Shiny_HW

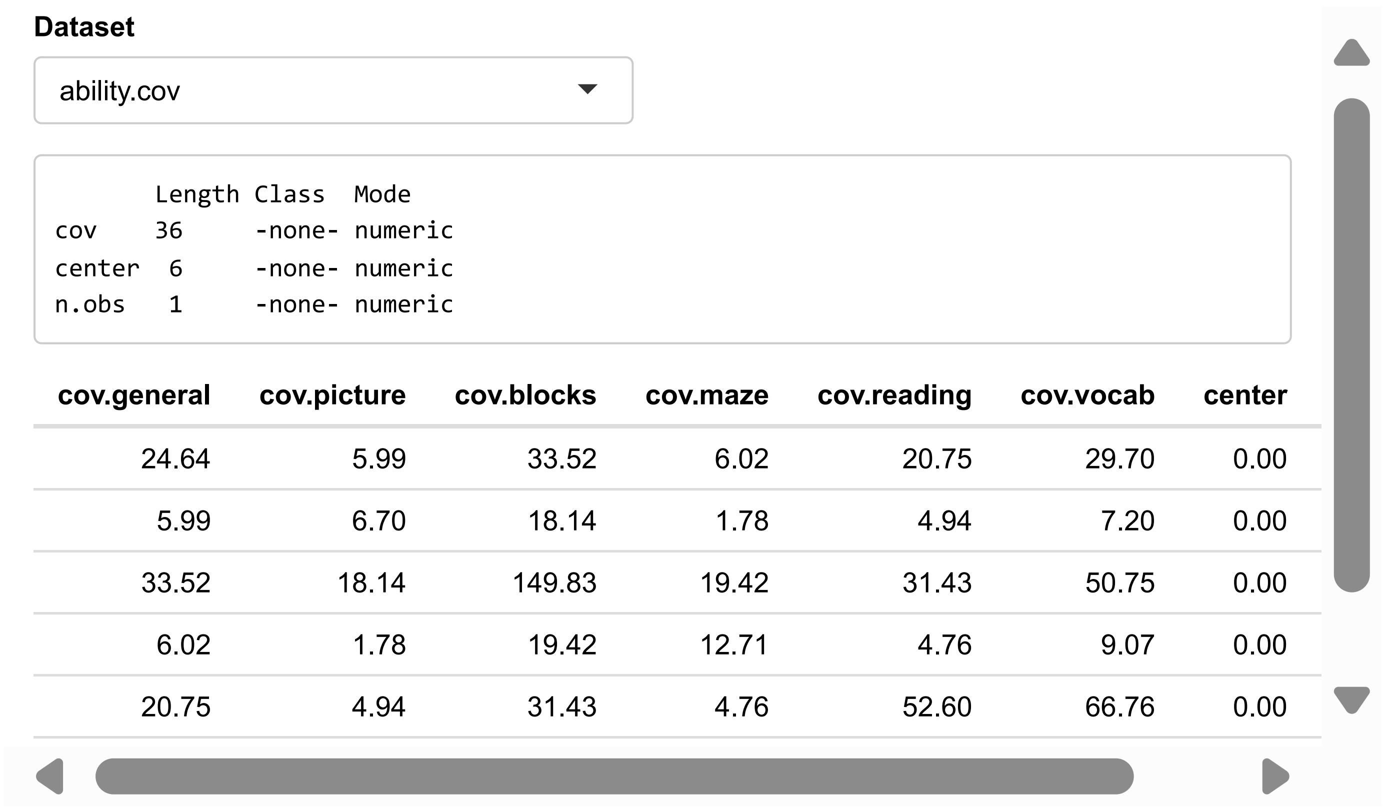
Ruihang Han

2024-11-15

```
library(shiny)
```

Hadley_1

```
ui <- fluidPage(
  selectInput("dataset", label = "Dataset", choices = ls("package:datasets")),
  verbatimTextOutput("summary"),
  tableOutput("table")
)


server <- function(input, output, session) {
  output$summary <- renderPrint({
    dataset <- get(input$dataset, "package:datasets")
    summary(dataset)
  })

  output$table <- renderTable({
    dataset <- get(input$dataset, "package:datasets")
    dataset
  })
}

shinyApp(ui, server)
```

