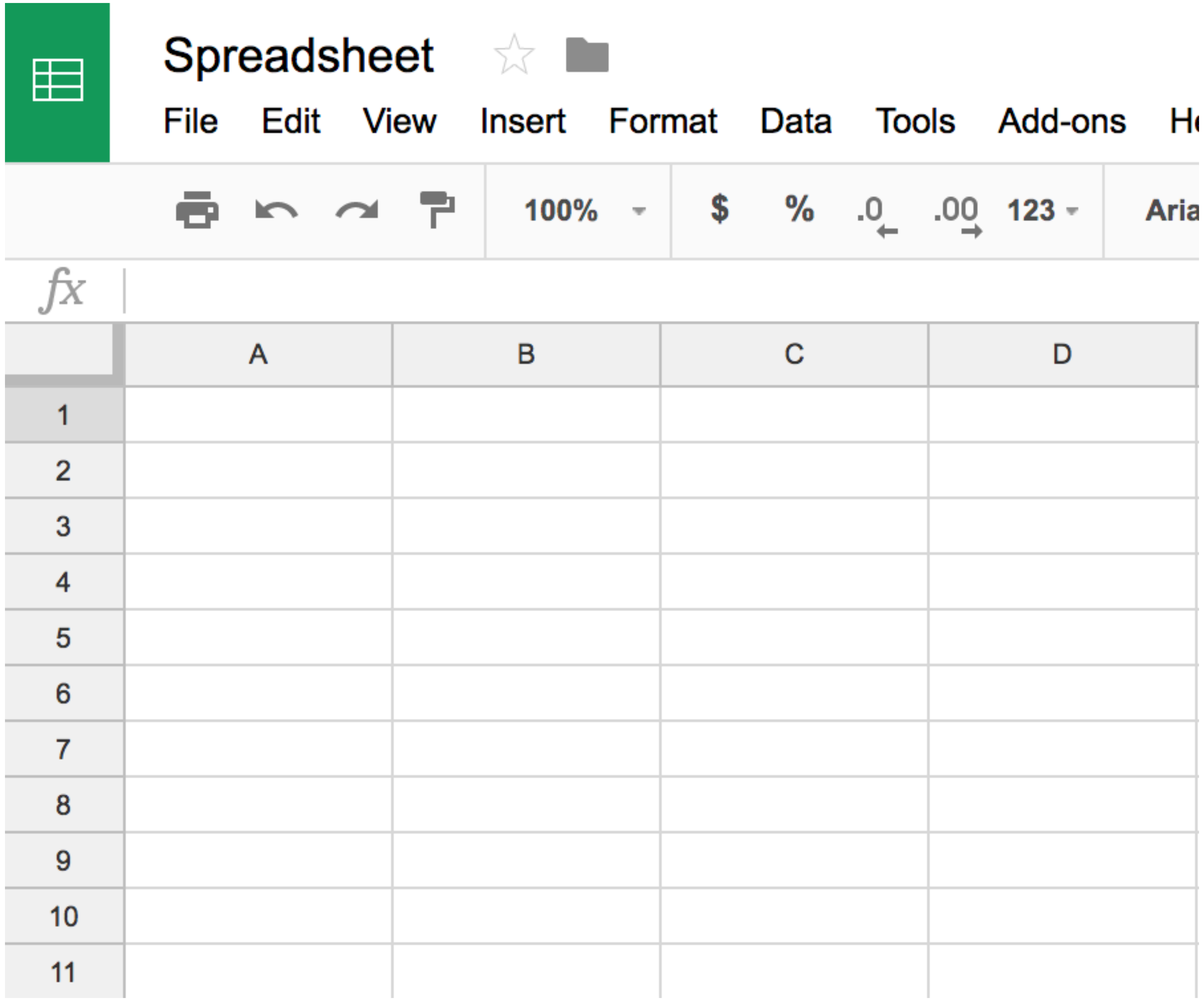
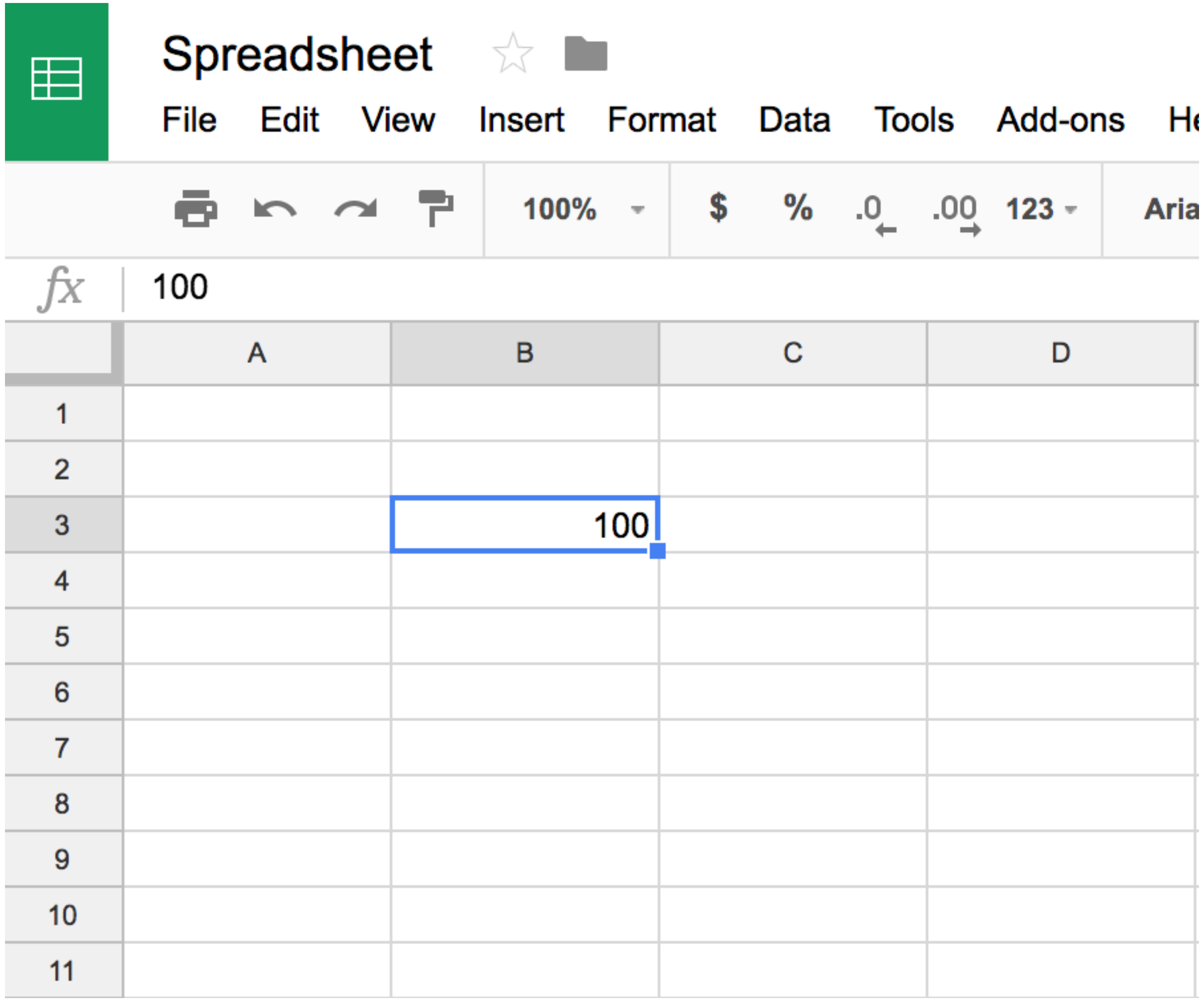
BUILDING WEB APPLICATIONS IN R WITH SHINY

