Lab 04 - EDA for Multiple Variables

Learning outcomes

In this lab you will learn and practise

- Exploratory data analysis for two or more variables
- The layered grammar of ggplot
- Build an interactive web application with Shiny

Consider to produce a R HTML Notebook using R Markdown to document your learning and observation for this lab. Make sure you practise the use of different heading levels, hyperlinks, bullet points (e.g. unordered and ordered lists) and insertion of images in R Markdown.

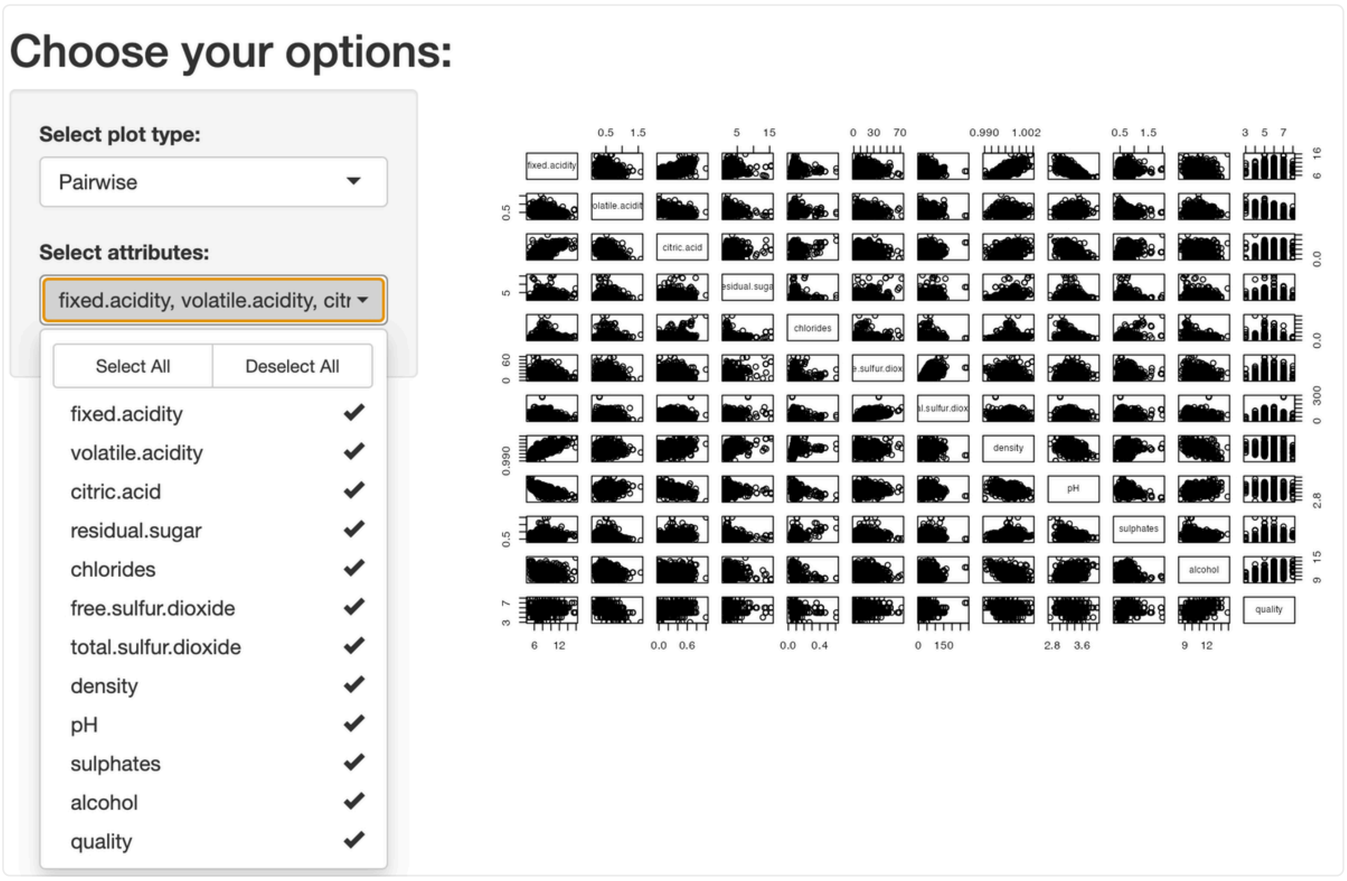
Grid of pair-wise scatter plots

Let's continue analysing the *red wine* data from last week, we read the data into a data frame using the following code:

```
a <- read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-
```

or download the `winequality-red.csv` file and read it from your local disk.

Exercise 1: Create a web application to produce the pairwise comparison with selected variables. An example interface is provided below for reference, but you are **not required** to follow the design.


Choose your options:

Select plot type:

Pairwise

Select attributes:

fixed.acidity, volatile.acidity, citr

Select All | Deselect All

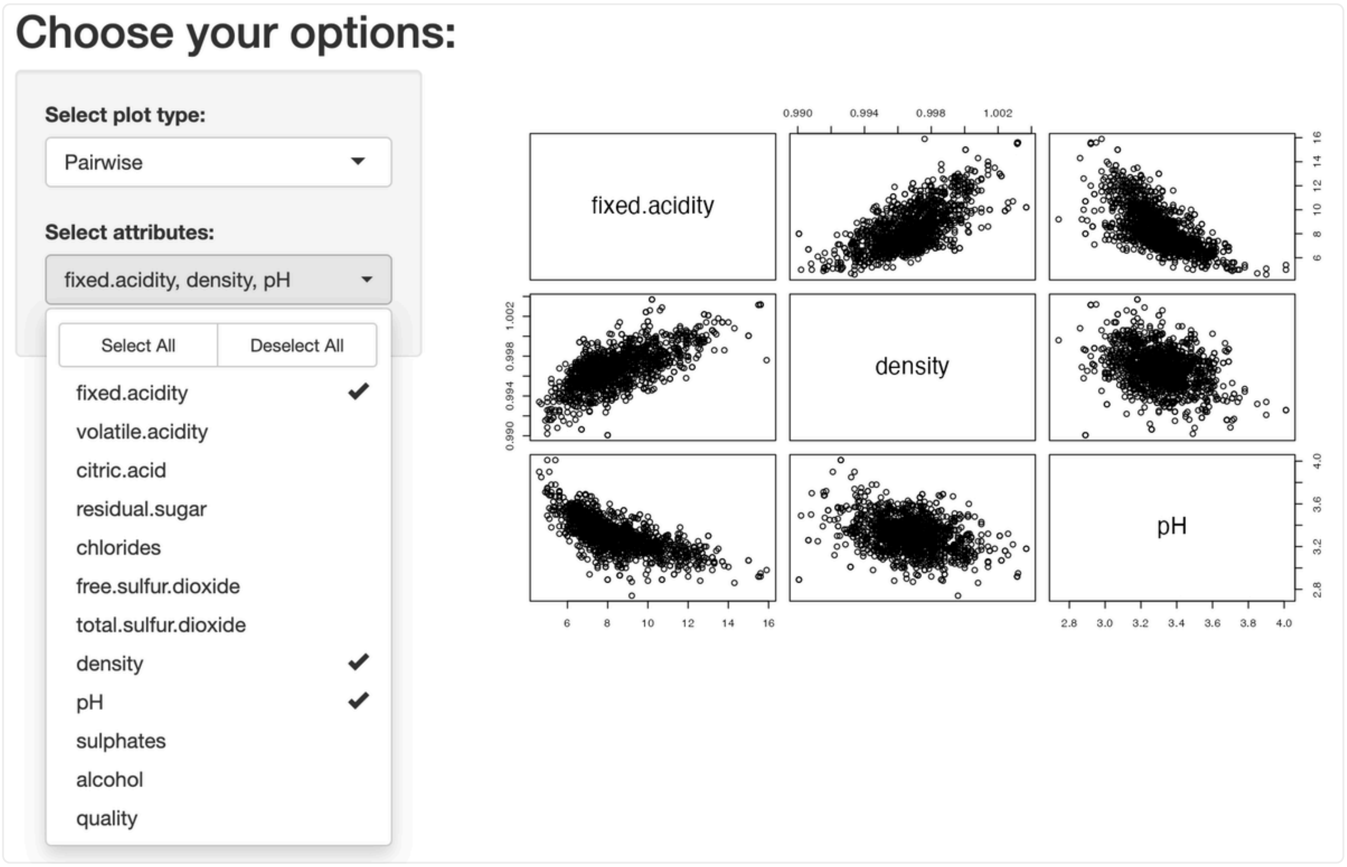
fixed.acidity ✓
volatile.acidity ✓
citric.acid ✓
residual.sugar ✓
chlorides ✓
free.sulfur.dioxide ✓
total.sulfur.dioxide ✓
density ✓
pH ✓
sulphates ✓
alcohol ✓
quality ✓

