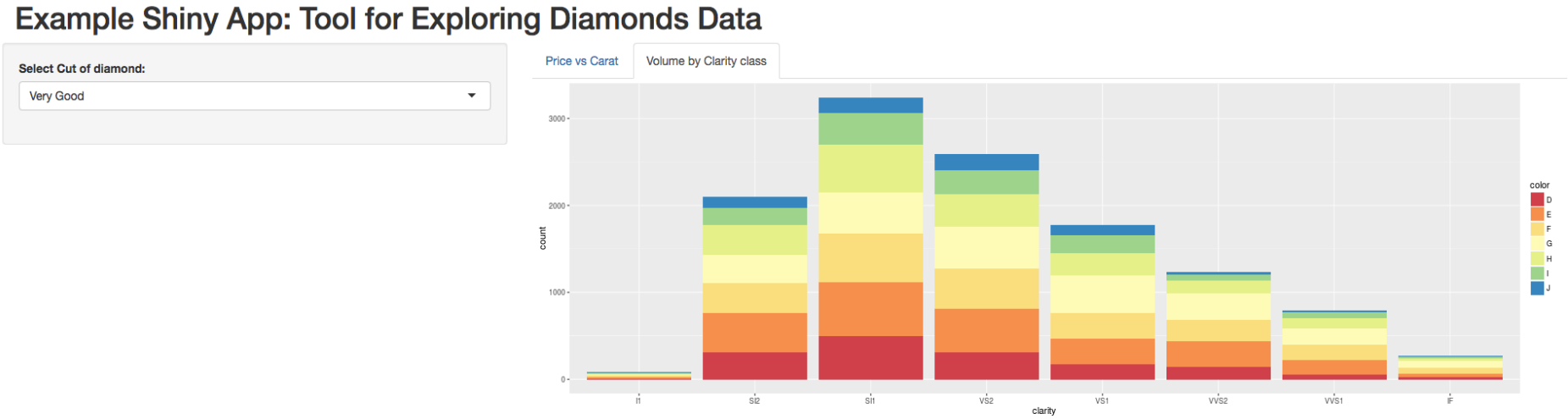
```
library(dplyr)

rladies_global %>%
  filter(city == 'Munich')
```

Shiny Apps: Developing Data Products with R

https://chiin.shinyapps.io/diamonds/



https://github.com/Chiin-Git

Diamonds dataset

```
> ?diamonds
```

diamonds {ggplot2}

Prices of 50,000 round cut diamonds

Description

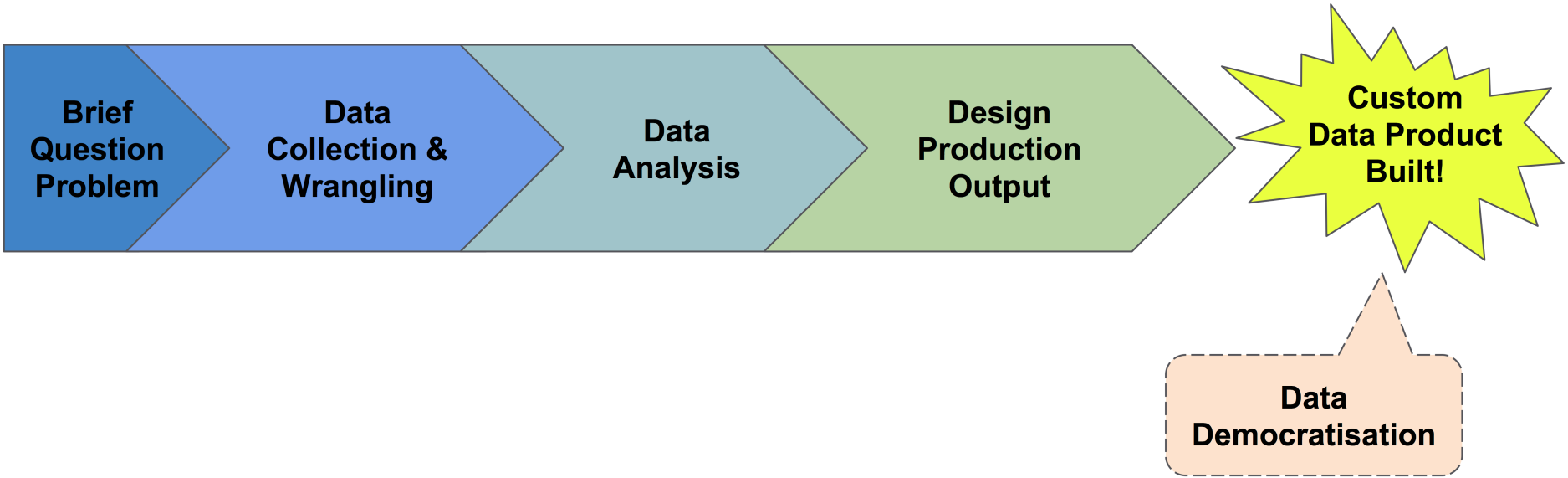
A dataset containing the prices and other attributes of almost 54,000 diamonds.

