

# R과 Shiny를 이용한 Web Application의 제작(I)

문건웅

2018/3/21

# 강사소개

문건웅

- 가톨릭대학교 의과대학 교수
- 성빈센트병원 순환기내과 재직
- R packages (CRAN)
  - mycor, moonBook, ztable(2015)
  - ggiraphExtra(2016)
  - dplyrAssist, editData, ggplotAssist(2017)
- 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

# Shiny로 어떤 앱을 만들 수 있나?

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

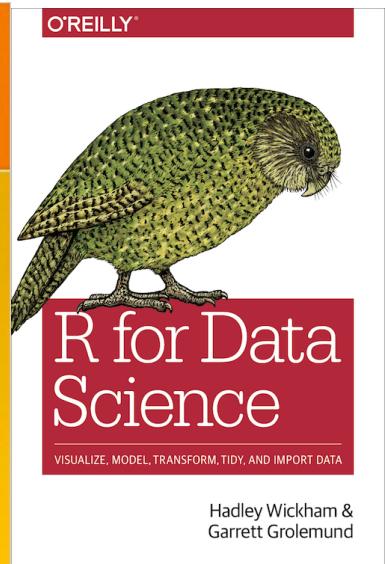
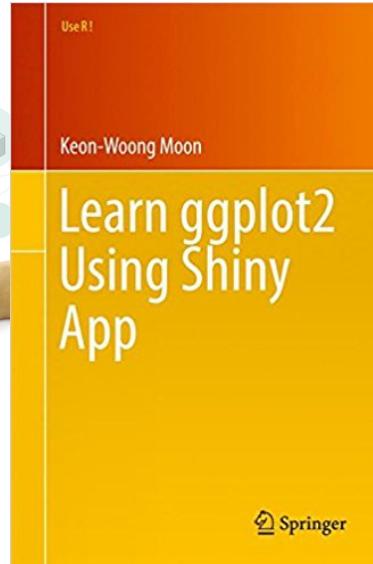
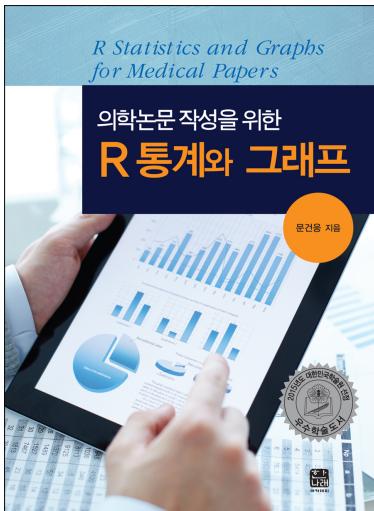
The screenshot displays the RStudio Shiny User Showcase website. At the top, there's a navigation bar with links for Products, Resources (which is currently selected), Pricing, About Us, and Blogs. Below the navigation is a search bar. The main content area features a heading "Shiny Apps for the Enterprise" followed by four examples:

- MARKETING EFFECTS**: Shows a dashboard with multiple charts and graphs.
- LOCATION TRACKER**: Displays a map with location data over time.
- DOWNLOAD MONITOR**: Visualizes streaming download rates using a bubble chart.
- PERSISTENT STORAGE**: Shows a data storage interface with tables and dropdown menus.

Below this section is another heading "Industry Specific Shiny Apps" followed by four more examples:

- TOURISM DASHBOARD**: Shows a dashboard for tourism data.
- GENOME BROWSER**: Displays a circular genome browser interface.
- ER OPTIMIZATION**: Shows a simulation for emergency room optimization.
- SUPPLY AND DEMAND**: Visualizes supply and demand data for police resources.

# R 을 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("knitr", "shiny", "rmarkdown")
install.packages("tidyverse", "DT", "moonBook")
```

- 6번째 앱에서 knitr Reports 중 pdf 다운로드를 위하여는 LaTeX 설치가 필요하다. (<http://ktug.or.kr>)
- 9번째 interactive plot 을 위해서는 다음 패키지들의 설치가 필요하다.

