

## *L18. Dashboard (2)*

Sim, Min Kyu, Ph.D., [mksim@seoultech.ac.kr](mailto:mksim@seoultech.ac.kr)



서울과학기술대학교 데이터사이언스학과

*rmarkdown + shiny*

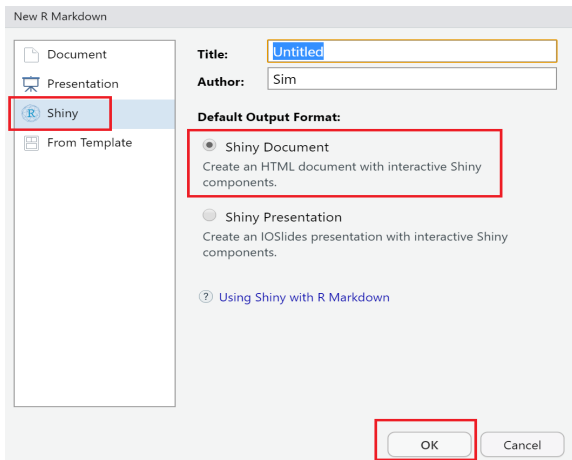
*flexdashboard + shiny*

*Input & Output*

*rmarkdown + shiny*

## Getting Started

1. `install.packages("shiny")`
2. 파일 → 새 파일 → Rmarkdown → Shiny



## 템플릿 읽어보기

# Untitled

Sim

2020 5 28

This R Markdown document is made interactive using Shiny. Unlike the more traditional workflow of creating static reports, you can now create documents that allow your readers to change the assumptions underlying your analysis and see the results immediately.

To learn more, see [Interactive Documents](#).

- R Markdown과 Shiny를 결합시켜 interactive한 문서를 만들 수 있다.
- 독자가 assumption과 parameter를 바꾸면서 결과를 즉각적으로 확인할 수 있다.
- 더 알고 싶으면  
([http://rmarkdown.rstudio.com/authoring\\_shiny.html](http://rmarkdown.rstudio.com/authoring_shiny.html))

# Inputs and Outputs

You can embed Shiny inputs and outputs in your document. Outputs are automatically updated whenever inputs change. This demonstrates how a standard R plot can be made interactive by wrapping it in the Shiny `renderPlot` function. The `selectInput` and `sliderInput` functions create the input widgets used to drive the plot.

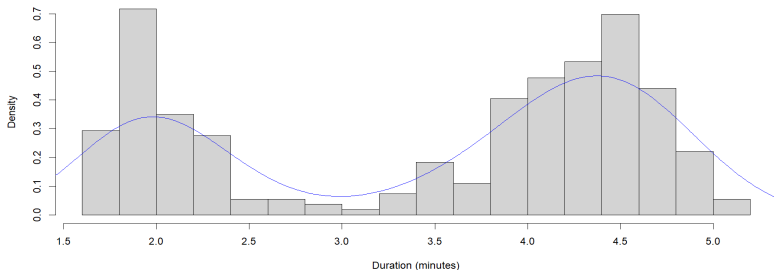
- 문서에 Shiny input과 Shiny output을 삽입할 수 있다.
- Input이 바뀔 때마다 output이 변경된다.
- 아래의 예제에서는...
- Input으로 두 개의 widget이 사용되었다: `selectInput()`, `sliderInput()`
- Output으로 하나의 interactive plot이 생성되었다: `renderPlot()`

## Inputs and Outputs

You can embed Shiny inputs and outputs in your document. Outputs are automatically updated whenever inputs change. This demonstrates how a standard R plot can be made interactive by wrapping it in the Shiny `renderPlot` function. The `selectInput` and `sliderInput` functions create the input widgets used to drive the plot.