Reactive flow

Reactivity, in spreadsheets



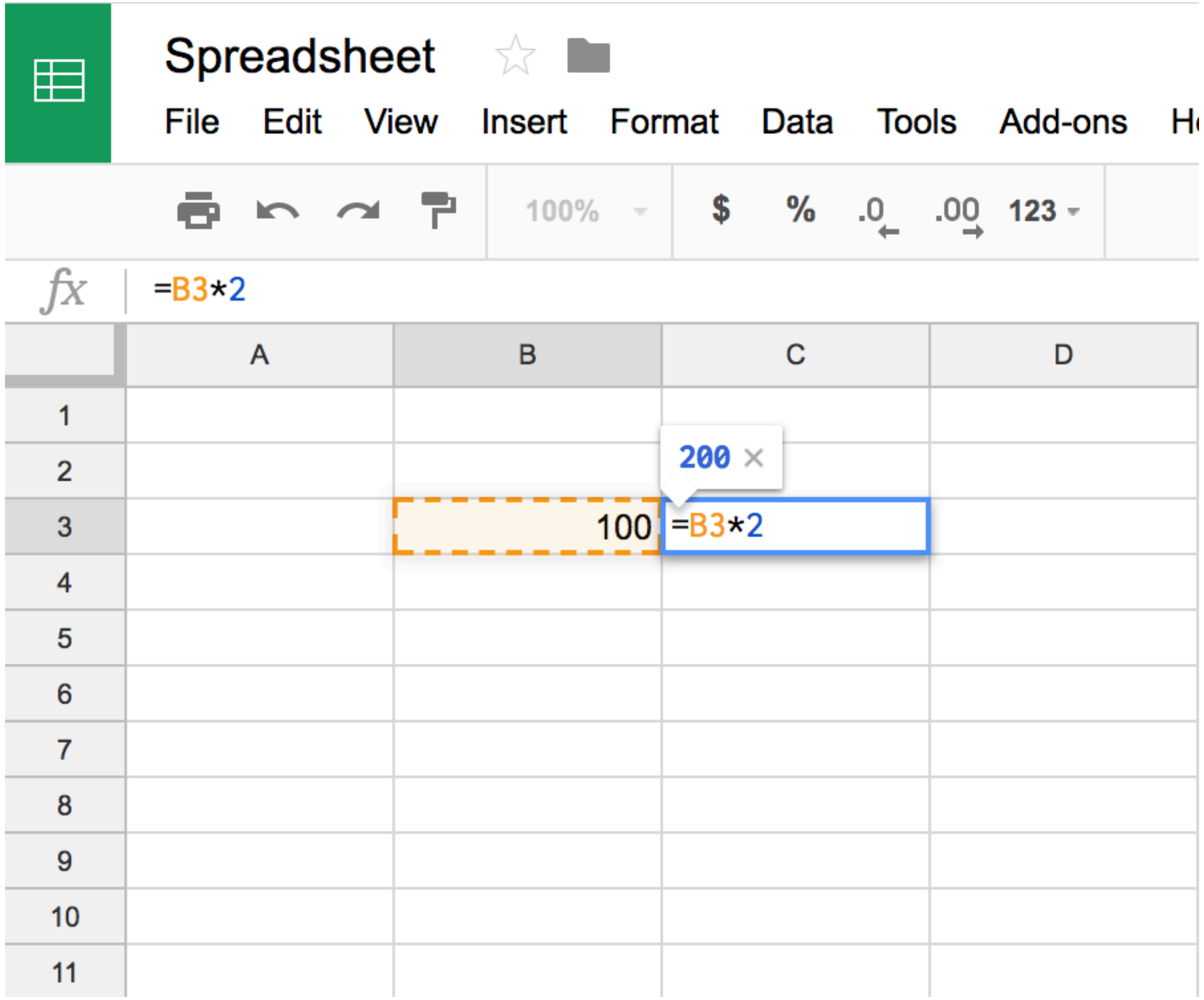
Reactivity, in spreadsheets



fx 100

	A	B	C	D
1				
2				
3		100		
4				
5				
6				
7				
8				
9				
10				
11				

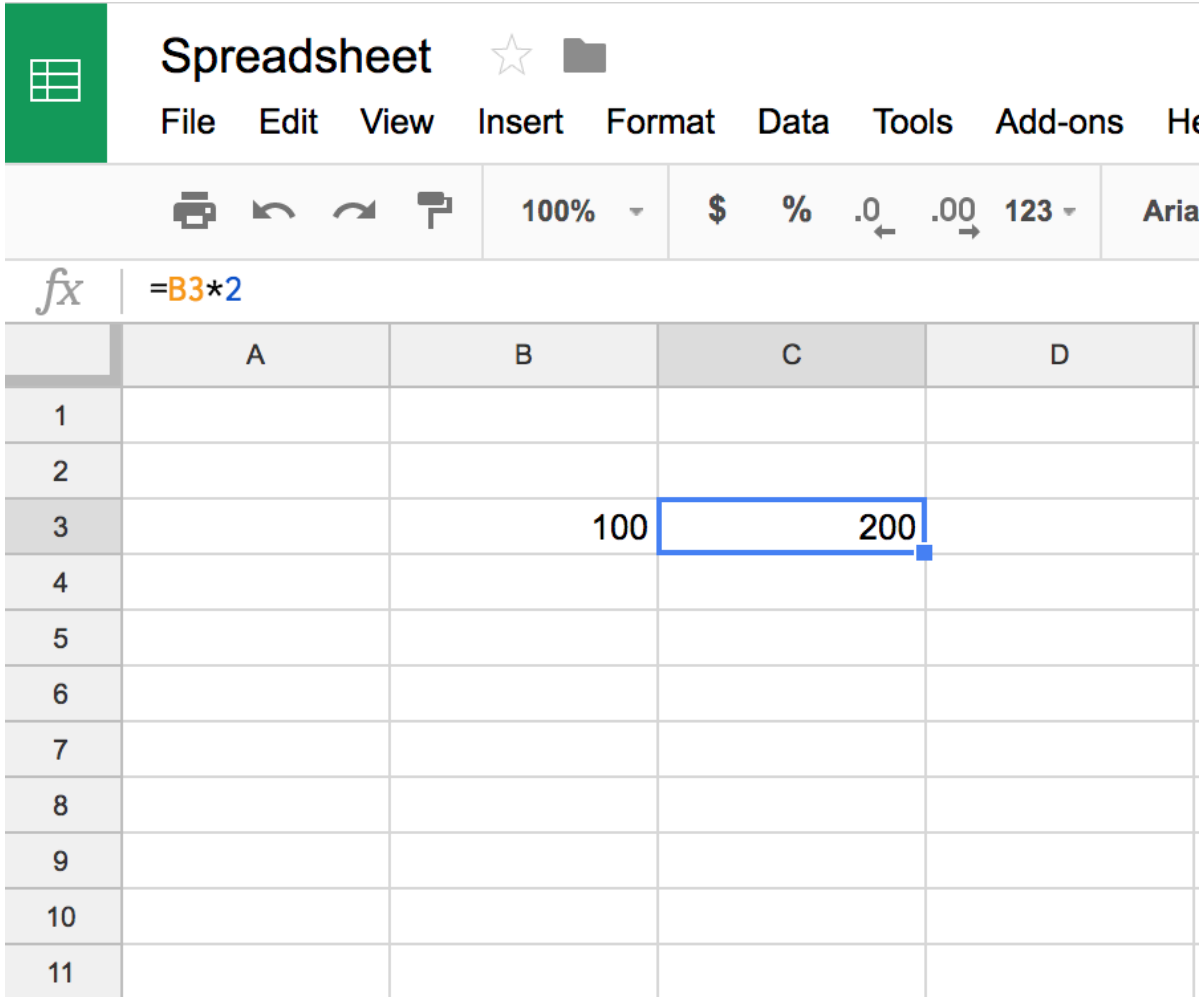
Reactivity, in spreadsheets



=B3*2

	A	B	C	D
1				
2				
3		100	=B3*2	
4				
5				
6				
7				
8				
9				
10				
11				

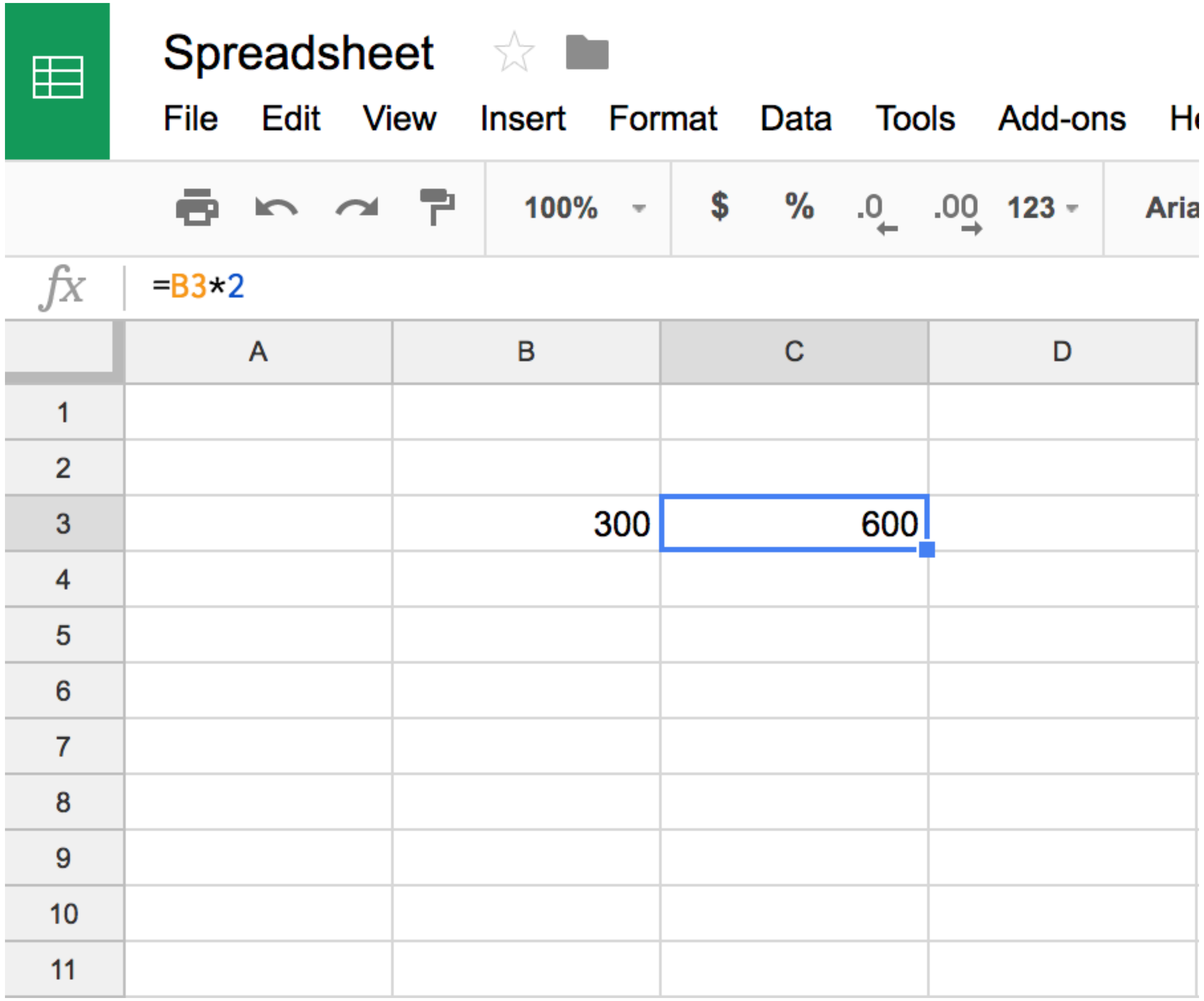
Reactivity, in spreadsheets



=B3*2

	A	B	C	D
1				
2				
3		100	200	
4				
5				
6				
7				
8				
9				
10				
11				

Reactivity, in spreadsheets



=B3*2

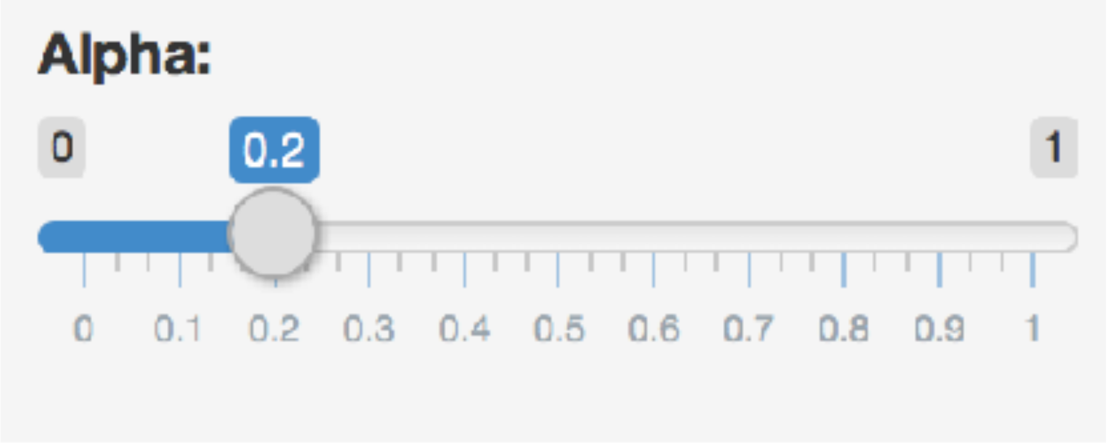
	A	B	C	D
1				
2				
3		300	600	
4				
5				
6				
7				
8				
9				
10				
11				

Reactions

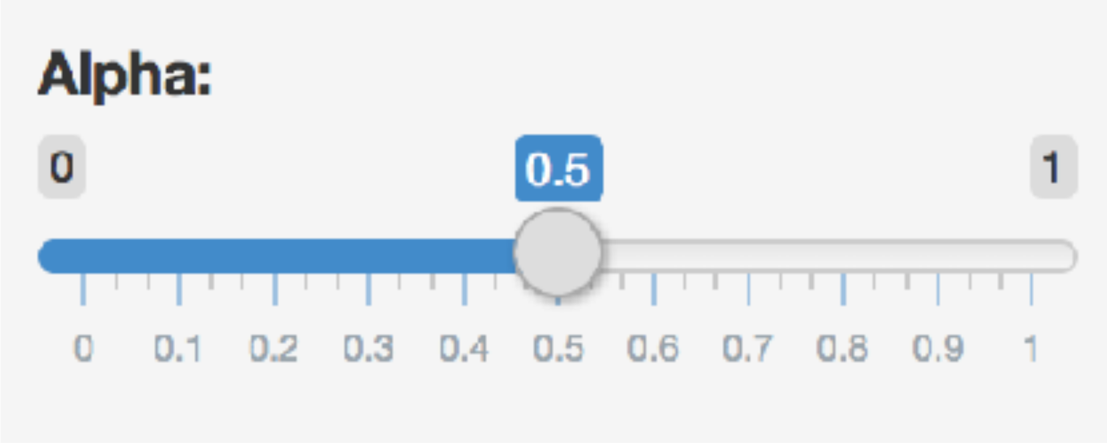
