STA3115_homework2

2021122006 Jaehun Shon

2024-10-08

Problem 1: Reshaping data and assessing spread-location

The long-format reshaped data is shown below:

```r
cytof_longer <- cytof %>%
  pivot_longer(.,cols = colnames(cytof))
colnames(cytof_longer) <- c("protein_identity",
                            "protein_amount")

cytof_longer %>%
  head(10) %>%
  kable(col.names = c("Protein Identity", "Protein Amount"),
        format = "markdown")
```

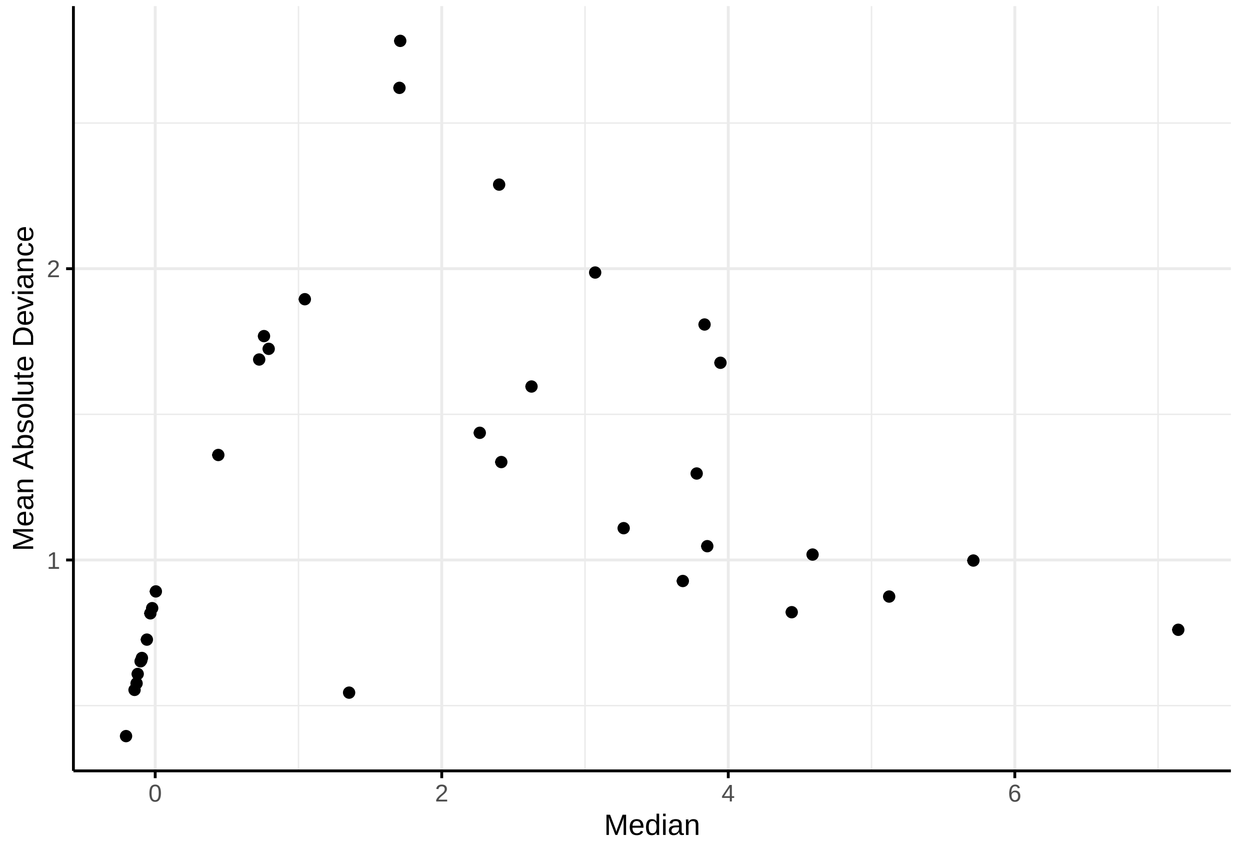
Protein Identity	Protein Amount
NKp30	0.1875955
KIR3DL1	3.6156932
NKp44	-0.5605694
KIR2DL1	-0.2936654
GranzymeB	2.4778929
CXCR6	-0.1447005
CD161	-0.3152872
KIR2DS4	1.9449705
NKp46	4.0818316
NKG2D	2.6200784

```r
mm <- function(data) {return(c(median(data), mad(data)))}

MM <- sapply(colnames(cytof), function(identity){
 tmp <- cytof[[identity]] %>% mm(.)
}) %>% t %>% as_tibble()

MM.c <- cbind(colnames(cytof), MM)
colnames(MM.c) <- c("Protein_identity", "Median", "Median Absolute Deviance")

MM.c %>%
  head(10) %>%
  kable(col.names = c("Protein Identity", "Median", "Median Absolute Deviance"),
        format = "markdown")
```

Protein Identity	Median	Median Absolute Deviance
NKp30	3.7795649	1.2968866
KIR3DL1	-0.0212163	0.8345779
NKp44	0.7592514	1.7686227
KIR2DL1	1.7048951	2.6208342
GranzymeB	3.6828239	0.9278530
CXCR6	-0.0581425	0.7266041
CD161	0.7256933	1.6882296
KIR2DS4	1.7102810	2.7821422
NKp46	3.8535019	1.0473857
NKG2D	2.6265640	1.5955079

Median value of each protein amount is shown above. For showing the deviance, median absolute deviance is used.

```r
colnames(MM) <- c("median","mad")
plot2 <-ggplot(MM, aes(x = median, y = mad)) +
  geom_point() +
  labs(x = "Median", y = "Mean Absolute Deviance") +
  theme_minimal() +
  theme(axis.line = element_line(linewidth = 0.5, colour = "black")) +
  theme(axis.ticks = element_line(linewidth = 0.5))


plot2
```



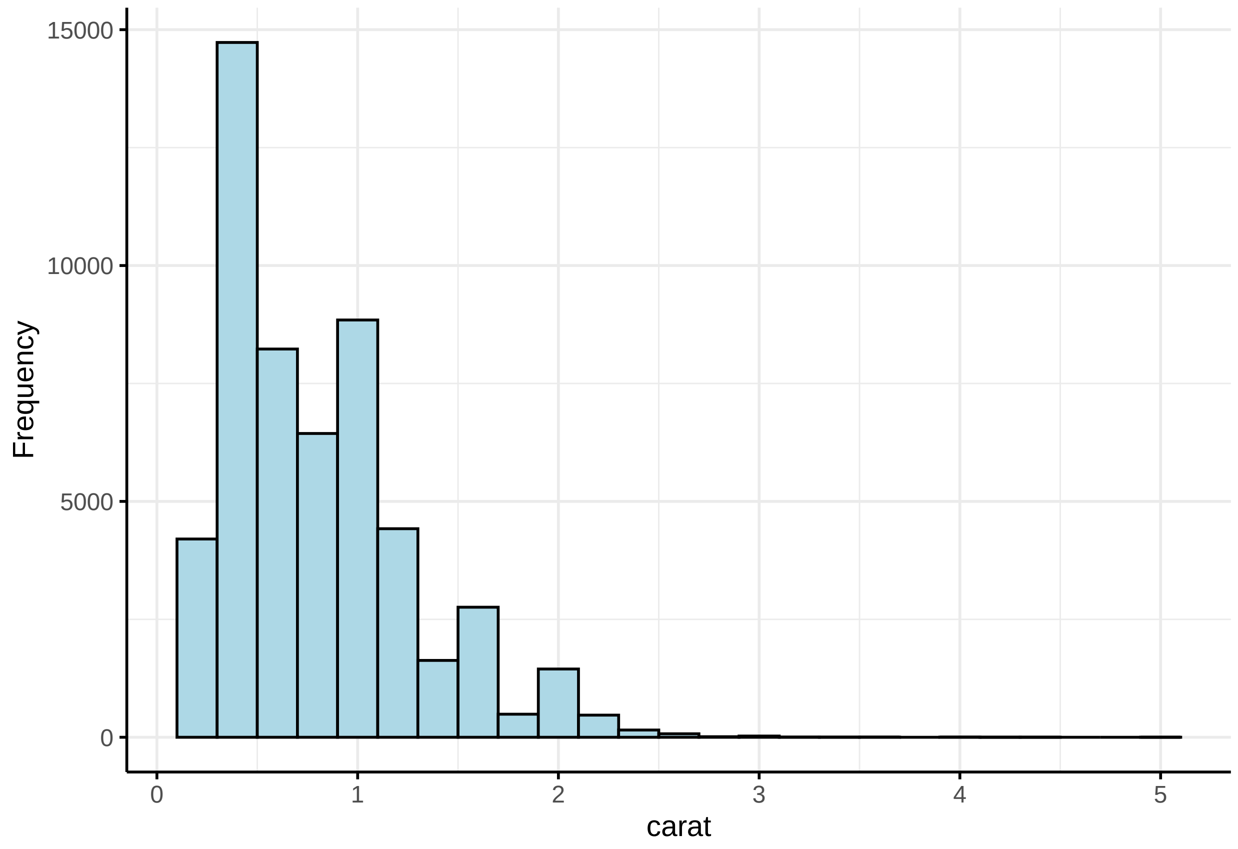
The plot displays the relationship between the median absolute deviation (MAD) of protein amounts and their median values. We observe that for proteins with a median less than 2, the MAD increases substantially as the median value increases. However, for proteins with a median above 2, this trend reverses: as the median increases, the MAD decreases.

