Diamond Data

Diamond Data

This project explores both R's diamond data set using tidyverse's data visualization and data analysis packages.

What is Diamond Data?

Stata is home to a number of pre-loaded data sets, as are the different packages within it. Diamonds contains information about ~54,000 different diamonds, recording everything from price to size to cut to color to clarity.

```
#Package set up
options(repos = c(CRAN = "https://cran.rstudio.com/"))

install.packages("ggridges")
```

```
The downloaded binary packages are in
    /var/folders/jx/19wzy4r974zgwcx8kgshk9400000gn/T//RtmpQyxrpZ/downloaded_packages
```

```
install.packages("dbscan")
```

```
The downloaded binary packages are in
    /var/folders/jx/19wzy4r974zgwcx8kgshk9400000gn/T//RtmpQyxrpZ/downloaded_packages
```

```
install.packages("leaps")
```

```
The downloaded binary packages are in
    /var/folders/jx/19wzy4r974zgwcx8kgshk9400000gn/T//RtmpQyxrpZ/downloaded_packages
```

```
install.packages("caret")
```

```
The downloaded binary packages are in
    /var/folders/jx/19wzy4r974zgwcx8kgshk9400000gn/T//RtmpQyxrpZ/downloaded_packages
```

```
library(caret)
```

```
Loading required package: ggplot2

Loading required package: lattice
```

```
library(ggplot2)
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(scales)
library(ggridges)
library(purrr)
```

```
Attaching package: 'purrr'

The following object is masked from 'package:scales':

    discard

The following object is masked from 'package:caret':

    lift
```

```
library(dbscan)
```

```
Attaching package: 'dbscan'

The following object is masked from 'package:stats':
```

```
    as.dendrogram
```

```
library(leaps)
```

Of course, the first step of any data analysis is getting a broad overview of what the data set (or sets) look like. Using a few simple commands, we are able to determine that we have 53,940 rows of data (each a specific diamond) and ten columns (each a unique variable.) The variables are as follows: carat (a measure of diamond size), cut (a qualitative measure of how well a diamond was cut), color (self explanatory, from best to worse), clarity (a qualitative measure of clearness), depth (depth percentage), table (width ratio), price (at sale, in dollars), x (length), y (width), and z (depth).

```
#Loading data and conducting exploratory analyses
data(diamonds)
dim(diamonds)
```

```
[1] 53940    10
```

```
nrow(diamonds)
```

```
[1] 53940
```

```
ncol(diamonds)
```

```
[1] 10
```

```
#Heads and tails
print(head(diamonds))
```

```
# A tibble: 6 × 10
  carat cut       color clarity depth table price     x     y     z
  <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

```
print(tail(diamonds))
```

```
# A tibble: 6 × 10
```

```
  carat cut       color clarity depth table price     x     y     z
  <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1  0.72 Premium   D     SI1      62.7    59  2757  5.69  5.73  3.58
2  0.72 Ideal     D     SI1      60.8    57  2757  5.75  5.76  3.5
3  0.72 Good      D     SI1      63.1    55  2757  5.69  5.75  3.61
4  0.7  Very Good D     SI1      62.8    60  2757  5.66  5.68  3.56
5  0.86 Premium   H     SI2      61      58  2757  6.15  6.12  3.74
6  0.75 Ideal     D     SI2      62.2    55  2757  5.83  5.87  3.64
```

```r
#Other stuff
summary(diamonds)
```

```
     carat               cut        color        clarity          depth      
 Min.   :0.2000   Fair      : 1610   D: 6775   SI1    :13065   Min.   :43.00  
 1st Qu.:0.4000   Good      : 4906   E: 9797   VS2    :12258   1st Qu.:61.00  
 Median :0.7000   Very Good:12082   F: 9542   SI2    : 9194   Median :61.80  
 Mean   :0.7979   Premium   :13791   G:11292   VS1    : 8171   Mean   :61.75  
 3rd Qu.:1.0400   Ideal     :21551   H: 8304   VVS2   : 5066   3rd Qu.:62.50  
 Max.   :5.0100                     I: 5422   VVS1   : 3655   Max.   :79.00  
                                    J: 2808   (Other): 2531                  
     table           price             x                y         
 Min.   :43.00   Min.   :  326   Min.   : 0.000   Min.   : 0.000  
 1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710   1st Qu.: 4.720  
 Median :57.00   Median : 2401   Median : 5.700   Median : 5.710  
 Mean   :57.46   Mean   : 3933   Mean   : 5.731   Mean   : 5.735  
 3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540   3rd Qu.: 6.540  
 Max.   :95.00   Max.   :18823   Max.   :10.740   Max.   :58.900  

       z         
 Min.   : 0.000  
 1st Qu.: 2.910  
 Median : 3.530  
 Mean   : 3.539  
 3rd Qu.: 4.040  
 Max.   :31.800  
```

```r
str(diamonds)
```

```
tibble [53,940 × 10] (S3: tbl_df/tbl/data.frame)
 $ carat  : num [1:53940] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
 $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
 $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
```

```
$ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
$ depth  : num [1:53940] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
$ table  : num [1:53940] 55 61 65 58 58 57 57 55 61 61 ...
$ price  : int [1:53940] 326 326 327 334 335 336 336 337 337 338 ...
$ x      : num [1:53940] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
$ y      : num [1:53940] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
$ z      : num [1:53940] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

```r
names(diamonds)
```

```
[1] "carat"   "cut"     "color"   "clarity" "depth"   "table"   "price"
[8] "x"       "y"       "z"
```

Understanding Variable Distributions

A great next step in any exploratory data analysis is to understand how each variable is distributed. Here, data visualization takes greater precedence, as do simple summary statistics.

```r
#Assigning visually appealing custom colors
diamond_blue <- "#2E86AB"
diamond_gold <- "#F18F01"
diamond_silver <- "#C5C3C6"

#Creating a shared custom theme to make my final graphs more visually appealing
# Custom theme
custom_theme <- theme_minimal() +
  theme(
    plot.title = element_text(size = 14, face = "bold", hjust = 0.5, margin = margin(b
    plot.subtitle = element_text(size = 11, hjust = 0.5, color = "gray40", margin = ma
    axis.title = element_text(size = 12, face = "bold"),
    axis.text = element_text(size = 10),
    panel.grid.minor = element_blank(),
    panel.grid.major = element_line(color = "gray90", size = 0.3),
    plot.background = element_rect(fill = "white", color = NA),
    panel.background = element_rect(fill = "white", color = NA)
  )
```

```
Warning: The `size` argument of `element_line()` is deprecated as of ggplot2 3.4.0.
ℹ Please use the `linewidth` argument instead.
```

Carat

Based on the analysis below, it is clear that carat is strongly right skewed and unipolar, with a mean of 0.7979 and standard deviation of 0.4740112.

```
summary(diamonds$carat)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.2000  0.4000  0.7000  0.7979  1.0400  5.0100 
```

```
sd(diamonds$carat)
```

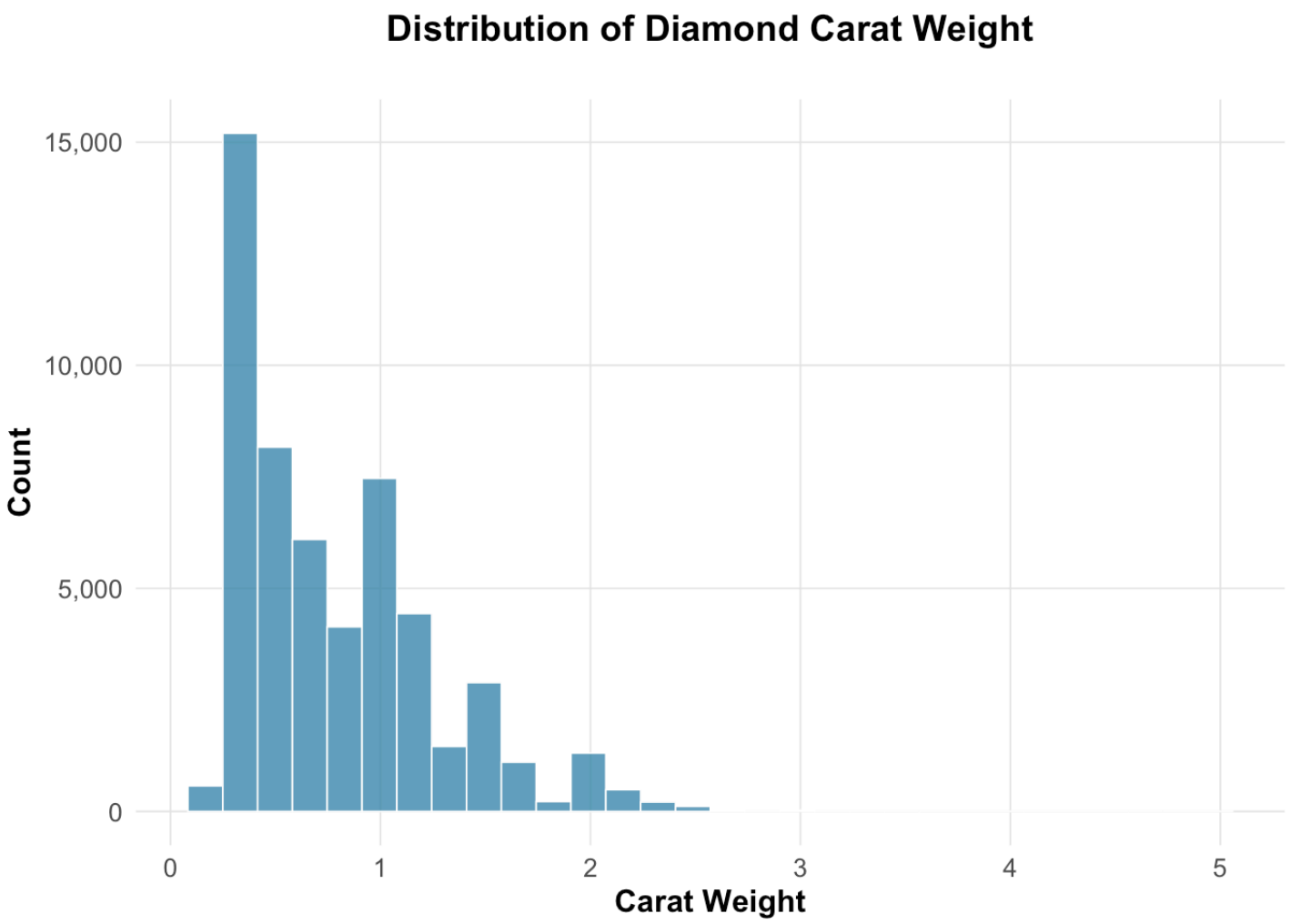
```
[1] 0.4740112
```

```
# Creating a Histogram
p1 <- ggplot(diamonds, aes(x = carat)) +
  geom_histogram(bins = 30,
                 fill = diamond_blue,
                 color = "white",
                 alpha = 0.8,
                 size = 0.3) +
  labs(
    title = "Distribution of Diamond Carat Weight",
    x = "Carat Weight",
    y = "Count"
  ) +
  custom_theme +
  scale_y_continuous(labels = scales::comma_format())
```

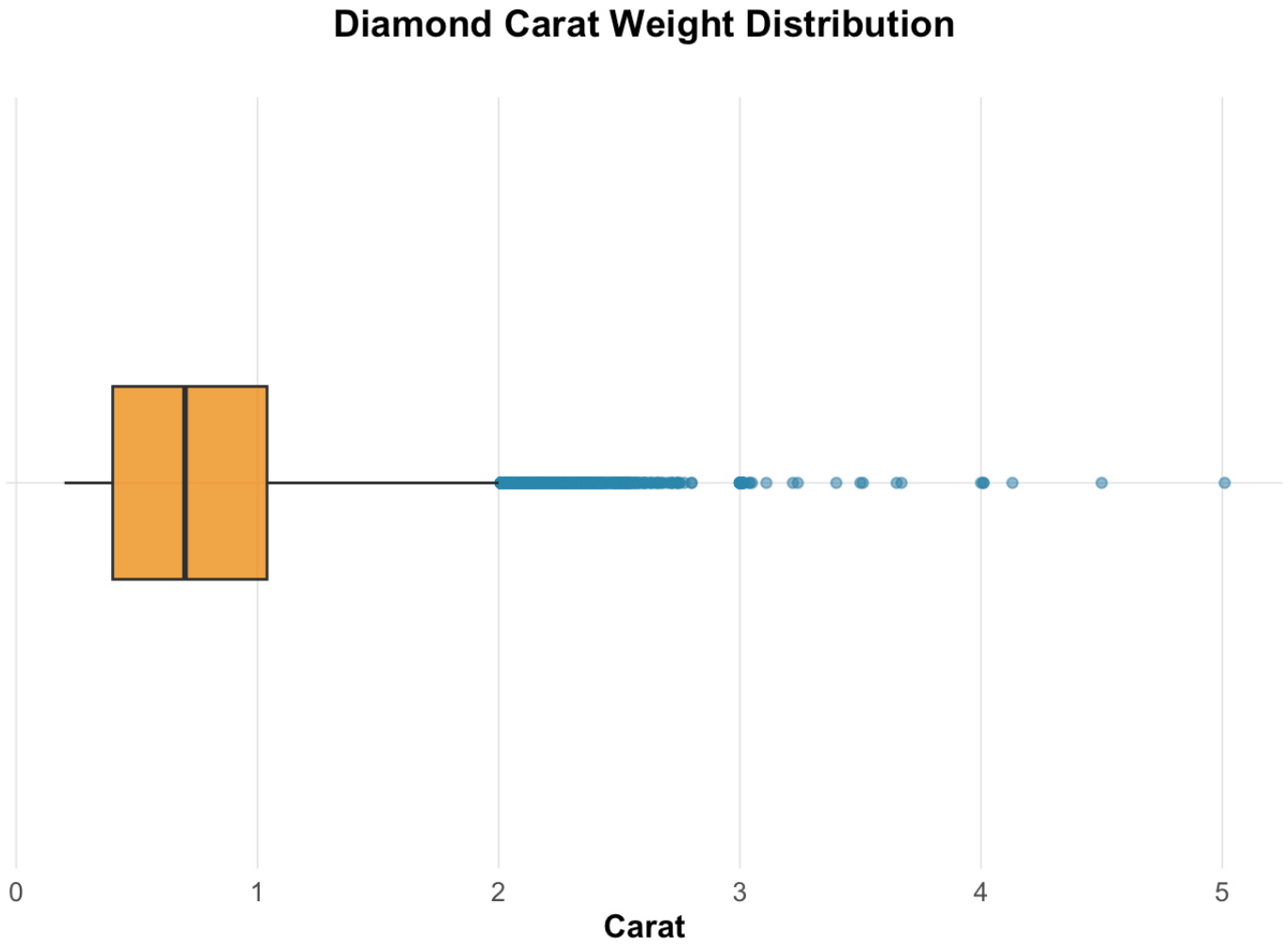
```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
ℹ Please use `linewidth` instead.
```

```
# Creating a boxplot
p2 <- ggplot(diamonds, aes(x = carat, y = "")) +
  geom_boxplot(width = 0.3,
               fill = diamond_gold,
               alpha = 0.8,
               outlier.color = diamond_blue,
               outlier.alpha = 0.6) +
  labs(
    title = "Diamond Carat Weight Distribution",
    x = "Carat",
    y = ""
```

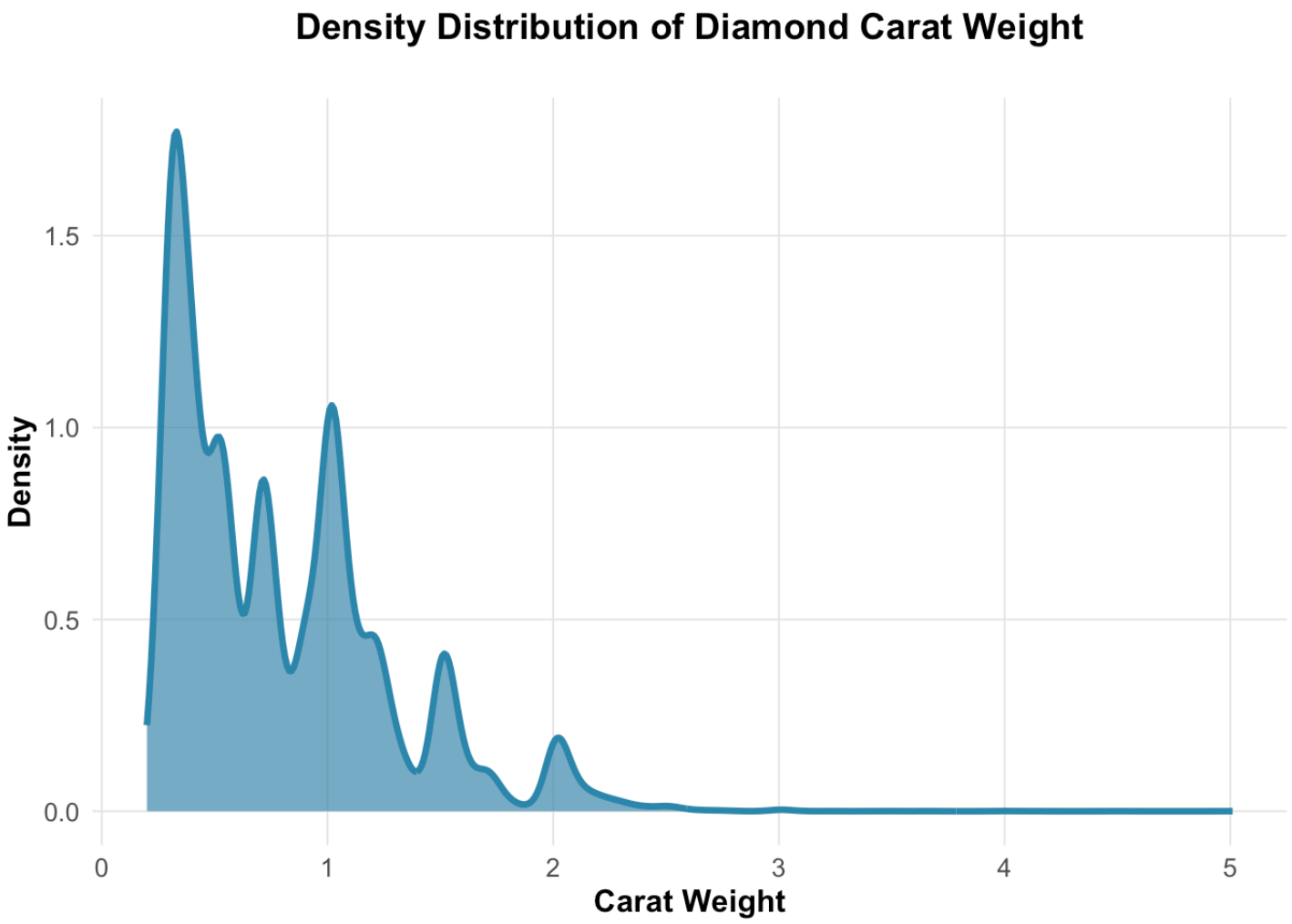
```
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())

# Creating a nice density plot
p3 <- ggplot(diamonds, aes(x = carat)) +
  geom_density(fill = diamond_blue,
               alpha = 0.7,
               color = diamond_blue,
               size = 1.2) +
  labs(
    title = "Density Distribution of Diamond Carat Weight",
    x = "Carat Weight",
    y = "Density"
  ) +
  custom_theme

# Creating a nice ECDF (empirical cumulative distribution function)
p4 <- ggplot(diamonds, aes(x = carat)) +
  stat_ecdf(color = diamond_blue,
            size = 1.2,
            alpha = 0.8) +
  geom_hline(yintercept = c(0.25, 0.5, 0.75),
             linetype = "dotted",
             color = diamond_gold,
             alpha = 0.7) +
  labs(
    title = "Cumulative Distribution of Diamond Carat Weight",
    x = "Carat Weight",
    y = "Cumulative Probability"
  ) +
  custom_theme +
  scale_y_continuous(labels = scales::percent_format())

# Displaying all plots
print(p1)
```


Distribution of Diamond Carat Weight
15,000
10,000
5,000
0
Count
0
1
2
3
4
5
Carat Weight

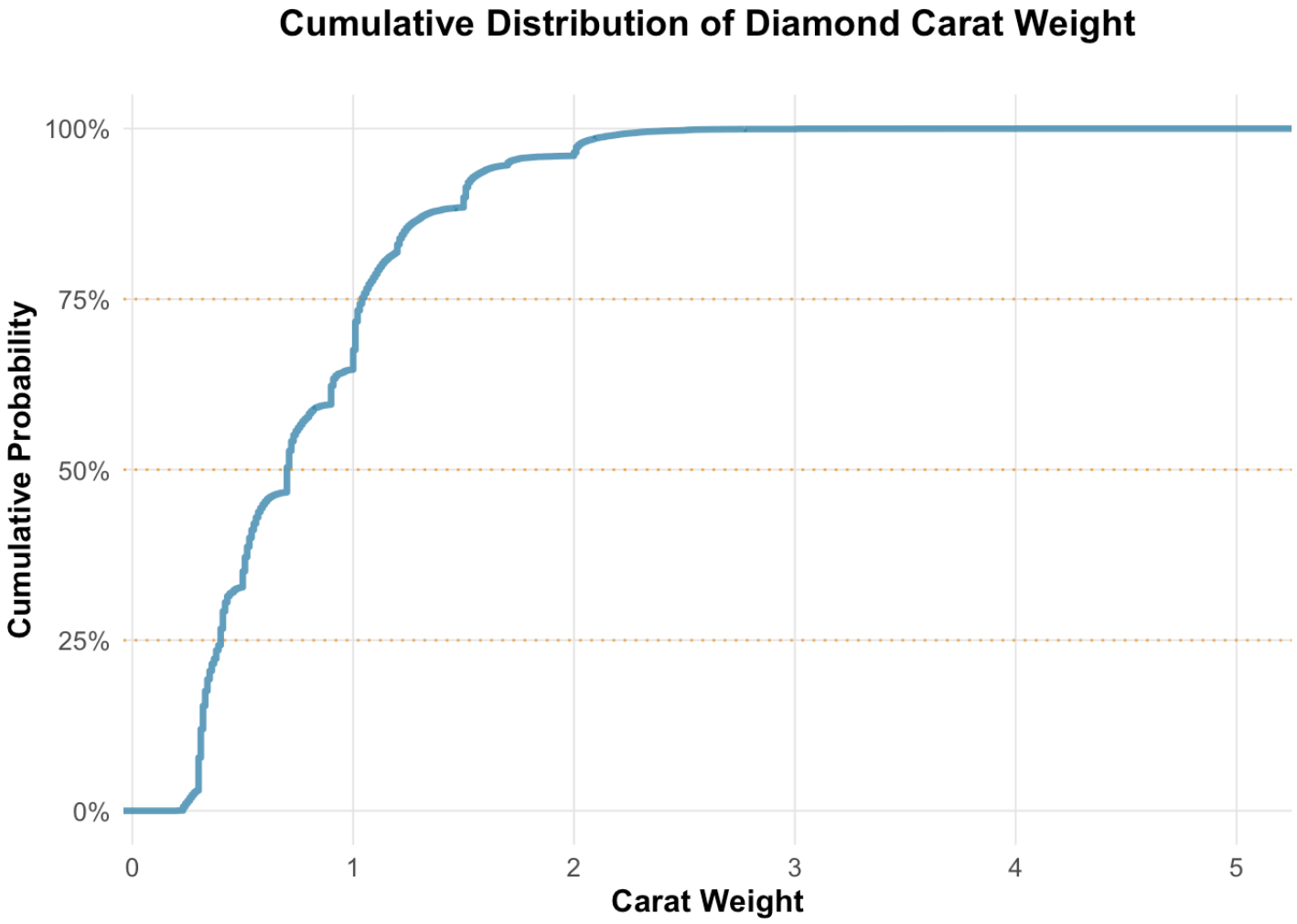
```
print(p2)
```




```
print(p3)
```


Density Distribution of Diamond Carat Weight
Density
1.5
1.0
0.5
0.0
0
1
2
3
4
5
Carat Weight

```
print(p4)
```



Cut

Based on the analysis below, it is clear that cut is not uniformly distributed. Instead, the highest grade is the most common at 40% of all observations, followed by the second highest (about 25%) and third highest grades (about 22%) respectively.

```
#Prepping a nice color scheme for cut levels
cut_colors <- c("Fair" = "#C5C3C6", "Good" = "#A8DADC", "Very Good" = "#457B9D",
                "Premium" = "#2E86AB", "Ideal" = "#F18F01")

# Finding summary statistics
diamonds |>
  count(cut) |>
  mutate(percentage = scales::percent(n / sum(n), accuracy = 0.1)) |>
  arrange(desc(n)) |>
  print()
```

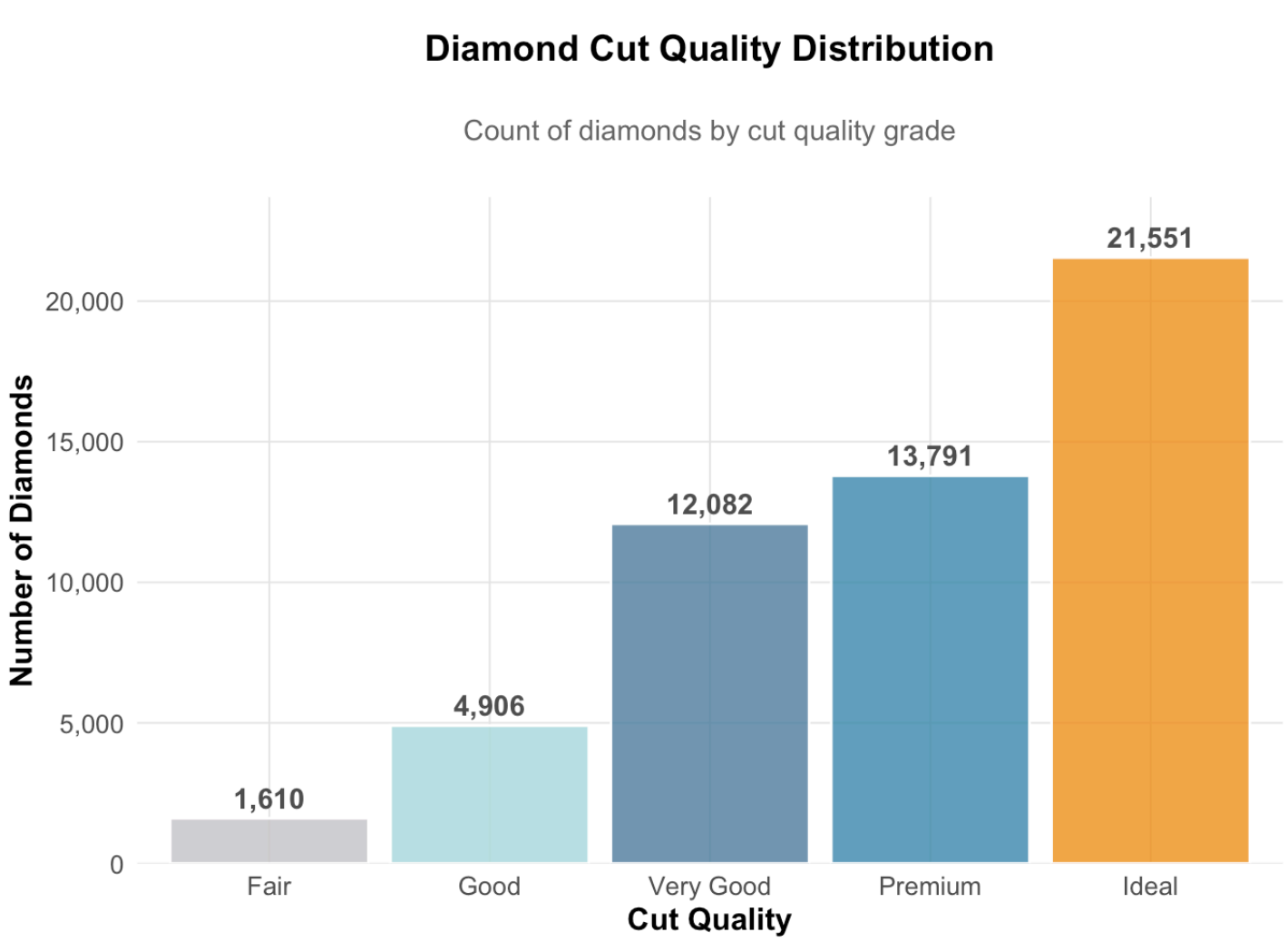
```
# A tibble: 5 × 3
  cut           n percentage
  <ord>     <int> <chr>
1 Ideal     21551 40.0%
2 Premium   13791 25.6%
3 Very Good 12082 22.4%
4 Good       4906 9.1%
5 Fair       1610 3.0%
```

```
# 1. Making a bar chart
p1 <- ggplot(diamonds, aes(x = cut)) +
  geom_bar(aes(fill = cut),
           alpha = 0.8,
           color = "white",
           size = 0.5) +
  geom_text(stat = "count",
            aes(label = comma(after_stat(count))),
            vjust = -0.5,
            fontface = "bold",
            color = "gray30") +
  scale_fill_manual(values = cut_colors) +
  labs(
    title = "Diamond Cut Quality Distribution",
    subtitle = "Count of diamonds by cut quality grade",
    x = "Cut Quality",
    y = "Number of Diamonds"
  ) +
  custom_theme +
  theme(legend.position = "none") +
  scale_y_continuous(labels = comma_format(),
                     expand = expansion(mult = c(0, 0.1)))

# 2. Making a pie chart
p2 <- diamonds |>
  count(cut) |>
  mutate(percentage = n / sum(n) * 100,
         label = paste0(cut, "\n", round(percentage, 1), "%")) |>
  ggplot(aes(x = "", y = n, fill = cut)) +
  geom_bar(stat = "identity",
           width = 1,
           color = "white",
           size = 2) +
  geom_text(aes(label = label),
```

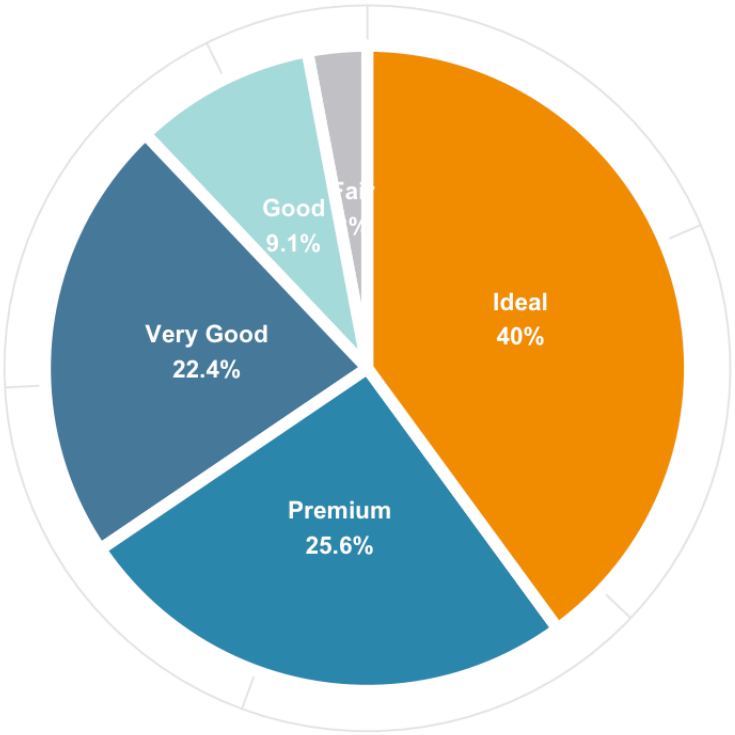
```
            position = position_stack(vjust = 0.5),
            fontface = "bold",
            color = "white",
            size = 3) +
  coord_polar("y", start = 0) +
  scale_fill_manual(values = cut_colors) +
  labs(
    title = "Diamond Cut Distribution",
    subtitle = "Proportional breakdown by cut quality"
  ) +
  custom_theme +
  theme(
    axis.text = element_blank(),
    axis.ticks = element_blank(),
    axis.title = element_blank(),
    panel.grid = element_blank(),
    legend.position = "none"
  )

# 3. Making a stacked bar chart
p3 <- diamonds |>
  count(cut) |>
  mutate(percentage = n / sum(n)) |>
  ggplot(aes(x = 1, y = percentage, fill = cut)) +
  geom_bar(stat = "identity",
           color = "white",
           size = 1.5,
           width = 0.6) +
  geom_text(aes(label = paste0(cut, "\n", scales::percent(percentage, accuracy = 0.1))
            position = position_stack(vjust = 0.5),
            fontface = "bold",
            color = "white",
            size = 3.5) +
  scale_fill_manual(values = cut_colors) +
  labs(
    title = "Diamond Cut Quality Proportions",
    subtitle = "Relative distribution as percentages"
  ) +
  custom_theme +
  theme(
    axis.text = element_blank(),
    axis.ticks = element_blank(),
    axis.title = element_blank(),
    panel.grid = element_blank(),
    legend.position = "none"
```

```
  ) +
  coord_flip()

# 4. Enhanced dot plot with size and color
p4 <- diamonds |>
  count(cut) |>
  ggplot(aes(x = cut, y = 1)) +
  geom_point(aes(size = n, color = cut),
             alpha = 0.8,
             stroke = 2) +
  geom_text(aes(label = comma(n)),
            vjust = -2,
            fontface = "bold",
            color = "gray30") +
  scale_size_continuous(range = c(10, 30),
                        guide = "none") +
  scale_color_manual(values = cut_colors) +
  labs(
    title = "Diamond Cut Distribution",
    subtitle = "Bubble size represents count of diamonds",
    x = "Cut Quality",
    y = ""
  ) +
  custom_theme +
  theme(
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank(),
    panel.grid.major.y = element_blank(),
    legend.position = "none"
  ) +
  ylim(0.5, 1.5)

# 5. Cool text-based visualization
p5 <- diamonds |>
  count(cut) |>
  ggplot(aes(x = cut, y = 1)) +
  geom_text(aes(label = cut,
                size = n,
                color = cut),
            fontface = "bold",
            alpha = 0.8) +
  geom_text(aes(label = paste("n =", comma(n))),
            vjust = 2,
            color = "gray50",
            size = 3) +
```

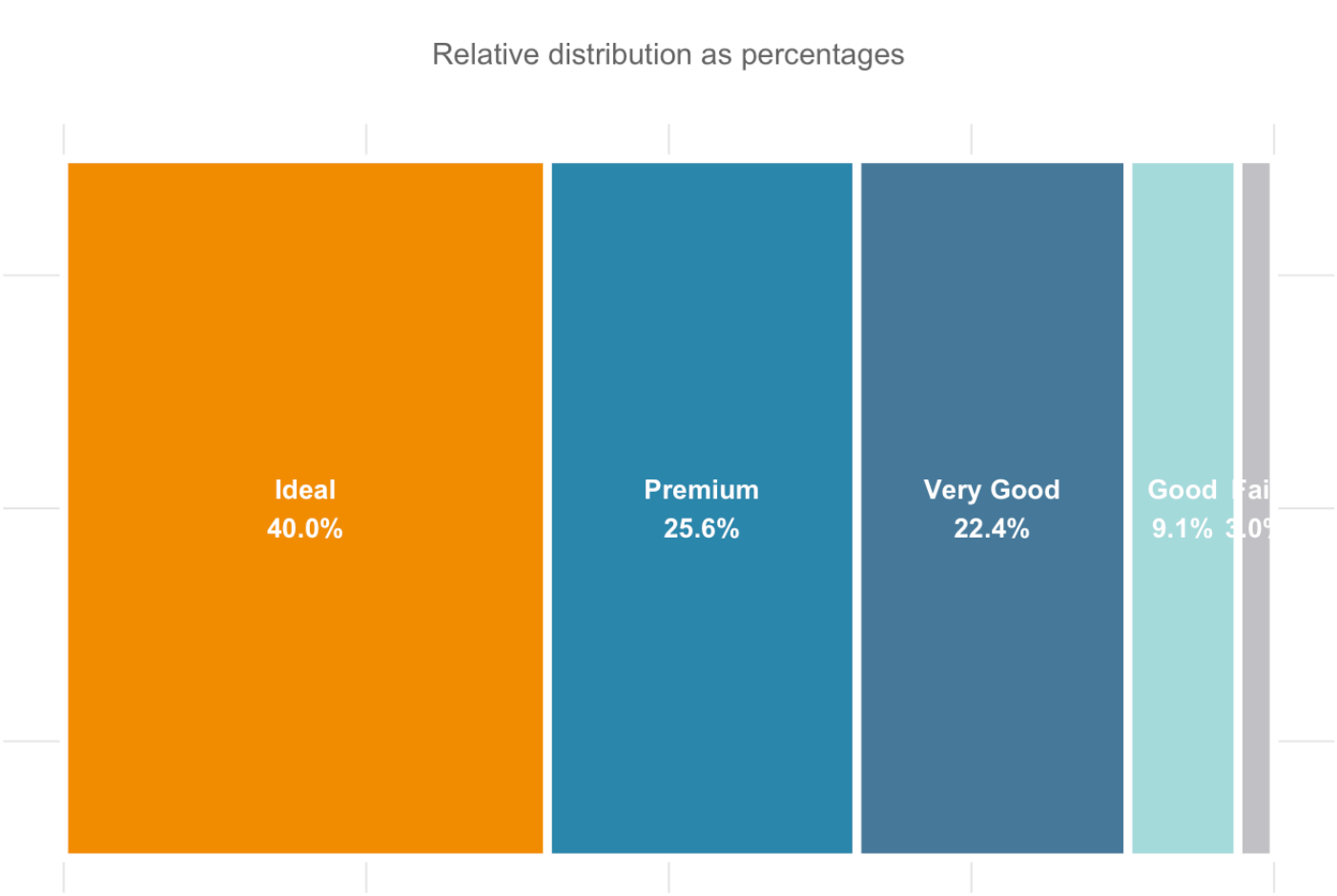
```
  scale_size_continuous(range = c(4, 12), guide = "none") +
  scale_color_manual(values = cut_colors) +
  labs(
    title = "Diamond Cut Distribution",
    subtitle = "Text size proportional to frequency",
    x = "Cut Quality",
    y = ""
  ) +
  custom_theme +
  theme(
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank(),
    panel.grid = element_blank(),
    legend.position = "none"
  ) +
  ylim(0.5, 1.5)

# Display all plots
print(p1)
```


Diamond Cut Quality Distribution
Count of diamonds by cut quality grade
Number of Diamonds
20,000
15,000
10,000
5,000
0
1,610
4,906
12,082
13,791
21,551
Fair
Good
Very Good
Premium
Ideal
Cut Quality

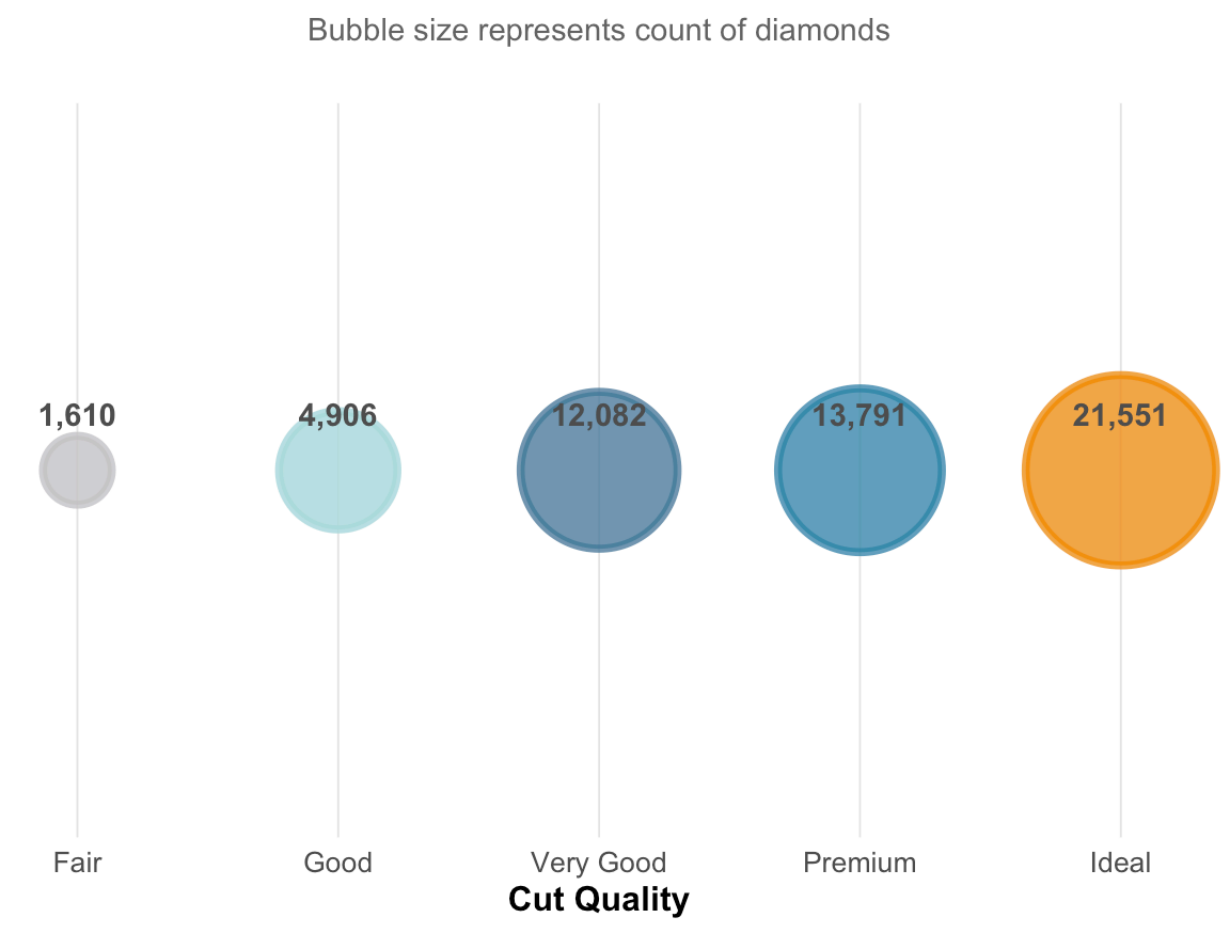
```
print(p2)
```


Diamond Cut Distribution
Proportional breakdown by cut quality
Ideal
40%
Premium
25.6%
Very Good
22.4%
Good
9.1%

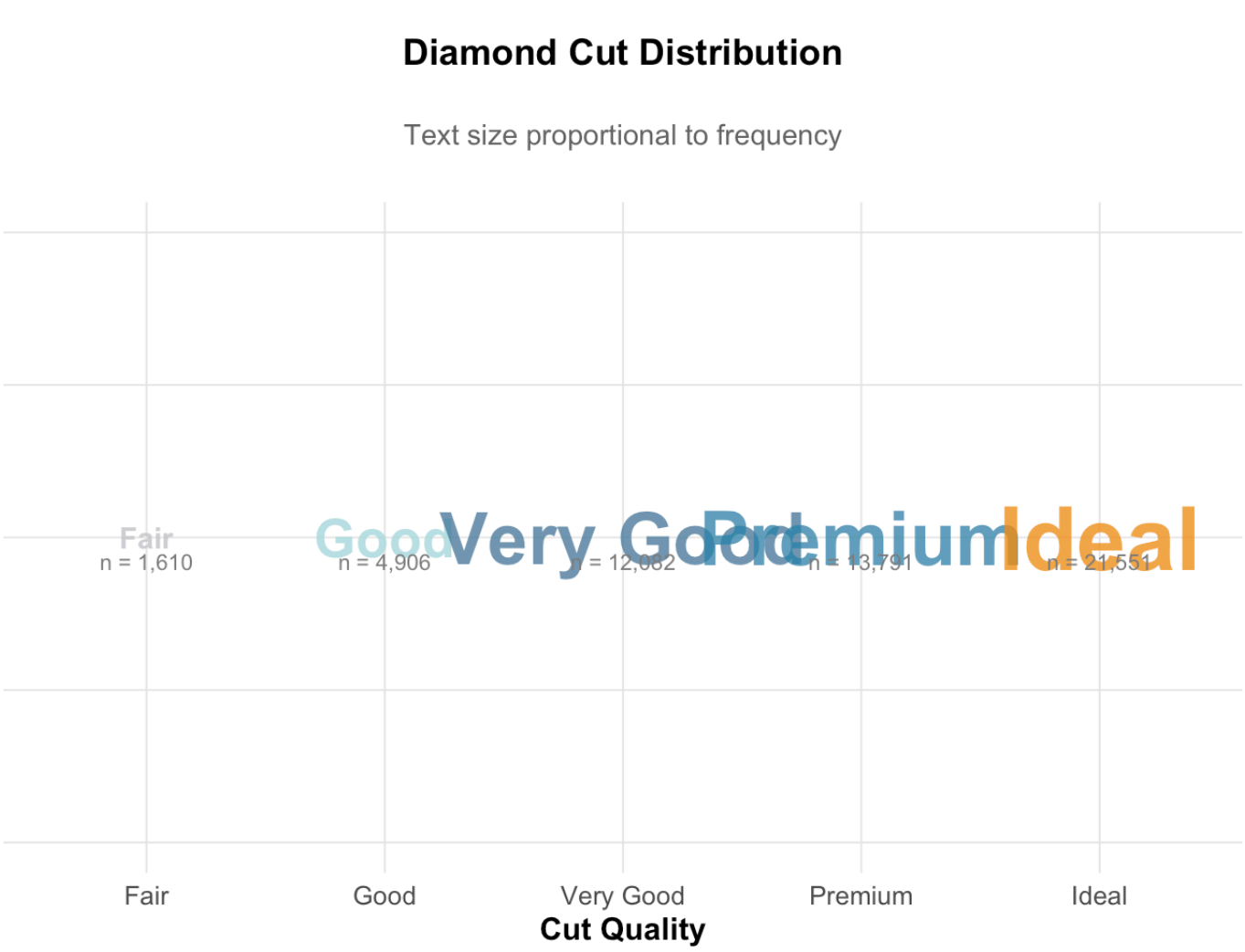
```
print(p3)
```


Diamond Cut Quality Proportions
Relative distribution as percentages
Ideal
40.0%
Premium
25.6%
Very Good
22.4%
Good
9.1%

```
print(p4)
```


Diamond Cut Distribution
Bubble size represents count of diamonds
1,610
4,906
12,082
13,791
21,551
Fair
Good
Very Good
Premium
Ideal
Cut Quality

```
print(p5)
```



Color

Based on the analysis below, it is clear that color is not uniformly distributed, but is much more uniform than cut. Instead, G is the most common at 20.9% of all observations, followed by E (18.2%) and F (17.7%) respectively.

```
#Prepping a nice color scheme for color levels
color_colors <- c("D" = "#FF6B6B", "E" = "#4ECDC4", "F" = "#45B7D1", "G" = "#96CEB4",
                  "H" = "#FECA57", "I" = "#FF9FF3", "J" = "#54A0FF")

# Finding summary statistics
diamonds |>
  count(color) |>
  mutate(percentage = scales::percent(n / sum(n), accuracy = 0.1)) |>
  arrange(desc(n)) |>
  print()
```