```
install.packages("ggiraph", "ggiraphExtra")
```

# 예제 파일 및 앱 소스파일

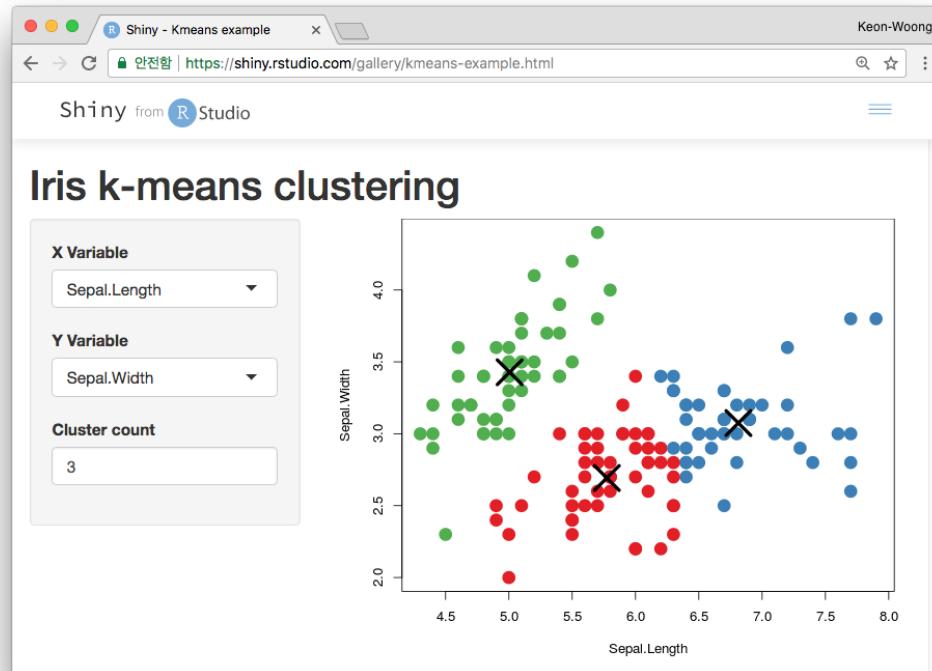
이번 강의에 사용되는 앱 및 소스 파일들은 다음 `github`에서 다운로드 받을수 있다.

<https://github.com/cardiomoon/shinyLecture2>

# Introduction of Shiny

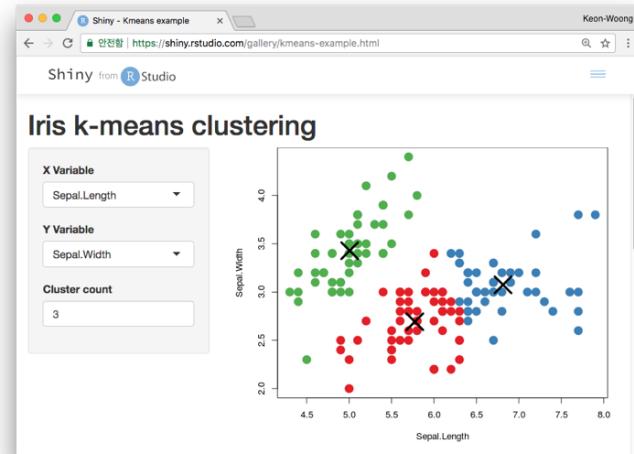
1. The First Shiny App
2. Shiny App Template 사용
3. The 2nd App: Reactivity
4. \*Input()
5. \*Output()
6. Server function()
7. Share Your Shiny App

# 1. The First Shiny App



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

