Different plots

Oleksii Stroganov

2024-03-31

```
# Tidyverse and ggplot magic
library(tidyverse)
library(ggthemes)
library(ggsci)
library(ggsankey)
library(patchwork)

# Install required packages if missing:
# install.packages(c("tidyverse","ggthemes","ggsci","devtools","patchwork"))
# devtools::install_github("davidsjoberg/ggsankey")
```

Adaptive ggsci

To use beautiful color palettes ggsci for plots that need more than 10 colors we can use following function that can generate new colors from given dataset

```
# Adaptive ggsci (required for sankey plot)

adaptive_pal <- function(values) {
  force(values)
  function(n = 10) {
    if (n <= length(values)) {
      values[seq_len(n)]
    } else {
      colorRampPalette(values, alpha = TRUE)(n)
    }
  }
}

pal_npg_adaptive <- function(palette = c("nrc"), alpha = 1) {
  palette <- match.arg(palette)

  if (alpha > 1L | alpha <= 0L) stop("alpha must be in (0, 1]")

  raw_cols <- ggsci:::ggsci_db$"npg"[[palette]]
  raw_cols_rgb <- col2rgb(raw_cols)
  alpha_cols <- rgb(
    raw_cols_rgb[1L, ], raw_cols_rgb[2L, ], raw_cols_rgb[3L, ],
    alpha = alpha * 255L, names = names(raw_cols),
    maxColorValue = 255L
  )

  adaptive_pal(unname(alpha_cols))
```

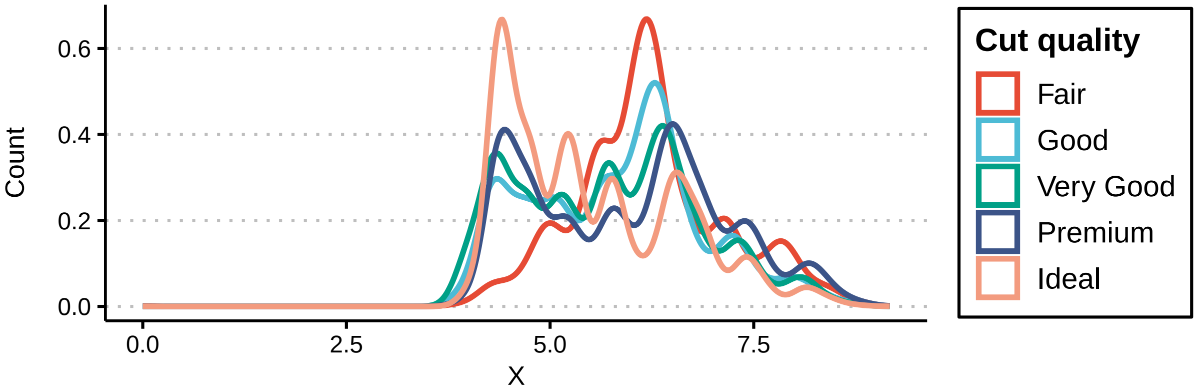
```r
}

scale_color_npg_adaptive <- function(palette = c("nrc"), alpha = 1, ...) {
  palette <- match.arg(palette)
  discrete_scale("colour", "npg", pal_npg_adaptive(palette, alpha), ...)
}

scale_fill_npg_adaptive <- function(palette = c("nrc"), alpha = 1, ...) {
  palette <- match.arg(palette)
  discrete_scale("fill", "npg", pal_npg_adaptive(palette, alpha), ...)
}
```

