Practical 9: Visualizing data

ADS2

2023-11-16, A331, A332

Work through this guide alone or in groups. Facilitators are here to help. The time it takes to complete this practical can vary between individuals – this is OK. Do not worry if you do not finish within the session.

Learning Objectives

- Use ggplot2 for data visualization
- Think critically about data visualization choices

1. Overplotting

In the lecture, you have learnt how to use ggplot2 to generate a scatter plot from the dataset `diamonds`. Please repeat it and generate the following plot. You can get the dataset by `data(diamonds)`.

```
# Make the basic plot

g <- ggplot(data = diamonds,
            mapping = aes(x = carat, y = price, group = cut))
g1 <- g + geom_point(stat = "identity", aes(colour = cut))
g1
```

This plot is not very illustrative due to numerous plots that overlap with each other. You can resize the point size to 0.1 or change the transparency to alpha 1/5 to make it look better.

```
# Change the size and transparency of the dots

g + geom_point(position = "identity",
               aes(colour = cut),
               size = 0.1,
               alpha = 0.2)
```

2. Each observation is presented as a round dot. You can assign another shape by the `shape` argument to another number, such as 18. Not to say that it will help with visualization, but for the sake of practice, it is fine.

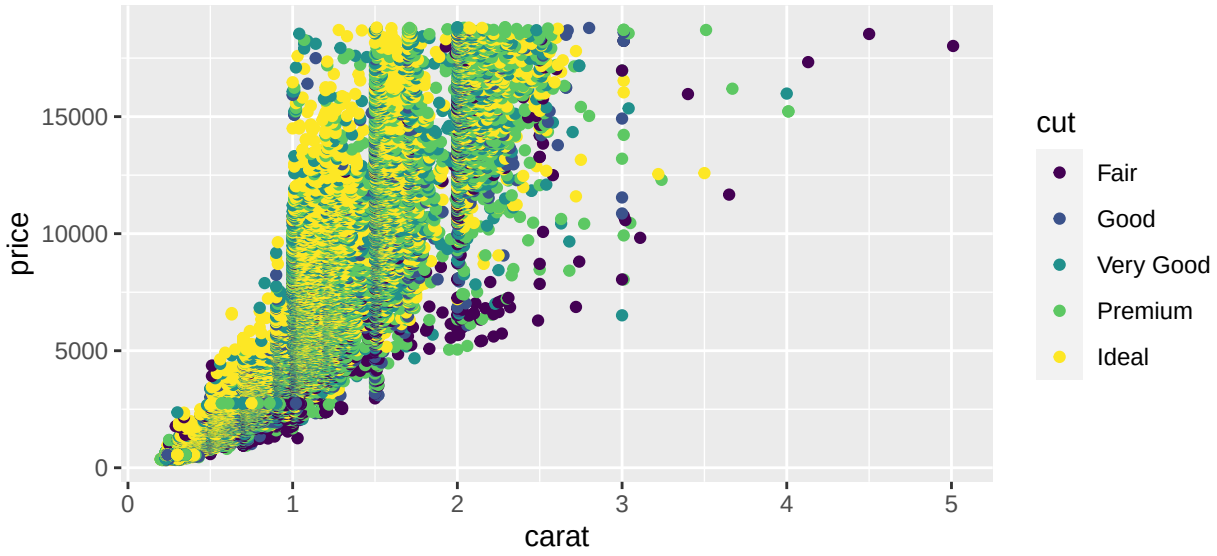


Figure 1: Making a basic scatter plot

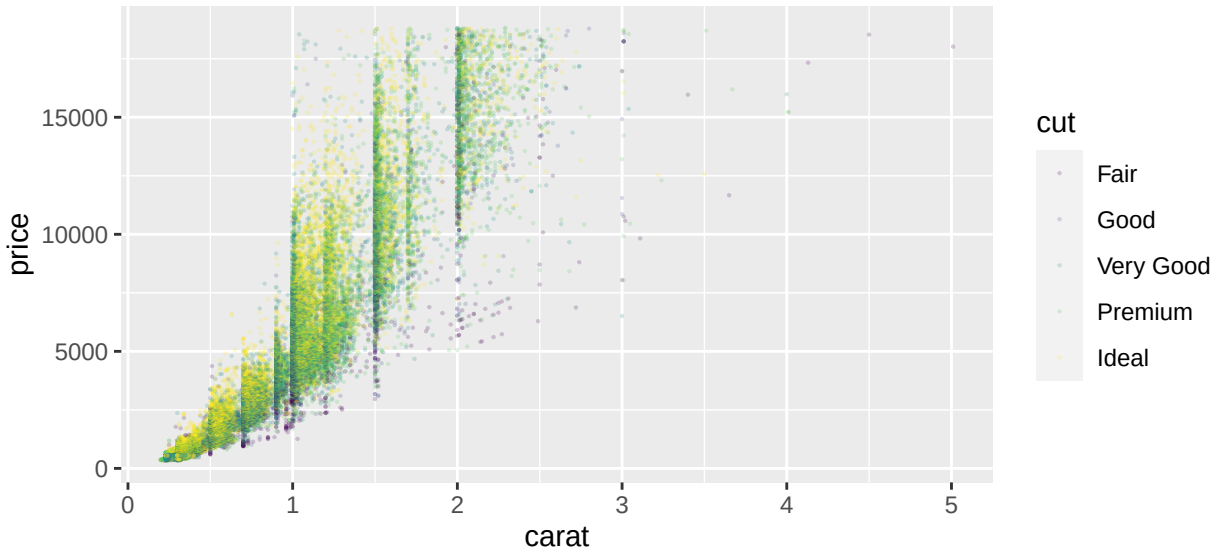


Figure 2: Changing the parameter of the dots

```r
# Change the shape of the dots

g + geom_point(
    position = "jitter",
    aes(colour = cut),
    size = 0.6,
    alpha = 0.4,
    shape = 18
  ) +
  guides(color = guide_legend(override.aes = list(size = 6, alpha = 0.5)))
```