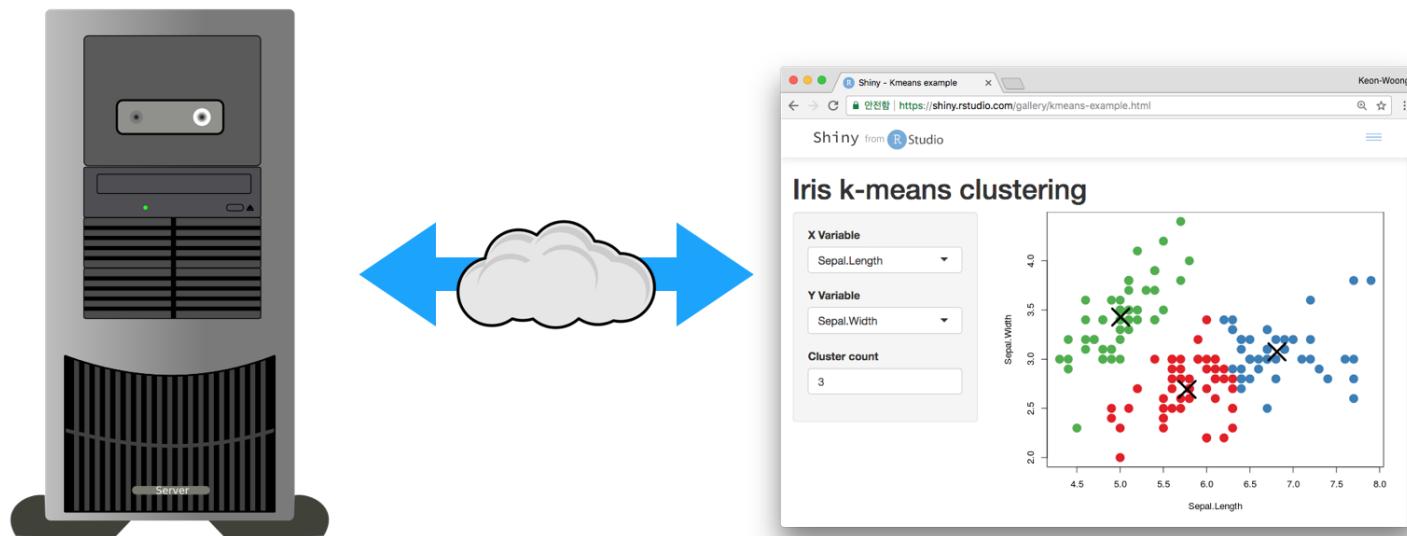
# Shiny App은 R을 운영하는 컴퓨터에 의해 유지된다.



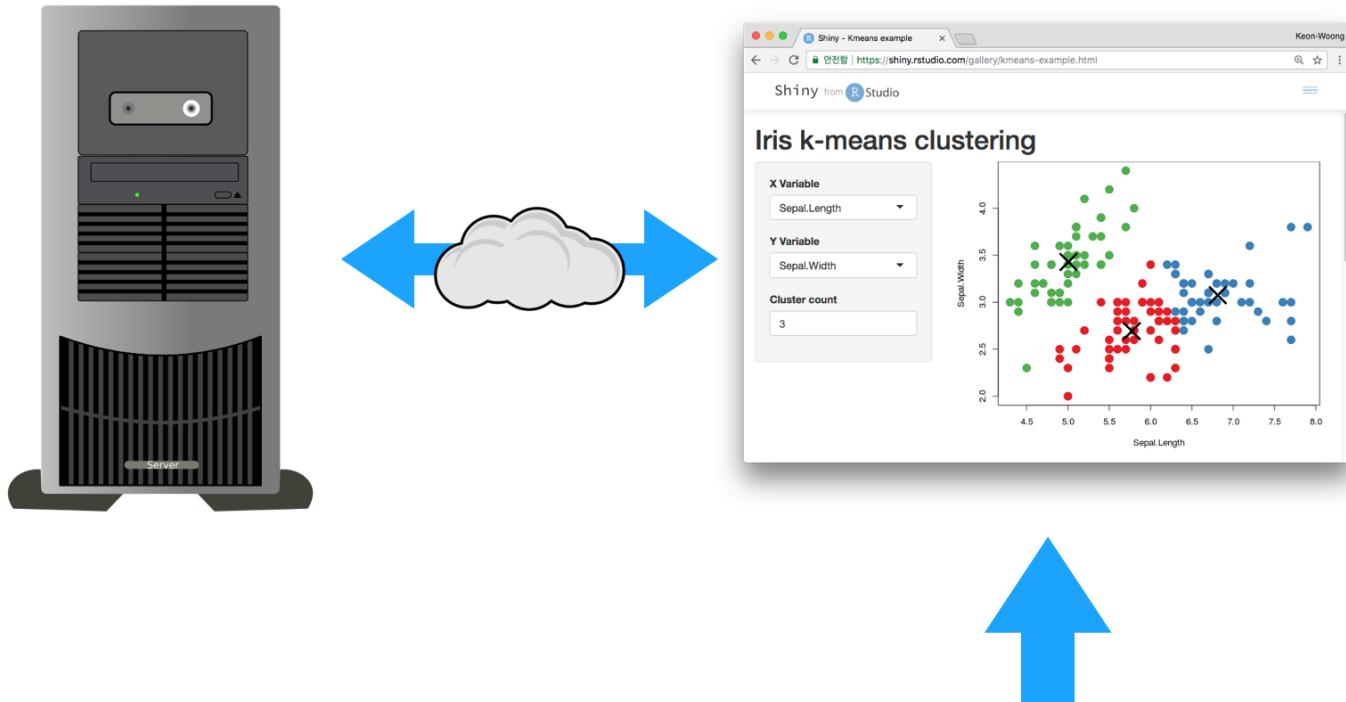
R console에서 다음 R 명령어를 실행시켜 보자

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

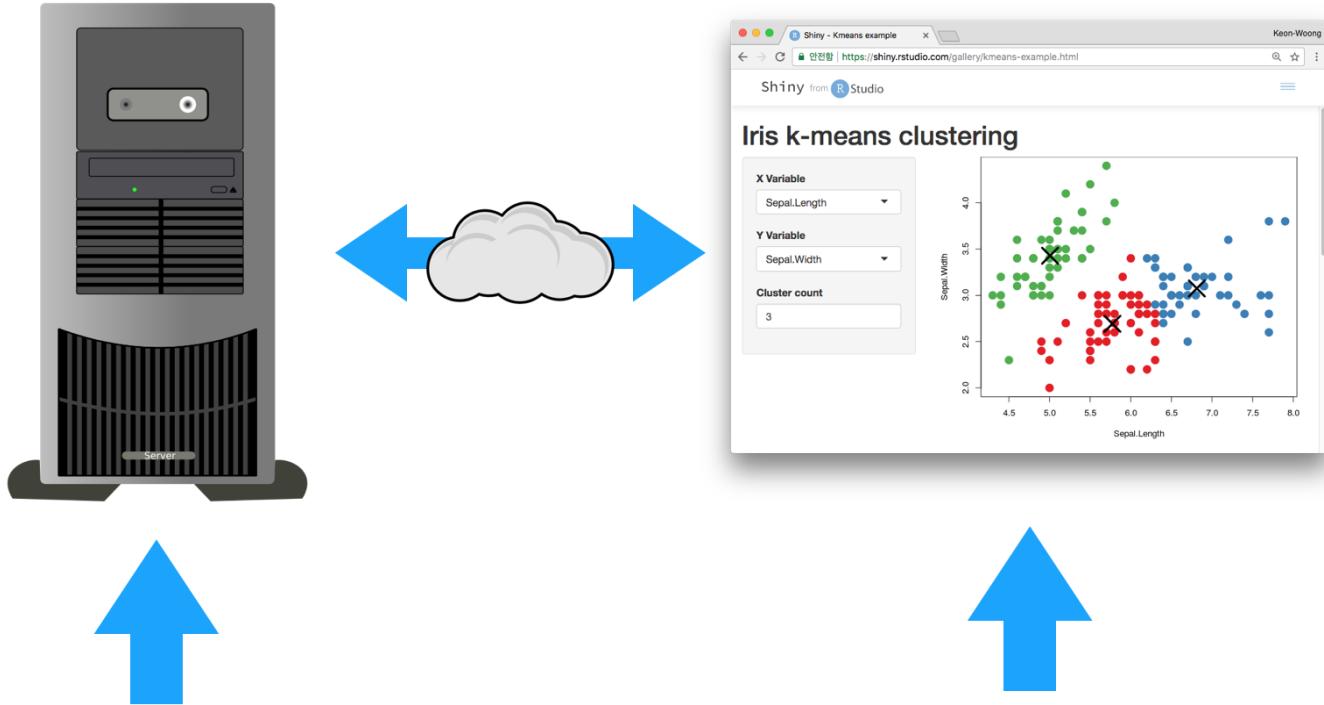
# 어떤 Shiny App은 R을 운영하는 서버에 의해 유 지된다.



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



## User Interface



**Server Instructions**

**User Interface**

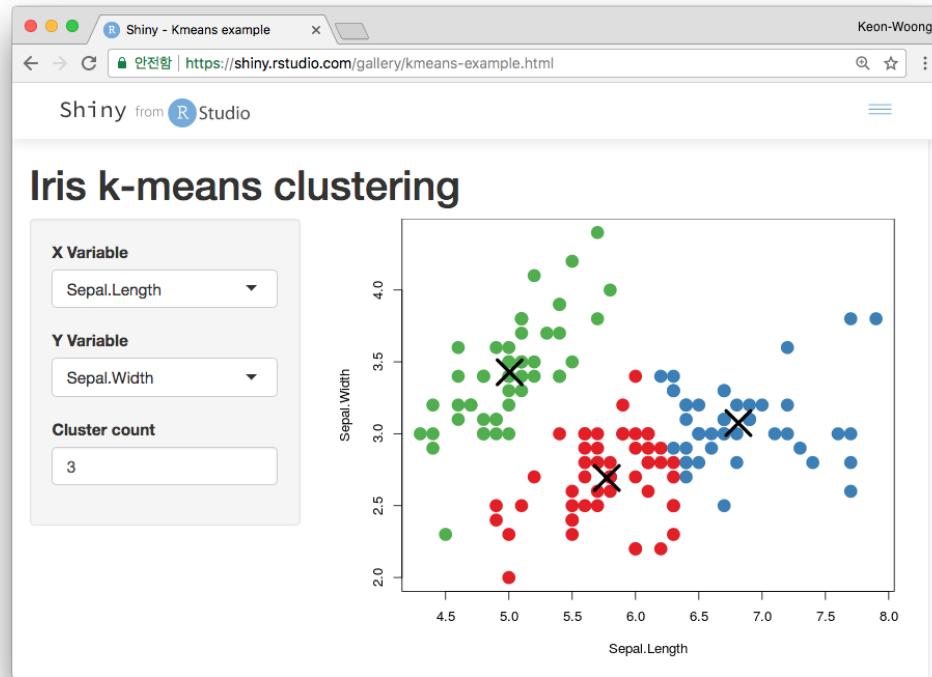
# app.R

<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) {
```