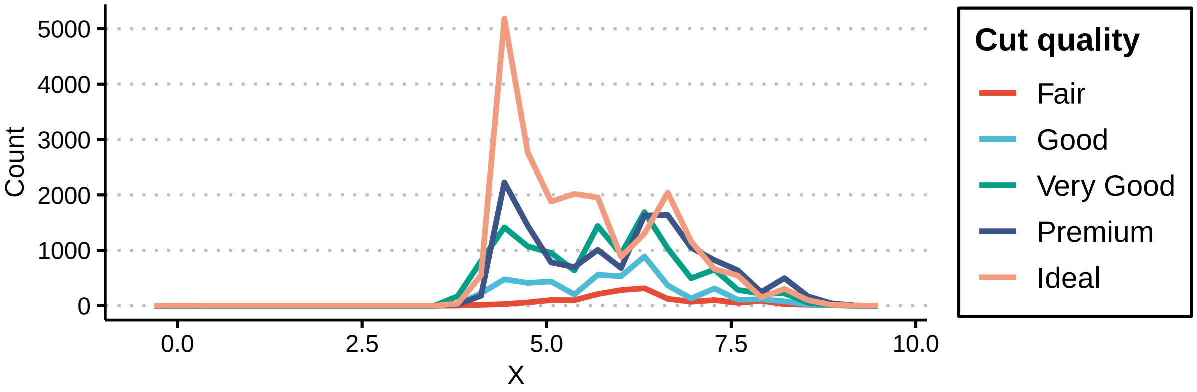
Create an example dataset

```r
# Use example dataset
smaller <- diamonds |>
  filter(carat < 3)
```

Density and Freqpoly plots

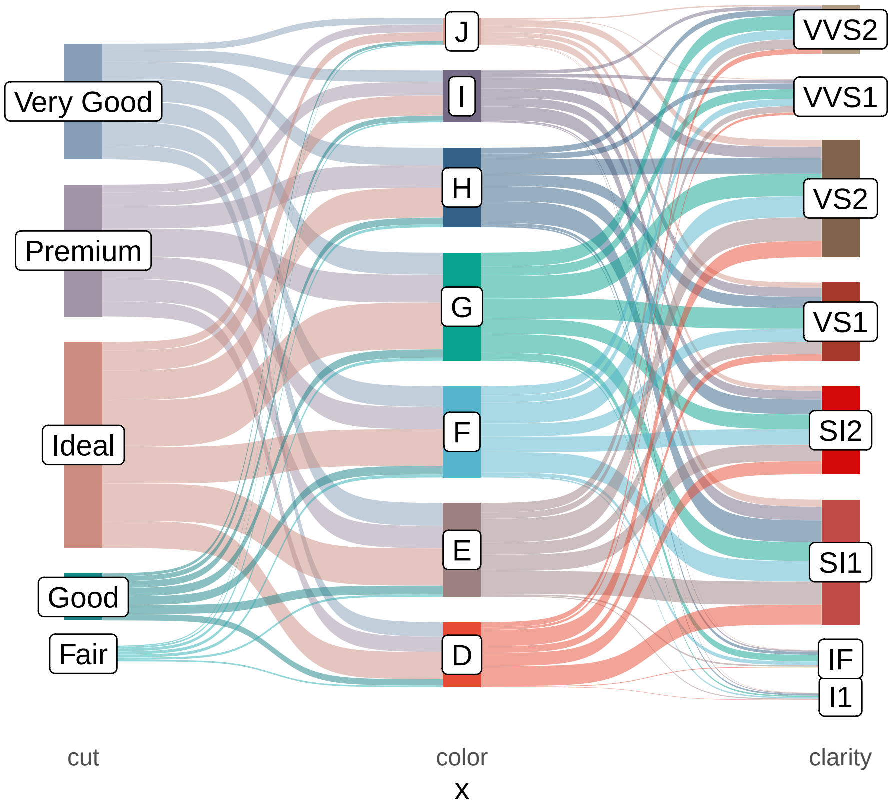
```r
p1_1 <- smaller |>
  ggplot(aes(x = x, color = cut)) +
  geom_density(linewidth = 1) +
  labs(x = "X", y = "Count") +
  guides(color=guide_legend(title="Cut quality")) +
  scale_color_npg() +
  theme_clean() +
  theme(
    plot.background = element_blank()
  )

p1_2 <- smaller |>
  ggplot(aes(x = x, color = cut)) +
  geom_freqpoly(linewidth = 1) +
  labs(x = "X", y = "Count") +
  guides(color=guide_legend(title="Cut quality")) +
  scale_color_npg() + # Color palette
  theme_clean() +
  theme(
    plot.background = element_blank()
  )

# Run this to generate plots
p1_1 / p1_2
```