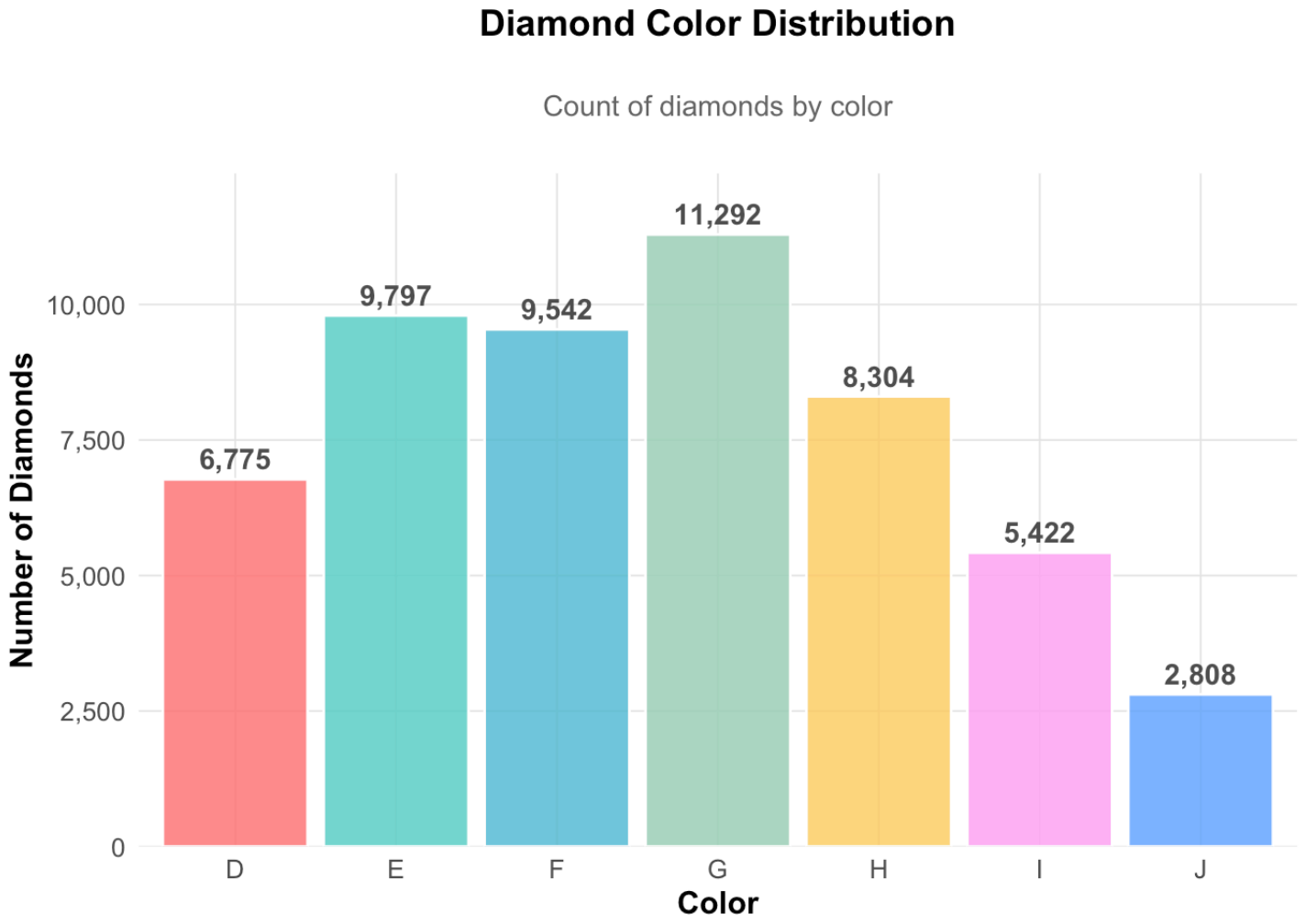
```
# A tibble: 7 × 3
  color     n percentage
  <ord> <int> <chr>
1 G     11292 20.9%
2 E      9797 18.2%
3 F      9542 17.7%
4 H      8304 15.4%
5 D      6775 12.6%
6 I      5422 10.1%
7 J      2808 5.2%
```

```
# 1. Making a bar chart
p1 <- ggplot(diamonds, aes(x = color)) +
  geom_bar(aes(fill = color),
           alpha = 0.8,
           color = "white",
           size = 0.5) +
  geom_text(stat = "count",
            aes(label = comma(after_stat(count))),
            vjust = -0.5,
            fontface = "bold",
            color = "gray30") +
  scale_fill_manual(values = color_colors) +
  labs(
    title = "Diamond Color Distribution",
    subtitle = "Count of diamonds by color",
    x = "Color",
    y = "Number of Diamonds"
  ) +
  custom_theme +
  theme(legend.position = "none") +
  scale_y_continuous(labels = comma_format(),
                     expand = expansion(mult = c(0, 0.1)))

# 2. Making a pie chart
p2 <- diamonds |>
  count(color) |>
  mutate(percentage = n / sum(n) * 100,
         label = paste0(color, "\n", round(percentage, 1), "%")) |>  # Changed 'cut' t
  ggplot(aes(x = "", y = n, fill = color)) +  # Changed 'cut' to 'color'
  geom_bar(stat = "identity",
           width = 1,
           color = "white",
```

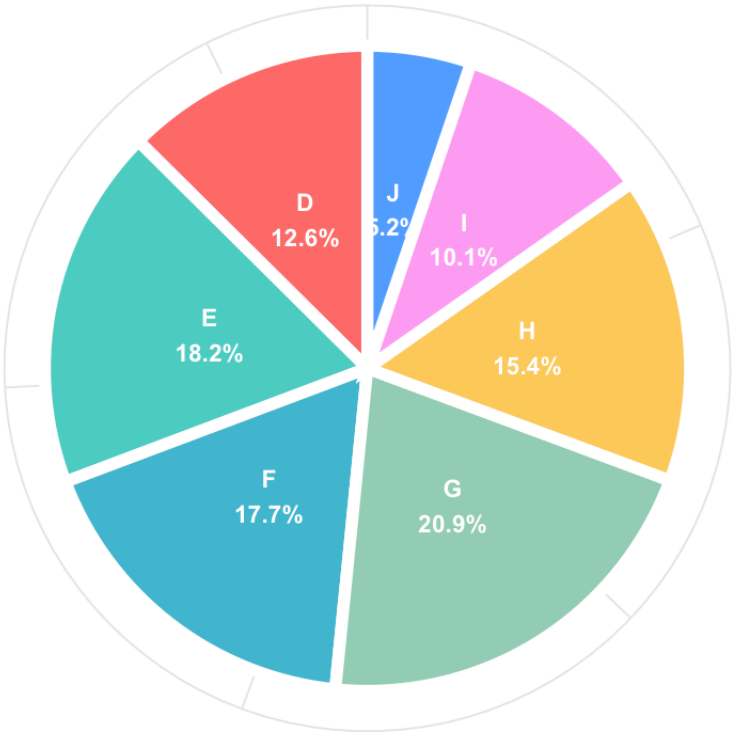
```
            size = 2) +
  geom_text(aes(label = label),
            position = position_stack(vjust = 0.5),
            fontface = "bold",
            color = "white",
            size = 3) +
  coord_polar("y", start = 0) +
  scale_fill_manual(values = color_colors) +  # This is correct
  labs(
    title = "Diamond Color Distribution",
    subtitle = "Proportional breakdown by color"
  ) +
  custom_theme +
  theme(
    axis.text = element_blank(),
    axis.ticks = element_blank(),
    axis.title = element_blank(),
    panel.grid = element_blank(),
    legend.position = "none"
  )

# 3. Making a stacked bar chart
p3 <- diamonds |>
  count(color) |>
  mutate(percentage = n / sum(n)) |>
  ggplot(aes(x = 1, y = percentage, fill = color)) +
  geom_bar(stat = "identity",
           color = "white",
           size = 1.5,
           width = 0.6) +
  geom_text(aes(label = paste0(color, "\n", scales::percent(percentage, accuracy = 0.1
            position = position_stack(vjust = 0.5),
            fontface = "bold",
            color = "white",
            size = 3.5) +
  scale_fill_manual(values = color_colors) +
  labs(
    title = "Diamond Color Proportions",
    subtitle = "Relative distribution as percentages"
  ) +
  custom_theme +
  theme(
    axis.text = element_blank(),
    axis.ticks = element_blank(),
    axis.title = element_blank(),
```

```
    panel.grid = element_blank(),
    legend.position = "none"
  ) +
  coord_flip()

# 4. Enhanced dot plot with size and color
p4 <- diamonds |>
  count(color) |>
  ggplot(aes(x = color, y = 1)) +
  geom_point(aes(size = n, color = color),
             alpha = 0.8,
             stroke = 2) +
  geom_text(aes(label = comma(n)),
            vjust = -2,
            fontface = "bold",
            color = "gray30") +
  scale_size_continuous(range = c(10, 30),
                        guide = "none") +
  scale_color_manual(values = color_colors) +
  labs(
    title = "Diamond Color Distribution",
    subtitle = "Bubble size represents count of diamonds",
    x = "Cut Quality",
    y = ""
  ) +
  custom_theme +
  theme(
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank(),
    panel.grid.major.y = element_blank(),
    legend.position = "none"
  ) +
  ylim(0.5, 1.5)

# 5. Cool text-based visualization
p5 <- diamonds |>
  count(color) |>
  ggplot(aes(x = color, y = 1)) +
  geom_text(aes(label = color,
                size = n,
                color = color),
            fontface = "bold",
            alpha = 0.8) +
  geom_text(aes(label = paste("n =", comma(n))),
            vjust = 2,
```

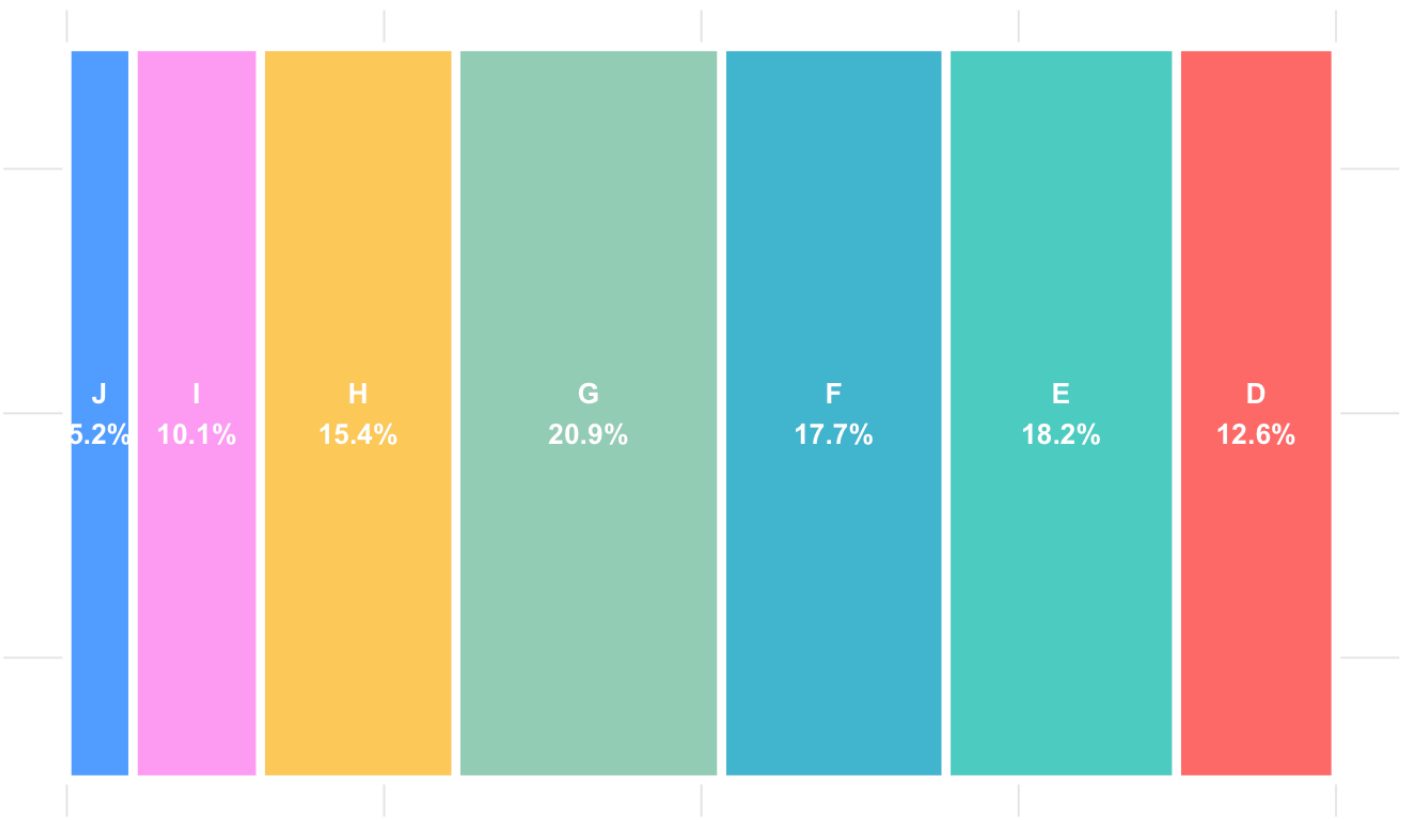
```
            color = "gray50",
            size = 3) +
  scale_size_continuous(range = c(4, 12), guide = "none") +
  scale_color_manual(values = color_colors) +
  labs(
    title = "Diamond Color Distribution",
    subtitle = "Text size proportional to frequency",
    x = "Color",
    y = ""
  ) +
  custom_theme +
  theme(
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank(),
    panel.grid = element_blank(),
    legend.position = "none"
  ) +
  ylim(0.5, 1.5)

# Display all plots
print(p1)
```


Diamond Color Distribution
Count of diamonds by color
Number of Diamonds
10,000
7,500
5,000
2,500
0
6,775
9,797
9,542
11,292
8,304
5,422
2,808
D
E
F
G
H
I
J
Color

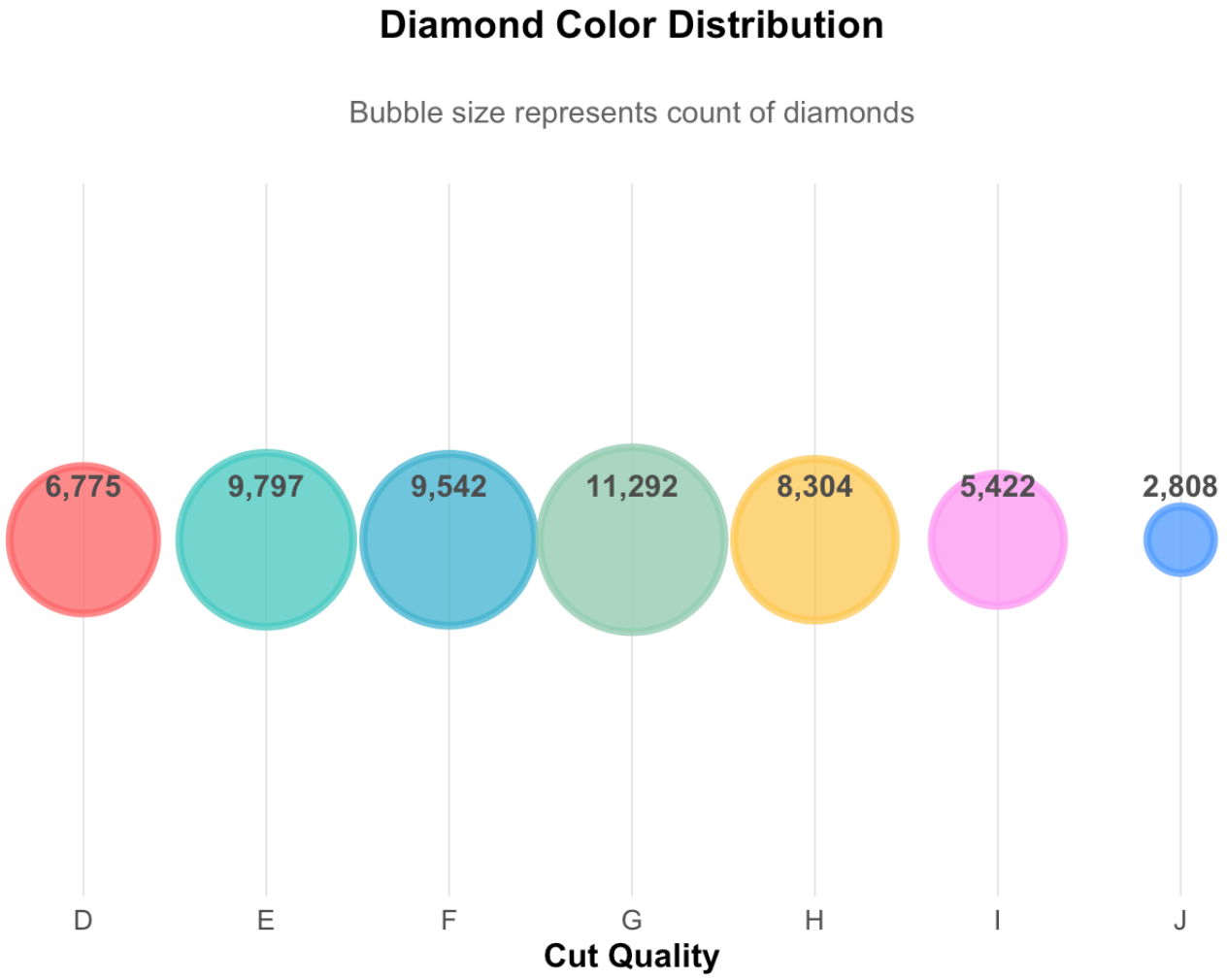
```
print(p2)
```


Diamond Color Distribution
Proportional breakdown by color
D
12.6%
J
5.2%
I
10.1%
H
15.4%
E
18.2%
F
17.7%
G
20.9%

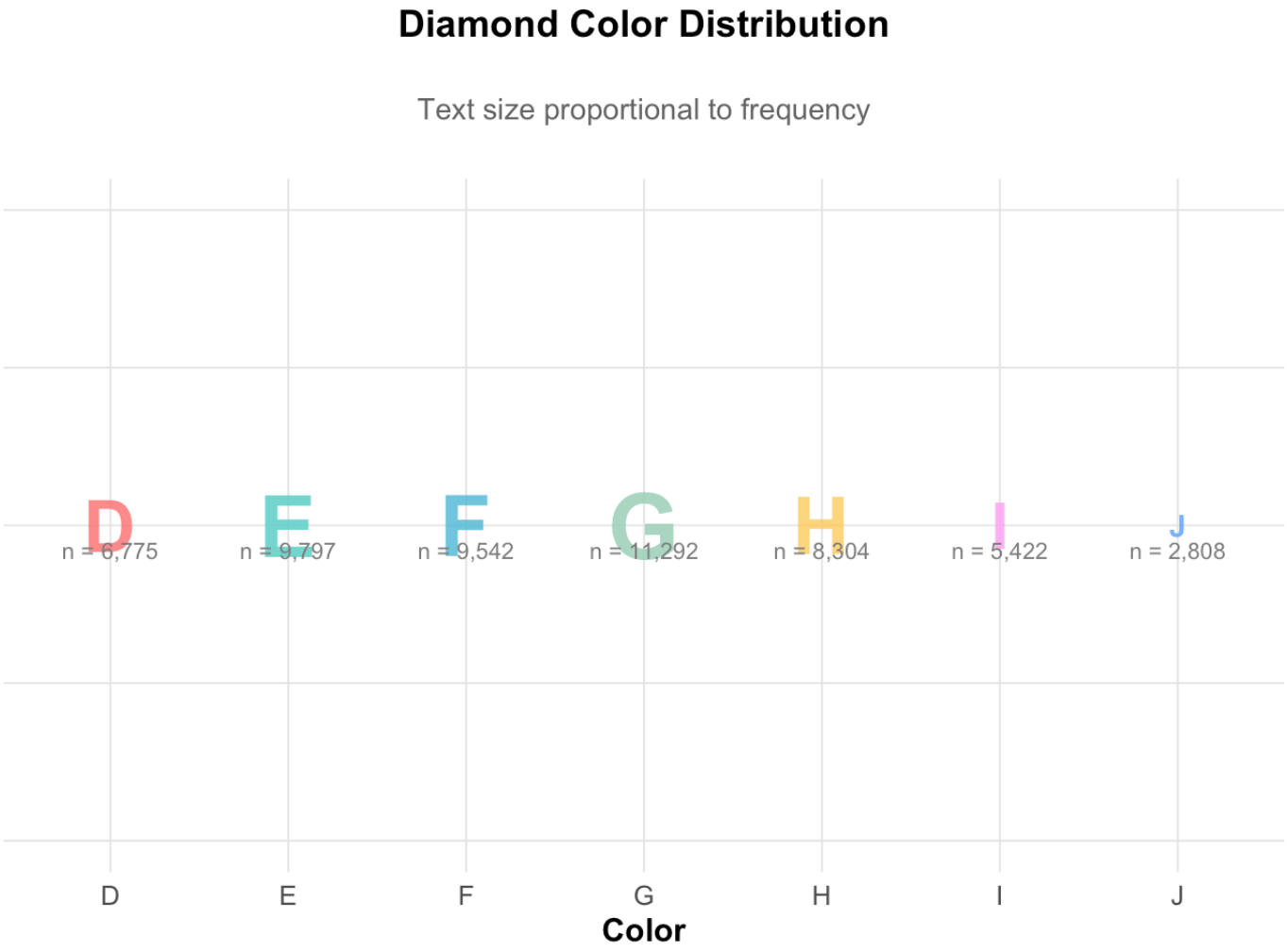
```
print(p3)
```


Diamond Color Proportions
Relative distribution as percentages
J
5.2%
I
10.1%
H
15.4%
G
20.9%
F
17.7%
E
18.2%
D
12.6%

```
print(p4)
```


Diamond Color Distribution
Bubble size represents count of diamonds
6,775
9,797
9,542
11,292
8,304
5,422
2,808
D
E
F
G
H
I
J
Cut Quality

```
print(p5)
```



Clarity

Based on the analysis below, it is clear that clarity is not uniformly distributed (though it is still semi-uniform). Instead, SI1 is the most common at 24.2% of all observations, followed by the second highest (22.7%) and third highest grades (17%) respectively.

```
#Setting up a clarity color scale
clarity_colors <- c("FL" = "#E8F4FD",
                    "IF" = "#B8E0D2",
                    "VVS1" = "#95D5B2",
                    "VVS2" = "#74C69D",
                    "VS1" = "#52B788",
                    "VS2" = "#40916C",
                    "SI1" = "#2D6A4F",
                    "SI2" = "#1B4332",
                    "I1" = "#081C15")
```

```
# Finding summary statistics
diamonds |>
  count(clarity) |>
  mutate(percentage = scales::percent(n / sum(n), accuracy = 0.1)) |>
  arrange(desc(n)) |>
  print()
```

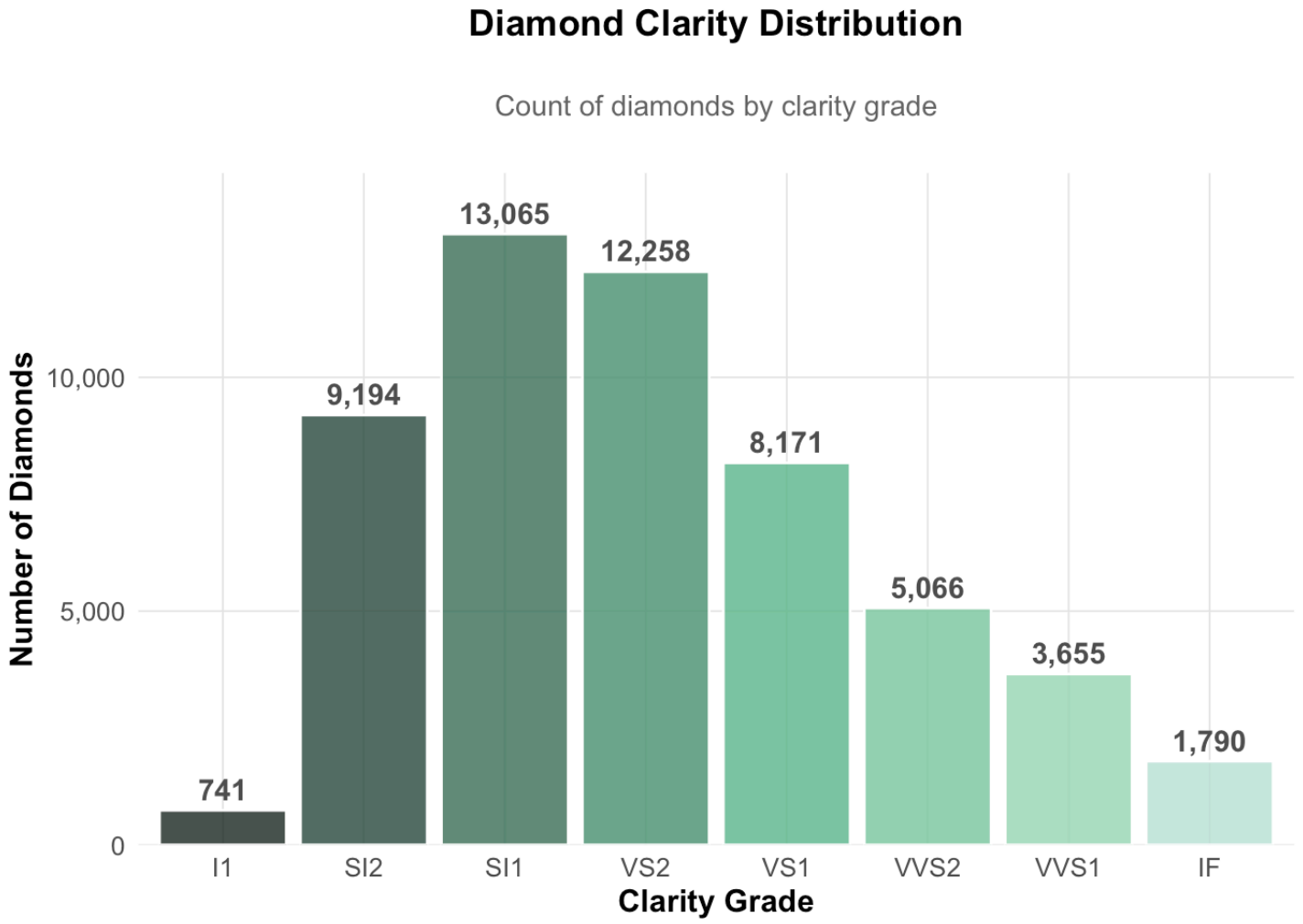
```
# A tibble: 8 × 3
  clarity     n percentage
  <ord>   <int> <chr>
1 SI1     13065 24.2%
2 VS2     12258 22.7%
3 SI2      9194 17.0%
4 VS1      8171 15.1%
5 VVS2     5066 9.4%
6 VVS1     3655 6.8%
7 IF       1790 3.3%
8 I1        741 1.4%
```

```
# 1. Making a bar chart
p1 <- ggplot(diamonds, aes(x = clarity)) +
  geom_bar(aes(fill = clarity),
           alpha = 0.8,
           color = "white",
           size = 0.5) +
  geom_text(stat = "count",
            aes(label = comma(after_stat(count))),
            vjust = -0.5,
            fontface = "bold",
            color = "gray30",
            size = 4) +
  scale_fill_manual(values = clarity_colors) +
  labs(
    title = "Diamond Clarity Distribution",
    subtitle = "Count of diamonds by clarity grade",
    x = "Clarity Grade",
    y = "Number of Diamonds"
  ) +
  custom_theme +
  theme(legend.position = "none") +
  scale_y_continuous(labels = comma_format(),
                     expand = expansion(mult = c(0, 0.1)))
```

```r
# 2. Making a pie chart
p2 <- diamonds |>
  count(clarity) |>
  mutate(percentage = n / sum(n) * 100,
         label = paste0(clarity, "\n", round(percentage, 1), "%")) |>
  ggplot(aes(x = "", y = n, fill = clarity)) +
  geom_bar(stat = "identity",
           width = 1,
           color = "white",
           size = 3) +
  geom_text(aes(label = label),
            position = position_stack(vjust = 0.5),
            fontface = "bold",
            color = "white",
            size = 4.5) +
  coord_polar("y", start = 0) +
  scale_fill_manual(values = clarity_colors) +
  labs(
    title = "Diamond Clarity Distribution",
    subtitle = "Proportional breakdown by clarity grade"
  ) +
  custom_theme +
  theme(
    axis.text = element_blank(),
    axis.ticks = element_blank(),
    axis.title = element_blank(),
    panel.grid = element_blank(),
    legend.position = "none",
    plot.title = element_text(size = 16, face = "bold", hjust = 0.5),
    plot.subtitle = element_text(size = 13, hjust = 0.5, color = "gray40")
  )

# 3. Making a stacked bar chart
p3 <- diamonds |>
  count(clarity) |>
  mutate(percentage = n / sum(n)) |>
  ggplot(aes(x = 1, y = percentage, fill = clarity)) +
  geom_bar(stat = "identity",
           color = "white",
           size = 1.5,
           width = 0.6) +
  geom_text(aes(label = paste0(clarity, "\n", scales::percent(percentage, accuracy = 0
            position = position_stack(vjust = 0.5),
            fontface = "bold",
            color = "white",
```

```
            size = 3.5) +
  scale_fill_manual(values = clarity_colors) +
  labs(
    title = "Diamond Clarity Proportions",
    subtitle = "Relative distribution as percentages"
  ) +
  custom_theme +
  theme(
    axis.text = element_blank(),
    axis.ticks = element_blank(),
    axis.title = element_blank(),
    panel.grid = element_blank(),
    legend.position = "none"
  ) +
  coord_flip()

# 4. Enhanced dot plot with size and color
p4 <- diamonds |>
  count(clarity) |>
  ggplot(aes(x = clarity, y = 1)) +
  geom_point(aes(size = n, color = clarity),
             alpha = 0.8,
             stroke = 2) +
  geom_text(aes(label = comma(n)),
            vjust = -2,
            fontface = "bold",
            color = "gray30") +
  scale_size_continuous(range = c(8, 25),
                        guide = "none") +
  scale_color_manual(values = clarity_colors) +
  labs(
    title = "Diamond Clarity Distribution",
    subtitle = "Bubble size represents count of diamonds",
    x = "Clarity Grade",
    y = ""
  ) +
  custom_theme +
  theme(
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank(),
    panel.grid.major.y = element_blank(),
    legend.position = "none"
  ) +
  ylim(0.5, 1.5)
```

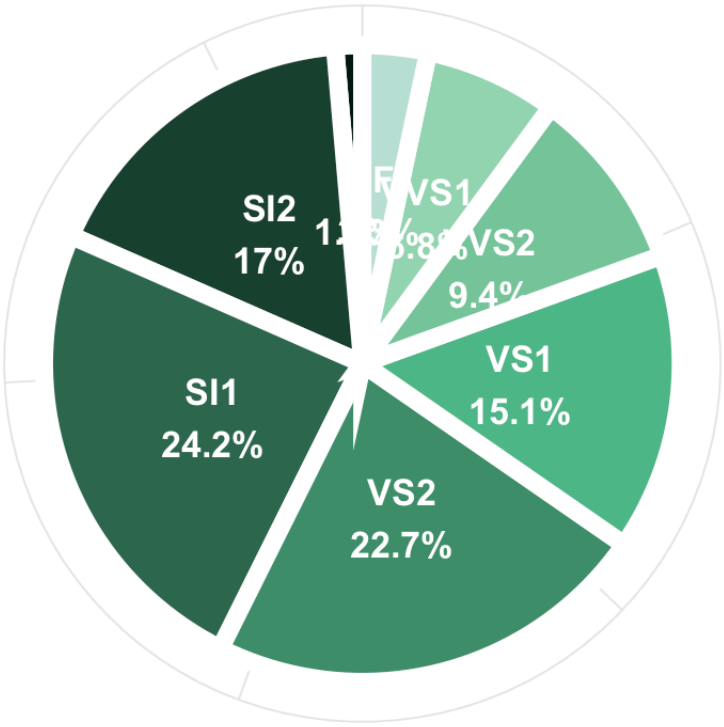
```r
# 5. Cool text-based visualization
p5 <- diamonds |>
  count(clarity) |>
  ggplot(aes(x = clarity, y = 1)) +
  geom_text(aes(label = clarity,
                size = n,
                color = clarity),
            fontface = "bold",
            alpha = 0.9) +
  geom_text(aes(label = paste("n =", comma(n))),
            vjust = 2.5,
            color = "gray50",
            size = 3) +
  scale_size_continuous(range = c(3, 8), guide = "none") +
  scale_color_manual(values = clarity_colors) +
  labs(
    title = "Diamond Clarity Distribution",
    subtitle = "Text size proportional to frequency",
    x = "Clarity Grade",
    y = ""
  ) +
  custom_theme +
  theme(
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank(),
    panel.grid = element_blank(),
    legend.position = "none"
  ) +
  ylim(0.5, 1.5)

# Display all plots
print(p1)
```


Diamond Clarity Distribution
Count of diamonds by clarity grade
Number of Diamonds
10,000
5,000
0
741
9,194
13,065
12,258
8,171
5,066
3,655
1,790
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF
Clarity Grade

```
print(p2)
```

Diamond Clarity Distribution

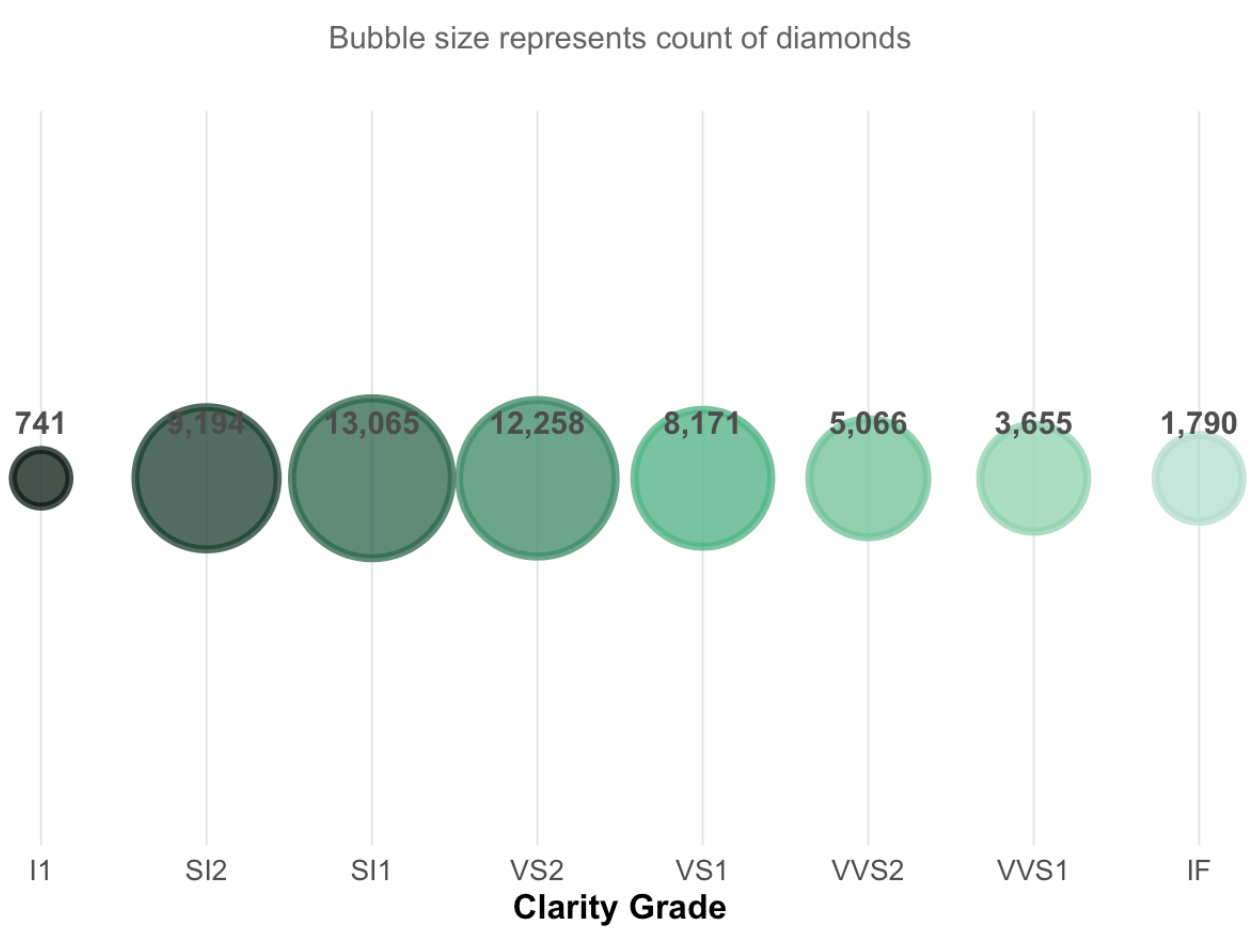
Proportional breakdown by clarity grade



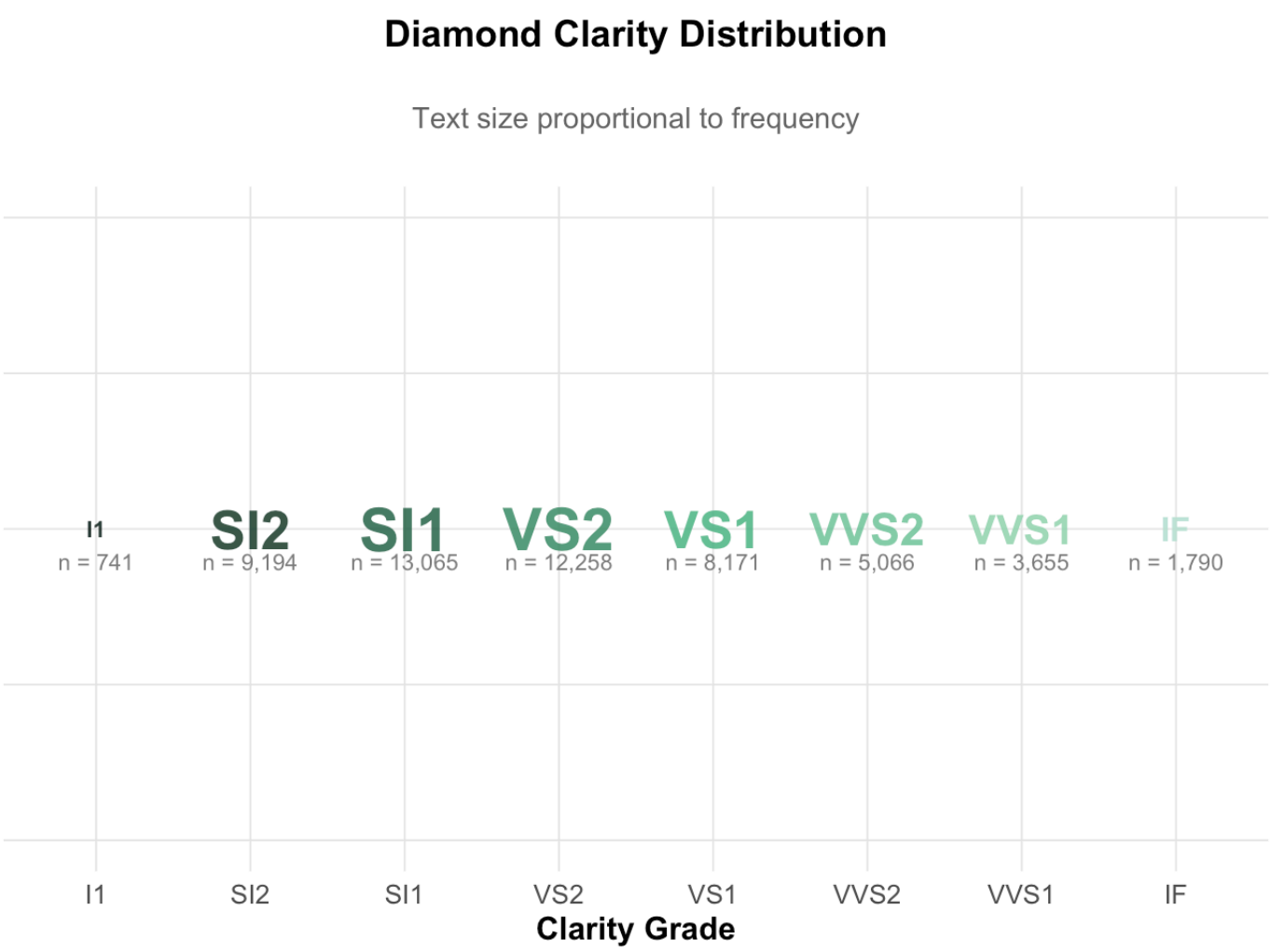

```
print(p3)
```


Diamond Clarity Proportions
Relative distribution as percentages
IF
.3%
VVS1
6.8%
VVS2
9.4%
VS1
15.1%
VS2
22.7%
SI1
24.2%
SI2
17.0%

```
print(p4)
```


Diamond Clarity Distribution
Bubble size represents count of diamonds
741
9,194
13,065
12,258
8,171
5,066
3,655
1,790
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF
Clarity Grade

```
print(p5)
```



Depth

Based on the analysis below, it is clear that carat is nearly normal and unipolar, with a mean of 61.75 and standard deviation of 1.432621.

```
summary(diamonds$depth)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  43.00   61.00   61.80   61.75   62.50   79.00 
```

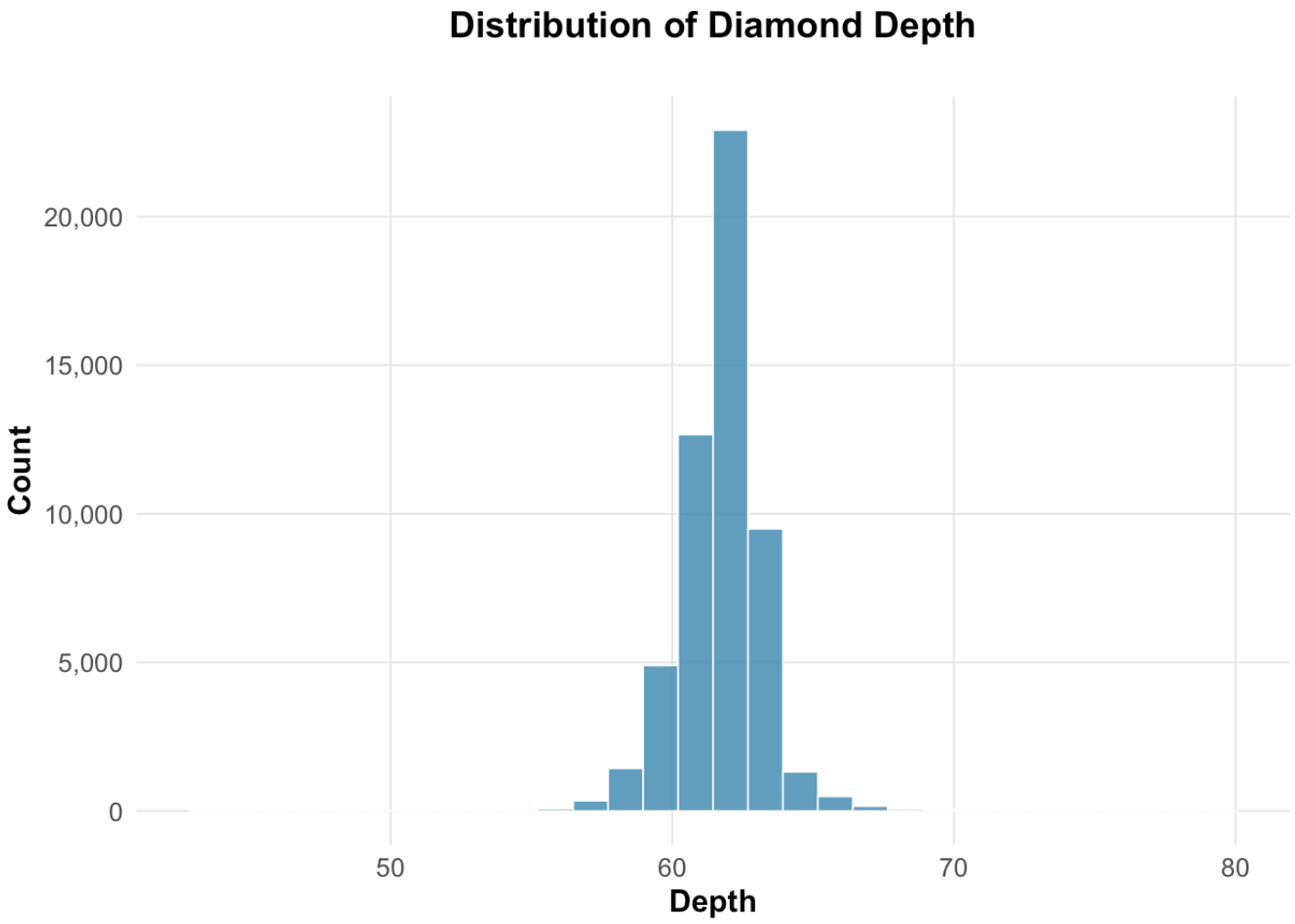
```
sd(diamonds$depth)
```

```
[1] 1.432621
```

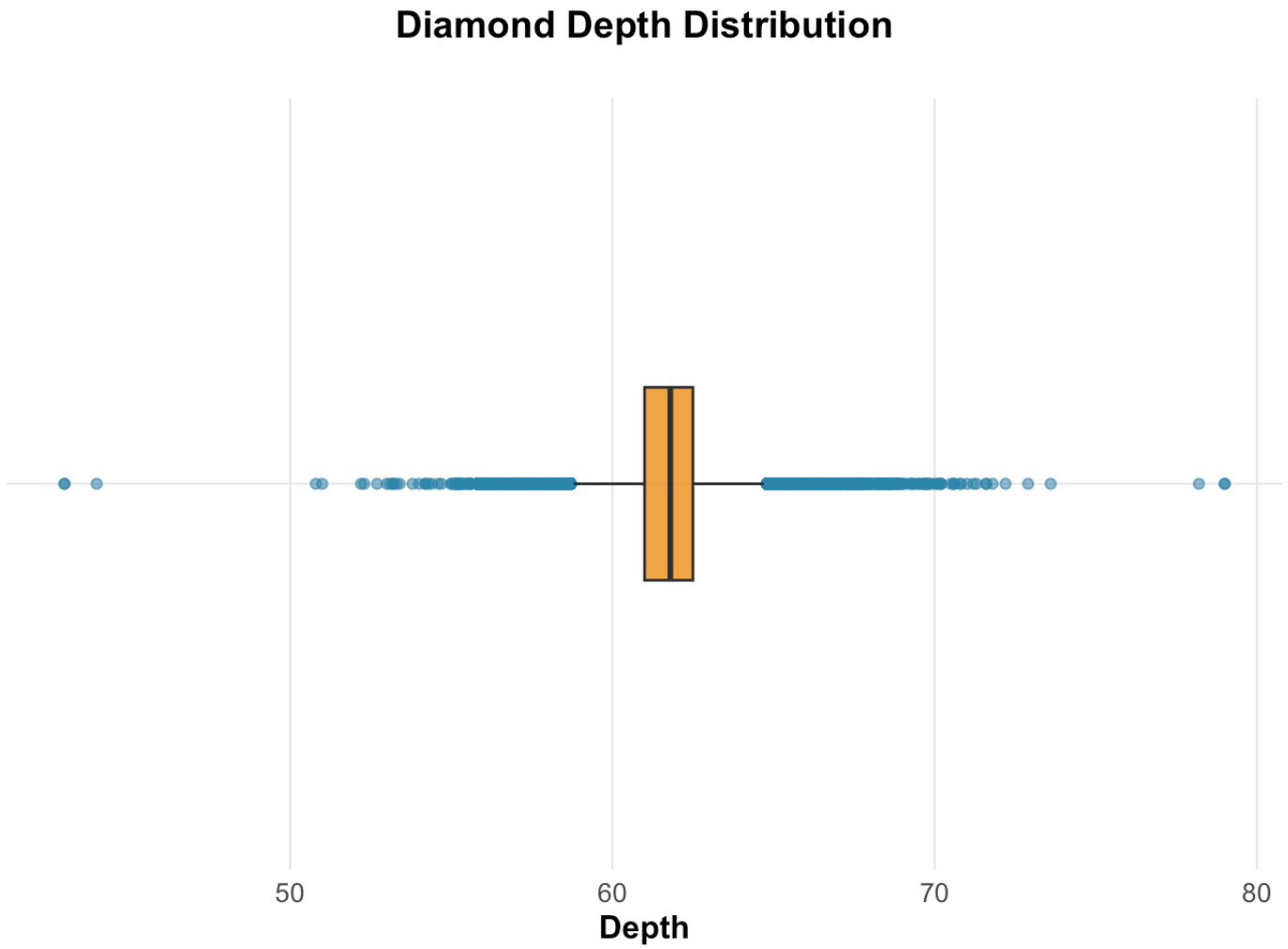
```
# Creating a Histogram
```

```
p1 <- ggplot(diamonds, aes(x = depth)) +
  geom_histogram(bins = 30,
                 fill = diamond_blue,
                 color = "white",
                 alpha = 0.8,
                 size = 0.3) +
  labs(
    title = "Distribution of Diamond Depth",
    x = "Depth",
    y = "Count"
  ) +
  custom_theme +
  scale_y_continuous(labels = scales::comma_format())

# Creating a boxplot
p2 <- ggplot(diamonds, aes(x = depth, y = "")) +
  geom_boxplot(width = 0.3,
               fill = diamond_gold,
               alpha = 0.8,
               outlier.color = diamond_blue,
               outlier.alpha = 0.6) +
  labs(
    title = "Diamond Depth Distribution",
    x = "Depth",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())

# Creating a nice density plot
p3 <- ggplot(diamonds, aes(x = depth)) +
  geom_density(fill = diamond_blue,
               alpha = 0.7,
               color = diamond_blue,
               size = 1.2) +
  labs(
    title = "Density Distribution of Diamond Depth",
    x = "Depth",
    y = "Density"
  ) +
  custom_theme

# Creating a nice ECDF (empirical cumulative distribution function)
p4 <- ggplot(diamonds, aes(x = depth)) +
```