Hint: You might need `conditionalPanel` inside `sidebarPanel`. In `renderPlot`, use `if-else if` to check the chosen plot type:

```
sidebarPanel(
      selectInput("plot_type", "Select plot type:", choices = c("Pairwise")),
      conditionalPanel(
        condition = "input.plot_type == 'Pairwise'",
        pickerInput(
          inputId = "selected_attributes",
          label = "Select attributes:",
          choices = names(a),
          multiple = TRUE,
          options = list(`actions-box` = TRUE)
        )
      )
),
mainPanel(
      plotOutput("plot",height = "400px", width = "600px")
    )
```

```
# in the server:
output$plot <- renderPlot({
    if (input$plot_type == "Pairwise") {
      selected_attrs <- input$selected_attributes
      if (length(selected_attrs) < 2) {
        return(NULL)
      }
      pairs(a[, selected_attrs])
    }
})
```

Which pairs of variables seem to have clear relations? Are they linear, polynomial, exponential relations, or cluster-like? For a selected set of pairs that are of interest, we can produce a plot like below:


Choose your options:

Select plot type: Pairwise

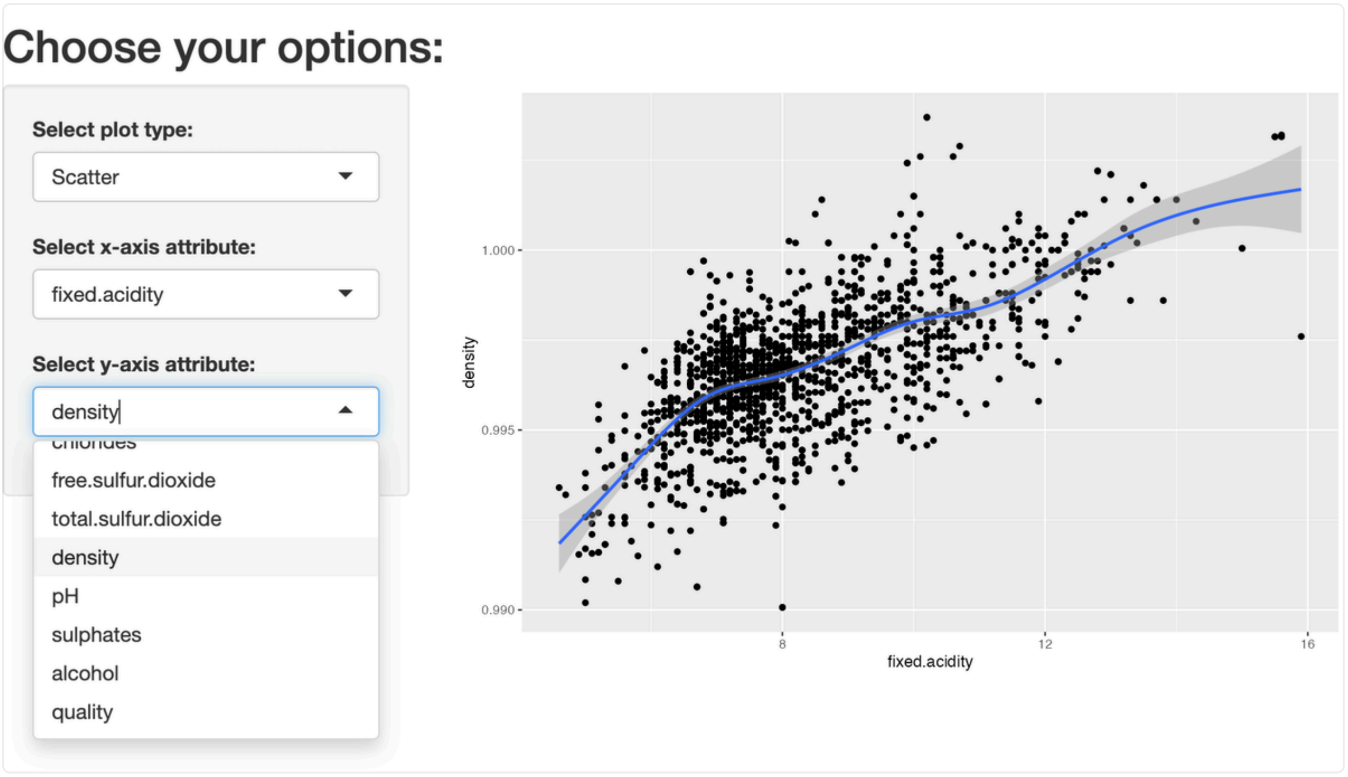
Select attributes: fixed.acidity, density, pH

Plots for two continuous variables

Scatter plots

From the pair-wise grid, there seems to be linear relation between `fixed.acidity` and `density`, let's plot the scatter plot and fit a smooth line through it.

Exercise 2: Modify your web application to produce the two plots below:



Choose your options:

Select plot type:

Scatter

Select x-axis attribute:

fixed.acidity

Select y-axis attribute:

density

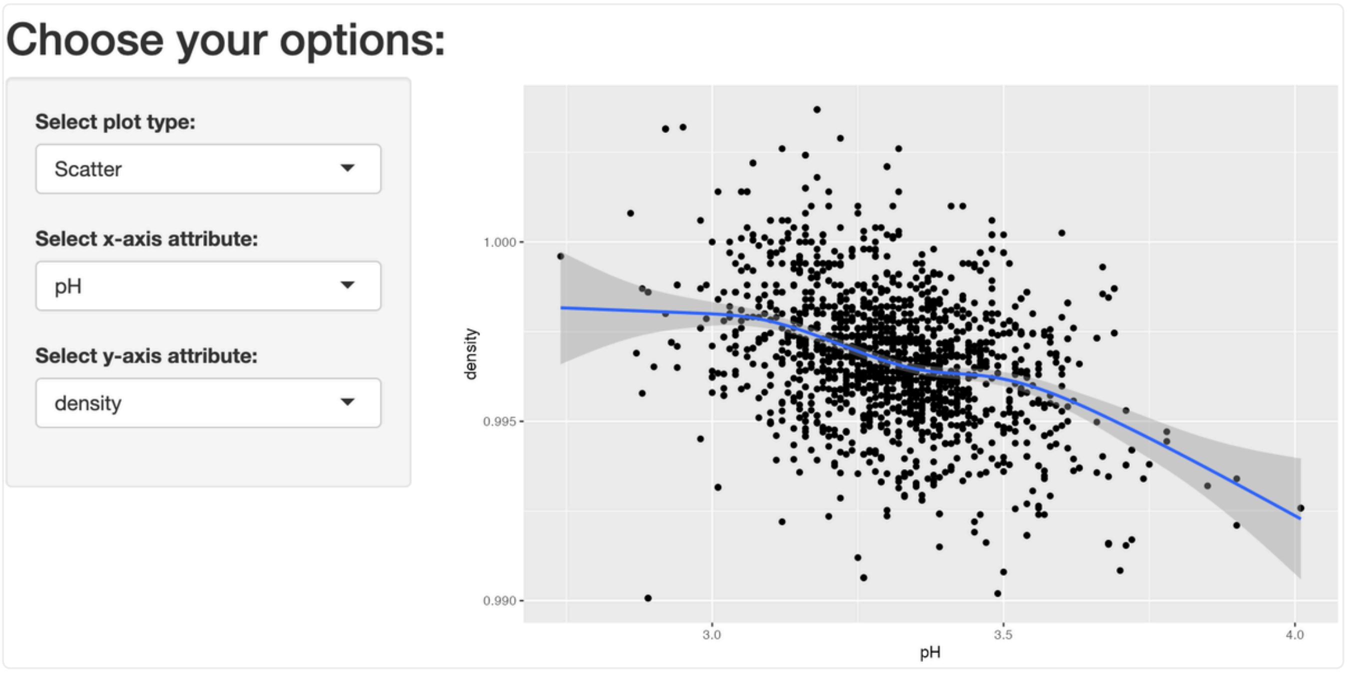
Hint: add one more option in `choices` and one more `conditionalPanel` in `sidebarPanel:`

```
sidebarPanel(
      selectInput("plot_type", "Select plot type:", choices = c("Scatter", "Pa
      conditionalPanel(
        condition = "input.plot_type == 'Pairwise'",
        pickerInput(
          inputId = "selected_attributes",
          label = "Select attributes:",
          choices = names(a),
          multiple = TRUE,
          options = list(`actions-box` = TRUE)
        )
      ),
      conditionalPanel(
        condition = "input.plot_type == 'Scatter'",
        selectInput("x_attr", "Select x-axis attribute:", names(a)),
        selectInput("y_attr", "Select y-axis attribute:", names(a))
      )
),
mainPanel(
      plotOutput("plot",height = "400px", width = "600px")
    )
```

```r
# in the server:
output$plot <- renderPlot({
    if (input$plot_type == "Pairwise") {
      selected_attrs <- input$selected_attributes
      if (length(selected_attrs) < 2) {
        return(NULL)
      }
      pairs(a[, selected_attrs])
    } else if(input$plot_type == "Scatter") {
      x_attr <- input$x_attr
      y_attr <- input$y_attr
      ggplot(a, aes_string(x = x_attr, y = y_attr)) +
        geom_point()+geom_smooth()
    }

})
```

What about `pH` and `density`?



Choose your options:

Select plot type: Scatter

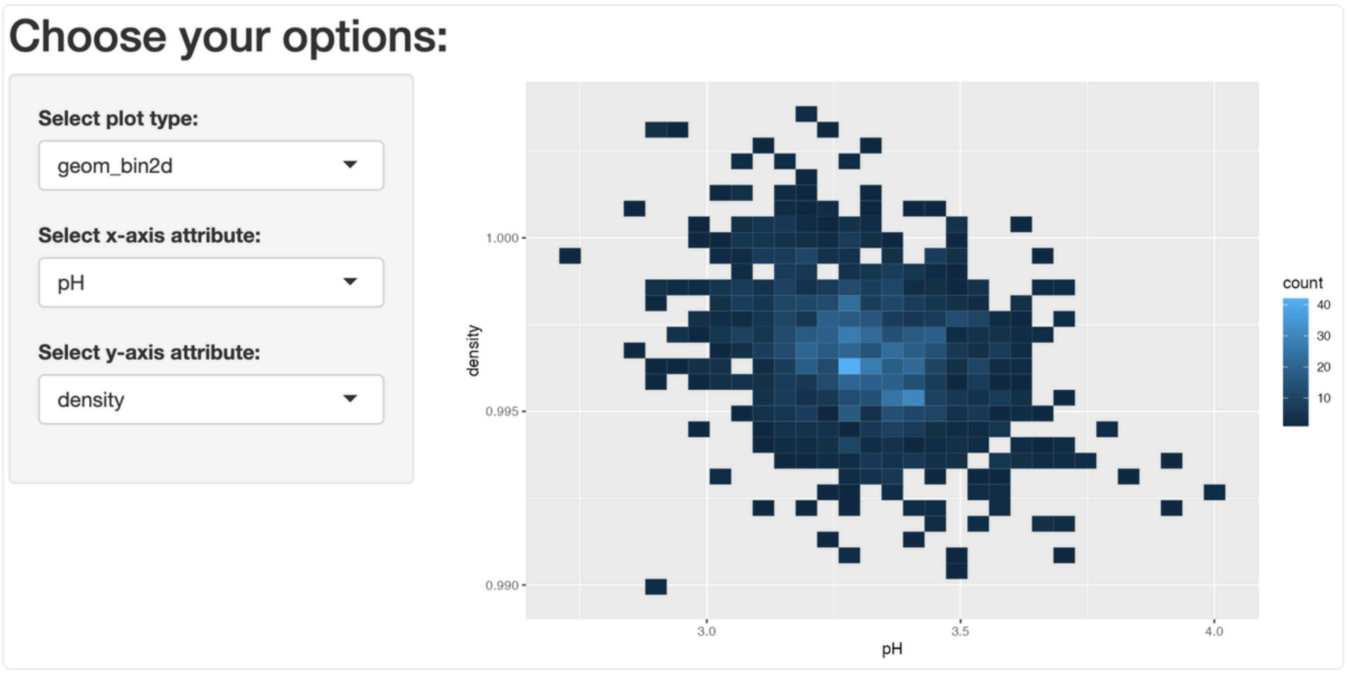
Select x-axis attribute: pH

Select y-axis attribute: density

Two dimensional binning (`geom_bin2d` and `geom_hex`)

`ggplot` has a geom `geom_bin2d` for drawing 2D histograms with binning on both `x` and `y` axes, and the counts (frequencies) are displayed as a varied colour intensity for each polygon-shaped bin.

Exercise 3: Modify your web application to produce the 2D histogram below:



