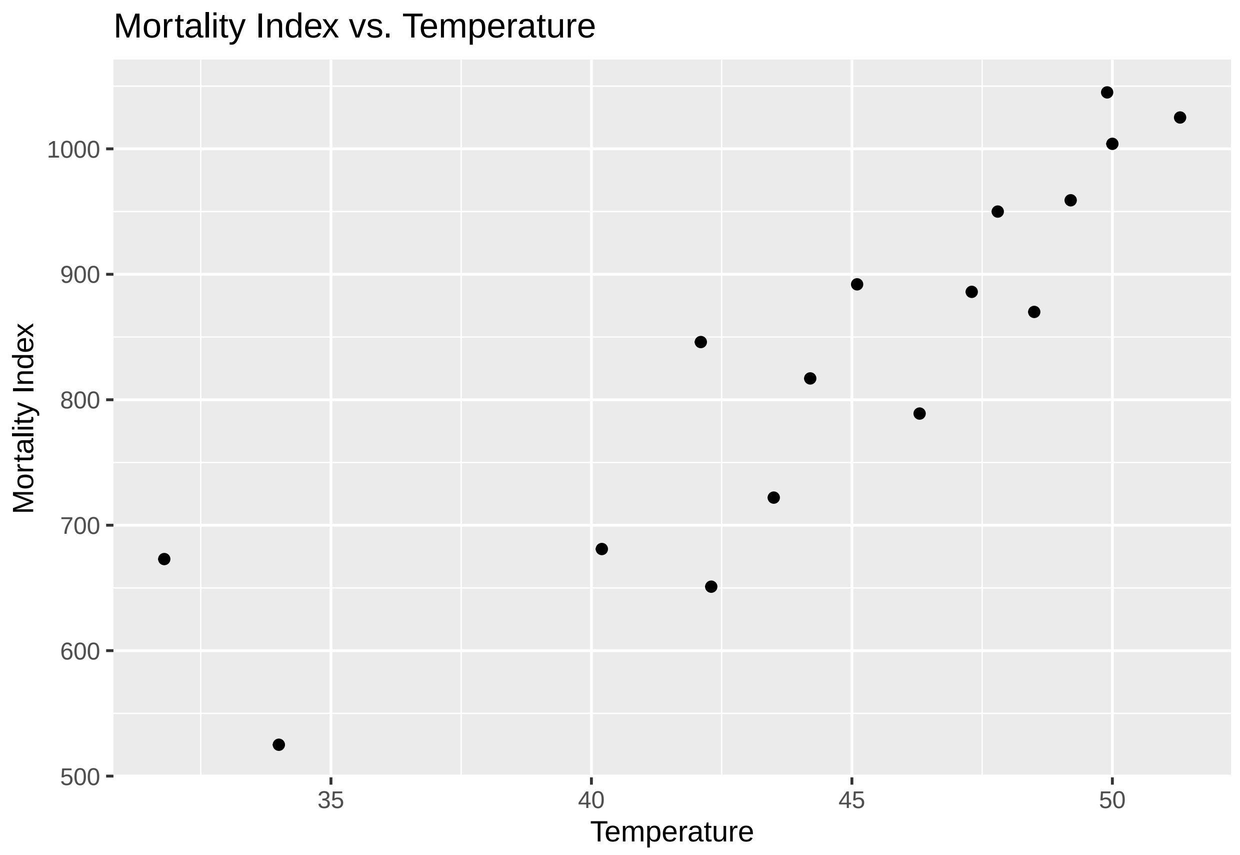
```r
rm(list=ls())
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(broom)
data <- read_csv("mortality_by_latitude.csv")
```

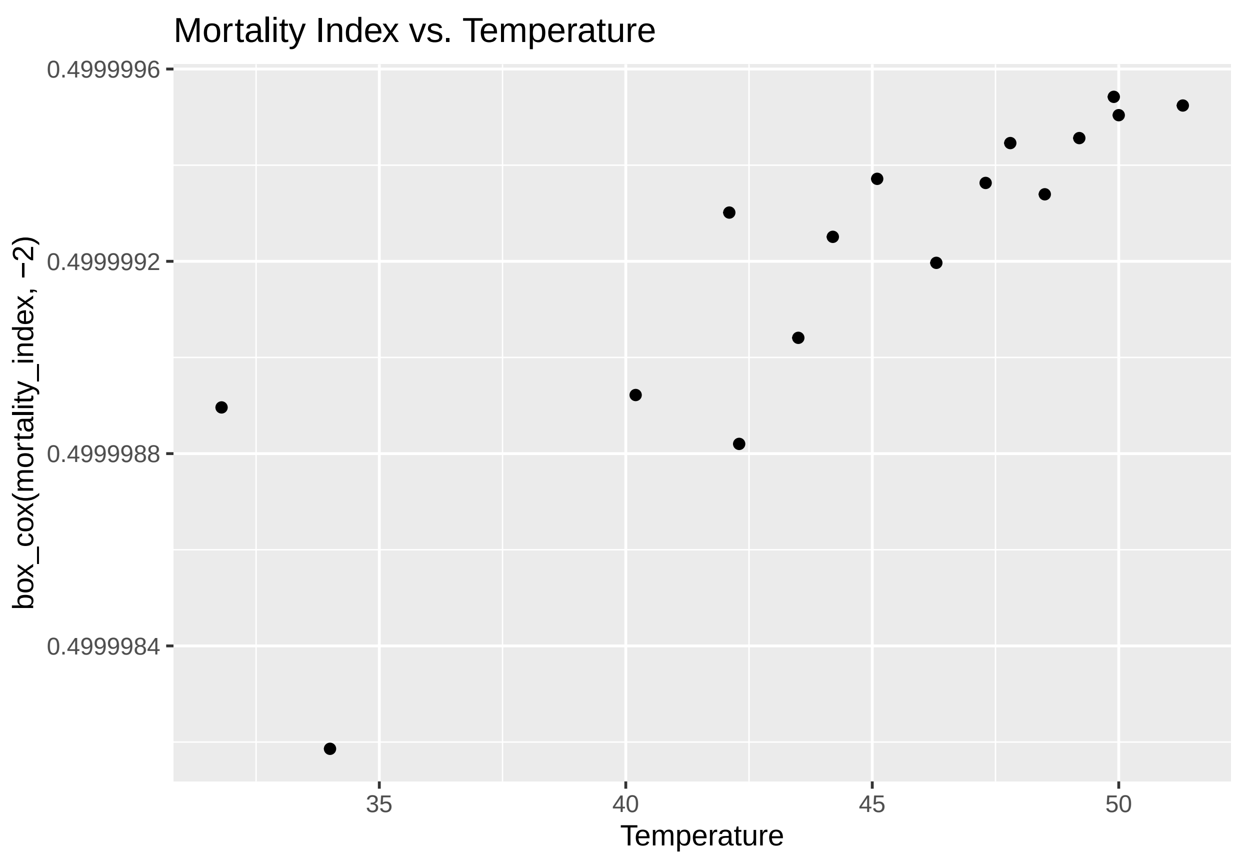
```
## Rows: 16 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (3): latitude, mortality_index, temperature
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