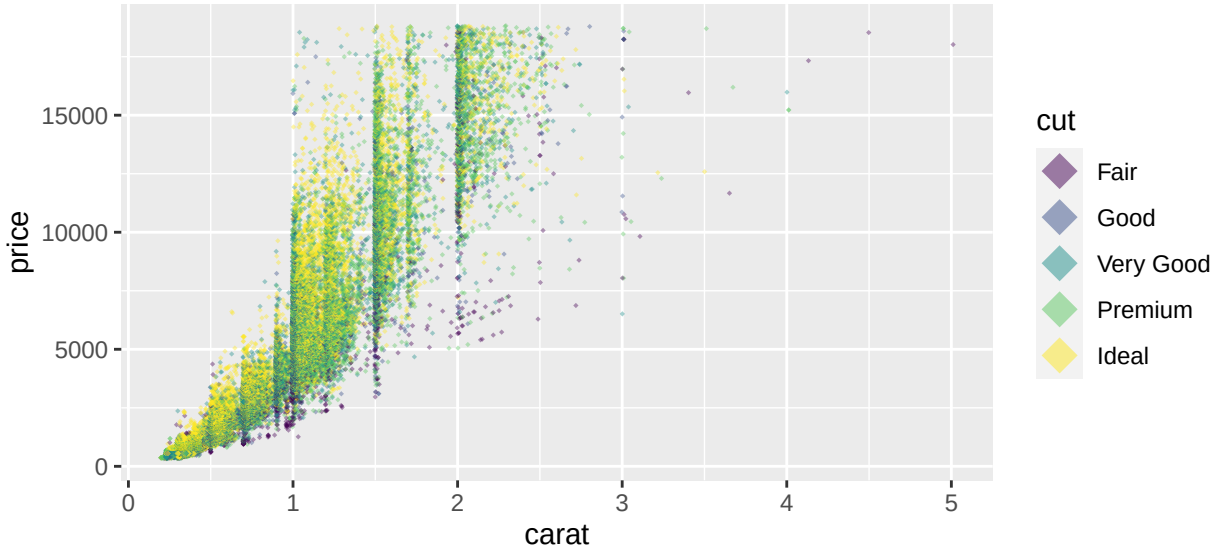


Figure 3: Changing the shape of the dots

2. Rewrite the code

Again, each `geom_XXXX` is just a shortcut for `layer()` with some value prespecified. It is absolutely fine to use `stat_XXXX` if the situation requires. You will just need to specify the correct `geom` and `position` arguments.

```r
gg1 <-
  ggplot(data = diamonds,
         mapping = aes(x = carat, y = price, group = cut))
g2 <- gg1 + stat_identity(
  mapping = aes(color = cut),
  size = 0.6,
  geom = "point",
  shape = 3,
  position = "jitter"
)
g2
```

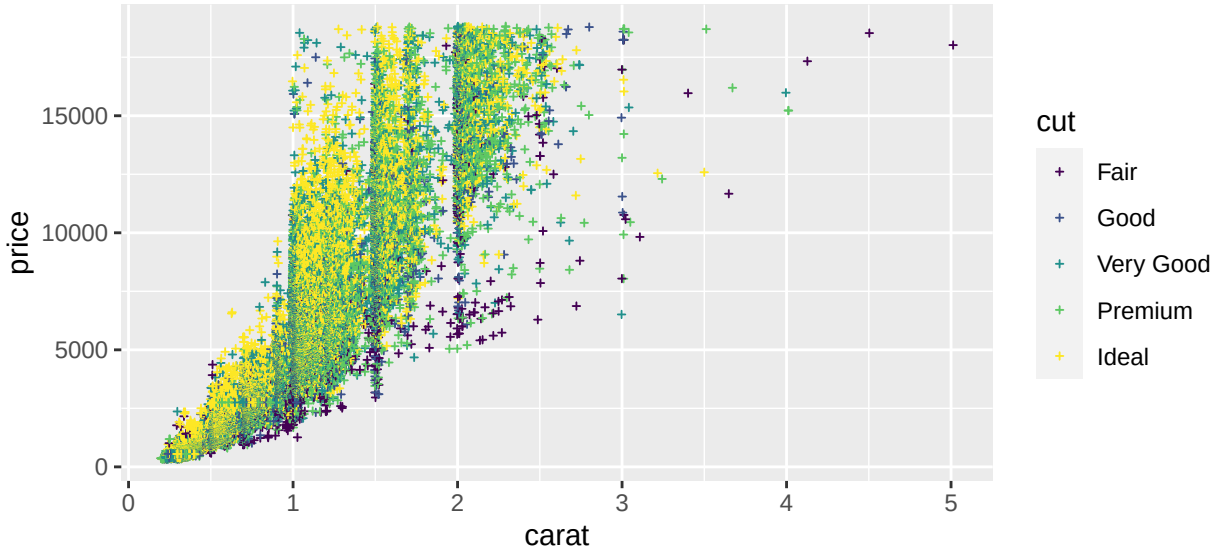


Figure 4: Use `stat_XXXX` instead of `geom_XXXX` to build plots

3. Build plots layer by layer

Suppose, you want to investigate the relationship of the cut quality, clarity of the diamond, and its mass in carats.