**Geyser eruption duration**



- Input: `selectInput()`, `sliderInput()`
- Output: `renderPlot()`

```

```{r eruptions, echo=FALSE}
inputPanel(
  selectInput("n_breaks", label = "Number of bins:",
             choices = c(10, 20, 35, 50), selected = 20),

  sliderInput("bw_adjust", label = "Bandwidth adjustment:",
             min = 0.2, max = 2, value = 1, step = 0.2)
)

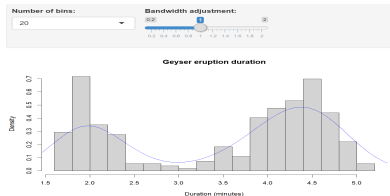
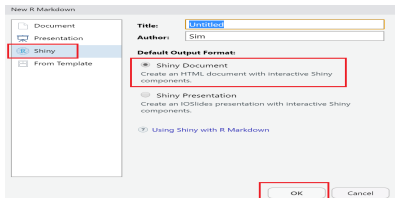
renderPlot({
  hist(faithful$eruptions, probability = TRUE,
       breaks = as.numeric(input$n_breaks),
       xlab = "Duration (minutes)",
       main = "Geyser eruption duration")|
  dens <- density(faithful$eruptions, adjust = input$bw_adjust)
  lines(dens, col = "blue")
})
```

```

1. inputPanel()과 renderXXX()로 구분되어 있음
2. selectInput()의 결과가 inputId="n\_breaks"로 저장됨
3. sliderInput()의 결과가 inputId="bw\_adjust"로 저장됨
4. renderPlot()에서는 이들 인풋을 각각 input\$n\_breaks와 input\$bw\_adjust로 사용함



# Summary



- (Sec. 2) 대시보드로 만들 방법은 없을까?
- (Sec. 3) 다양한 종류의 Input widget과 output 형식은?

```

---{r eruptions, echo=FALSE}
inputPanel(
  selectInput("n_breaks", label = "Number of bins:",
             choices = c(10, 20, 35, 50), selected = 20),

  sliderInput("bw_adjust", label = "Bandwidth adjustment:",
             min = 0.2, max = 2, value = 1, step = 0.2)
)

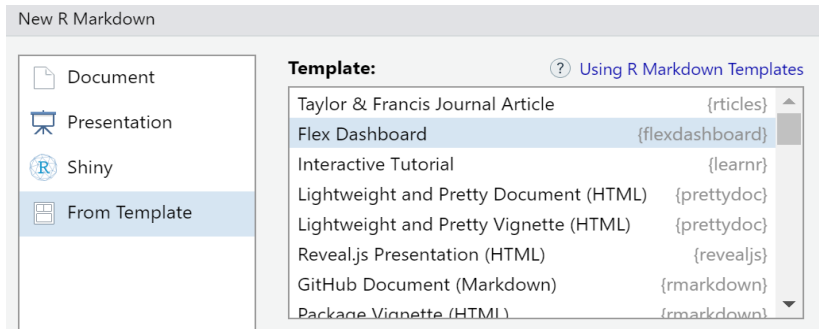
renderPlot({
  hist(faithful$eruptions, probability = TRUE,
       breaks = as.numeric(input$n_breaks),
       xlab = "Duration (minutes)",
       main = "Geyser eruption duration")
  dens <- density(faithful$eruptions, adjust = input$bw_adjust)
  lines(dens, col = "blue")
})
...

```

*flexdashboard + shiny*

# Getting Started

## 1. 파일 → 새 파일 → Rmarkdown → From Template → Flexdashboard



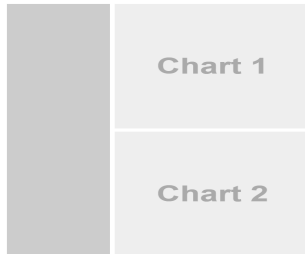
## 2. Flexdashboard의 Shiny 템플릿

- ([https://rmarkdown.rstudio.com/flexdashboard/layouts.html#input\\_sidebar](https://rmarkdown.rstudio.com/flexdashboard/layouts.html#input_sidebar))

### Input Sidebar

This layout demonstrates how to add a sidebar to a flexdashboard page (Shiny-based dashboards will often present user input controls in a sidebar). To include a sidebar you add the `.sidebar` class to a level 2 header (-----):

```
1 ---
2 title: "Sidebar"
3 output: flexdashboard::flex_dashboard
4 runtime: shiny
5 ---
6
7 Inputs (.sidebar)
8 -----
9
10 ```{r}
11 # shiny inputs defined here
12 ```
13
14 Column
15 -----
16
17 ### Chart 1
18 ```{r}
19 ```
20
21
22 ### Chart 2
23 ```{r}
24 ```
25
26
```