Format

A data frame with 53940 rows and 10 variables:

- **price**: price is US dollars ($326 - $18,823)
- **carat**: weight of the diamond (0.2 – 5.01)
- **cut**: quality of the cut (Fair, Good, Very Good, Premium, Ideal)
- **color**: diamond color, from J (worst) to D (best)
- … and six more

Example Use-Case: Diamonds



Slides courtesy of @Chiin-Git (R-Ladies London): http://bit.ly/2xZ47F4

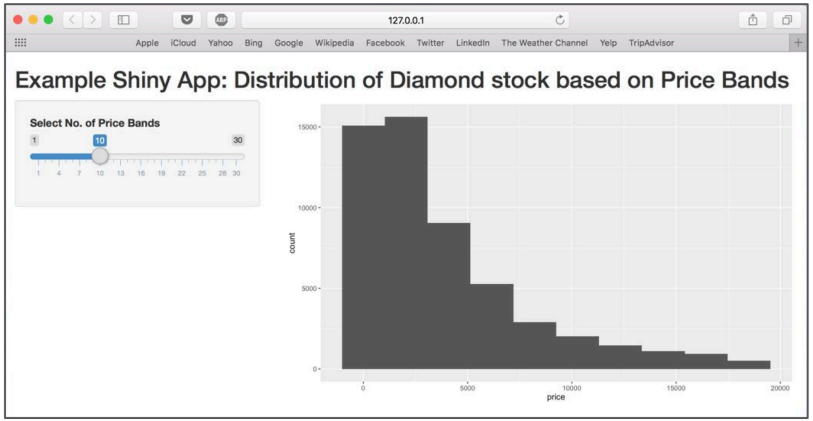
Shiny App: ui & server components

ui object:
Code for creating the web page

server object:
Code for the computer running the R calculations (e.g. your laptop, a server somewhere)

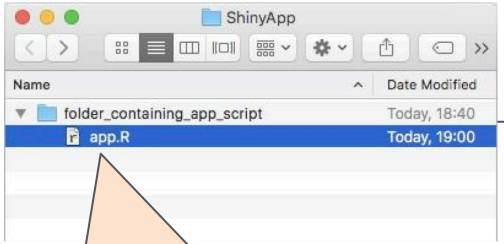
shinyApp(ui = **ui**, server = **server**)

EXAMPLE OUTPUT



Slides courtesy of @Chiin-Git (R-Ladies London): http://bit.ly/2xZ47F4

Shiny App: File & Directory instructions



Making “single-file” shiny apps:

- create one script
- call it “app.R”
- save app.R in it’s own directory by itself
- Note: there is an alternative, original method involving creation of two separate files, ui.R and server.R with a minor difference in code syntax

