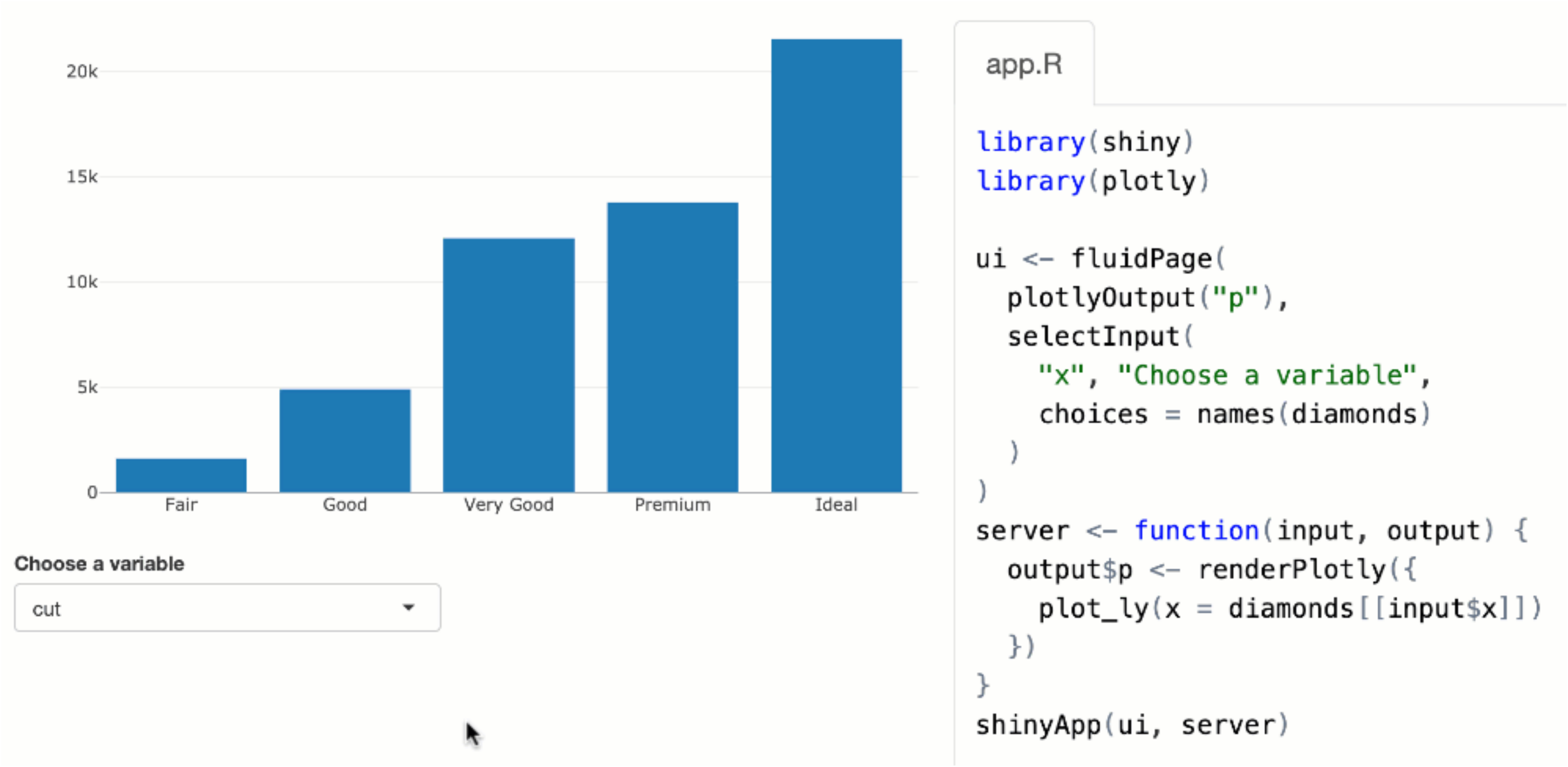
Shiny: **Interactive** webapps in R

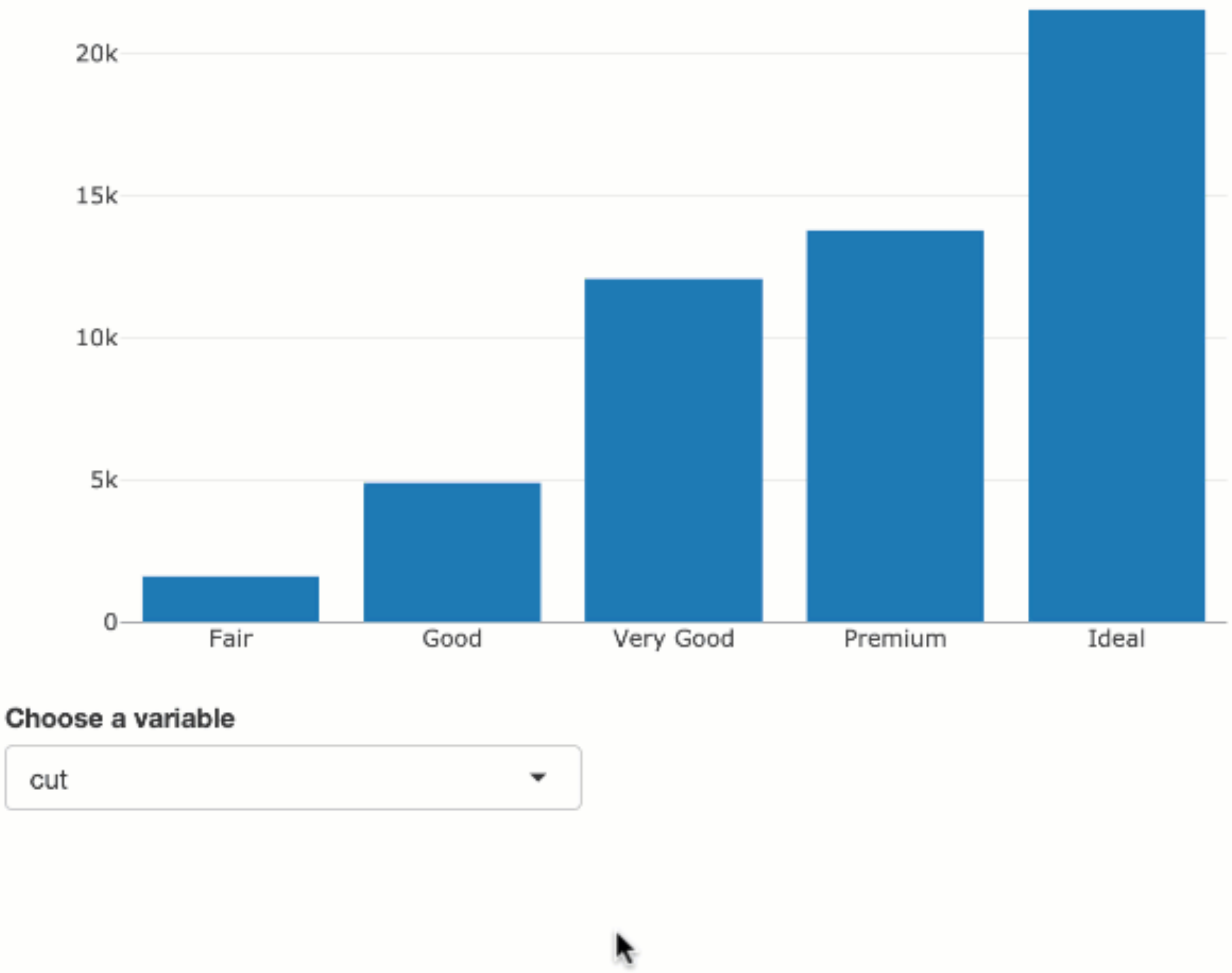
- Easily turn your R code into an interactive GUI.

- Allow users to **quickly explore** different parameters, models/algorithms, other information