The **input$** list stores the current value of each input object under its name.

```
# Set alpha level
sliderInput(inputId = "alpha",
            label = "Alpha:",
            min = 0, max = 1,
            value = 0.5)
```

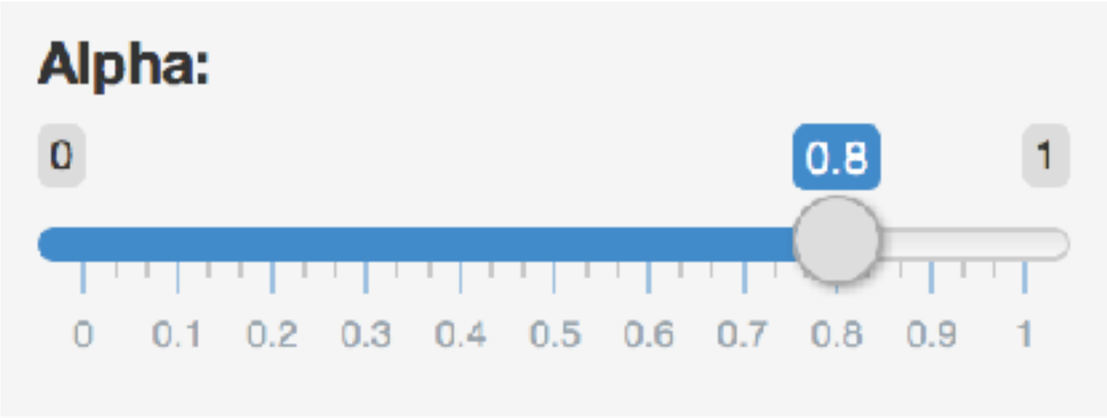
input$alpha



input$alpha = 0.2



input$alpha = 0.5



input$alpha = 0.8

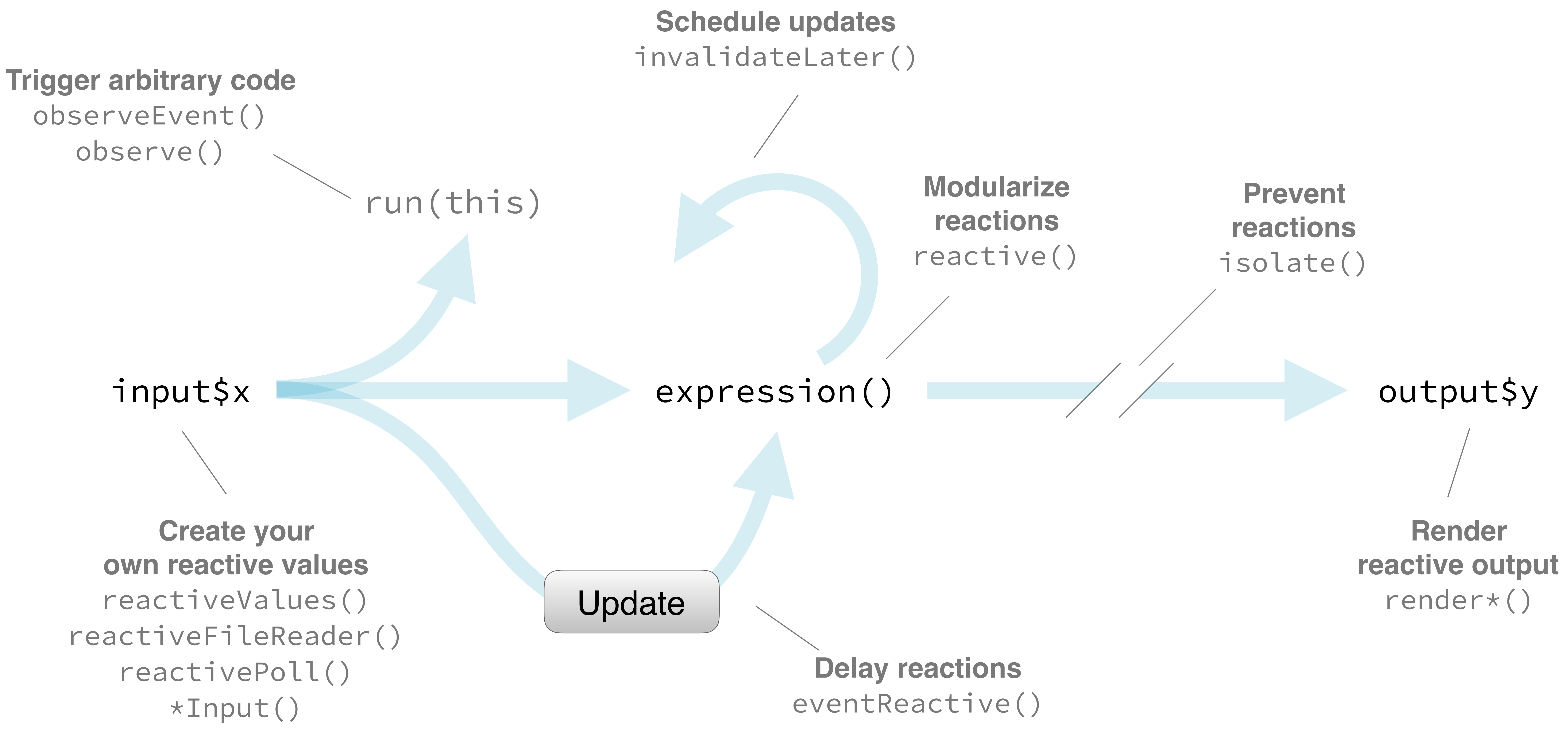
Reactivity 101

Reactivity automatically occurs when an input value is used to render an output object.

```
# Define server function required to create the scatterplot
server <- function(input, output) {

  # Create the scatterplot object the plotOutput function is expecting
  output$scatterplot <- renderPlot({
    ggplot(data = movies, aes_string(x = input$x, y = input$y)) +
      geom_point(alpha = input$alpha)
  })
}
```