Template for app.R script

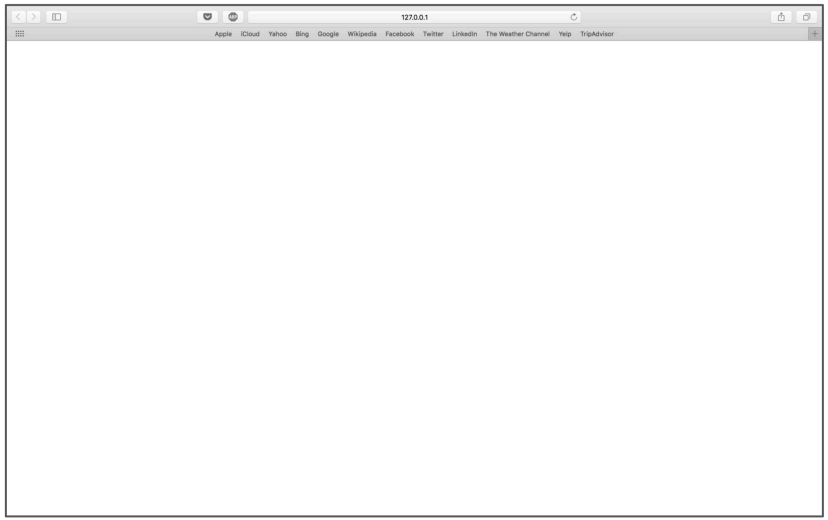
```
#Remember to load & define variables
for the global environment,
e.g. packages, objects

library(shiny)

ui <- basicPage()

server <- function(input, output) {}

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

Slides courtesy of @Chiin-Git (R-Ladies London): http://bit.ly/2xZ47F4

Try building some apps - 1. Bare minimum, i.e empty app

app.R

```
#Remember to load & define variables for the global environment, e.g. packages, objects

library(shiny)

ui <- basicPage()

server <- function(input, output) {}

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



Slides courtesy of @Chiin-Git (R-Ladies London): http://bit.ly/2xZ47F4

Try building some apps - 2. Basic ui Layout

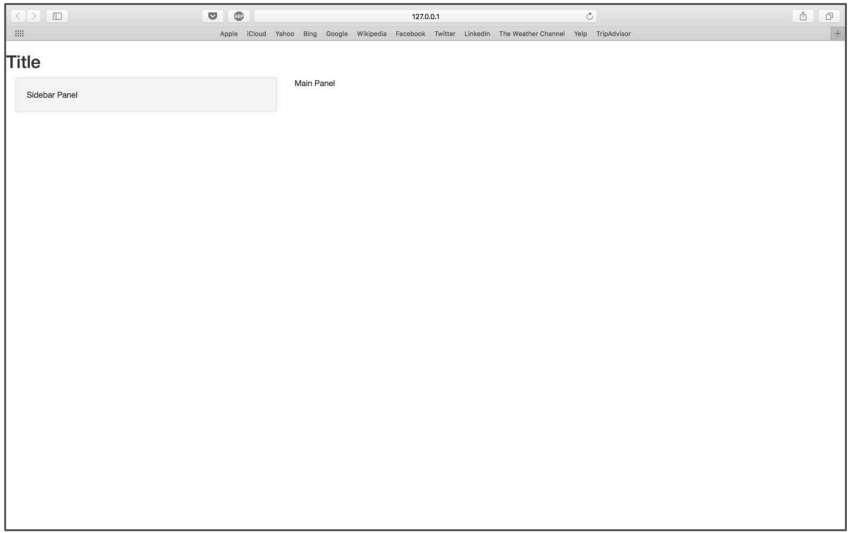
app.R

```
#Remember to load & define variables for the global environment, e.g. packages, objects

library(shiny)
```

```
ui <- pageWithSidebar(
                        titlePanel("Title"),
                        sidebarPanel("Sidebar Panel"),
                        mainPanel("Main Panel")
                      )
```

```
server <- function(input, output) {}
```