For this purpose, you may use boxplots (*What do they show you?*). You can use `geom_boxplot()` for this task.

```
g <- ggplot(data = diamonds, mapping = aes(x = clarity, y = carat))
g3.1 <- g + geom_boxplot(outlier.size = 0.8)
g3.1
```

Adding a linear fitting may not be the best idea for this plot as the X-axis is set to discrete qualitative categories. But for the sake of practice, we can do that. So, we generate another layer, call it `sm`, and apply it to the plot we generated before:

```
sm <- geom_smooth(
  mapping = aes(x = as.integer(clarity)),
  # Notice how the x-value was changed

  method = "lm",
  se = F,
  size = 0.7
)
g3.2 <- g3.1 + sm
g3.2
```

We can apply different layers over the same data. Let's do it here:

```
g <- ggplot(data = diamonds, mapping = aes(x = clarity, y = carat, color = cut))
# As we need to split our data according to the `cut` variable, I decided to
```

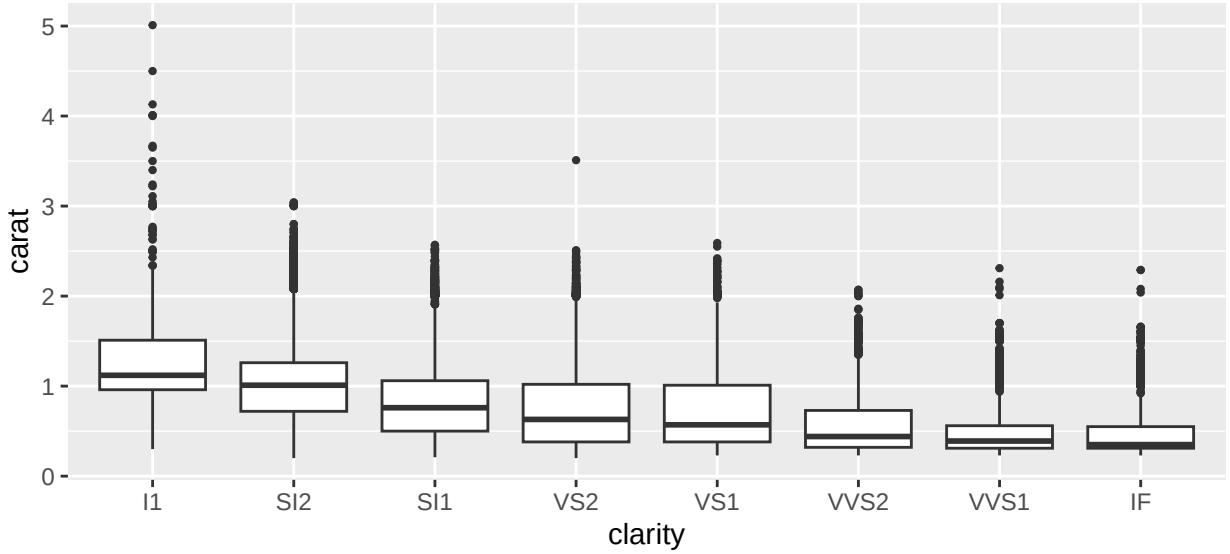


Figure 5: Making a boxplot

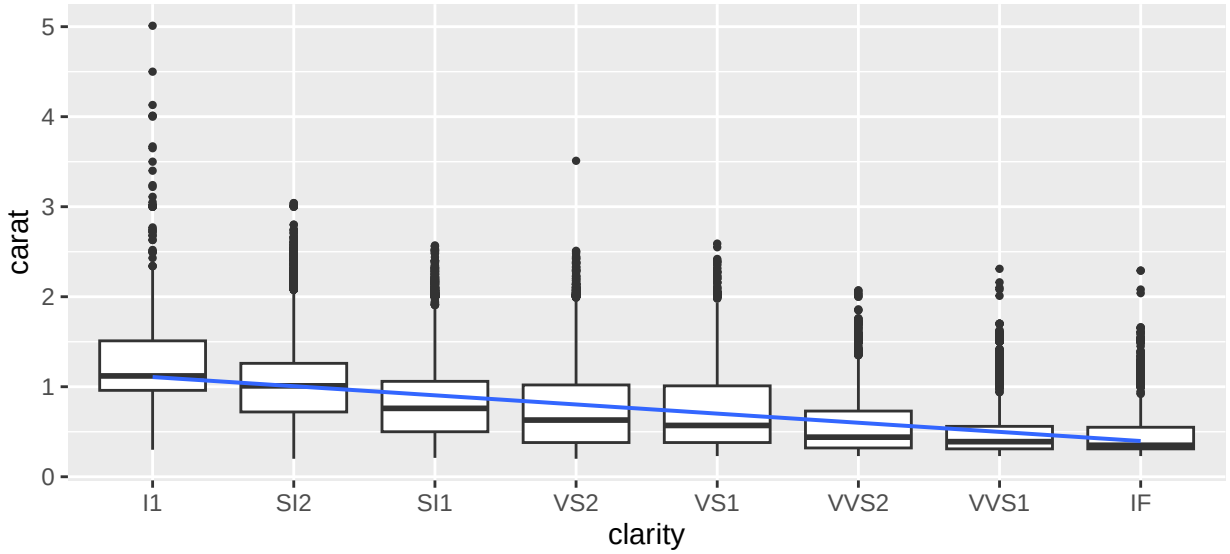


Figure 6: Adding another layer to the original boxplot

```
# change the very original `ggplot` object

g3.3 <- g + geom_boxplot(outlier.size = 0.8) + sm
g3.3
```