Additionally, we observe that the variability of the median absolute deviations (MAD) themselves differs based on the median protein amounts. Specifically, the spread of MAD values for proteins with a median above 2 is larger than for those with a median below 2.

Problem 2: Creating histograms with modifications - ggplot2 review

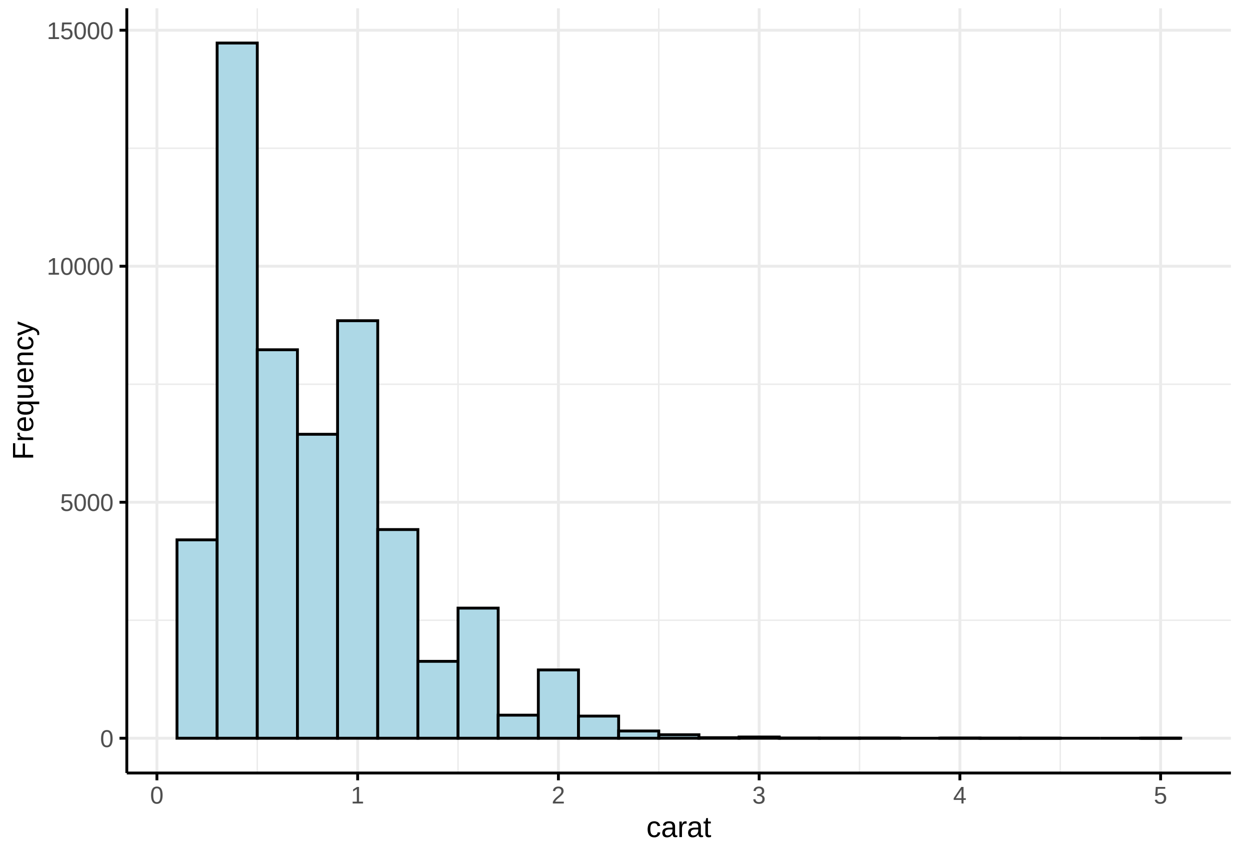
Histogram 1

```r
hist.1 <- ggplot(diamonds, aes(x = carat)) +
  geom_histogram(binwidth = 0.2, fill = 'lightblue', color = 'black') +
  labs(x = 'carat', y = 'Frequency') +
  theme_minimal() +
  theme(axis.line = element_line(linewidth = 0.5,
                                 colour = "black")) +
  theme(axis.ticks = element_line(linewidth = 0.5))

hist.1
```



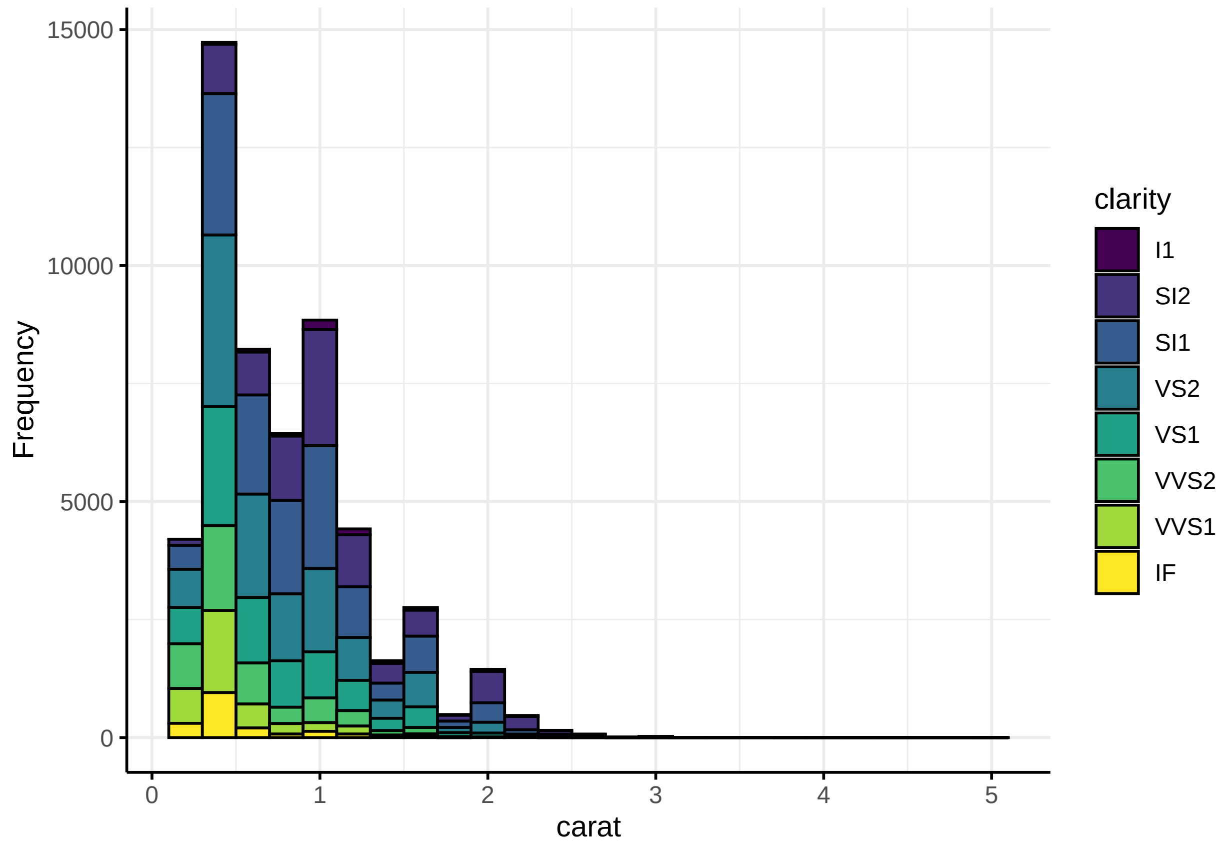
Histogram 2

```r
hist.2 <- ggplot(diamonds, aes(x = carat)) +
  geom_bar(stat = 'bin', binwidth = 0.2, fill = 'lightblue', color = 'black') +
  labs(x = 'carat', y = 'Frequency') +
  theme_minimal() +
  theme(axis.line = element_line(linewidth = 0.5,
                                 colour = "black")) +
  theme(axis.ticks = element_line(linewidth = 0.5))

hist.2
```



The histogram 2 is created using `geom_bar()`, which results in same shape with Histogram 2

```r
ggplot(diamonds, aes(x = carat, fill = clarity)) +
  geom_histogram(binwidth = 0.2, color = 'black') +
  labs(x = 'carat', y = 'Frequency') +
  theme_minimal() +
  theme(axis.line = element_line(linewidth = 0.5,
                                 colour = "black")) +
  theme(axis.ticks = element_line(linewidth = 0.5))
```



The histogram above additionally shows the proportion of identities of clarity of each bar, by using the code `aes(x = carat, fill = clarity)` and default position option `position = "stack"`.