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



Slides courtesy of @Chiin-Git (R-Ladies London): http://bit.ly/2xZ47F4

Try building some apps - 3. More basic ui Layout

app.R

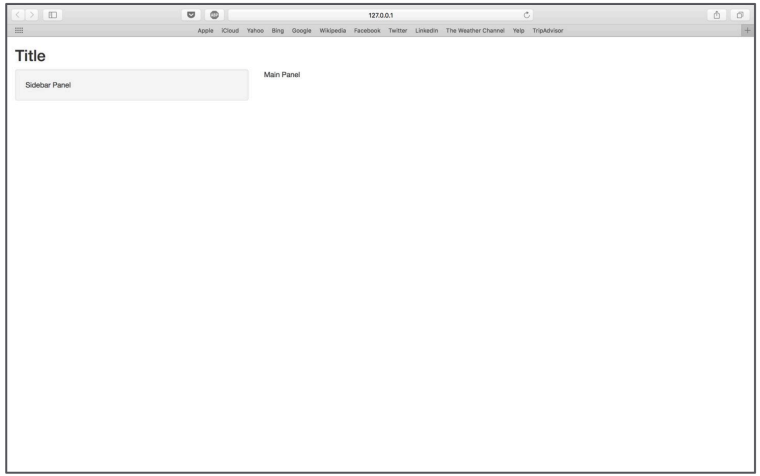
```
#Remember to load & define variables for the global
environment, e.g. packages, objects

library(shiny)
```

```
ui <- fluidPage(
                titlePanel("Title"),
                sidebarLayout(
                        sidebarPanel("Sidebar Panel"),
                        mainPanel("Main Panel")
                        )
                )
```

```
server <- function(input, output) {}
```

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

Try resizing your browser and see how the layout changes

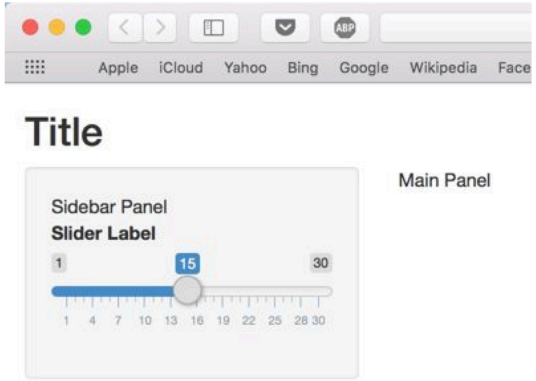


Slides courtesy of @Chiin-Git (R-Ladies London): http://bit.ly/2xZ47F4

Try building some apps - 4. Basic ui Layout with an Input (slider)

app.R

```
#Remember to load & define variables for the global environment, e.g. packages, objects
library(shiny)

ui <- fluidPage(
                titlePanel("Title"),
                sidebarLayout(
                                sidebarPanel("Sidebar Panel",
                                                sliderInput(
                                                                inputId="bins",
                                                                label="Slider Label",
                                                                min=1,
                                                                max=30,
                                                                value=15
                                                                  )
                                                ),
                                mainPanel("Main Panel")
                                )
                )
server <- function(input, output) {}
shinyApp(ui = ui, server = server)
```

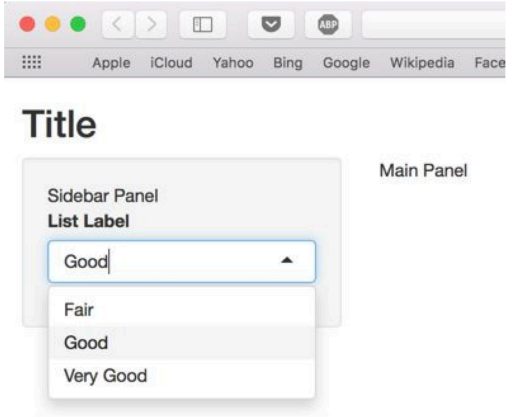


Slides courtesy of @Chiin-Git (R-Ladies London): http://bit.ly/2xZ47F4

Try building some apps - 5. Basic ui Layout with an Input (list select)