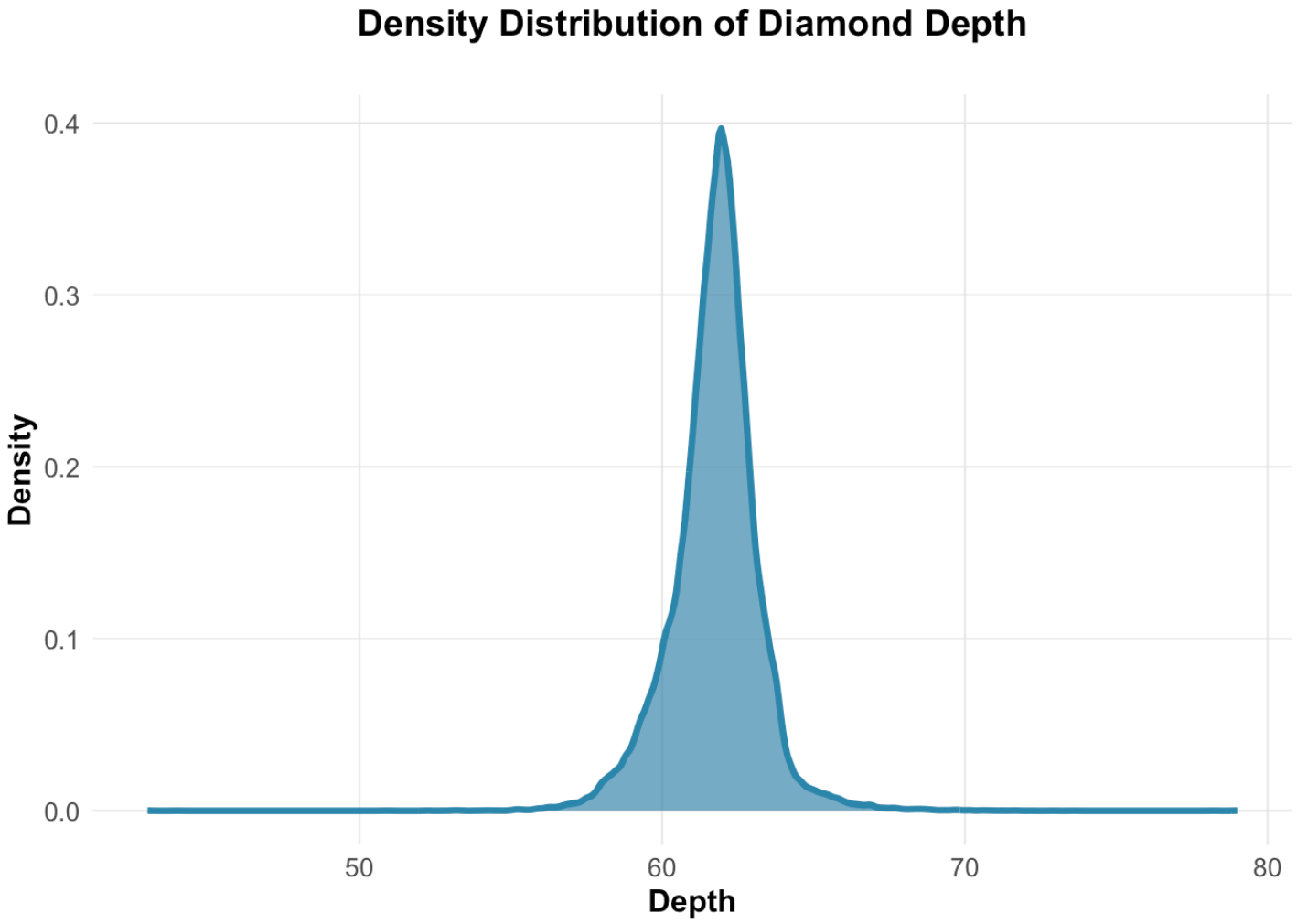
```
  stat_ecdf(color = diamond_blue,
            size = 1.2,
            alpha = 0.8) +
  geom_hline(yintercept = c(0.25, 0.5, 0.75),
             linetype = "dotted",
             color = diamond_gold,
             alpha = 0.7) +
  labs(
    title = "Cumulative Distribution of Diamond Depth",
    x = "Depth",
    y = "Cumulative Probability"
  ) +
  custom_theme +
  scale_y_continuous(labels = scales::percent_format())

# Displaying all plots
print(p1)
```



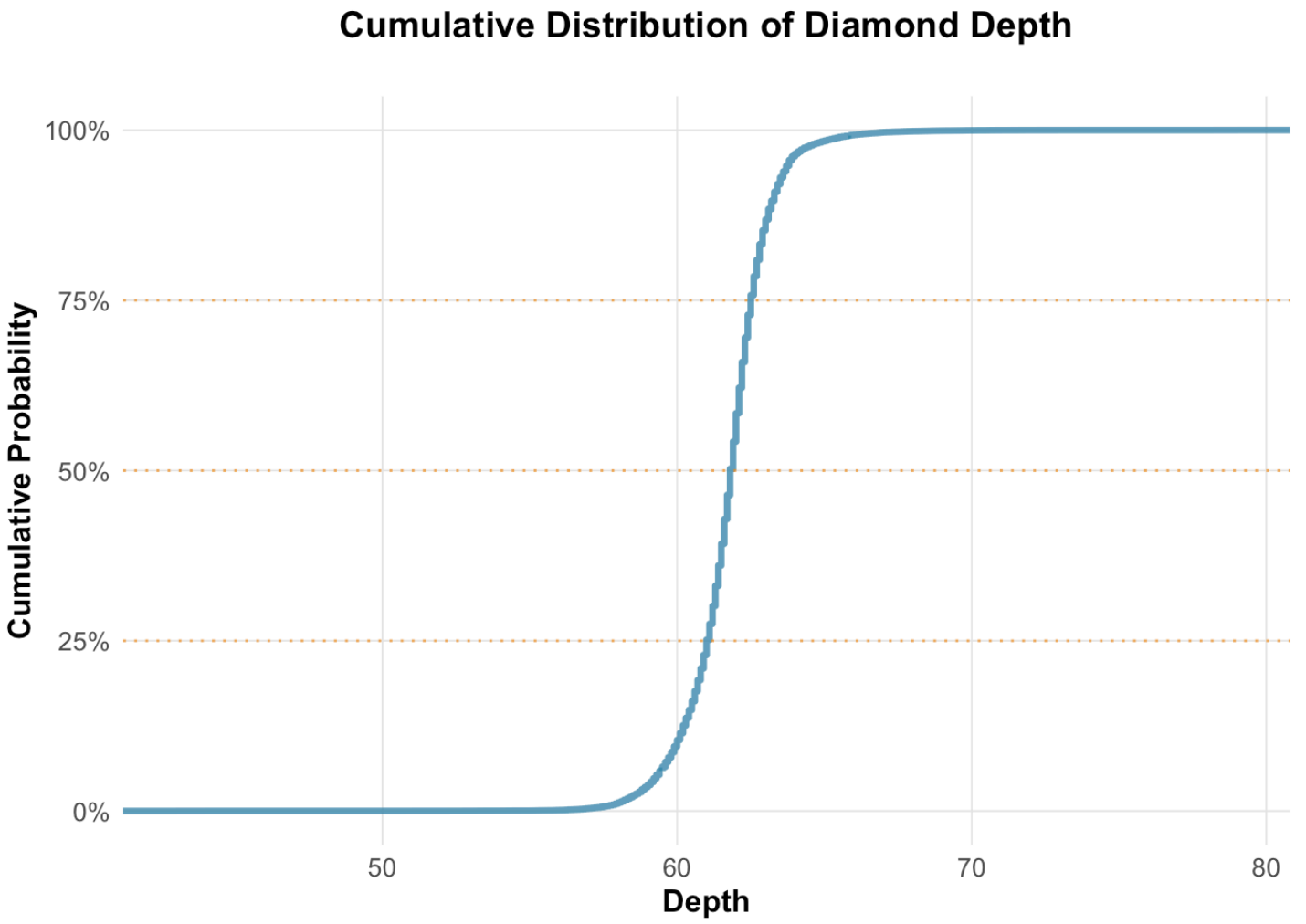
```
print(p2)
```




```
print(p3)
```


Density Distribution of Diamond Depth
Density
0.4
0.3
0.2
0.1
0.0
50
60
70
80
Depth

```
print(p4)
```



Table

Based on the analysis below, it is clear that carat is mostly normal and highly multipolar, with a mean of 57.46 and standard deviation of 2.234491.

```
summary(diamonds$table)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  43.00   56.00   57.00   57.46   59.00   95.00 
```

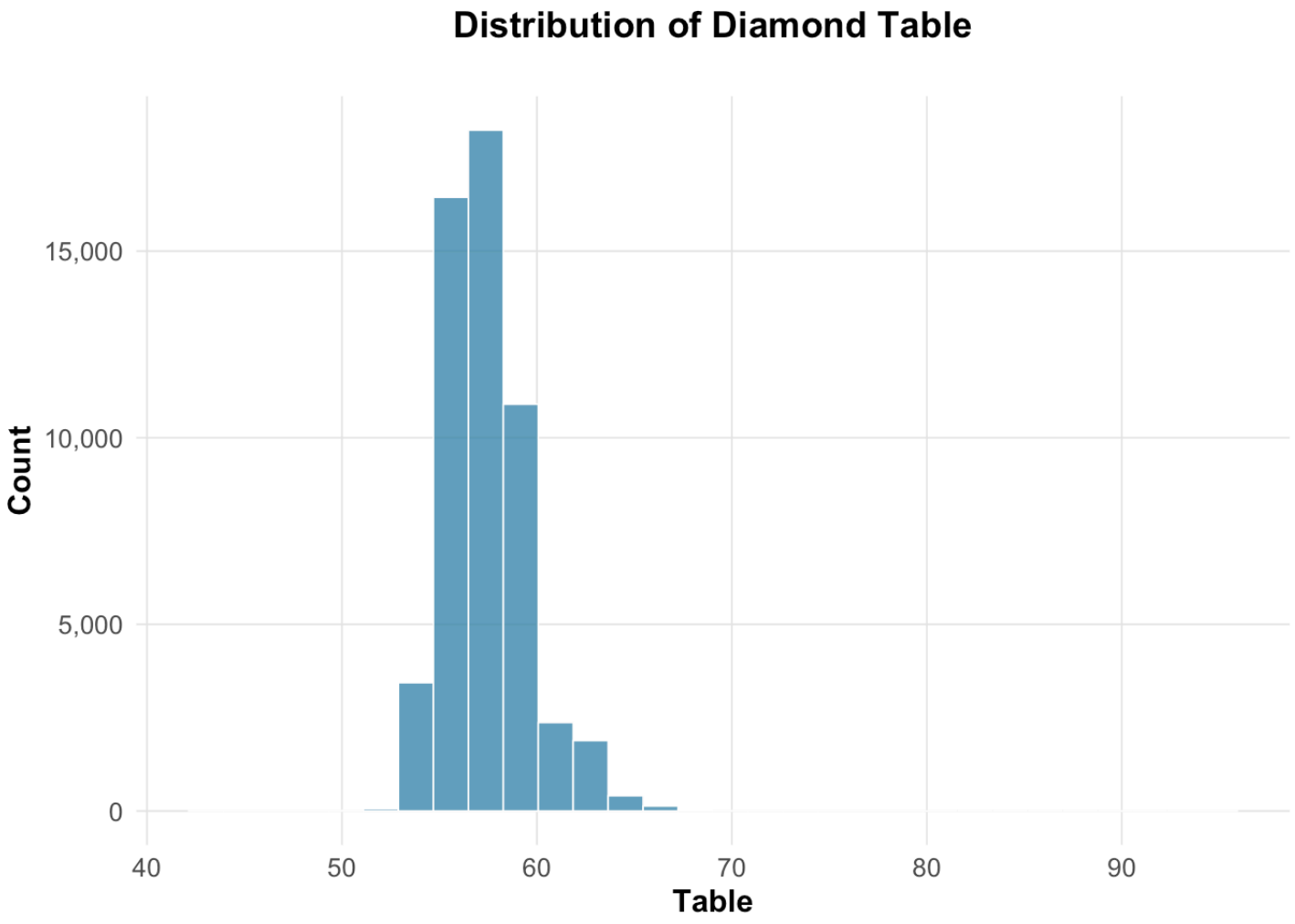
```
sd(diamonds$table)
```

```
[1] 2.234491
```

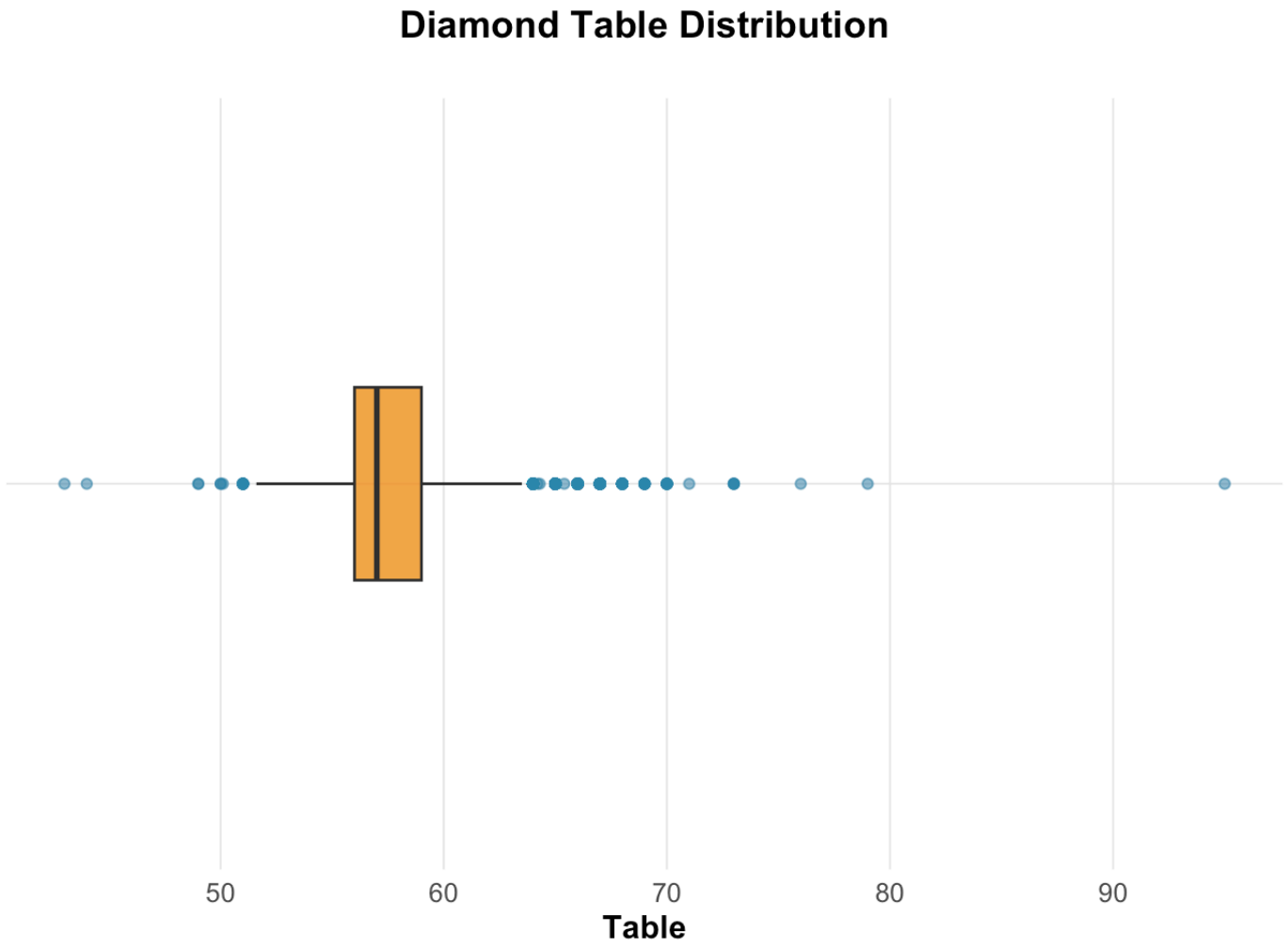
```
# Creating a Histogram
```

```r
p1 <- ggplot(diamonds, aes(x = table)) +
  geom_histogram(bins = 30,
                 fill = diamond_blue,
                 color = "white",
                 alpha = 0.8,
                 size = 0.3) +
  labs(
    title = "Distribution of Diamond Table",
    x = "Table",
    y = "Count"
  ) +
  custom_theme +
  scale_y_continuous(labels = scales::comma_format())

# Creating a boxplot
p2 <- ggplot(diamonds, aes(x = table, y = "")) +
  geom_boxplot(width = 0.3,
               fill = diamond_gold,
               alpha = 0.8,
               outlier.color = diamond_blue,
               outlier.alpha = 0.6) +
  labs(
    title = "Diamond Table Distribution",
    x = "Table",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())

# Creating a nice density plot
p3 <- ggplot(diamonds, aes(x = table)) +
  geom_density(fill = diamond_blue,
               alpha = 0.7,
               color = diamond_blue,
               size = 1.2) +
  labs(
    title = "Density Distribution of Diamond Table",
    x = "Table",
    y = "Density"
  ) +
  custom_theme

# Creating a nice ECDF (empirical cumulative distribution function)
p4 <- ggplot(diamonds, aes(x = table)) +
```

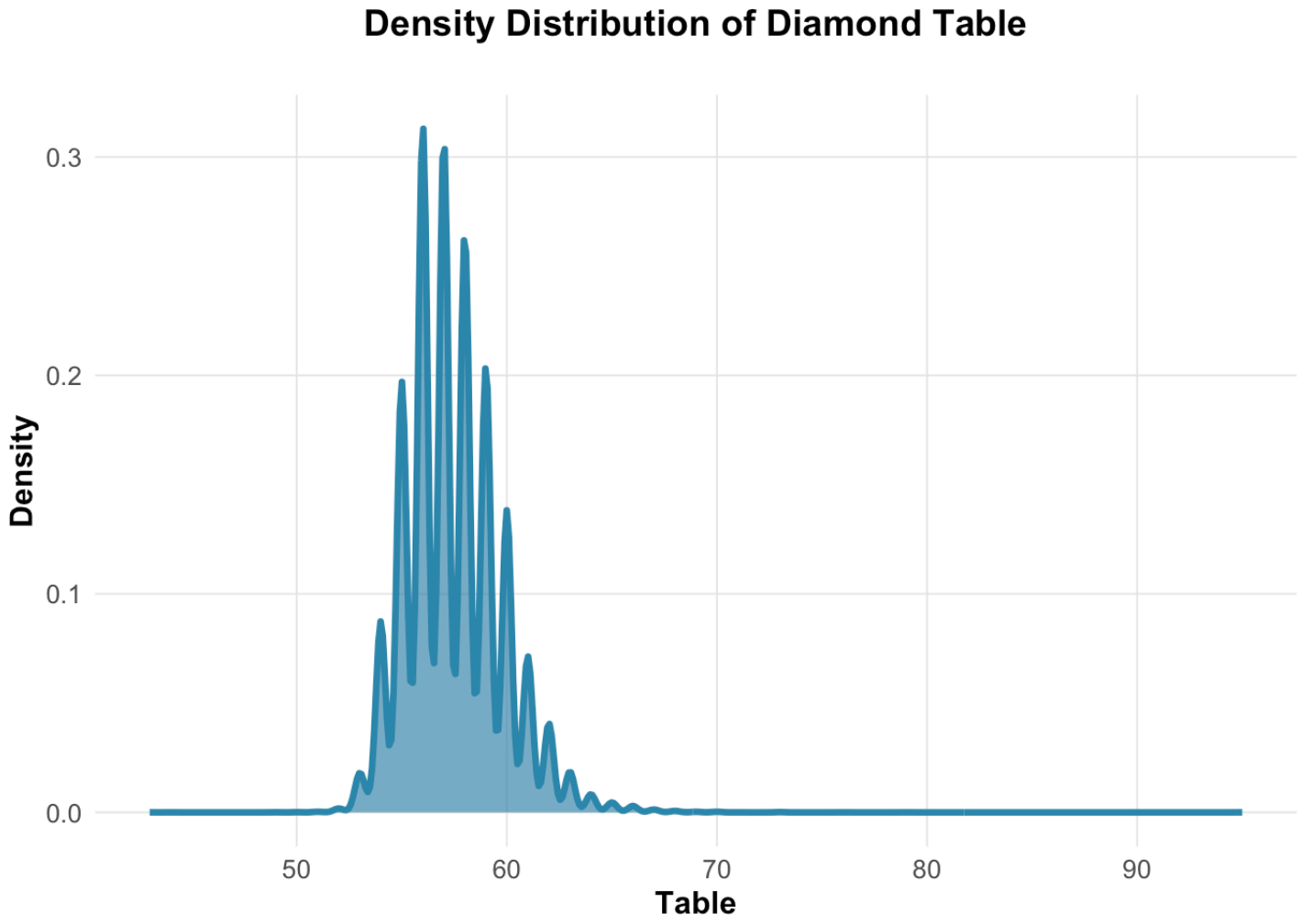
```
  stat_ecdf(color = diamond_blue,
            size = 1.2,
            alpha = 0.8) +
  geom_hline(yintercept = c(0.25, 0.5, 0.75),
             linetype = "dotted",
             color = diamond_gold,
             alpha = 0.7) +
  labs(
    title = "Cumulative Distribution of Table",
    x = "Table",
    y = "Cumulative Probability"
  ) +
  custom_theme +
  scale_y_continuous(labels = scales::percent_format())

# Displaying all plots
print(p1)
```



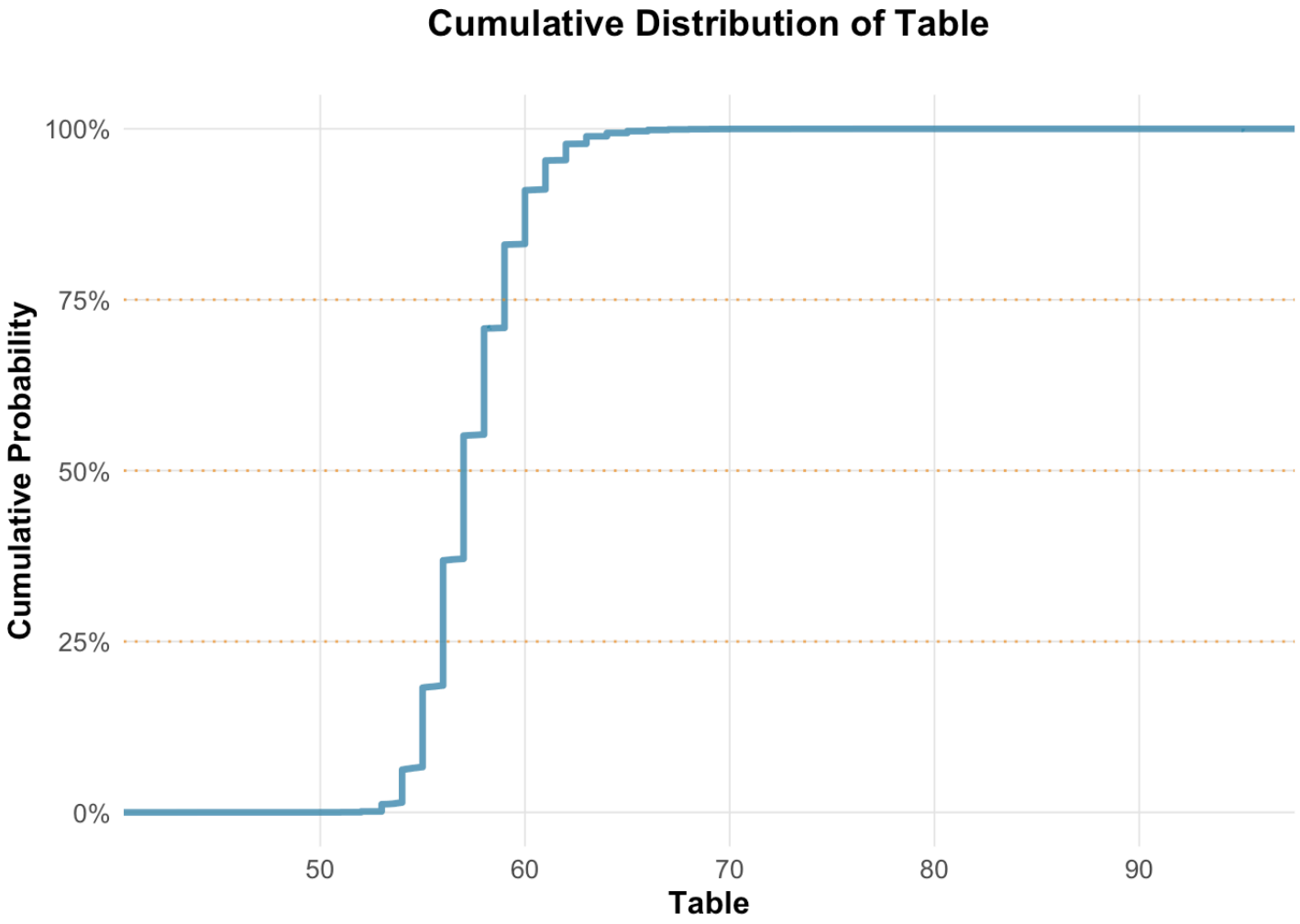
```
print(p2)
```




```
print(p3)
```


Density Distribution of Diamond Table
0.3
0.2
0.1
0.0
Density
50
60
70
80
90
Table

```
print(p4)
```



Price

Based on the analysis below, it is clear that price is very right tailed yet unipolar (or perhaps bipolar)b, with a mean of 3933 and standard deviation of 3989.44.

```
summary(diamonds$price)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    326     950    2401    3933    5324   18823 
```

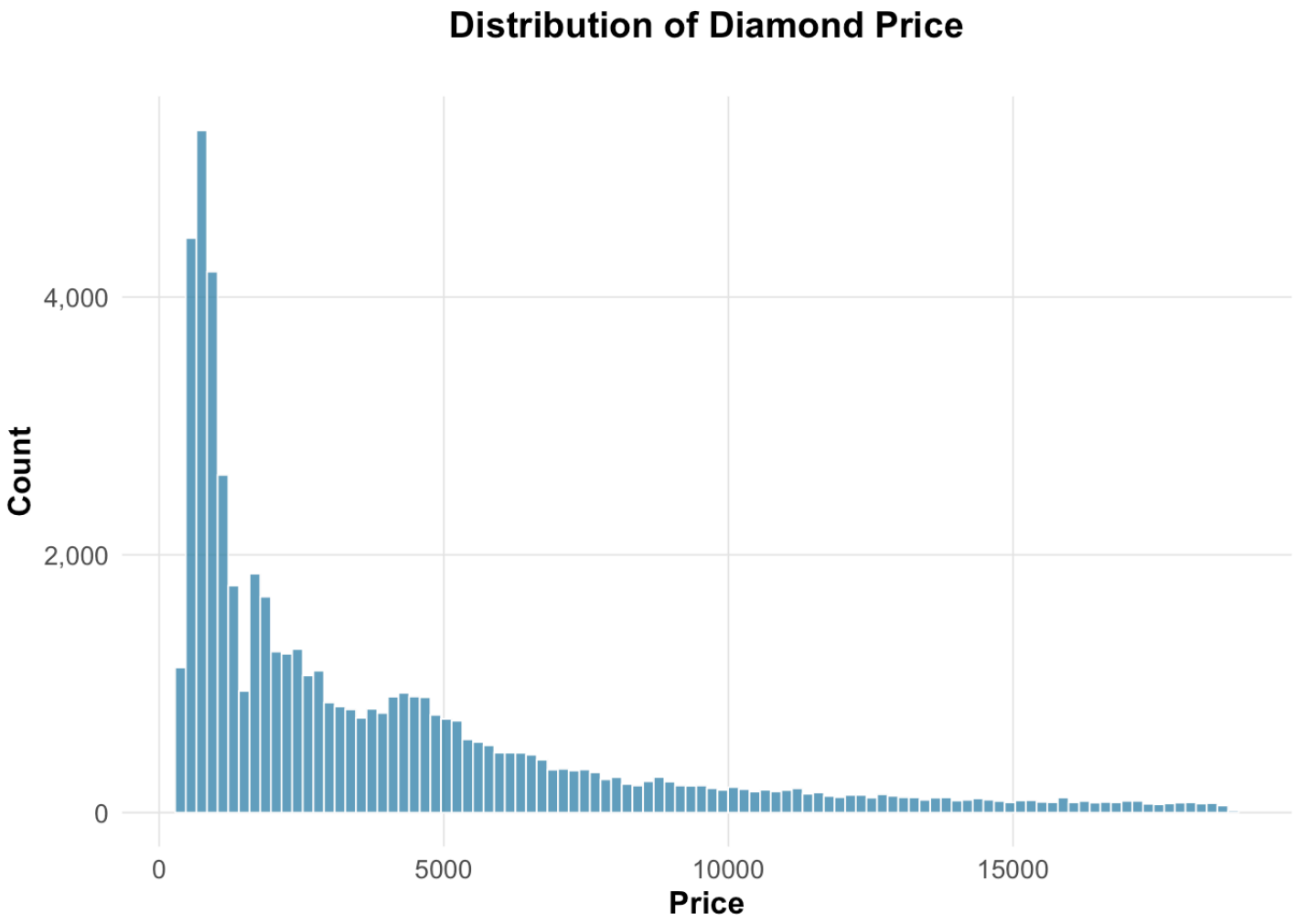
```
sd(diamonds$price)
```

```
[1] 3989.44
```

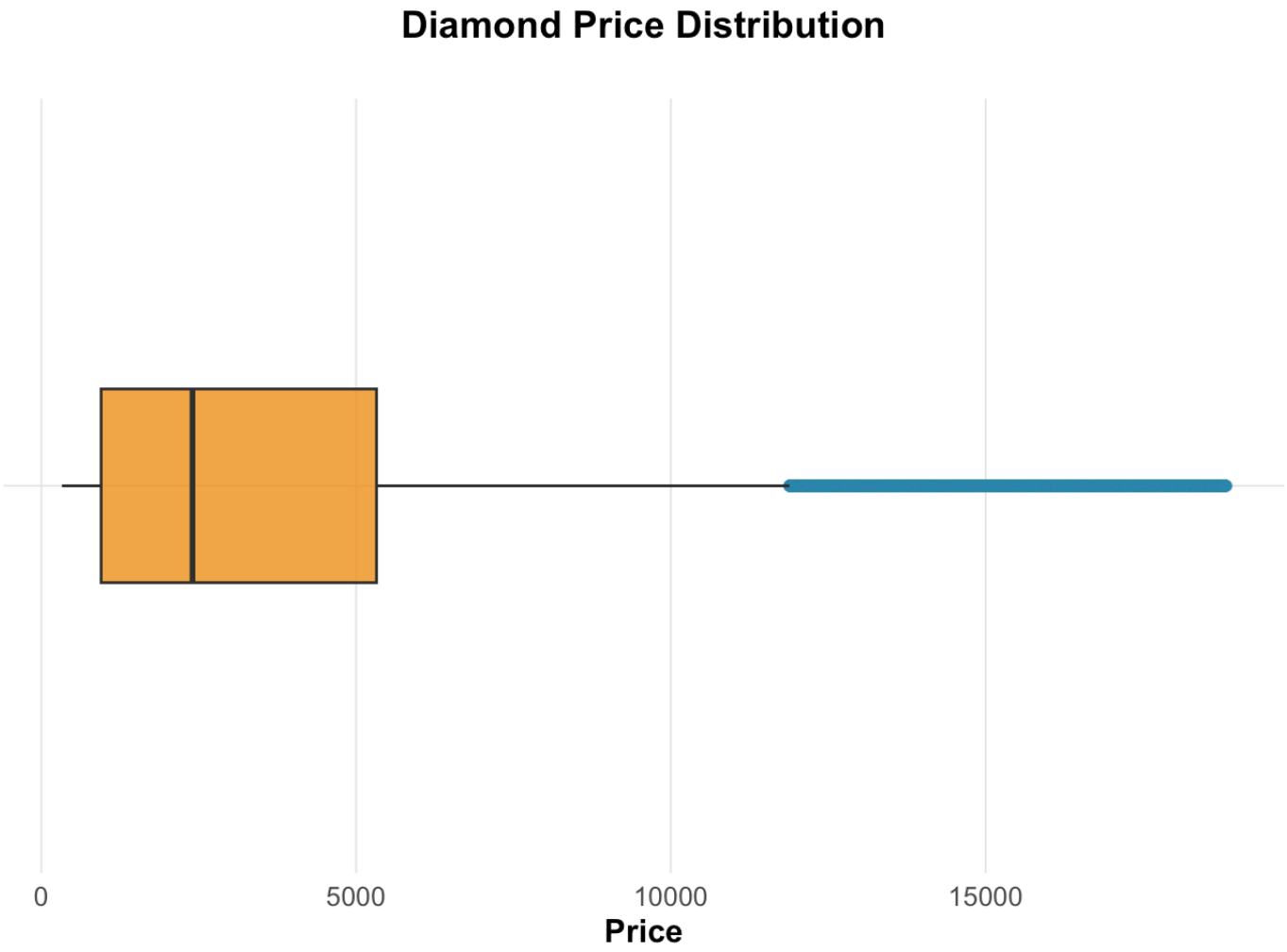
```
# Creating a Histogram
```

```
p1 <- ggplot(diamonds, aes(x = price)) +
  geom_histogram(bins = 100,
                 fill = diamond_blue,
                 color = "white",
                 alpha = 0.8,
                 size = 0.3) +
  labs(
    title = "Distribution of Diamond Price",
    x = "Price",
    y = "Count"
  ) +
  custom_theme +
  scale_y_continuous(labels = scales::comma_format())

# Creating a boxplot
p2 <- ggplot(diamonds, aes(x = price, y = "")) +
  geom_boxplot(width = 0.3,
               fill = diamond_gold,
               alpha = 0.8,
               outlier.color = diamond_blue,
               outlier.alpha = 0.6) +
  labs(
    title = "Diamond Price Distribution",
    x = "Price",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())

# Creating a nice density plot
p3 <- ggplot(diamonds, aes(x = price)) +
  geom_density(fill = diamond_blue,
               alpha = 0.7,
               color = diamond_blue,
               size = 1.2) +
  labs(
    title = "Density Distribution of Diamond Price",
    x = "Price",
    y = "Density"
  ) +
  custom_theme

# Creating a nice ECDF (empirical cumulative distribution function)
p4 <- ggplot(diamonds, aes(x = price)) +
```

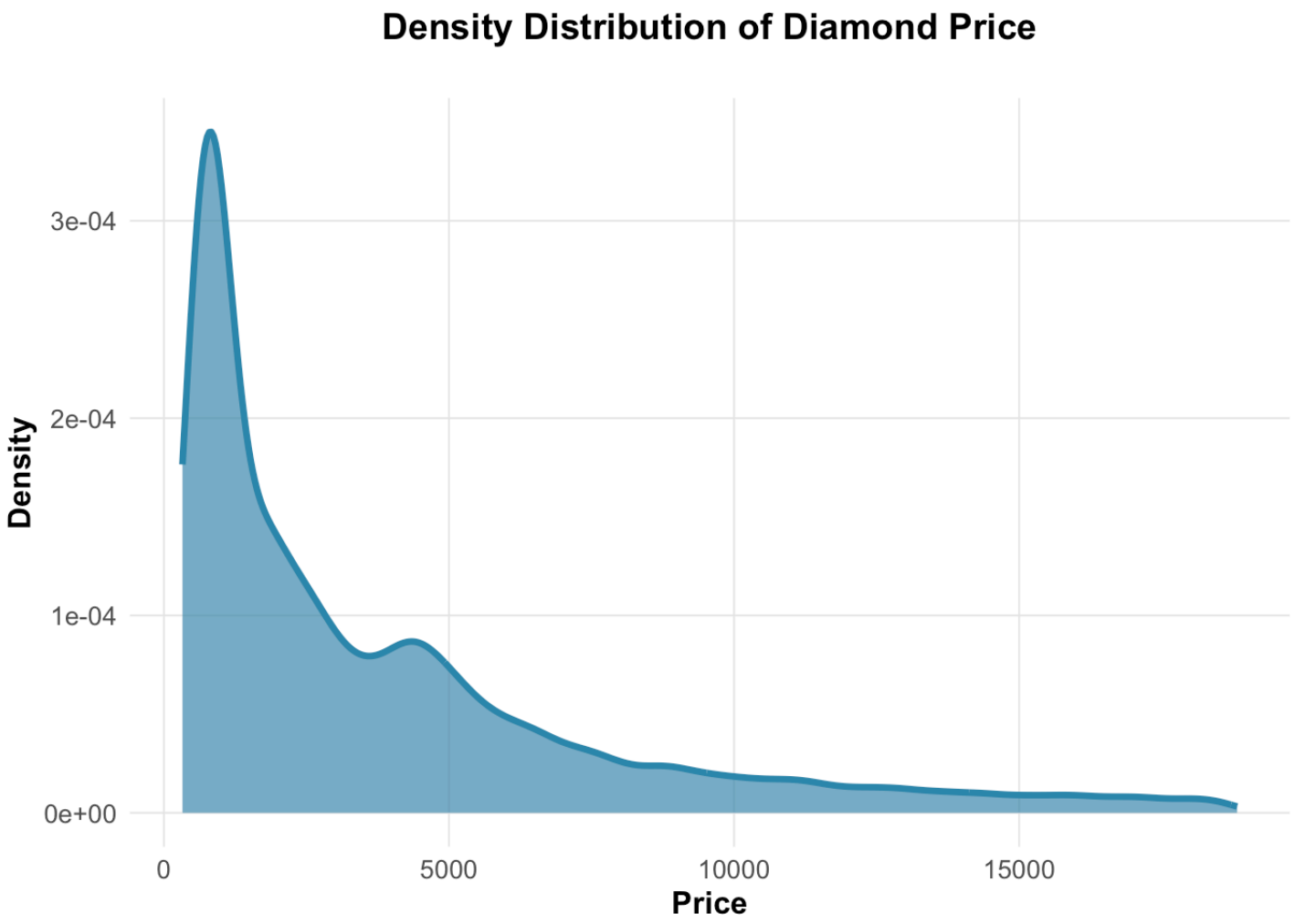
```
  stat_ecdf(color = diamond_blue,
            size = 1.2,
            alpha = 0.8) +
  geom_hline(yintercept = c(0.25, 0.5, 0.75),
             linetype = "dotted",
             color = diamond_gold,
             alpha = 0.7) +
  labs(
    title = "Cumulative Distribution of Diamond Price",
    x = "Price",
    y = "Cumulative Probability"
  ) +
  custom_theme +
  scale_y_continuous(labels = scales::percent_format())

# Displaying all plots
print(p1)
```



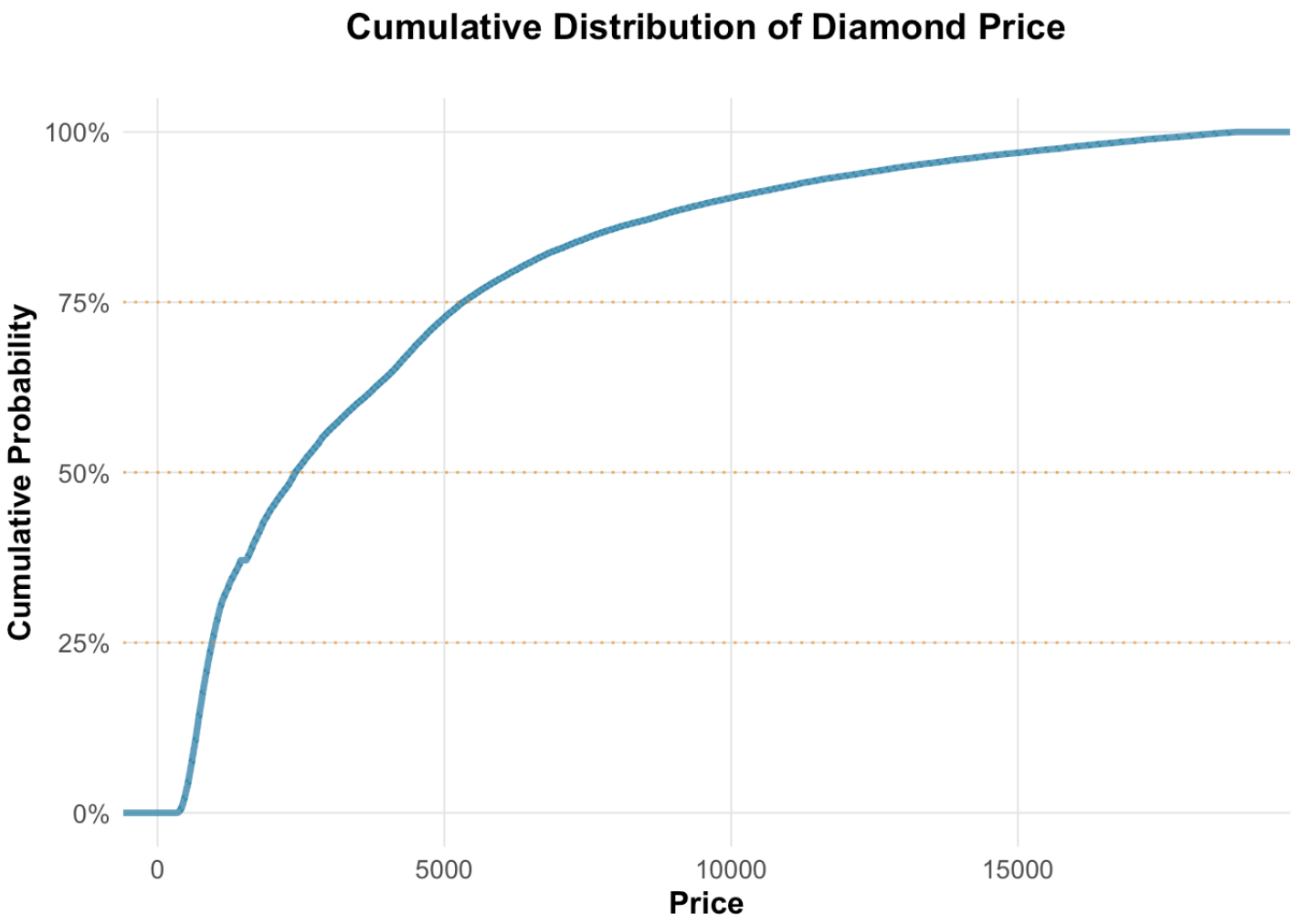
```
print(p2)
```




```
print(p3)
```


Density Distribution of Diamond Price
Density
3e-04
2e-04
1e-04
0e+00
0
5000
10000
15000
Price

```
print(p4)
```



X (Length)

Based on the analysis below, it is clear that carat is semi normal and multipolar, with a mean of 5.731 and standard deviation of 1.121761.

```
summary(diamonds$x)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.000   4.710   5.700   5.731   6.540  10.740 
```

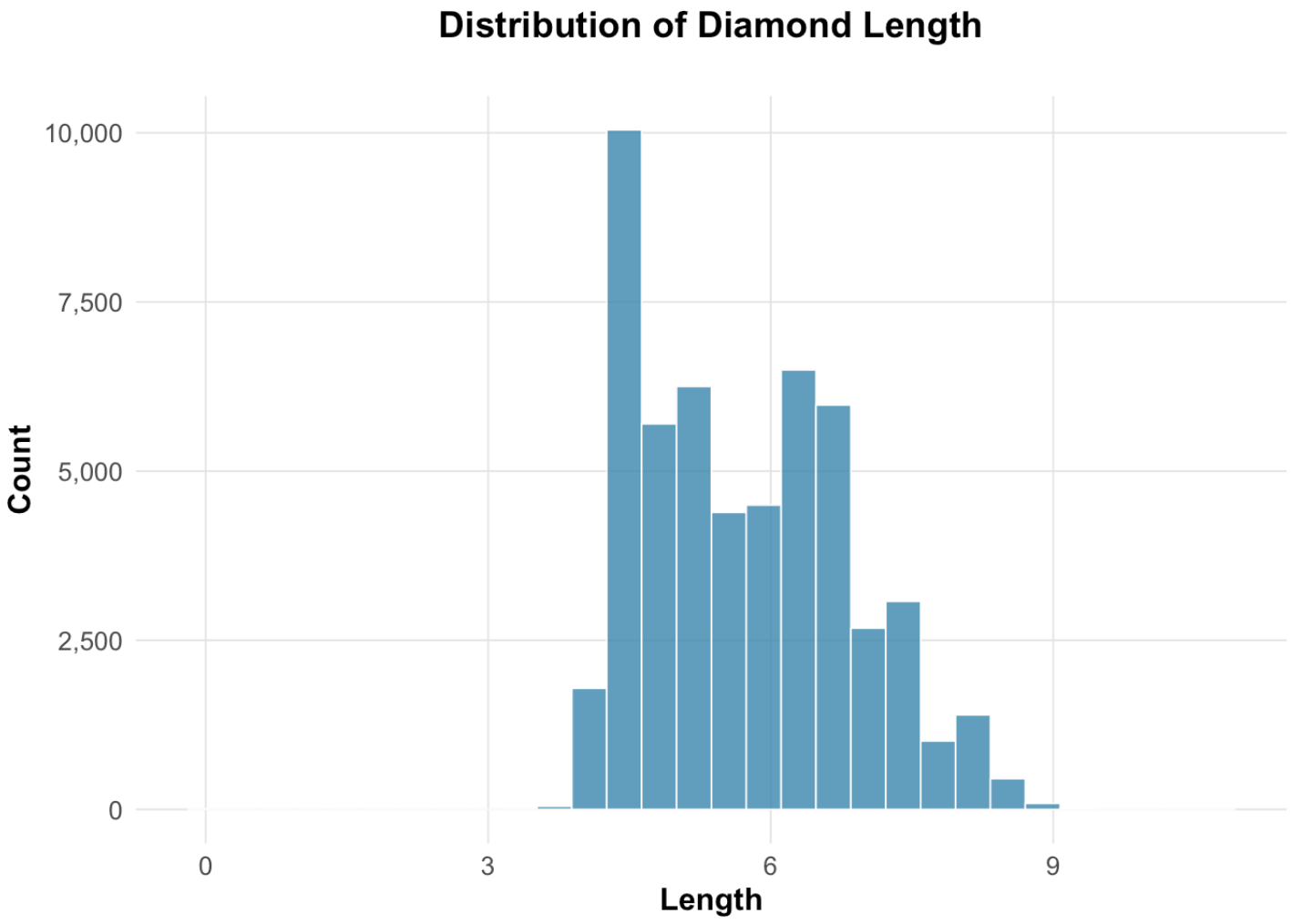
```
sd(diamonds$x)
```

```
[1] 1.121761
```