## 1
ggplot(data=data, aes(x=temperature, y=mortality_index)) +
  geom_point() +
  xlab("Temperature") +
  ylab("Mortality Index") +
  ggtitle("Mortality Index vs. Temperature")
```

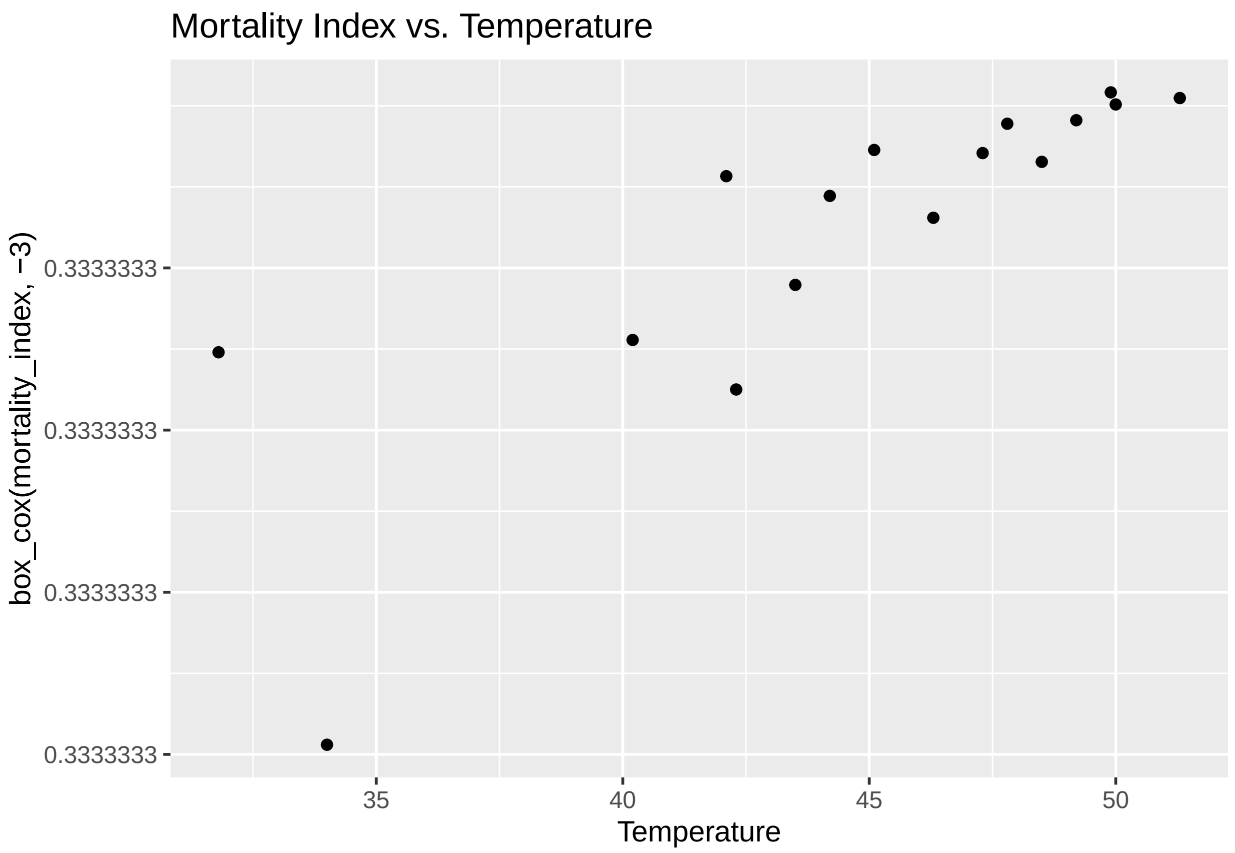


This plot is hollow upward because if I take three points on a curve, the middle point is below the line joining the other two.