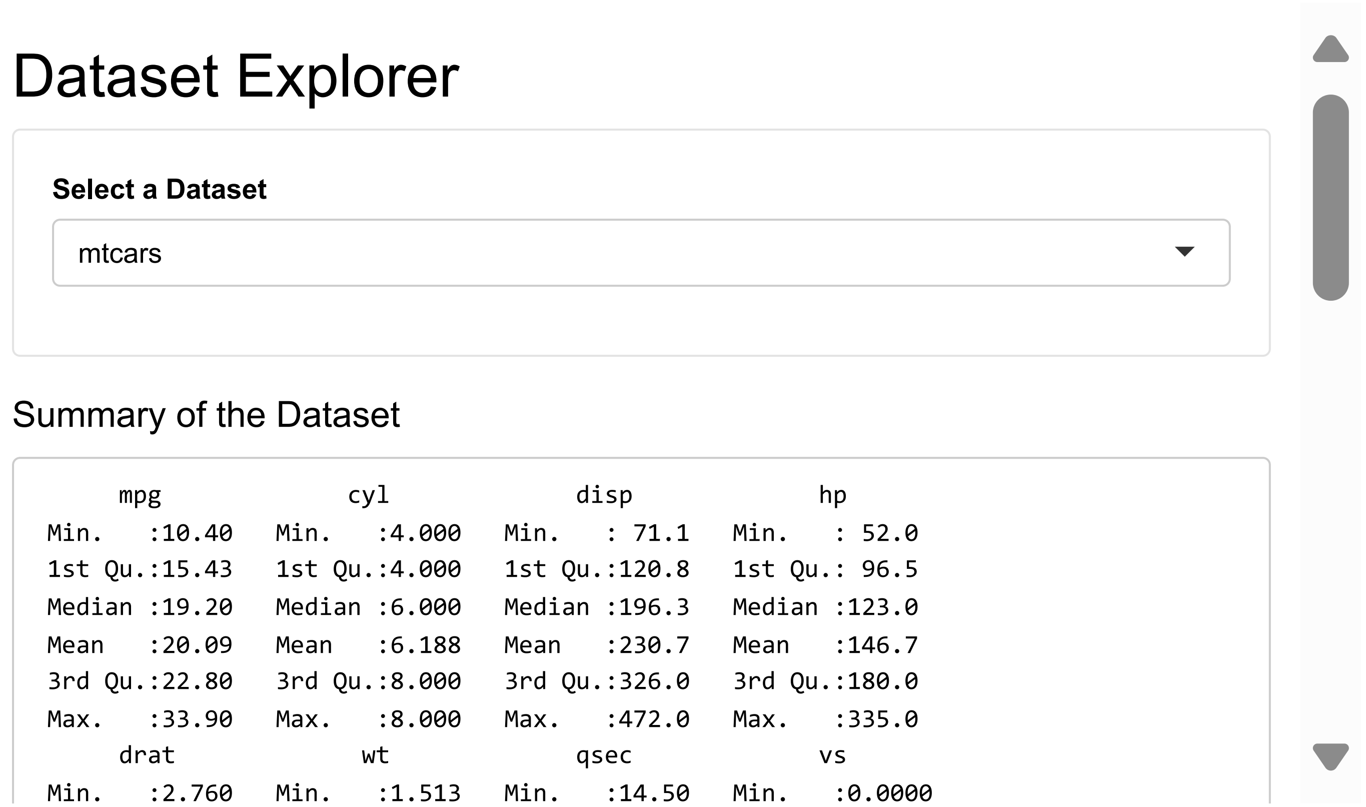


```
       Length Class  Mode
cov    36     -none- numeric
center  6     -none- numeric
n.obs   1     -none- numeric
```

cov.general	cov.picture	cov.blocks	cov.maze	cov.reading	cov.vocab	center
24.64	5.99	33.52	6.02	20.75	29.70	0.00
5.99	6.70	18.14	1.78	4.94	7.20	0.00
33.52	18.14	149.83	19.42	31.43	50.75	0.00
6.02	1.78	19.42	12.71	4.76	9.07	0.00
20.75	4.94	31.43	4.76	52.60	66.76	0.00