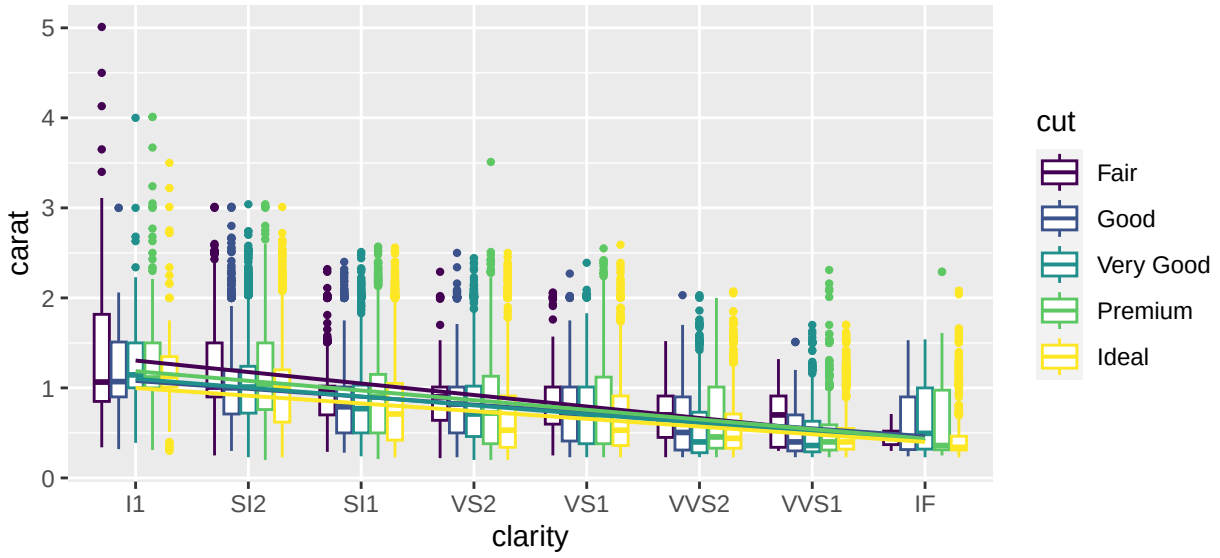


Figure 7: Changing the plot with different aesthetics and layers further

```
g3.4 <-
  g3.3 + ggtitle("Carats vs clarity") + # I added the general title by `ggtitle`
  theme(
    plot.title = element_text(hjust = 0.5),
    # I decided to make this label in the middle
    axis.text.x = element_text(angle = 45),
    legend.position = "bottom" # I decided to change the location of the graph legend
  )
g3.5 <- g3.4 + scale_color_brewer(palette = 3)
g3.5 + facet_wrap(~ color)
```

6. Save the plot to a png file.

4. Scale the y axis

1. Start from the plot in 3.2, transform the y-axis scale to log10 using `scale_y_continuous`. Pay attention to the change of y-axis. What should be the unit? Please change the y-axis label to include the unit using `ylab`.

```
g4.1 <- g3.2 + scale_y_continuous(trans = "log10") + ylab("Carats, log10")
g4.1
```

