

# R 기반 웹 프로그래밍 -Rshiny Package-

한국환경정책·평가연구원 (KEI)  
부연구위원 진대용

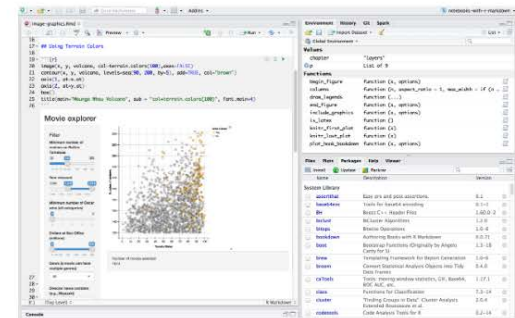
# R Shiny

- R에 기반한 웹 프로그래밍 패키지

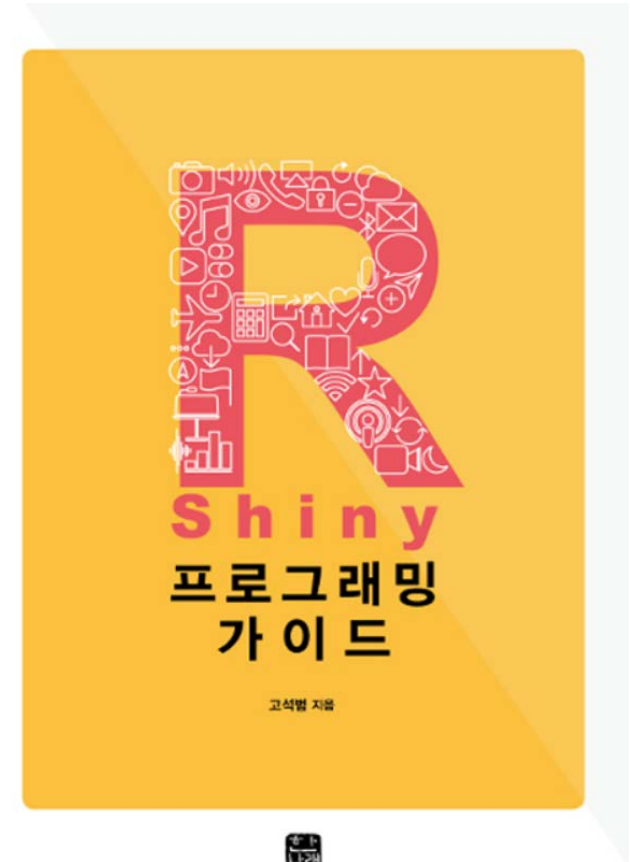
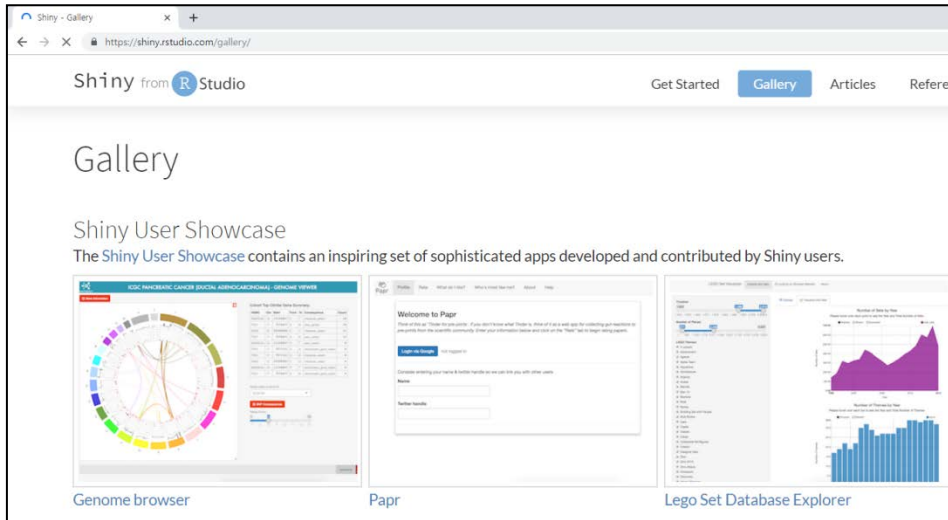
Interact. Analyze. Communicate.