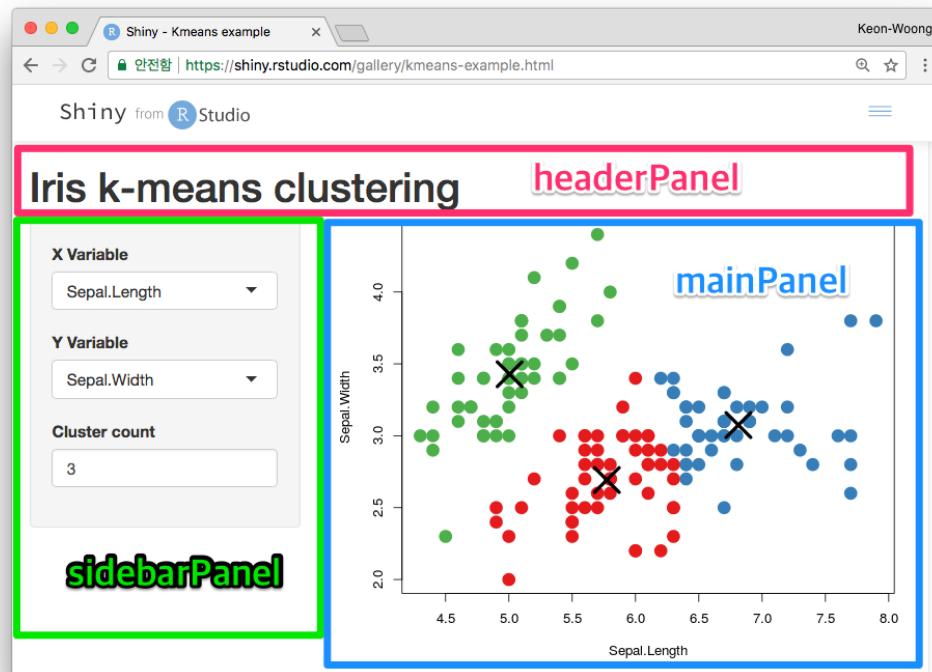
# Shiny App



# 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

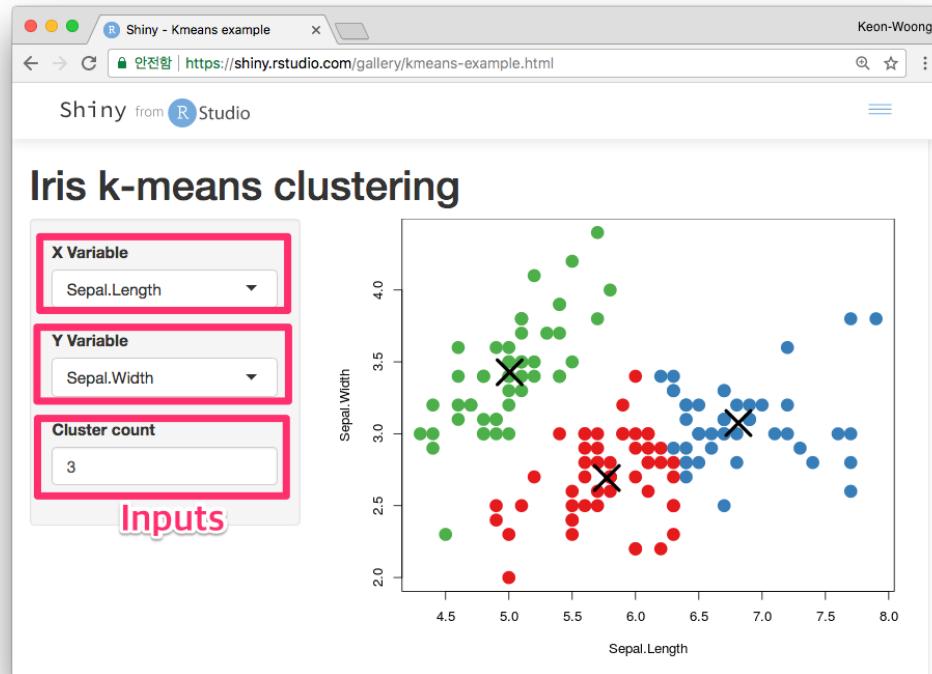


# Inputs

```
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')
  )
)
```

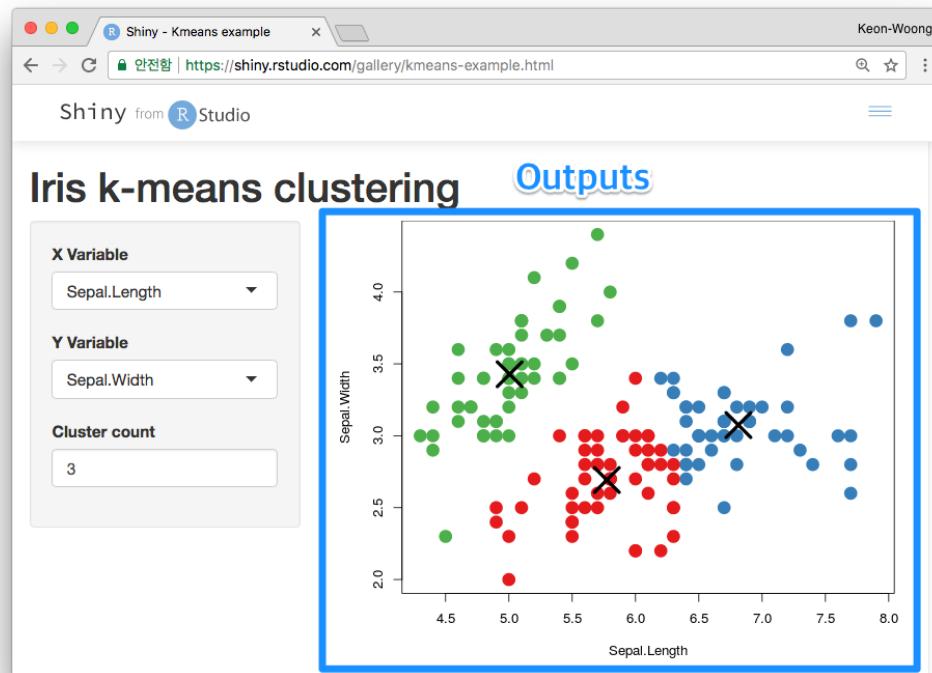
# Inputs



# 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 App Template 사용

# Minimal Valid Shiny App

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

```
library(shiny)

ui <- fluidPage()

server <- function(input,output){}

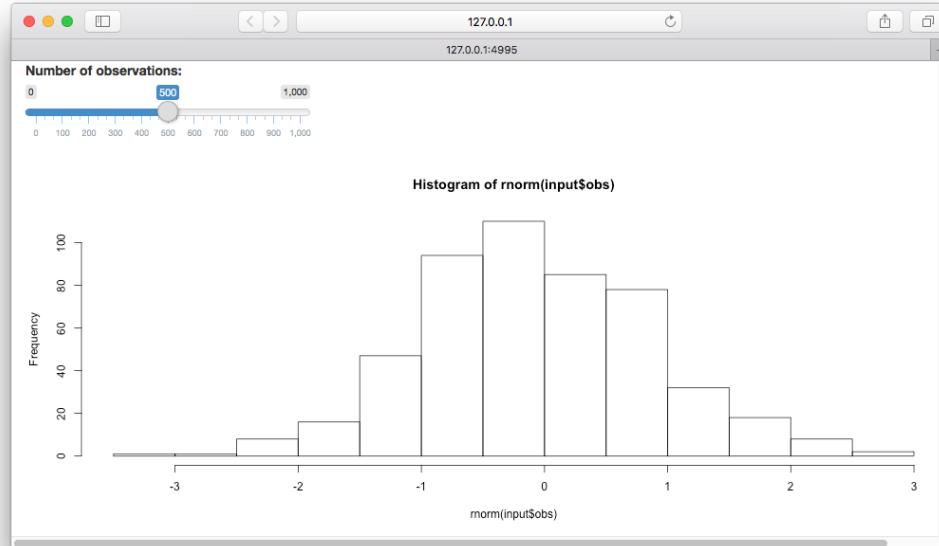
shinyApp(ui=ui,server=server)
```

# 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>
```

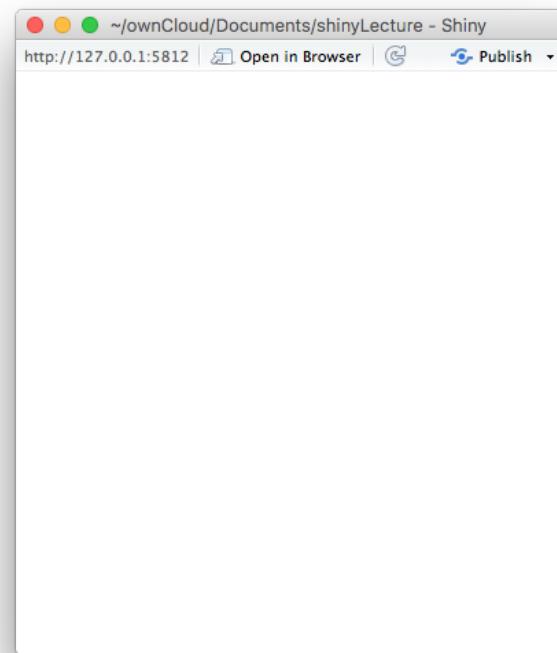
# \*input() 함수를 사용하여 input 만들기

```
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 observations",
    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)

- Choice A
- Choice B
- Choice C

Choice 1

Choice 1  
Choice 2

0 5 10

0 2 4 6 8 10

Apply Changes

Enter text

**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)

Action

**actionButton**(inputId, label, icon, ...)

Link

**actionLink**(inputId, label, icon, ...)

- Choice 1
- Choice 2
- Choice 3

Check me



Choose File

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

**checkboxInput**(inputId, label, value)

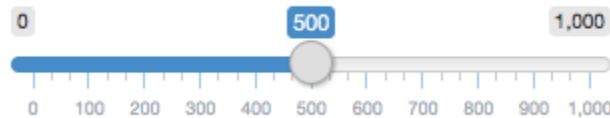
**dateInput**(inputId, label, value, min, max, format, startview, weekstart, language)