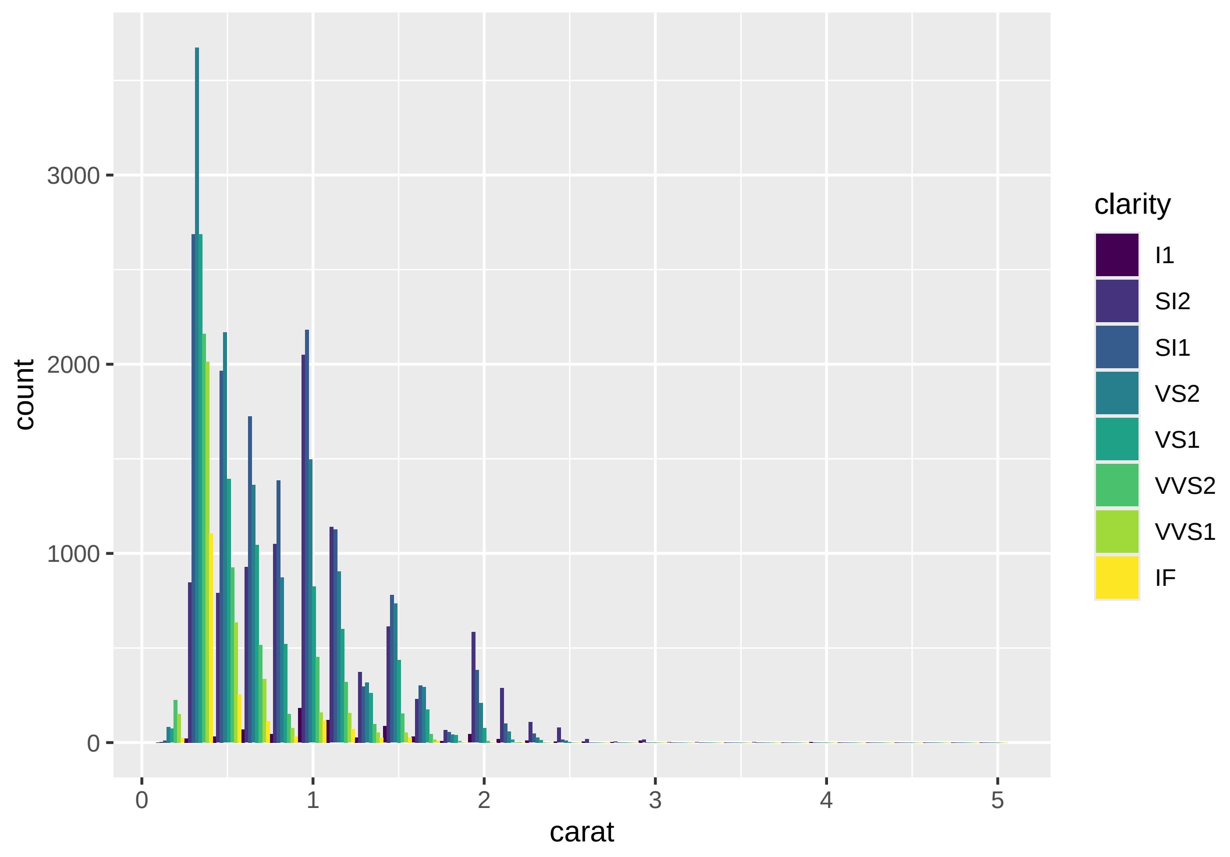
I will show another position adjustment option, which are shown below:

Option in geom_histogram

Dodge option

By applying `position = "dodge"` option, we can compare the frequency of each clarity level, but there are quite a few level of clarity, so visibility is low.

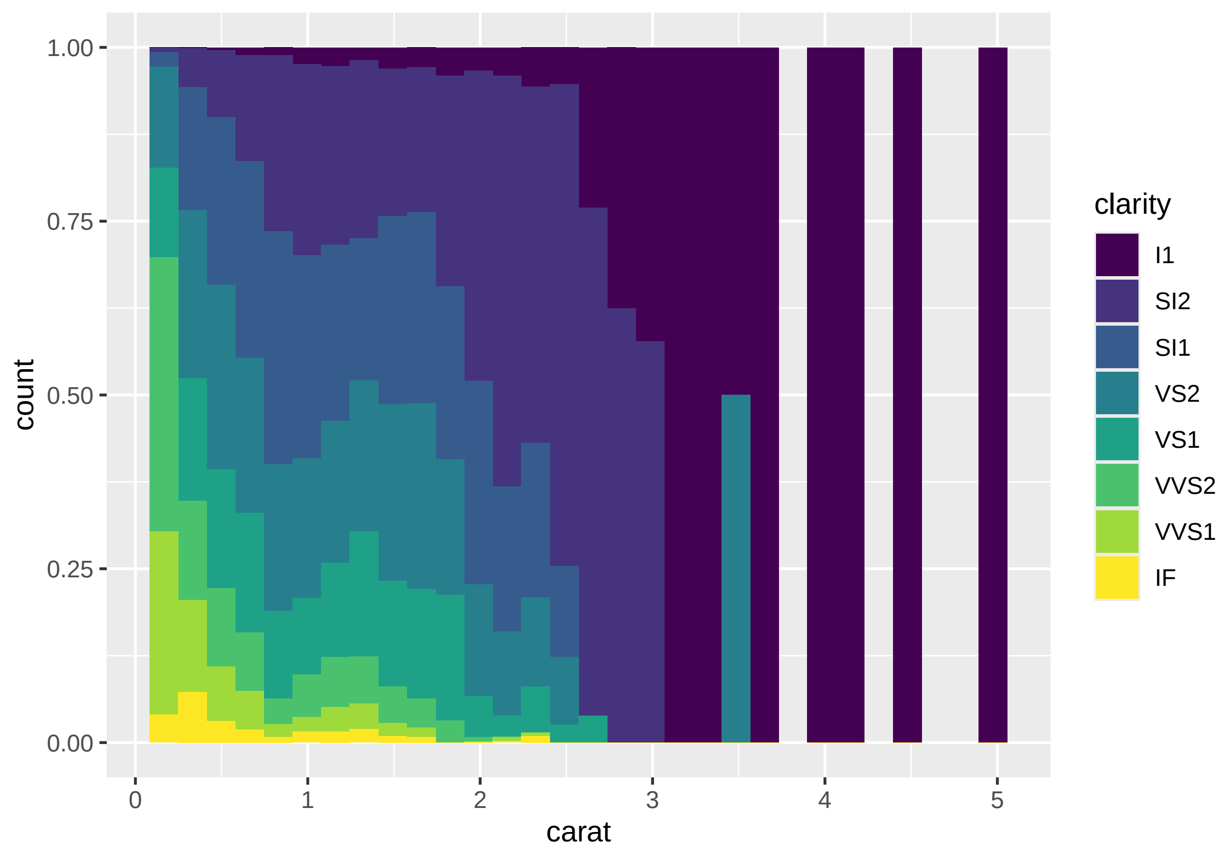
```
ggplot(diamonds, aes(x = carat, fill = clarity)) +
  geom_histogram(position = "dodge")
```



Fill option

We can compare the relative proportion of each clarity levels for each intervals by `position = "fill"` option.

```
ggplot(diamonds, aes(x = carat, fill = clarity)) +
  geom_histogram(position = "fill")
```

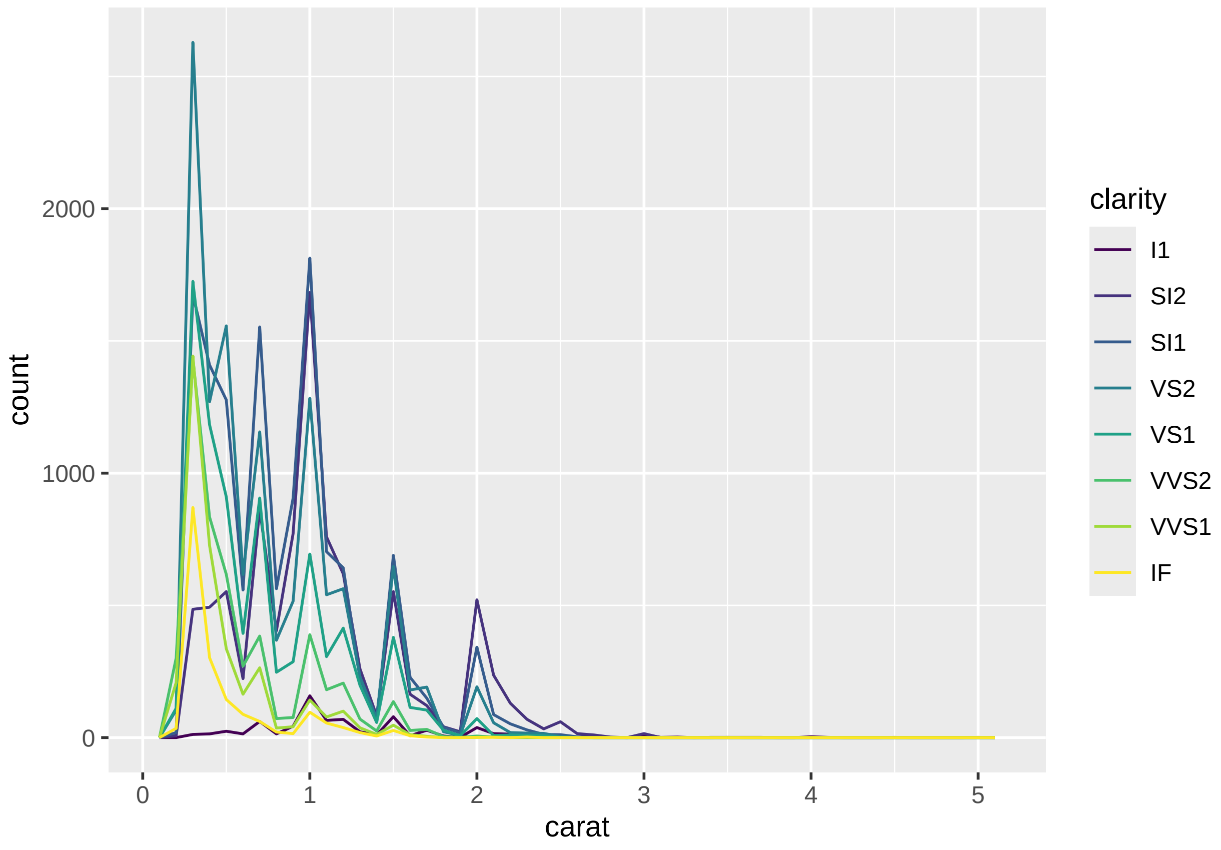


Other geometries similar to histogram

Frequency Polygen

Frequency polygen is similar to histogram, but it use line rather than bar. Visibility is also low for same reason with `position = "dodge"` option.