Reactive flow



Trigger arbitrary code
`observeEvent()`
`observe()`

`run(this)`

Schedule updates
`invalidateLater()`

Modularize reactions
`reactive()`

Prevent reactions
`isolate()`

`input$x`

`expression()`

`output$y`

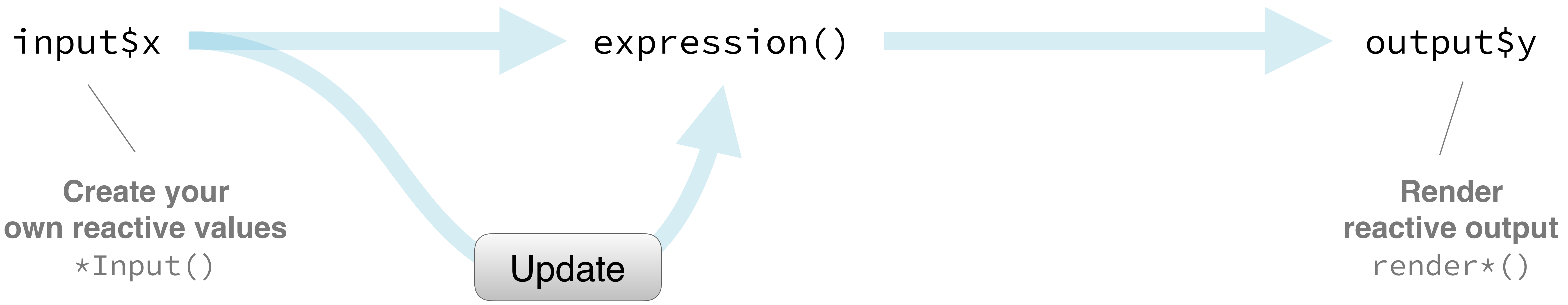
Update

Create your own reactive values
`reactiveValues()`
`reactiveFileReader()`
`reactivePoll()`
`*Input()`

Delay reactions
`eventReactive()`

Render reactive output
`render*()`

Reactive flow, simplified



input$x

expression()

output$y

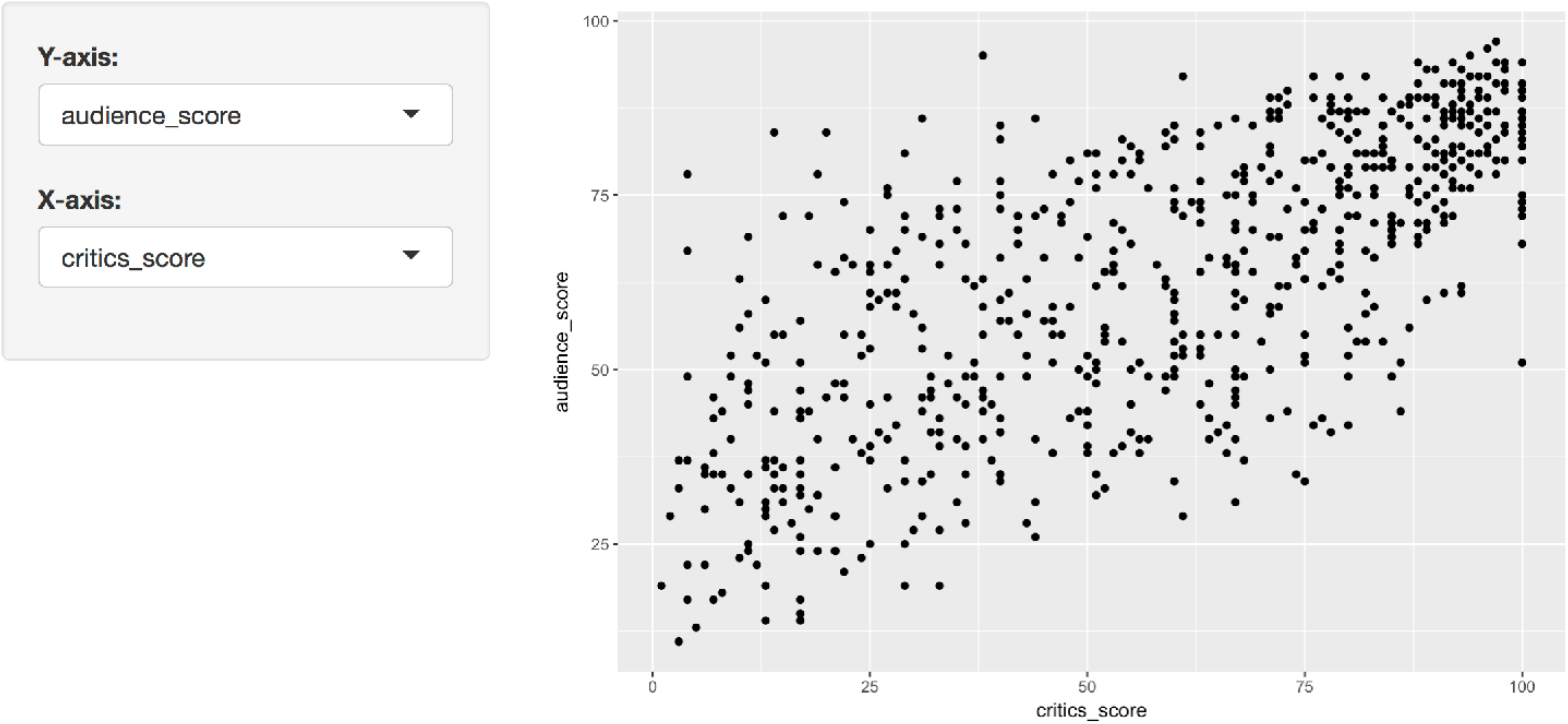
Create your own reactive values
*Input()

Render reactive output
render*()

Let's practice!

BUILDING WEB APPLICATIONS IN R WITH SHINY

UI inputs



Y-axis:

audience_score

X-axis:

critics_score

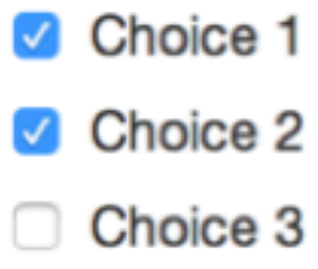
Inputs

collect values from the user

Access the current value of an input object with **input$<inputId>**. Input values are **reactive**.

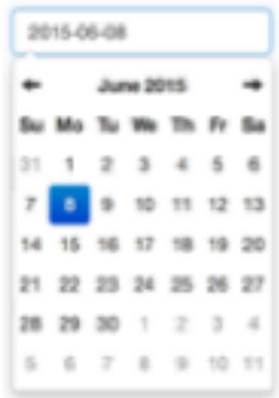
actionButton(inputId, label, icon, …)

actionLink(inputId, label, icon, …)

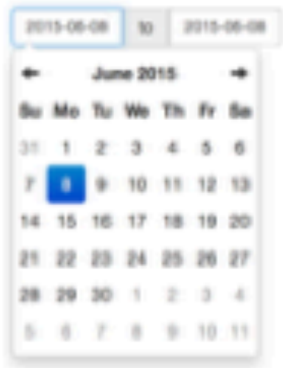


checkboxGroupInput(inputId, label, choices, selected, inline)

checkboxInput(inputId, label, value)



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

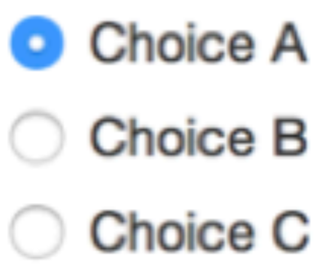


dateRangeInput(inputId, label, start, end, min, max, format, startview, weekstart, language, separator)

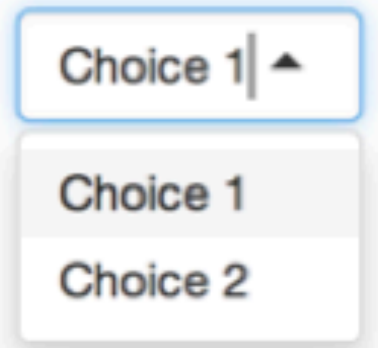
fileInput(inputId, label, multiple, accept)

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

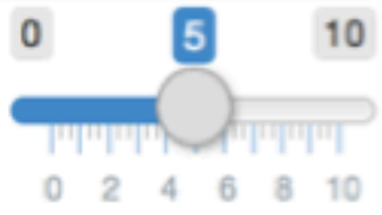
passwordInput(inputId, label, value)



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

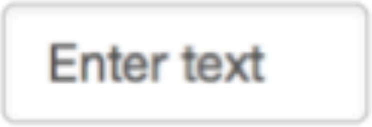


selectInput(inputId, label, choices, selected, multiple, selectize, width, size) (also selectizeInput())



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)

checkboxInput

Add a checkbox input to specify whether the data plotted should be shown in a data table.

1. **ui:** Add an input widget that the user can interact with to check/uncheck the box.

2. **ui:** Add an output defining where the data table should appear.

3. **server:** Add a reactive expression that creates the data table *if* the checkbox is checked.

checkboxInput

Add a checkbox input to specify whether the data plotted should be shown in a data table.

1. **ui:** Add an input widget that the user can interact with to check/uncheck the box.

```
# Show data table
checkboxInput(inputId = "show_data",
              label = "Show data table",
              value = TRUE)
```

Watch for commas!

```
sidebarPanel(
      # Select variable for y-axis
      selectInput(inputId = "y", label = "Y-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score",
"audience_score", "runtime"),
                  selected = "audience_score"),
      # Select variable for x-axis
      selectInput(inputId = "x", label = "X-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score",
"audience_score", "runtime"),
                  selected = "critics_score"),
      # Show data table
      checkboxInput(inputId = "show_data",
                    label = "Show data table",
                    value = TRUE)
    )
```

checkboxInput

Add a checkbox input to specify whether the data plotted should be shown in a data table.

2. **ui:** Add an output to the UI defining where the data table should appear.