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

**fileInput**(inputId, label, multiple, accept)

# 구문(Syntax)

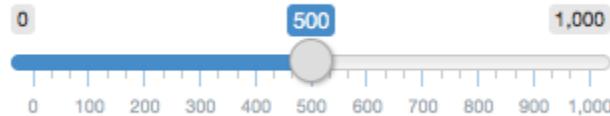
Number of observations:



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

# 구문(Syntax)

Number of observations:

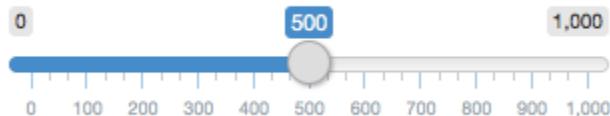


```
sliderInput(inputId = "obs", label = "Number of observations:", ...)
```

input name  
(for internal use)

# 구문(Syntax)

Number of observations:



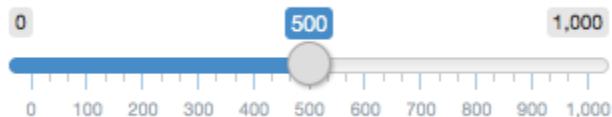
```
sliderInput(inputId = "obs", label = "Number of observations:", ...)
```

input name  
(for internal use)

label to display

# 구문(Syntax)

Number of observations:



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

input name  
(for internal use)

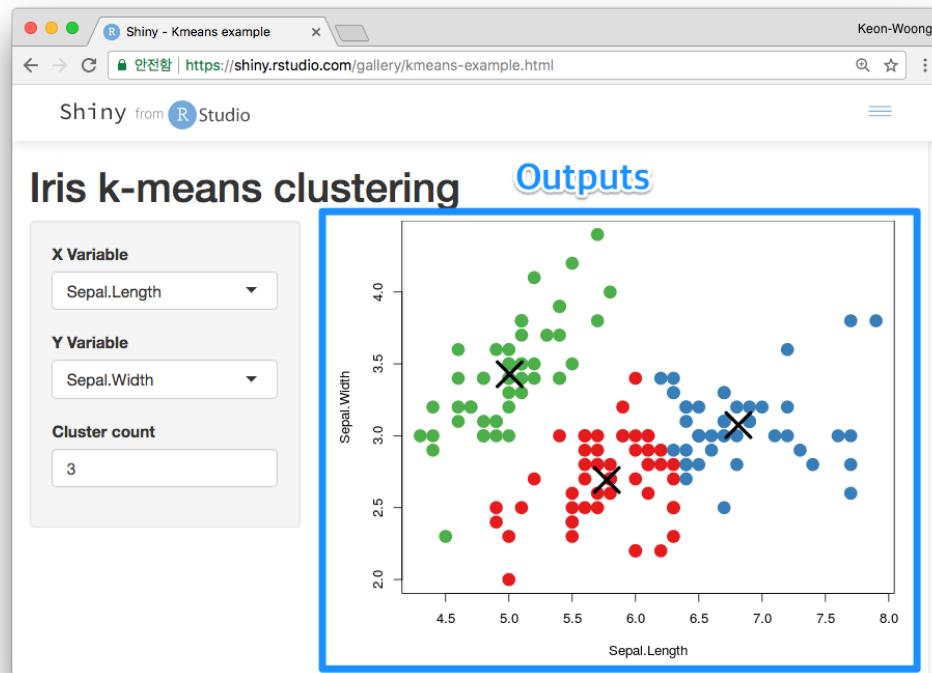
label to display

input-specific  
argument

?sliderInput

# Output()

# Outputs



# 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 UI에 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 observations",
    min = 0, max = 1000, value =
    plotOutput("distPlot"))

)
server <- function(input,output)
shinyApp(ui=ui,server=server)
```



# Server function

# Server() 함수의 3가지 규칙

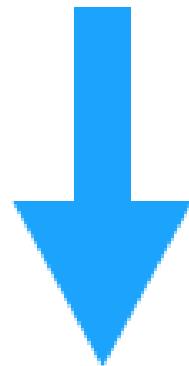
```
server <- function(input,output){  
}  
}
```

# 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

<b>function</b>	<b>creates</b>
<code>renderDataTable()</code>	An interactive table
<code>renderImage</code>	An image(save as a link to a source file)
<code>renderPlot</code>	A plot
<code>renderPrint()</code>	A code block of printed output
<code>renderTable()</code>	A table
<code>renderText()</code>	A character string
<code>renderUI()</code>	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))  
  })  
}
```