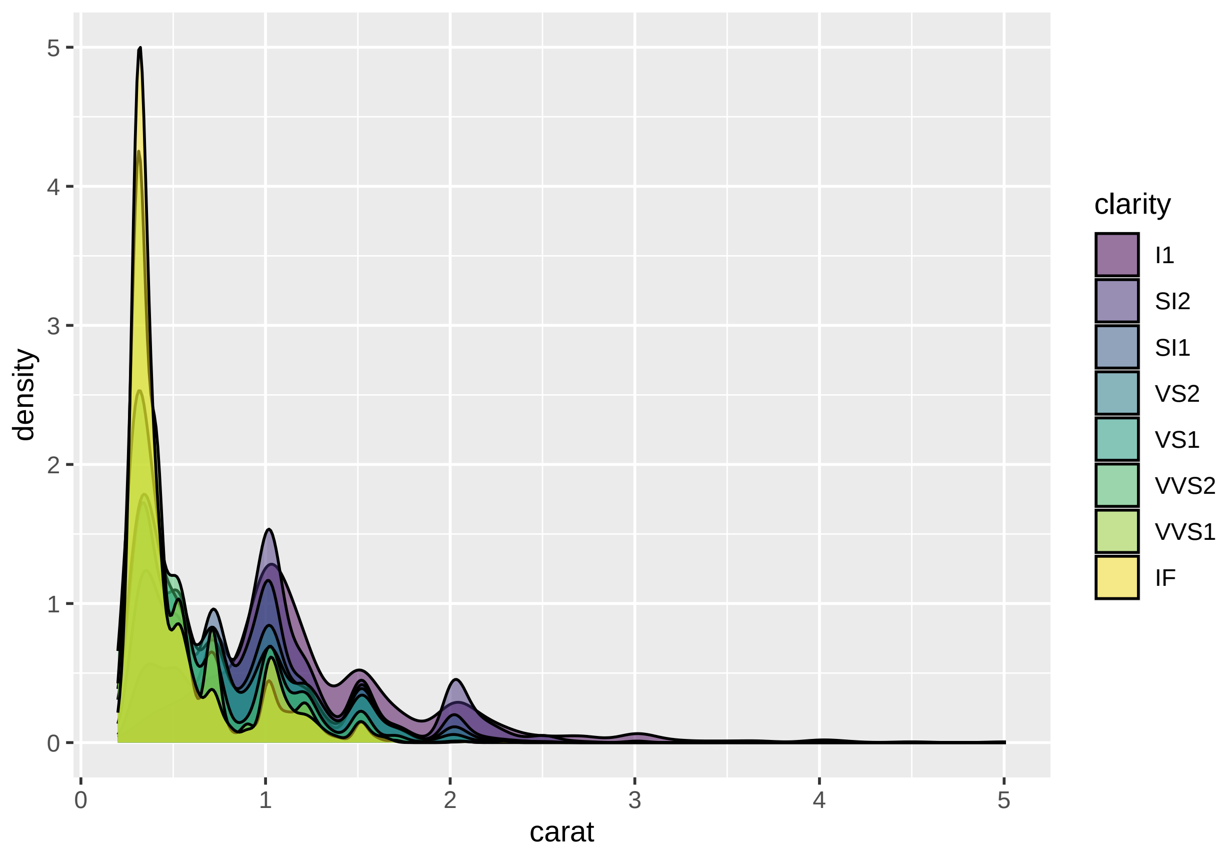
```r
ggplot(diamonds, aes(x = carat, color = clarity)) +
  geom_freqpoly(binwidth = 0.1)
```



Density

It shows smoother graph compared to `geom_freqpoly()`.

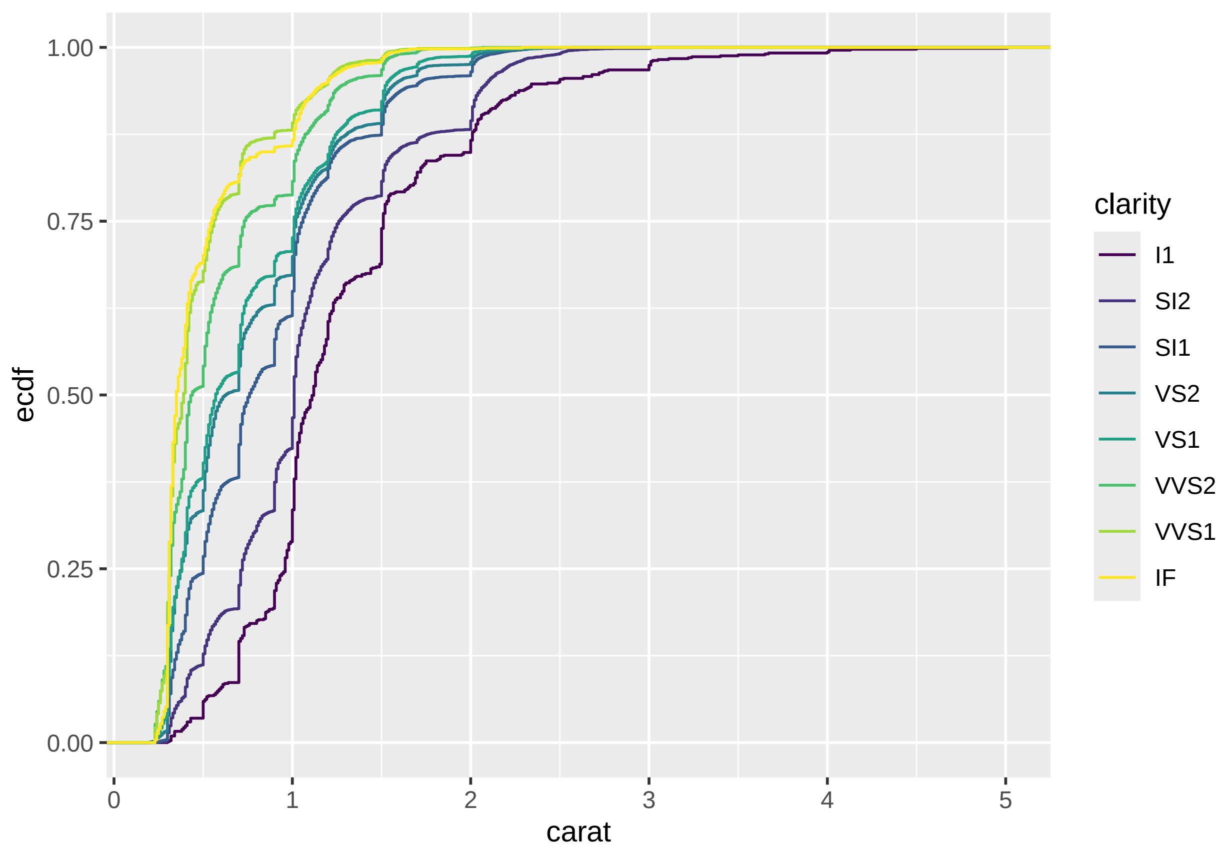
```r
ggplot(diamonds, aes(x = carat, fill = clarity)) +
  geom_density(alpha = 0.5)
```



Cumulative Density Function

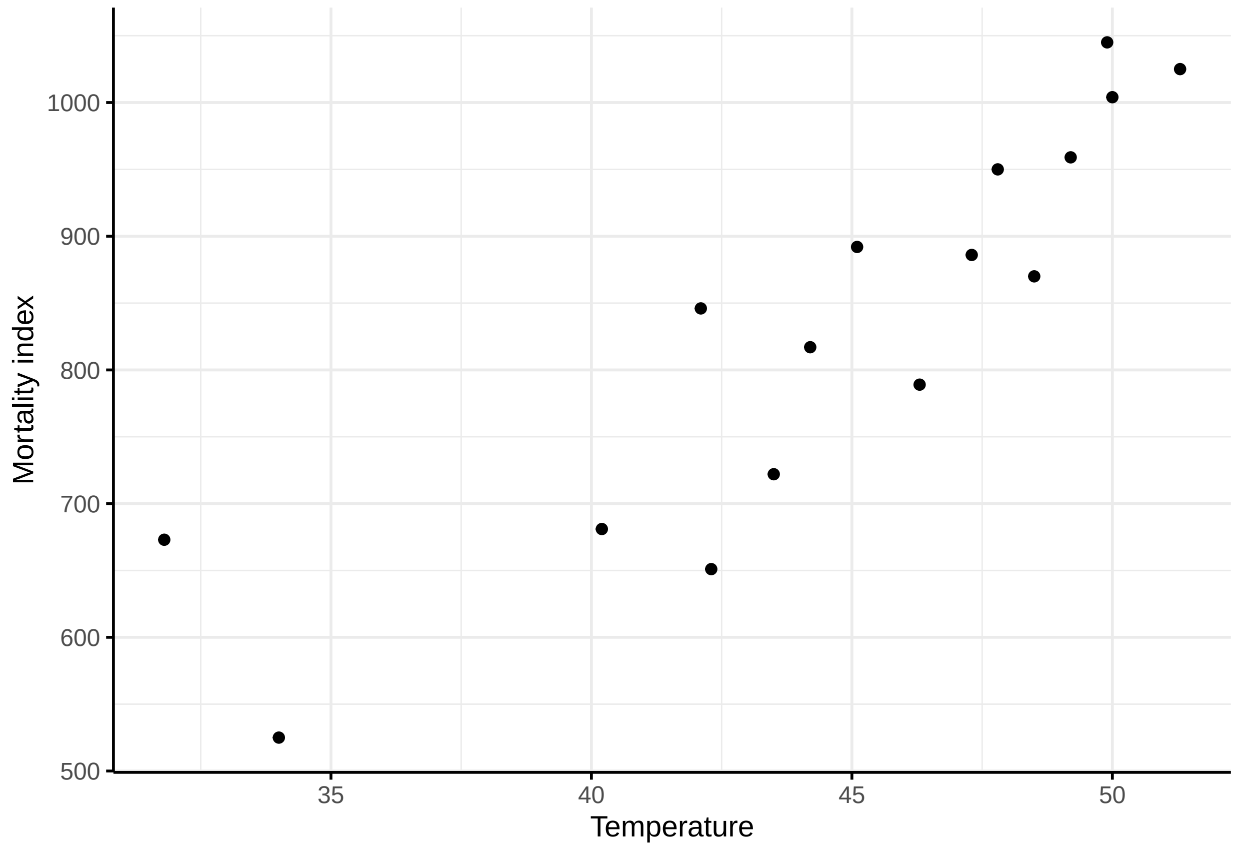
The code `stat_ecdf()` shows cdf of each clarity level.

```r
ggplot(diamonds, aes(x = carat, color = clarity)) +
  stat_ecdf()
```