- YAML header에 `runtime: shiny` 추가 (들여쓰기 주의)
- 두 개의 컬럼(level-2)을 생성하여,
- 하나는 인풋으로 하나는 아웃풋으로 사용
- 인풋에 해당하는 컬럼(level-2)의 헤더에 `{.sidebar}`로 클래스 지정

### 3. Header 수정, 인풋 컬럼, 아웃풋 컬럼 작성

```
---
title: "fd + shiny"
output:
  flexdashboard::flex_dashboard:
    orientation: columns
    vertical_layout: fill
runtime: shiny
---
```

```
```{r setup, include=FALSE}
library(flexdashboard)
library(shiny)
```
```

Inputs {.sidebar}

---

```
```{r}
selectInput("myVar", label = "Choose a number",
  choices = c(10,20,30), selected = 20)
```
```

Column

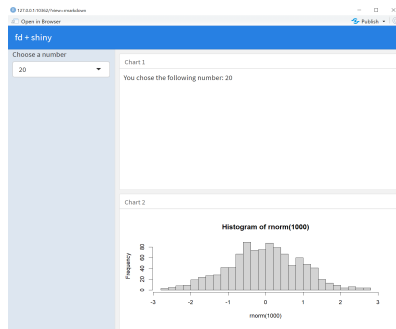
---

### Chart 1

```
```{r}
renderText({
  paste("You chose the following number:", input$myVar)
})
```
```

### Chart 2

```
```{r}
renderPlot({
  hist(rnorm(1000), as.numeric(input$myVar))
})
```
```



#### 4. 작성 프로세스와 팁

- 종이에 직접 개형을 design하고 계획을 가지고 구현 시작
- Sec. 3를 참고하여 Input widget을 계획
- frame만 설계된 상태에서 실행하여 compile확인
- `renderText()`등으로 인풋이 잘 입력되고 있는지 확인
- 자동 완성 기능 활용
  - 코드 작성전에 `library(shiny)`를 실행하여 자동완성 기능을 활성화
  - Tab 키를 눌러가면서 코드 완성
  - Widget 별로 존재하는 상세한 문법을 굳이 알 필요없음
  - 아래 그림 참조
- shiny application을 render하는 컴퓨터는 반드시 인터넷에 연결되어 있어야 함!

Inputs {.sidebar}

```

{r}
selectInput()
selectInput

```

Column

☒ `inputId =`  
☐ `label =`  
☐ `choices =`  
☐ `selected =`  
☐ `multiple =`  
☐ `selectize =`

**inputId**

The input slot that will be used to access the value.

Press F1 for additional help

## *Input & Output*

# Input

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

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

Current Value:

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

"shinyActionButtonValue"

### Single checkbox

☒ Choice A

Current Value:

### Checkbox group

☒ Choice 1  
☐ Choice 2  
☐ Choice 3

Current Values:

### Date input

Current Value:

### Date range

to

Current Values:

### File input

No file selected

Current Value:

- ‘Current Value’ 창에 widget이 입력받는 값이 보임



## Numeric input

Current Value:

[See Code](#)

## Radio buttons

- ☒ Choice 1  
☐ Choice 2  
☐ Choice 3

Current Values:

[See Code](#)

## Select box

Current Value:

[See Code](#)

## Slider



Current Value:

[See Code](#)

## Slider range



Current Values:

[See Code](#)

## Text input

Current Value:

[See Code](#)

- Widget을 선택하여 ‘See Code’ 버튼을 누르고 ‘show below’ 버튼을 누르면 상세한 설명이 보임

server.R ui.R show below

```
function(input, output) {  
  # You can access the value of the widget with input$num, e.g.  
  output$value <- renderPrint(input$num)  
}
```

server.R ui.R show below

```
fluidPage(  
  # Copy the line below to make a number input box into the UI.  
  numericInput("num", label = h3("Numeric input"), value = 1),  
  hr(),  
  fluidRow(column(3, verbatimTextOutput("value")))  
)
```

## Numeric input

[1] 1

## Number Selector

by RStudio, Inc.

```
numericInput(inputId, label, value, min = NA, max = NA, step = NA)
```

Creates a box you can use to enter numeric values. You can type a number or scroll through values with the box's scroll bar. The widget will pass the value shown in the box to the server as a double (e.g. number).