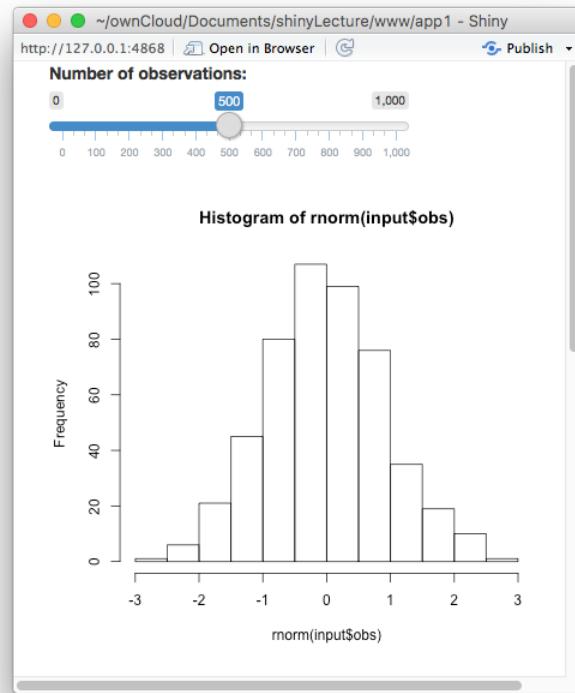
# Reactivity

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

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

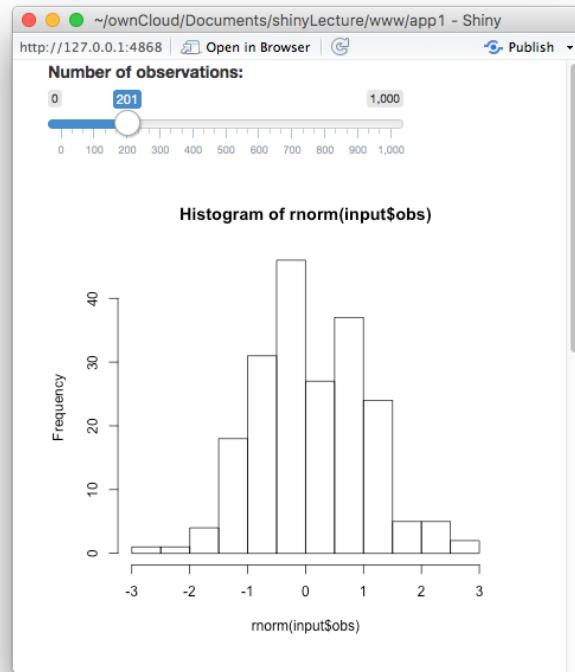
input\$obs

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

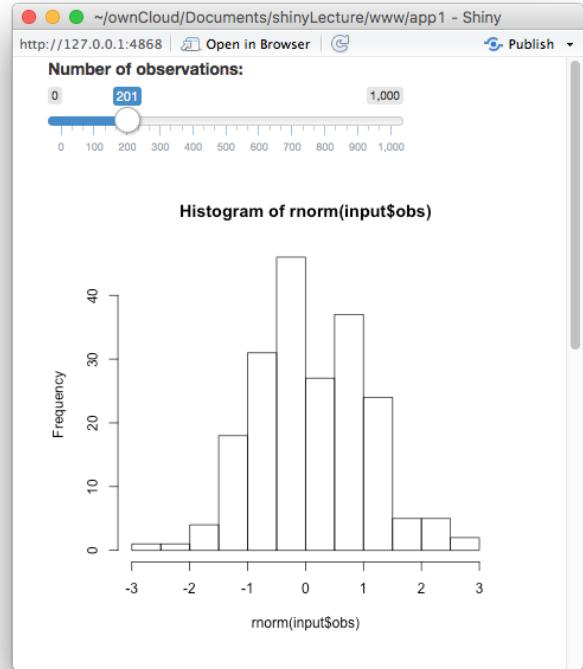


input\$obs

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



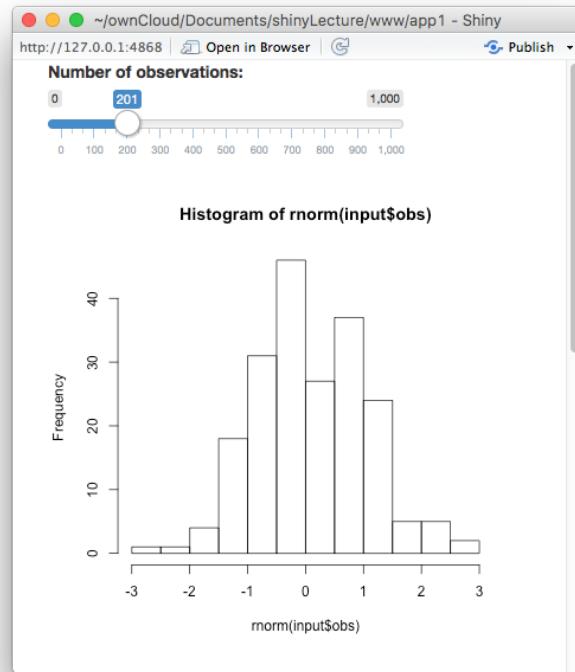
input\$obs



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

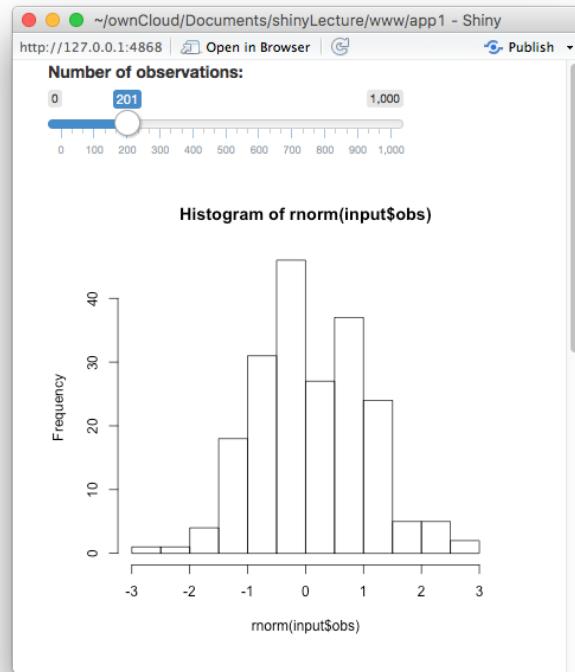
input\$obs

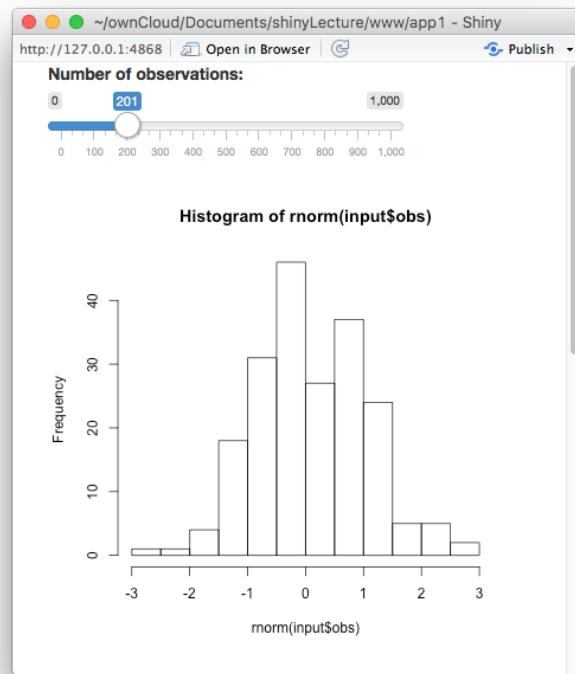
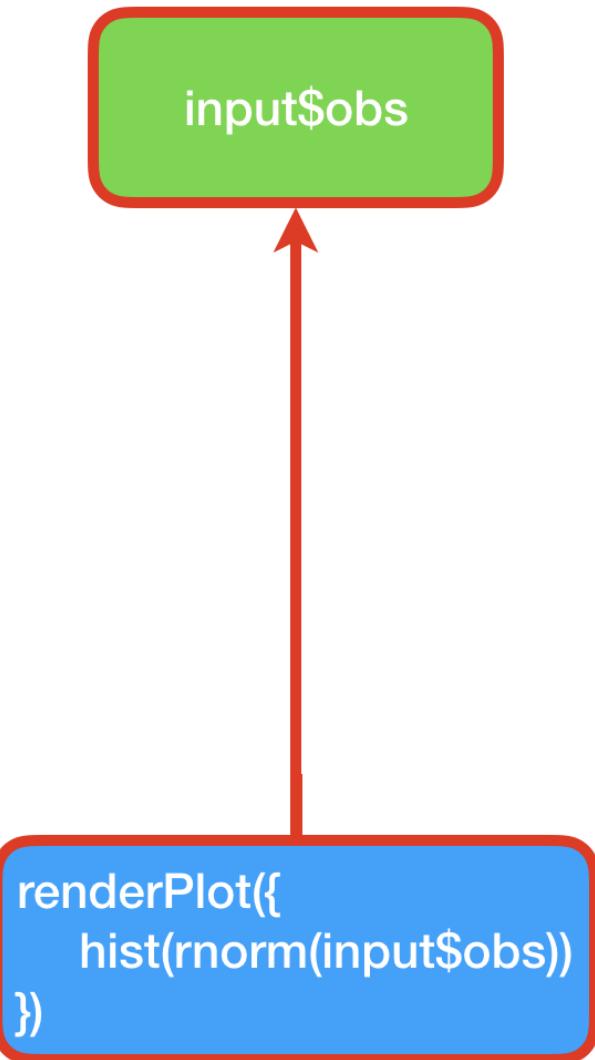
```
renderPlot({  
  hist(rnorm(input$obs))  
})
```



