoon/shinyLecture2/tree/master/inst/app13

PDF 다운로드를 위해서는

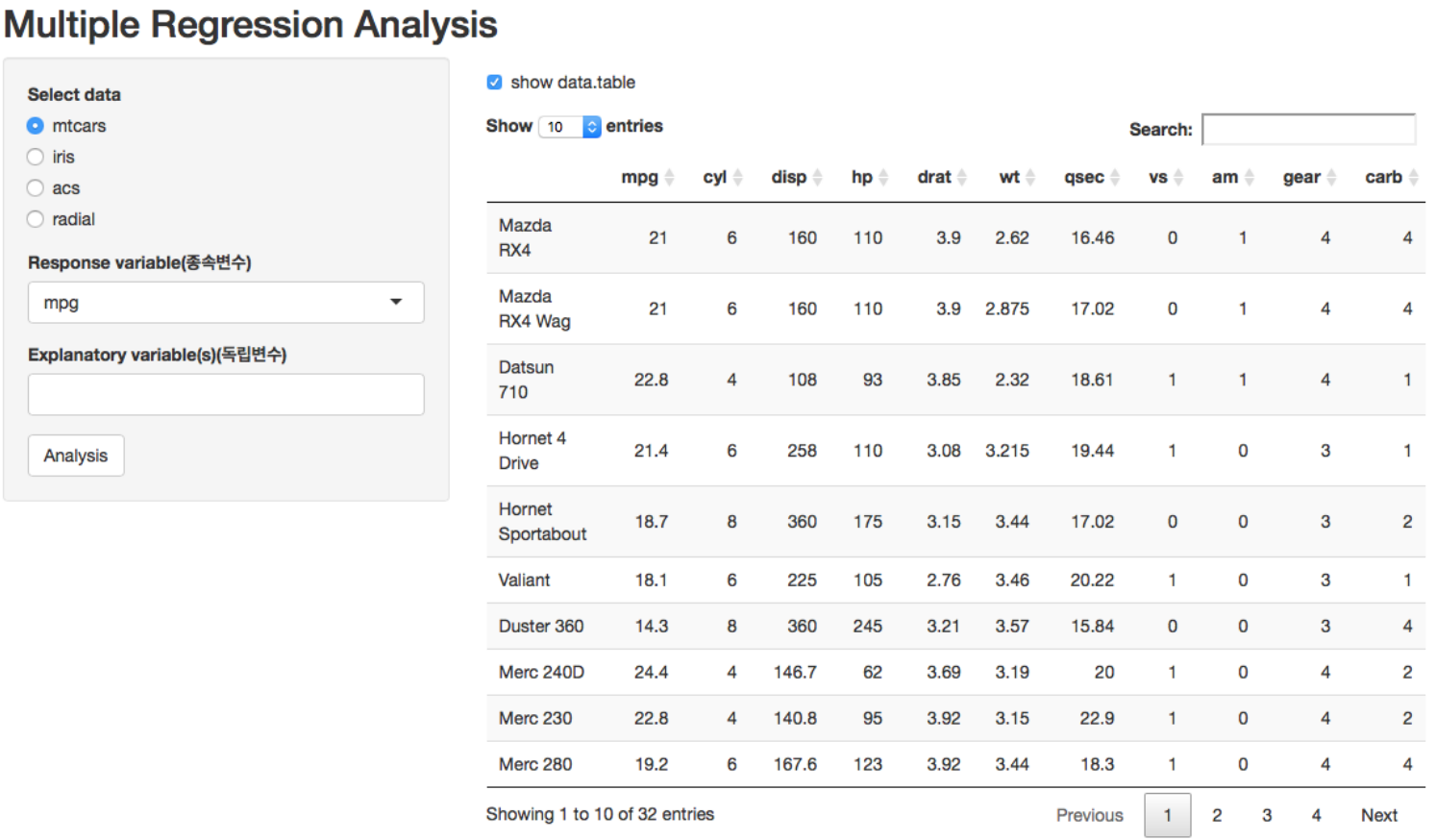
- 자신의 컴퓨터에 LaTex이 설치되어 있어야 한다. (http://ktug.or.kr)
- 또는 LaTex가 설치된 shiny server에서 shiny app을 실행하여야 한다.

7. Basic DataTable




```
shiny::runGitHub('shinyLecture2', 'cardiomoon',subdir='inst/app14')
```

8. Advanced App - Multiple Reactive Outputs




```
shiny::runGitHub('shinyLecture2', 'cardiomoon',subdir='inst/app15')
```