Take a fresh, interactive approach to telling your data story with Shiny. Let users interact with your data and your analysis. And do it all with R.

Shiny is an R package that makes it easy to build interactive web apps straight from R. You can host standalone apps on a webpage or embed them in R Markdown documents or build dashboards. You can also extend your Shiny apps with CSS themes, htmlwidgets, and JavaScript actions.



# 참고자료



## Interactive Web Apps with shiny Cheat Sheet

learn more at [shiny.rstudio.com](https://shiny.rstudio.com)

**Studio**

**Basics**

A Shiny app is a web page (UI) connected to a computer running a live R session (Server).

Users can manipulate the UI, which will cause the server to update the UI's displays (by running R code).

**App template**

Begin writing a new app with this template. Preview the app by running the code at the R command line.

```
library(shiny)
ui <- fluidPage()
server <- function(input, output){
  shinyApp(ui = ui, server = server)
}
```

• ui - nested R functions that assemble an HTML user

**Building an App** - Complete the template by adding arguments to fluidPage() and a body to the server

Add inputs to the UI with `*input()` functions. Add outputs with `*output()` functions. Tell server how to render outputs with R in the server function. To do this:

1. Refer to outputs with `output$<id>`
2. Refer to inputs with `input$<id>`
3. Wrap code in a `render()` function before saving to output

```
library(shiny)
ui <- fluidPage(
  numericInput(inputId = "n",
    "Sample size", value = 25),
  plotOutput(outputId = "hist")
)
server <- function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$n))
  })
  shinyApp(ui = ui, server = server)
}
```

Save your template as `app.R`. Alternatively, split your template into two files named `ui.R` and `server.R`.

```
# ui.R
library(shiny)
ui <- fluidPage(
  numericInput(inputId = "n",
    "Sample size", value = 25),
  plotOutput(outputId = "hist")
)

# server.R
server <- function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$n))
  })
}

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

ui.R contains everything you would save to ui. server.R ends with the function you would save to server. No need to call shinyApp().

Save each app as a directory that contains an `app.R` file (or a `server.R` file and a `ui.R` file) plus optional extra files.

- The directory name is the name of the app (optional) defines objects available to both ui.R and server.R
- DESCRIPTION (optional) used in showcase mode
- README (optional) data, scripts, etc.
- <other files> (optional) directory of files to share with web

Launch apps with `runApp(<path to directory>)`

**Inputs - collect v**

Access the current value of `$<inputId>`. Input values at

Action `actionButton`

Link `actionLink`

☒ Choice 1 ☒ Choice 2 ☐ Choice 3

☒ Check me ☐ checkbox

dateInput max, from language

dateRange end, min, weekstar

Choose File  fileInput(n accept)

# Shiny 기본 뼈대

```
library(shiny)
ui <- fluidPage{
```

ui.R

```
  # 입출력 위젯
```

```
}
```

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

```
{
```

```
  # 서버코드
```

```
}
```

server.R

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

# 첫 웹 프로그램 : Hello World

```
ui.R* x server.R x
1 library(shiny)
2 ui <- fluidPage(
3
4   # 입력 출력 위젯
5   titlePanel('Hello world')
6 )
7
```