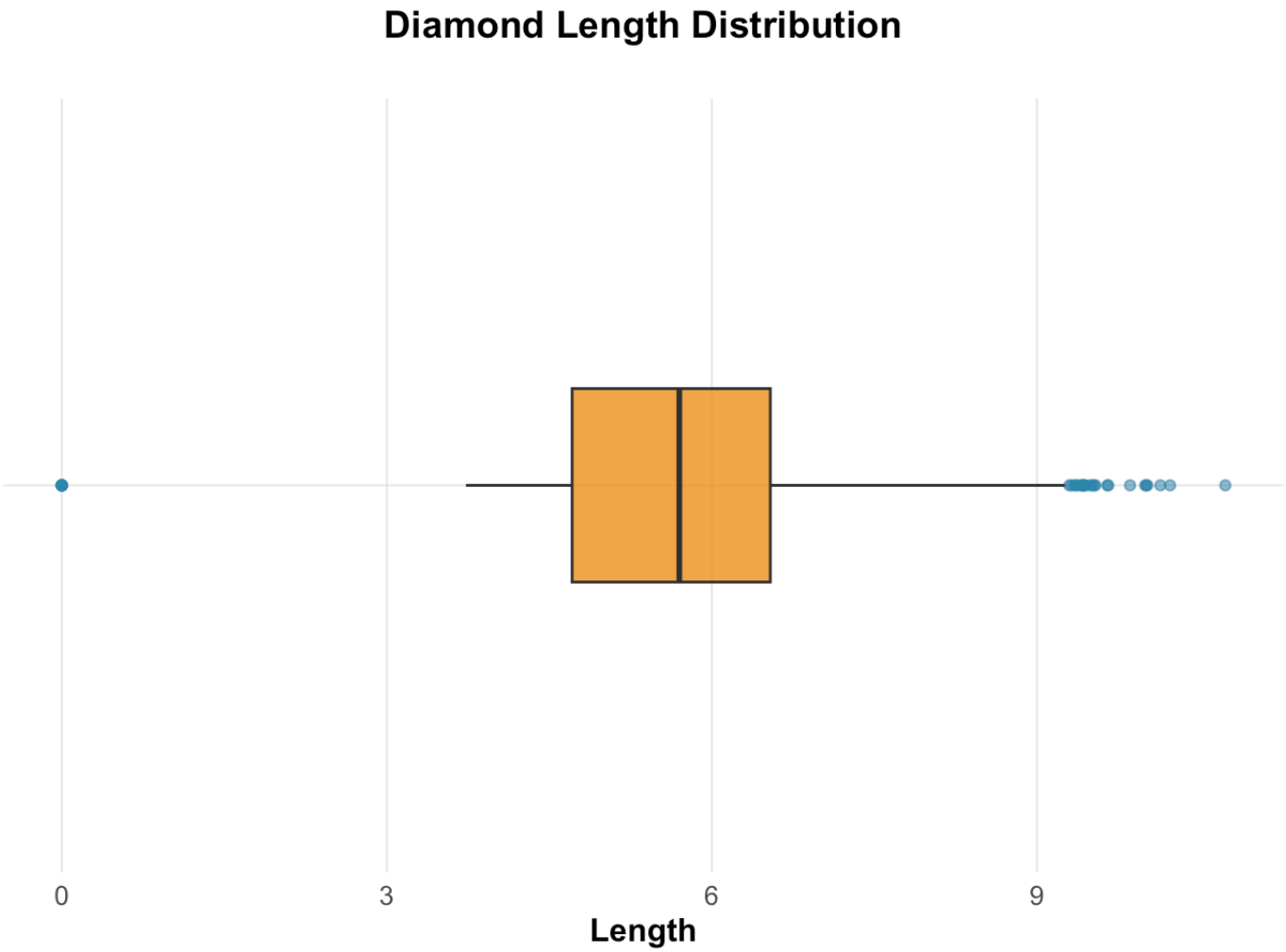
```
# Creating a Histogram
```

```r
p1 <- ggplot(diamonds, aes(x = x)) +
  geom_histogram(bins = 30,
                 fill = diamond_blue,
                 color = "white",
                 alpha = 0.8,
                 size = 0.3) +
  labs(
    title = "Distribution of Diamond Length",
    x = "Length",
    y = "Count"
  ) +
  custom_theme +
  scale_y_continuous(labels = scales::comma_format())

# Creating a boxplot
p2 <- ggplot(diamonds, aes(x = x, y = "")) +
  geom_boxplot(width = 0.3,
               fill = diamond_gold,
               alpha = 0.8,
               outlier.color = diamond_blue,
               outlier.alpha = 0.6) +
  labs(
    title = "Diamond Length Distribution",
    x = "Length",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())

# Creating a nice density plot
p3 <- ggplot(diamonds, aes(x = x)) +
  geom_density(fill = diamond_blue,
               alpha = 0.7,
               color = diamond_blue,
               size = 1.2) +
  labs(
    title = "Density Distribution of Diamond Length",
    x = "Length",
    y = "Density"
  ) +
  custom_theme

# Creating a nice ECDF (empirical cumulative distribution function)
p4 <- ggplot(diamonds, aes(x = x)) +
```

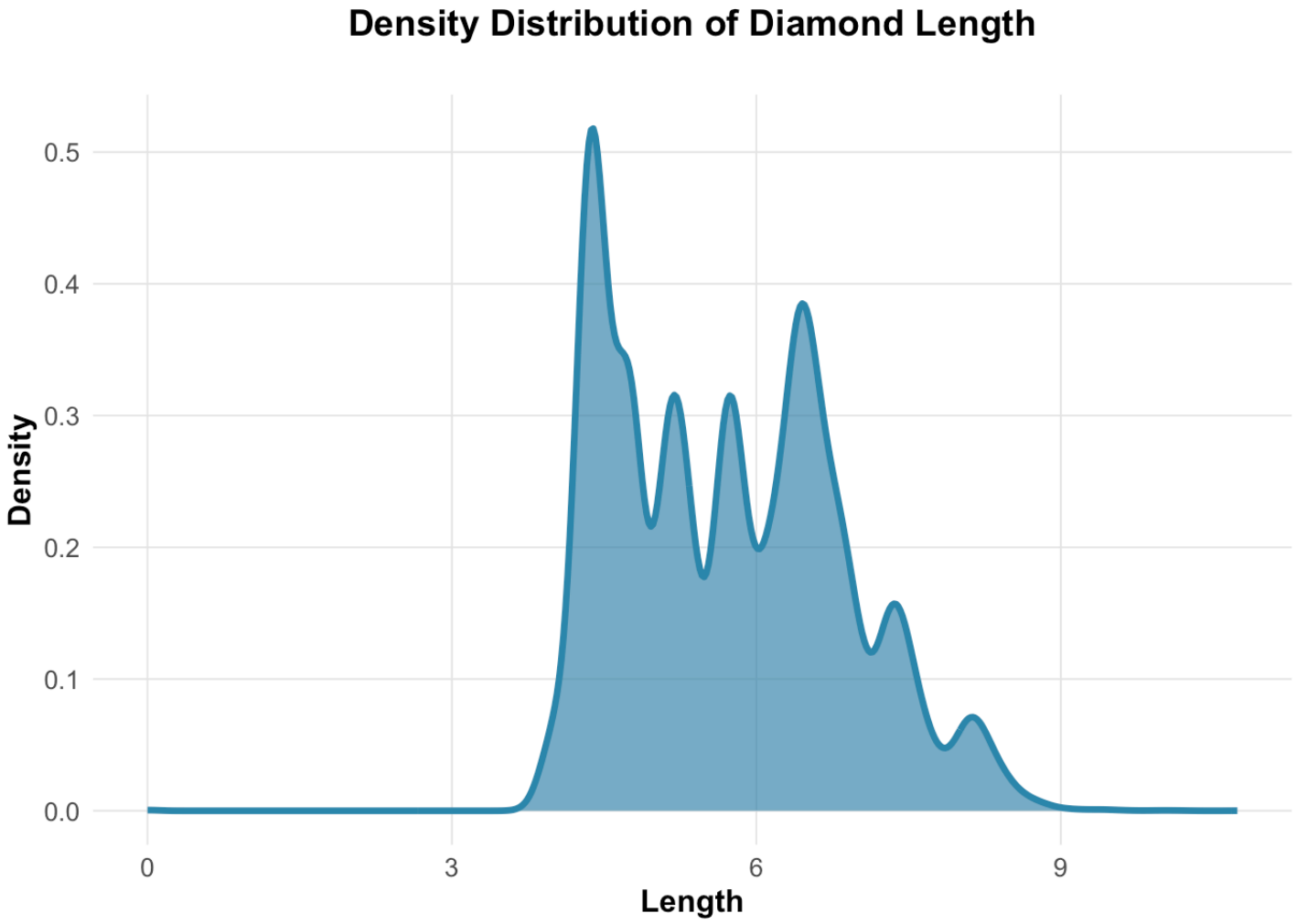
```
  stat_ecdf(color = diamond_blue,
            size = 1.2,
            alpha = 0.8) +
  geom_hline(yintercept = c(0.25, 0.5, 0.75),
             linetype = "dotted",
             color = diamond_gold,
             alpha = 0.7) +
  labs(
    title = "Cumulative Distribution of Diamond Length",
    x = "Length",
    y = "Cumulative Probability"
  ) +
  custom_theme +
  scale_y_continuous(labels = scales::percent_format())

# Displaying all plots
print(p1)
```



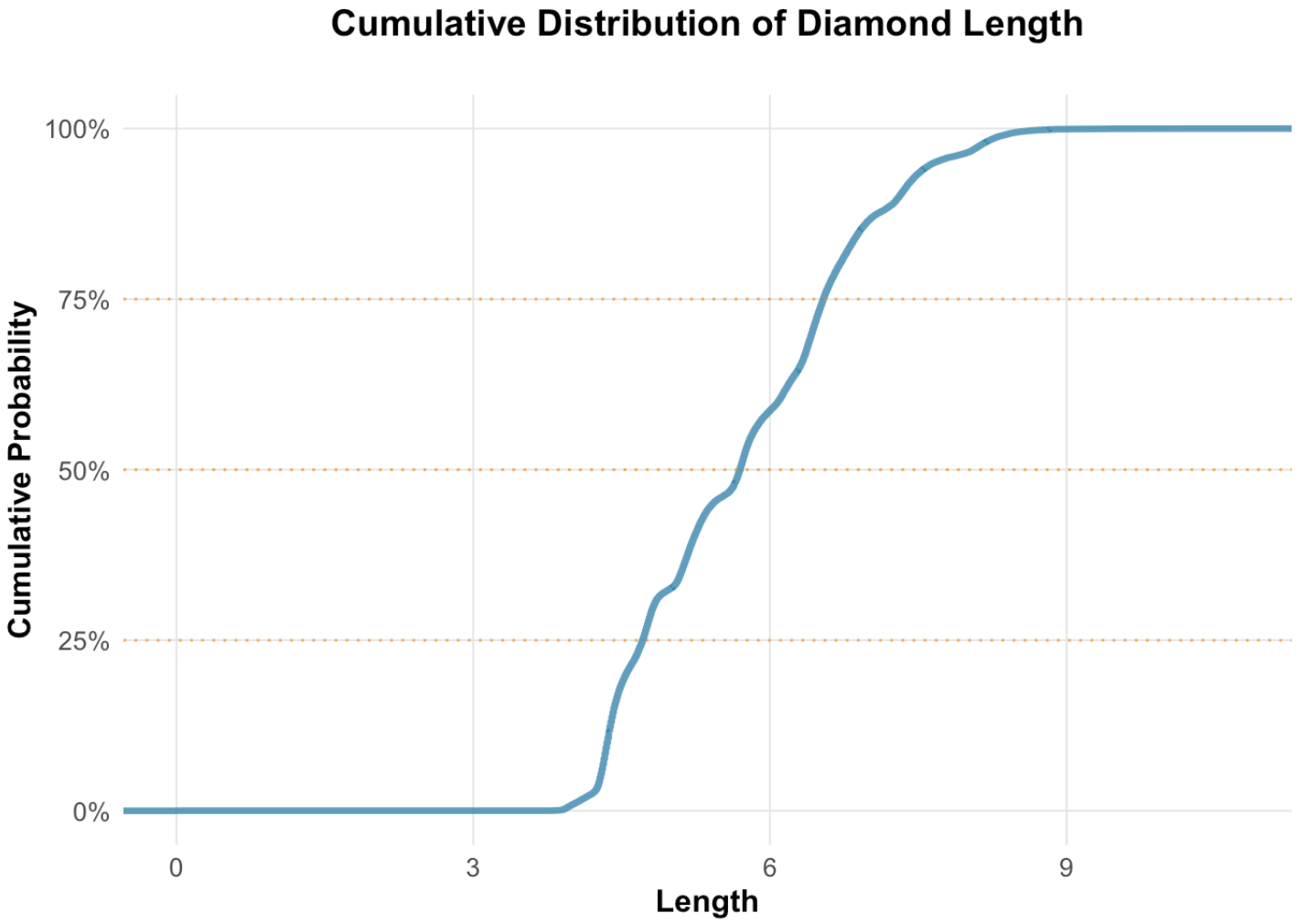
```
print(p2)
```




```
print(p3)
```


Density Distribution of Diamond Length
Density
0.5
0.4
0.3
0.2
0.1
0.0
0
3
6
9
Length

```
print(p4)
```



Y (Width)

Based on the analysis below, it is clear that carat is semi normal and multipolar, with a mean of 5.735 and standard deviation of 1.142.

```
summary(diamonds$y)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.000   4.720   5.710   5.735   6.540  58.900 
```

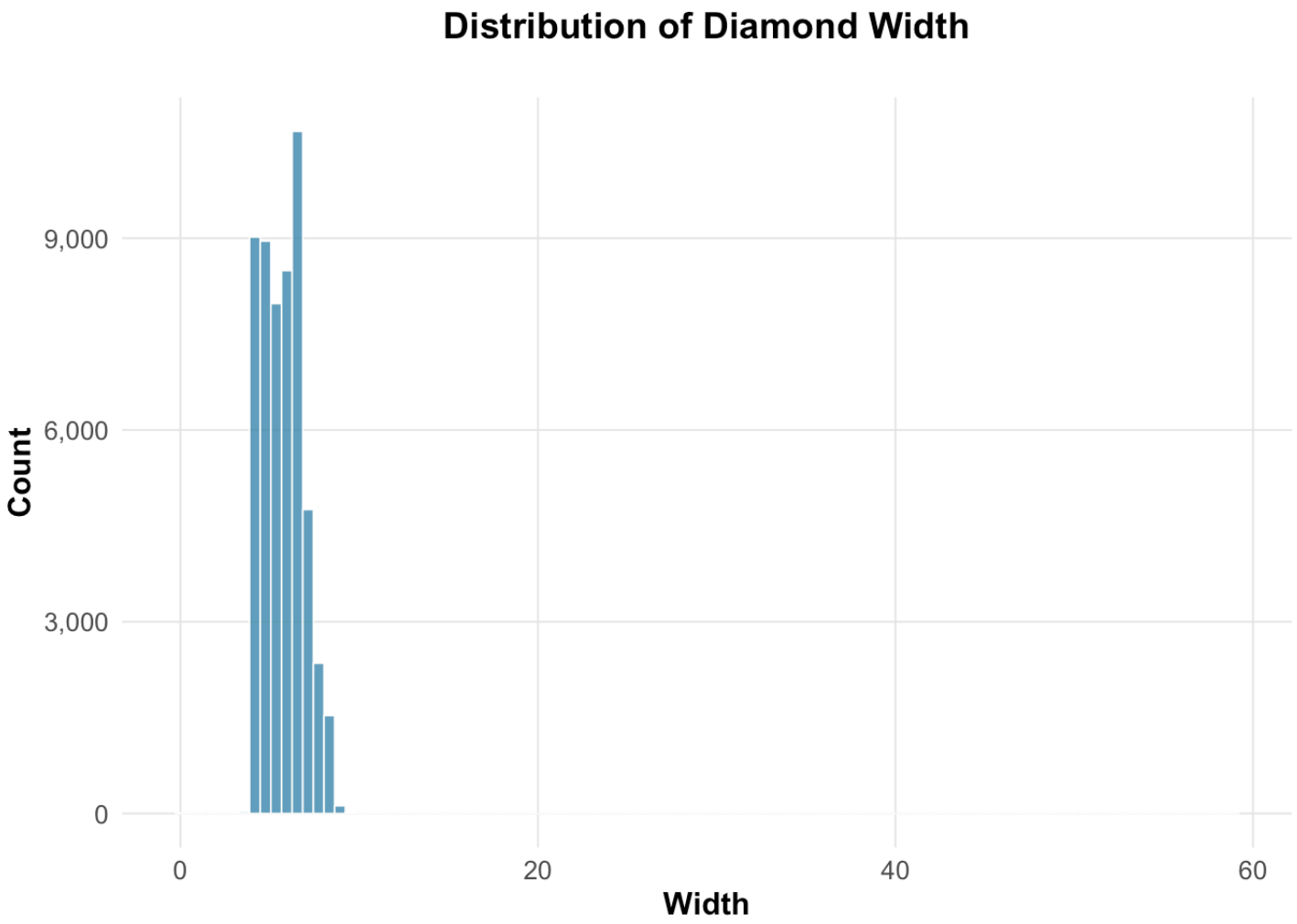
```
sd(diamonds$y)
```

```
[1] 1.142135
```

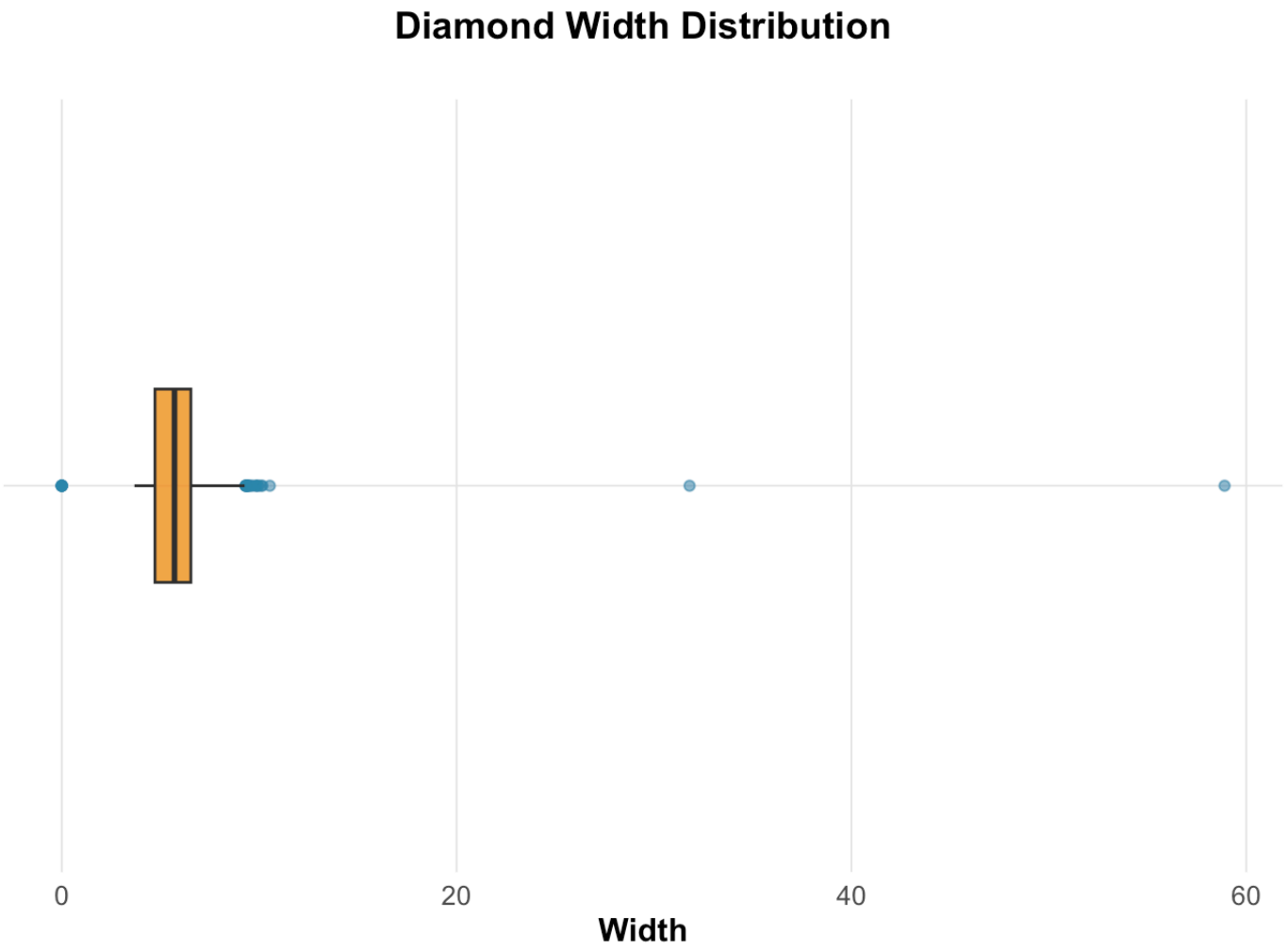
```
# Creating a Histogram
```

```
p1 <- ggplot(diamonds, aes(x = y)) +
  geom_histogram(bins = 100,
                 fill = diamond_blue,
                 color = "white",
                 alpha = 0.8,
                 size = 0.3) +
  labs(
    title = "Distribution of Diamond Width",
    x = "Width",
    y = "Count"
  ) +
  custom_theme +
  scale_y_continuous(labels = scales::comma_format())

# Creating a boxplot
p2 <- ggplot(diamonds, aes(x = y, y = "")) +
  geom_boxplot(width = 0.3,
               fill = diamond_gold,
               alpha = 0.8,
               outlier.color = diamond_blue,
               outlier.alpha = 0.6) +
  labs(
    title = "Diamond Width Distribution",
    x = "Width",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())

# Creating a nice density plot
p3 <- ggplot(diamonds, aes(x = y)) +
  geom_density(fill = diamond_blue,
               alpha = 0.7,
               color = diamond_blue,
               size = 1.2) +
  labs(
    title = "Density Distribution of Diamond Width",
    x = "Width",
    y = "Density"
  ) +
  custom_theme +
  xlim(0, 15)

# Creating a nice ECDF (empirical cumulative distribution function)
```

```
p4 <- ggplot(diamonds, aes(x = y)) +
  stat_ecdf(color = diamond_blue,
            size = 1.2,
            alpha = 0.8) +
  geom_hline(yintercept = c(0.25, 0.5, 0.75),
             linetype = "dotted",
             color = diamond_gold,
             alpha = 0.7) +
  labs(
    title = "Cumulative Distribution of Diamond Width",
    x = "Width",
    y = "Cumulative Probability"
  ) +
  custom_theme +
  scale_y_continuous(labels = scales::percent_format())

# Displaying all plots
print(p1)
```

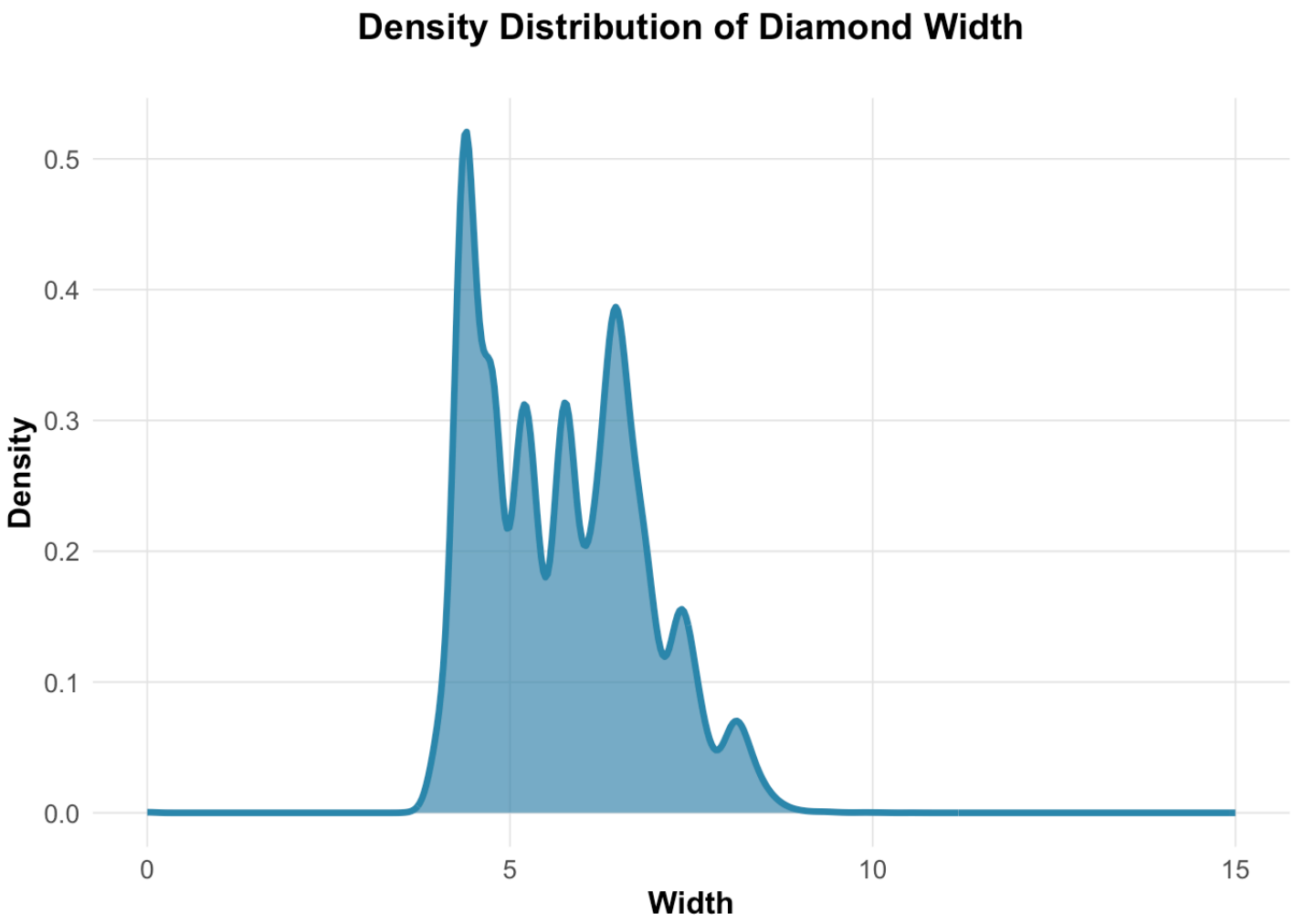


```
print(p2)
```

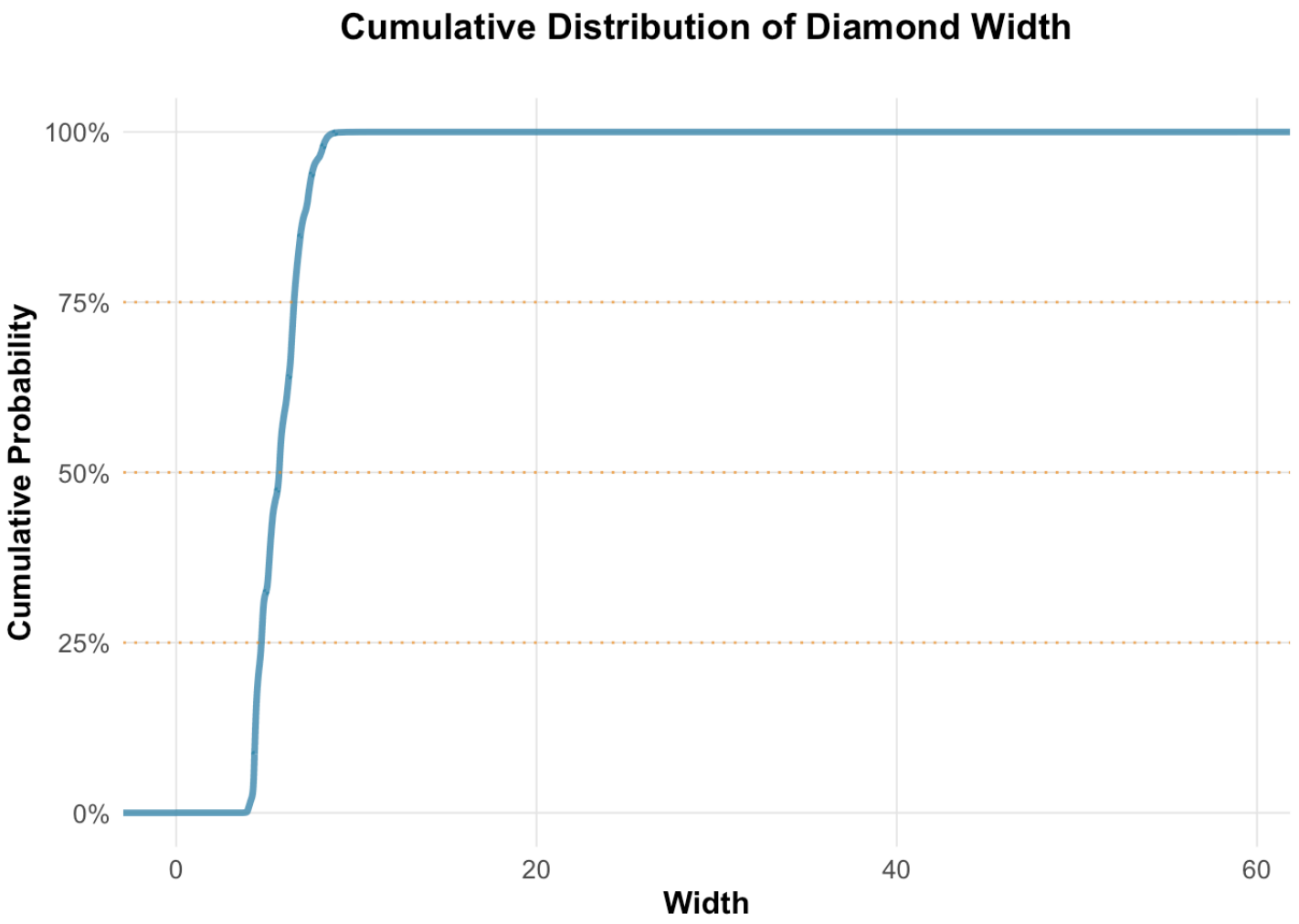



```
print(p3)
```

```
Warning: Removed 2 rows containing non-finite outside the scale range
(`stat_density()`).
```


Density Distribution of Diamond Width
Density
0.0
0.1
0.2
0.3
0.4
0.5
0
5
10
15
Width

```
print(p4)
```



Z (Depth)

Based on the analysis below, it is clear that carat is semi normal and multipolar, with a mean of 3.539 and standard deviation of 0.706.

```
summary(diamonds$z)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.000   2.910   3.530   3.539   4.040  31.800 
```

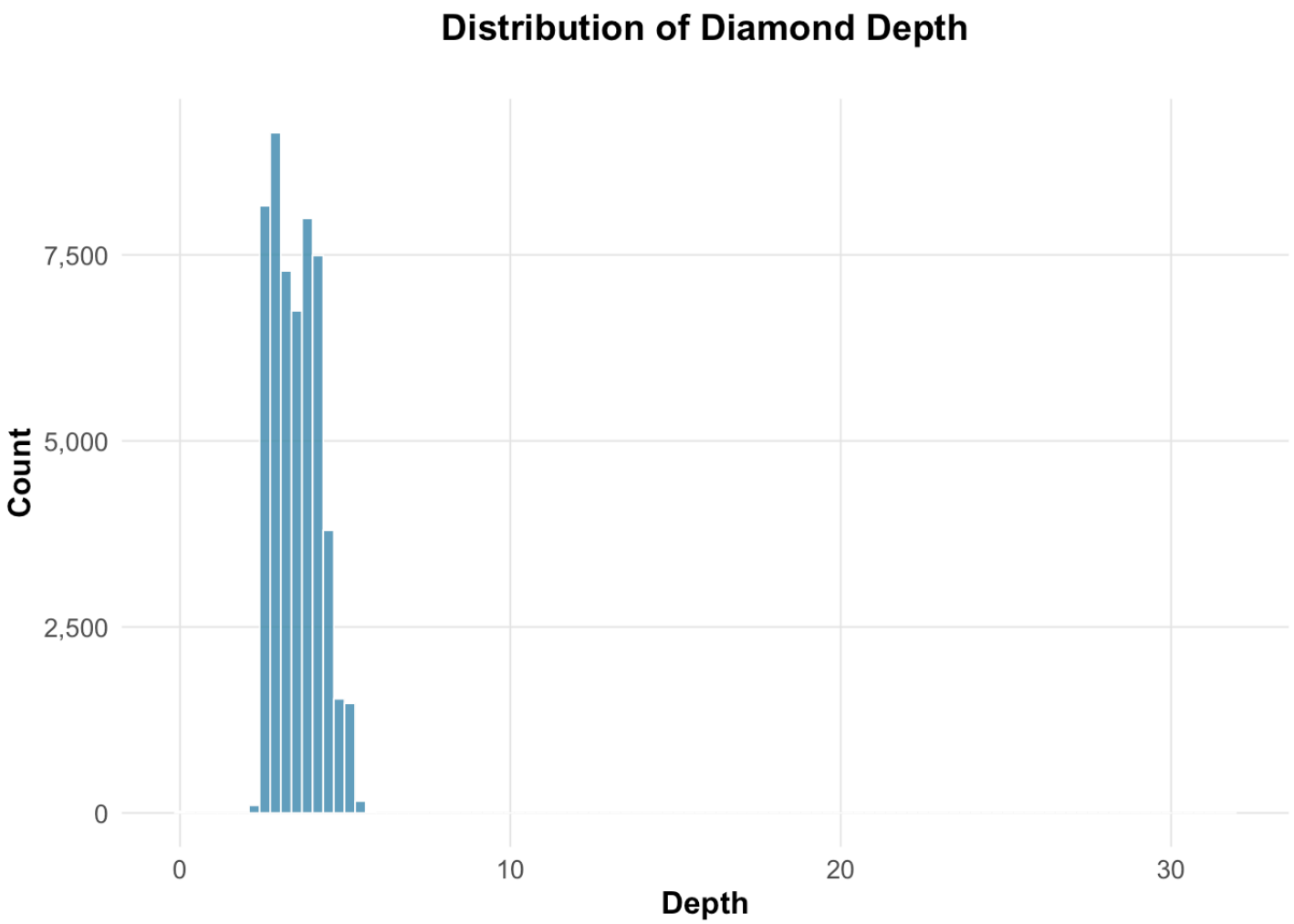
```
sd(diamonds$z)
```

```
[1] 0.7056988
```

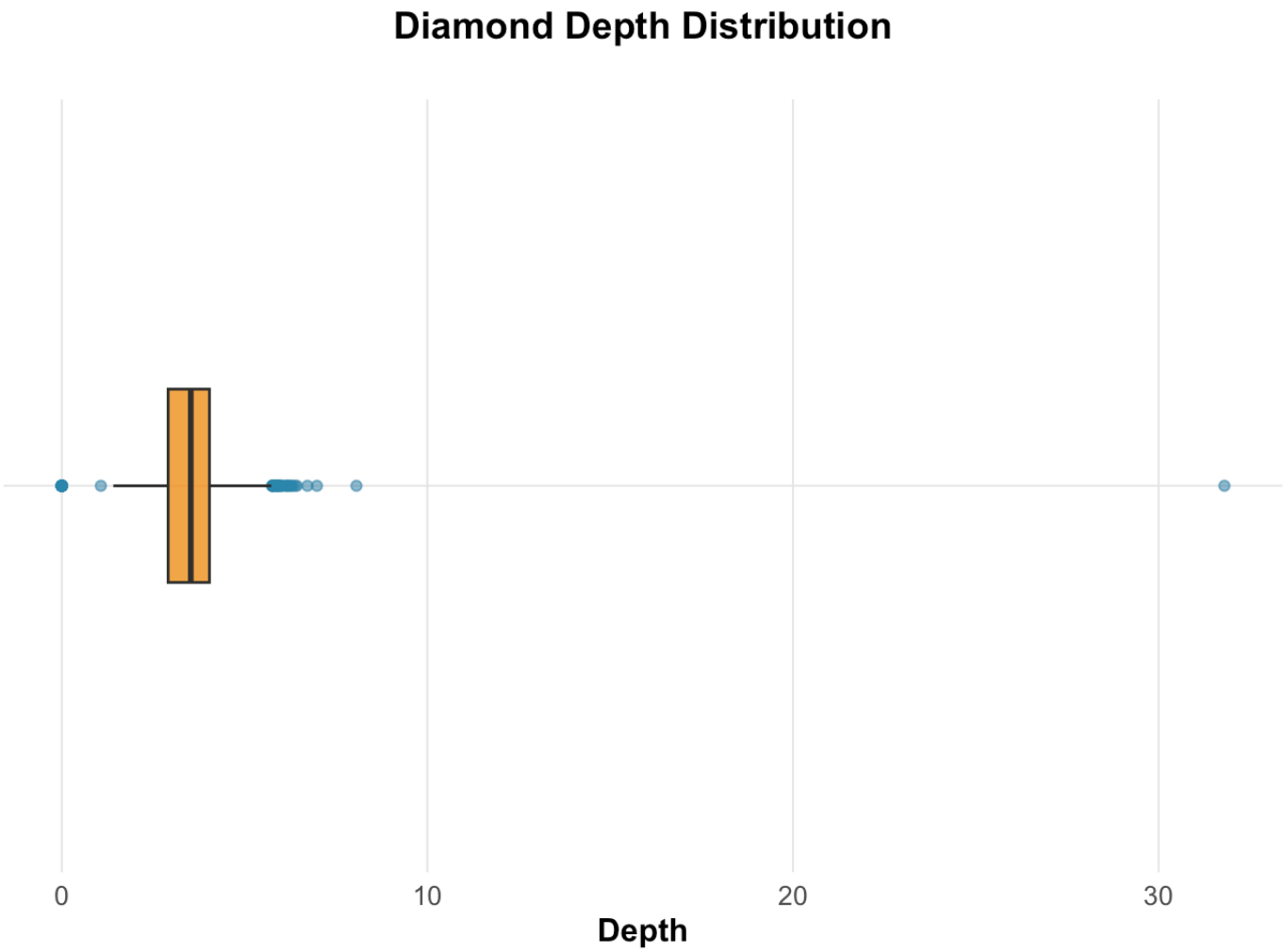
```
# Creating a Histogram
```

```
p1 <- ggplot(diamonds, aes(x = z)) +
  geom_histogram(bins = 100,
                 fill = diamond_blue,
                 color = "white",
                 alpha = 0.8,
                 size = 0.3) +
  labs(
    title = "Distribution of Diamond Depth",
    x = "Depth",
    y = "Count"
  ) +
  custom_theme +
  scale_y_continuous(labels = scales::comma_format())

# Creating a boxplot
p2 <- ggplot(diamonds, aes(x = z, y = "")) +
  geom_boxplot(width = 0.3,
               fill = diamond_gold,
               alpha = 0.8,
               outlier.color = diamond_blue,
               outlier.alpha = 0.6) +
  labs(
    title = "Diamond Depth Distribution",
    x = "Depth",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())

# Creating a nice density plot
p3 <- ggplot(diamonds, aes(x = z)) +
  geom_density(fill = diamond_blue,
               alpha = 0.7,
               color = diamond_blue,
               size = 1.2) +
  labs(
    title = "Density Distribution of Diamond Depth",
    x = "Depth",
    y = "Density"
  ) +
  custom_theme +
  xlim(0, 15)

# Creating a nice ECDF (empirical cumulative distribution function)
```

```
p4 <- ggplot(diamonds, aes(x = z)) +
  stat_ecdf(color = diamond_blue,
            size = 1.2,
            alpha = 0.8) +
  geom_hline(yintercept = c(0.25, 0.5, 0.75),
             linetype = "dotted",
             color = diamond_gold,
             alpha = 0.7) +
  labs(
    title = "Cumulative Distribution of Diamond Depth",
    x = "Depth",
    y = "Cumulative Probability"
  ) +
  custom_theme +
  scale_y_continuous(labels = scales::percent_format())

# Displaying all plots
print(p1)
```

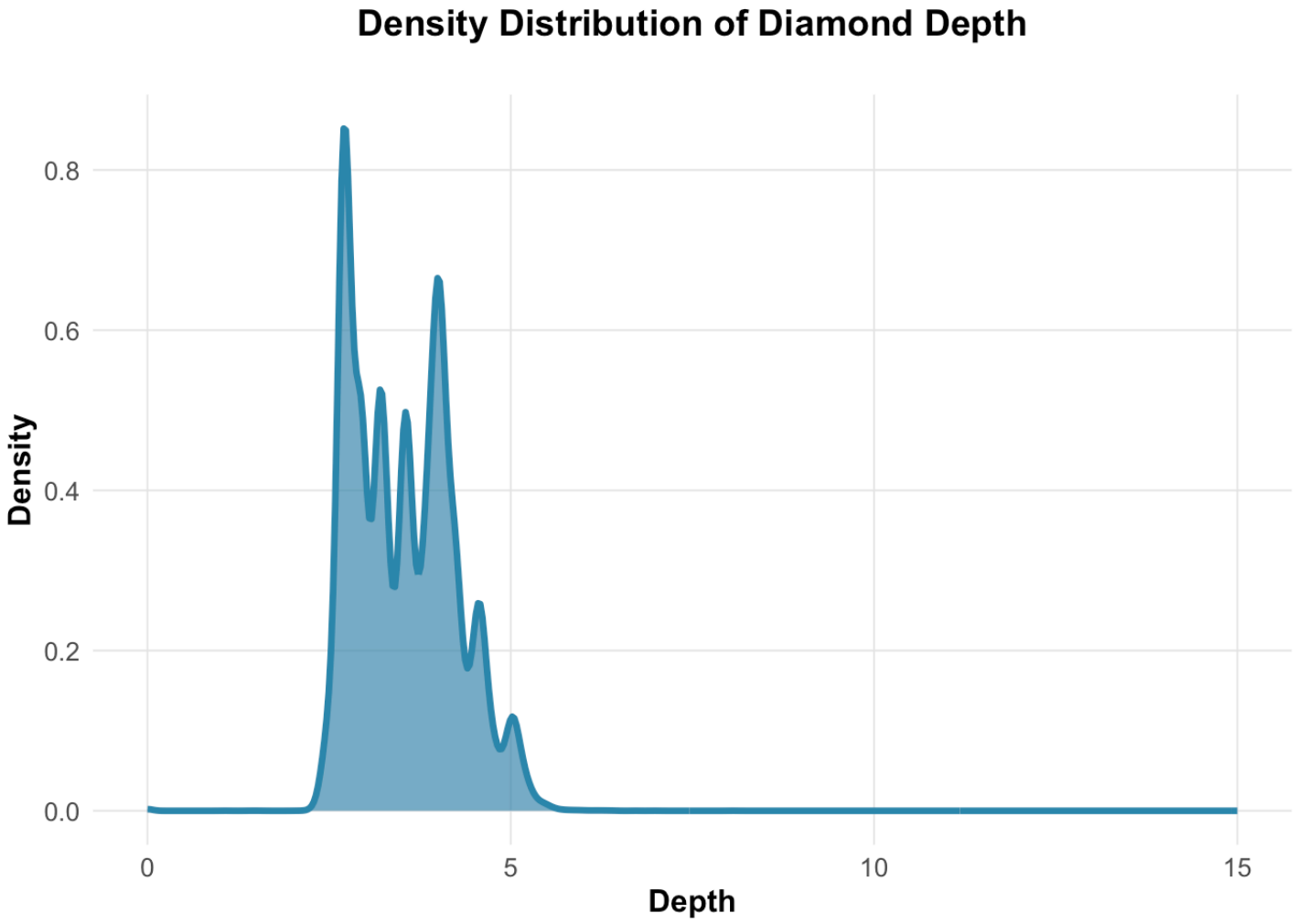


```
print(p2)
```

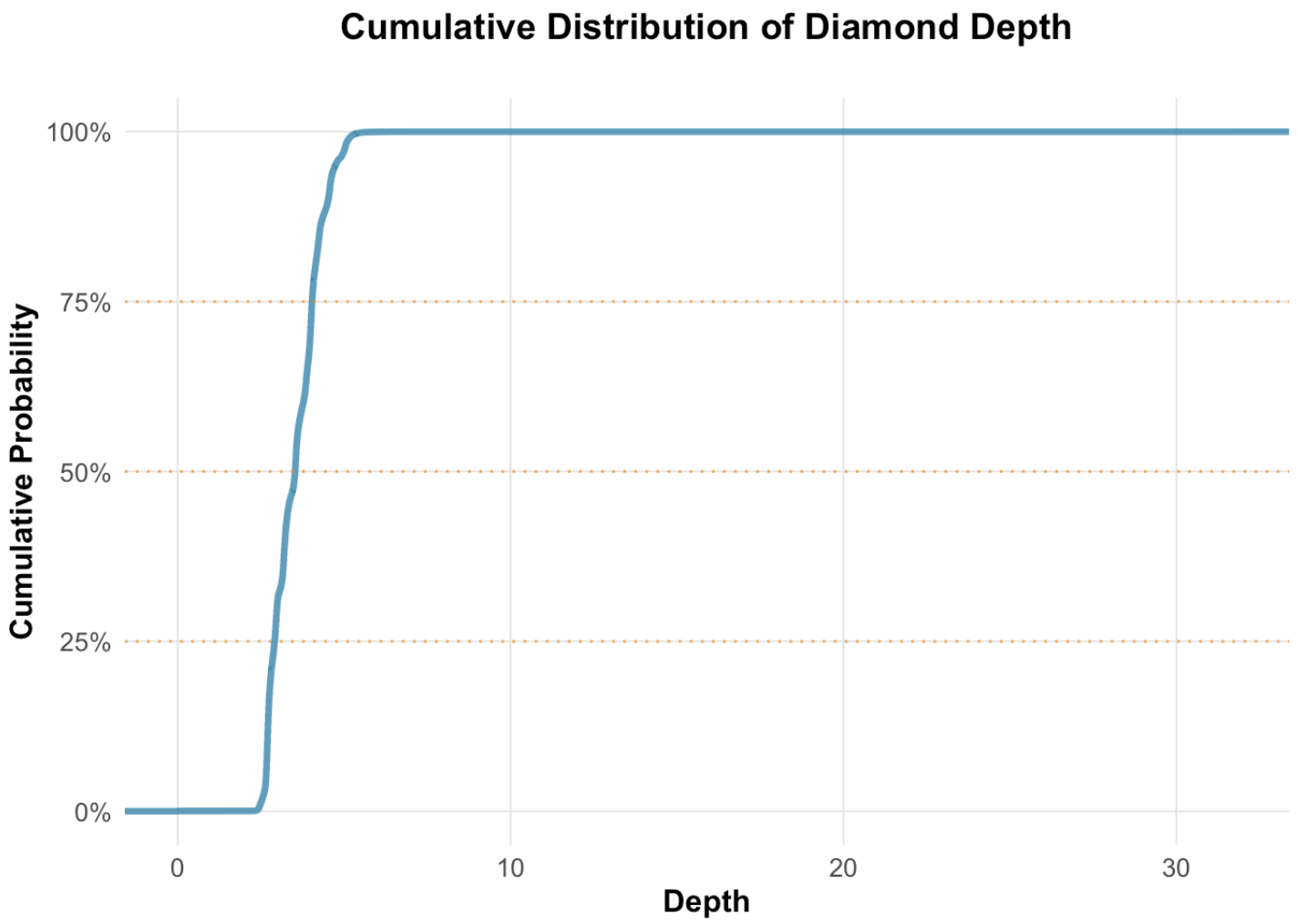



```
print(p3)
```

```
Warning: Removed 1 row containing non-finite outside the scale range
(`stat_density()`).
```


Density Distribution of Diamond Depth
Density
0.0
0.2
0.4
0.6
0.8
0
5
10
15
Depth

```
print(p4)
```



Relationship between variables

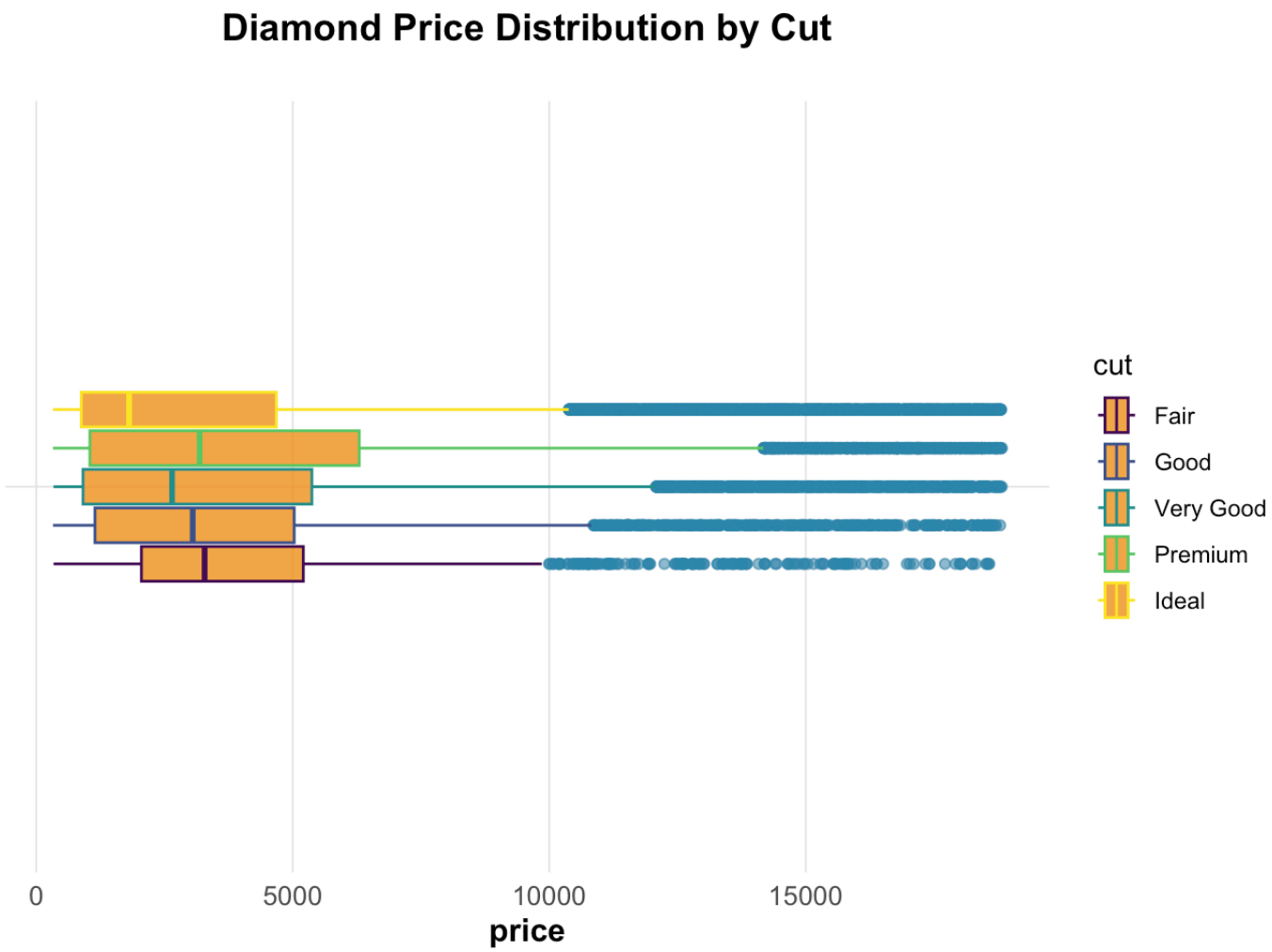
Having understood the distributions of individual variables, the next step is to understand how variables relate to one another. As we're primarily interested in understanding diamond price, we will look at how our different variables relate to this variable.