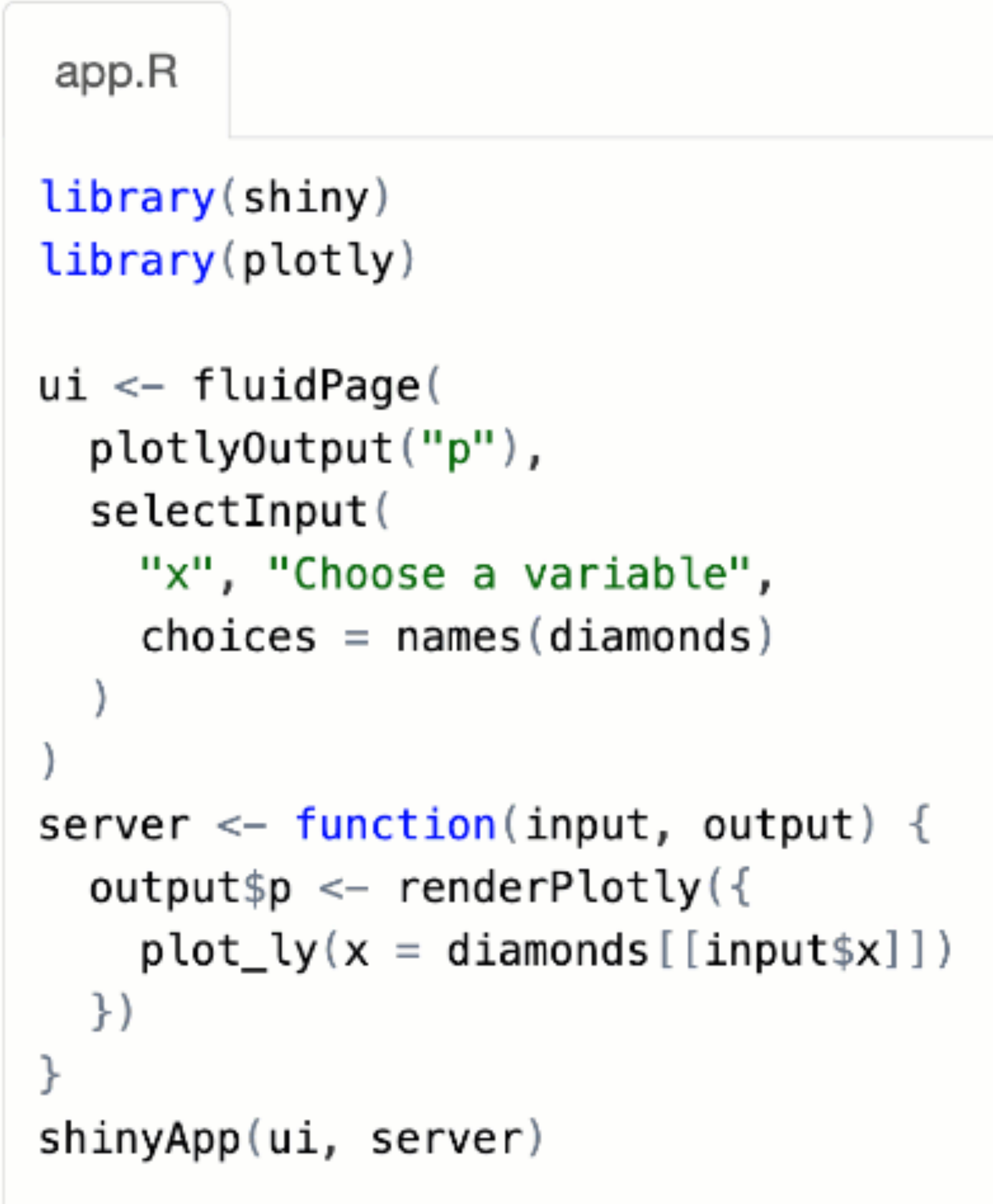
```r
library(shiny)
library(plotly)

ui <- fluidPage(
  plotlyOutput("p"),
  selectInput(
    "x", "Choose a variable",
    choices = names(diamonds)
  )
)
server <- function(input, output) {
  output$p <- renderPlotly({
    plot_ly(x = diamonds[[input$x]])
  })
}
shinyApp(ui, server)
```