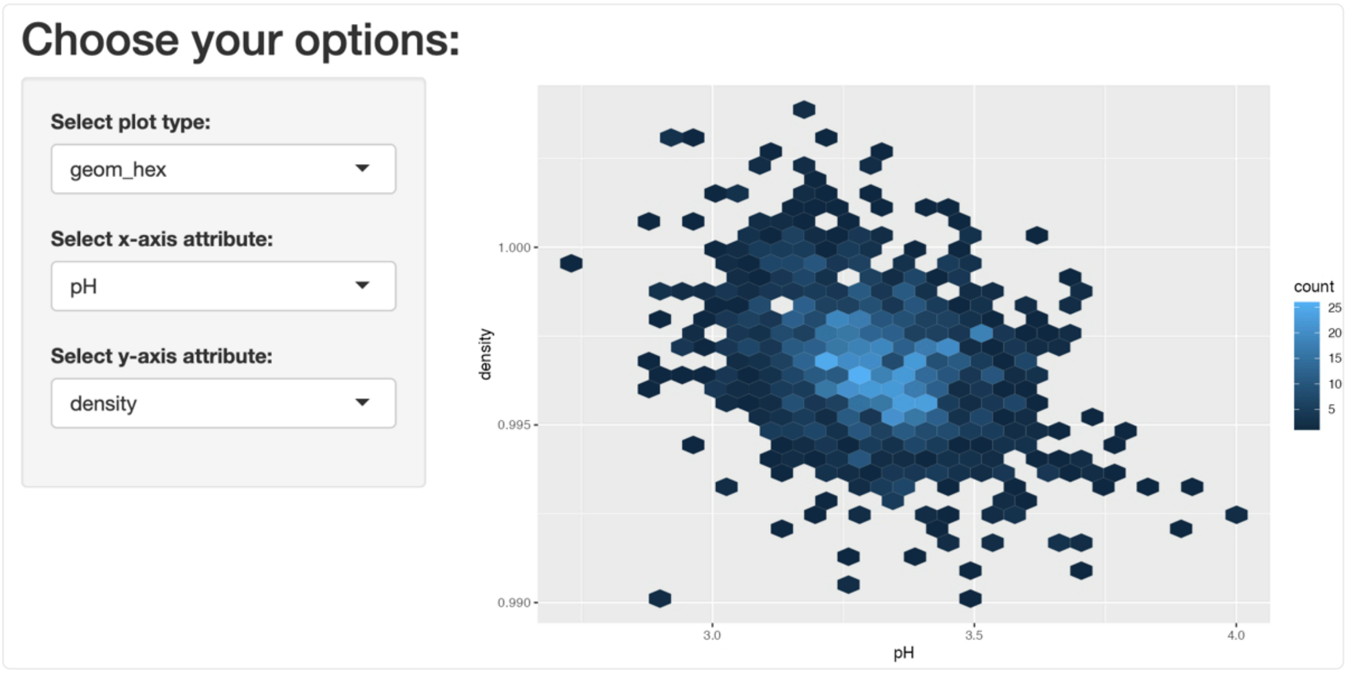
```
sidebarPanel(
      selectInput("plot_type", "Select plot type:", choices = c("Scatter", "Pa
      conditionalPanel(
        condition = "input.plot_type == 'Pairwise'",
        pickerInput(
          inputId = "selected_attributes",
          label = "Select attributes:",
          choices = names(a),
          multiple = TRUE,
          options = list(`actions-box` = TRUE)
        )
      ),
      conditionalPanel(
        condition = "input.plot_type == 'Scatter'|| input.plot_type == 'geom_b
        selectInput("x_attr", "Select x-axis attribute:", names(a)),
        selectInput("y_attr", "Select y-axis attribute:", names(a))
      )
),
mainPanel(
      plotOutput("plot",height = "400px", width = "600px")
    )
```

```
# in the server:
output$plot <- renderPlot({
    if (input$plot_type == "Pairwise") {
      selected_attrs <- input$selected_attributes
      if (length(selected_attrs) < 2) {
        return(NULL)
      }
      pairs(a[, selected_attrs])
    } else if(input$plot_type == "Scatter") {
      x_attr <- input$x_attr
      y_attr <- input$y_attr
      ggplot(a, aes_string(x = x_attr, y = y_attr)) +
        geom_point()+geom_smooth()
    } else if (input$plot_type == "geom_bin2d"){
      x_attr <- input$x_attr
      y_attr <- input$y_attr
      ggplot(a, aes_string(x=x_attr, y=y_attr) ) +
        geom_bin2d()
    }

})
```

You can use the more aesthetic appealing `hexbin` for displaying the same information.

Exercise 4: Modify your web application to produce the following hexbin plot:


Choose your options:

Select plot type: geom_hex

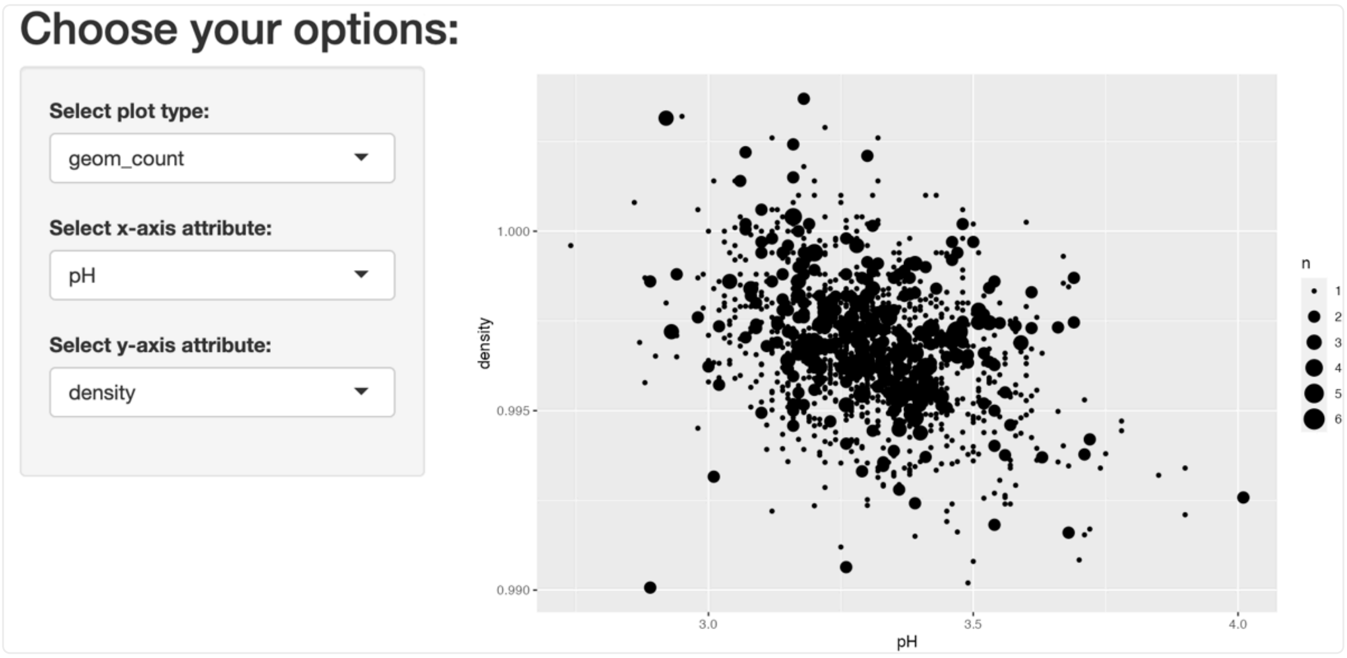
Select x-axis attribute: pH

Select y-axis attribute: density

geom_count() for two continuous variables

`geom_count()` works for both continuous variables and categorical variables. Let's try this plot for `density` against `pH`.

Exercise 5: Modify your code to generate the following plot:


Choose your options:

Select plot type: geom_count

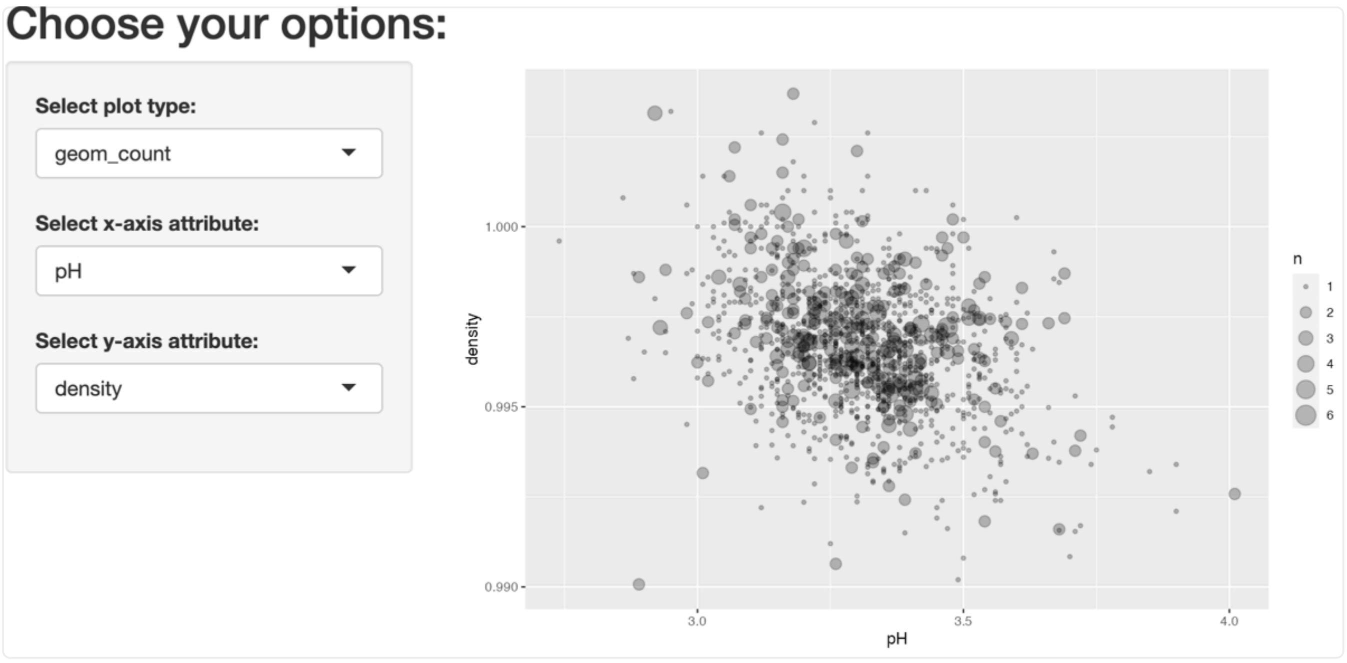
Select x-axis attribute: pH

Select y-axis attribute: density

The sizes of the dots correspond to the counts (i.e., frequencies) of co-occurrences of the values of the two variables `pH` and `density`. The counts are shown as letter `n` in the legend.

We can see that the dots are super-imposed on top of each other in the diagram above, making it hard to visualize dots that are being covered. This problem can be overcome by using transparency (i.e., using the `alpha` aesthetic). Below is the `geom_count` plot with `alpha = 0.25`.

Exercise 6: Modify your `geom_count` plot with `alpha = 0.25`. It would be better if your application enables the user to enter the alpha value.


Choose your options:

Select plot type: geom_count

Select x-axis attribute: pH

Select y-axis attribute: density

Using categorical variables for aesthetic mapping

As we have seen in the lecture slides, we can add categorical variables onto a two variable plots through aesthetic mapping, such as `colour`, `size` or `shape`.

For the exercises below, we firstly turn the `a$quality` variable into categorical using the `factor` function:

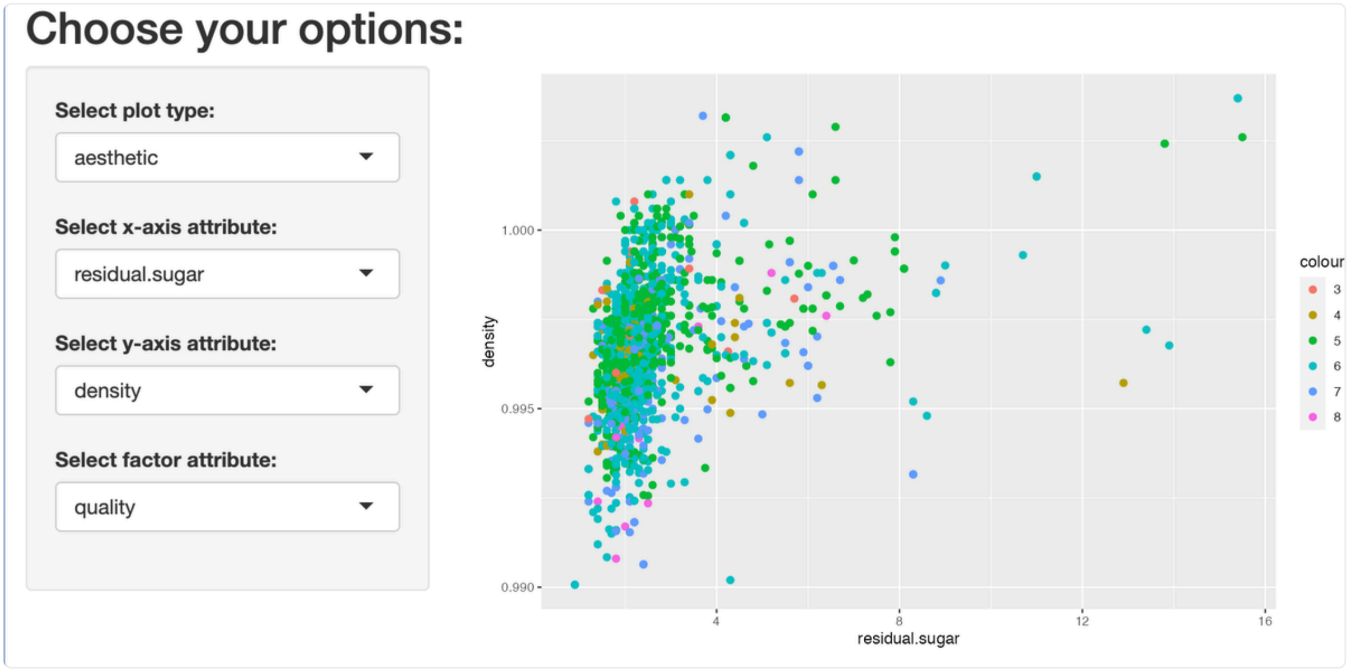
```
a$quality <- factor(a$quality)
```

We then generate various scatter plots between the two variables `residual.sugar` and `density` factored by the categorical variable `quality` that we have modified.

Use the `colour` aesthetic

In the sample code below, we use the `colour` aesthetics. We also use `size` to give a slight increase to the marker size of all the points.

```
ggplot(data=a, mapping = aes(x=residual.sugar, y=density)) +
  geom_point(mapping = aes(colour = quality), size=2.0)
```


Choose your options:

Select plot type: aesthetic

Select x-axis attribute: residual.sugar

Select y-axis attribute: density

Select factor attribute: quality

The plot above shows that high quality red wine (pink dots for `quality=8`) seems to mostly have less residual sugar. This is an interesting finding that is difficult to detect from inspecting the numerical values in the dataset.

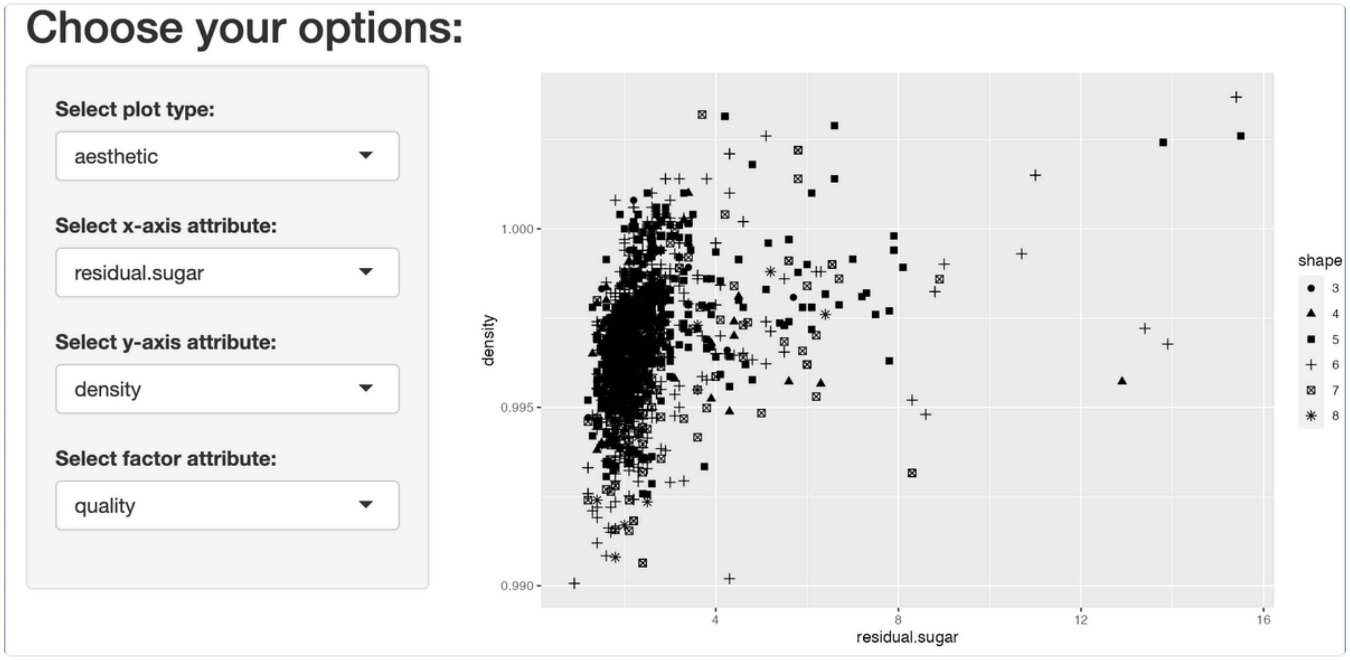
We can also put the `colour` aesthetic together in the `aes` mapping in `ggplot`, as shown in the code below:

```
ggplot(data=a, mapping = aes(x=residual.sugar, y=density, colour=quality)) +
  geom_point()
```

Run the code above to confirm that the plot is identical.