Hadley_2

```r
library(shiny)

ui <- fluidPage(
  titlePanel("Dataset Explorer"),
  sidebarLayout(
    sidebarPanel(
      selectInput("selected_dataset", label = "Select a Dataset",
                  choices = ls("package:datasets"),
                  selected = "mtcars")
    ),
    mainPanel(
      h4("Summary of the Dataset"),
      verbatimTextOutput("dataset_summary"),
      h4("Preview of the Dataset"),
      tableOutput("dataset_table")
    )
  )
)

server <- function(input, output, session) {
  # Reactive function to retrieve the chosen dataset
  chosen_data <- reactive({
    get(input$selected_dataset, "package:datasets")
  })

  output$dataset_summary <- renderPrint({
    # Call reactive function
    summary(chosen_data())
  })

  output$dataset_table <- renderTable({
    # Display first 15 rows to keep the output manageable
    head(chosen_data(), 15)
  })
}

shinyApp(ui, server)
```



Dataset Explorer

Select a Dataset

mtcars

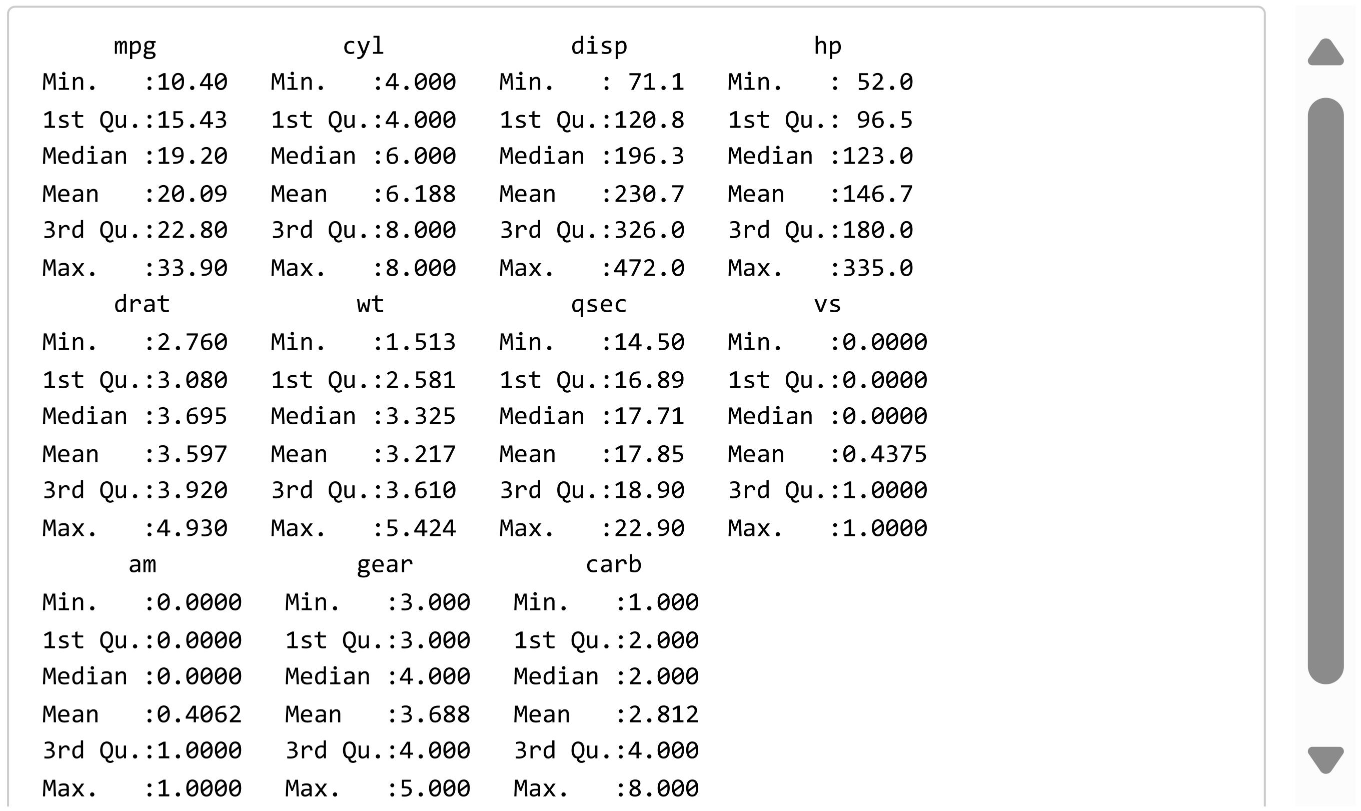
Summary of the Dataset

```
     mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
      drat             wt             qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
```