Price's relationship with categorical variables

The data set is home to a total of 3 categorical variables: cut, color, and clarity.

With respect to cut, it is clear that lower cuts of diamonds seem to have a higher median price than higher cuts. While on the surface this seems strange, the likely reason is that poorly cut diamonds are more likely to have other traits associated with higher prices (such as larger sizes). The distributions are each uni polar and highly right tailed.

```
ggplot(diamonds, aes(x = price, y = "", color = cut)) +
  geom_boxplot(width = 0.3,
               fill = diamond_gold,
               alpha = 0.8,
               outlier.color = diamond_blue,
               outlier.alpha = 0.6) +
  labs(
    title = "Diamond Price Distribution by Cut",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```



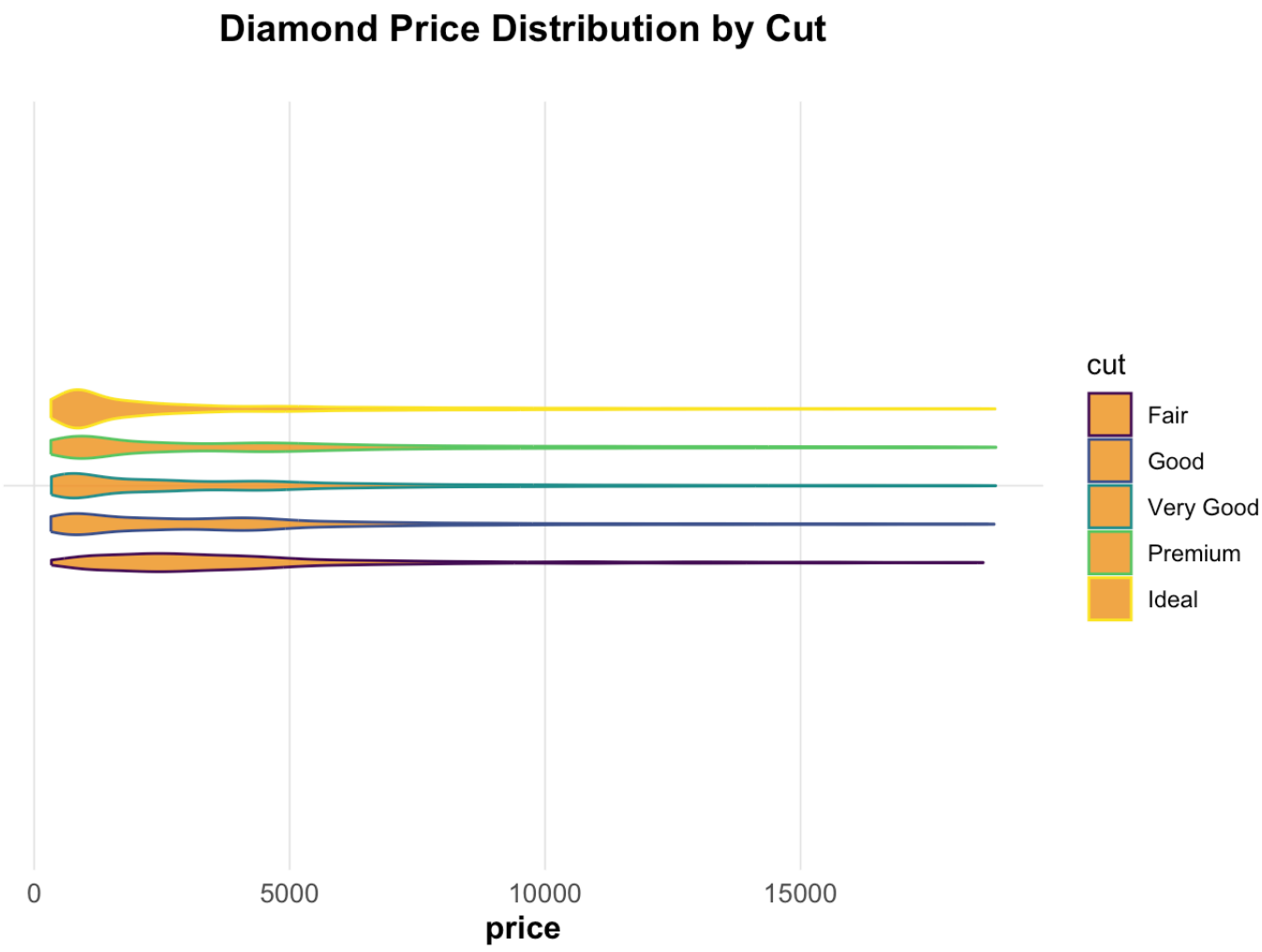

```
ggplot(diamonds, aes(x=price)) +
  geom_histogram(width = 0.3,
               fill = diamond_gold,
```

```
              alpha = 0.8,
              outlier.color = diamond_blue,
              outlier.alpha = 0.6) +
  facet_wrap(~cut, scales = "free_y") +
    labs(
    title = "Diamond Price Distribution by Cut",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
Warning in geom_histogram(width = 0.3, fill = diamond_gold, alpha = 0.8, :
Ignoring unknown parameters: `outlier.colour` and `outlier.alpha`

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

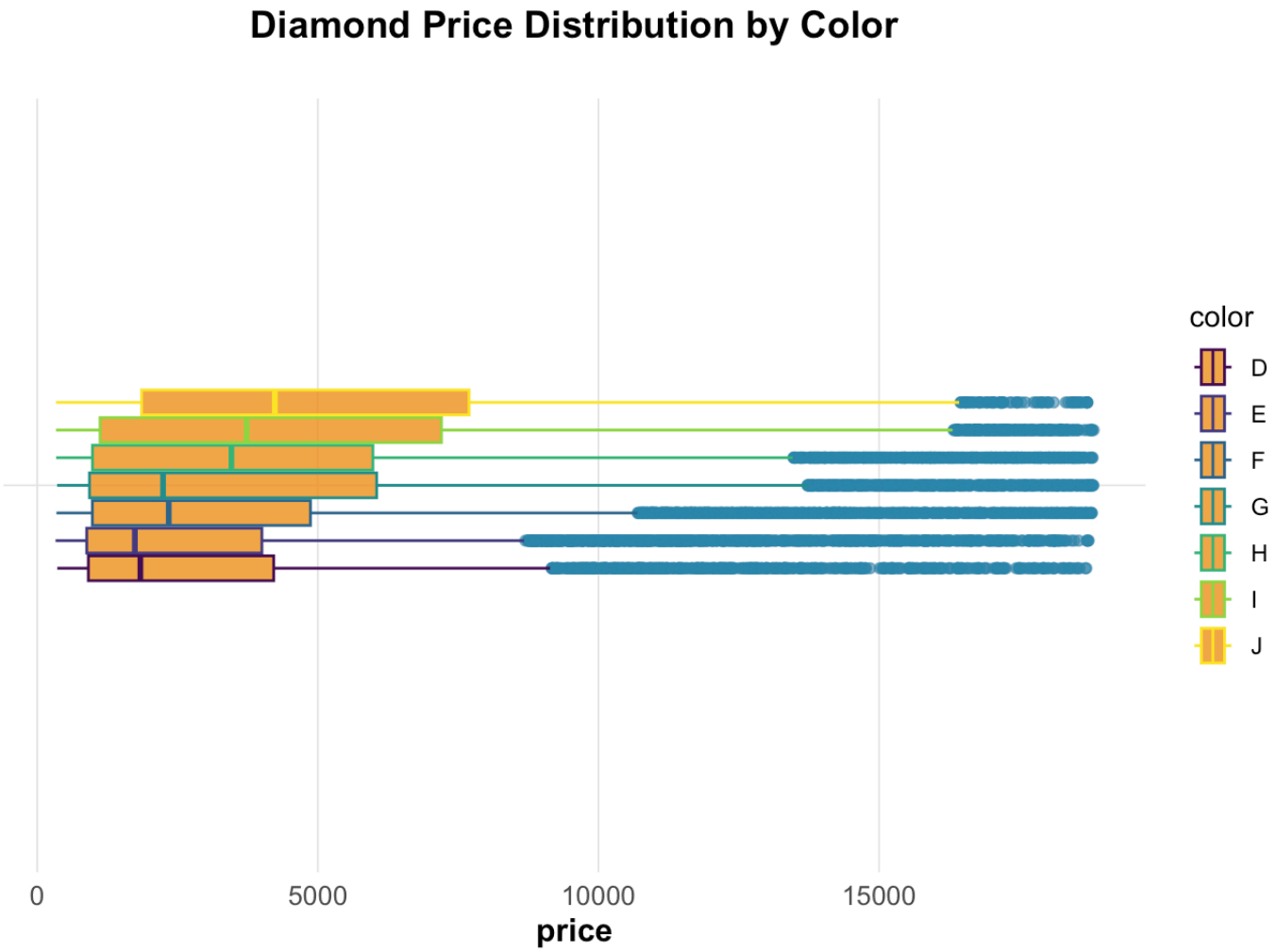
```
ggplot(diamonds, aes(x=price, y = "", color = cut)) +
  geom_violin(width = 0.3,
              fill = diamond_gold,
              alpha = 0.8,
              outlier.color = diamond_blue,
              outlier.alpha = 0.6) +
    labs(
    title = "Diamond Price Distribution by Cut",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
Warning in geom_violin(width = 0.3, fill = diamond_gold, alpha = 0.8,
outlier.color = diamond_blue, : Ignoring unknown parameters: `outlier.colour`
and `outlier.alpha`
```



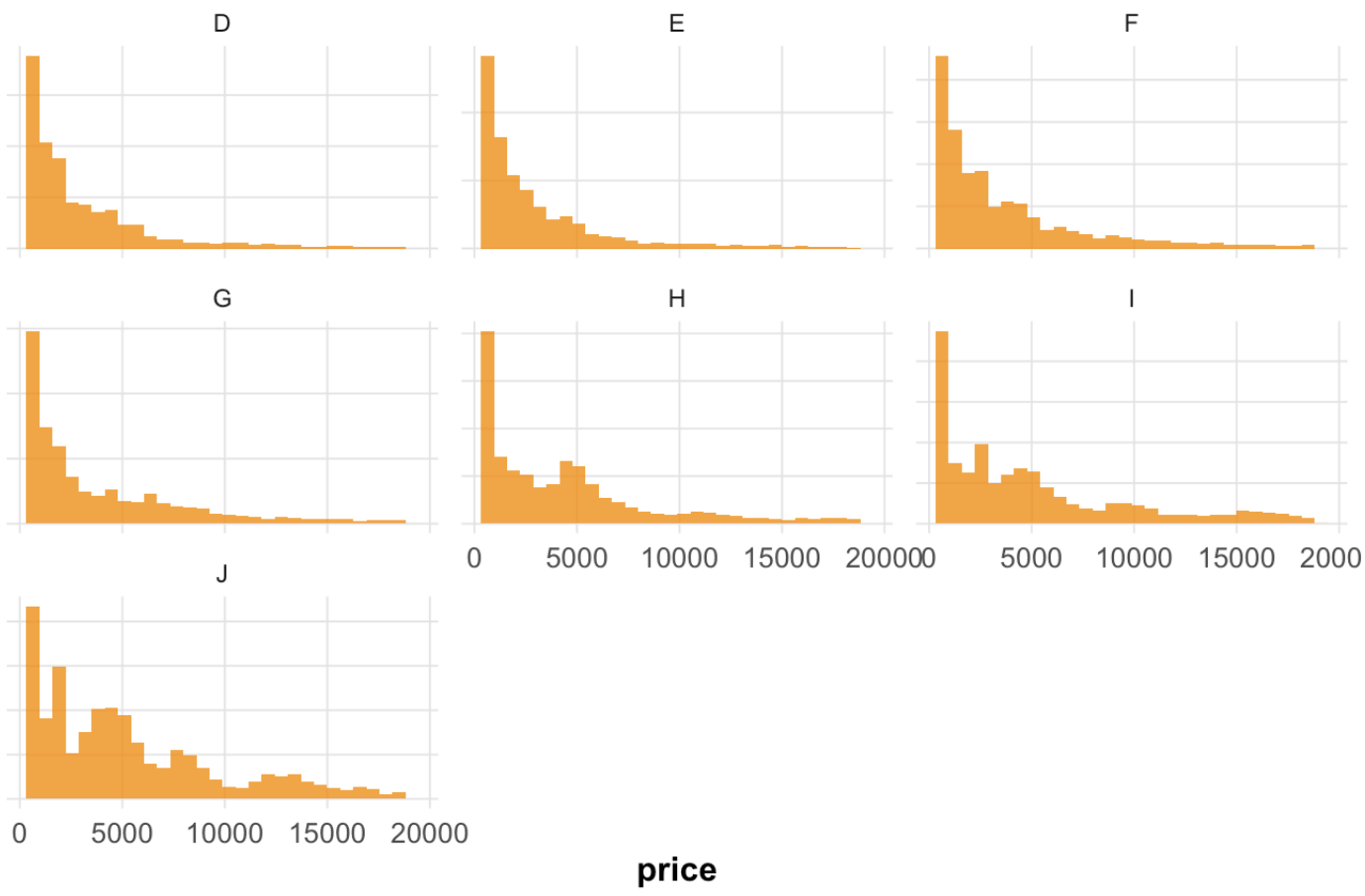
Here, more desirable colors are associated with higher median prices and higher price ranges in general. Again, the distributions are highly right tailed and mostly unipolar.

```
ggplot(diamonds, aes(x = price, y = "", color = color)) +
  geom_boxplot(width = 0.3,
               fill = diamond_gold,
               alpha = 0.8,
               outlier.color = diamond_blue,
               outlier.alpha = 0.6) +
  labs(
    title = "Diamond Price Distribution by Color",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```



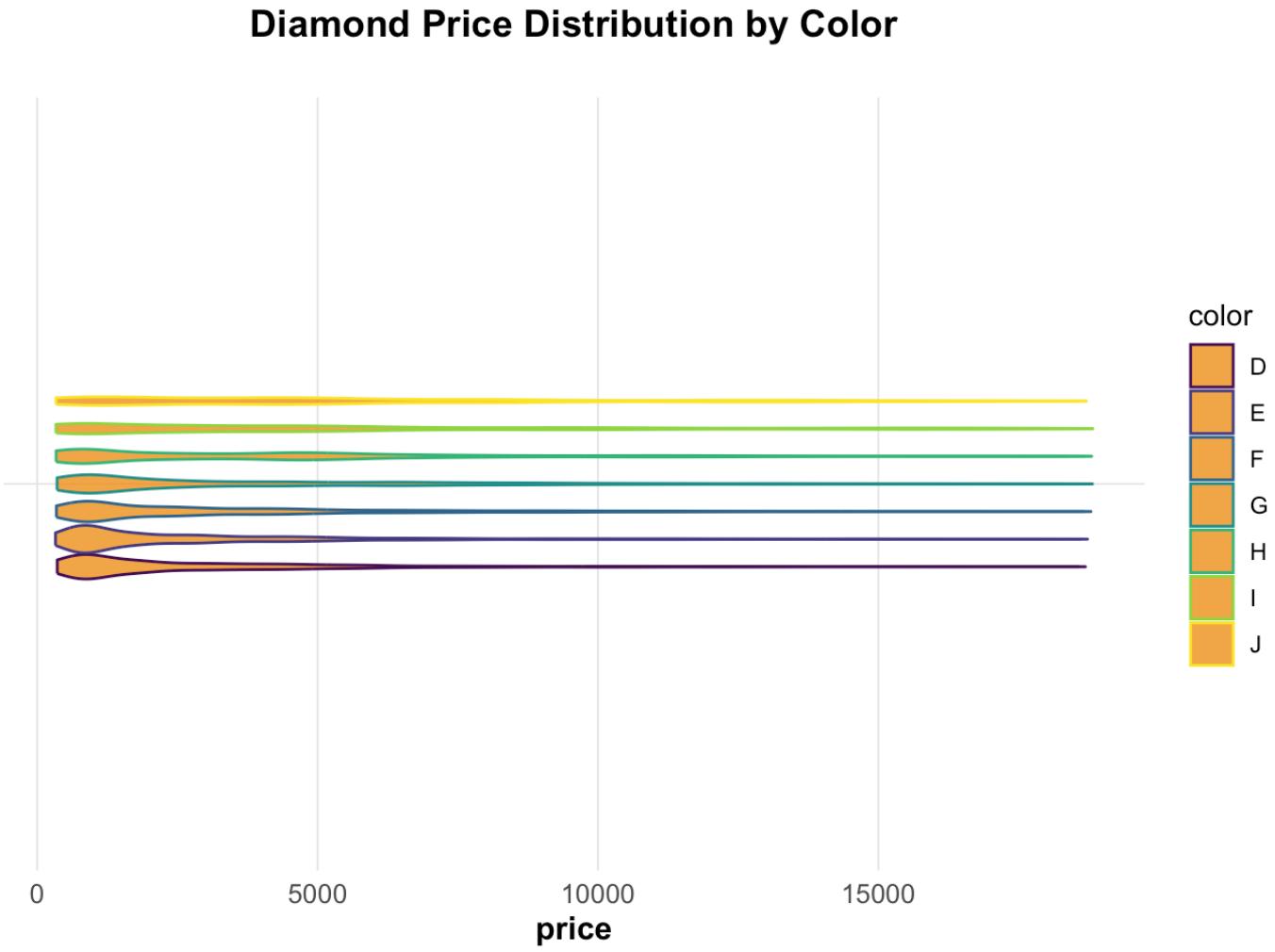
```
ggplot(diamonds, aes(x=price)) +
  geom_histogram(width = 0.3,
                 fill = diamond_gold,
                 alpha = 0.8,
                 outlier.color = diamond_blue,
                 outlier.alpha = 0.6) +
  facet_wrap(~color, scales = "free_y") +
    labs(
    title = "Diamond Price Distribution by Color",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
Warning in geom_histogram(width = 0.3, fill = diamond_gold, alpha = 0.8, :
Ignoring unknown parameters: `outlier.colour` and `outlier.alpha`

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



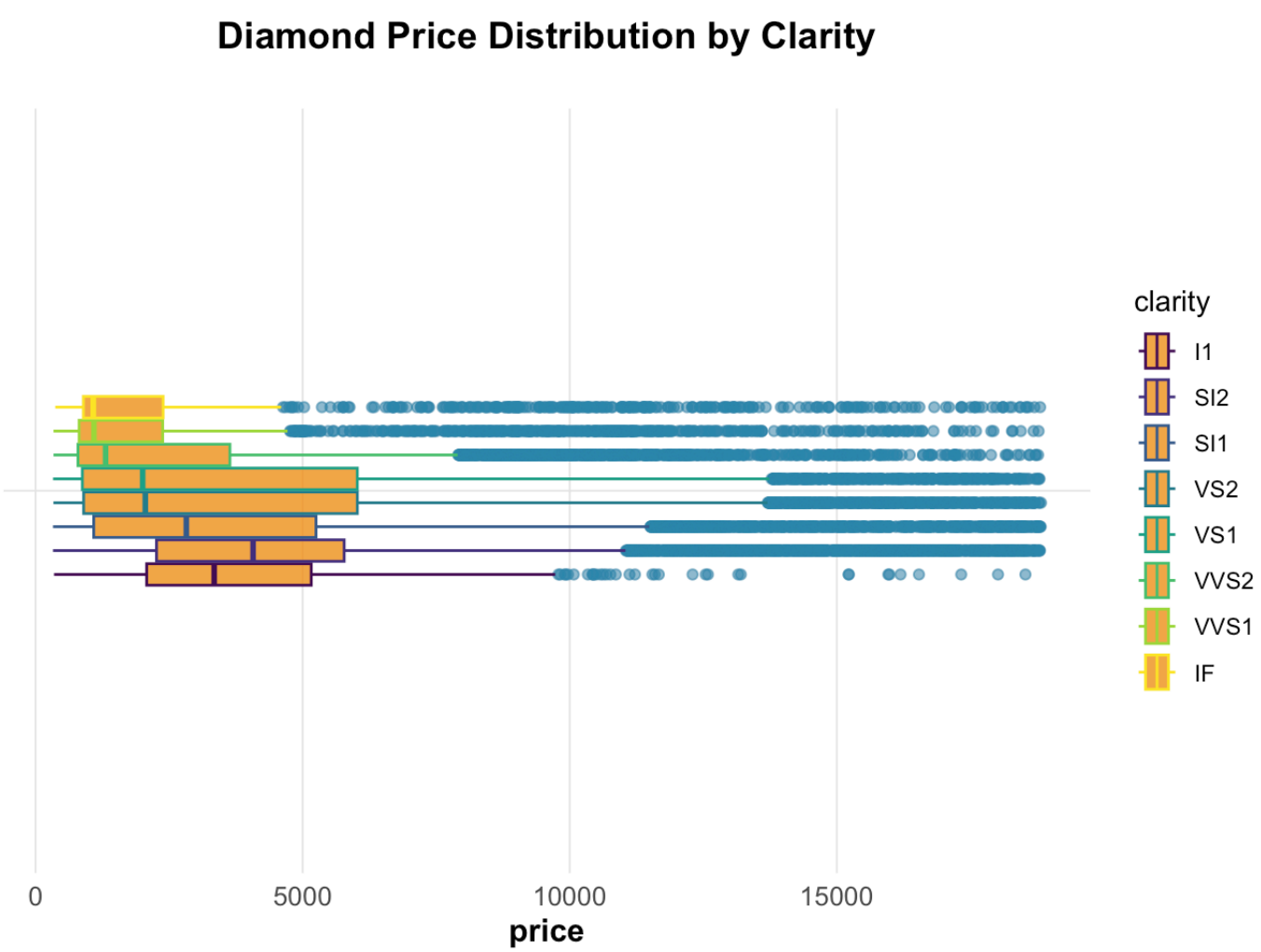

```
ggplot(diamonds, aes(x=price, y = "", color = color)) +
  geom_violin(width = 0.3,
              fill = diamond_gold,
              alpha = 0.8,
              outlier.color = diamond_blue,
              outlier.alpha = 0.6) +
    labs(
    title = "Diamond Price Distribution by Color",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
Warning in geom_violin(width = 0.3, fill = diamond_gold, alpha = 0.8,
outlier.color = diamond_blue, : Ignoring unknown parameters: `outlier.colour`
and `outlier.alpha`
```



With respect to clarity, it is clear that lower clarity diamonds seem to have a higher median price than higher clarity. While on the surface this seems strange, the likely reason is that less clear diamonds are more likely to have other traits associated with higher prices (such as larger sizes), just like with cut. The distributions are each uni polar and highly right tailed.

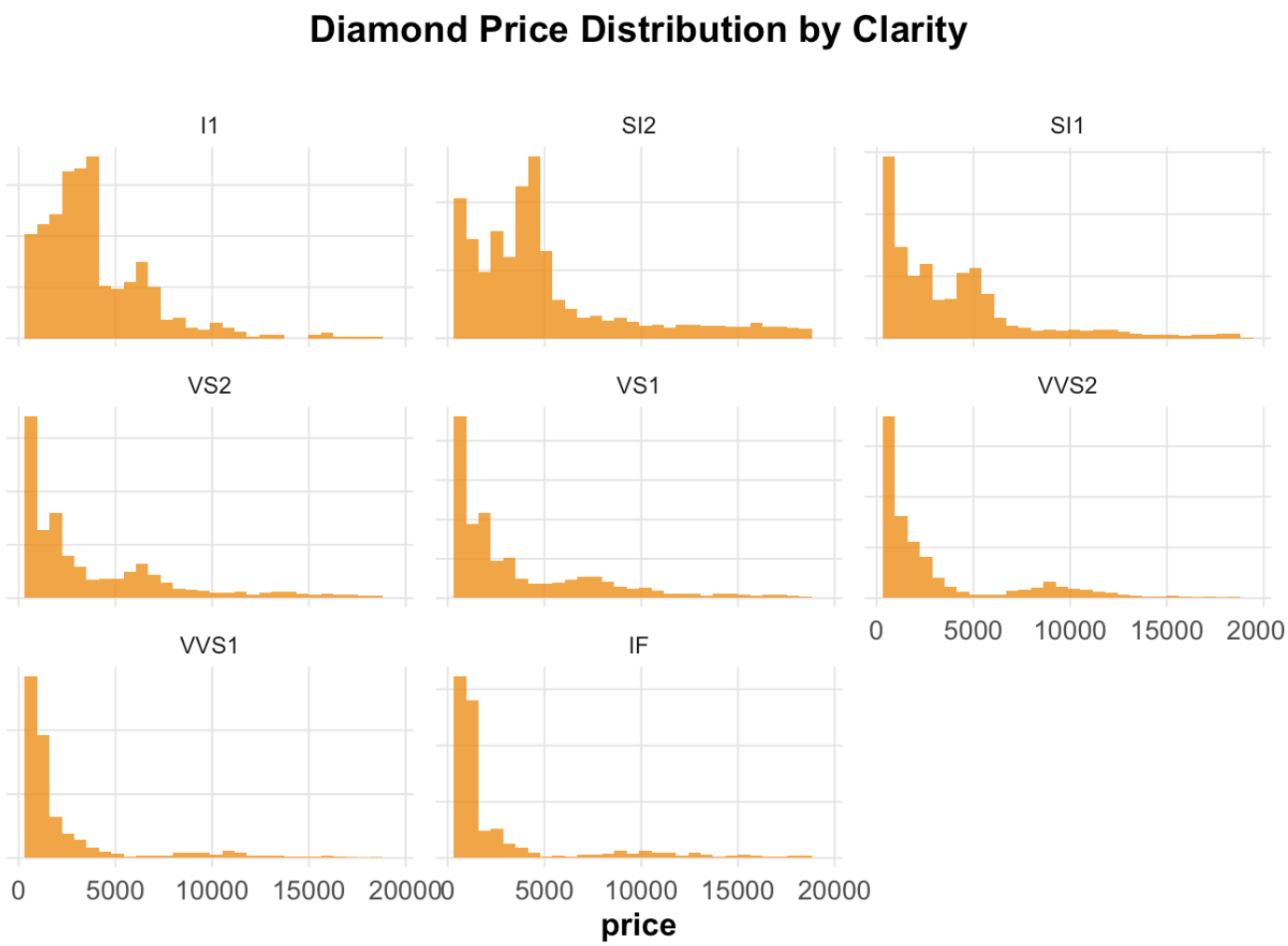
```
ggplot(diamonds, aes(x = price, y = "", color = clarity)) +
  geom_boxplot(width = 0.3,
               fill = diamond_gold,
               alpha = 0.8,
               outlier.color = diamond_blue,
               outlier.alpha = 0.6) +
  labs(
    title = "Diamond Price Distribution by Clarity",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```




```
ggplot(diamonds, aes(x=price)) +
  geom_histogram(width = 0.3,
                 fill = diamond_gold,
                 alpha = 0.8,
                 outlier.color = diamond_blue,
                 outlier.alpha = 0.6) +
  facet_wrap(~clarity, scales = "free_y") +
    labs(
    title = "Diamond Price Distribution by Clarity",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
Warning in geom_histogram(width = 0.3, fill = diamond_gold, alpha = 0.8, :
```

```
Ignoring unknown parameters: `outlier.colour` and `outlier.alpha`

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(diamonds, aes(x=price, y = "", color = clarity)) +
  geom_violin(width = 0.3,
              fill = diamond_gold,
              alpha = 0.8,
              outlier.color = diamond_blue,
              outlier.alpha = 0.6) +
    labs(
    title = "Diamond Price Distribution by Clarity",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
Warning in geom_violin(width = 0.3, fill = diamond_gold, alpha = 0.8,
outlier.color = diamond_blue, : Ignoring unknown parameters: `outlier.colour`
and `outlier.alpha`
```

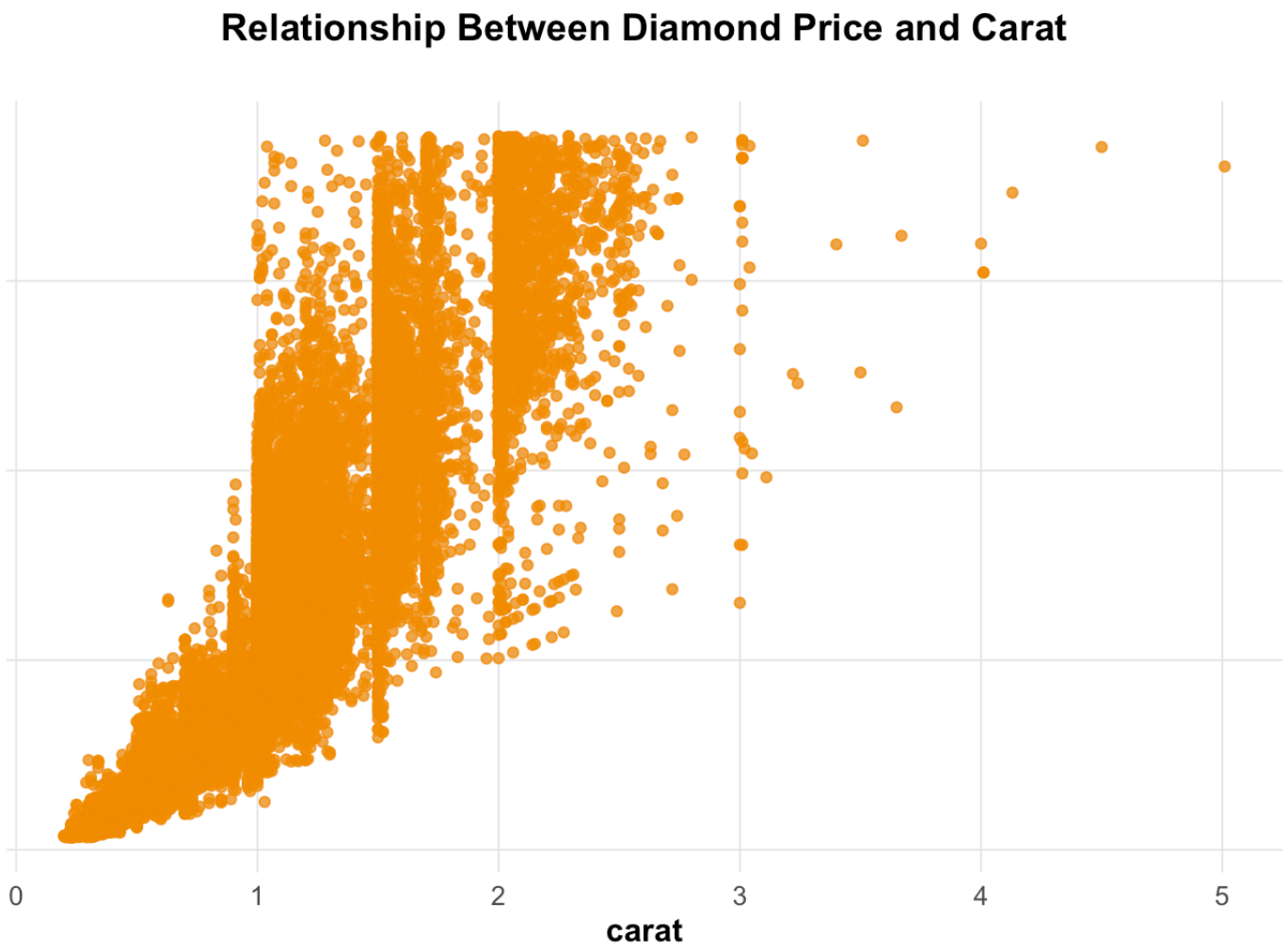
Price's relationship with numerical variables

Having investigated price's relationship with the categorical variables, the next step is to investigate its relationship with the other numerical variables through a mixture of regressions, scatterplots, hexbin plots, and heatmaps. It is clear that price is positively associated with carat, depth, x, y, z and negative associated with depth, though these relationships are usually nonlinear and are oddly shaped.

```
install.packages("hexbin")
```

```
The downloaded binary packages are in
    /var/folders/jx/19wzy4r974zgwcx8kgshk9400000gn/T//RtmpQyxrpZ/downloaded_packages
```

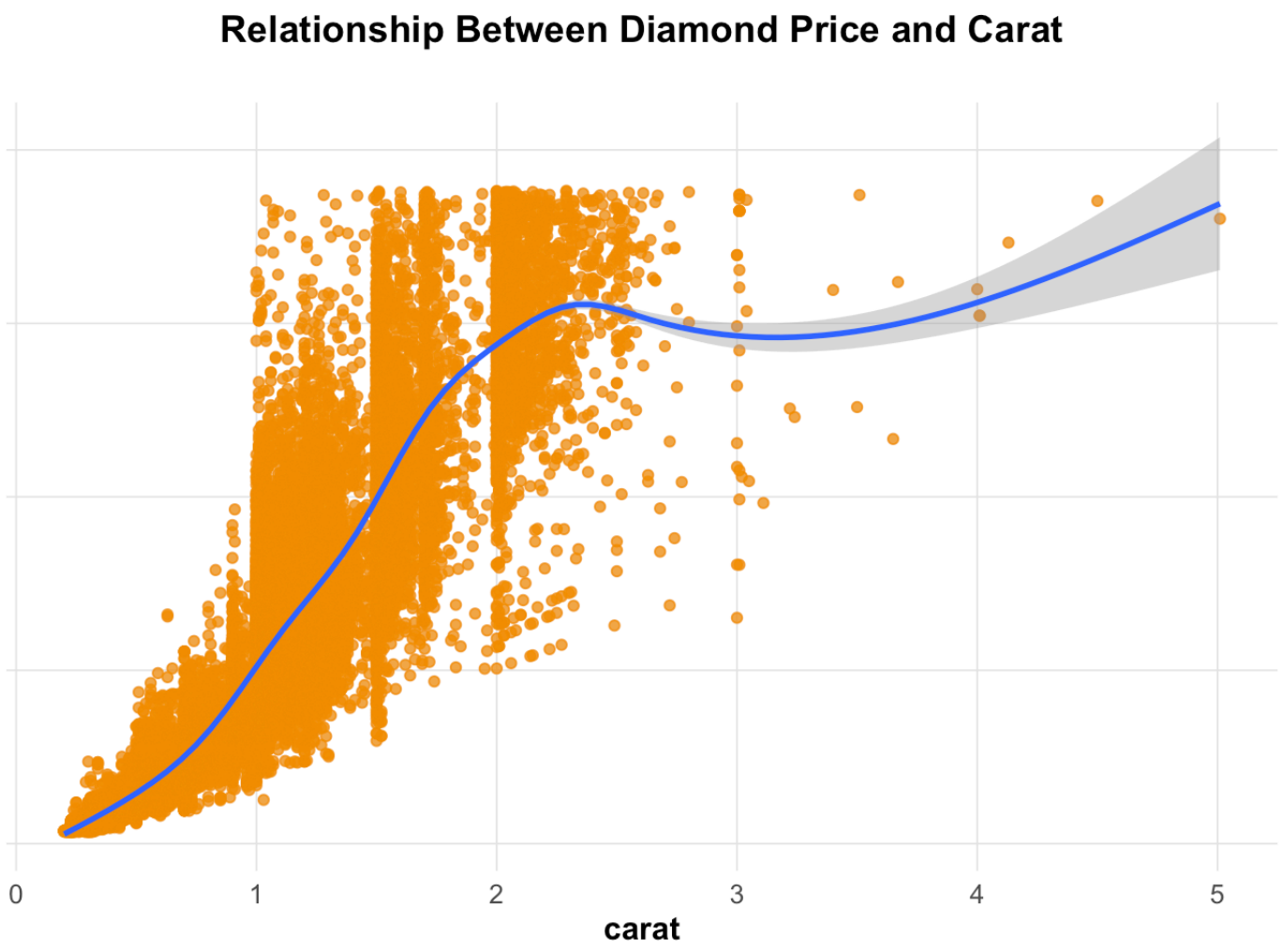
```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point(color = diamond_gold,
             alpha = 0.8) +
    labs(
    title = "Relationship Between Diamond Price and Carat",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```




```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point(color = diamond_gold,
             alpha = 0.8) +
```

```
  geom_smooth() +
    labs(
    title = "Relationship Between Diamond Price and Carat",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

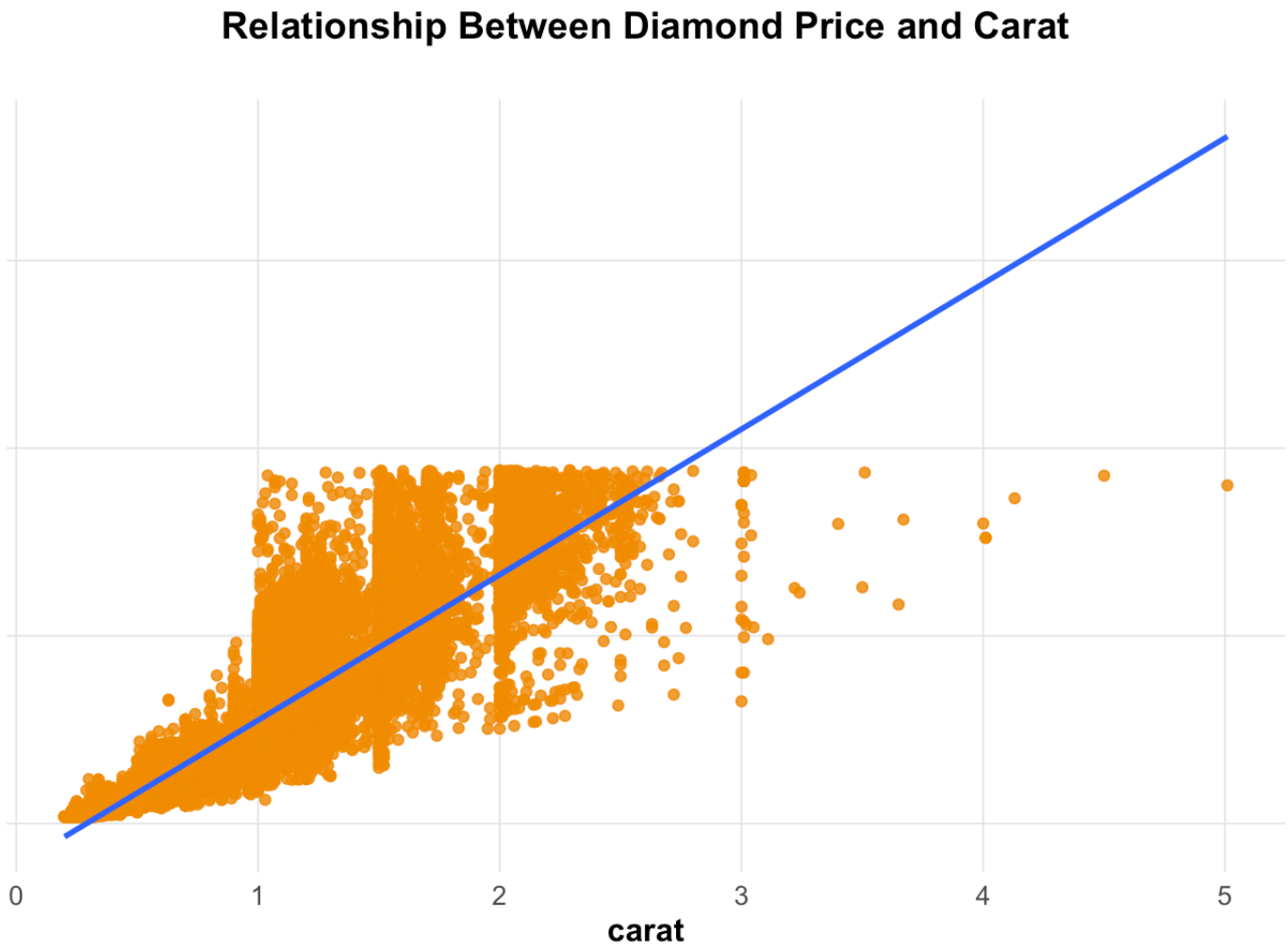
```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```




```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point(color = diamond_gold,
             alpha = 0.86) +
  geom_smooth(method = "lm") +
    labs(
    title = "Relationship Between Diamond Price and Carat",
```

```
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

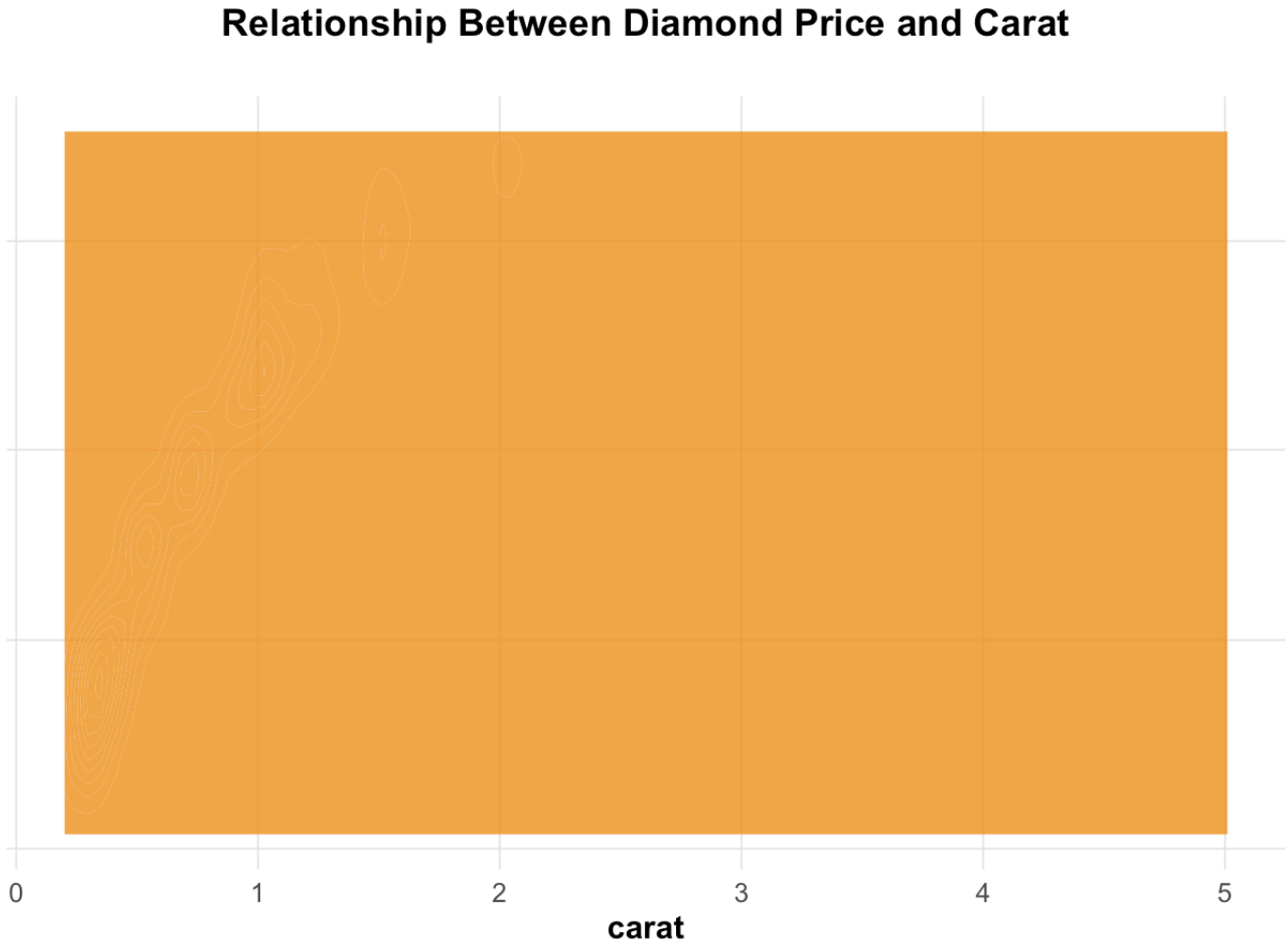
```
`geom_smooth()` using formula = 'y ~ x'
```




```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_density2d_filled(width = 0.3,
              fill = diamond_gold,
              alpha = 0.8,
              outlier.color = diamond_blue,
              outlier.alpha = 0.6) +
  scale_y_log10() +
    labs(
    title = "Relationship Between Diamond Price and Carat",
```