Problem 3: Finding linear relationships in bivariate data

```
df <- mortality_by_latitude
ggplot(data = df, aes(x = temperature, y = mortality_index)) +
  geom_point() +
  labs(x = "Temperature", y = "Mortality index") +
  theme_minimal() +
  theme(axis.line = element_line(linewidth = 0.5,
                                  colour = "black")) +
  theme(axis.ticks = element_line(linewidth = 0.5))
```

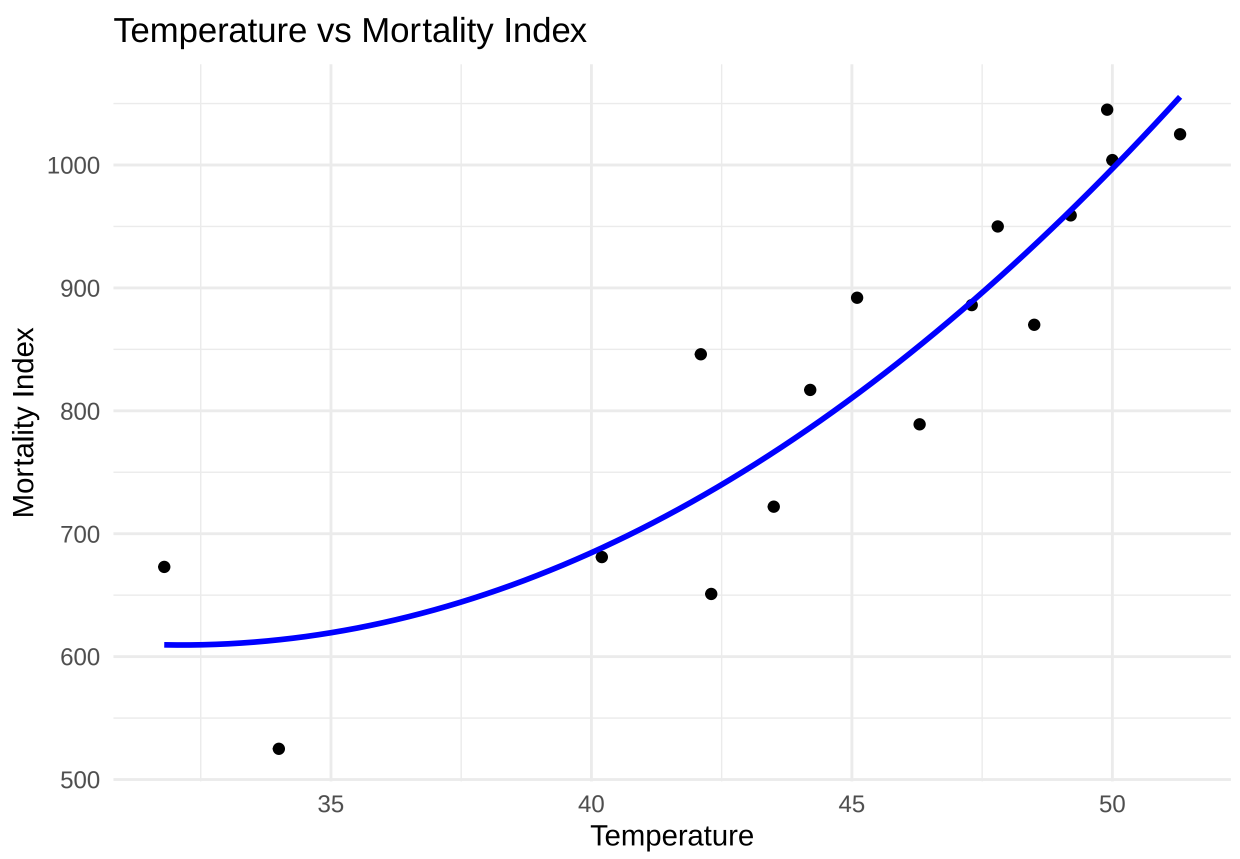


The plot is "hollow up". Here is my reason why the plot is hollow up.

First, at least, the plot cannot be hollow down, because checking the overall appearance of the plot, the increase rate of mortality rate by temperature increase is not decreasing.

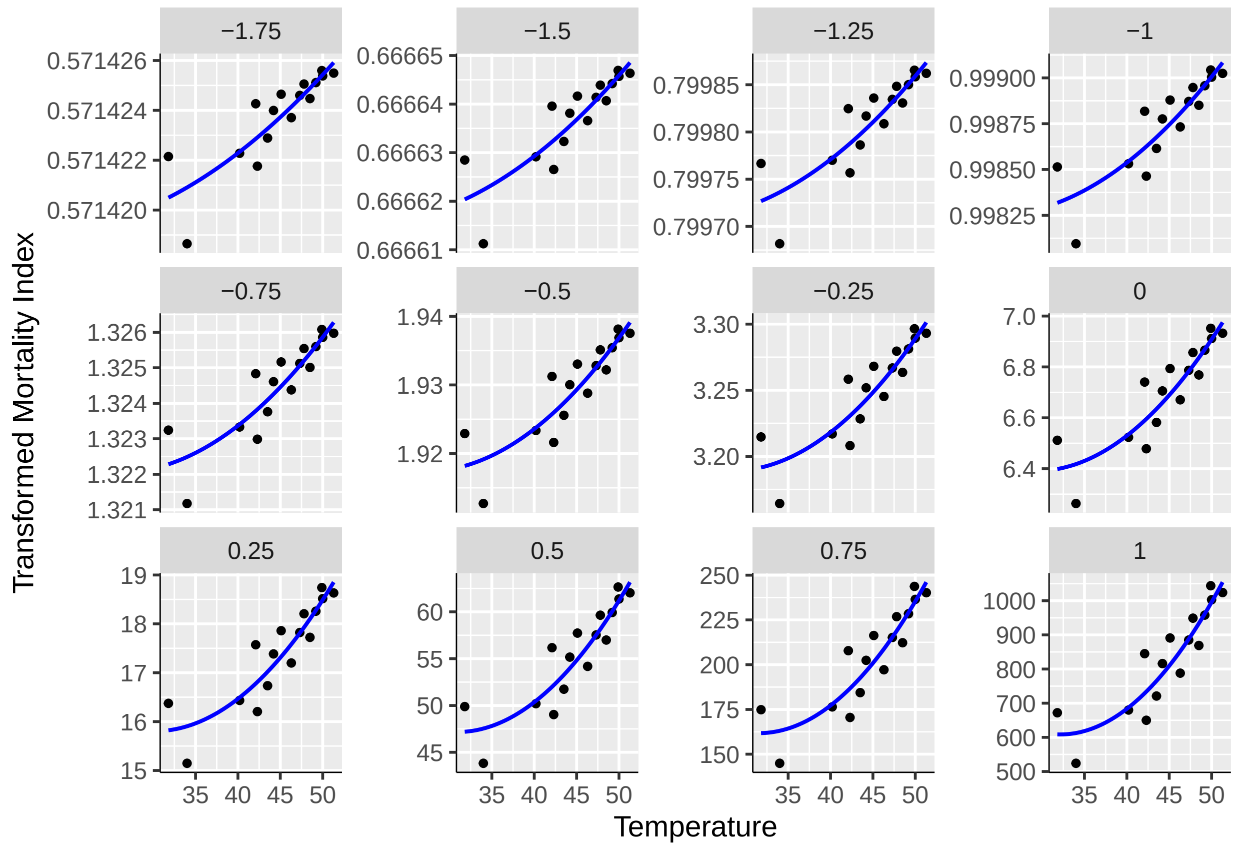
But only observing is not always correct, so I conducted a polynomial regression with degree 2, because the degree 2 polynomial is either convex or concave.

```r
ggplot(data = df, aes(x = temperature, y = mortality_index)) +
  geom_point() +
  stat_smooth(method = "lm", formula = y ~ x + I(x^2), se = FALSE, color = "blue") +
  labs(title = "Temperature vs Mortality Index",
       x = "Temperature",
       y = "Mortality Index") +
  theme_minimal()
```