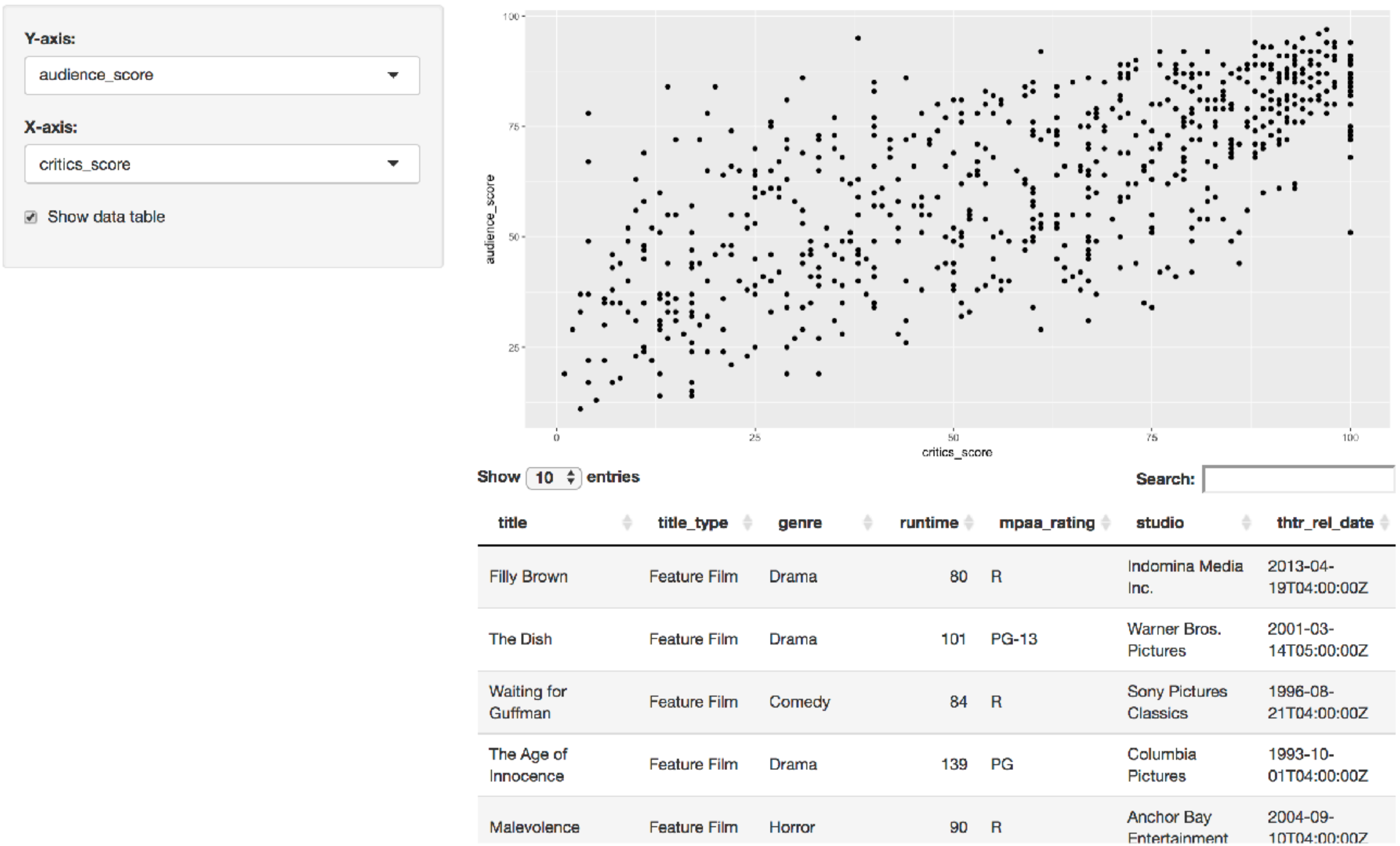
```
mainPanel(
  # Show scatterplot
  plotOutput(outputId = "scatterplot"),
  # Show data table
  DT::dataTableOutput(outputId = "moviestable")
)
```

checkboxInput

Add a checkbox input to specify whether the data plotted should be shown in a data table.

3. **server:** Add a reactive expression that creates the data table *if* the checkbox is checked.

```
# Print data table if checked
output$moviestable <- DT::renderDataTable({
  if(input$show_data){
    DT::datatable(data = movies %>% select(1:7),
    options = list(pageLength = 10),
    rownames = FALSE)
  }
})
```



Y-axis:

audience_score ▾

X-axis:

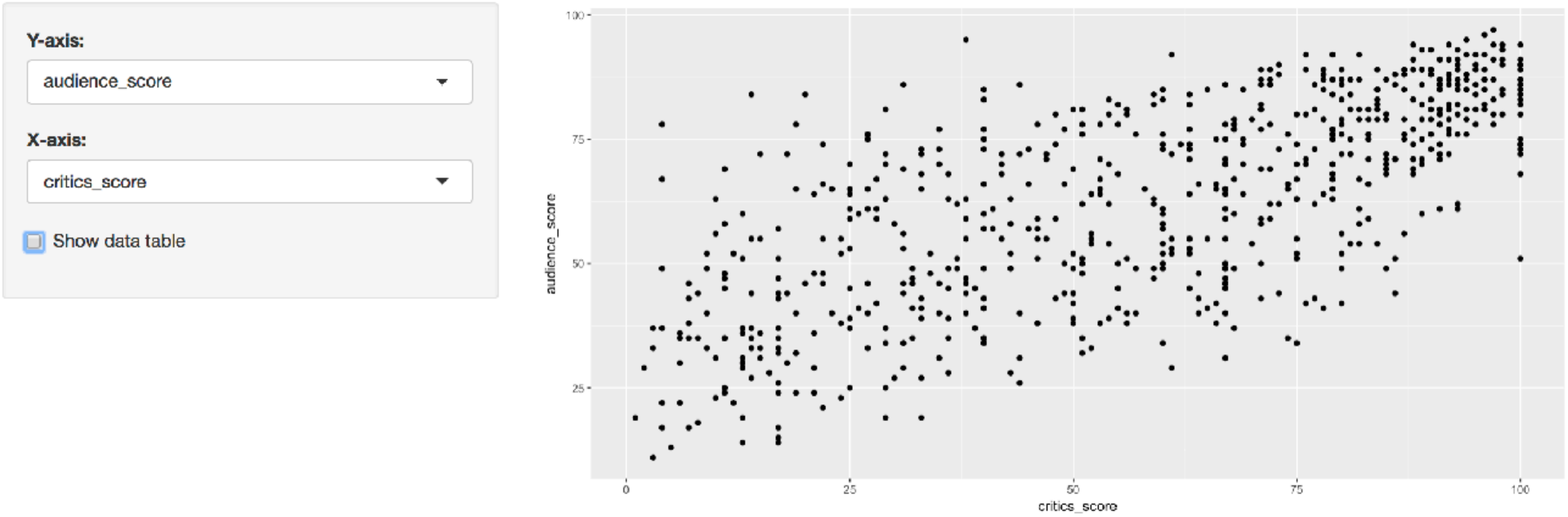
critics_score ▾

☑ Show data table

Show 10 entries

Search:

title	title_type	genre	runtime	mpaa_rating	studio	thtr_rel_date
Filly Brown	Feature Film	Drama	80	R	Indomina Media Inc.	2013-04-19T04:00:00Z
The Dish	Feature Film	Drama	101	PG-13	Warner Bros. Pictures	2001-03-14T05:00:00Z
Waiting for Guffman	Feature Film	Comedy	84	R	Sony Pictures Classics	1996-08-21T04:00:00Z
The Age of Innocence	Feature Film	Drama	139	PG	Columbia Pictures	1993-10-01T04:00:00Z
Malevolence	Feature Film	Horror	90	R	Anchor Bay Entertainment	2004-09-10T04:00:00Z



Y-axis:

audience_score

X-axis:

critics_score

□ Show data table

Scoping

- We saw that the data loaded on top of the Shiny app is visible to the server.
- It is also visible to the UI.

```
# Display number of observations
HTML(paste0("The dataset has ", nrow(movies),
            “observations."))
```

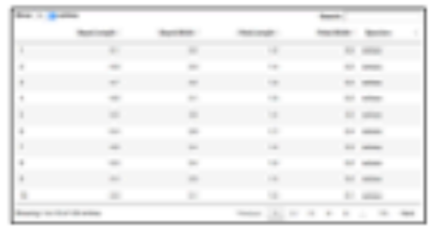
Let's practice!

BUILDING WEB APPLICATIONS IN R WITH SHINY

Rendering functions

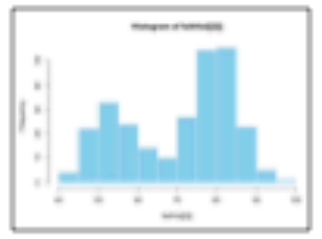
Outputs - render*() and *Output() functions work together to add R output to the UI

works with



DT::**renderDataTable**(expr, options, callback, escape, env, quoted) — **dataTableOutput**(outputId, icon, ...)