2.3.5 # 1. Pair render functions with output functions library(shiny)

App with verbatimTextOutput for renderPrint(summary(mtcars))

```
ui <- fluidPage(verbatimTextOutput("summary_output"))
server <- function(input, output, session) { output$summary_output <- renderPrint({ summary(mt
cars) }) }
shinyApp(ui, server)
```



```
     mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
      drat             wt             qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
       am              gear            carb      
 Min.   :0.0000   Min.   :3.000   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
 Median :0.0000   Median :4.000   Median :2.000  
 Mean   :0.4062   Mean   :3.688   Mean   :2.812  
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
 Max.   :1.0000   Max.   :5.000   Max.   :8.000  
```

App with textOutput for renderText("Good morning!")

```
ui <- fluidPage(textOutput("text_output"))
server <- function(input, output, session) { output$text_output <- renderText({ "Good mornin
g!" }) }
shinyApp(ui, server)
```

Good morning!

App with verbatimTextOutput for renderPrint(t.test(1:5, 2:6))

```
ui <- fluidPage(verbatimTextOutput("ttest_output"))
server <- function(input, output, session) { output$ttest_output <- renderPrint({ t.test(1:5,
2:6) }) }
shinyApp(ui, server)
```

```

	Welch Two Sample t-test

data:  1:5 and 2:6
t = -1, df = 8, p-value = 0.3466
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.306004  1.306004
sample estimates:
mean of x mean of y
        3         4
```