The fitted polynomial with degree 2 shows convex shape. Therefore the plot is hollow up.

```r
BC_trans <- function(x, tau) {
  ifelse(tau == 0, log(x), (x^tau - 1) / tau)
}

taus <- seq(-1.75, 1, by = .25)
df_expanded <- df %>%
  crossing(tau = taus) %>%
  mutate(mort_idx_transformed = BC_trans(mortality_index, tau))

ggplot(data = df_expanded, aes(x = temperature, y = mort_idx_transformed)) +
  geom_point(size = 1) +
  stat_smooth(method = "lm", formula = y ~ x + I(x^2), se = FALSE, color = "blue", linewidth = 0.7) +
  labs(x = "Temperature", y = "Transformed Mortality Index") +
  theme(
    axis.line = element_line(linewidth = 0.25, colour = 'black'),
    axis.ticks = element_line(linewidth = 0.5)
  ) +
  facet_wrap(~ tau, scales = "free_y")
```

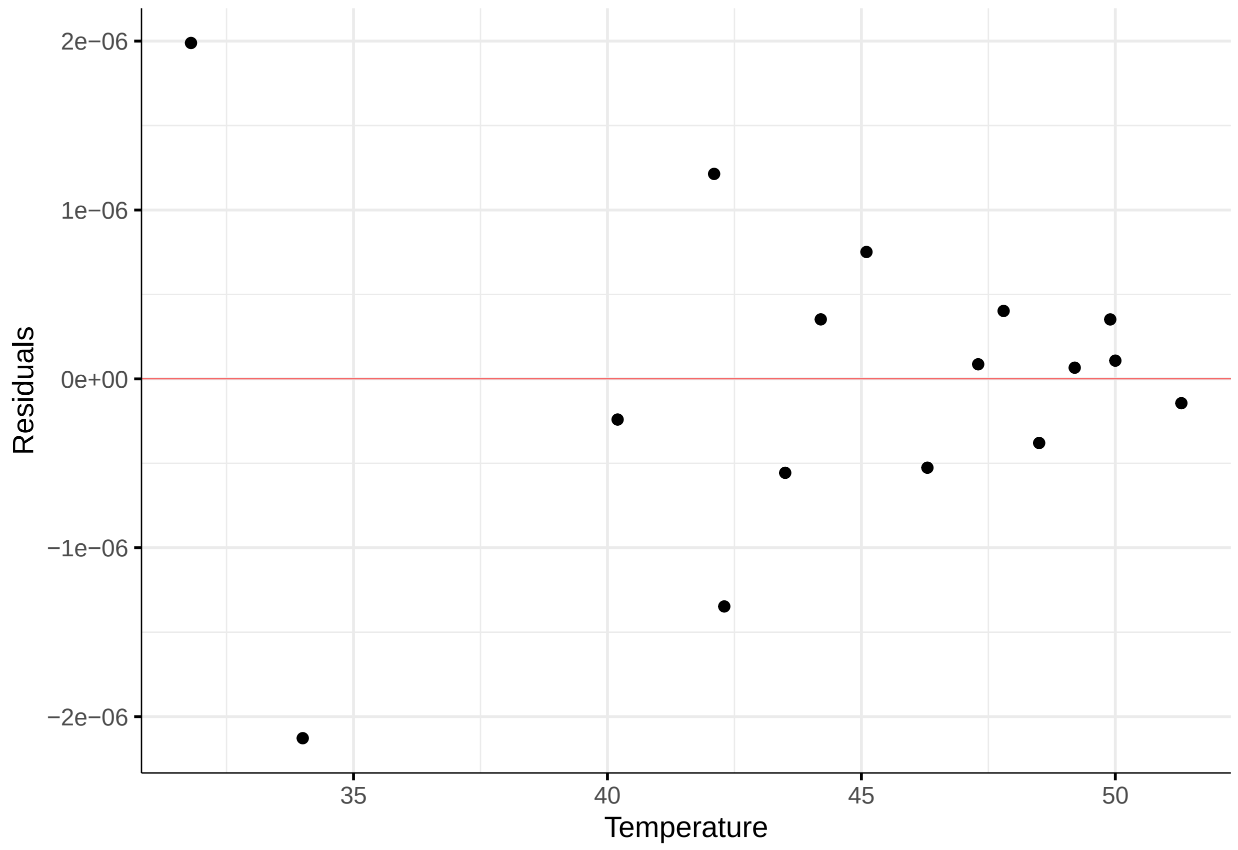


To straighten the relationship, box-cox transformation is used to the variable 'Mortality index'. And the plot above shows the transformation by each `tau`.

The logic is similar with the logic in determination of hollow up or down, where the polynomial with degree 2 is used.

I concluded that box-cox transformation with `tau = -1.75` would straighten the relationship.

```r
tau <- -1.75
df %>%
  rowwise() %>%
  mutate(transformed = BC_trans(mortality_index, tau)) %>%
  ungroup() -> df.1

lin_reg <- lm(transformed ~ temperature, data = df.1)

df.1 <- df.1 %>%
  mutate(residuals = lin_reg$residuals)

ggplot(data = df.1, aes(x = temperature, y = residuals)) +
  geom_point() +
  geom_abline(intercept = 0,
              slope = 0,
              color = 'red',
              linewidth = 0.15) +
  theme_minimal() +
  labs(x = "Temperature", y = "Residuals") +
  theme(
    axis.line = element_line(linewidth = 0.25, colour = 'black'),
    axis.ticks = element_line(linewidth = 0.5)
  )
```



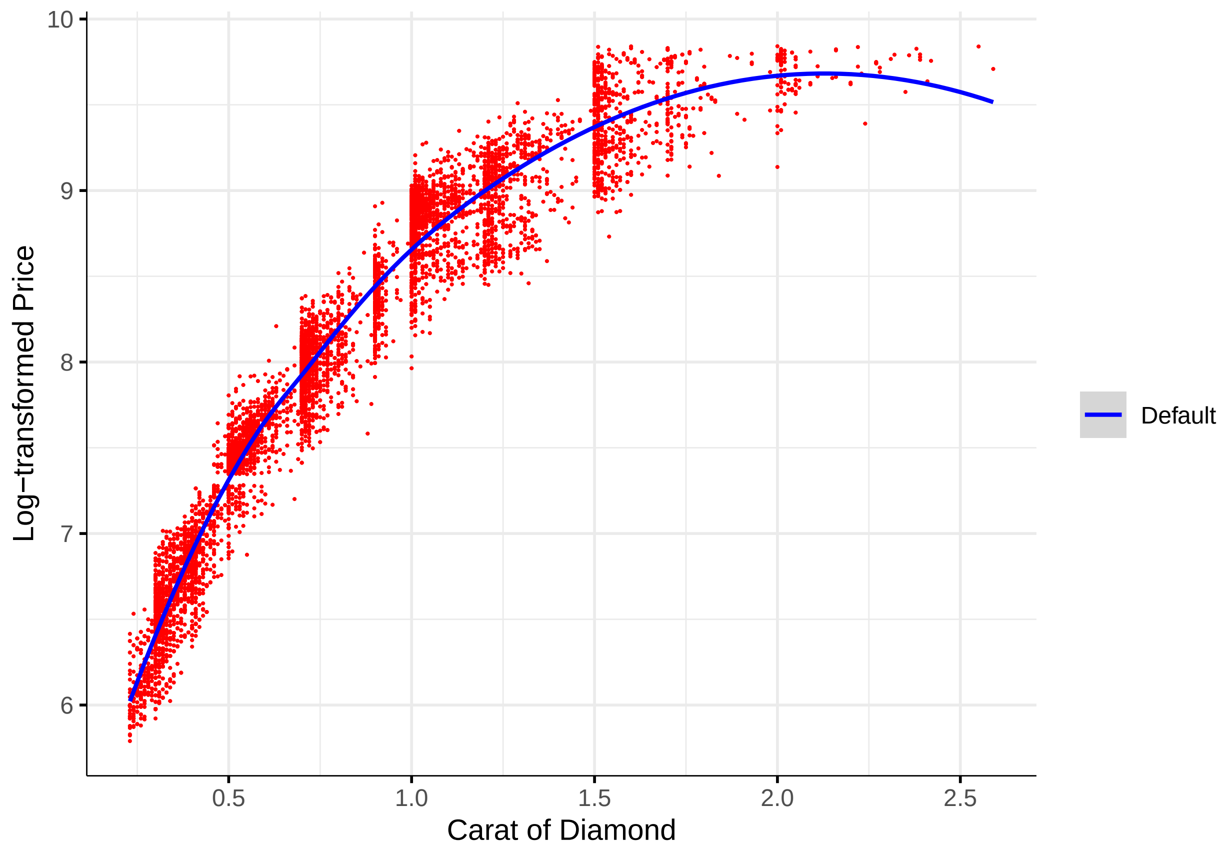
The plot above displays the residuals from applying a simple linear regression to data that has been Box-Cox transformed with a parameter `tau = -1.75`.

A noticeable pattern is that the spread of residuals decreases as temperature increases. In my opinion, this occurs because the variability of the mortality index was greater at lower temperatures.