Use the `shape` aesthetic

Exercise 7: Now try using the `shape` aesthetic to generate the plot below:



Choose your options:

Select plot type: aesthetic

Select x-axis attribute: residual.sugar

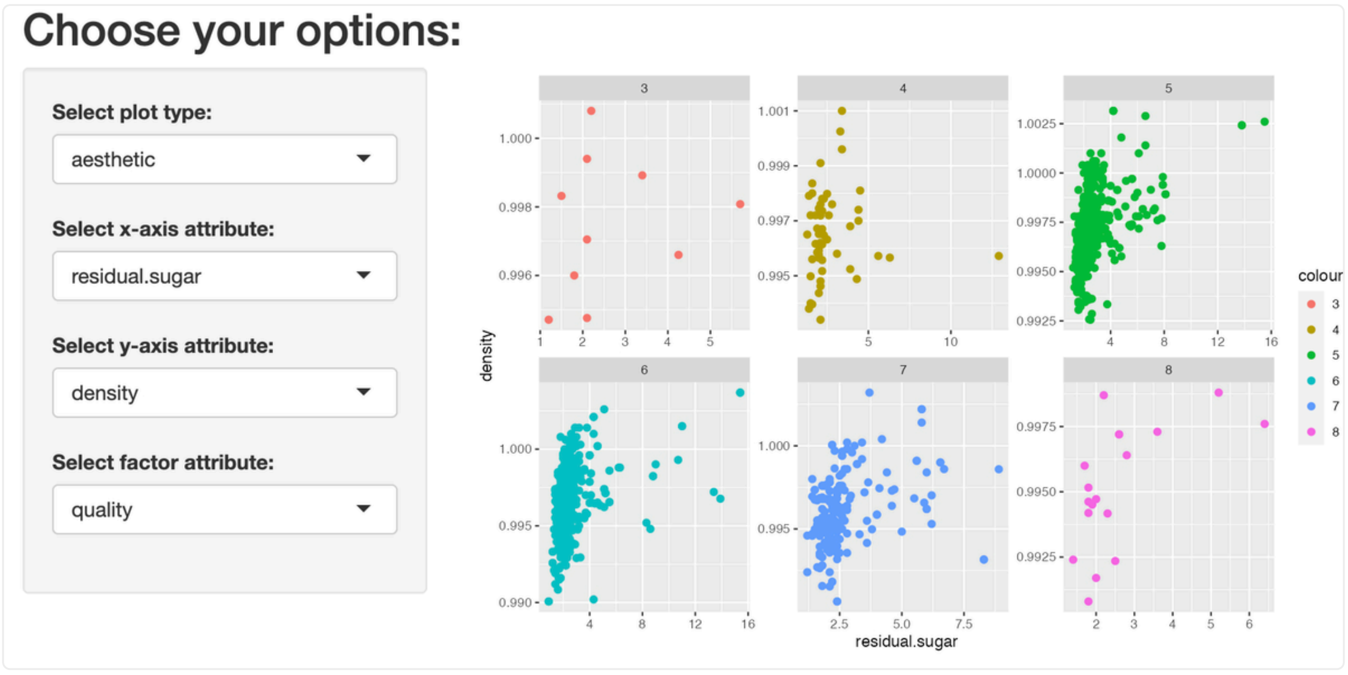
Select y-axis attribute: density

Select factor attribute: quality

Use facet_wrap()

Inspect the variable `quality` and find out how many levels there are. Rather than overlaying the points for different quality values into a single diagram, we can use `facet_wrap` to separate them.

Exercise 8: Use `facet_wrap` to generate the plot below:



Choose your options:

Select plot type: aesthetic

Select x-axis attribute: residual.sugar

Select y-axis attribute: density

Select factor attribute: quality

What intuition can you draw from this facet grid? (High quality wine seems to have both lower density and residual sugar? Perhaps more data points for wine at quality 8 would be useful to see the variability)

Plotting two categorical variables

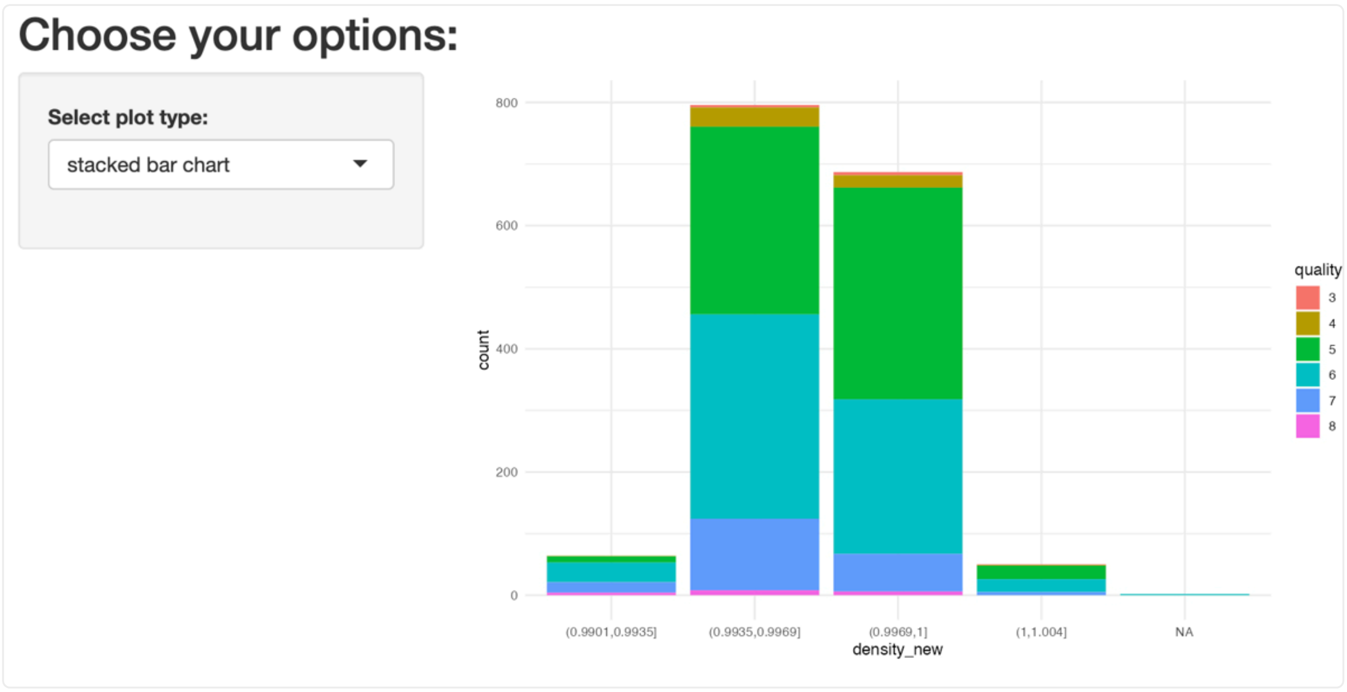
In the wine dataset, we only have one categorical variable, we can use binning to turn a continuous variable into discrete. For example, to convert the continuous variable `a$density` to form a new categorical variable `a$density_new` having 4 levels, we can do the following:

```
a$density_new <-
  cut(a$density, breaks=c(seq(from = min(a$density), to = max(a$density), leng
```

Together with the `quality` variable that we have factored above, we now have two categorical variables for plotting the various charts below.