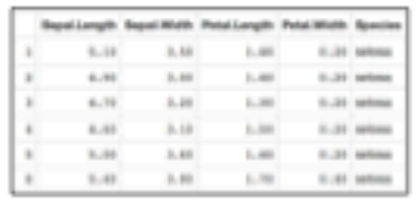
renderImage(expr, env, quoted, deleteFile) — **imageOutput**(outputId, width, height, click, dblclick, hover, hoverDelay, inline, hoverDelayType, brush, clickId, hoverId)



renderPlot(expr, width, height, res, ..., env, quoted, func) — **plotOutput**(outputId, width, height, click, dblclick, hover, hoverDelay, inline, hoverDelayType, brush, clickId, hoverId)

```
'data.frame':   3 obs. of  2 variables:
 $ Sepal.Length: num  5.1 4.9 4.7
 $ Sepal.Width : num  3.5 3 3.2
```

renderPrint(expr, env, quoted, func, width) — **verbatimTextOutput**(outputId)



	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.10	3.50	1.40	0.20	setosa
2	4.90	3.00	1.40	0.20	setosa
3	4.70	3.20	1.30	0.20	setosa
4	4.60	3.10	1.50	0.20	setosa
5	5.00	3.60	1.40	0.20	setosa
6	5.40	3.90	1.70	0.40	setosa

renderTable(expr,..., env, quoted, func) — **tableOutput**(outputId)

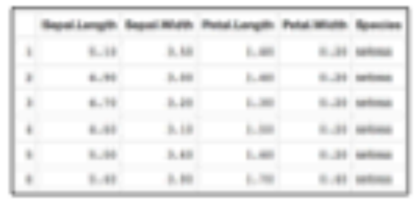
foo

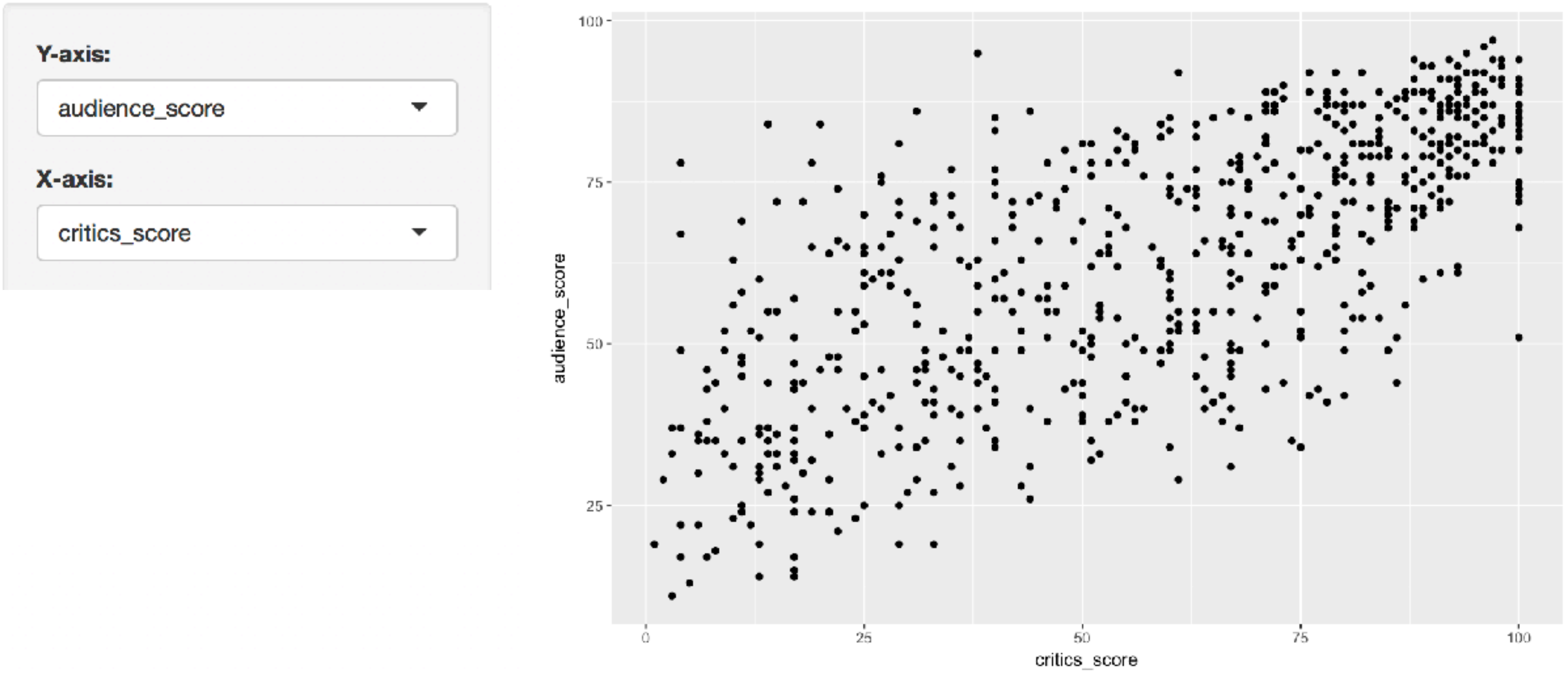
renderText(expr, env, quoted, func) — **textOutput**(outputId, container, inline)

renderUI(expr, env, quoted, func) — **uiOutput**(outputId, inline, container, ...) & **htmlOutput**(outputId, inline, container, ...)

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

works with

	DT::**renderDataTable**(expr, options, callback, escape, env, quoted)	**dataTableOutput**(outputId, icon, …)
	renderImage(expr, env, quoted, deleteFile)	**imageOutput**(outputId, width, height, click, dblclick, hover, hoverDelay, inline, hoverDelayType, brush, clickId, hoverId)
	renderPlot(expr, width, height, res, …, env, quoted, func)	**plotOutput**(outputId, width, height, click, dblclick, hover, hoverDelay, inline, hoverDelayType, brush, clickId, hoverId)
	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, …)

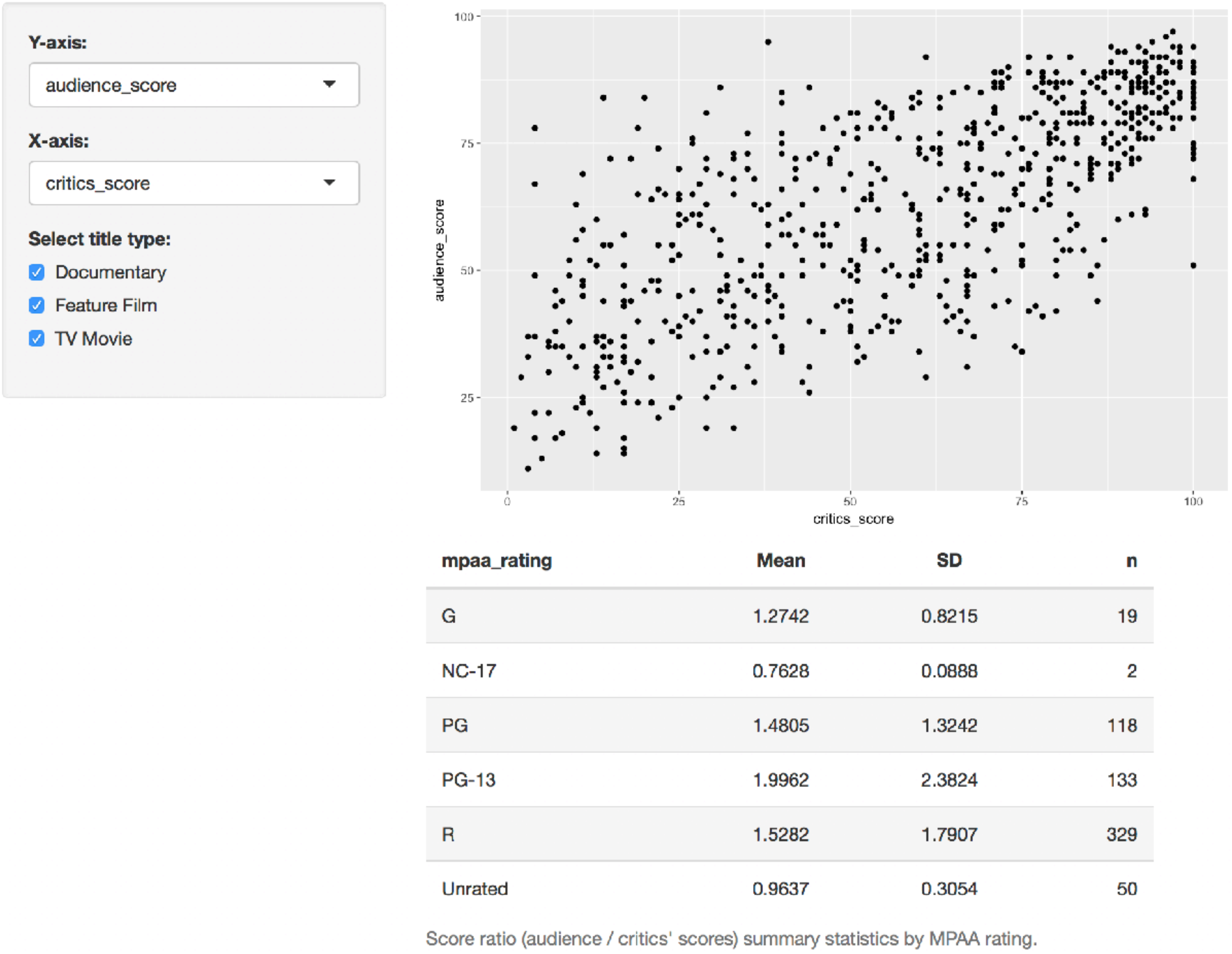


Y-axis:

audience_score

X-axis:

critics_score



Y-axis:

audience_score ▾

X-axis:

critics_score ▾

Select title type:

☑ Documentary

☑ Feature Film

☑ TV Movie

mpaa_rating	Mean	SD	n
G	1.2742	0.8215	19
NC-17	0.7628	0.0888	2
PG	1.4805	1.3242	118
PG-13	1.9962	2.3824	133
R	1.5282	1.7907	329
Unrated	0.9637	0.3054	50

Score ratio (audience / critics' scores) summary statistics by MPAA rating.

renderTable

Add a table beneath the plot displaying summary statistics for a new variable: `score_ratio = audience_score / critics_score`.