```
ui.R* x server.R* x
1 server <- function(input,output,session)
2 {
3   # 서버 코드
4 }
5
6 shinyApp(ui,server)
```

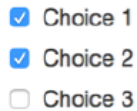
# 입력 위젯함수



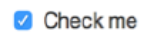
**actionButton**(inputId, label, icon, ...)

Link

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



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



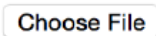
**checkboxInput**(inputId, label, value)



**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)



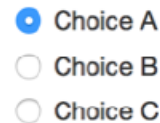
**numericInput**(inputId, label, value, min, max, step)



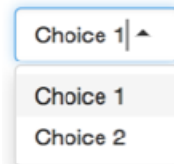
**numericInput**(inputId, label, value, min, max, step)



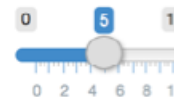
**passwordInput**(inputId, label, value)



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



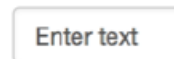
**selectInput**(inputId, label, choices, selected, multiple, selectize, width, size) (also [selectizeInput\(\)](#))



**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)

# 출력 위젯함수

**Outputs** - `render*()` and `*Output()` functions work together to add R output to the UI

`DT::renderDataTable(expr,  
options, callback, escape,  
env, quoted)`

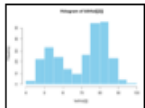


`dataTableOutput(outputId, icon, ...)`



`renderImage(expr, env, quoted, deleteFile)`

`imageOutput(outputId, width, height, click,  
dblclick, hover, hoverDelay, hoverDelayType,  
brush, clickId, hoverId, inline)`



`renderPlot(expr, width, height, res, ..., env,  
quoted, func)`

`plotOutput(outputId, width, height, click,  
dblclick, hover, hoverDelay, hoverDelayType,  
brush, clickId, hoverId, inline)`

\*Data Frame\*: 3 obs. of 2 variables:  
1 Sepal.Length: min 5.3 4.9 4.7  
1 Sepal.Width: 1 min 3.5 3 3.2

`renderPrint(expr, env, quoted, func,  
width)`

`verbatimTextOutput(outputId)`

`renderTable(expr, ..., env, quoted, func)`

`tableOutput(outputId)`

**foo**

`renderText(expr, env, quoted, func)`

`textOutput(outputId, container, inline)`



`renderUI(expr, env, quoted, func)`

`uiOutput(outputId, inline, container, ...)`  
& `htmlOutput(outputId, inline, container, ...)`

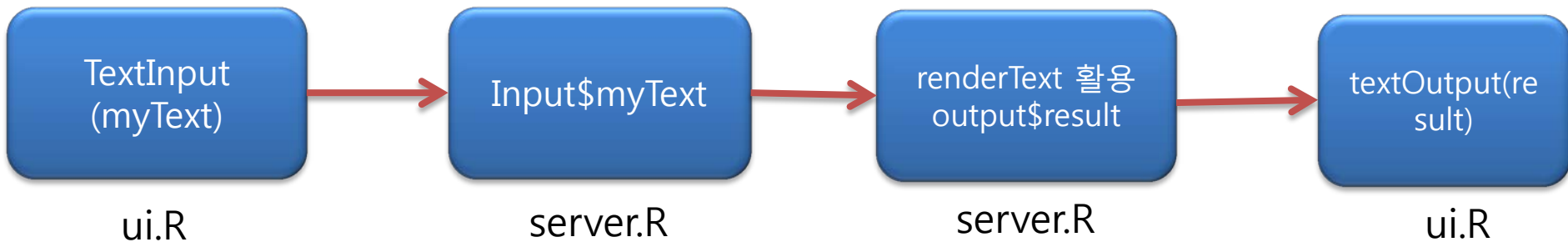
# TextInput widget

```
ui.R x server.R x
1 library(shiny)
2 ui <- fluidPage(
3   # 입출력 위젯
4   titlePanel('Hello world'),
5   textInput("myText", "텍스트를 입력하세요"),
6   textOutput("result")
7 )
```