Produce a stacked bar chart

Exercise 9: Produce a stacked bar chart as shown in the diagram below. It would be better if your application allows the user to select x-axis variable.


Choose your options:

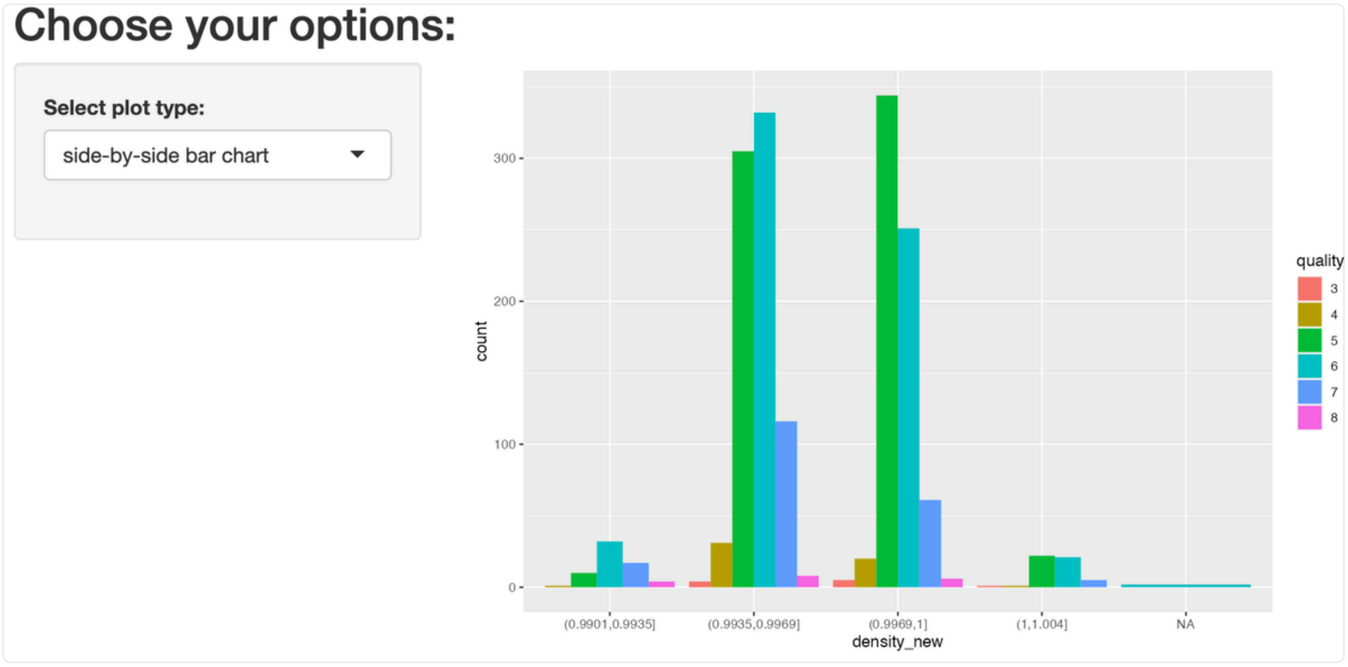
Select plot type:

stacked bar chart

This stacked bar chart certainly makes it easy to visualise the distribution of wine quality for different ranges of density values.

Produce a side-by-side bar chart

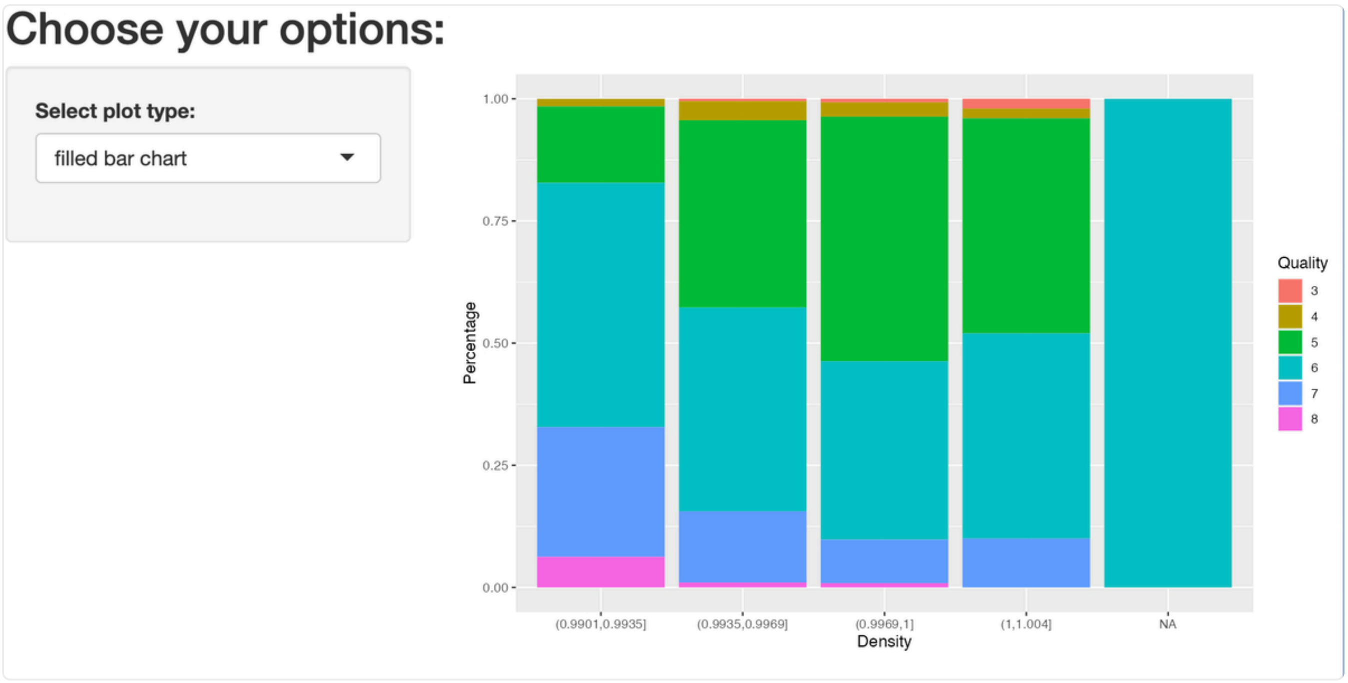
Exercise 10: Produce all of the following plots. It would be better if your application allows the user to select x-axis variable.