app.R

```
#Remember to load & define variables for the global environment, e.g. packages, objects
library(shiny)

ui <- fluidPage(
        titlePanel("Title"),
        sidebarLayout(
                sidebarPanel("Sidebar Panel",
                        selectInput(
                                inputId="list",
                                label="List Label",
                                choices=c("Fair","Good","Very Good"),
                                selected="Good"
                                  )
                                ),
                 mainPanel("Main Panel")
                    )
            )

server <- function(input, output) {}

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

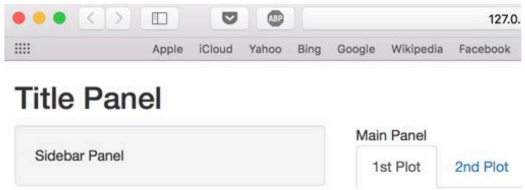


Slides courtesy of @Chiin-Git (R-Ladies London): http://bit.ly/2xZ47F4

Try building some apps - 6. Tab ui Layout with an Output (plot)

app.R

```
#Remember to load & define variables for the global environment, e.g. packages, objects
library(shiny)

ui <- fluidPage(
        titlePanel("Title"),
        sidebarLayout(
            sidebarPanel("Sidebar Panel"),
            mainPanel("Main Panel",
                      tabsetPanel(
                          tabPanel(title="1st Plot", plotOutput(outputId = "plot1")),
                          tabPanel(title = "2nd Plot", plotOutput(outputId = "plot2"))
                                  )
                      )
            )
        )

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

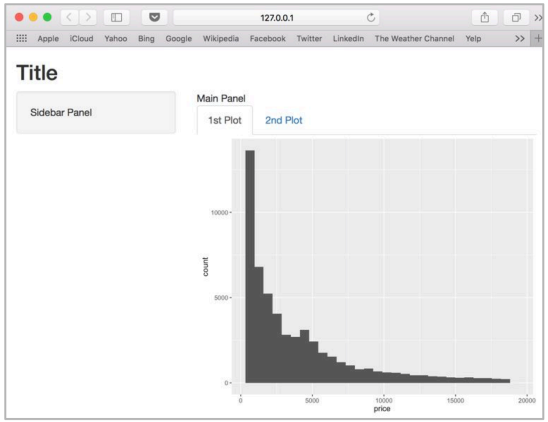


Slides courtesy of @Chiin-Git (R-Ladies London): http://bit.ly/2xZ47F4

Try building some apps - 7. Tab ui Layout rendering 1 Output (plot)

app.R

```
#Remember to load & define variables for the global environment, e.g. packages, objects
library(shiny)
library(ggplot2)

ui <- fluidPage(
        titlePanel("Title"),
        sidebarLayout(
            sidebarPanel("Sidebar Panel"),
            mainPanel("Main Panel",
                      tabsetPanel(
                        tabPanel(title="1st Plot", plotOutput(outputId = "plot1")),
                        tabPanel(title = "2nd Plot", plotOutput(outputId = "plot2"))
                                  )
                      )
                    )
        )

server <- function(input, output) {

                      output$plot1 <- renderPlot({
                        ggplot(data=diamonds, aes(x=price)) + geom_histogram()
                                                 })
                          }

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

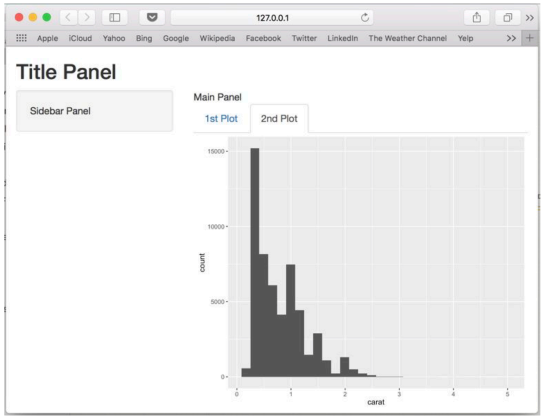


Slides courtesy of @Chiin-Git (R-Ladies London): http://bit.ly/2xZ47F4

Try building some apps - 8. Tab ui Layout rendering 2 Outputs (plot)

app.R