```
ui.R x server.R x
1 server <- function(input, output, session)
2 {
3   # 서버 코드
4   output$result <- renderText({
5     input$myText
6   })
7 }
8
9 shinyApp(ui, server)
```



# TextInput widget 출력과정



# 반응성 맥락

- 반응성 맥락
  - 여러 객체(변수)들이 엮여 있는것
  - 보통의 방법으로는 객체에 접근 불가

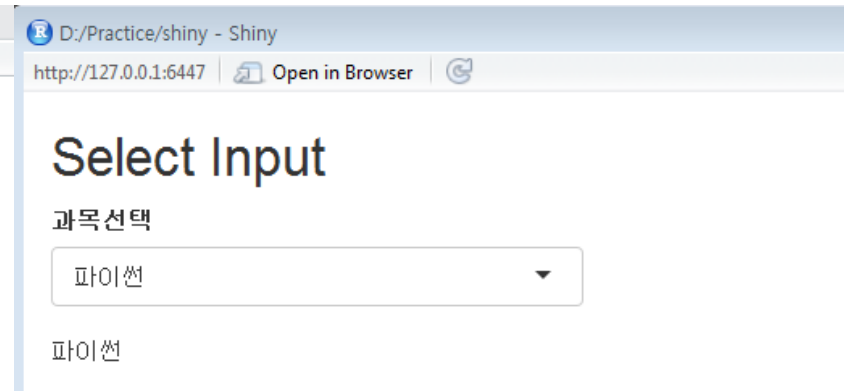
# NumericInput widget

```
server.R x ui.R x
1 setwd("D:/Practice/shiny")
2 server <- function(input,output,session)
3 {
4   output$result <- renderText({
5     input$myNum
6   })
7 }
8
9 shinyApp(ui,server)
```

```
server.R x ui.R x
1 library(shiny)
2
3 setwd("D:/Practice/shiny")
4 ui <- fluidPage(
5   # 입력 출력 위젯
6   titlePanel("Numeric Input"),
7   numericInput("myNum","숫자를 입력하세요:",10),
8   textOutput("result")
9 )
10
```

# SelectInput widget

```
1 setwd("D:/Practice/shiny")
2 server <- function(input,output,session)
3 {
4   output$result <- renderText({
5     input$sel
6   })
7 }
8
9 shinyApp(ui,server)
```



```
1 library(shiny)
2
3 setwd("D:/Practice/shiny")
4 ui <- fluidPage(
5
6   # 입력 위젯
7   titlePanel("Select Input"),
8   selectInput("sel", "과목선택",
9     choices = list('프로그래밍' = c('C++', '파이썬', 'R'),
10                   '수학' = c('미적', '통계', '벡터'))),
11   textOutput("result"))
```

# Switch 구문

- `switch(2,"red","green","blue")`
- `switch("color", "color" = "red", "shape" = "square", "length" = 5 )`

# RadioButtons widget

```
1 library(shiny)
2
3 setwd("D:/Practice/shiny")
4 ui <- fluidPage(
5
6   # 입력력 위젯
7   titlePanel("Select Input"),
8   radioButtons("dist", "Distribution type:",
9     c("Normal" = "norm",
10       "Uniform" = "unif",
11       "Log-normal" = "lnorm",
12       "Exponential" = "exp")),
13   plotOutput("result"))
```

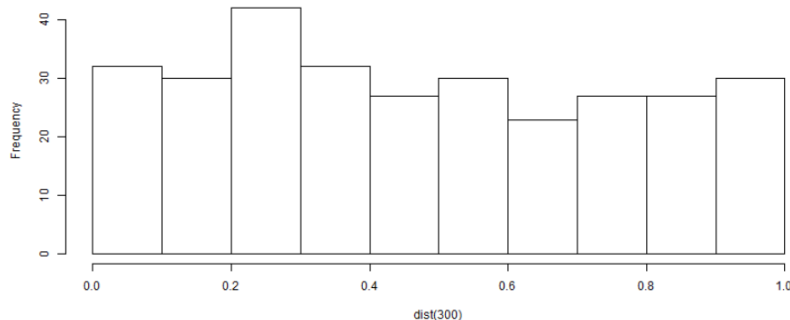
D:/Practice/shiny - Shiny  
http://127.0.0.1:7149 Open in Browser Publish

## Select Input

Distribution type:

- ☐ Normal
- ☒ Uniform
- ☐ Log-normal
- ☐ Exponential

Histogram of dist(300)

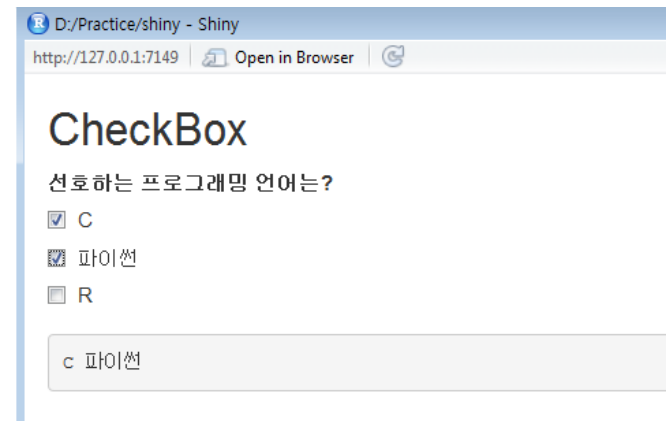