```
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

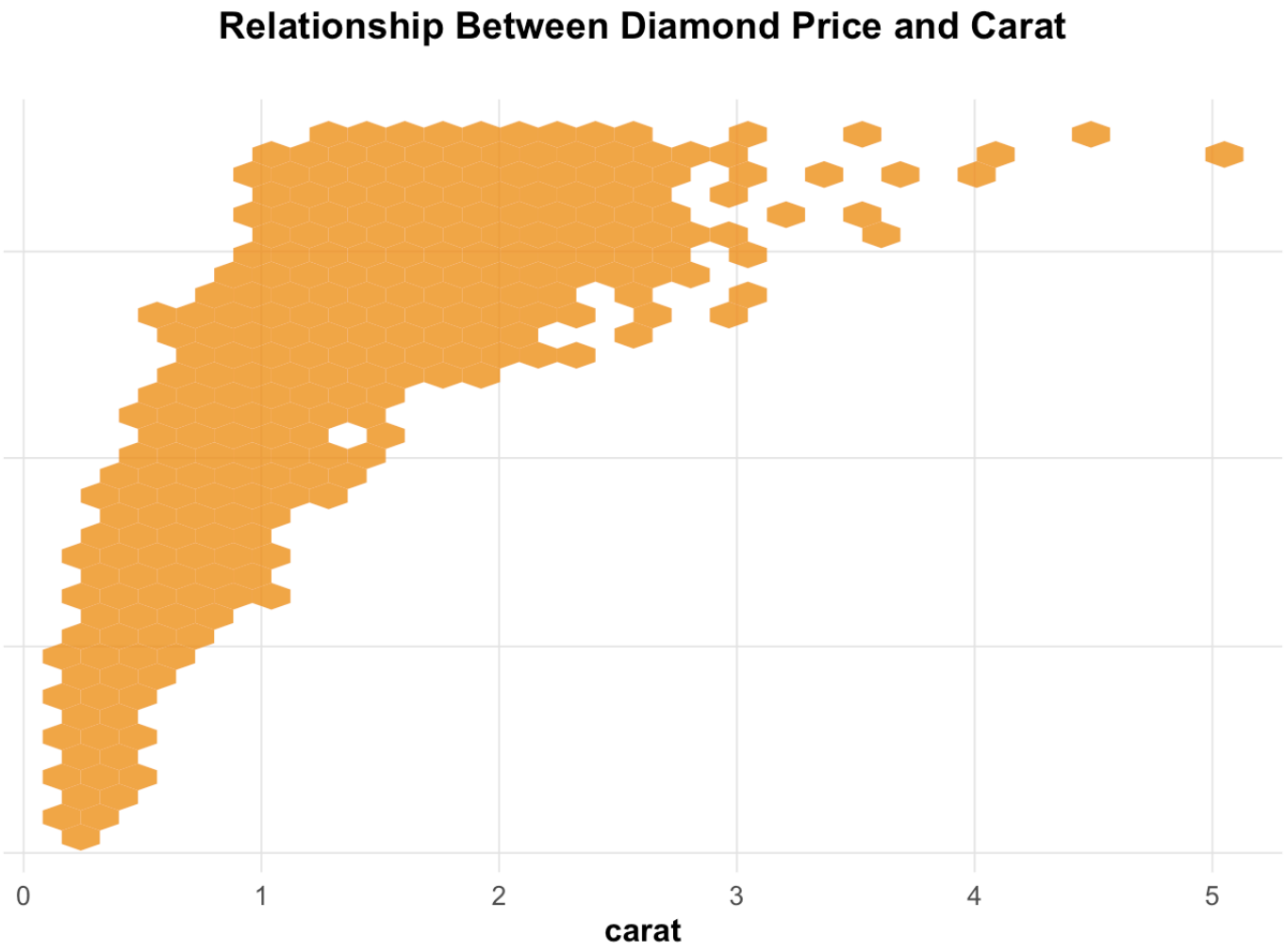
```
Warning in geom_density2d_filled(width = 0.3, fill = diamond_gold, alpha = 0.8,
: Ignoring unknown parameters: `width`, `outlier.colour`, and `outlier.alpha`
```




```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_hex(width = 0.3,
               fill = diamond_gold,
               alpha = 0.8,
               outlier.color = diamond_blue,
               outlier.alpha = 0.6) +
  scale_y_log10() +
    labs(
```

```
    title = "Relationship Between Diamond Price and Carat",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

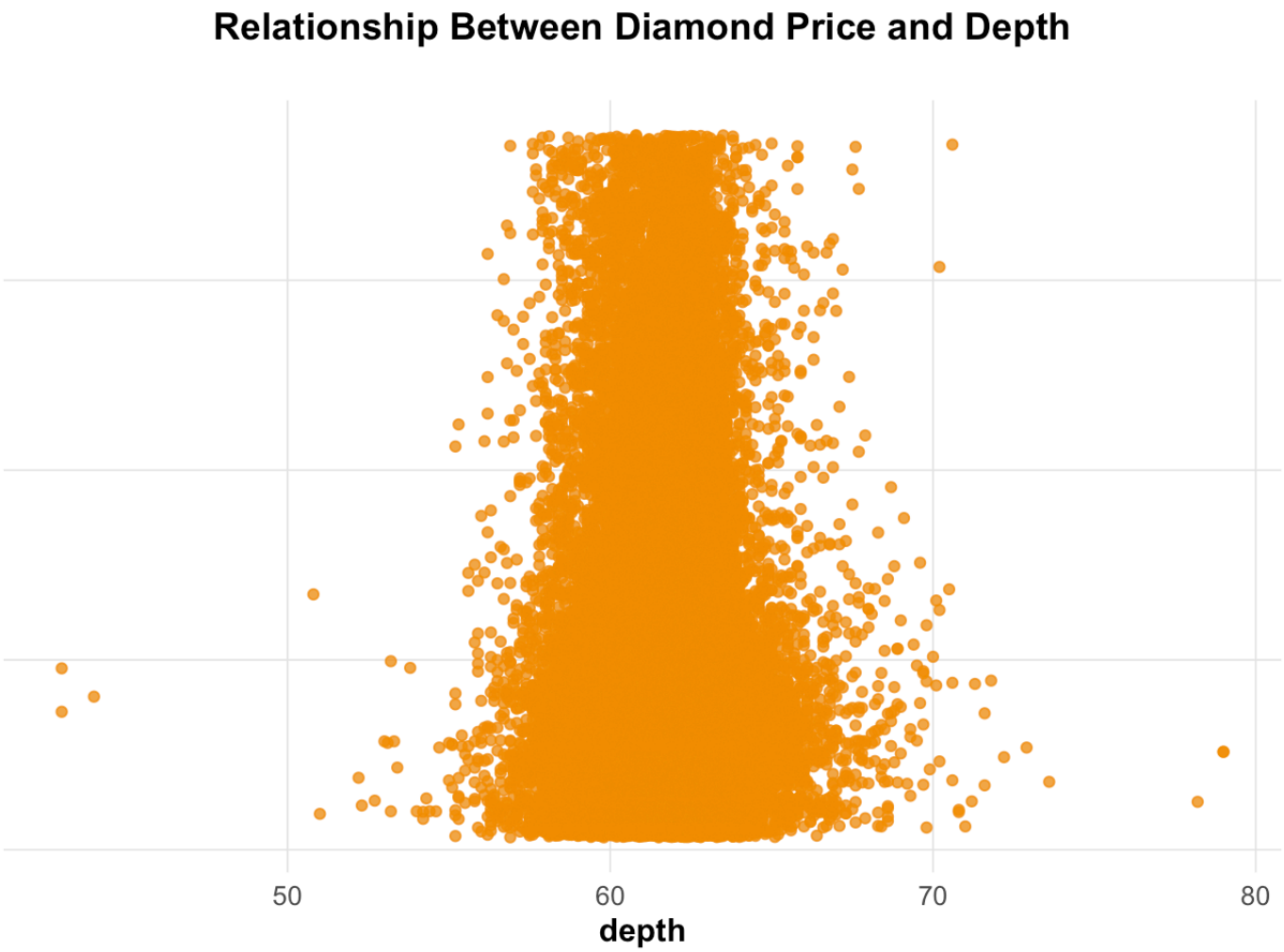
```
Warning in geom_hex(width = 0.3, fill = diamond_gold, alpha = 0.8,
outlier.color = diamond_blue, : Ignoring unknown parameters: `width`,
`outlier.colour`, and `outlier.alpha`
```




```
install.packages("hexbin")
```

```
The downloaded binary packages are in
    /var/folders/jx/19wzy4r974zgwcx8kgshk9400000gn/T//RtmpQyxrpZ/downloaded_packages
```

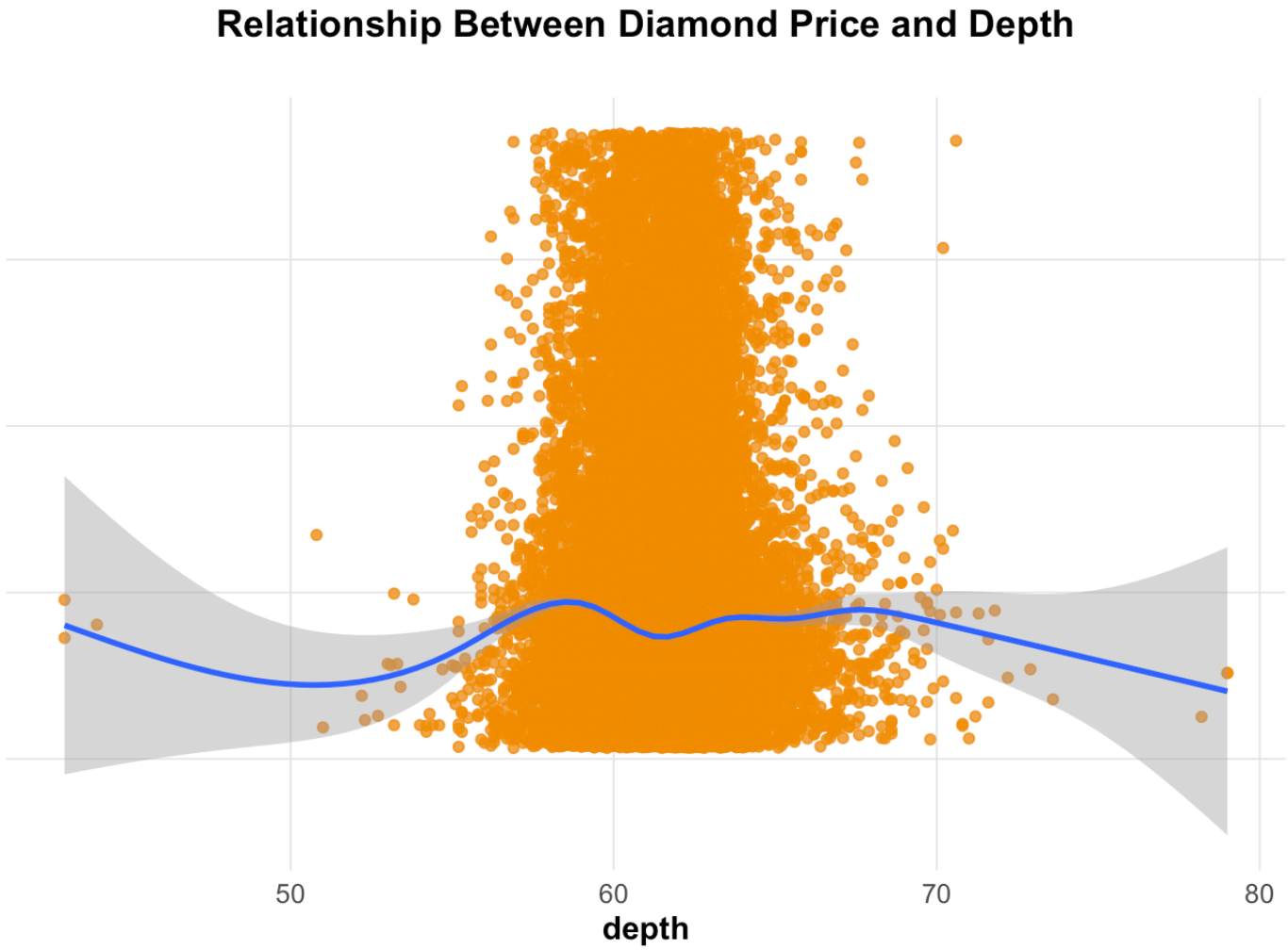
```
ggplot(diamonds, aes(x = depth, y = price)) +
  geom_point(color = diamond_gold,
             alpha = 0.8) +
    labs(
    title = "Relationship Between Diamond Price and Depth",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```




```
ggplot(diamonds, aes(x = depth, y = price)) +
  geom_point(color = diamond_gold,
             alpha = 0.8) +
  geom_smooth() +
    labs(
    title = "Relationship Between Diamond Price and Depth",
```

```
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

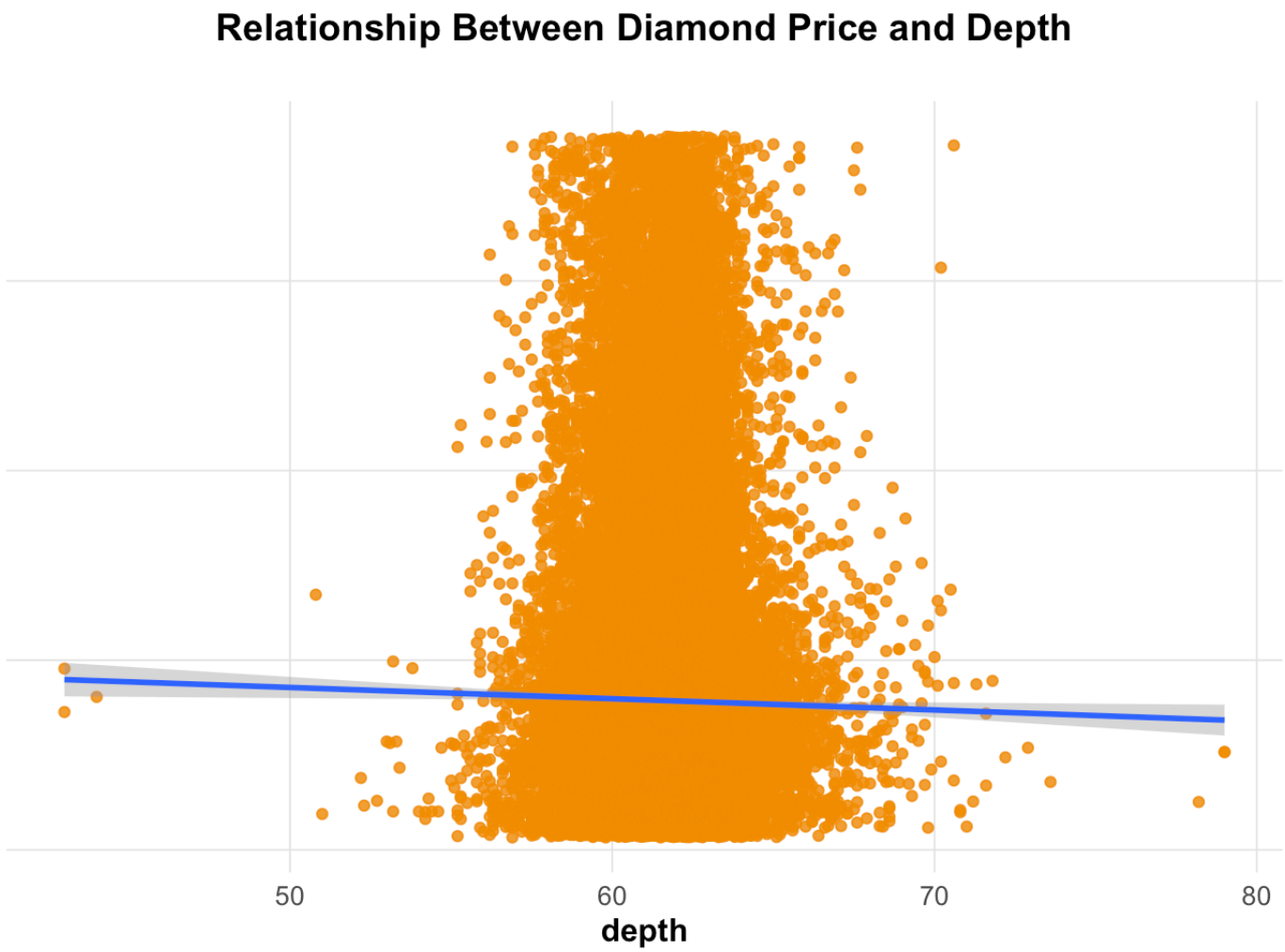
```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```




```
ggplot(diamonds, aes(x = depth, y = price)) +
  geom_point(color = diamond_gold,
             alpha = 0.86) +
  geom_smooth(method = "lm") +
    labs(
    title = "Relationship Between Diamond Price and Depth",
    y = ""
  ) +
  custom_theme +
```

```
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

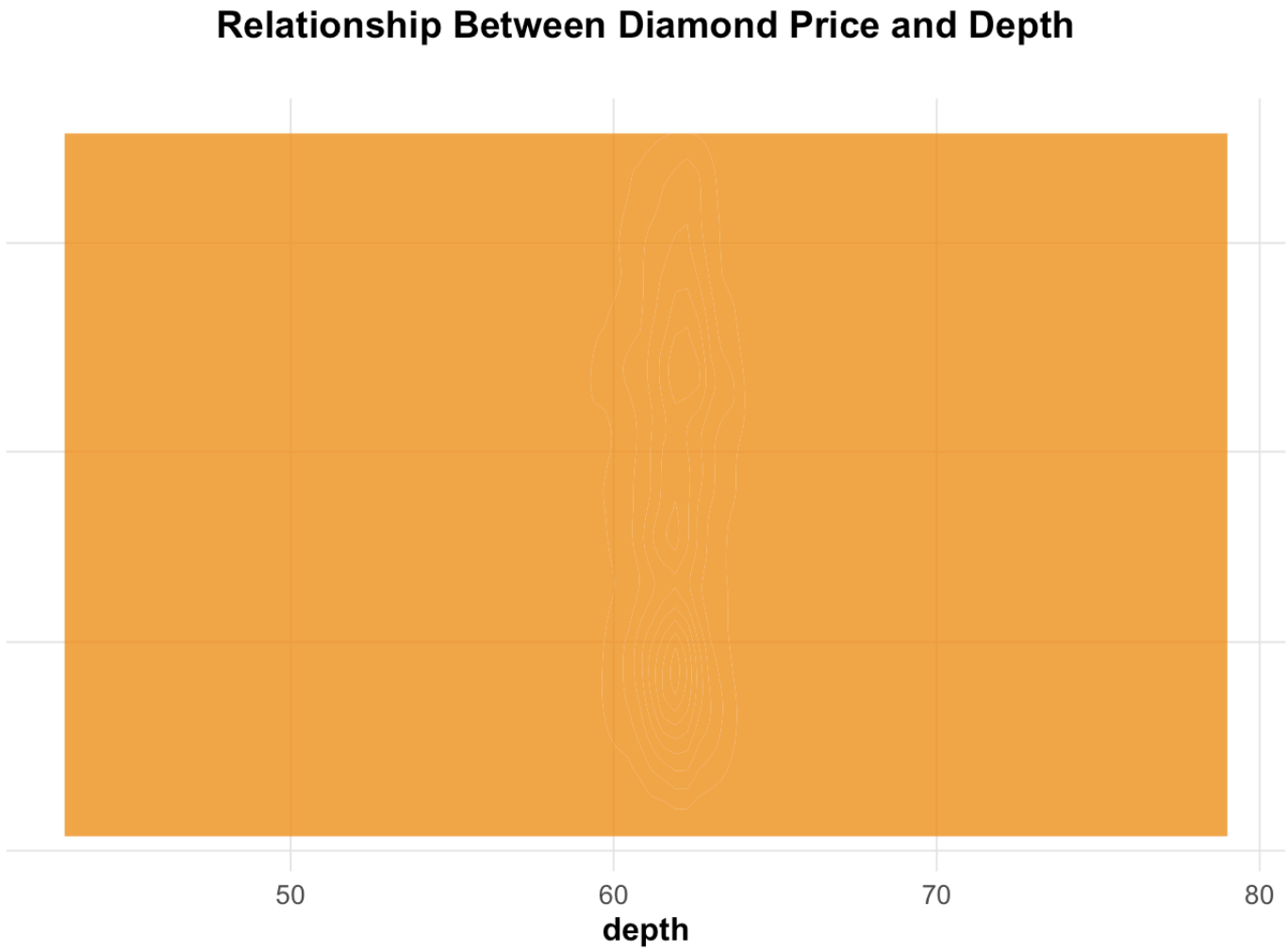
```
`geom_smooth()` using formula = 'y ~ x'
```




```
ggplot(diamonds, aes(x = depth, y = price)) +
  geom_density2d_filled(width = 0.3,
              fill = diamond_gold,
              alpha = 0.8,
              outlier.color = diamond_blue,
              outlier.alpha = 0.6) +
  scale_y_log10() +
    labs(
    title = "Relationship Between Diamond Price and Depth",
    y = ""
  ) +
  custom_theme +
```

```
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

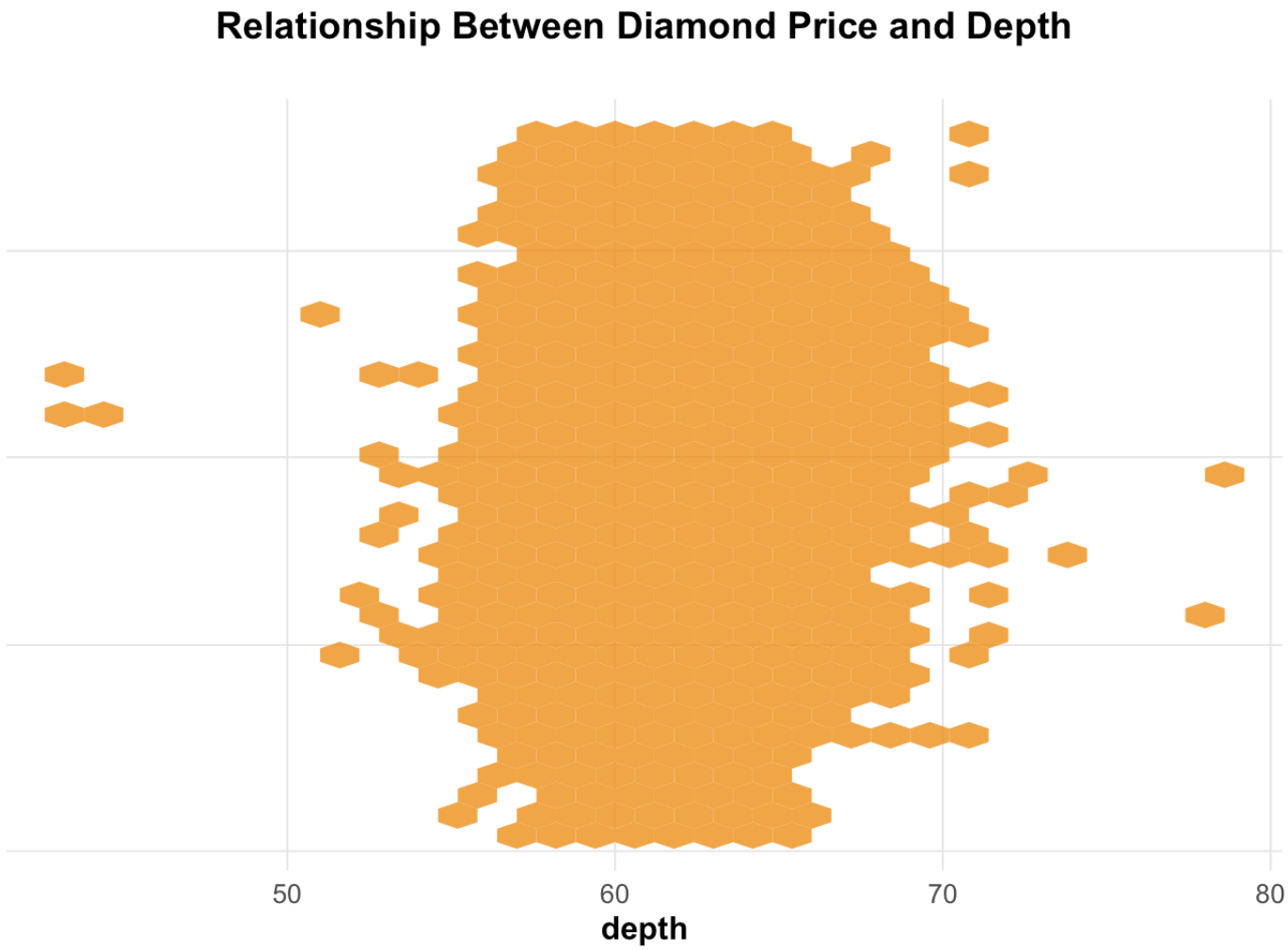
```
Warning in geom_density2d_filled(width = 0.3, fill = diamond_gold, alpha = 0.8,
: Ignoring unknown parameters: `width`, `outlier.colour`, and `outlier.alpha`
```




```
ggplot(diamonds, aes(x = depth, y = price)) +
  geom_hex(width = 0.3,
               fill = diamond_gold,
               alpha = 0.8,
               outlier.color = diamond_blue,
               outlier.alpha = 0.6) +
  scale_y_log10() +
    labs(
    title = "Relationship Between Diamond Price and Depth",
    y = ""
  ) +
```

```
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
Warning in geom_hex(width = 0.3, fill = diamond_gold, alpha = 0.8,
outlier.color = diamond_blue, : Ignoring unknown parameters: `width`,
`outlier.colour`, and `outlier.alpha`
```




```
install.packages("hexbin")
```

```
The downloaded binary packages are in
    /var/folders/jx/19wzy4r974zgwcx8kgshk9400000gn/T//RtmpQyxrpZ/downloaded_packages
```

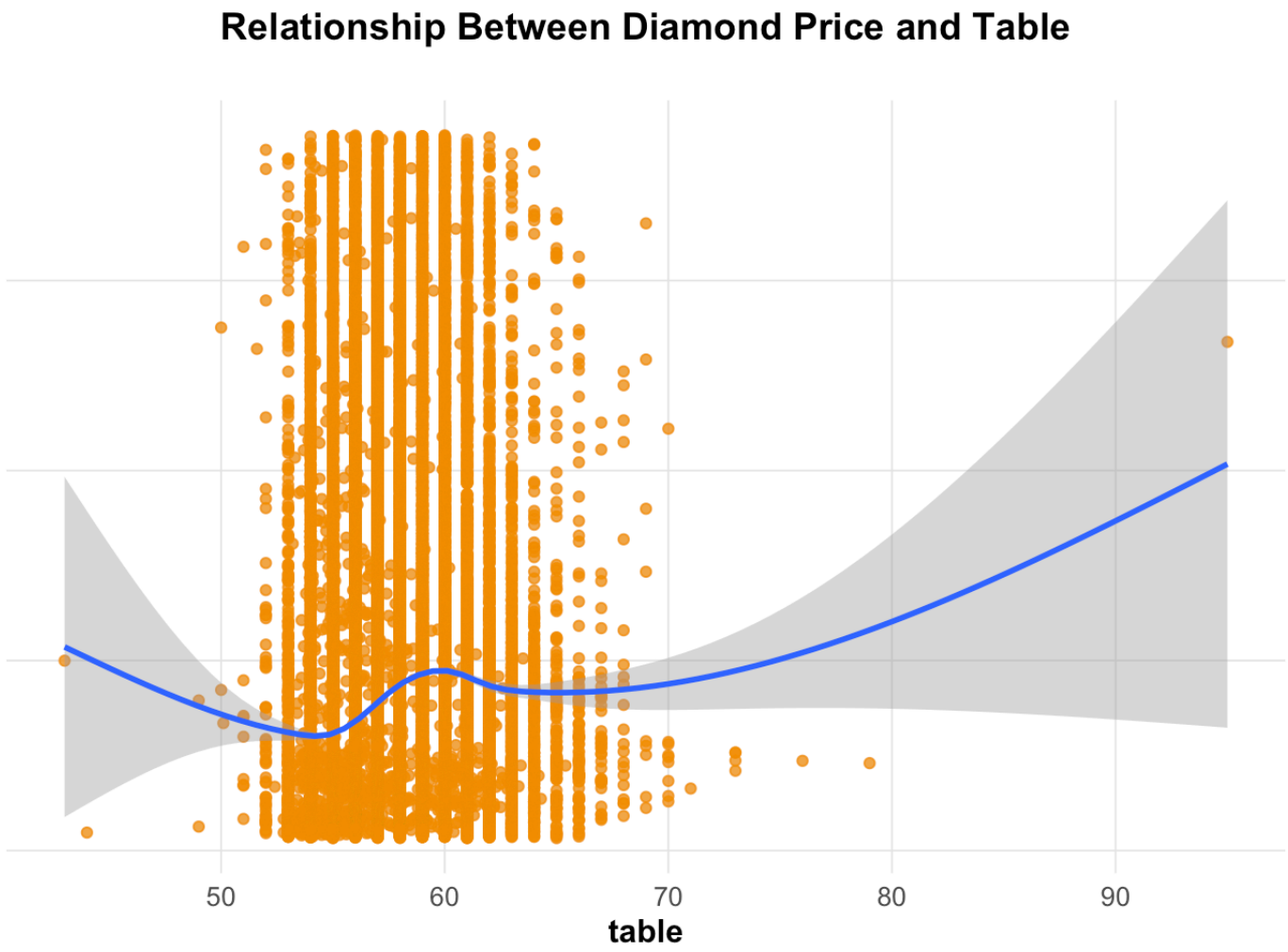
```
ggplot(diamonds, aes(x = table, y = price)) +
  geom_point(color = diamond_gold,
```

```
              alpha = 0.8) +
    labs(
    title = "Relationship Between Diamond Price and Table",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```


```
ggplot(diamonds, aes(x = table, y = price)) +
  geom_point(color = diamond_gold,
             alpha = 0.8) +
  geom_smooth() +
    labs(
    title = "Relationship Between Diamond Price and Table",
    y = ""
  ) +
```

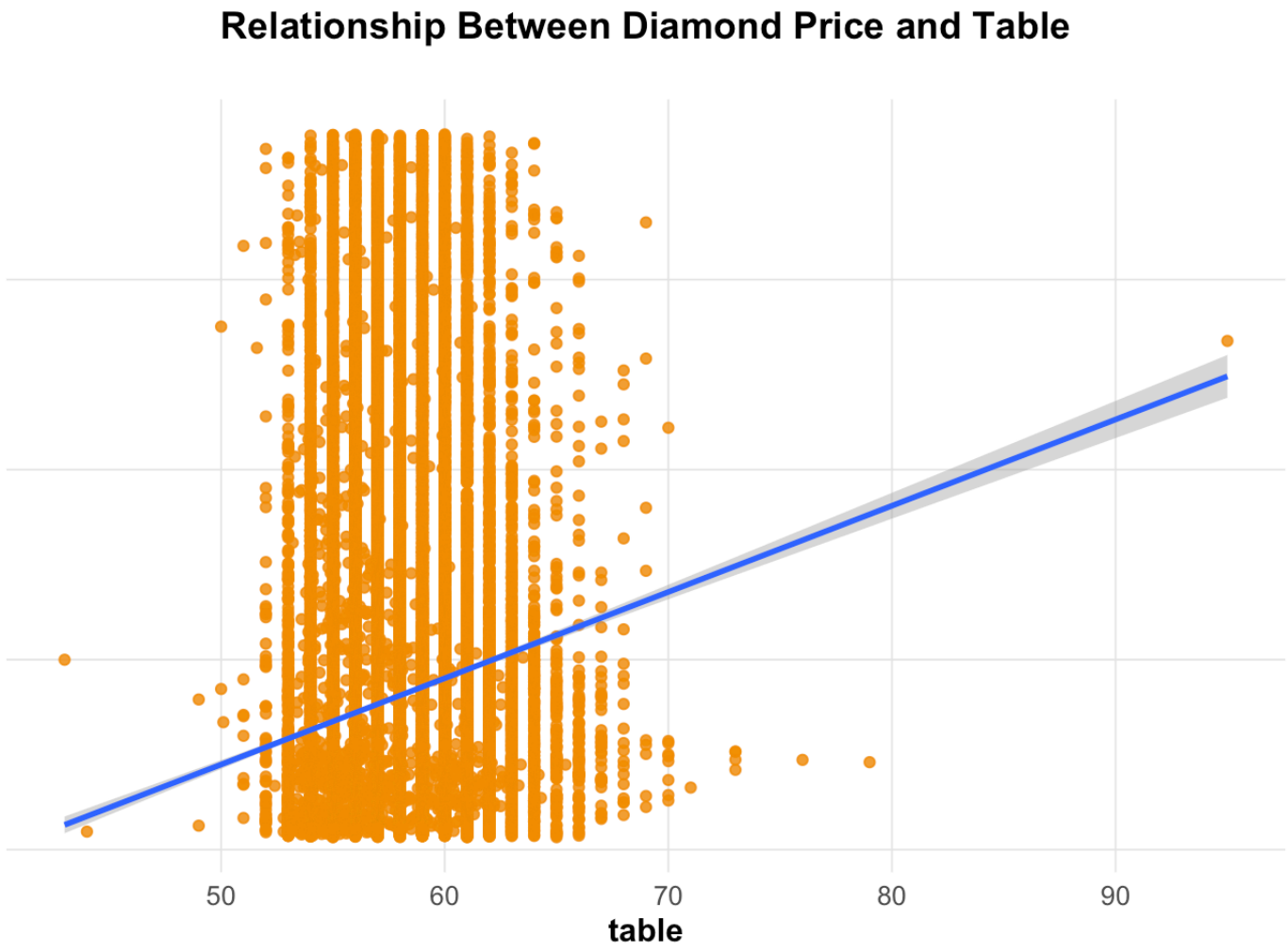
```
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



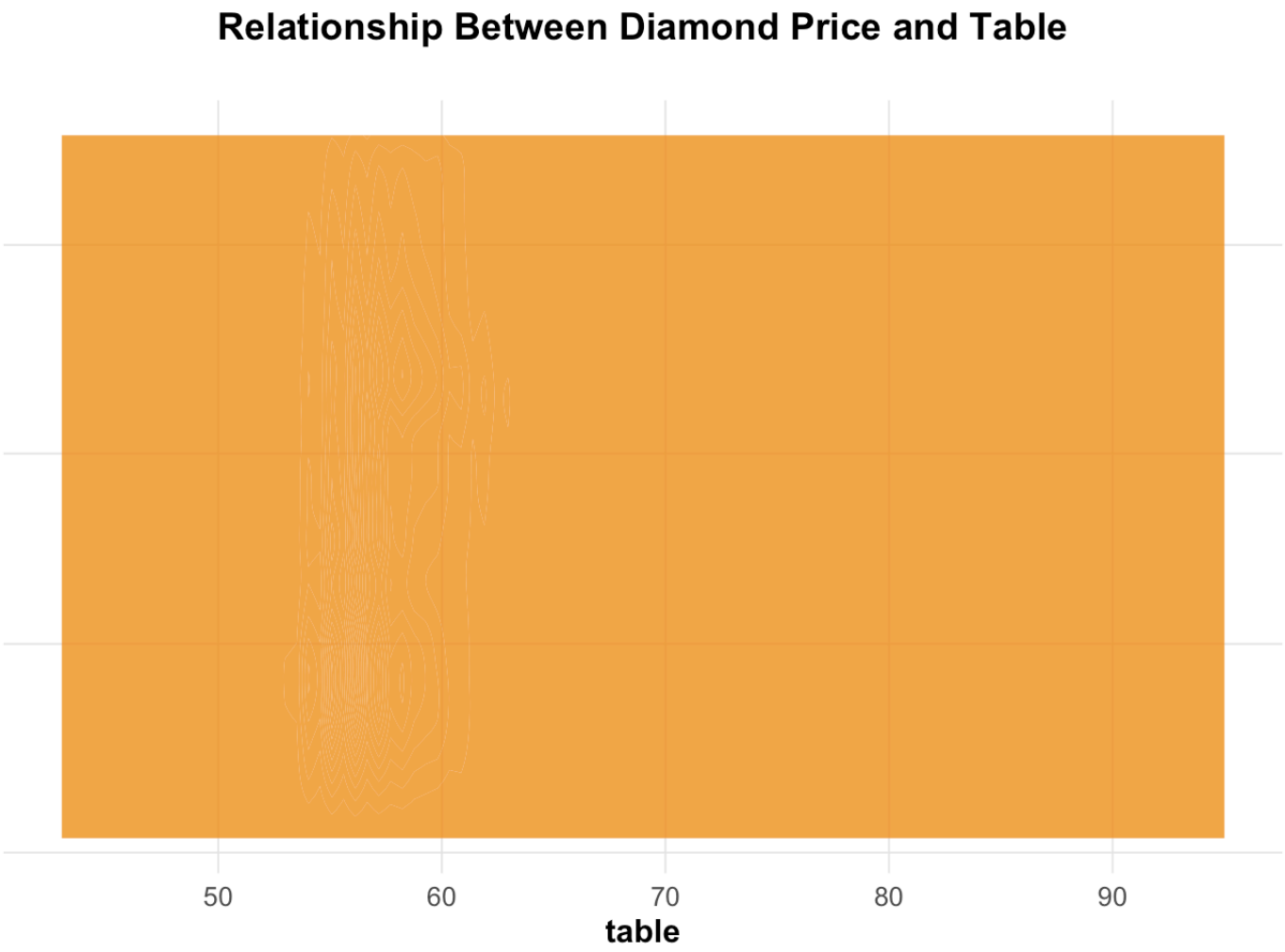

```
ggplot(diamonds, aes(x = table, y = price)) +
  geom_point(color = diamond_gold,
             alpha = 0.86) +
  geom_smooth(method = "lm") +
    labs(
    title = "Relationship Between Diamond Price and Table",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
`geom_smooth()` using formula = 'y ~ x'
```



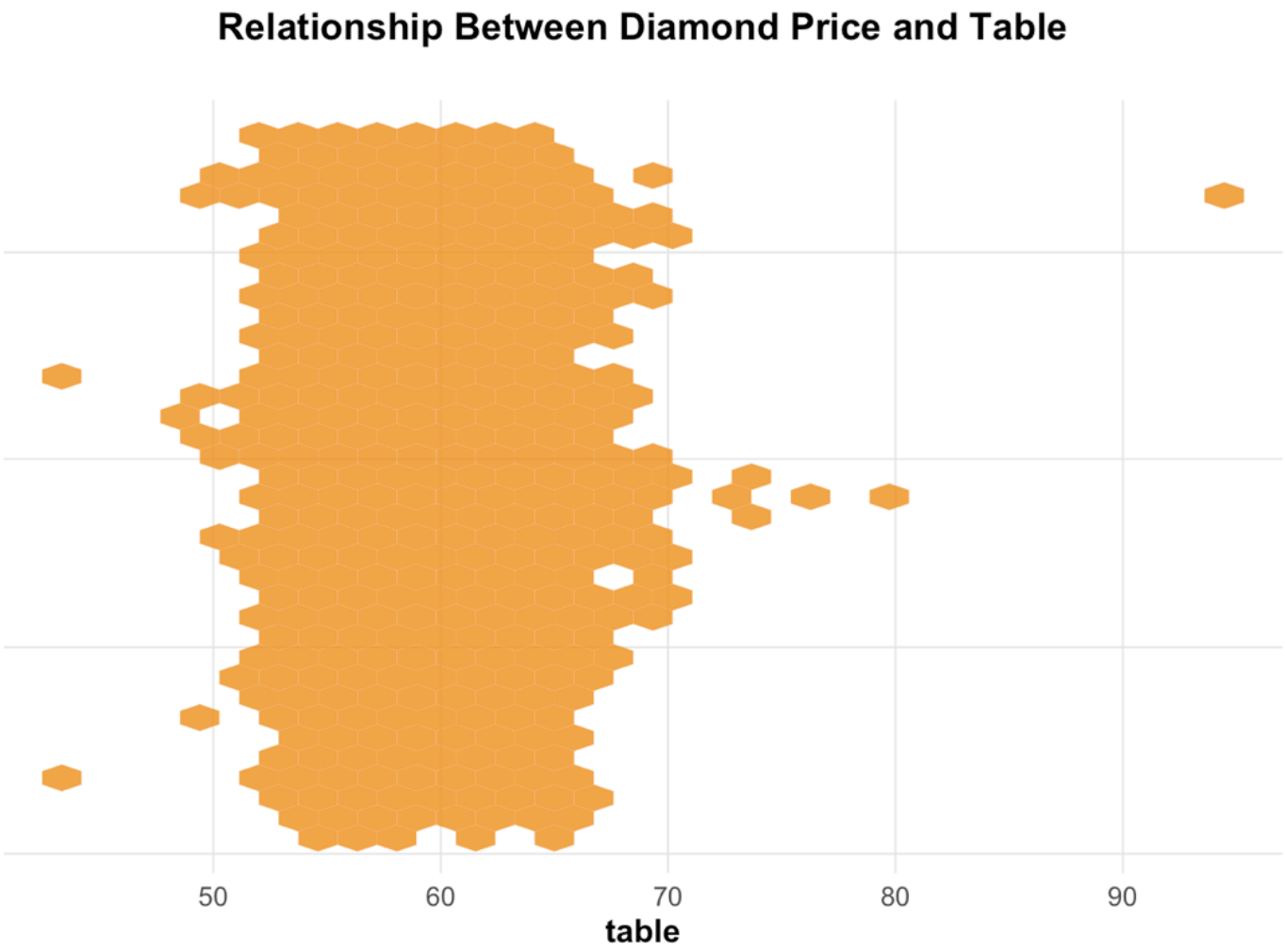

```
ggplot(diamonds, aes(x = table, y = price)) +
  geom_density2d_filled(width = 0.3,
                fill = diamond_gold,
                alpha = 0.8,
                outlier.color = diamond_blue,
                outlier.alpha = 0.6) +
  scale_y_log10() +
    labs(
    title = "Relationship Between Diamond Price and Table",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
Warning in geom_density2d_filled(width = 0.3, fill = diamond_gold, alpha = 0.8,
: Ignoring unknown parameters: `width`, `outlier.colour`, and `outlier.alpha`
```




```
ggplot(diamonds, aes(x = table, y = price)) +
  geom_hex(width = 0.3,
               fill = diamond_gold,
               alpha = 0.8,
               outlier.color = diamond_blue,
               outlier.alpha = 0.6) +
  scale_y_log10() +
    labs(
    title = "Relationship Between Diamond Price and Table",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
Warning in geom_hex(width = 0.3, fill = diamond_gold, alpha = 0.8,
outlier.color = diamond_blue, : Ignoring unknown parameters: `width`,
`outlier.colour`, and `outlier.alpha`
```

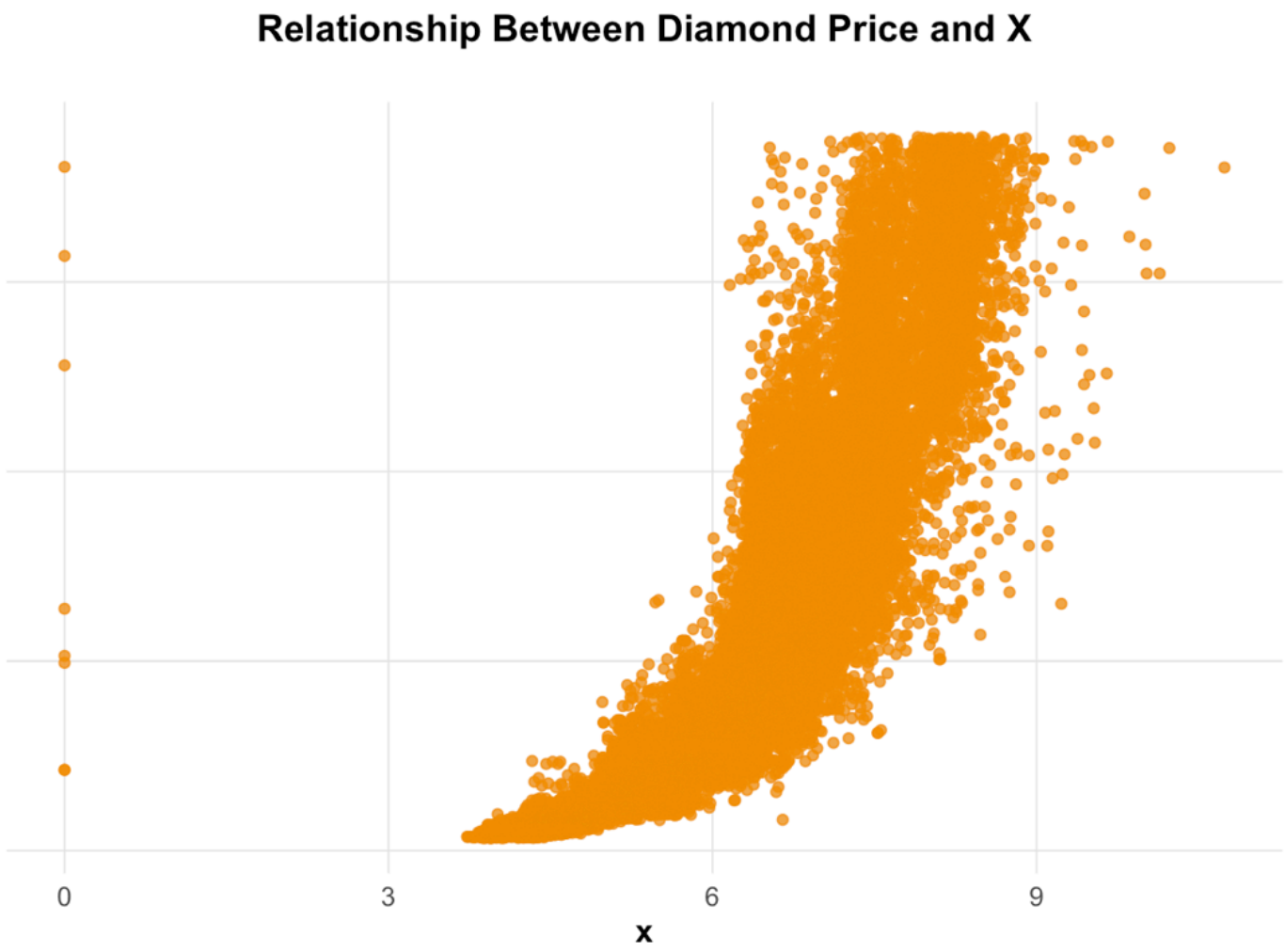



```
install.packages("hexbin")
```

```
The downloaded binary packages are in
    /var/folders/jx/19wzy4r974zgwcx8kgshk9400000gn/T//RtmpQyxrpZ/downloaded_packages
```