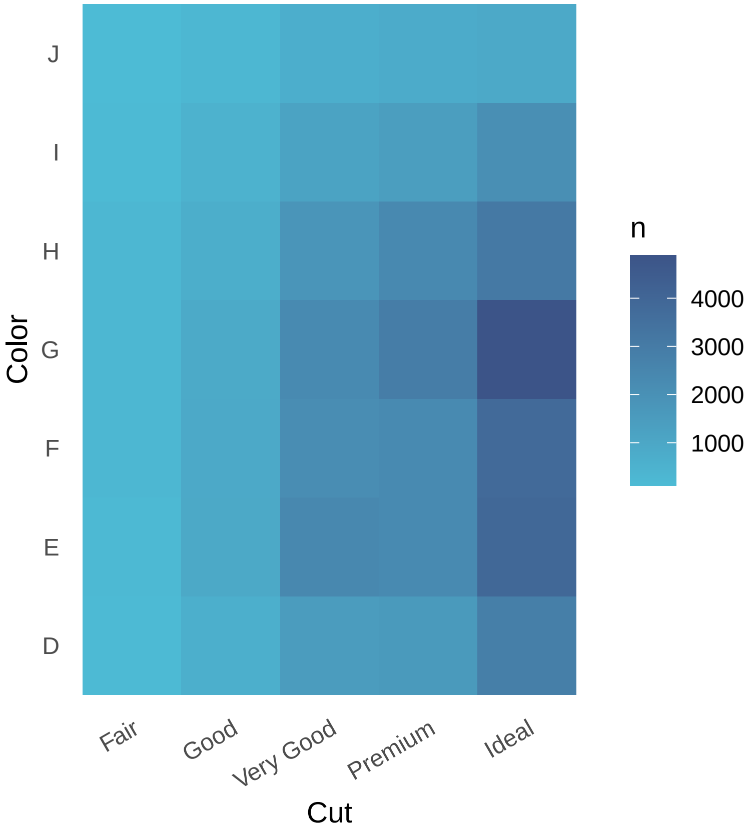
Sankey plot

```
smaller |>
  make_long(cut, color, clarity) |>
  ggplot(aes(
    x = x,
    next_x = next_x,
    node = node,
    next_node = next_node,
    fill = factor(node),
    label = node)) +
  geom_sankey(flow.alpha = 0.5, node.color = 0) +
  geom_sankey_label(size = 4, color = "black", fill = "white") +
  scale_fill_npg_adaptive() + # Color palette
  theme_sankey() +
  theme(
    legend.position = "none",
  )
```