```
#Remember to load & define variables for the global environment, e.g. packages, objects
library(shiny)
library(ggplot2)

ui <- fluidPage(
        titlePanel("Title"),
        sidebarLayout(
                sidebarPanel("Sidebar Panel"),
                mainPanel("Main Panel",
                        tabsetPanel(
                           tabPanel(title="1st Plot", plotOutput(outputId = "plot1")),
                           tabPanel(title = "2nd Plot", plotOutput(outputId = "plot2"))
                                )
                        )
                )
        )

server <- function(input, output) {

                output$plot1 <- renderPlot({
                   ggplot(data=diamonds, aes(x=price)) + geom_histogram()
                                })

                output$plot2 <- renderPlot({
                   ggplot(data=diamonds, aes(x=carat)) + geom_histogram()
                                })
                        }

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

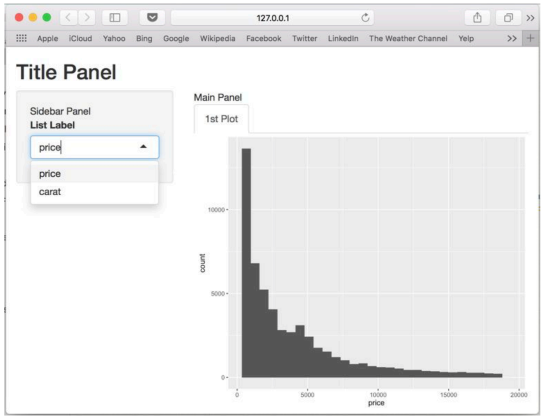


Slides courtesy of @Chiin-Git (R-Ladies London): http://bit.ly/2xZ47F4

Try building some apps - 9. Reactive ui, single Input-Output (list select & plot)

app.R

```
#Remember to load & define variables for the global environment, e.g. packages, objects
library(shiny)
library(ggplot2)

ui <- fluidPage(
        titlePanel("Title"),
        sidebarLayout(
            sidebarPanel("Sidebar Panel",
                         selectInput(
                             inputId="list",
                             label="List Label",
                             choices=c("price", "carat"),
                             selected="price"
                                 ),
                         ),
            mainPanel("Main Panel",
                      tabsetPanel(
                          tabPanel(title="1st Plot", plotOutput(outputId = "plot1"))
                                  )
                      )
                  )
        )

server <- function(input, output) {

                      output$plot1 <- renderPlot({
                          ggplot(data=diamonds, aes_string(x=input$list)) + geom_histogram()
                                                 })
                  }
shinyApp(ui = ui, server = server)
```