Problem 4: Loess smoothing and model comparison

```
VS1.d <- diamonds %>%
  subset(clarity == "VS1")

loessplot <- ggplot(data = VS1.d, aes(x = carat,
                                      y = log(price),
                                      colour = "Default")) +
  scale_colour_manual("", values = c("Default"="blue")) +
  geom_point(color = 'red', size = 0.05) +
  geom_smooth(method = 'loess', level = 0, linewidth = 0.75) +
  theme_minimal() +
  labs(x = "Carat of Diamond", y = "Log-transformed Price") +
  theme(
    axis.line = element_line(linewidth = 0.25, colour = 'black'),
    axis.ticks = element_line(linewidth = 0.5)
  )

loessplot
```

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



The loess fitted plot is shown above.

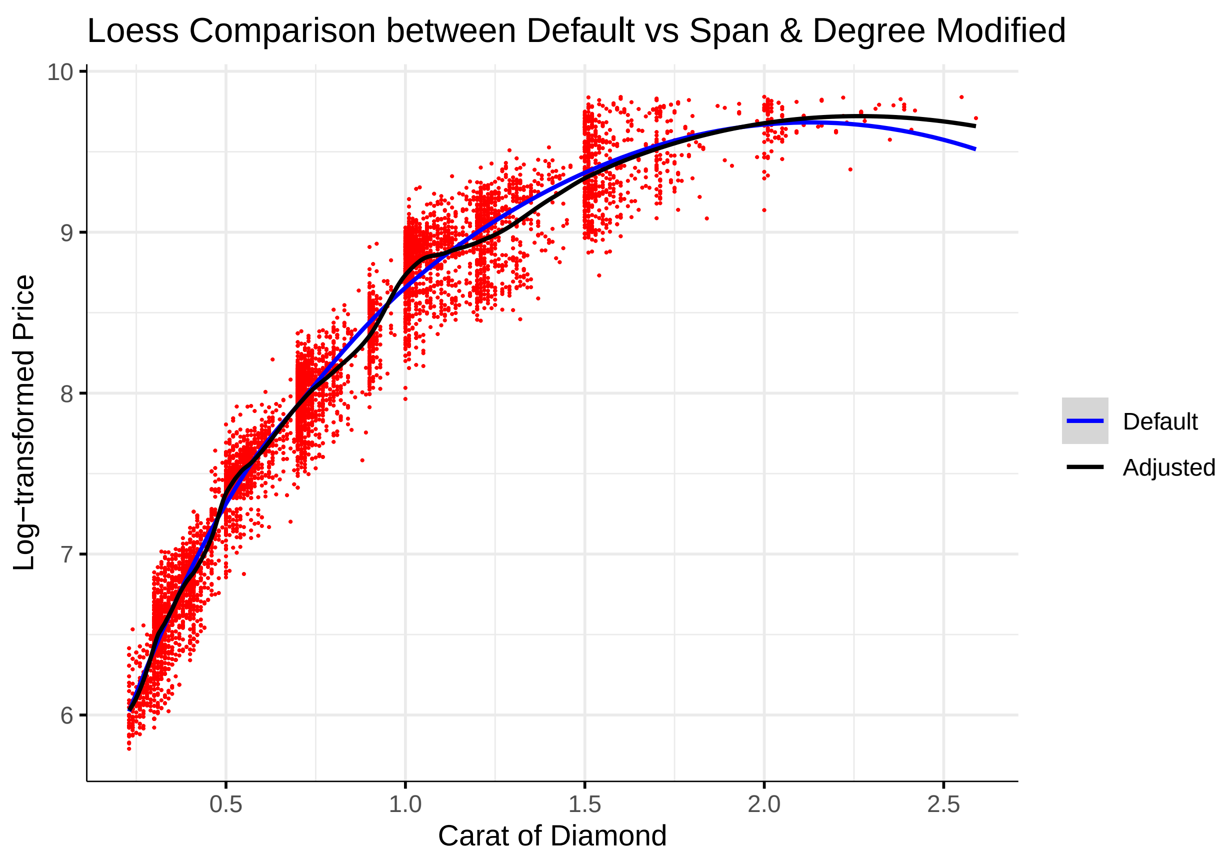
```
span.v <- c(0.2, 0.35, 0.5, 0.75, 1)
degree.v <- c(0, 1, 2)

loess_df <- lapply(span.v, function(span) {
  sapply(degree.v, function(degree) {
    ll <- summary(loess(log(price) ~ carat,
                  data = VS1.d,
                  span = span,
                  degree = degree))[[5]] %>% round(., 3)
  })
}) %>% do.call(cbind, .) %>% as.data.frame()%>%
  rownames_to_column(var = "degree") %>%
  mutate(degree = paste0("Degree ", degree.v)) %>% as_tibble()

colnames(loess_df) <- c("", paste0("Span ", span.v))
```

```r
loess_df %>%
  kable(format = "markdown")
```

	Span 0.2	Span 0.35	Span 0.5	Span 0.75	Span 1
Degree 0	0.198	0.227	0.274	0.428	0.865
Degree 1	0.185	0.189	0.194	0.205	0.281
Degree 2	0.183	0.185	0.188	0.190	0.196

I created a table by varying the values of `span` and `degree`. The default settings are `degree = 2` and `span = 0.75`, which result in a residual standard error of 0.190. The lowest residual standard error was achieved when `span = 0.2` and `degree = 2`, where I thought it is most appropriate.

```r
span = 0.2
degree = 2
grid <- seq(min(VS1.d$carat), max(VS1.d$carat), length.out = 1000)

pred_vals <- loess(log(price) ~ carat, data = VS1.d,
                     span = span, degree = degree) %>%
  predict(., newdata = grid) %>% cbind(., grid)

loessplot.1 <- loessplot +
  geom_line(data = pred_vals, aes(x = grid, y = ., colour = "Adjusted"),
            linewidth = 0.75) +
  scale_colour_manual("",
                      breaks = c("Default", "Adjusted"),
                      values = c("Default" = "blue",
                                 "Adjusted" = "black")) +
  labs(title = "Loess Comparison between Default vs Span & Degree Modified")

loessplot.1
```

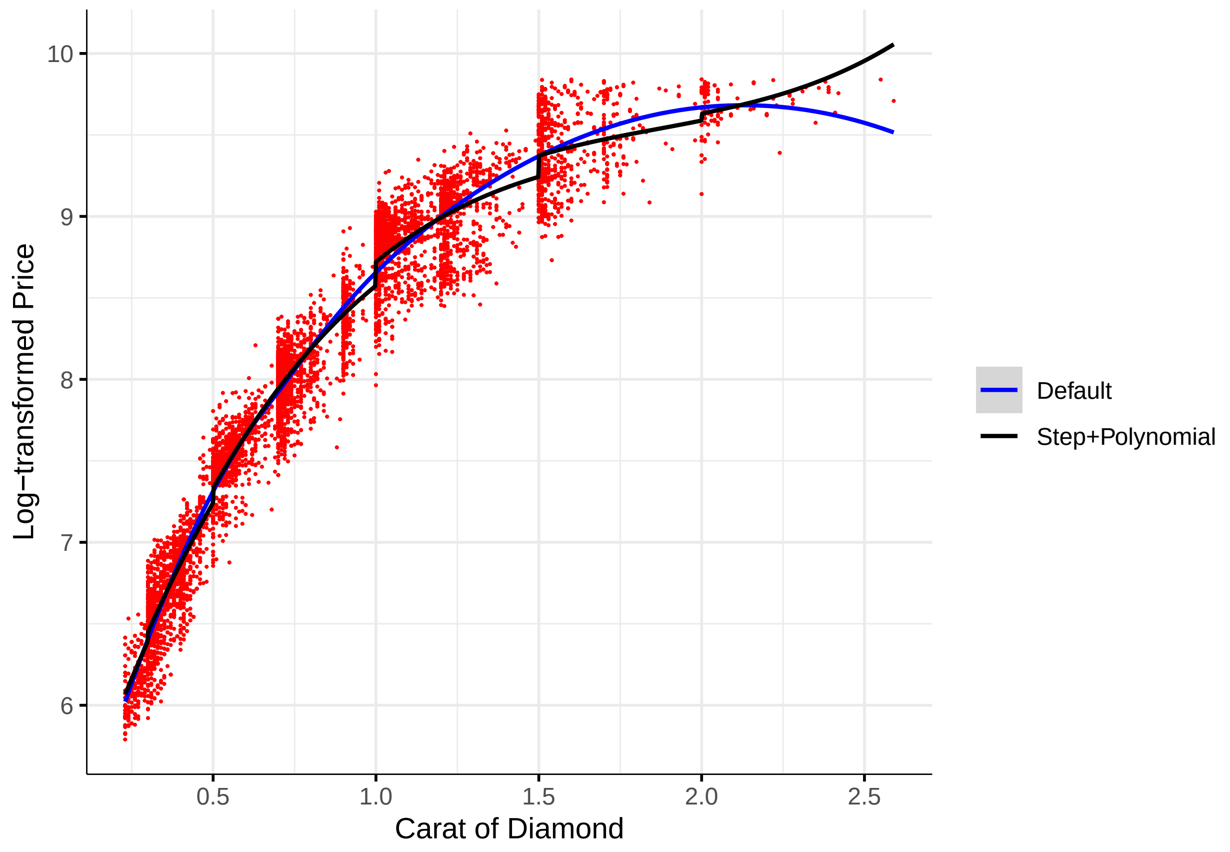


After checking the residual standard error for various composition of `degree` and `span`, I choosed the loess with `degree = 2` and `span = 0.2`.