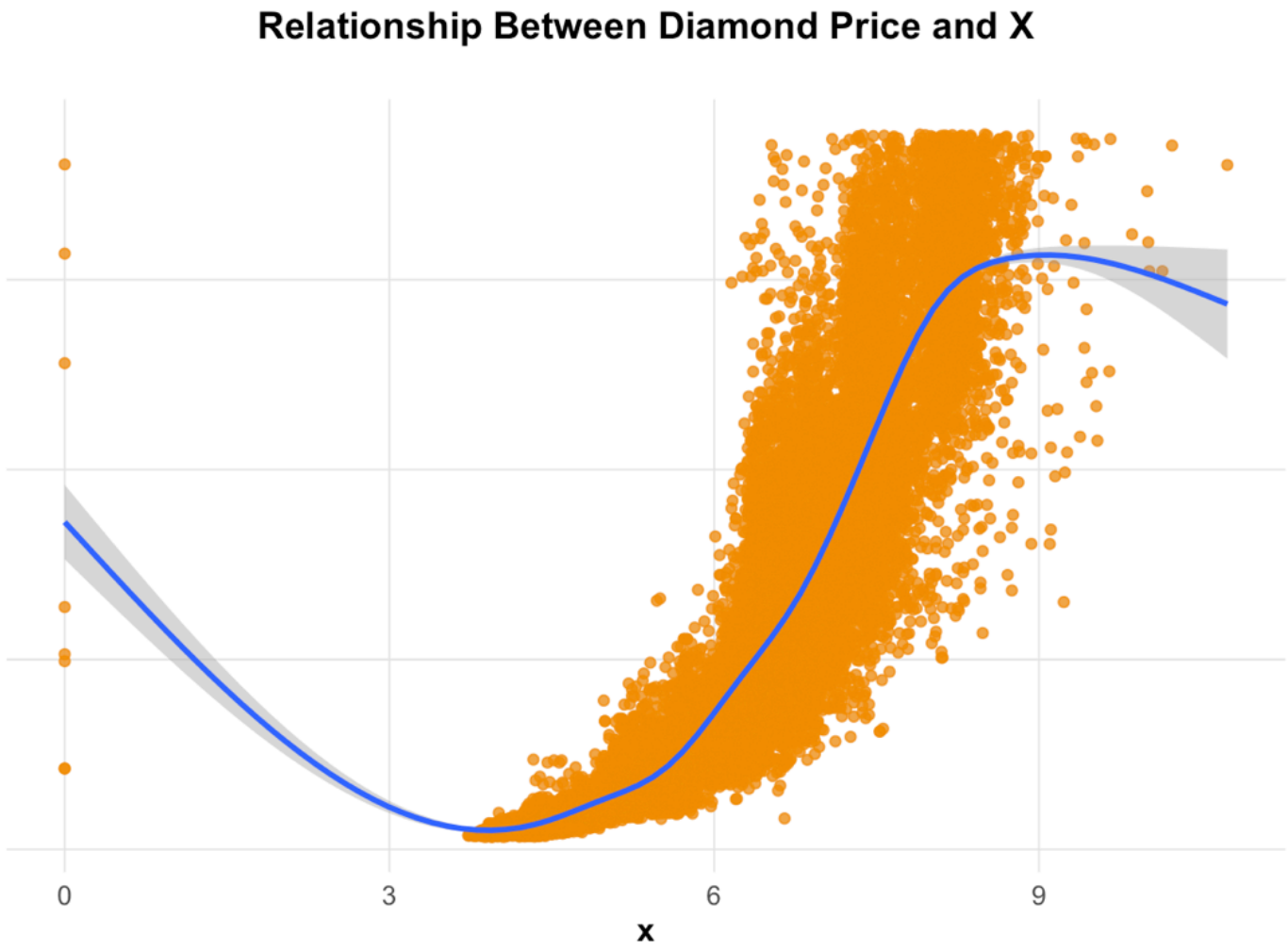
```
ggplot(diamonds, aes(x = x, y = price)) +
  geom_point(color = diamond_gold,
             alpha = 0.8) +
    labs(
    title = "Relationship Between Diamond Price and X",
    y = ""
```

```
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```



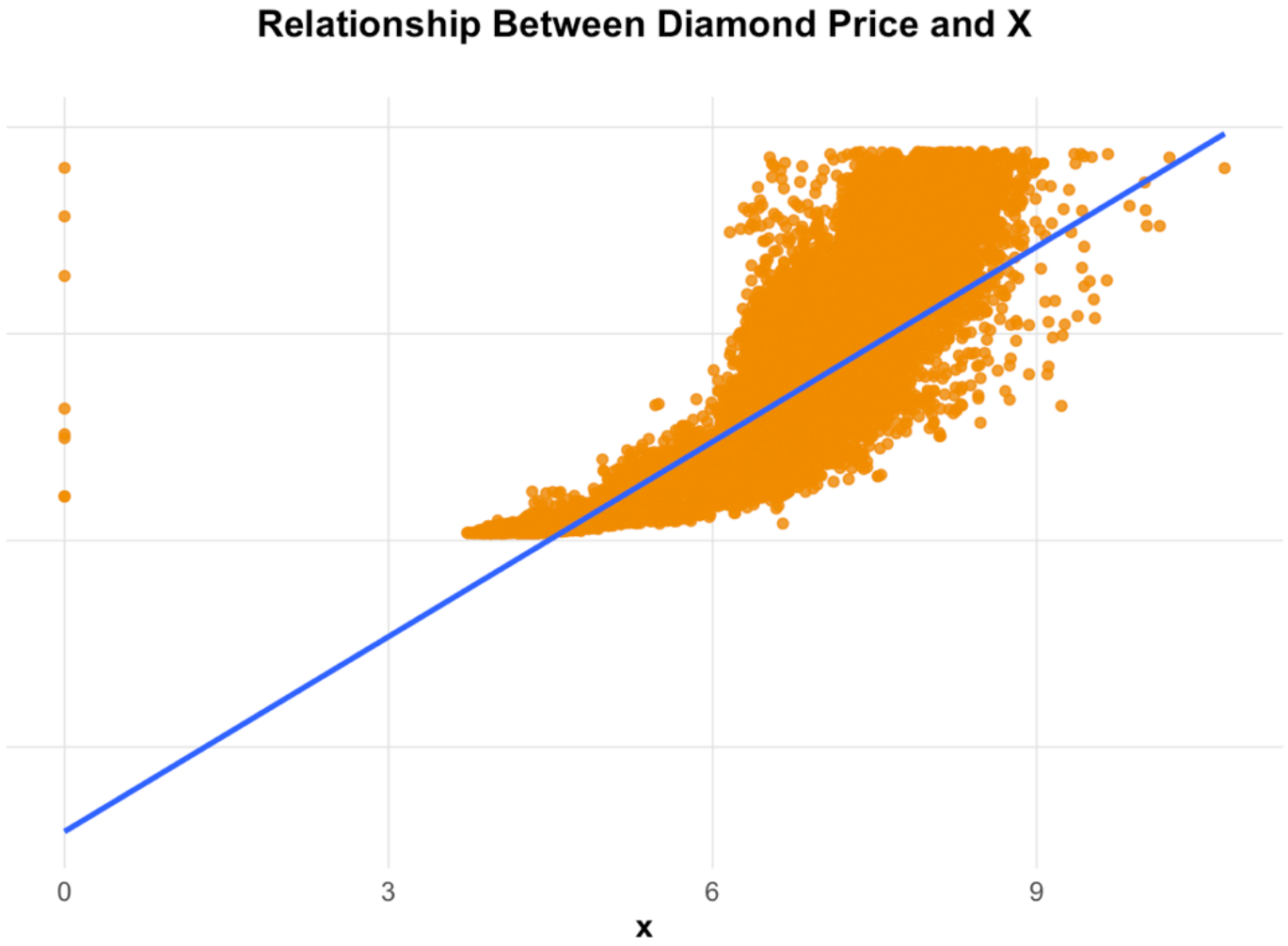

```
ggplot(diamonds, aes(x = x, y = price)) +
  geom_point(color = diamond_gold,
             alpha = 0.8) +
  geom_smooth() +
    labs(
    title = "Relationship Between Diamond Price and X",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



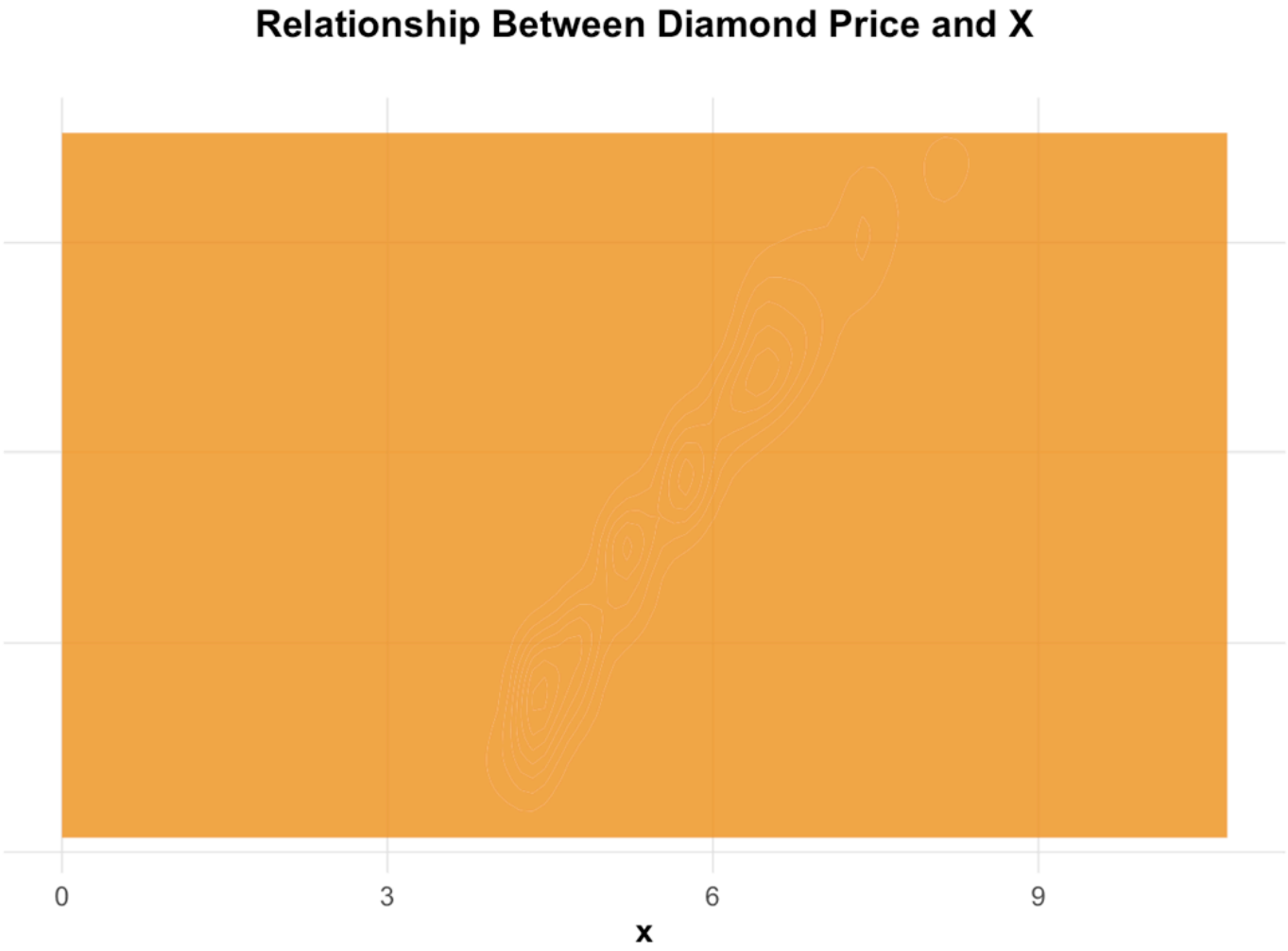

```
ggplot(diamonds, aes(x = x, y = price)) +
  geom_point(color = diamond_gold,
             alpha = 0.86) +
  geom_smooth(method = "lm") +
    labs(
    title = "Relationship Between Diamond Price and X",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
`geom_smooth()` using formula = 'y ~ x'
```




```
ggplot(diamonds, aes(x = x, y = price)) +
  geom_density2d_filled(width = 0.3,
               fill = diamond_gold,
               alpha = 0.8,
               outlier.color = diamond_blue,
               outlier.alpha = 0.6) +
  scale_y_log10() +
    labs(
    title = "Relationship Between Diamond Price and X",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
Warning in geom_density2d_filled(width = 0.3, fill = diamond_gold, alpha = 0.8,
: Ignoring unknown parameters: `width`, `outlier.colour`, and `outlier.alpha`
```




```
ggplot(diamonds, aes(x = x, y = price)) +
  geom_hex(width = 0.3,
               fill = diamond_gold,
               alpha = 0.8,
               outlier.color = diamond_blue,
               outlier.alpha = 0.6) +
  scale_y_log10() +
    labs(
    title = "Relationship Between Diamond Price and X",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
Warning in geom_hex(width = 0.3, fill = diamond_gold, alpha = 0.8,
outlier.color = diamond_blue, : Ignoring unknown parameters: `width`,
```

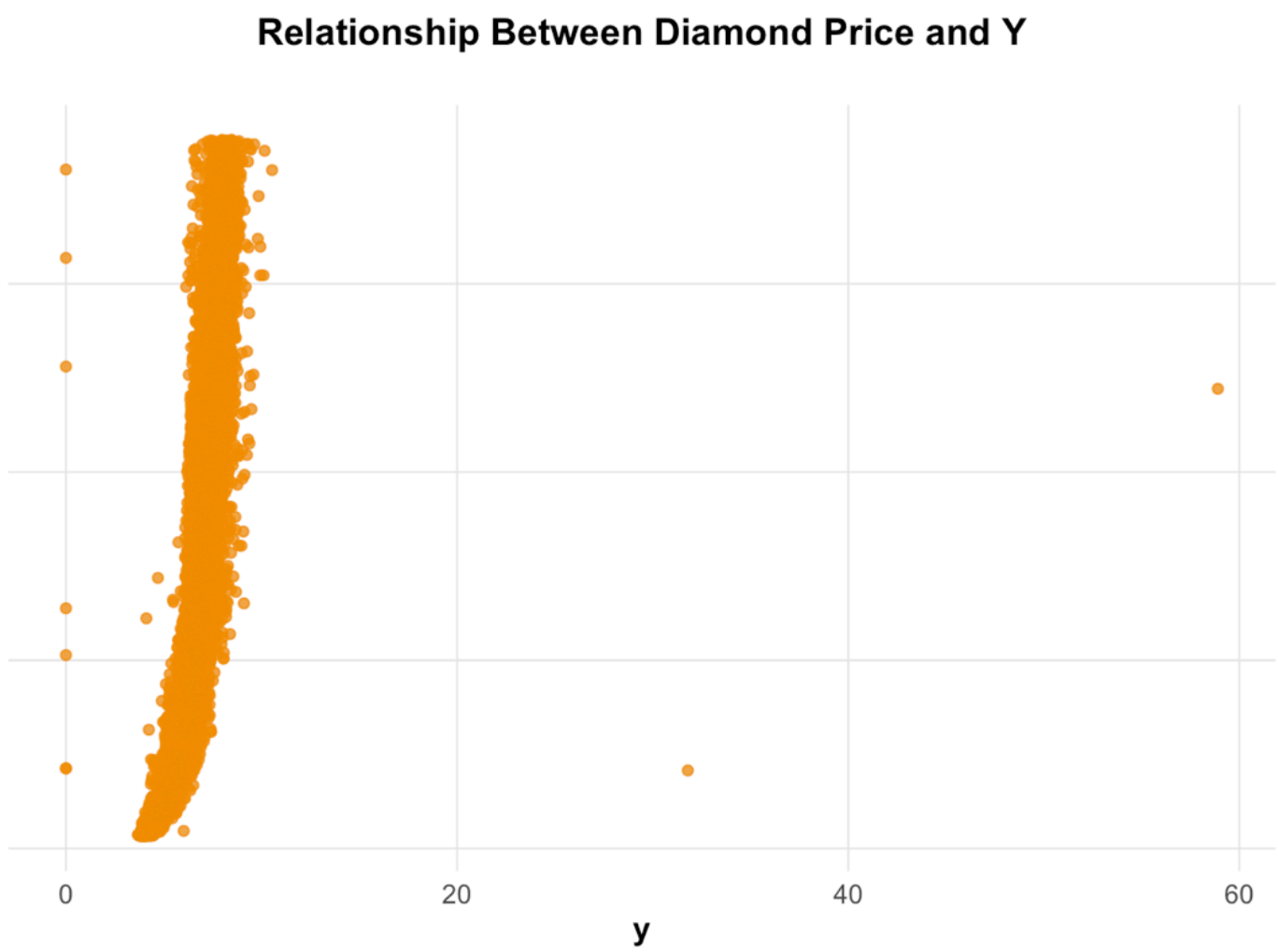
`outlier.colour`, and `outlier.alpha`


```
install.packages("hexbin")
```

```
The downloaded binary packages are in
    /var/folders/jx/19wzy4r974zgwcx8kgshk9400000gn/T//RtmpQyxrpZ/downloaded_packages
```

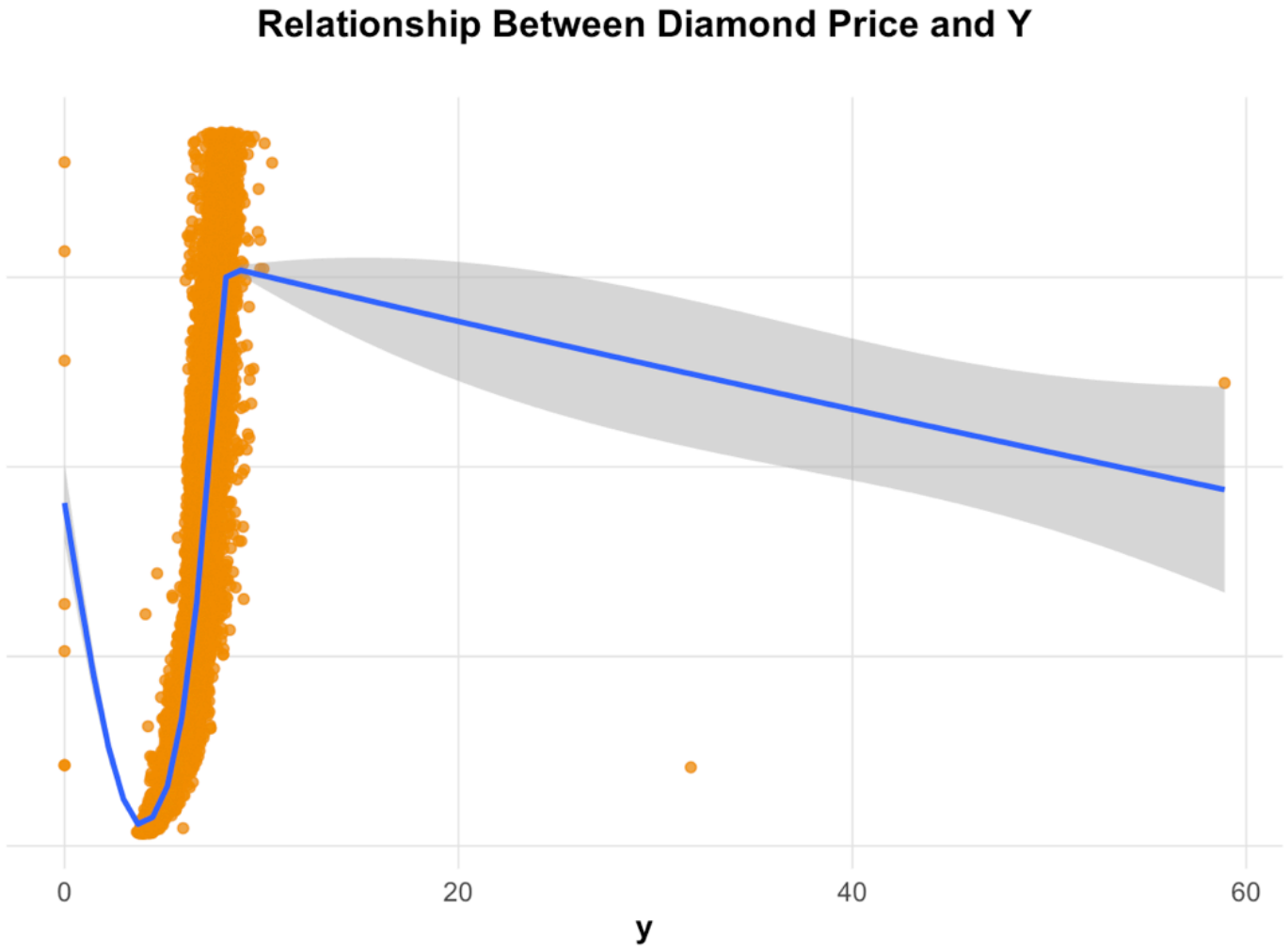
```
ggplot(diamonds, aes(x = y, y = price)) +
  geom_point(color = diamond_gold,
             alpha = 0.8) +
    labs(
    title = "Relationship Between Diamond Price and Y",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
```

```
        axis.ticks.y = element_blank())
```



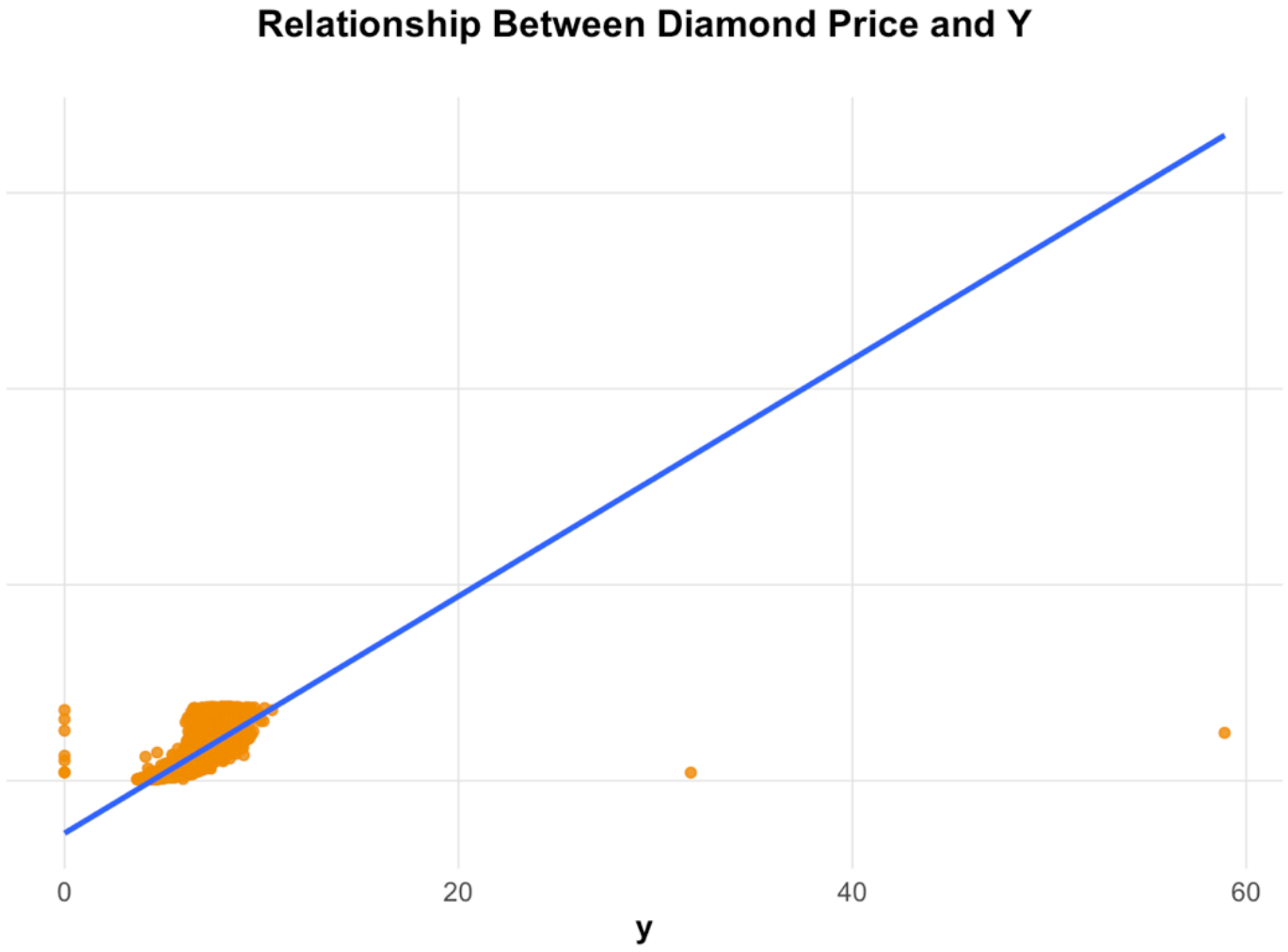

```
ggplot(diamonds, aes(x = y, y = price)) +
  geom_point(color = diamond_gold,
             alpha = 0.8) +
  geom_smooth() +
    labs(
    title = "Relationship Between Diamond Price and Y",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



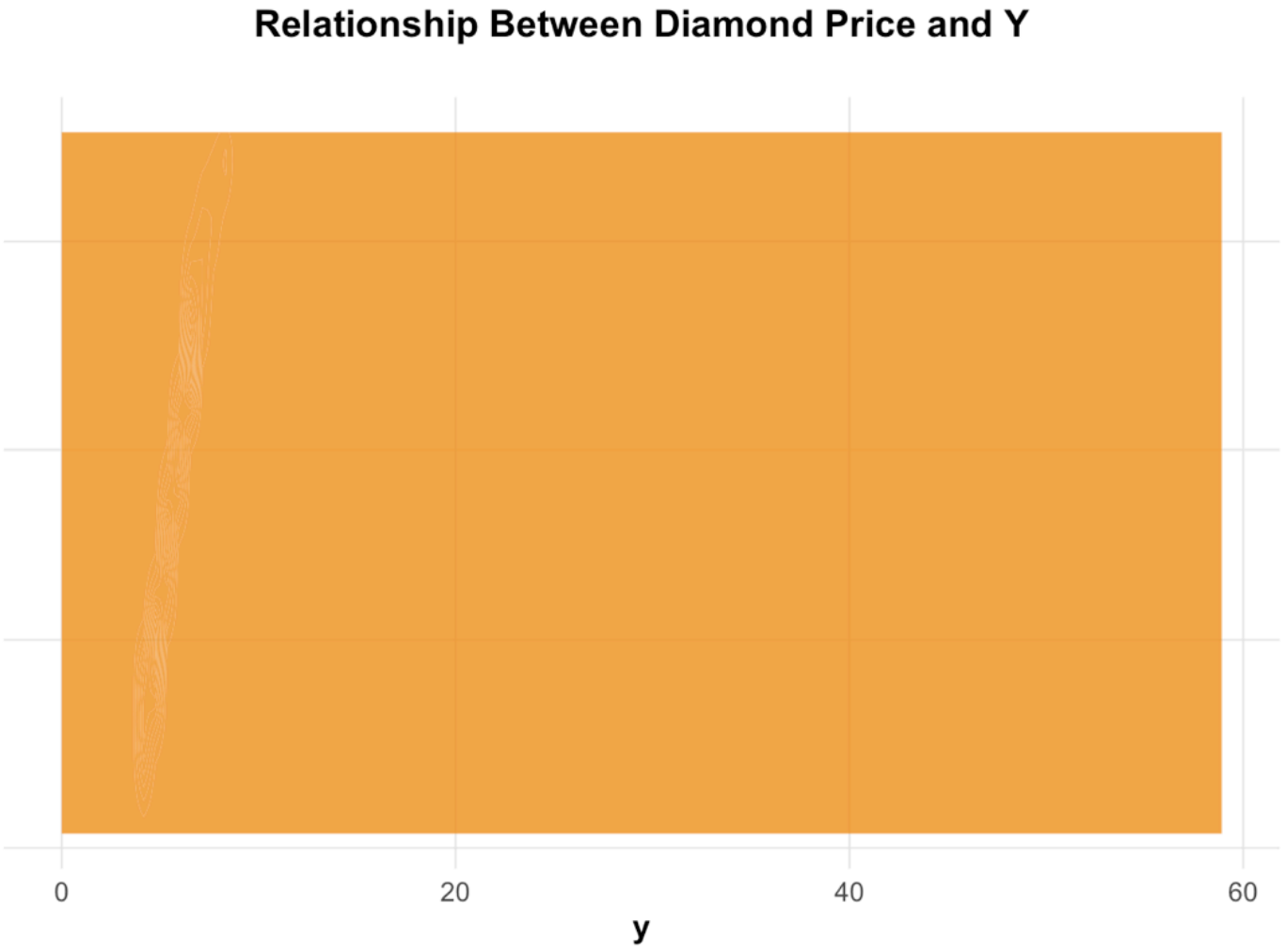

```
ggplot(diamonds, aes(x = y, y = price)) +
  geom_point(color = diamond_gold,
             alpha = 0.86) +
  geom_smooth(method = "lm") +
    labs(
    title = "Relationship Between Diamond Price and Y",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
`geom_smooth()` using formula = 'y ~ x'
```




```
ggplot(diamonds, aes(x = y, y = price)) +
  geom_density2d_filled(width = 0.3,
              fill = diamond_gold,
              alpha = 0.8,
              outlier.color = diamond_blue,
              outlier.alpha = 0.6) +
  scale_y_log10() +
    labs(
    title = "Relationship Between Diamond Price and Y",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

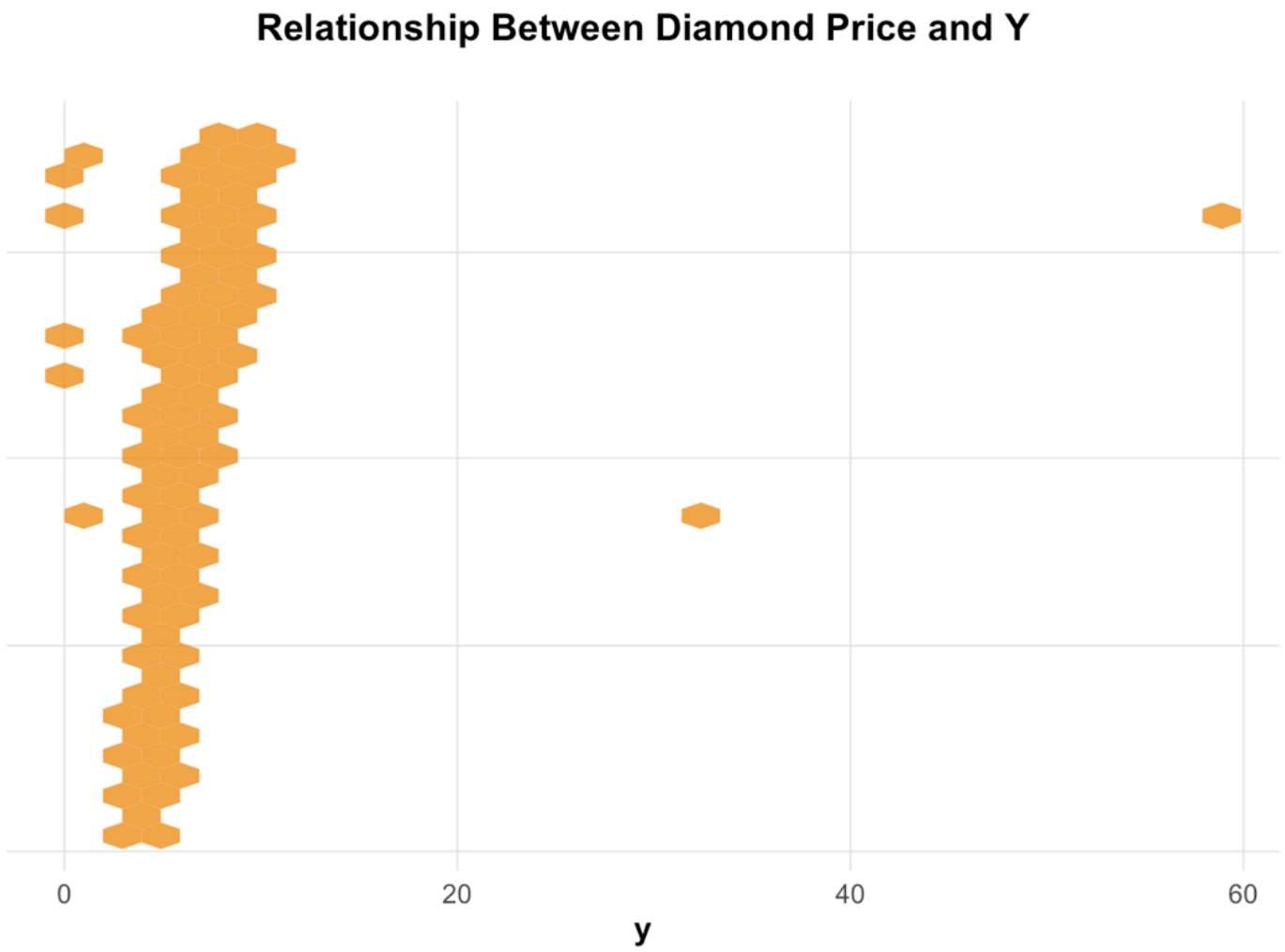
```
Warning in geom_density2d_filled(width = 0.3, fill = diamond_gold, alpha = 0.8,
: Ignoring unknown parameters: `width`, `outlier.colour`, and `outlier.alpha`
```




```
ggplot(diamonds, aes(x = y, y = price)) +
  geom_hex(width = 0.3,
               fill = diamond_gold,
               alpha = 0.8,
               outlier.color = diamond_blue,
               outlier.alpha = 0.6) +
  scale_y_log10() +
    labs(
    title = "Relationship Between Diamond Price and Y",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
Warning in geom_hex(width = 0.3, fill = diamond_gold, alpha = 0.8,
outlier.color = diamond_blue, : Ignoring unknown parameters: `width`,
```

`outlier.colour`, and `outlier.alpha`

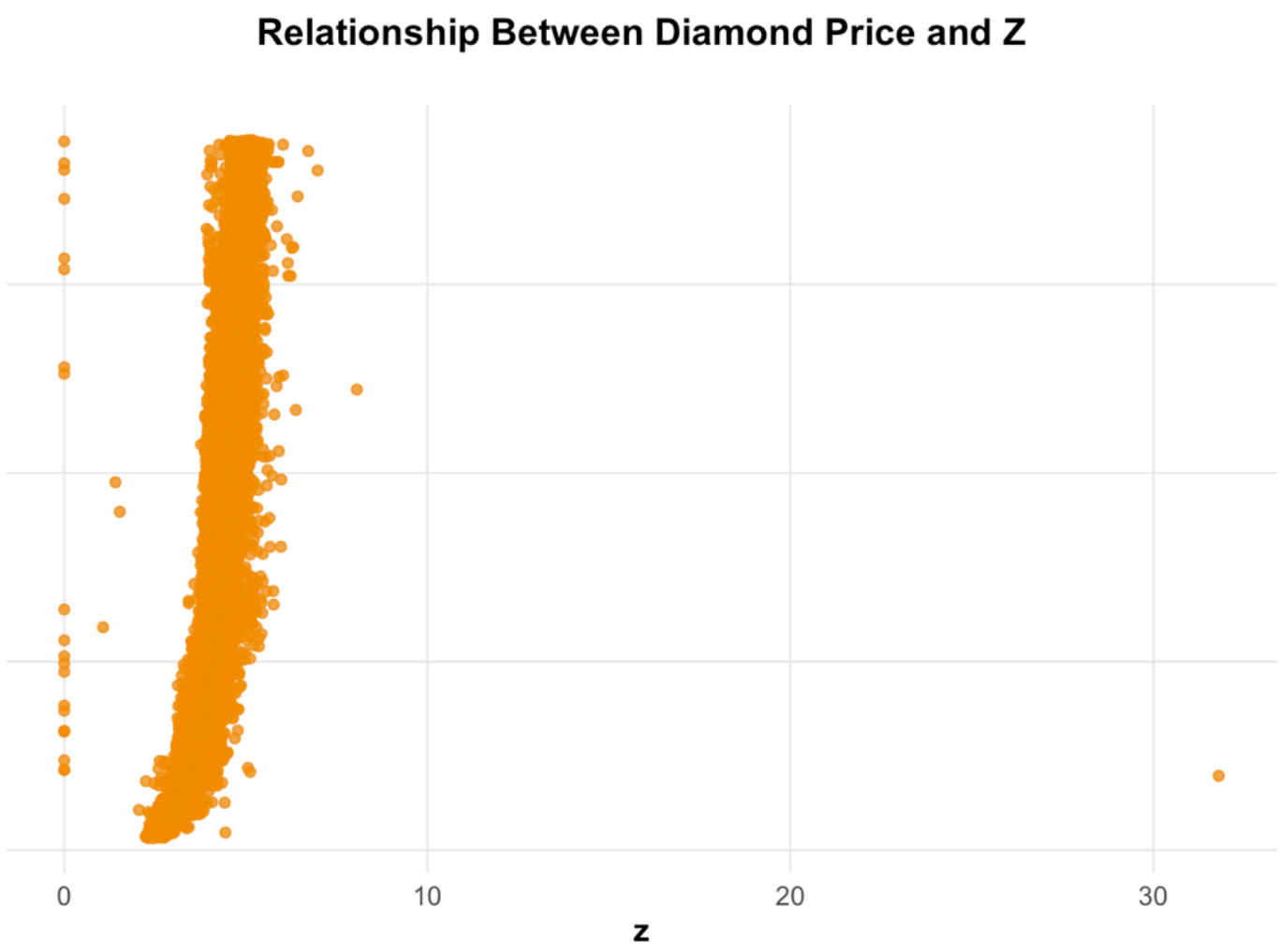



```
install.packages("hexbin")
```

```
The downloaded binary packages are in
    /var/folders/jx/19wzy4r974zgwcx8kgshk9400000gn/T//RtmpQyxrpZ/downloaded_packages
```

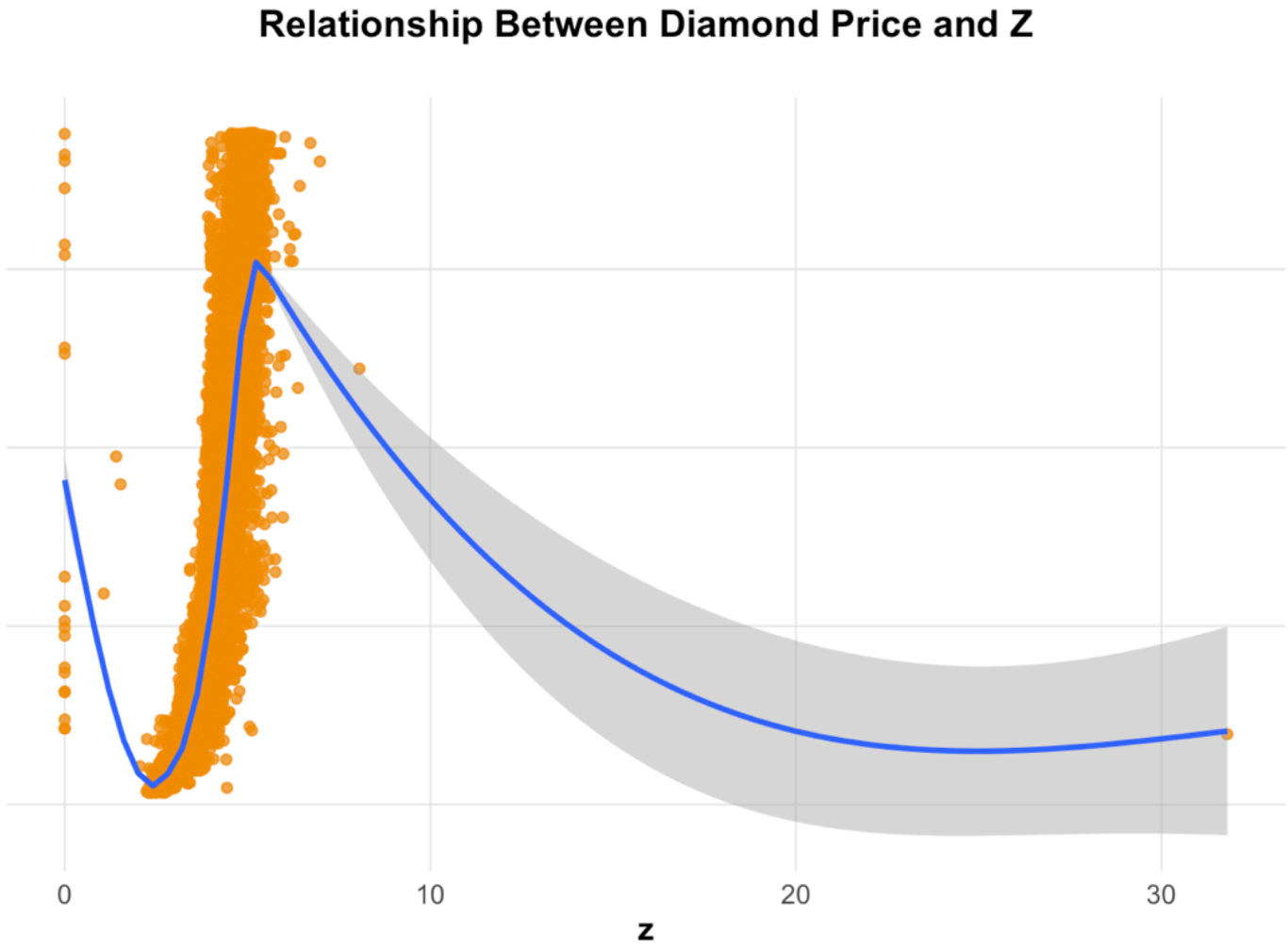
```
ggplot(diamonds, aes(x = z, y = price)) +
  geom_point(color = diamond_gold,
             alpha = 0.8) +
    labs(
    title = "Relationship Between Diamond Price and Z",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
```

```
        axis.ticks.y = element_blank())
```



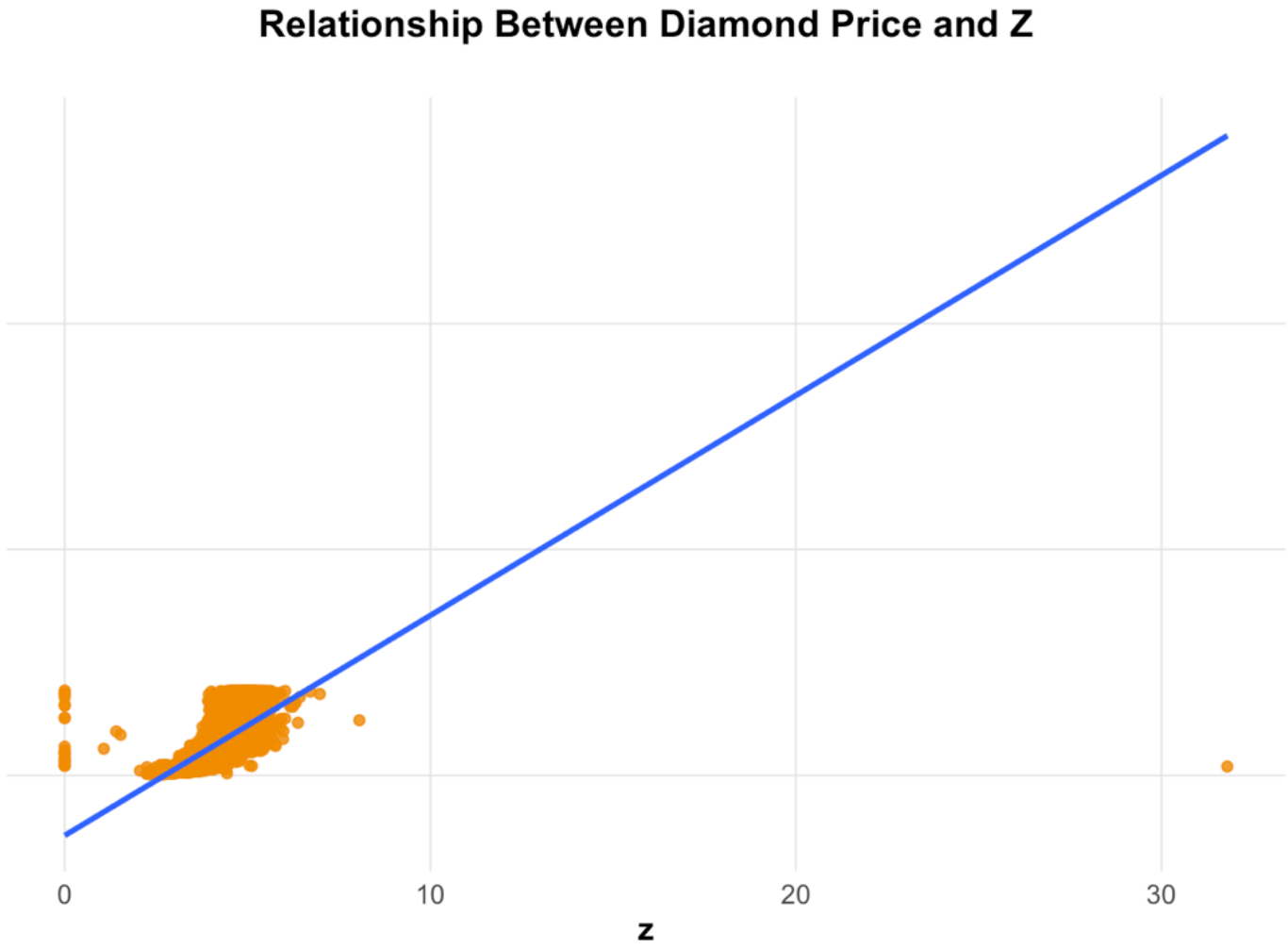

```
ggplot(diamonds, aes(x = z, y = price)) +
  geom_point(color = diamond_gold,
             alpha = 0.8) +
  geom_smooth() +
    labs(
    title = "Relationship Between Diamond Price and Z",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



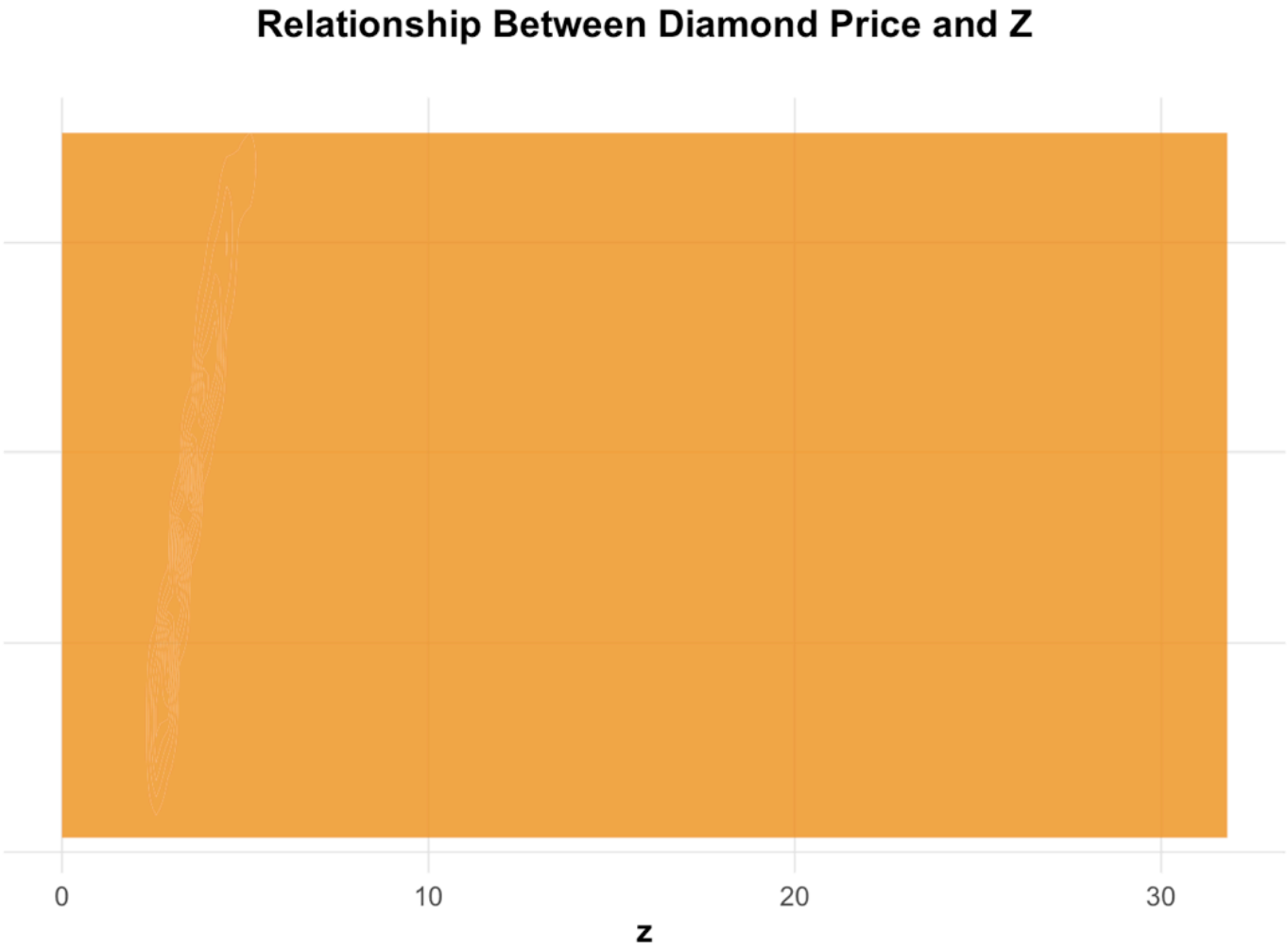

```
ggplot(diamonds, aes(x = z, y = price)) +
  geom_point(color = diamond_gold,
             alpha = 0.86) +
  geom_smooth(method = "lm") +
    labs(
    title = "Relationship Between Diamond Price and Z",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
`geom_smooth()` using formula = 'y ~ x'
```




```
ggplot(diamonds, aes(x = z, y = price)) +
  geom_density2d_filled(width = 0.3,
              fill = diamond_gold,
              alpha = 0.8,
              outlier.color = diamond_blue,
              outlier.alpha = 0.6) +
  scale_y_log10() +
    labs(
    title = "Relationship Between Diamond Price and Z",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