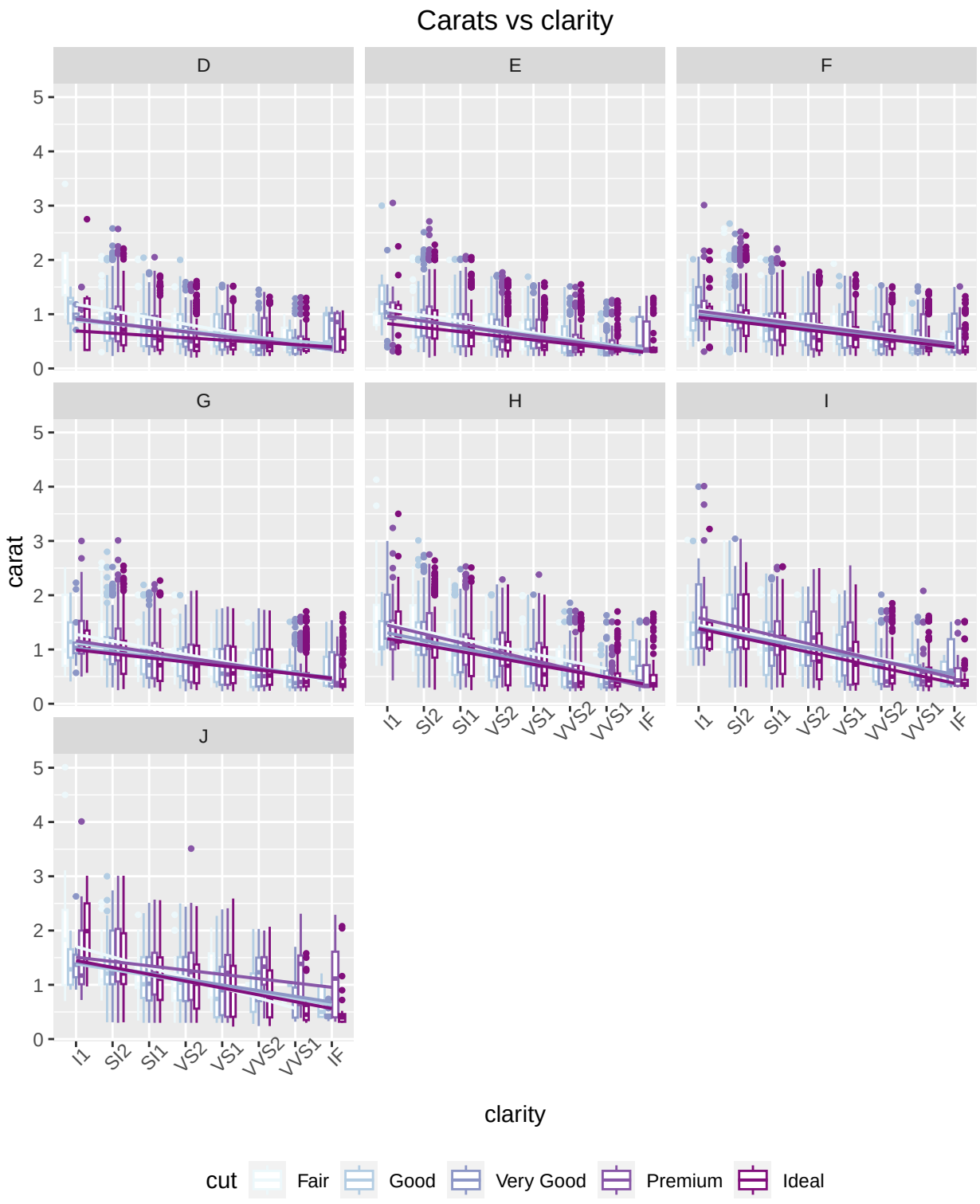


Figure 8: Making facetted plots

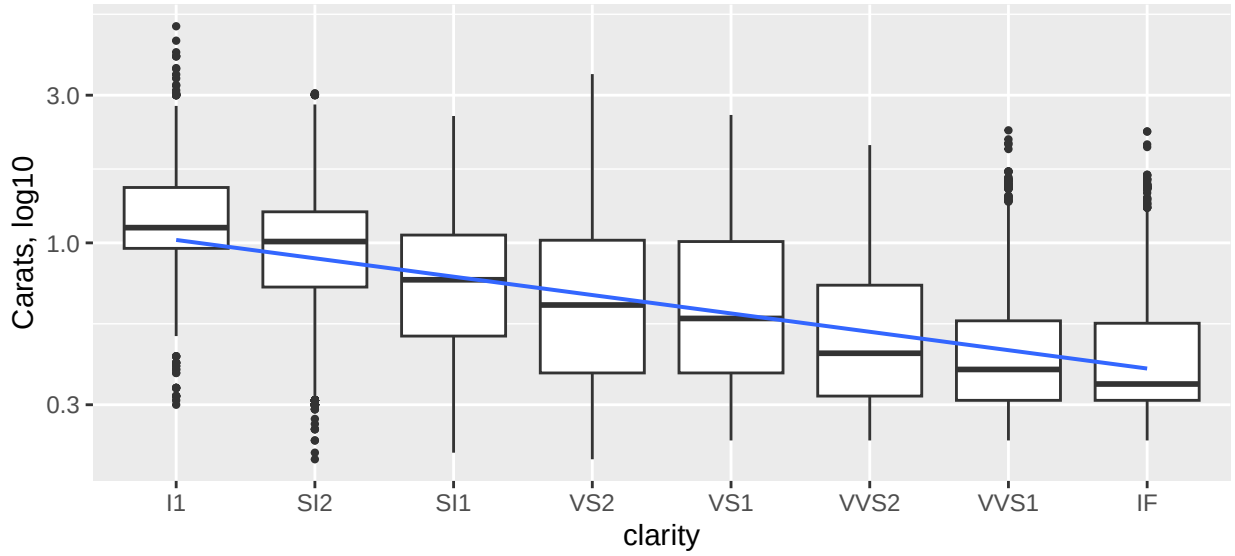


Figure 9: Changing scales

2. redo the boxplot in 3.1 by changing the y aesthetics to `log10(carat)`. Compare the y-axis here with the one in 3.1. Change the y-axis label to include the unit.

Here, you will change the data *within* your very original data file. Let's plot them to compare

```
library(cowplot)
# This package allows you locating several `ggplot` objects one by another.

g4.2.1 <- g +
  geom_boxplot(mapping = aes(y = carat)) +
  theme(legend.position = "none") +
  ggtitle("Original plot")
g4.2.2 <- g + geom_boxplot(mapping = aes(y = log10(carat))) +
  ggtitle("Change the aesthetic")
g4.2.3 <- g4.2.1 +
  scale_y_continuous(trans = "log10") +
  ggtitle("Change the Y-scale")

plot_grid(g4.2.1, g4.2.2, g4.2.3, NULL,
          rel_widths = c(0.75, 1),
          nrow = 2, byrow = T)
```

As you can see, there is a clear difference in the output of graphs. In particular, the values may change, but the axis may require additional attention.

3. Add a layer of linear fitting to the plot in 4.2 by `+ sm` from point 3.2. Do you see a problem? Please fix the problem by change the aesthetics in `sm`.

```
g4.3.1 <- g4.2.2 + sm + theme(legend.position = "none")
g4.3.2 <- g4.2.2 + geom_smooth(
  mapping = aes(x = as.integer(clarity),
```