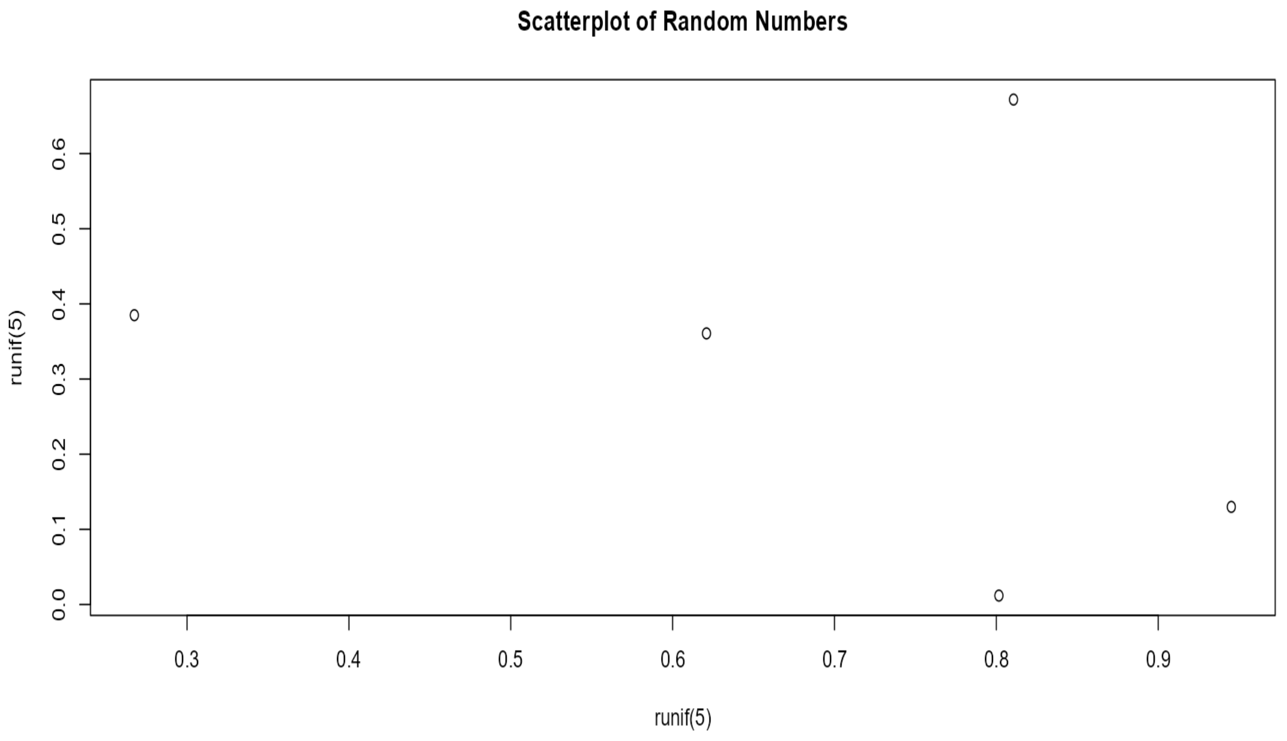
App with verbatimTextOutput for renderText(str(lm(mpg ~ wt, data = mtcars)))

```
ui <- fluidPage(verbatimTextOutput(”lm_output”))
server <- function(input, output, session) { output$lm_output <- renderPrint({ str(lm(mpg ~ w
t, data = mtcars)) }) }
shinyApp(ui, server)
```

```
List of 12
 $ coefficients : Named num [1:2] 37.29 -5.34
  ..- attr(*, "names")= chr [1:2] "(Intercept)" "wt"
 $ residuals    : Named num [1:32] -2.28 -0.92 -2.09 1.3 -0.2 ...
  ..- attr(*, "names")= chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 D
 $ effects      : Named num [1:32] -113.65 -29.116 -1.661 1.631 0.111 ...
  ..- attr(*, "names")= chr [1:32] "(Intercept)" "wt" "" "" ...
 $ rank         : int 2
 $ fitted.values: Named num [1:32] 23.3 21.9 24.9 20.1 18.9 ...
  ..- attr(*, "names")= chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 D
 $ assign       : int [1:2] 0 1
 $ qr           :List of 5
  ..$ qr   : num [1:32, 1:2] -5.657 0.177 0.177 0.177 0.177 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
  .. .. ..$ : chr [1:2] "(Intercept)" "wt"
  .. ..- attr(*, "assign")= int [1:2] 0 1
  ..$ qraux: num [1:2] 1.18 1.05
  ..$ pivot: int [1:2] 1 2
  ..$ tol  : num 1e-07
  ..$ rank : int 2
```

2. Customized Shiny app with scatterplot and alt text

```
ui <- fluidPage(plotOutput("scatterplot"))
server <- function(input, output, session) { output$scatterplot <- renderPlot({ plot(runif(5),
runif(5), main = "Scatterplot of Random Numbers") }) }
shinyApp(ui, server)
```



3. renderDataTable() with options to suppress controls

```
library(DT)
ui <- fluidPage(DTOutput("table"))
server <- function(input, output, session) { output$table <- renderDataTable(mtcars, options =
list(pageLength = 5, searching = FALSE, ordering = FALSE, info = FALSE, lengthChange = FALSE, p
aging = FALSE)) }
shinyApp(ui, server)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21	6	160	110	3.9	2.62	16.46	0	1	4	4
Mazda RX4 Wag	21	6	160	110	3.9	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.46	20.22	1	0	3	1
Duster 360	14.3	8	360	245	3.21	3.57	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.19	20	1	0	4	2

4. reactable instead of renderDataTable