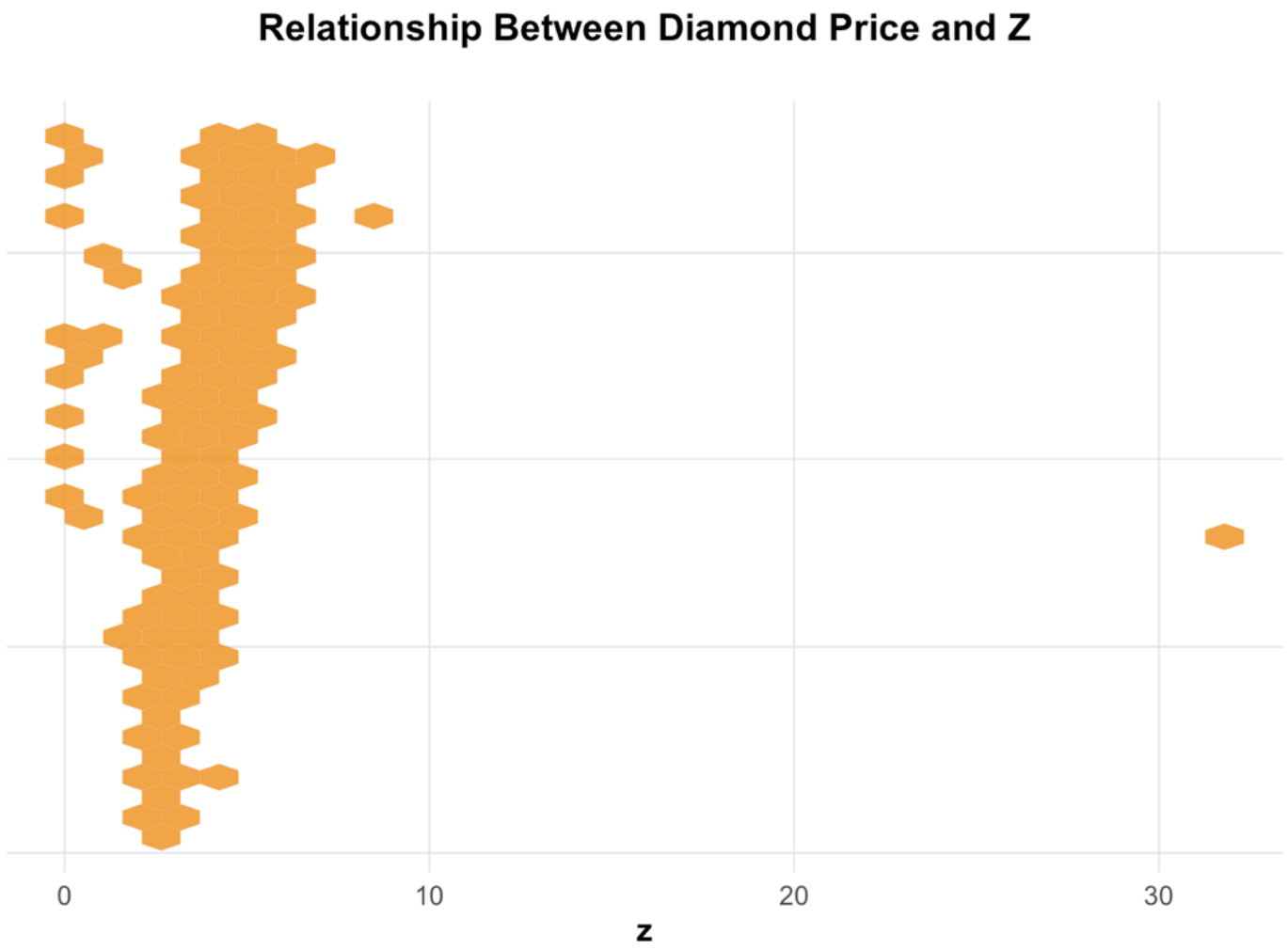
```
Warning in geom_density2d_filled(width = 0.3, fill = diamond_gold, alpha = 0.8,
: Ignoring unknown parameters: `width`, `outlier.colour`, and `outlier.alpha`
```




```
ggplot(diamonds, aes(x = z, y = price)) +
  geom_hex(width = 0.3,
               fill = diamond_gold,
               alpha = 0.8,
               outlier.color = diamond_blue,
               outlier.alpha = 0.6) +
  scale_y_log10() +
    labs(
    title = "Relationship Between Diamond Price and Z",
    y = ""
  ) +
  custom_theme +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

```
Warning in geom_hex(width = 0.3, fill = diamond_gold, alpha = 0.8,
outlier.color = diamond_blue, : Ignoring unknown parameters: `width`,
```

`outlier.colour`, and `outlier.alpha`



Data Clustering

Having explored both individual variables and the relationship between variables, the next key step is to see what features can be identified in the data. One of these features is clustering, i.e. if we can tease out certain clusters within the data, which in our case could help us identify different groups of diamonds.

K-means clustering on scaled features

First, we conduct a k-means clustering analysis, which (as the same suggests) breaks the data into k clusters based on average distance from k number of points. Using an elbow chart (which helps us identify the optimal number of clusters), we can see that our data clusters quite well into three clusters based on price: high-end, mid-end, and low-end respectively. This implies that diamond

sales largely bucket into three categories.

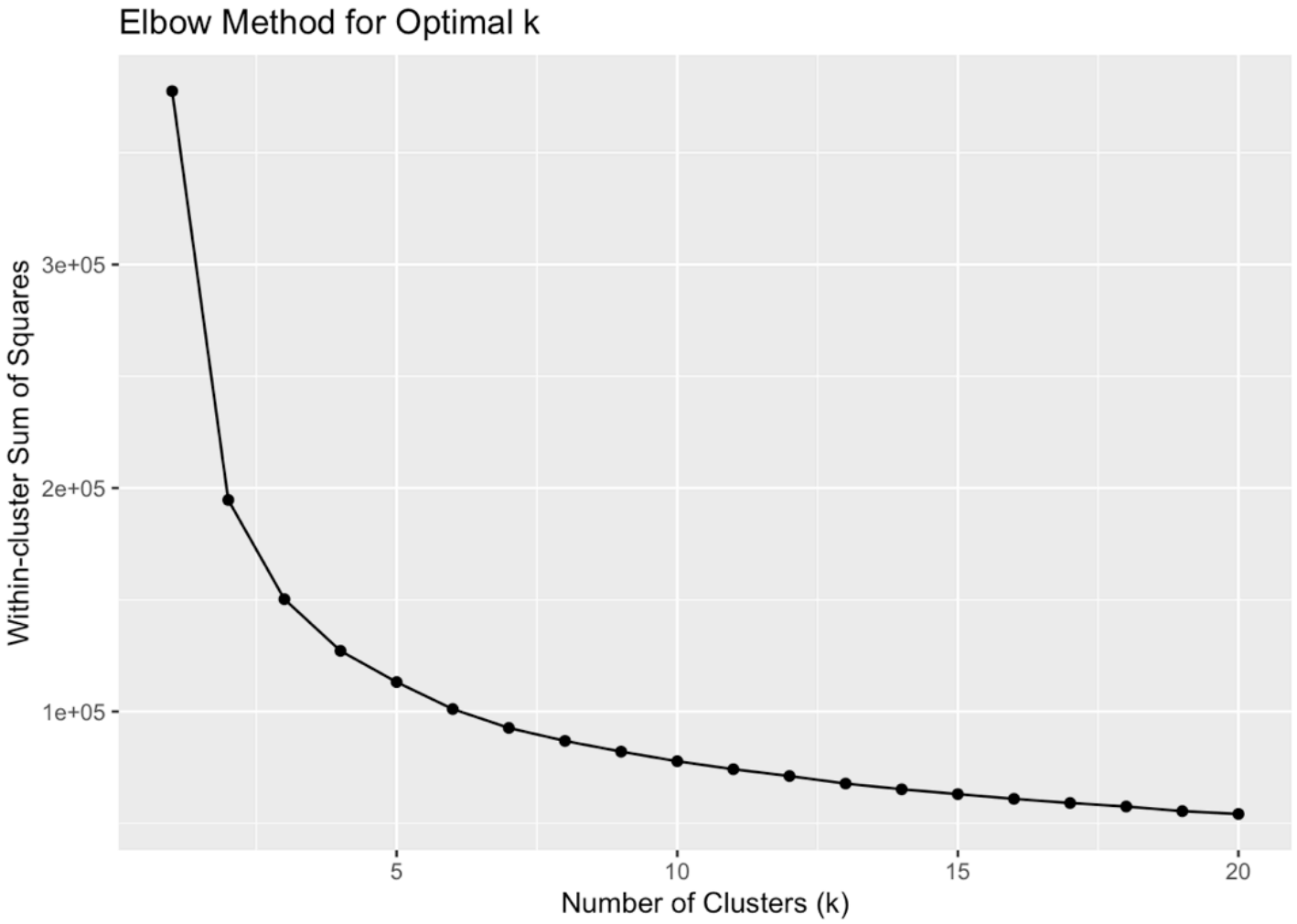
```
diamonds_numeric <- diamonds |>
  select(carat, depth, table, price, x, y, z) |>
  scale()

set.seed(1)
k3 <- kmeans(diamonds_numeric, centers = 3, nstart = 25)

diamonds_clustered <- diamonds |>
  mutate(cluster = as.factor(k3$cluster))

# Function to compute total within-cluster sum of squares
wss <- function(k) {
  kmeans(diamonds_numeric, k, nstart = 25)$tot.withinss
}

# Compute WSS for k = 1 to 10
k_values <- 1:20
wss_values <- map_dbl(k_values, wss)
```

```
Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)
Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)
Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)
Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)

Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations

Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)

Warning: did not converge in 10 iterations

Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)
Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)
Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)

Warning: did not converge in 10 iterations

Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)

Warning: did not converge in 10 iterations

Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)
```

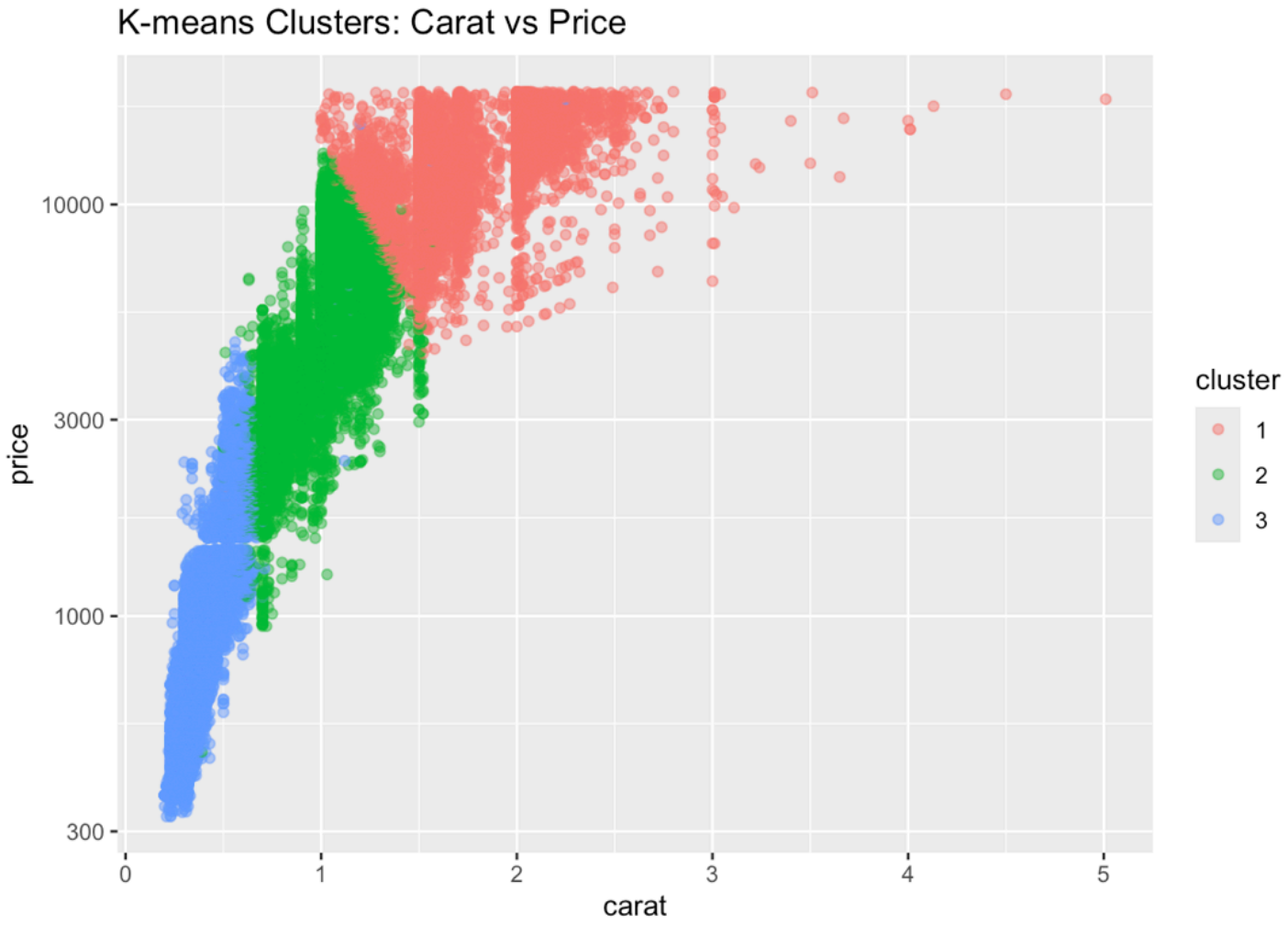
```
Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)

Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations

Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)

Warning: did not converge in 10 iterations

Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)
Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)

Warning: did not converge in 10 iterations

Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)
Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)
Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)
Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)

Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations

Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)
Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)

Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations

Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)

Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations

Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)

Warning: did not converge in 10 iterations

Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)
```

```
Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations

Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)
Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)

Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations

Warning: Quick-TRANSfer stage steps exceeded maximum (= 2697000)

Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations
Warning: did not converge in 10 iterations
```

```r
# Plot elbow curve
data.frame(k = k_values, wss = wss_values) |>
  ggplot(aes(x = k, y = wss)) +
  geom_line() +
  geom_point() +
  labs(title = "Elbow Method for Optimal k",
       x = "Number of Clusters (k)",
       y = "Within-cluster Sum of Squares")
```



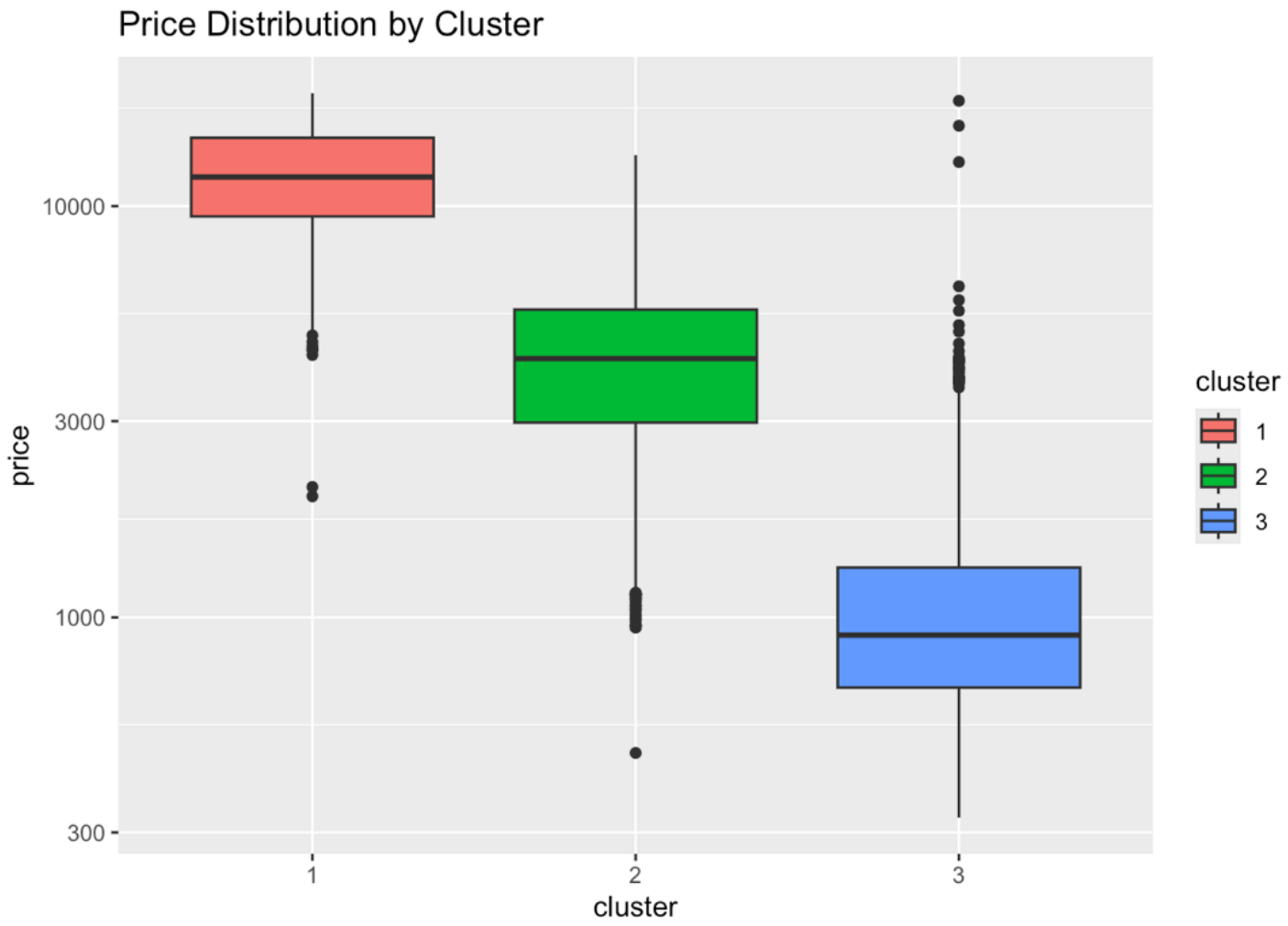

```
# Scatter plot colored by cluster
ggplot(diamonds_clustered, aes(x = carat, y = price, color = cluster)) +
  geom_point(alpha = 0.5) +
  scale_y_log10() +
  labs(title = "K-means Clusters: Carat vs Price")
```




```
# Summary statistics by cluster
diamonds_clustered %>%
  group_by(cluster) %>%
  summarise(
    count = n(),
    avg_carat = mean(carat),
    avg_price = mean(price),
    avg_depth = mean(depth),
    avg_table = mean(table)
  )
```

```
# A tibble: 3 × 6
  cluster count avg_carat avg_price avg_depth avg_table
  <fct>   <int>     <dbl>     <dbl>     <dbl>     <dbl>
1 1        7208     1.70     12076.      61.7      58.0
2 2       21819     0.951     4524.      61.8      57.8
3 3       24913     0.402     1059.      61.7      57.0
```

```
# Box plots by cluster
ggplot(diamonds_clustered, aes(x = cluster, y = price, fill = cluster)) +
  geom_boxplot() +
  scale_y_log10() +
  labs(title = "Price Distribution by Cluster")
```



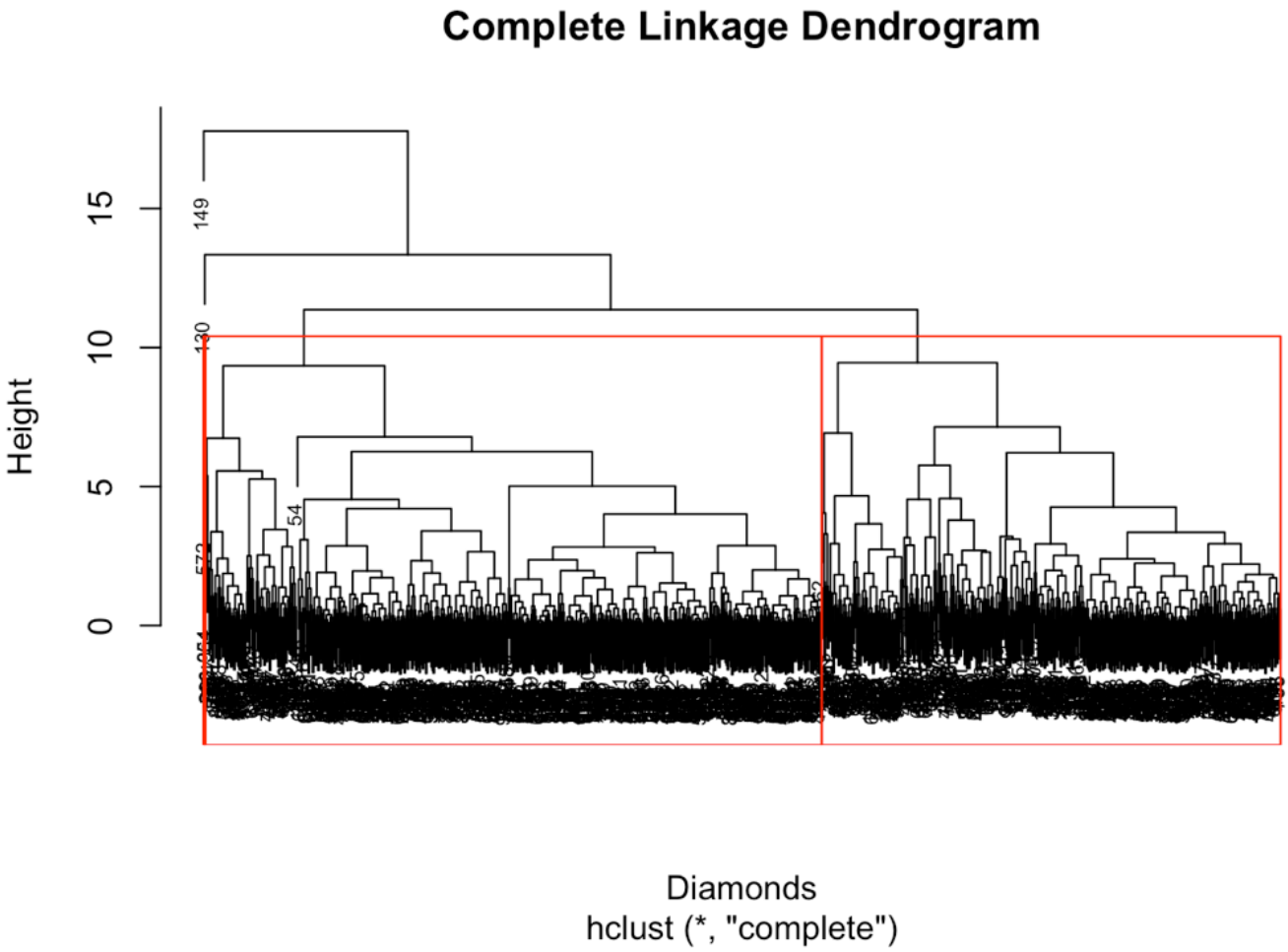
Hierarchical clustering with dendrograms

In addition to the more classic k-means clustering, we can do a hierachical clustering using dendrograms. This is a form of unsupervised machine learning that identifies distance between pairs (in my case, I used multiple methods of computing distance) and repeatedly merges the closest pairs until we have a fixed number of remaining pairs (which become our clusters.) Here, again, the optimal number of clusters is three, with similar results. I chose to use what is called Ward's Method, which tries the minimize in-cluster variance, as I wanted to create outlier-light clusters.

```
install.packages("ggdendro")
```

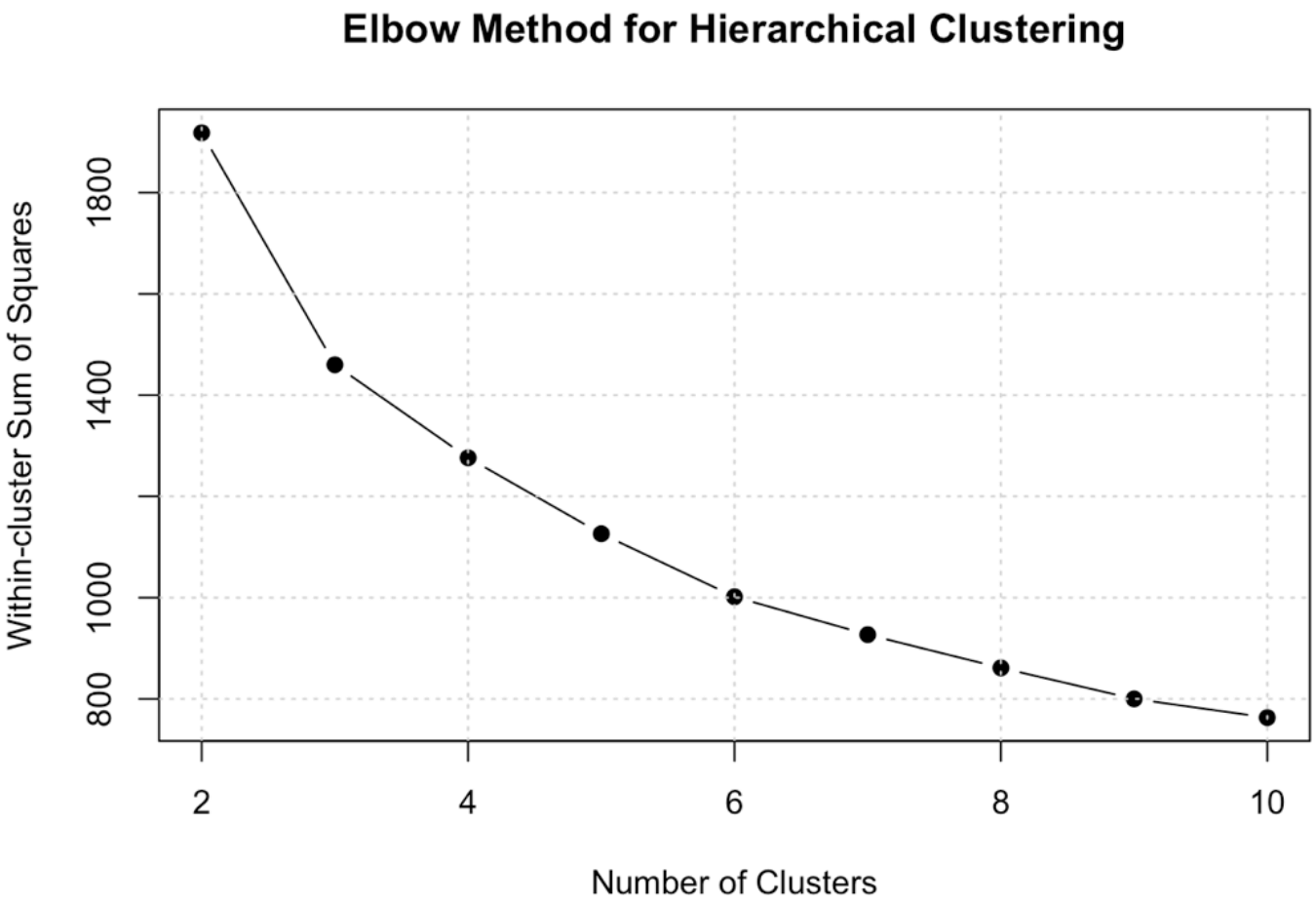
```
The downloaded binary packages are in
    /var/folders/jx/19wzy4r974zgwcx8kgshk9400000gn/T//RtmpQyxrpZ/downloaded_packages
```

```
set.seed(1)
diamonds_sample <- diamonds[sample(nrow(diamonds), 1000), ]  # Sampling to maximize sp

# Selecting and scaling numerical variables
diamonds_scaled <- diamonds_sample |>
  select(carat, price, depth, table, x, y, z) |>
  scale()

# Computing distance matrix
dist_matrix <- dist(diamonds_scaled, method = "euclidean")

# Performing hierarchical clustering
hc_complete <- hclust(dist_matrix, method = "complete")
hc_ward <- hclust(dist_matrix, method = "ward.D2")
hc_average <- hclust(dist_matrix, method = "average")

# Basic dendrogram
plot(hc_complete, main = "Complete Linkage Dendrogram",
     xlab = "Diamonds", ylab = "Height", cex = 0.6)

# Adding colored clusters (cut into 4 groups)
rect.hclust(hc_complete, k = 4, border = "red")
```

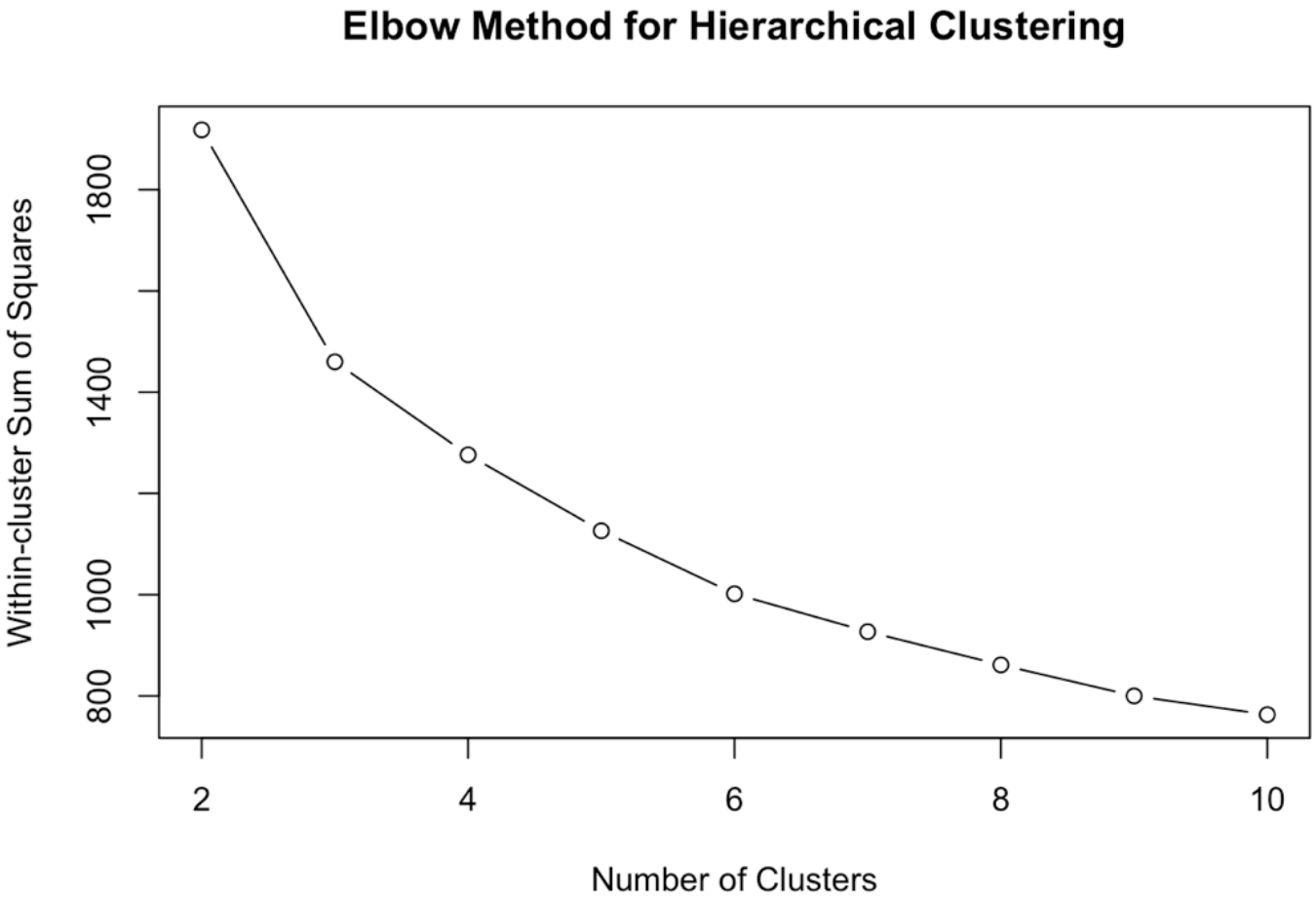



```r
# Cutting dendrogram into different numbers of clusters and evaluate
wss_hc <- function(k, hc_result, data) {
  clusters <- cutree(hc_result, k = k)
  sum(sapply(1:k, function(i) {
    cluster_data <- data[clusters == i, , drop = FALSE]
    if(nrow(cluster_data) > 1) {
      sum(dist(cluster_data)^2) / (2 * nrow(cluster_data))
    } else {
      0
    }
  }))
}

k_values <- 2:10  # Start from 2 for hierarchical
wss_values <- sapply(k_values, wss_hc, hc_ward, diamonds_scaled)

plot(k_values, wss_values, type = "b", pch = 19,
```

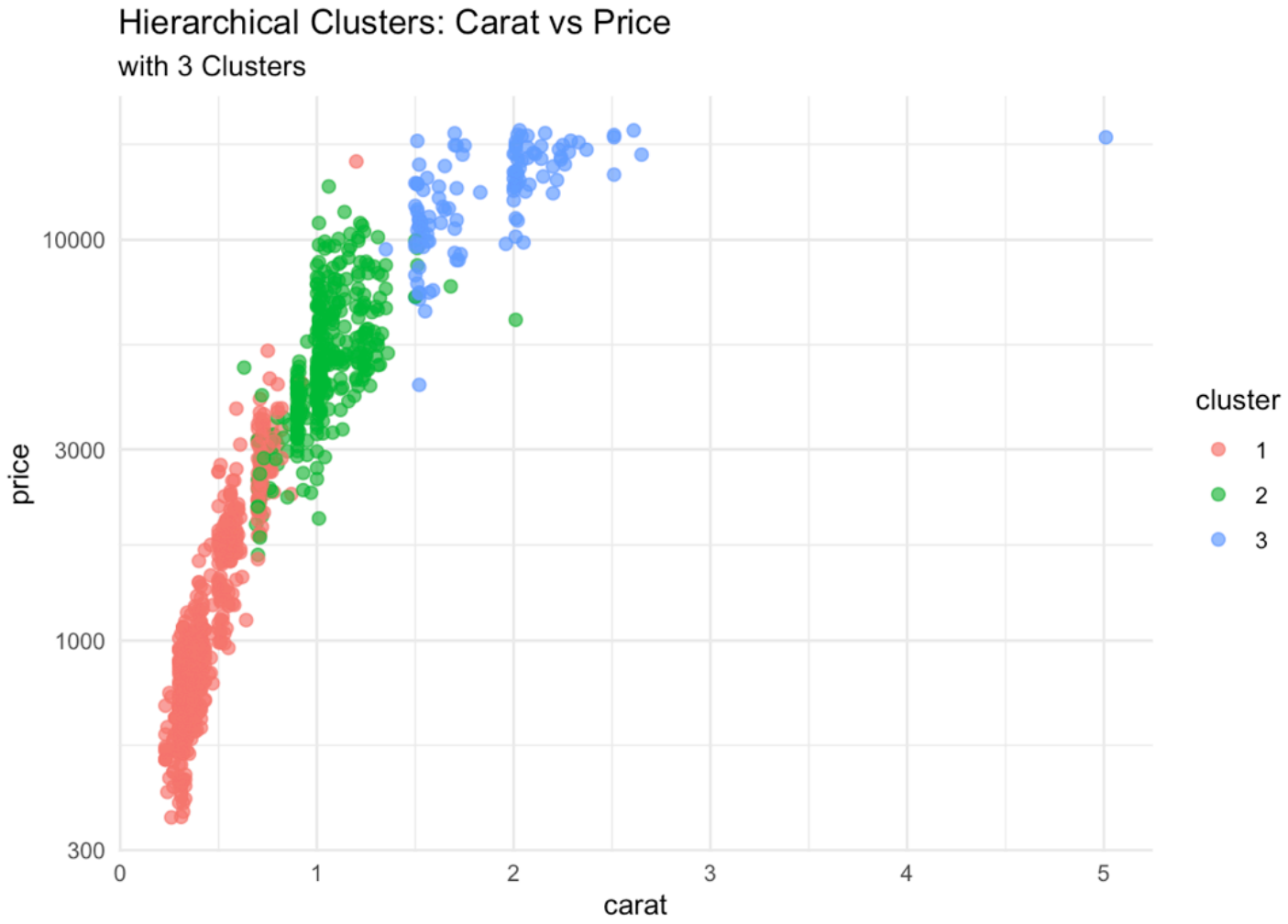
```
     xlab = "Number of Clusters",
     ylab = "Within-cluster Sum of Squares",
     main = "Elbow Method for Hierarchical Clustering")
grid()
```




```
# Testing different numbers of clusters
k_values <- 2:10
wss_values <- sapply(k_values, wss_hc, hc_ward, diamonds_scaled)

# Plotting the elbow chart
plot(k_values, wss_values, type = "b",
     xlab = "Number of Clusters", ylab = "Within-cluster Sum of Squares",
     main = "Elbow Method for Hierarchical Clustering")
```




```r
# Cutting dendrogram into the optimal 3 clusters
clusters <- cutree(hc_ward, k = 3)

# Adding cluster assignments into dataset
diamonds_hc <- diamonds_sample %>%
  mutate(cluster = as.factor(clusters))

# Visualizing clusters
ggplot(diamonds_hc, aes(x = carat, y = price, color = cluster)) +
  geom_point(alpha = 0.7, size = 2) +
  scale_y_log10() +
  labs(title = "Hierarchical Clusters: Carat vs Price",
       subtitle = "with 3 Clusters") +
  theme_minimal()
```




```
# Cluster summary statistics
diamonds_hc |>
  group_by(cluster) |>
  summarise(
    count = n(),
    avg_carat = mean(carat),
    avg_price = mean(price),
    avg_depth = mean(depth),
    price_range = paste(min(price), "-", max(price))
  )
```

```
# A tibble: 3 × 6
  cluster count avg_carat avg_price avg_depth price_range
  <fct>   <int>     <dbl>     <dbl>     <dbl> <chr>
1 1         529     0.460     1386.      61.8 362 - 15686
2 2         349     1.04      5278.      61.9 1636 - 13588
3 3         122     1.86     13202.      61.9 4345 - 18757
```

Price Prediction

Having explored various variables, their relationships to eachother, and hidden groups in our data, it is time to find ways to accurately predict price.

Traditional Regression

Using R's excellent regsubsets function, we are able to identify which combination of our variables has the most predictive power. This turns out to be carat, cut, color, and clarity which provide an astounding 91.9% R Squared. This model as a Root Mean Square Error of 1132.74.

```
bestvarset <- regsubsets(price ~ ., data=diamonds, nvmax = 9)
summary_results <- summary(bestvarset)

max_adj_r2_index <- which.max(summary_results$adjr2)
print(names(coef(bestvarset, max_adj_r2_index))[-1])
```

```
[1] "carat"     "cut.L"     "color.L"   "color.Q"   "clarity.L" "clarity.Q"
[7] "clarity.C" "clarity^4" "x"
```

```
#Optimal model
lm(price ~ carat + cut + color + clarity + x, data = diamonds)
```

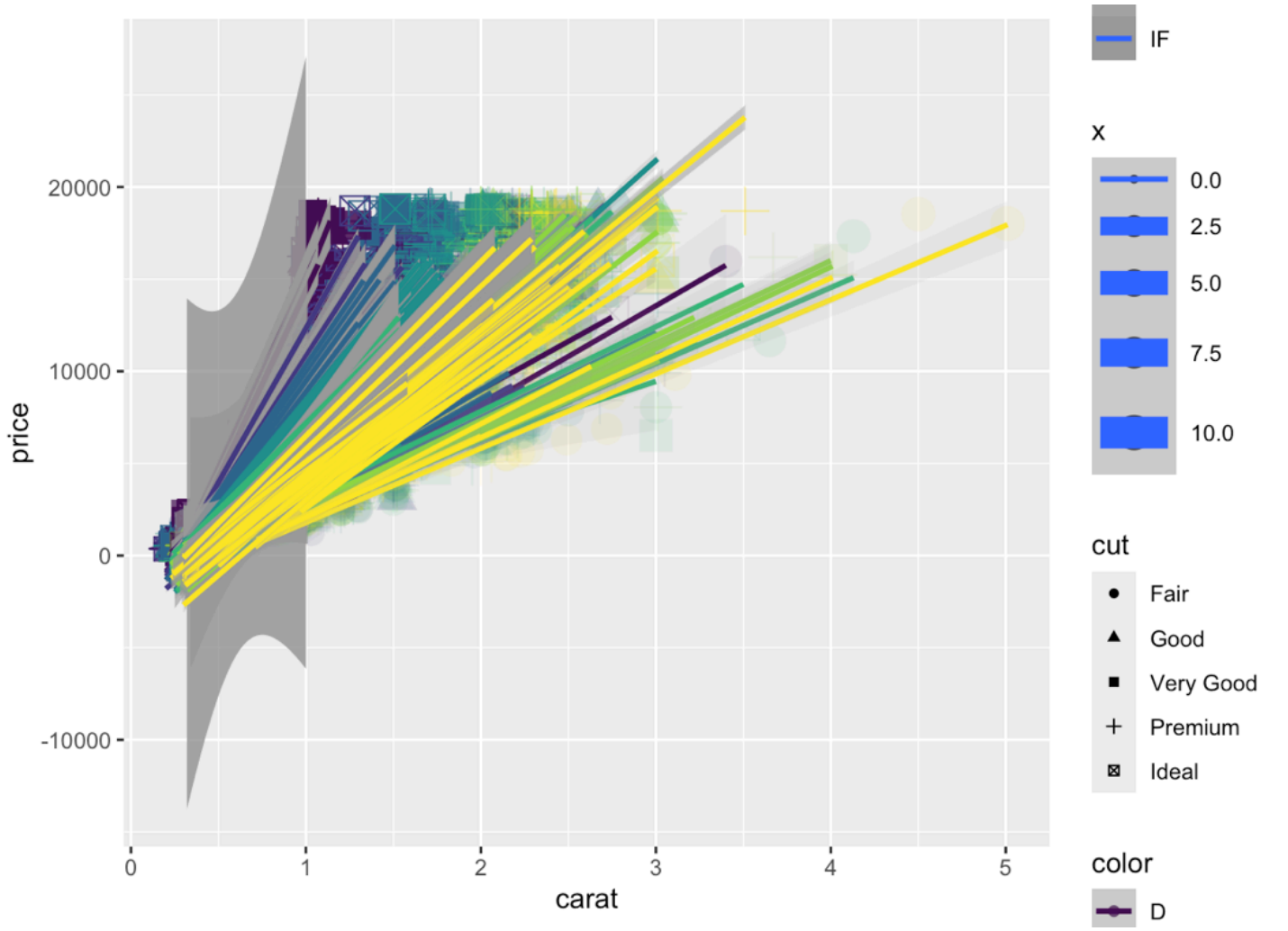
```

Call:
lm(formula = price ~ carat + cut + color + clarity + x, data = diamonds)

Coefficients:
(Intercept)        carat        cut.L        cut.Q        cut.C        cut^4
    -19.662    11088.570      748.416     -350.750      175.058      -16.094
    color.L      color.Q      color.C      color^4      color^5      color^6
  -1960.204     -670.927     -163.081       37.177      -96.777      -47.718
  clarity.L    clarity.Q    clarity.C    clarity^4    clarity^5    clarity^6
   4133.676    -1923.477      988.176     -368.223      240.542        2.879
  clarity^7            x
     92.468     -959.485
```

```
summary(lm(price ~ carat + cut + color + clarity + x, data = diamonds))$r.squared
```

```
[1] 0.9193801
```

```r
# Saving my model to so I can visualize its effectiveness
final_model <- lm(price ~ carat + cut + color + clarity + x, data = diamonds)

# Generating predictions to make my graph
diamonds$predicted_price <- predict(final_model)

# Creating a sc
ggplot(diamonds, aes(x = carat, y = price, color = color, shape = cut, alpha = clarity
  geom_point() +
  geom_smooth(method = "lm")
```

```
Warning: Using shapes for an ordinal variable is not advised

`geom_smooth()` using formula = 'y ~ x'

Warning in qt((1 - level)/2, df): NaNs produced

Warning in qt((1 - level)/2, df): NaNs produced

Warning: The following aesthetics were dropped during statistical transformation:
size.
ℹ This can happen when ggplot fails to infer the correct grouping structure in
  the data.
ℹ Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?

Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
-Inf
Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
-Inf
```




```
predictions <- predict(final_model, diamonds)

#Calculating RMSE
rmse <- sqrt(mean((diamonds$price - predictions)^2))
print(paste("RMSE:", round(rmse, 2)))
```

```
[1] "RMSE: 1132.74"
```

Random Forest

Next, we use a random forest regression, which creates a number of decision trees and combines their predictions in an ensemble. This is a marked improvement over our simple linear regression, raising R squared to 98.2 and lowering RMSE to 530.53

```
install.packages("randomForest")
```

```
The downloaded binary packages are in
    /var/folders/jx/19wzy4r974zgwcx8kgshk9400000gn/T//RtmpQyxrpZ/downloaded_packages
```

```
library(randomForest)
```

```
randomForest 4.7-1.2

Type rfNews() to see new features/changes/bug fixes.


Attaching package: 'randomForest'

The following object is masked from 'package:dplyr':

    combine

The following object is masked from 'package:ggplot2':

    margin
```

```
set.seed(1)
train_index <- createDataPartition(diamonds$price, p = 0.7, list = FALSE)
train_data <- diamonds[train_index, ]
test_data  <- diamonds[-train_index, ]

set.seed(1)
rf_model <- randomForest(
  price ~ .,
  data = train_data,
  ntree = 500,
  mtry = 4,
  importance = TRUE
)

print(rf_model)
```

```
Call:
 randomForest(formula = price ~ ., data = train_data, ntree = 500,      mtry = 4,
importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4
```

```
          Mean of squared residuals: 282992
                    % Var explained: 98.21
```

```
predictions <- predict(rf_model, newdata = test_data)

rmse <- sqrt(mean((predictions - test_data$price)^2))
r2   <- cor(predictions, test_data$price)^2

cat("RMSE:", rmse, "\n")
```

```
RMSE: 530.5341
```

```
cat("R-squared:", r2, "\n")
```

```
R-squared: 0.9826007
```

XGBOOST

Having tried a traditional linear regression and a Random Forest, next I try XBGOOST, another ensemble method based on decision trees. But where Random Forest simultaneously builds and averages a bunch of trees, XGBOOST adds them iteratively in hopes of getting the best model possible. Here, it performs highly similarly to the Random Forest, albeit with a slightly lower RMSE (525.6942

```
install.packages("xgboost")
```

```
The downloaded binary packages are in
    /var/folders/jx/19wzy4r974zgwcx8kgshk9400000gn/T//RtmpQyxrpZ/downloaded_packages
```

```
install.packages("Matrix")
```

```
The downloaded binary packages are in
    /var/folders/jx/19wzy4r974zgwcx8kgshk9400000gn/T//RtmpQyxrpZ/downloaded_packages
```

```
install.packages("caret")
```

```
The downloaded binary packages are in
    /var/folders/jx/19wzy4r974zgwcx8kgshk9400000gn/T//RtmpQyxrpZ/downloaded_packages
```

```
library(xgboost)
```

```
Attaching package: 'xgboost'

The following object is masked from 'package:dplyr':

    slice
```

```
library(Matrix)
library(caret)

set.seed(1)
train_index <- createDataPartition(diamonds$price, p = 0.7, list = FALSE)
train_data  <- diamonds[train_index, ]
test_data   <- diamonds[-train_index, ]

train_matrix <- model.matrix(price ~ . - 1, data = train_data)
test_matrix  <- model.matrix(price ~ . - 1, data = test_data)

train_label <- train_data$price
test_label  <- test_data$price

dtrain <- xgb.DMatrix(data = train_matrix, label = train_label)
dtest  <- xgb.DMatrix(data = test_matrix, label = test_label)

params <- list(
  objective = "reg:squarederror",
  eval_metric = "rmse",
  eta = 0.05,
  max_depth = 6,
  subsample = 0.8,
  colsample_bytree = 0.8
)

set.seed(1)
xgb_model <- xgb.train(
  params = params,
  data = dtrain,
  nrounds = 5000,            # maximum number of boosting rounds
  watchlist = list(train = dtrain, test = dtest),
```

```
  early_stopping_rounds = 50,
  print_every_n = 50
)
```

```
[1] train-rmse:5315.569721  test-rmse:5353.467599
Multiple eval metrics are present. Will use test_rmse for early stopping.
Will train until test_rmse hasn't improved in 50 rounds.

[51]    train-rmse:676.060249   test-rmse:716.564552
[101]   train-rmse:479.570676   test-rmse:541.459743
[151]   train-rmse:448.795086   test-rmse:532.236078
[201]   train-rmse:424.527963   test-rmse:528.591381
[251]   train-rmse:405.245147   test-rmse:527.247273
[301]   train-rmse:389.752671   test-rmse:526.644558
[351]   train-rmse:376.818646   test-rmse:526.098443
[401]   train-rmse:364.854396   test-rmse:526.094250
Stopping. Best iteration:
[369]   train-rmse:372.436524   test-rmse:525.694238
```

```
preds <- predict(xgb_model, dtest)

# R-squared
r2 <- cor(preds, test_label)^2
cat("Test R-squared:", r2, "\n")
```

```
Test R-squared: 0.9828932
```

```
# RMSE
rmse <- sqrt(mean((preds - test_label)^2))
cat("Test RMSE:", rmse, "\n")
```

```
Test RMSE: 525.6942
```

Conclusion

After exploring R's diamonds dataset, it becomes clear that the ten variables contained within it are highly variable, consisting of everything from nearly normal variables to multipolar to nearly uniform. It also becomes apparent that the observations can be neatly divided into about three categories. Finally, with rather simple regression models and the limited data provided, we can construct models capable of capturing 98% percent of variation in price and predicting prices with an average error of about 500 dollars on average.