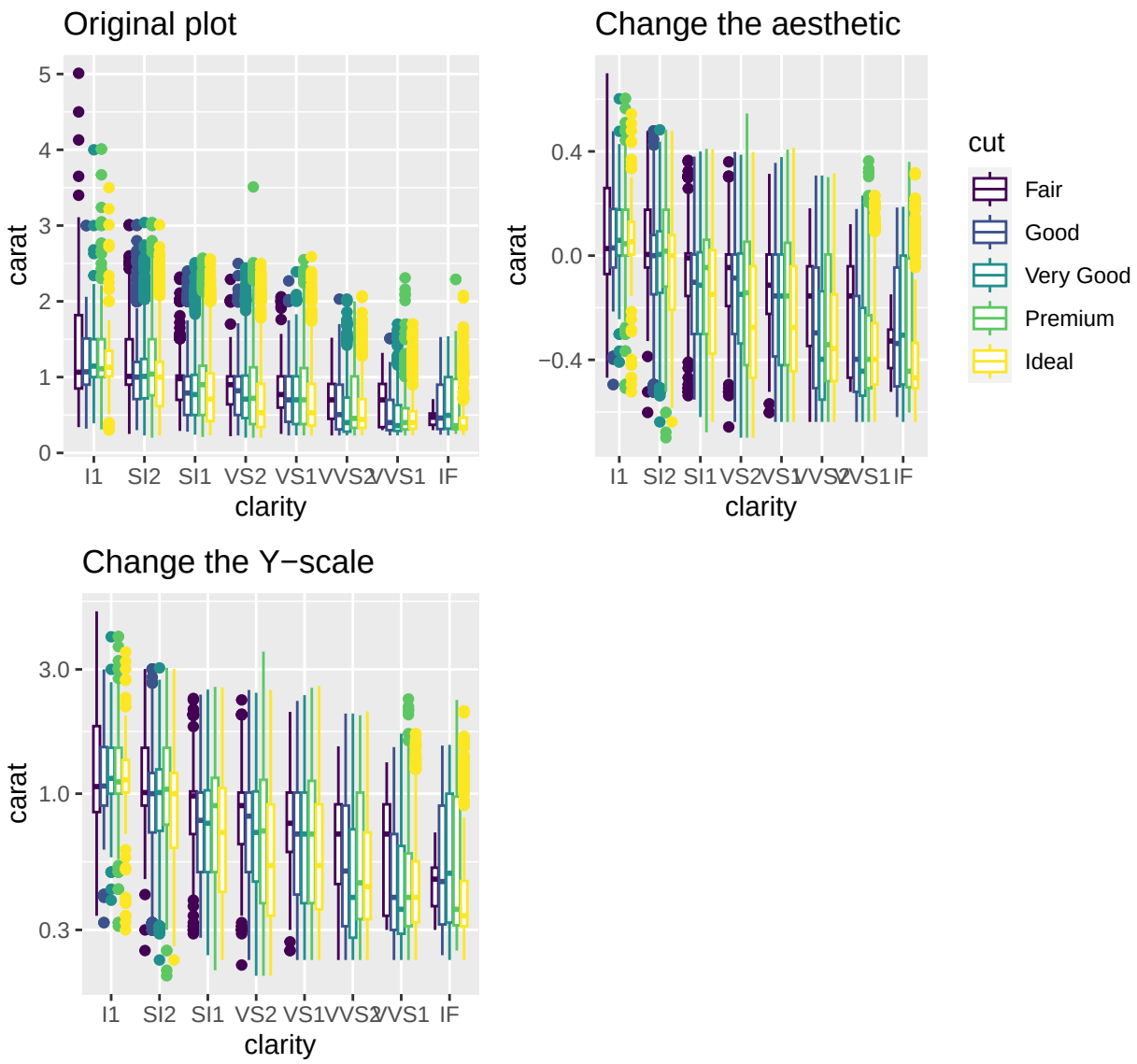
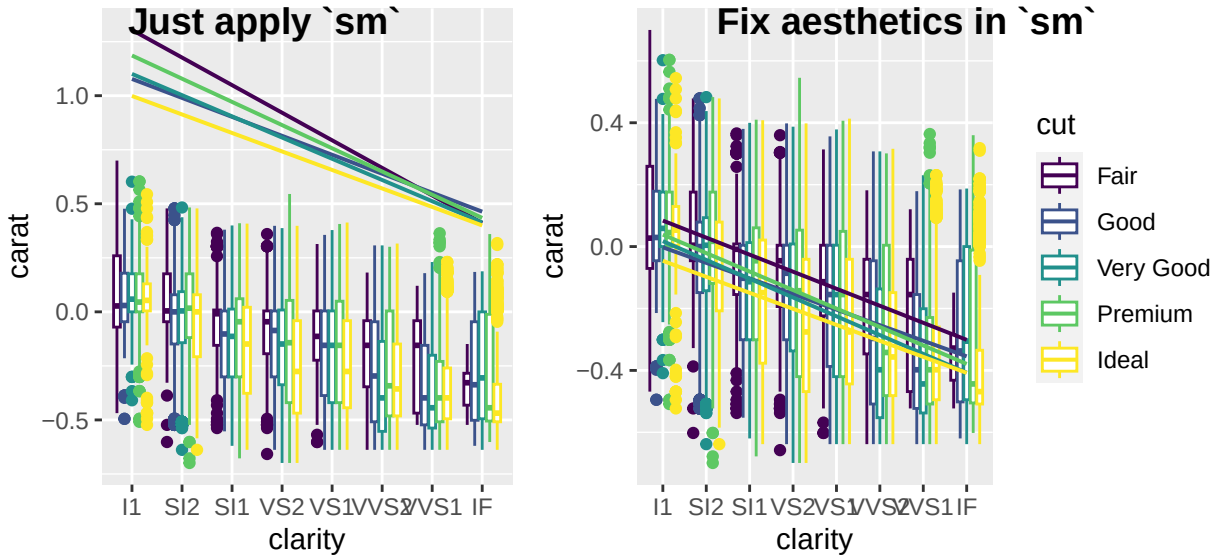


Figure 10: The difference between changing the axes scales and the whole aesthetics

```
                y = log10(carat)),
  # Note how I changed aesthetics in this layer
  method = "lm",
  se = F,
  size = 0.7
)

plot_grid(g4.3.1, g4.3.2,
          rel_widths = c(0.75, 1),
          labels = c("Just apply `sm`", "Fix aesthetics in `sm`"))
```



4. change the range of y-axis in the plot 4.1. set the limits to 0.3 to 1 using `scale_y_continuous`.

```
plot_grid(g4.1, g4.1 + ylim(c(0.3, 1)))
```

5. Jitter plot and scales.

Suppose, you want to show the relationship of the cut quality, clarity, mass, and price of the diamond on the same plot.

1. Sample out 100 cases from `diamonds` dataset. Generate a plot with a layer of boxplot and a layer of jitter plot like this using `geom_jitter`.

```
samp <- sample(x = 1:nrow(diamonds), size = 200)
gg5 <-
  ggplot(data = diamonds[samp,], mapping = aes(x = clarity, y = carat))
g5.1 <-  gg5 + geom_boxplot(outlier.size = 0.8) +
  geom_jitter(width = 0.1,
              mapping = aes(size = carat, color = price))
g5.1
```