```
library(reactable)
ui <- fluidPage(reactableOutput("table"))
server <- function(input, output, session) { output$table <- renderReactable({ reactable(mtcar
s, pagination = FALSE) }) }
shinyApp(ui, server)
```

	mpg	cyl	disp	hp	drat
Mazda RX4	21	6	160	110	3.9
Mazda RX4 Wag	21	6	160	110	3.9
Datsun 710	22.8	4	108	93	3.85
Hornet 4 Drive	21.4	6	258	110	3.08
Hornet Sportabout	18.7	8	360	175	3.15
Valiant	18.1	6	225	105	2.76
Duster 360	14.3	8	360	245	3.21
Merc 240D	24.4	4	146.7	62	3.69
Merc 230	22.8	4	140.8	95	3.92

3.3.6

1.

```
ui <- fluidPage(
  textInput("name", "What's your name?"),
  textOutput("greeting")
)
```

```
server1 <- function(input, output, session) {
  output$greeting <- renderText({
    paste0("Hello ", input$name)
  })
}
server2 <- function(input, output, session) {
  output$greeting <- renderText({
    paste0("Hello ", input$name)
  })
}
server3 <- function(input, output, session) {
  output$greeting <- renderText({
    paste0("Hello ", input$name)
  })
}
shinyApp(ui, server)
```

What's your name?

2. reactive graph1

input$ainput$b | | V V reactive(c) (c <- input$a + input$b) | V input$dreactive(e)(e < -c() + input$d) | | V V output$f (renderText(e()))

reactive graph2

input$x1input$x2 input$x3|||VVVreactive(x)(x < -input$x1 + input$x2 + input$x3)

input$y1input$y2 | | V V reactive(y) (y <- input$y1 + input$y2) | V output$z (renderText(x() / y()))

reactive graph3 input$a input$b input$c input$d | | | | V V V V reactive(a) reactive(b) reactive(c) reactive(d) (a <- input$a * 10)(b < -a() + input$b) (c <- b() / input$c)(d < -c()^input$d)

3.

```
# Define a reactive variable for the selected column
selected_var <- reactive(df[[input$var]])

# Define a reactive expression for the range of the selected variable
selected_var_range <- reactive({
  range(selected_var(), na.rm = TRUE)
})
```

This code will fail due to a naming conflict.
In R, `range` is a built-in function, so using it as the name of a reactive expression can cause unexpected errors or behavior.
To avoid overriding the base function, it’s recommended to use a different name for the reactive expression.

4.8 1.

```
              ┌────────────────────────┐
              │  input$code     │
              └──────────┬─────────────┘
                    │
          ┌──────────────▼────────────────┐
          │  selected()          │
          │  (filters injuries   │
          │   based on prod_code)│
          └─────┬────────────────┬────────┘
             │          │
┌───────────────────▼─────┐  ┌──────▼──────────────────┐
│  output$diag     │  │  output$body_part│
│  (table count    │  │  (table count     │
│   by diag)       │  │   by body_part)   │
└─────────────────────────┘  └─────────────────────────┘
              │
              │
          ┌─────▼───────────────────────┐
          │  output$location    │
          │  (table count       │
          │   by location)      │
          └─────▲───────────────────────┘
              │
       ┌───────────┴─────────────────────────┐
       │  summary()              │
       │  (count by age, sex,    │
       │   then joined with      │
       │   population)           │
       └──────────────┬──────────────────────┘
               │
      ┌───────────────▼──────────────────┐
      │  output$age_sex         │
      │  (line plot by age,     │
      │   n per age-sex group)  │
      └──────────────────────────────────────┘
```