1. Calculate the new variable.
2. **ui:** Add an input widget that the user can interact with to check boxes for selected title types.
3. **ui:** Add an output defining where the summary table should appear.
4. **server:** Add a reactive expression that creates the summary table.

renderTable

Add a table beneath the plot displaying summary statistics for a new variable: `score_ratio = audience_score / critics_score`.

1. Calculate the new variable.

```
# Create new variable:
# ratio of critics and audience scores
movies <- movies %>%
  mutate(score_ratio = audience_score / critics_score)
```

renderTable

Add a table beneath the plot displaying summary statistics for a new variable: `score_ratio = audience_score / critics_score`.

2. **ui:** Add an input widget that the user can interact with to check boxes for selected title types.

```r
# Subset for title types
checkboxGroupInput(inputId = "selected_title_type",
                   label = "Select title type:",
                   choices = levels(movies$title_type),
                   selected = levels(movies$title_type))
```

renderTable

Add a table beneath the plot displaying summary statistics for a new variable: `score_ratio = audience_score / critics_score`.

3. **ui:** Add an output defining where the summary table should appear.

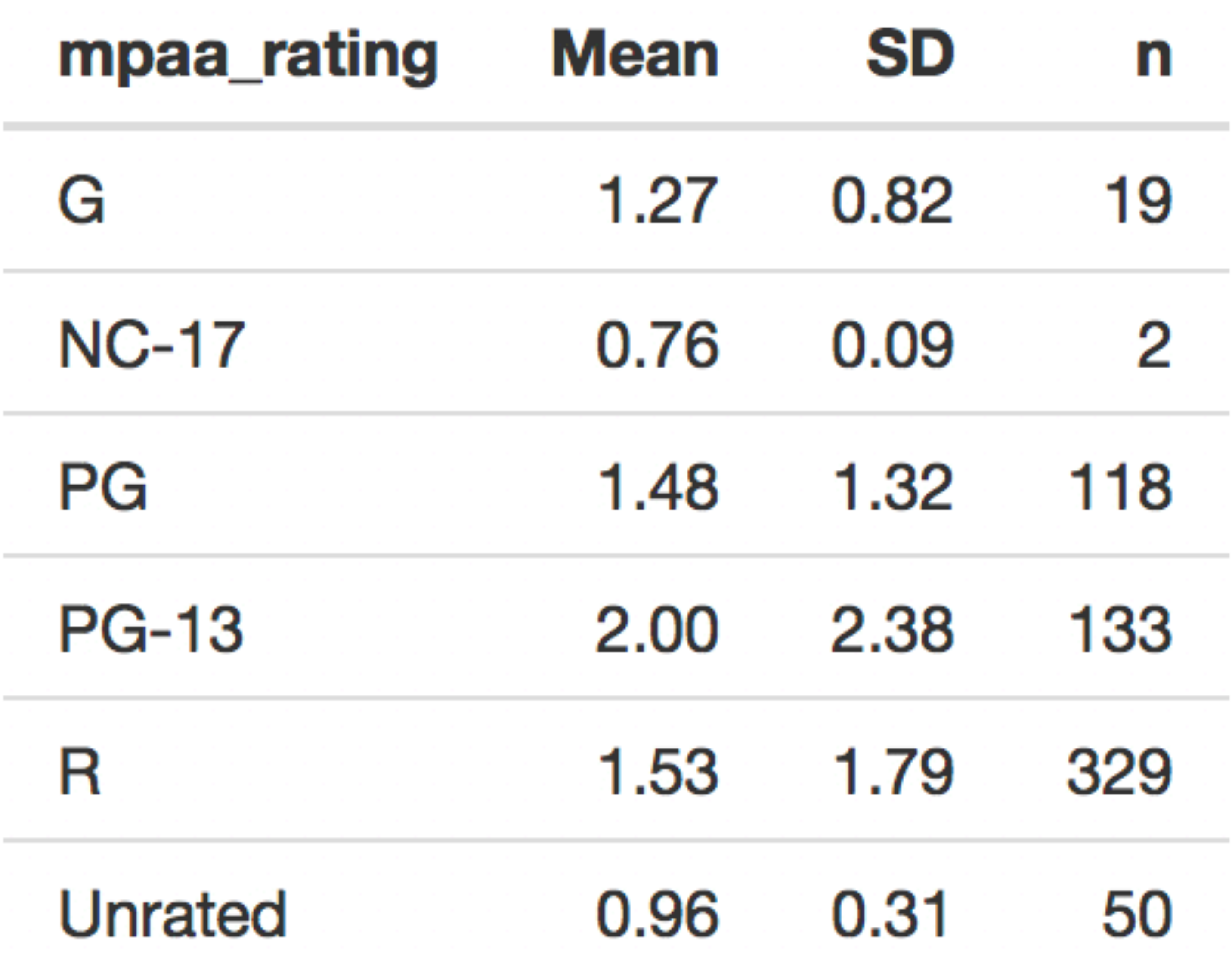
```
mainPanel(
  # Show scatterplot
  plotOutput(outputId = "scatterplot"),
  # Show data table
  tableOutput(outputId = "summarytable")
)
```

renderTable

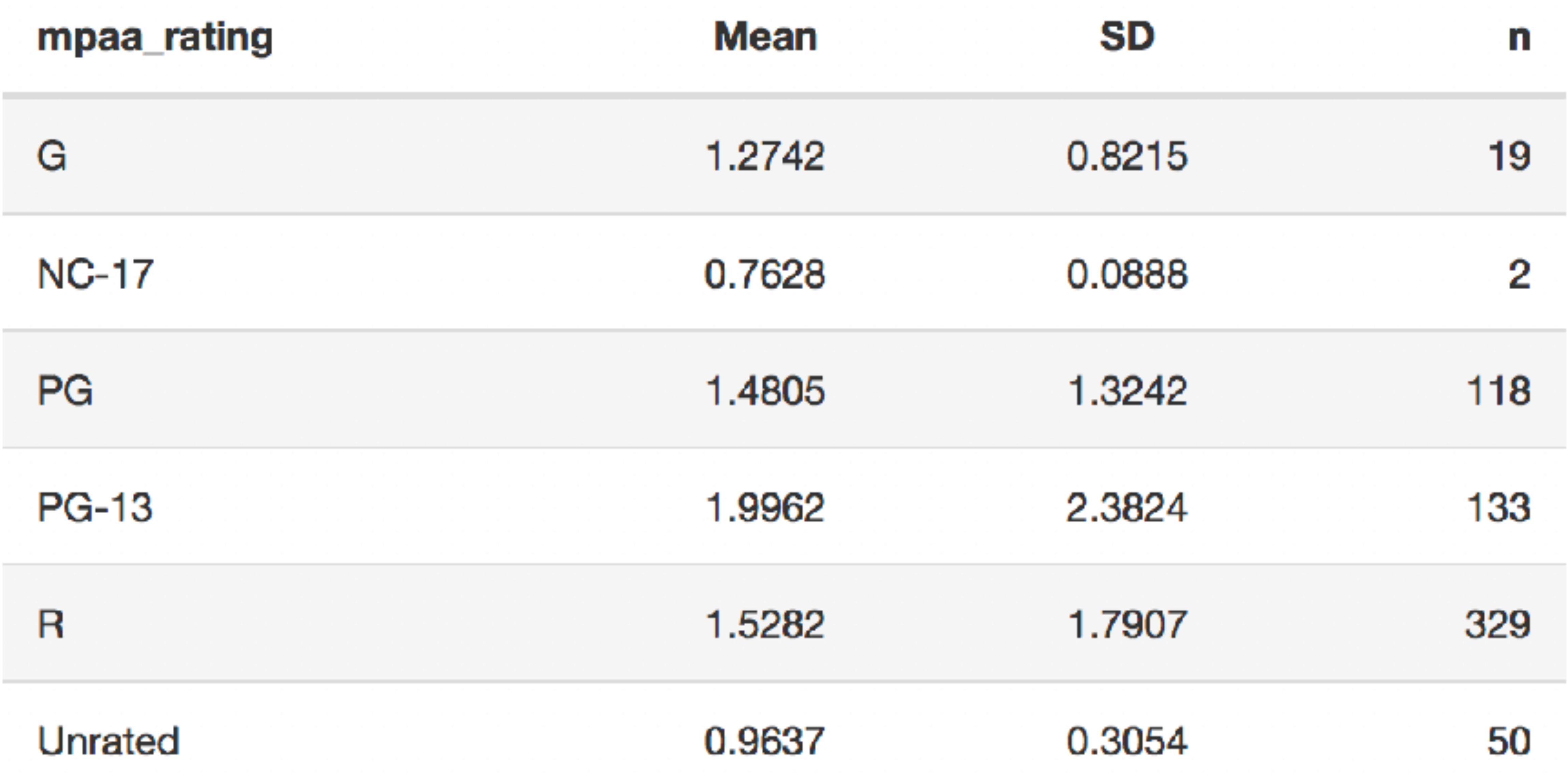
Add a table beneath the plot displaying summary statistics for a new variable: `score_ratio = audience_score / critics_score`.

4. **server:** Add a reactive expression that creates the summary table.

```
output$summarytable <- renderTable(
  {movies %>%
     filter(title_type %in% input$selected_title_type) %>%
     group_by(mpaa_rating) %>%
     summarise(Mean = mean(score_ratio), SD = sd(score_ratio), n = n())},
  striped = TRUE, spacing = "l", align = "lccr", digits = 4, width = "90%",
  caption = "Score ratio (audience / critics' scores) summary statistics by
MPAA rating."
)
```



mpaa_rating	Mean	SD	n
G	1.27	0.82	19
NC-17	0.76	0.09	2
PG	1.48	1.32	118
PG-13	2.00	2.38	133
R	1.53	1.79	329
Unrated	0.96	0.31	50



mpaa_rating	Mean	SD	n
G	1.2742	0.8215	19
NC-17	0.7628	0.0888	2
PG	1.4805	1.3242	118
PG-13	1.9962	2.3824	133
R	1.5282	1.7907	329
Unrated	0.9637	0.3054	50

Score ratio (audience / critics' scores) summary statistics by MPAA rating.

renderTable

Add a table beneath the plot displaying summary statistics for a new variable: `score_ratio = audience_score / critics_score`.