```
1 setwd("D:/Practice/shiny")
2 server <- function(input,output,session)
3 {
4   output$result <- renderPlot({
5     print(input$dist)
6     dist <- switch(input$dist,
7       norm = rnorm,
8       unif = runif,
9       lnorm = rlnorm,
10      exp = rexp,
11      rnorm)
12     hist(dist(300))
13   })
14 }
15 shinyApp(ui, server)
```

# Checkbox widget

```
server.R ui.R
1 library(shiny)
2
3 setwd("D:/Practice/shiny")
4 ui <- fluidPage(
5
6   # 입력 위젯
7   titlePanel("Select Input"),
8
9   checkboxGroupInput("subj", "선호하는 프로그래밍 언어는?",
10                      c("C", '파이썬', 'R')),
11   verbatimTextOutput("result")
12 )
```

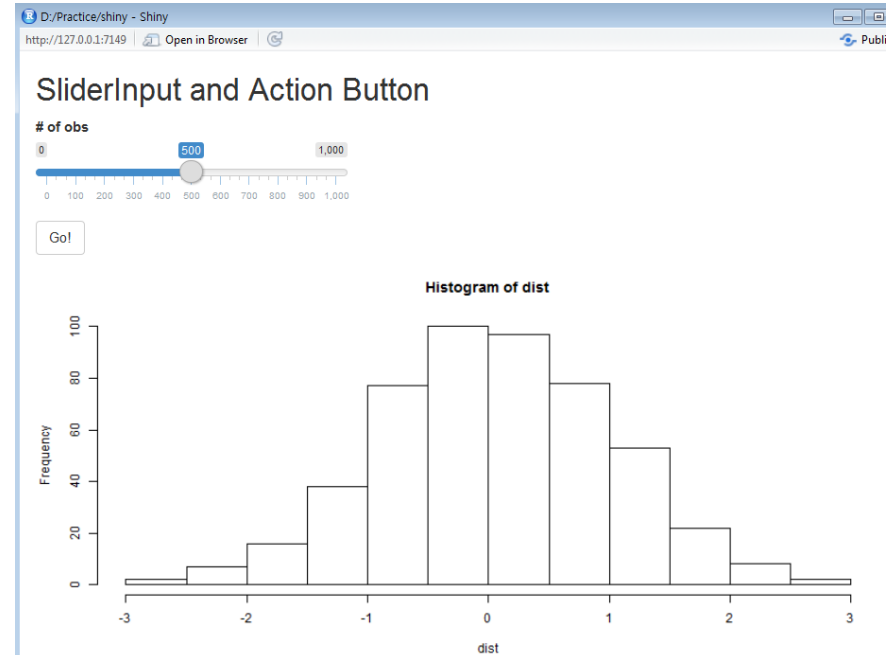
```
server.R ui.R
1 setwd("D:/Practice/shiny")
2 server <- function(input, output, session)
3 {
4   output$result <- renderText({
5     input$subj
6   })
7 }
8 shinyApp(ui, server)
```



# ActionButton Widget