```
              ┌────────────────────────┐
              │  input$code     │
              └──────────┬─────────────┘
                    │
          ┌──────────────▼────────────────┐
          │  selected()          │
          │  (filters injuries   │
          │   based on prod_code)│
          └─────┬────────────────┬────────┘
             │          │
┌───────────────────▼─────┐  ┌──────▼──────────────────┐
│  output$diag     │  │  output$body_part│
│  (table count    │  │  (table count     │
│   by diag)       │  │   by body_part)   │
└─────────────────────────┘  └─────────────────────────┘
              │
```

```
         |
     ┌───▼──────────────────┐
     │ output$location   │
     │ (table count      │
     │  by location)     │
     └───▲──────────────────┘
         |
   ┌────────┴───────────────────┐
   │ summary()              │
   │ (count by age, sex,    │
   │  then joined with      │
   │  population)           │
   └──────────┬──────────────────┘
             |
 ┌─────────────▼──────────────────────────────┐
 │ input$y                               │
 │ (user choice: rate vs count)          │
 └──────────────┬───────────────────────────────┘
             |
  ┌──────────▼────────────┐
  │ output$age_sex       │
  │ (conditional plot by │
  │  age: rate or count  │
  │  based on input$y)   │
  └────────────────────────────┘



      ┌───────────────────┐
      │ input$code    │
      └───────┬──────────┘
            |
     ┌─────────▼─────────┐
     │ selected()          │
     │ (filters injuries   │
     │  based on prod_code)│
     └────┬──────────┬────────┘
        |          |
┌──────────────▼───┐  ┌─────▼────────────┐
│ output$diag     │  │ output$body_part│
│ (table count    │  │ (table count     │
│  by diag)       │  │  by body_part)   │
└───────────────────┘  └──────────────────────┘
         |
         |
     ┌───▼──────────────────┐
     │ output$location   │
     │ (table count      │
     │  by location)     │
     └───▲──────────────────┘
         |
   ┌────────┴───────────────────┐
   │ summary()              │
   │ (count by age, sex,    │
   │  then joined with      │
```

```
        │  population)           │
        └───────────────┬───────────────────┘
                    │
       ┌───────────────▼───────────────────┐
       │ output$age_sex         │
       │ (conditional plot by   │
       │  age: rate or count)   │
       └───────────────┬───────────────────┘
                    │
       ┌───────────────▼───────────────────┐
       │ input$story            │
       └───────────────┬───────────────────┘
                    │
      ┌────────────────▼────────────────────┐
      │ narrative_sample()      │
      │ (gets new narrative     │
      │  on button click)       │
      └────────────────┬────────────────────┘
                    │
        ┌──────────────▼─────────────────────┐
        │ output$narrative        │
        │ (displays current       │
        │  narrative text)        │
        └────────────────────────────────────┘
```

2. If you flip fct_infreq() and fct_lump(), the code will lump all values first, then order by frequency. This would lead to a less accurate table where less common factors may end up lumped with more common ones, affecting the interpretability and accuracy of the summarized table.

3.

#column(4, sliderInput("num_rows", "Number of rows:", min = 1, max = 10, value = 5))

#output$diag <- renderTable(count_top(selected(), diag, n = input$num_rows), width = "100%") #output$body_part <- renderTable(count_top(selected(), body_part, n = input$num_rows), width = "100%") #output$location <- renderTable(count_top(selected(), location, n = input$num_rows), width = "100%")

4.

```
fluidRow(
  column(1, actionButton("prev_story", "Previous")),
  column(1, actionButton("next_story", "Next")),
  column(10, textOutput("narrative"))
)
```

Previous

Next

#narrative_index <- reactiveVal(1)

#observeEvent(input$next_story, { # current <- narrative_index() # narrative_index(min(current + 1, nrow(selected()))) #})

#observeEvent(input$prev_story, { # current <- narrative_index() # narrative_index(max(current - 1, 1)) #})

#output$narrative <- renderText({ # selected() %>% pull(narrative) %>% .[narrative_index()] #})