Slides courtesy of @Chiin-Git (R-Ladies London): http://bit.ly/2xZ47F4

Now Try the Example Use-Case: CODE

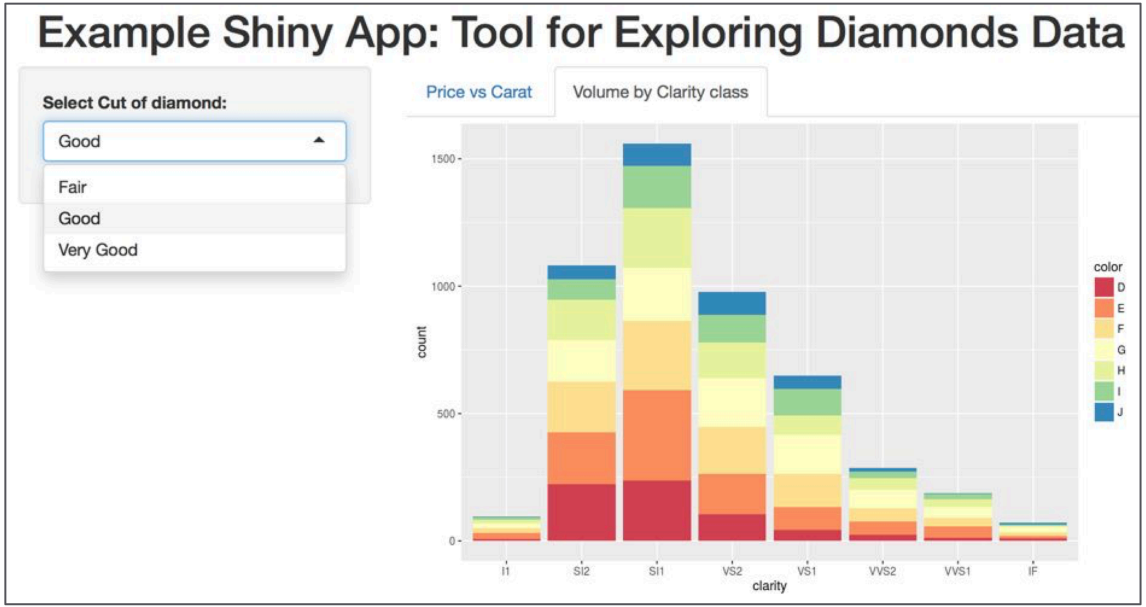
```
library(shiny)
library(ggplot2)

ui <- fluidPage(
                titlePanel("Example Shiny App: Distribution of Diamond stock by Price Bands"),
                sidebarLayout(
                                sidebarPanel(
                                sliderInput(inputId="bands",
                                                label="Select No. of Price Bands",
                                                min=1,
                                                max=30,
                                                value=15)
                                ),
                mainPanel(
                                plotOutput(outputId="plot")
                                )
                                )
        )

server <- function(input, output){

                                output$plot <- renderPlot({
                                                                ggplot(data=diamonds, aes(x=price)) + geom_histogram() + stat_bin(bins=input$bands)
                                                                })
                                }

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

Slides courtesy of @Chiin-Git (R-Ladies London): http://bit.ly/2xZ47F4

Replicate Chiin's Explorer Tool

You can deploy a limited no./usage of your apps for free on shinyapps.io (like I've done here to demonstrate), which is a freemium hosted version of Shiny Server

chiin.shinyapps.io/diamonds/



Slides courtesy of @Chiin-Git (R-Ladies London): http://bit.ly/2xZ47F4

Replicate Chiin's Explorer Tool: CODE

```
library(shiny)
library(ggplot2)
library(RColorBrewer)

ui <- fluidPage(
        titlePanel("Example Shiny App: Tool for Exploring Diamond Data"),
        sidebarLayout(
                sidebarPanel(
                        selectInput(inputId = "cut",
                                label = "Select Cut of Diamond:",
                                choices = c("Fair", "Good", "Very Good"),
                                selected = "Good")
                                 ),
                mainPanel(
                        tabsetPanel(
                                tabPanel("Price vs Carat",
                                        plotOutput(outputId = "scatter")
                                        ),
                        tabPanel("Volume by Clarity class",
                                plotOutput(outputId = "bar")
                                )
                                )
                        )
                )
        )

server <- function(input, output) {

    getDataset <- reactive({
                if (input$cut=="Fair") {
                return(diamonds[diamonds$cut=="Fair", ])
                } else if (input$cut=="Good") {
                return(diamonds[diamonds$cut=="Good", ])
                } else {
                return(diamonds[diamonds$cut=="Very Good", ])
                }
                        })

    output$scatter <- renderPlot({
                        ggplot(data=getDataset(), aes(x=price, y=carat)) + geom_point(aes(colour=color)) + scale_color_brewer(palette="Blues")
                        })

    output$bar <- renderPlot({
                        ggplot(data=getDataset(), aes(x=clarity)) + geom_bar(aes(fill=color)) + scale_fill_brewer(palette="Spectral")
                        })
                        }

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

Slides courtesy of @Chiin-Git (R-Ladies London): http://bit.ly/2xZ47F4




We're Daloha, Dan, Maggie & Pam

We recently joined forces to make **RLadies Munich** happen, but we'd be happy to have you join us on Meetup, Instagram, Twitter, Facebook and Slack