The black line in above plot shows the fitted loess.

```r
step = function(x, step_position) {
  return(ifelse(x <= step_position, 0, 1))
}
df <- data.frame(carat = grid)

lm.steps <- lm(
  log(price) ~ carat + I(carat^2) + I(carat^3) +
    step(carat, .3) + step(carat, .5) + step(carat, 1) +
    step(carat, 1.5) + step(carat, 2),
  data = VS1.d
)

df$pred <- predict(lm.steps, newdata = df)

plot_comparison <- loessplot +
  geom_line(data = df, aes(x = carat, y = pred, colour = "Step+Polynomial"),
            linewidth = 0.75) +
  scale_colour_manual("",
                      breaks = c("Default", "Step+Polynomial"),
                      values = c("Default" = "blue",
                                 "Step+Polynomial" = "black"))
```

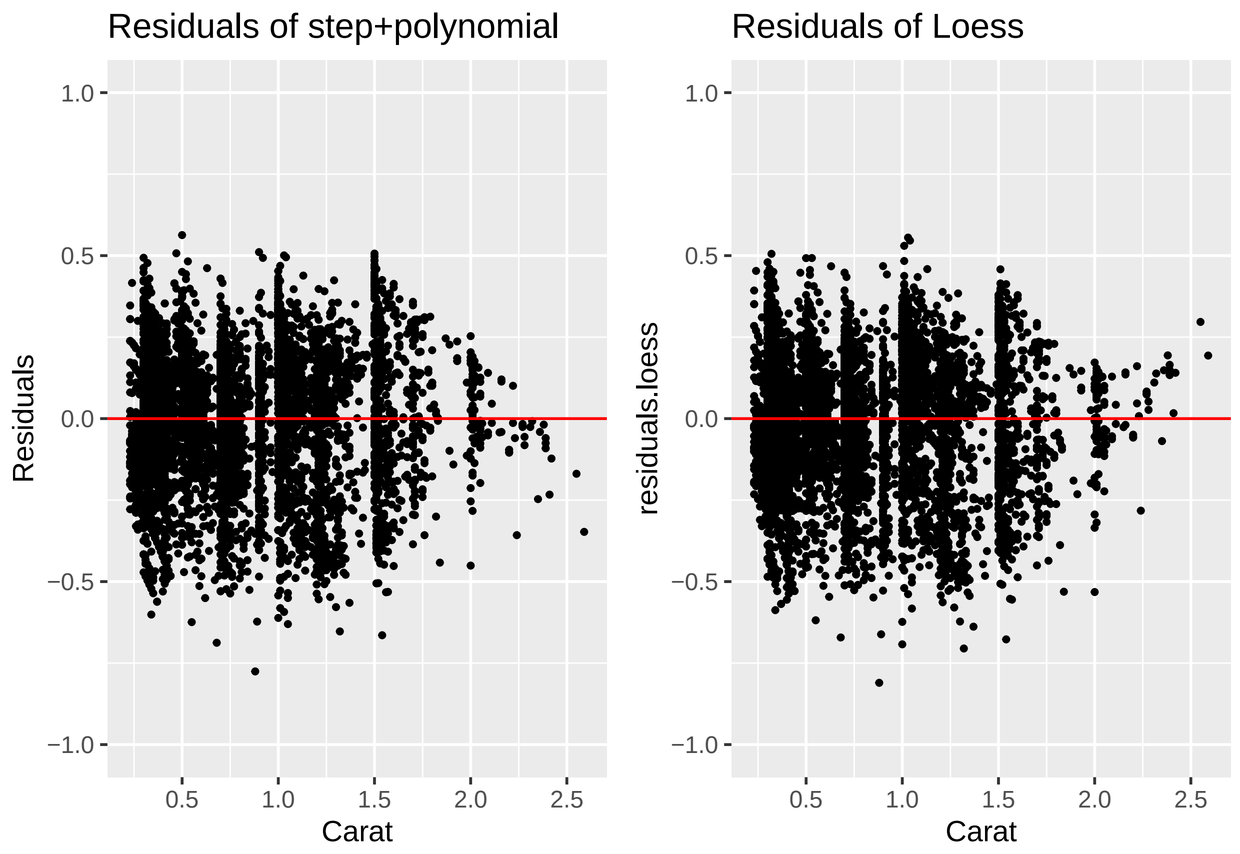
```
plot_comparison
```



The plot above shows the comparison between loess regression (blue) and step+polynomial regression (black). Residual comparison plot is shown below:

```
loess_model <- loess(
  log(price) ~ carat,
  data = VS1.d
)

VS1.d <- VS1.d %>%
  mutate(residuals.lm_steps = lm.steps$residuals) %>%
  mutate(residuals.loess = loess_model$residuals)


y_range <- range(c(VS1.d$residuals.lm_steps, VS1.d$residuals.loess))

p1 <- ggplot(VS1.d, aes(x = carat, y = residuals.lm_steps)) +
  geom_point(size = 0.75) +
  geom_hline(yintercept = 0, color = "red") +
  ylim(c(-1,1)) +
  labs(title = "Residuals of step+polynomial", x = "Carat", y = "Residuals")

p2 <- ggplot(VS1.d, aes(x = carat, y = residuals.loess)) +
  geom_point(size = 0.75) +
```

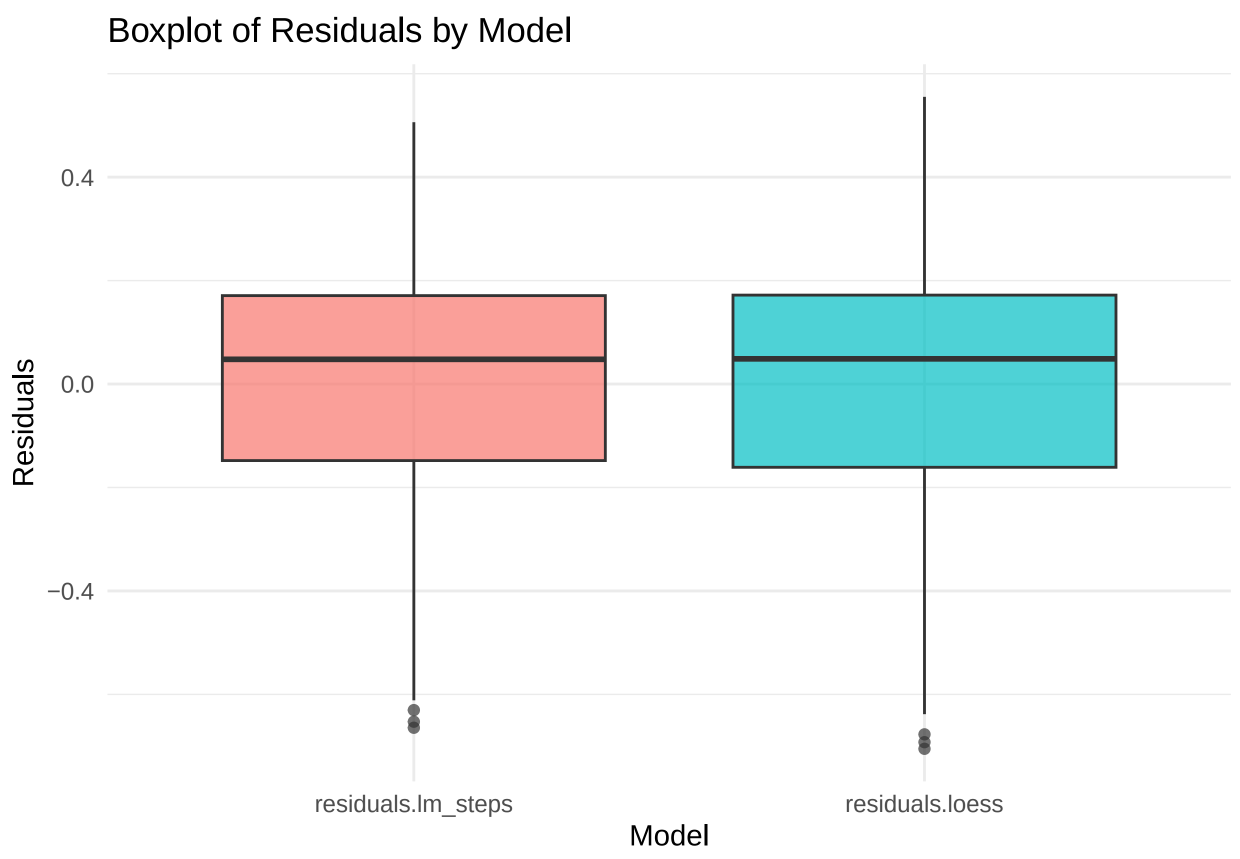
```r
  geom_hline(yintercept = 0, color = "red") +
  ylim(c(-1, 1)) +
  labs(title = "Residuals of Loess", x = "Carat")

grid.arrange(p1, p2, ncol = 2)
```



I couldn't speak easily which one is better, so I drawed a boxplot for residuals.

But when `carat < 1`, there is no significant difference between loess line and step+polynomial line, therefore compared the residuals with `carat >= 1`.

```r
residuals_long <- VS1.d %>%
  subset(carat >= 1) %>%
  select(residuals.lm_steps, residuals.loess) %>%
  pivot_longer(cols = everything(), names_to = "Model", values_to = "Residuals")

ggplot(residuals_long, aes(x = Model, y = Residuals, fill = Model)) +
  geom_boxplot(alpha = 0.7) +
  labs(title = "Boxplot of Residuals by Model",
       x = "Model",
       y = "Residuals") +
  theme_minimal() +
  theme(legend.position = "none")
```



I focused on observing the interquartile range (IQR) of the residuals. I found that the IQR of the residuals from the step+polynomial regression is smaller than that from the loess regression.

Therefore, I conclude that the step+polynomial model captures the data's distribution more accurately.