```
1 library(shiny)
2
3 setwd("D:/Practice/shiny")
4 ui <- fluidPage(
5
6   # 입출력 위젯
7   titlePanel("SliderInput and Action Button"),
8
9   sliderInput('si', '# of obs', 0, 1000, 500),
10  actionButton('goBtn', 'Go!'),
11  plotOutput('result')
12 )
```

```
1 setwd("D:/Practice/shiny")
2 server <- function(input, output, session)
3 {
4   output$result <- renderPlot({
5     # goBtn이 바뀔 경우 재실행
6     input$goBtn
7     print(input$goBtn)
8
9     dist <- isolate(rnorm(input$si))
10    hist(dist)
11  })
12 }
13 shinyApp(ui, server)
```





# DataTableOutput

```
1 library(shiny)
2
3 setwd("D:/Practice/shiny")
4 ui <- fluidPage(
5
6   # 입력력 위젯
7   titlePanel("DataTable output"),
8   dataTableOutput("result")
9 )
```

```
1 setwd("D:/Practice/shiny")
2 server <- function(input,output,session)
3 {
4   output$result <- renderDataTable(
5     {result <- mtcars
6       result
7     },
8     options = list(pageLength=10))
9 }
10
11 shinyApp(ui,server)
```

http://127.0.0.1:7353 Open in Browser Publish

### DataTable Output

Show 10 entries Search:

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
21	6	160	110	3.9	2.62	16.46	0	1	4	4
21	6	160	110	3.9	2.875	17.02	0	1	4	4
22.8	4	108	93	3.85	2.32	18.61	1	1	4	1
21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
18.7	8	360	175	3.15	3.44	17.02	0	0	3	2
18.1	6	225	105	2.76	3.46	20.22	1	0	3	1
14.3	8	360	245	3.21	3.57	15.84	0	0	3	4
24.4	4	146.7	62	3.69	3.19	20	1	0	4	2
22.8	4	140.8	95	3.92	3.15	22.9	1	0	4	2
19.2	6	167.6	123	3.92	3.44	18.3	1	0	4	4

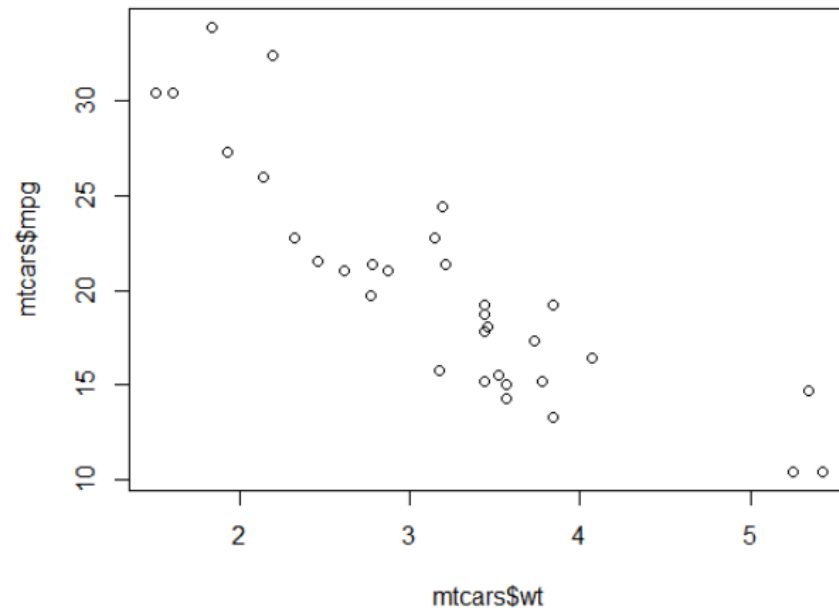
mpg cyl disp hp drat wt qsec vs am gear carb

Showing 1 to 10 of 32 entries

Previous 1 2 3 4 Next

# PlotOutput

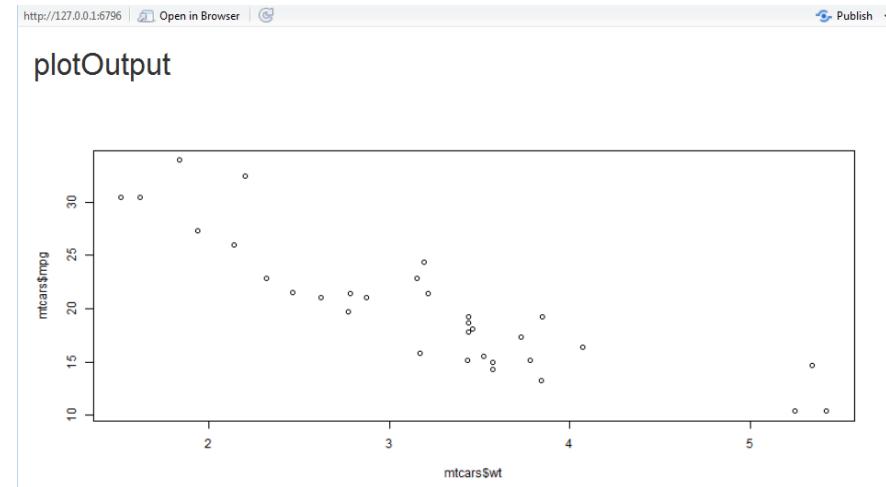
```
plot(mtcars$wt, mtcars$mpg)
```




# PlotOutput