Choose a variable

cut

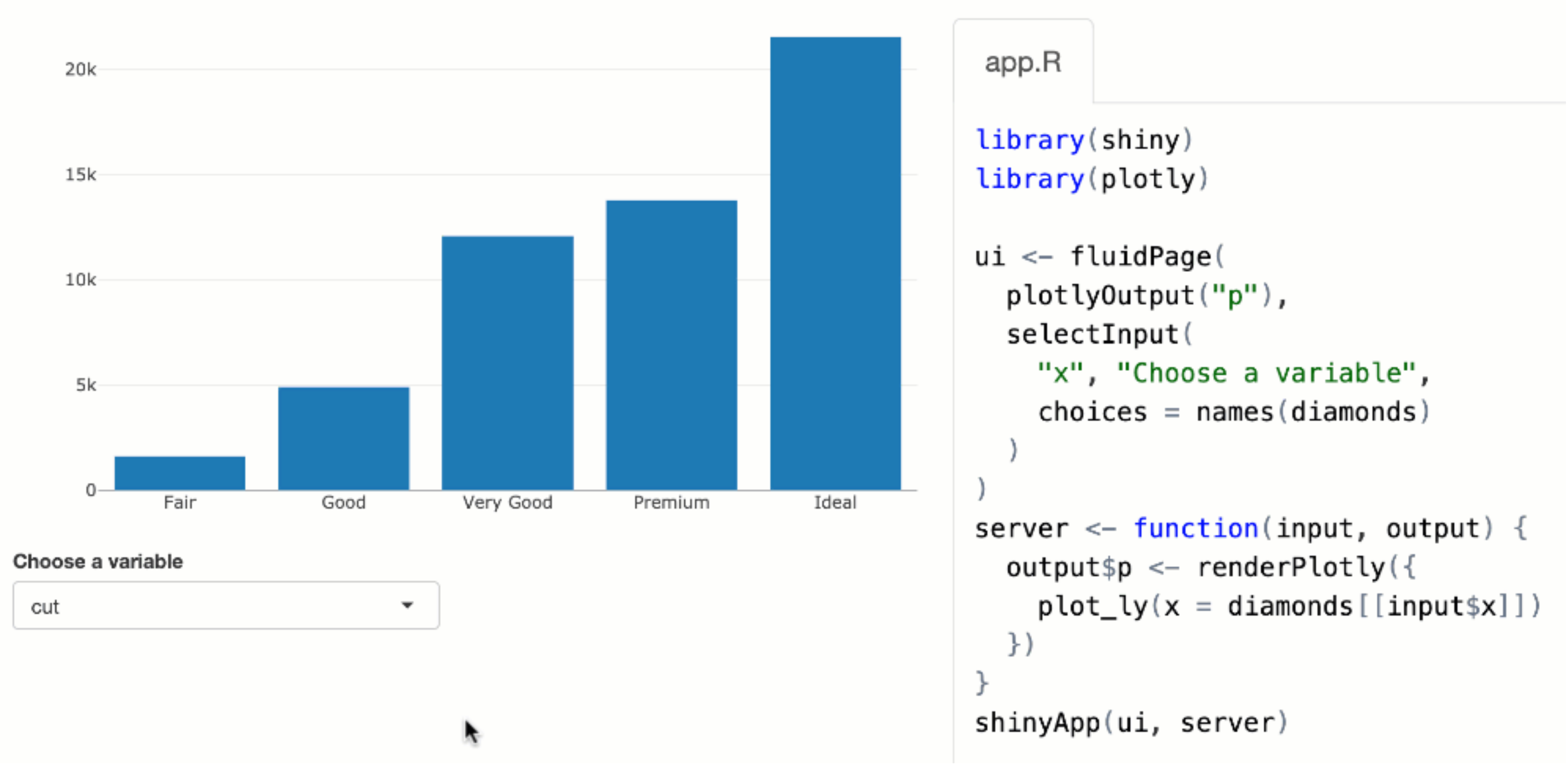


```r
library(shiny)
library(plotly)

ui <- fluidPage(
  plotlyOutput("p"),
  selectInput(
    "x", "Choose a variable",
    choices = names(diamonds)
  )
)
server <- function(input, output) {
  output$p <- renderPlotly({
    plot_ly(x = diamonds[[input$x]])
  })
}
shinyApp(ui, server)
```

Shiny: **Interactive** webapps in R

- Easily turn your R code into an interactive GUI.
- Allow users to **quickly explore** different parameters, models/algorithms, other information



Choose a variable

cut

app.R

```r
library(shiny)
library(plotly)

ui <- fluidPage(
  plotlyOutput("p"),
  selectInput(
    "x", "Choose a variable",
    choices = names(diamonds)
  )
)
server <- function(input, output) {
  output$p <- renderPlotly({
    plot_ly(x = diamonds[[input$x]])
  })
}
shinyApp(ui, server)
```

Interactivity is great, but **reproducibility suffers**

- Reproducing results is *possible* by replicating user events (or bookmarking), but results are **locked behind a GUI**
- Even if you can view the app's source code, the **domain logic is intertwined** with Shiny code
  - Methodology is less transparent
  - Harder to verify results are 'correct'