4. **server:** Add a reactive expression that creates the summary table.

```
output$summarytable <- renderTable(
  {movies %>%
     filter(title_type %in% input$selected_title_type) %>%
     group_by(mpaa_rating) %>%
     summarise(Mean = mean(score_ratio), SD = sd(score_ratio), n = n())},
  striped = TRUE, spacing = "l", align = "lccr", digits = 4, width = "90%",
  caption = "Score ratio (audience / critics' scores) summary statistics by
MPAA rating."
)
```

Recap

- Shiny has a variety of `render*` functions with corresponding `*Output` functions to create and display outputs.

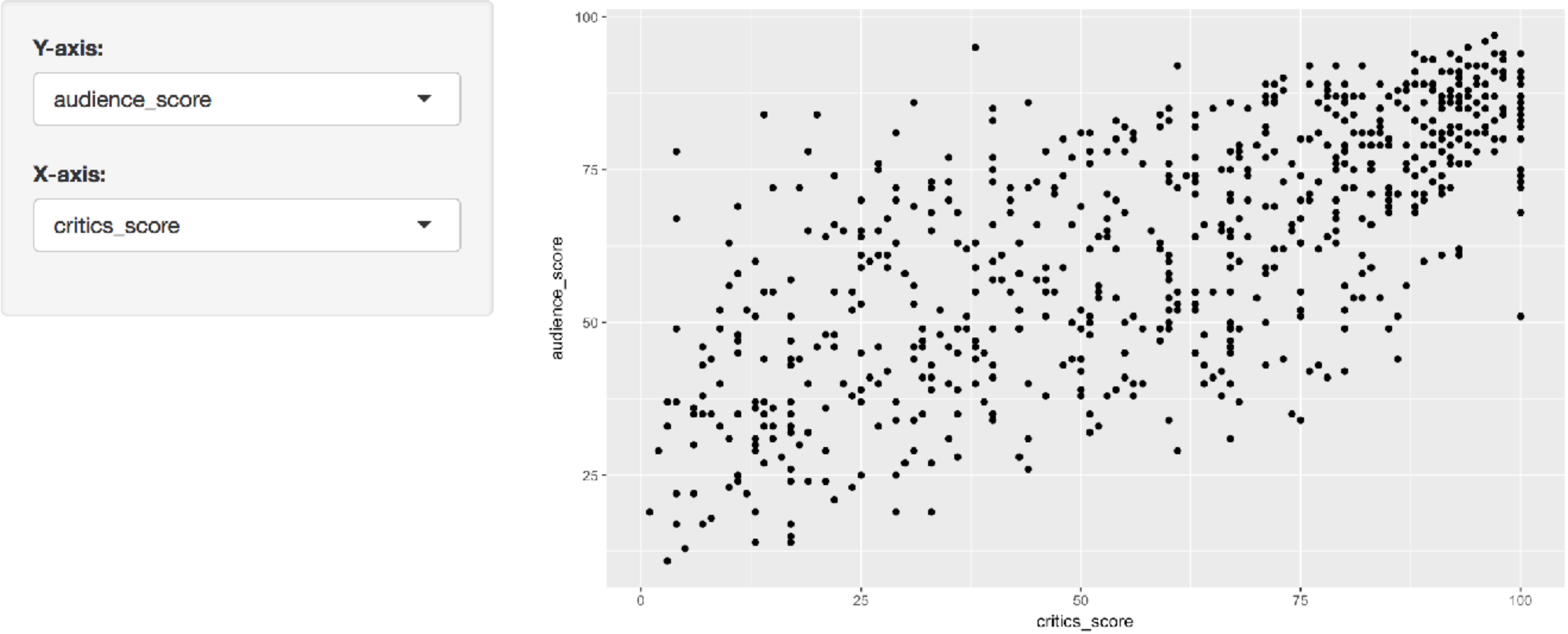
- `render*` functions can take on multiple arguments, the first being the expression for the desired output.

- The expression in the `render*` function should be wrapped in curly braces.

Let's practice!

BUILDING WEB APPLICATIONS IN R WITH SHINY

UI outputs

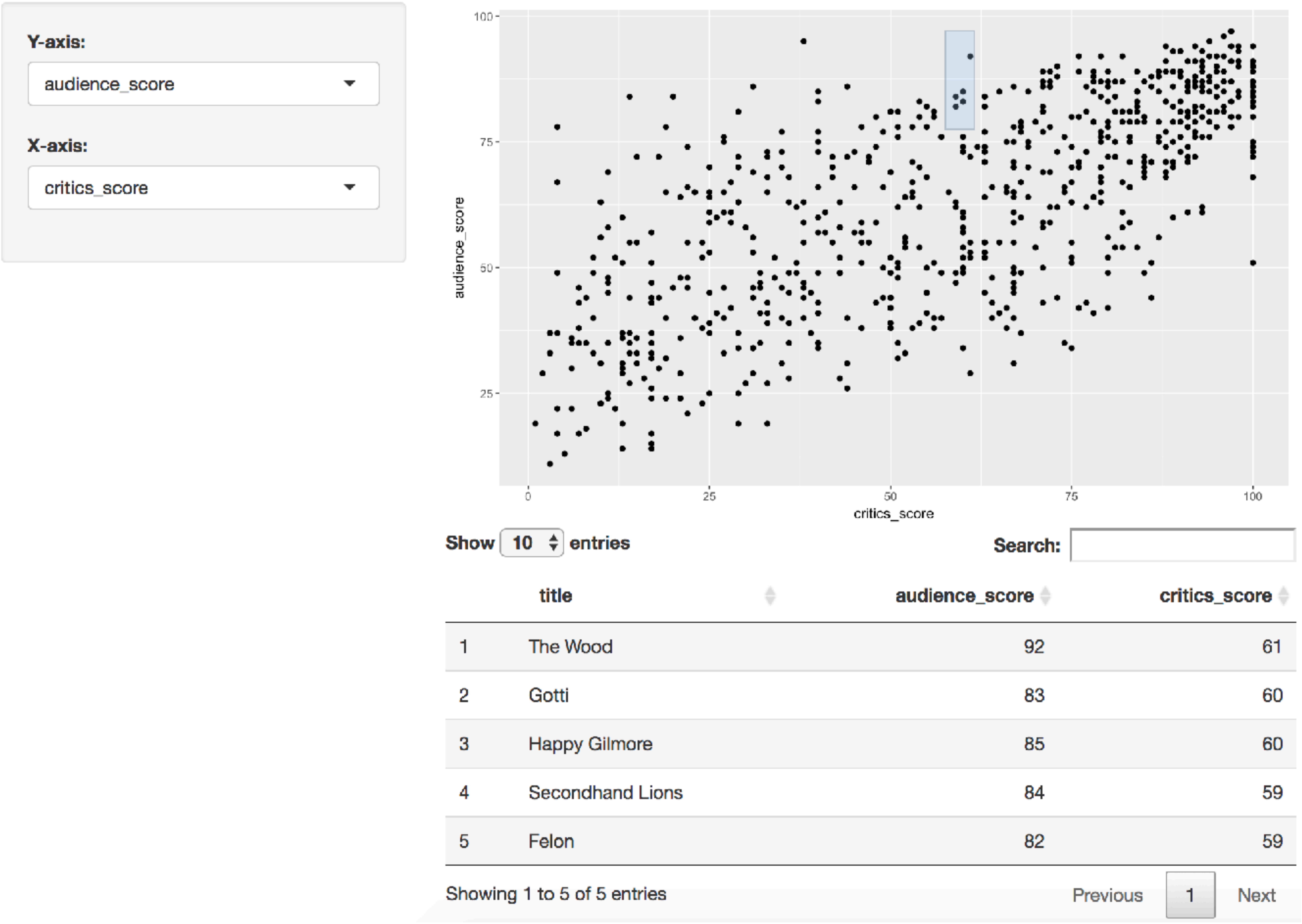


Y-axis:

audience_score

X-axis:

critics_score



Y-axis: audience_score

X-axis: critics_score

Show 10 entries

Search:

	title	audience_score	critics_score
1	The Wood	92	61
2	Gotti	83	60
3	Happy Gilmore	85	60
4	Secondhand Lions	84	59
5	Felon	82	59

Showing 1 to 5 of 5 entries

plotOutput

Select points on the plot via brushing, and report the selected points in a data table underneath the plot.

1. **ui:** Add functionality to `plotOutput` to select points via brushing.
2. **ui:** Add an output defining where the data table should appear.
3. **server:** Add a reactive expression that creates the data table for the selected points.

plotOutput

Select points on the plot via brushing, and report the selected points in a data table underneath the plot.

1. **ui:** Add functionality to `plotOutput` to select points via brushing.

```
# Show scatterplot with brushing capability
plotOutput(outputId = "scatterplot", brush = "plot_brush")
```

plotOutput

Select points on the plot via brushing, and report the selected points in a data table underneath the plot.

2. **ui:** Add an output defining where the data table should appear.

```
# Show data table
DT::dataTableOutput(outputId = "moviestable")
```

plotOutput

Select points on the plot via brushing, and report the selected points in a data table underneath the plot.

3. **server:** Add a reactive expression that creates the data table for the selected points.

```
# Print data table
output$moviestable <- DT::renderDataTable({
 brushedPoints(movies, input$plot_brush) %>%
  select(title, audience_score, critics_score)
})
```

Let's practice!