```
server.R x ui.R x
1 setwd("D:/Practice/shiny")
2 server <- function(input,output,session)
3 {
4   output$result <- renderPlot({
5     plot(mtcars$wt, mtcars$mpg)
6   })
7 }
8
9 shinyApp(ui,server)
```

```
server.R x ui.R x
1 library(shiny)
2
3 setwd("D:/Practice/shiny")
4 ui <- fluidPage(
5
6   # 입력력 위젯
7   titlePanel("plotOutput"),
8   plotOutput("result")
9 )
```



# Widget Gallery

Shiny from  Studio

Back to Gallery Get Code

## Shiny Widgets Gallery

For each widget below, the Current Value(s) window displays the value that the widget provides to shinyServer. Notice that the values change as you interact with the widgets.

### Action button

Action

Current Value:

```
[1] 0  
attr(,"class")  
[1] "integer" "shinyActionButtonValue"
```

See Code

### Single checkbox

☒ Choice A

Current Value:

```
[1] TRUE
```

See Code

### Checkbox group

☒ Choice 1  
☐ Choice 2  
☐ Choice 3

Current Values:

```
[1] "1"
```

See Code

### Date input

2014-01-01

Current Value:

```
[1] "2014-01-01"
```

See Code

### Date range

2019-04-06 to 2019-04-06

Current Values:

```
[1] "2019-04-06" "2019-04-06"
```

See Code

### File input

Browse... No file selected

Current Value:

```
NULL
```

See Code

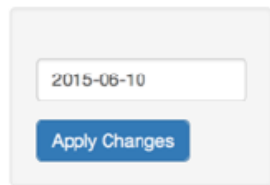
<https://shiny.rstudio.com/gallery/widget-gallery.html>

# 레이아웃 (Layout)

## Layouts

Combine multiple elements into a "single element" that has its own properties with a panel function, e.g.