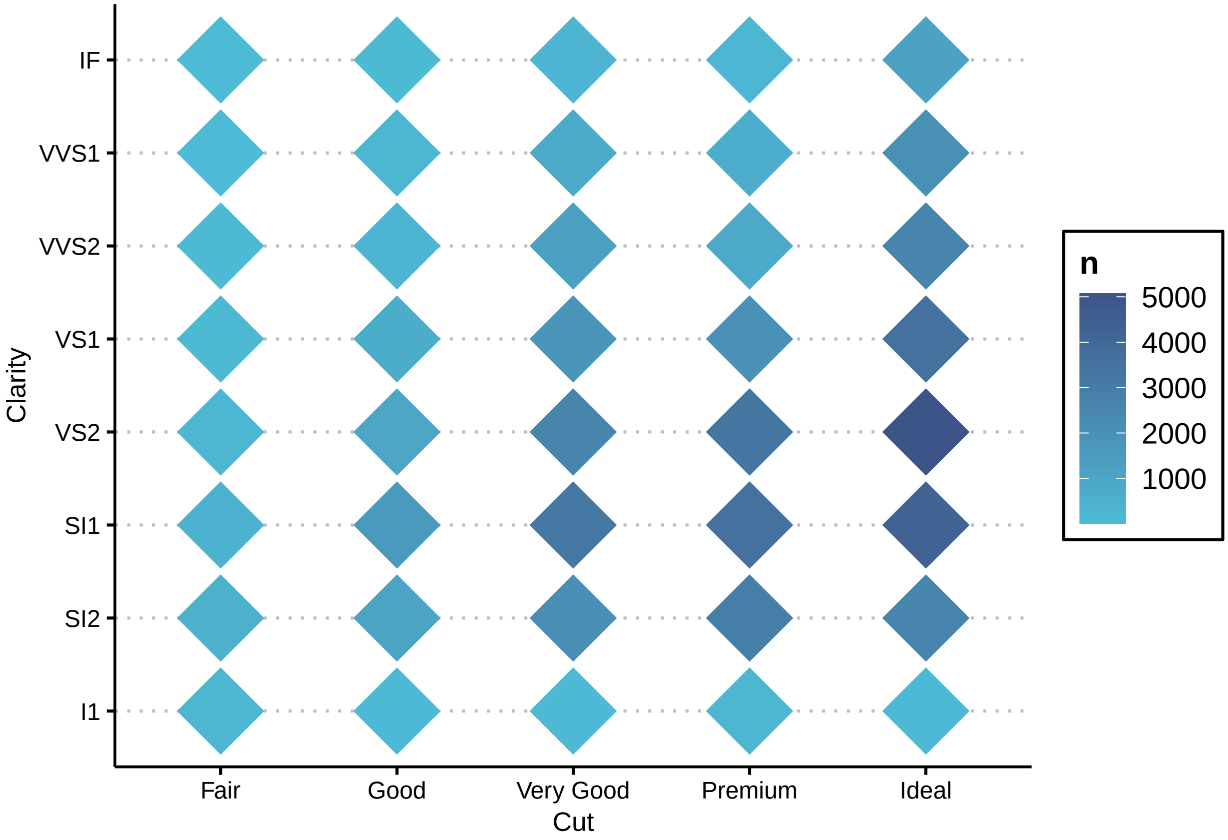
Heatmap

```
smaller |>
  group_by(
    cut,
    color
  ) |>
  summarise(
    n = n()
  ) |>
  ggplot(aes(x = cut, y = color, fill = n)) +
  geom_tile() +
  labs(
    x = "Cut",
    y = "Color"
  ) +
  coord_fixed() +
  scale_fill_gradient(
    low = pal_npg()(5)[2],
    high = pal_npg()(5)[4],
    na.value = "grey50",
  ) + # Color palette and legend title
  theme_minimal() +
  theme(
    panel.grid = element_blank(),
    axis.text.x = element_text(angle = 30, hjust = 1)
  )
```



Scatter plot

```
smaller |>
  group_by(
    cut,
    clarity
  ) |>
  summarise(
    n = n()
  ) |>
  ggplot(aes(x = cut, y = clarity, color = n)) +
  geom_point(size = 15, shape = 18) +
  labs(
    x = "Cut",
    y = "Clarity"
  ) +
  scale_color_gradient(
    low = pal_npg()(5)[2],
    high = pal_npg()(5)[4],
  ) + # Color palette and legend title
  theme_clean() +
  theme(
    plot.background = element_blank()
  )
```


IF
VVS1
VVS2
VS1
VS2
SI1
SI2
I1
Clarity
Fair
Good
Very Good
Premium
Ideal
Cut
n
5000
4000
3000
2000
1000