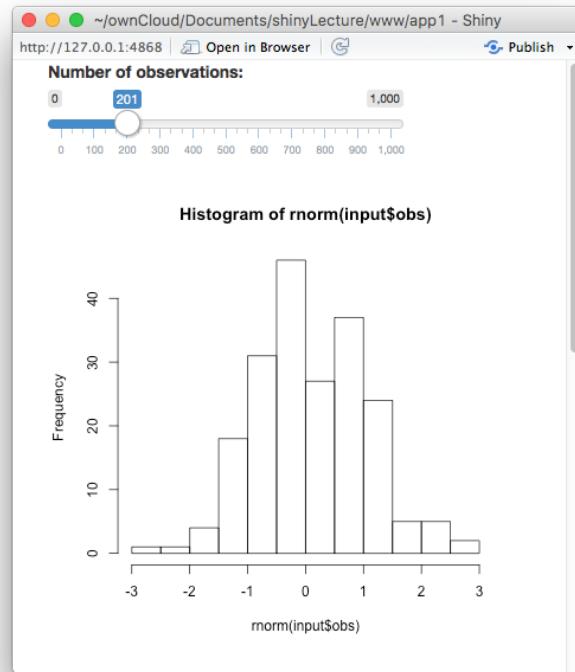
input\$obs

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



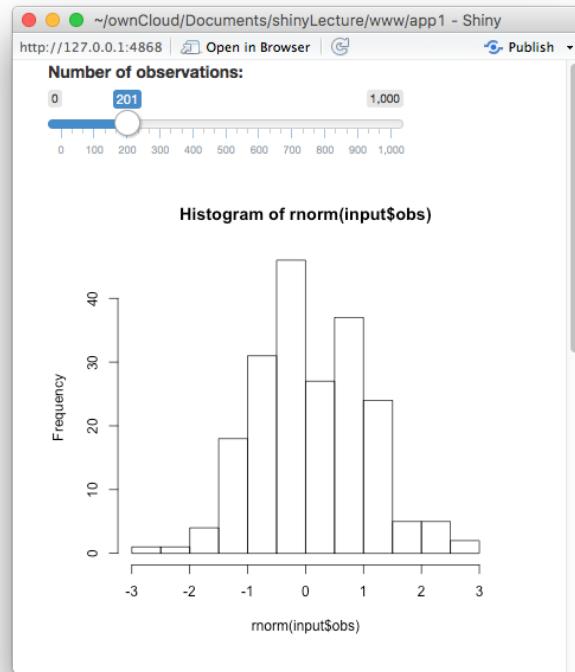


input\$obs



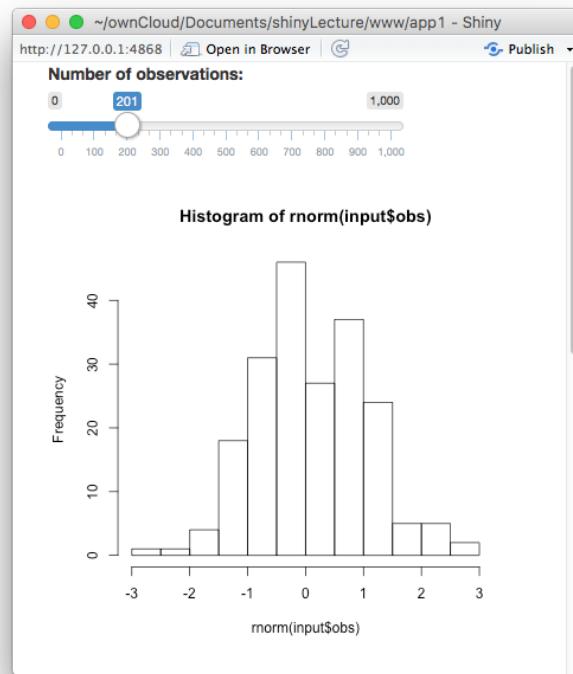
input\$obs

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



input\$obs

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



# Server function

server 함수내에서 input의 값을 output으로 전달하기 위해서는

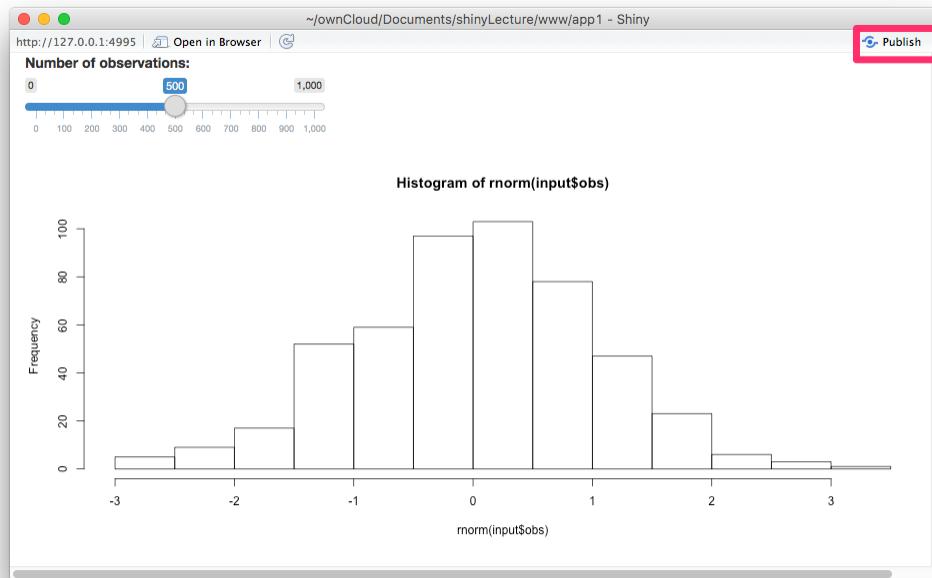
1. output의 객체를 저장할 때 **output\$** : output\$distPlot
2. output 의 객체를 만들때 **render\***(**o**) : renderPlot({})
3. input의 값을 접근할 때는 **input\$** : input\$obs