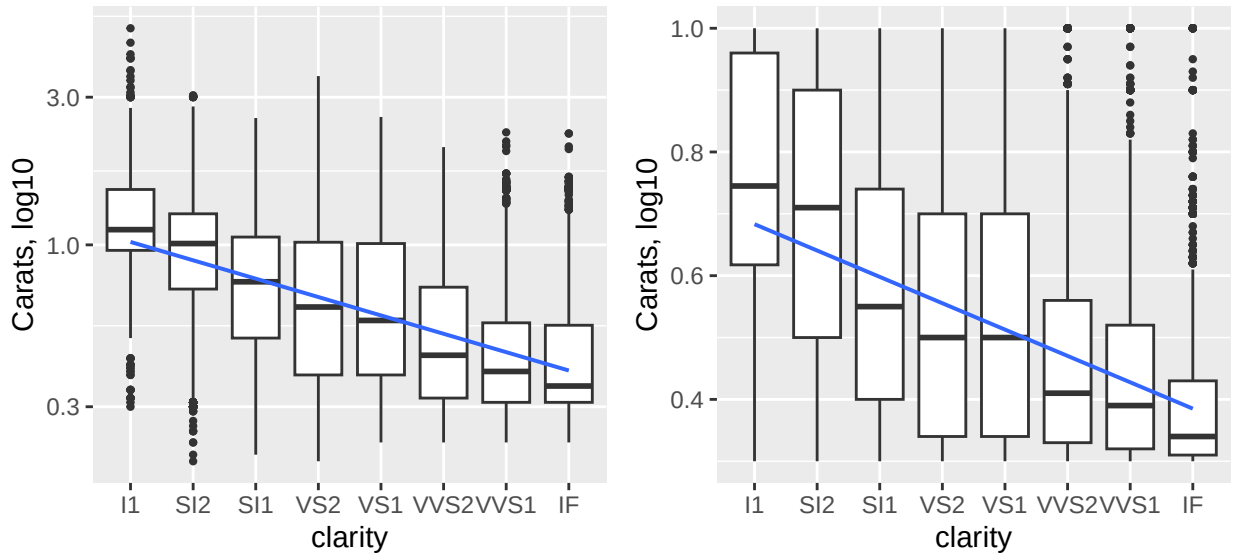
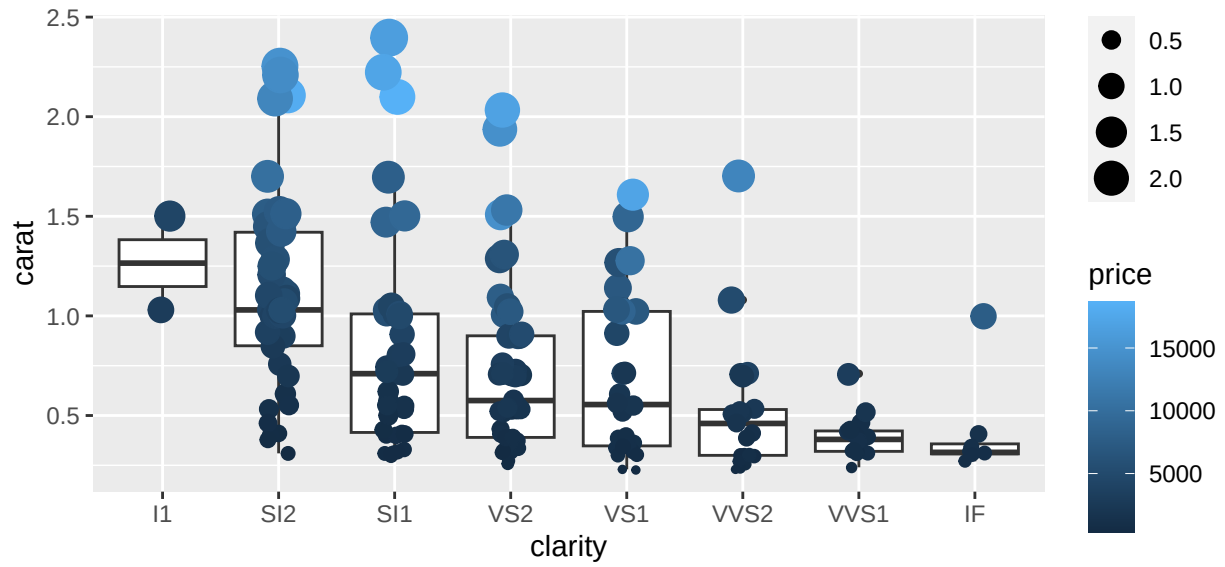


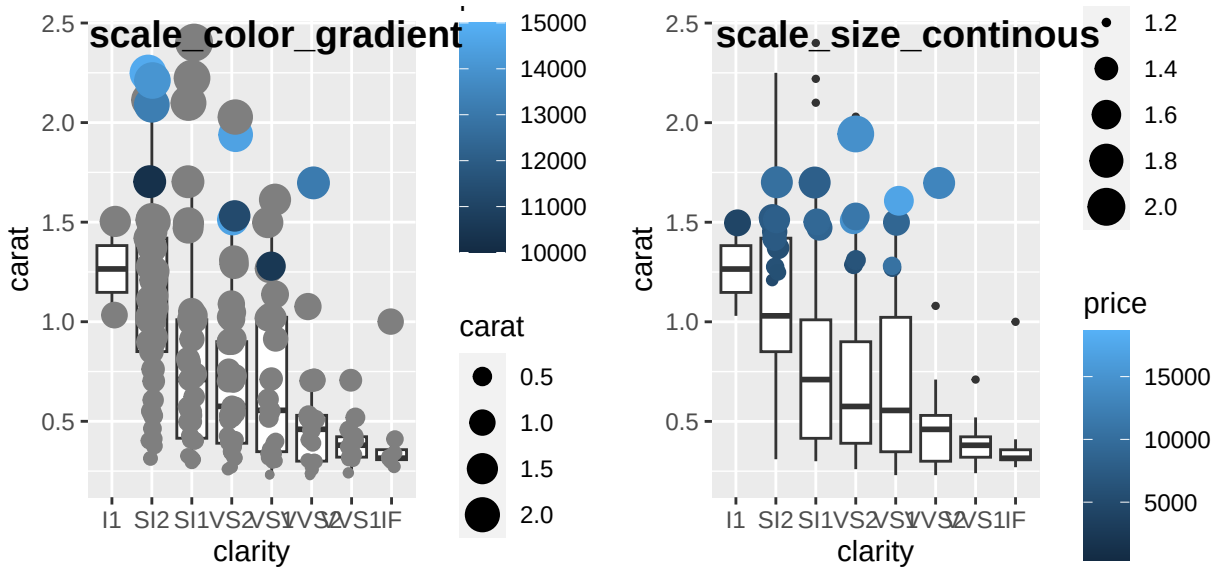
Figure 11: Changing the scale limits



2. Reset the color scale to only color the diamonds in the price `range(10000, 15000)`. Try `scale_color_gradient`.
3. Reset the size scale to only show the diamonds in the carat `range(1.2, 2)`. Try `scale_size_continous`.

```
g5.2 <- g5.1 + scale_color_gradient(limits = c(10000, 15000))
g5.3 <- g5.1 + scale_size_continuous(limits = c(1.2, 2))
plot_grid(
  g5.2,
  g5.3,
  ncol = 2,
  nrow = 1,
  labels = c("scale_color_gradient", "scale_size_continous"),
```

```
  hjust = -0.26
)
```