Choose your options:

Select plot type:

side-by-side bar chart

Produce a filled bar chart

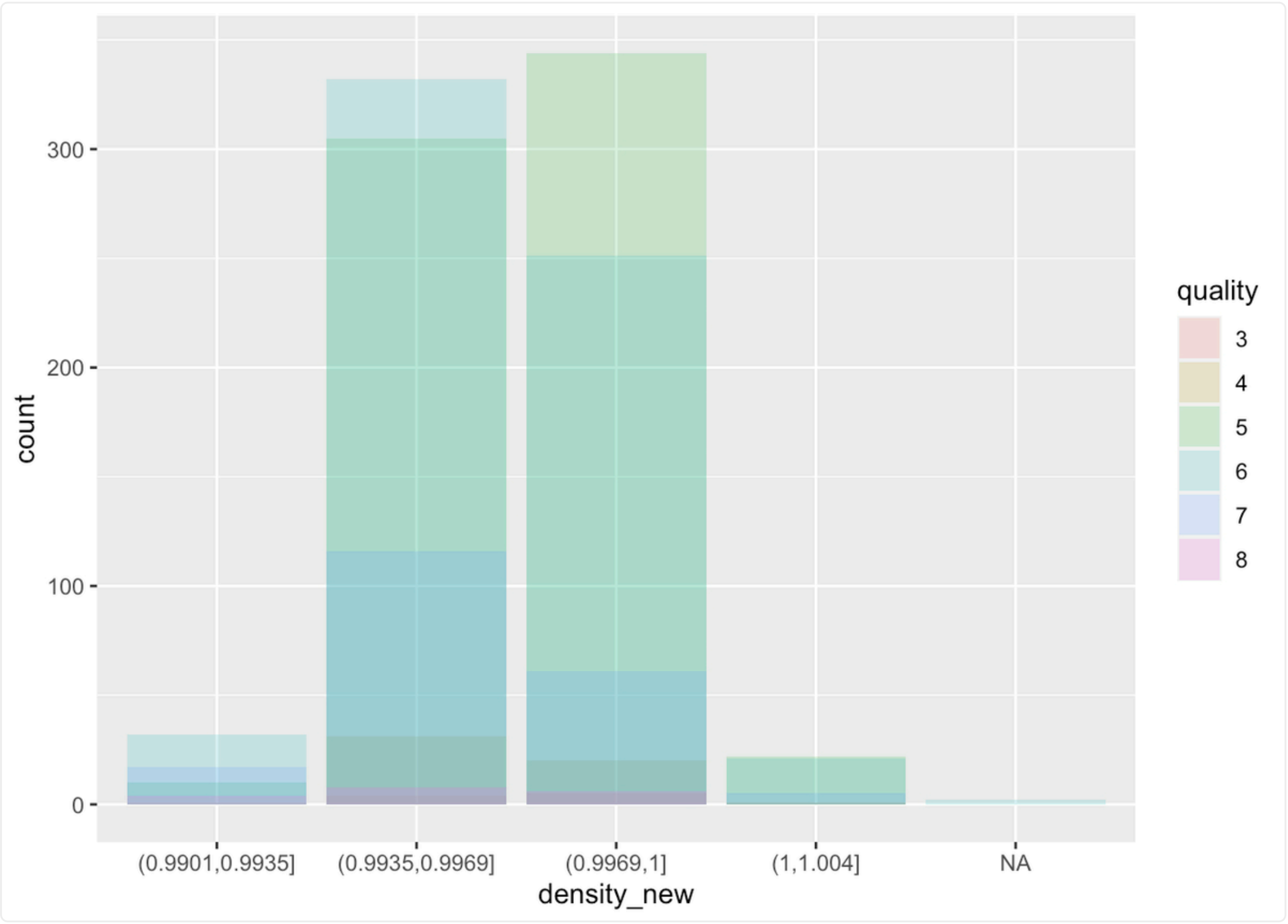

Choose your options:

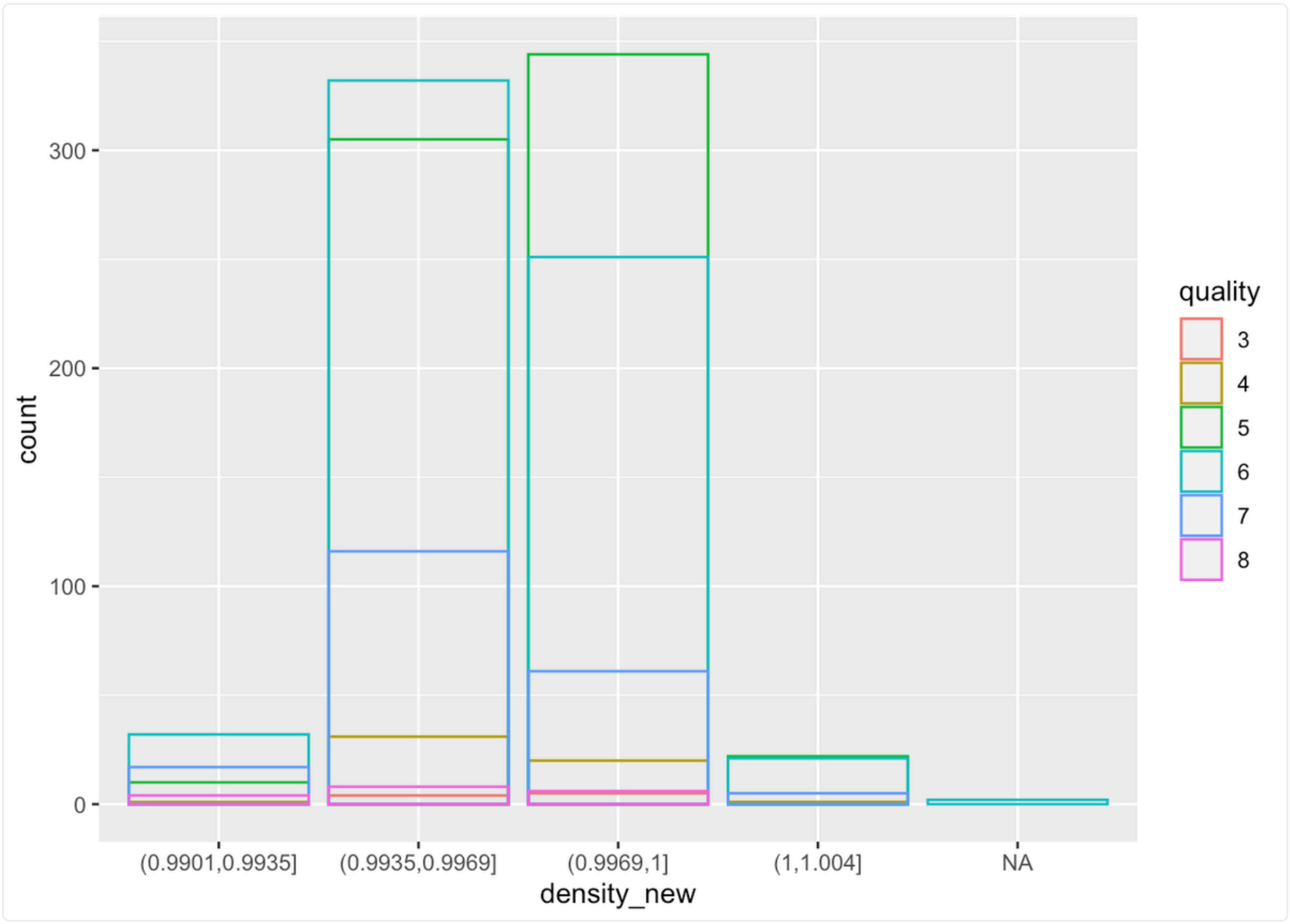
Select plot type:

filled bar chart

Produce an overlaying bar chart

Note that in the first plot we use the `fill` aesthetic for the secondary variable, whereas in the second plot we use the `colour` aesthetic for the secondary variable.





Continue learning on `swirl`

Complete other lessons from Exploratory Data Analysis on swirl(). We have briefly covered `apply()` in week 2's lecture. In particular, please look at lessons 13-15 (*Simulation*, *Dates and Times*, and *Basic Graphics*) if you haven't done so. These lessons should be quite easy if you have already attempted the lab exercises for weeks 2-4.

Preparation for your project

For any dataset, follow the iterative process of EDA (for both variation and co-variation type of questions)

- asking questions
- visualisation to obtain answers
- refining the questions
- further visualisation for answers

Document the process with justification of the choice of plot types, the resulting plots and observations in an R HTML notebook.

Previous
Shiny Apps in .Rmd File - Lab Shiny App Solution

Next
Lab 05 - Reading Files and Web Scrapping using R

Last updated 1 month ago