### Arguments

**inputId** The name to use to look up the value of the widget (as a character string)  
**label** A label to display above the number field  
**value** The initial number to display in the number field  
**min** The minimum number that can be selected  
**max** The maximum number that can be selected  
**step** The amount to increment the value by when a user clicks up or down on the scroll bar.

Make this widget by copying the code in ui.R.

## Discussion

### Control widget 비교

| 분류         | 이름              | 함수                   | 목적       | 입력값 구조                       | Note                     |
|------------|-----------------|----------------------|----------|------------------------------|--------------------------|
| Yes/No     | Single checkbox | checkboxInput()      | 단일 체크박스  | logical                      |                          |
|            | Checkbox group  | checkboxInputGroup() | 그룹 체크박스  | vector of characters         | 복수 입력 가능                 |
| Number     | Numeric input   | numericInput()       | 숫자 입력    | numeric                      | 키보드 입력                   |
|            | Slider          | sliderInput()        | 숫자 입력    | numeric                      | 마우스 입력                   |
|            | Slider range    | sliderInput()        | 숫자 범위 입력 | vector of numeric (length 2) | value argument로 복수 입력 설정 |
| Choice     | Select box      | selectInput()        | 드롭다운 선택  | character                    | 선택 후보가 많을때               |
|            | Radio buttons   | radioButtons()       | 라디오 버튼   | character                    | 선택 후보가 적을때               |
| Date       | Date input      | dateInput()          | 날짜 선택    | Date                         |                          |
|            | Date range      | dateRangeInput()     | 기간 선택    | vector of Date (length 2)    |                          |
| Text       | Text input      | textInput()          | 문자 입력    | character                    | 검색 키워드 등을 입력             |
| Controller | File input      | fileInput()          | 파일 입력    | character                    |                          |
|            | Action button   | actionButton()       | 누르는 액션   | logical                      | 다운로드 등의 기능 구현 가능         |

- Checkbox group과 Radio buttons의 차이는?
- Controller는 본 강의노트 이상의 지식이 필요
- Widget의 특성과 shiny 제작 목적을 고려해서 기획

## More about shiny

- ui.R에서 Input을 만들고 server.R에서 Output을 만드는 구조로 되어있음
- Web Programming 기반의 package이기 때문에 R 언어와는 형태가 많이 다른 함수들(ex. fluidRow(), verbatimTextOutput())을 잘 다루어야 함
- 대시보드 이외의 많은 application을 만들 수 있음
- R을 이용한 데이터 분석에서 가장 고급 단계로 여겨짐

## My take

- shiny를 쓰는 경우라면 대부분 dashboard를 이용한 monitoring이나 presentation이 목적이기 때문에,
- flexdashboard를 이용하여 frame을 잡고 input과 output만 지정해 주어도 대부분의 목적이 달성된다.
- shiny의 문법은 웹프로그래밍에 대한 이해도 필요하고 난해하지만, 그에 비해서 rmarkdown에 익숙하다면 flexdashboard를 이용해서 쉽게 구현이 가능하다.
- 소스파일이 하나라는 점만 해도 큰 장점이다.

## Output

- `render_YOUR_OBJECT()` 함수를 이용해서 결과물을 출력
- 기본적인 함수들
  - `renderText()`: character 출력
  - `renderTable()`: `data.frame()` 등을 출력
  - `renderPlot()`: `ggplot` 객체 등을 출력
- 패키지들에서 제공하는 함수들
  - `renderDygraph()`: `dygraph` 객체 (시계열 그래프)
  - `renderDataTable()`: `DataTable` 객체 (`data.frame`이나 `tibble`과 비슷함)
  - `renderWordCloud()`: `wordcloud` 객체
  - ...

"Non scholae sed vitae discimus."

"우리는 학교를 위해서가 아니라 인생을 위해서 배운다."