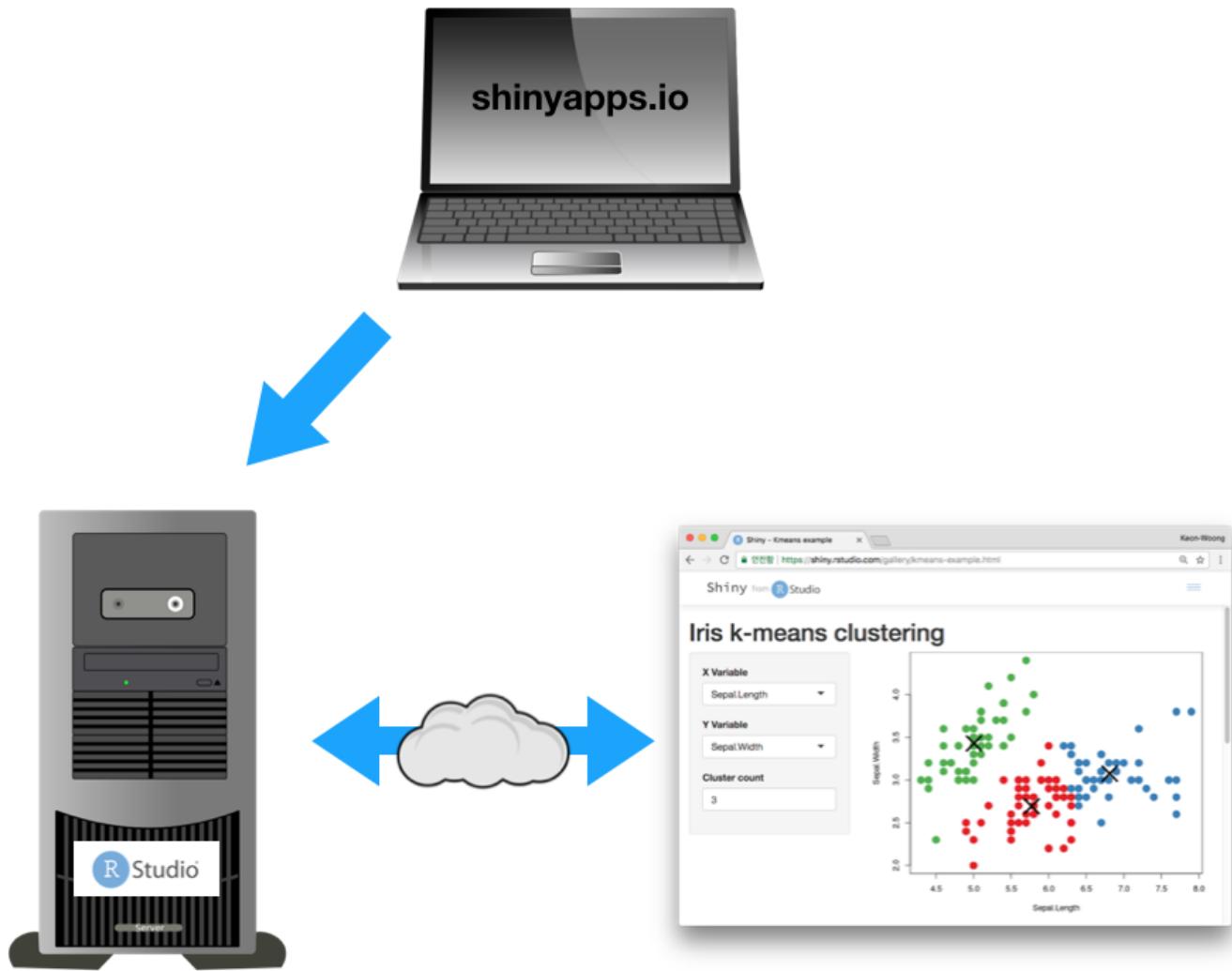
==> input의 값이 변할 때마다 reactivity 가 발생하여 output 객체를 rendering 한다

# Share Your Shiny App

- shinyapps.io
- Shiny Server

# shinyapps.io

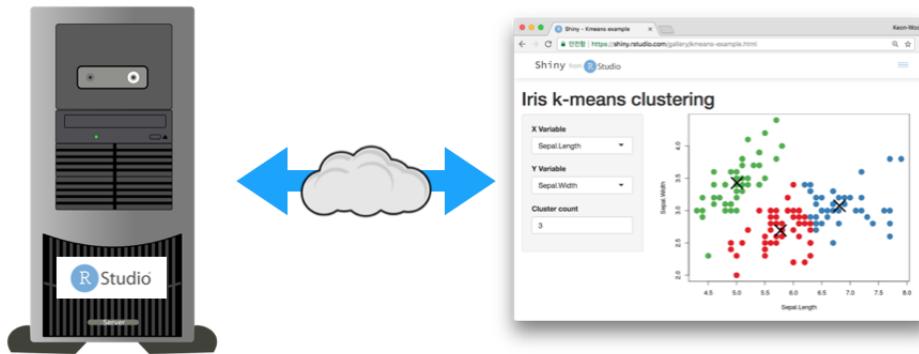




# shinyapps.io

A server maintained by RStudio

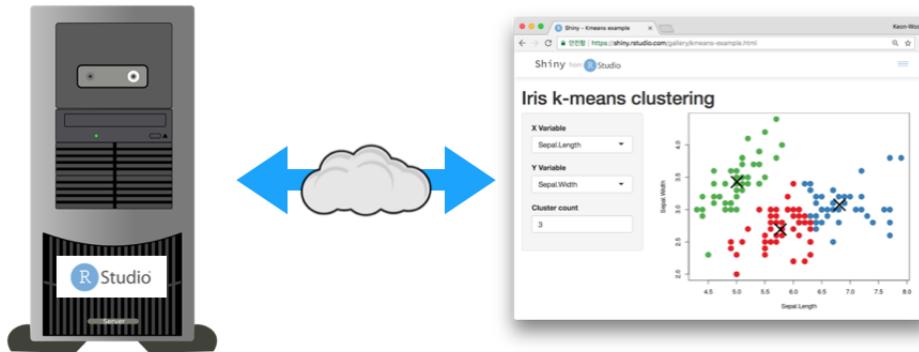
- free
- easy to use
- secure
- scalable



# shinyapps.io

A server maintained by RStudio

- free
- easy to use
- secure
- scalable



서버 세팅이 불가능 : 한글 LaTeX 환경 구축 불가

# Build Your Own Server : Shiny Server

<https://www.rstudio.com/products/shiny/shiny-server/>

The screenshot shows a web browser window for 'Shiny Server - RStudio'. The URL in the address bar is <https://www.rstudio.com/products/shiny/shiny-server/>. The page title is 'Put Shiny Web Apps Online'. The content discusses RStudio's capabilities for hosting shiny web applications and interactive documents. It mentions 'Shiny Server or Shiny Server Pro' for enterprise features like scaling, security, and admin tools. It also notes the availability of shinyapps.io plans for self-hosting. A section on RStudio Connect is mentioned. At the bottom, there are three download buttons: 'DOWNLOAD SHINY SERVER OPEN SOURCE' (white background), 'DOWNLOAD SHINY SERVER PRO' (blue background), and 'DOWNLOAD RSTUDIO CONNECT' (green background). To the right of the text, there is an image of a computer monitor displaying a shiny application with a map of the United States.

Keon-Woong

R Studio

rstudio::conf Products Resources Pricing About Us Blogs

Put Shiny Web Apps Online

RStudio lets you put shiny web applications and interactive documents online in the way that works best for you.

For Shiny applications, consider Shiny Server or Shiny Server Pro, which adds enterprise grade scaling, security, and admin features to the basic open source edition.

If you prefer for us to host your Shiny applications, one of our shinyapps.io plans is sure to work for you.

When you're ready, RStudio Connect is a new publishing platform for all the work your teams create in R. Share Shiny applications, R Markdown reports, dashboards, plots, APIs, and more in one convenient place. Use push-button publishing from the RStudio IDE, scheduled execution of reports, and flexible security policies to bring the power of data science to your entire enterprise.

DOWNLOAD SHINY SERVER OPEN SOURCE

DOWNLOAD SHINY SERVER PRO

DOWNLOAD RSTUDIO CONNECT

# Shiny Server Pro

<https://www.rstudio.com/products/shiny-server-pro/>

- Secure Access - LDAP, GoogleAuth, SSL and more
- Performance - fine tune at app and server level
- Management - monitor and control resource use
- Support - direct priority support