6. Position.

In the lecture, we discussed different position adjustments. Generate the plots that can describe the number of different categories of diamonds according to their `cut` and `clarity` similar to the one described in the fourth part of the lecture (the slide about the position adjustment). Use the position argument in `geom_bar`.

```
gg6 <- ggplot(data = diamonds, mapping = aes(x = clarity, group = cut))
g6.1 <-
  gg6 + geom_bar(mapping = aes(fill = cut), position = "stack") +
  ggtitle("Stacking") + theme(
    plot.title = element_text(hjust = 0.5),
    axis.text.y = element_text(size = 10),
    axis.text.x = element_text(size = 7),
    legend.position = "n",
    aspect.ratio = 7 / 7

  )
g6.2 <-
  gg6 + geom_bar(mapping = aes(fill = cut), position = "fill") +
  ggtitle("Filling") + theme(
    plot.title = element_text(hjust = 0.5),
    axis.text = element_text(size = 9),
    axis.text.y = element_text(size = 10),
    axis.text.x = element_text(size = 7),
    legend.position = "right",
    aspect.ratio = 7 / 5
  )
g6.3 <-
  gg6 + geom_bar(mapping = aes(fill = cut), position = "dodge") +
  ggtitle("Positioning near") + theme(
```

```
    plot.title = element_text(hjust = 0.5),
    axis.text.x = element_text(size = 7),
    axis.text.y = element_text(size = 10),
    legend.position = "n",
    aspect.ratio = 7 / 7
  )
plot_grid(
  plotlist = list(g6.1, g6.2, g6.3, NULL),
  rel_widths = c(0.75, 1.25),
  rel_heights = 1
)
```

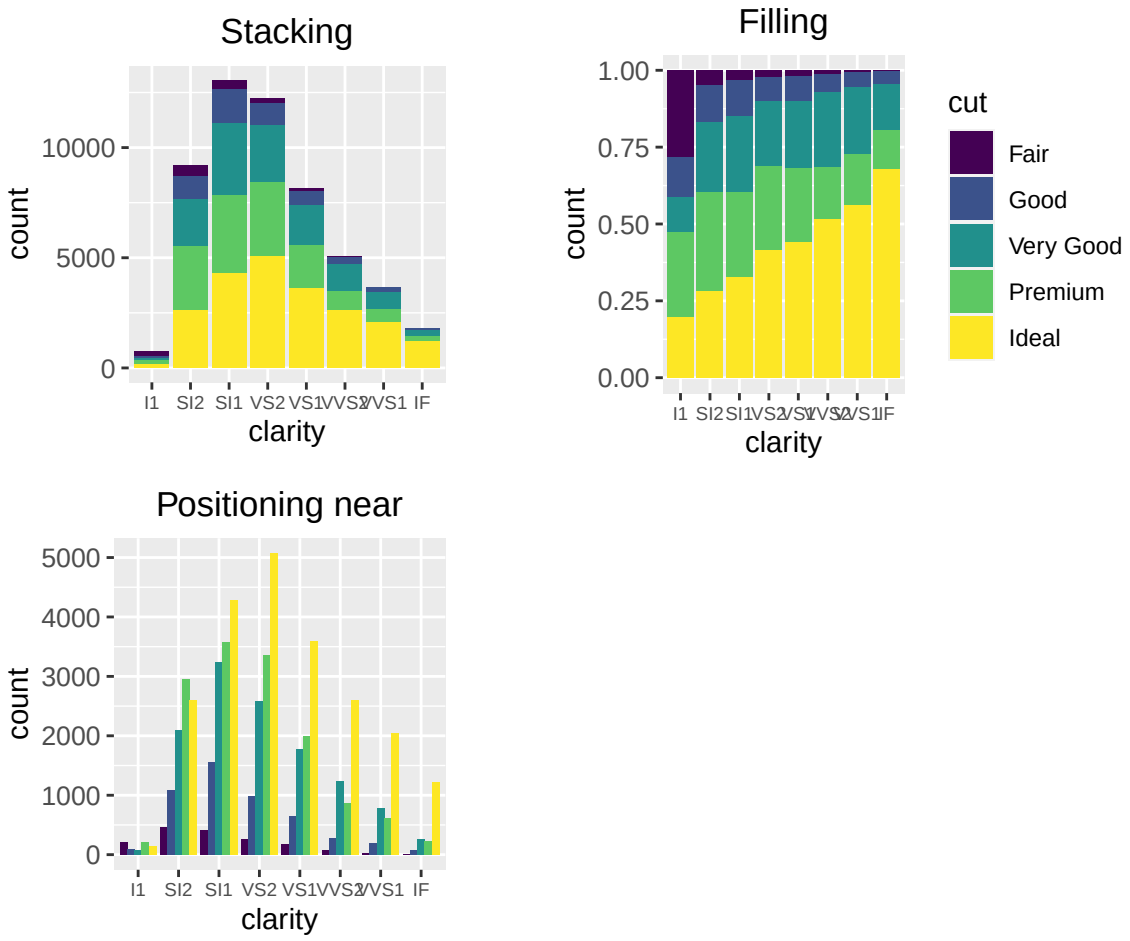


Figure 12: Adjusting the position

The original materials were created by Chaochen Wang in 2021.

The solution was created by Dmytro Shytikov in 2023