```
wellPanel(  
  dateInput("a", ""),  
  submitButton()  
)
```



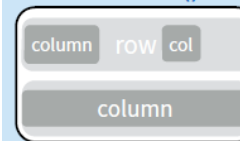
absolutePanel()  
conditionalPanel()  
fixedPanel()  
headerPanel()

inputPanel()  
mainPanel()  
navlistPanel()  
sidebarPanel()

tabpanel()  
tabsetPanel()  
titlePanel()  
wellPanel()

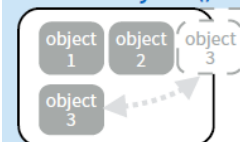
Organize panels and elements into a layout with a layout function. Add elements as arguments of the layout functions.

### fluidRow()



```
ui <- fluidPage(  
  fluidRow(column(width = 4),  
    column(width = 2, offset = 3)),  
  fluidRow(column(width = 12))  
)
```

### flowLayout()



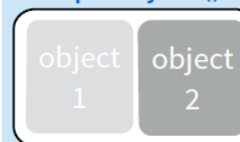
```
ui <- fluidPage(  
  flowLayout(# object 1,  
    # object 2,  
    # object 3  
  )  
)
```

### sidebarLayout()



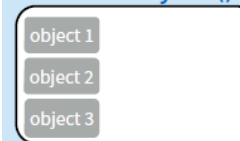
```
ui <- fluidPage(  
  sidebarLayout(  
    sidebarPanel(),  
    mainPanel()  
  )  
)
```

### splitLayout()



```
ui <- fluidPage(  
  splitLayout(# object 1,  
    # object 2  
  )  
)
```

### verticalLayout()



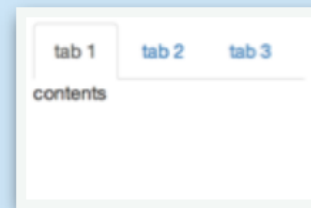
```
ui <- fluidPage(  
  verticalLayout(# object 1,  
    # object 2,  
    # object 3  
  )  
)
```

# Layer tabPanels



Layer tabPanels on top of each other,  
and navigate between them, with:

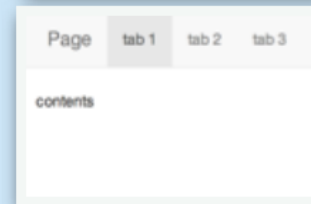
```
ui <- fluidPage( tabsetPanel(  
  tabPanel("tab 1", "contents"),  
  tabPanel("tab 2", "contents"),  
  tabPanel("tab 3", "contents")))
```



```
ui <- fluidPage( navlistPanel(  
  tabPanel("tab 1", "contents"),  
  tabPanel("tab 2", "contents"),  
  tabPanel("tab 3", "contents")))
```



```
ui <- navbarPage(title = "Page",  
  tabPanel("tab 1", "contents"),  
  tabPanel("tab 2", "contents"),  
  tabPanel("tab 3", "contents"))
```



# Reactive()

```
ui.R* x server.R* x
1 library(shiny)
2 ui <- fluidPage(
3   textInput("a", ""),
4   textInput("b", ""),
5   textOutput("z")
6 )
```

```
ui.R* x server.R* x
1 server <- function(input,output)
2 {
3   re <- reactive({
4     paste(input$a,input$b)
5   })
6
7   output$z <- renderText({
8     re()
9   })
10 }
11
12 shinyApp(ui, server)
```

# observeEvent()

```
server.R* x ui.R* x
1 library(shiny)
2 ui <- fluidPage(
3   textInput("a", "입 력 "),
4   actionButton("go", "GO!"),
5   textOutput("z")
6 )
```

```
server.R* x ui.R* x
1 server <- function(input, output){
2   observeEvent(input$go, {
3     output$z <- renderText({
4       input$a
5     })
6   })
7 }
8
9 shinyApp(ui, server)
```



# Example 확인

- `runExample()`

# 그 외 알아야 할 내용

- isolate() 함수
- observer() 함수
- eventReactive() 함수
- reactiveValues() 함수
- reactiveVal() 함수
- 테마
- ....