```
box_cox <- function(y, tau) {
    if (tau==0) {
      return(log10(y))
    } else {
      return((y^tau - 1) / tau)
    }
}

ggplot(data=data, aes(x=temperature, y=box_cox(mortality_index,-2))) +
  geom_point() +
  xlab("Temperature") +
  ggtitle("Mortality Index vs. Temperature")
```

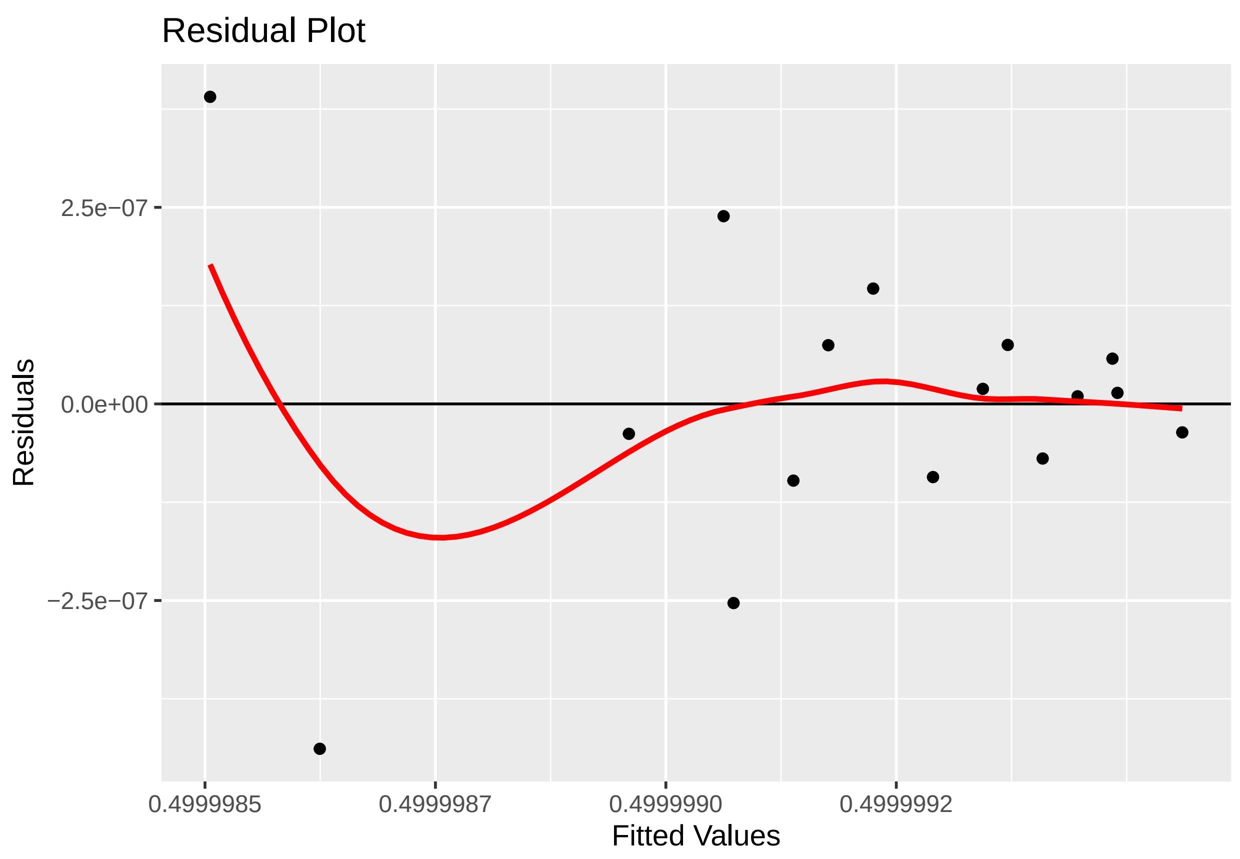


```
ggplot(data=data, aes(x=temperature, y=box_cox(mortality_index,-3))) +
  geom_point() +
  xlab("Temperature") +
  ggtitle("Mortality Index vs. Temperature")
```



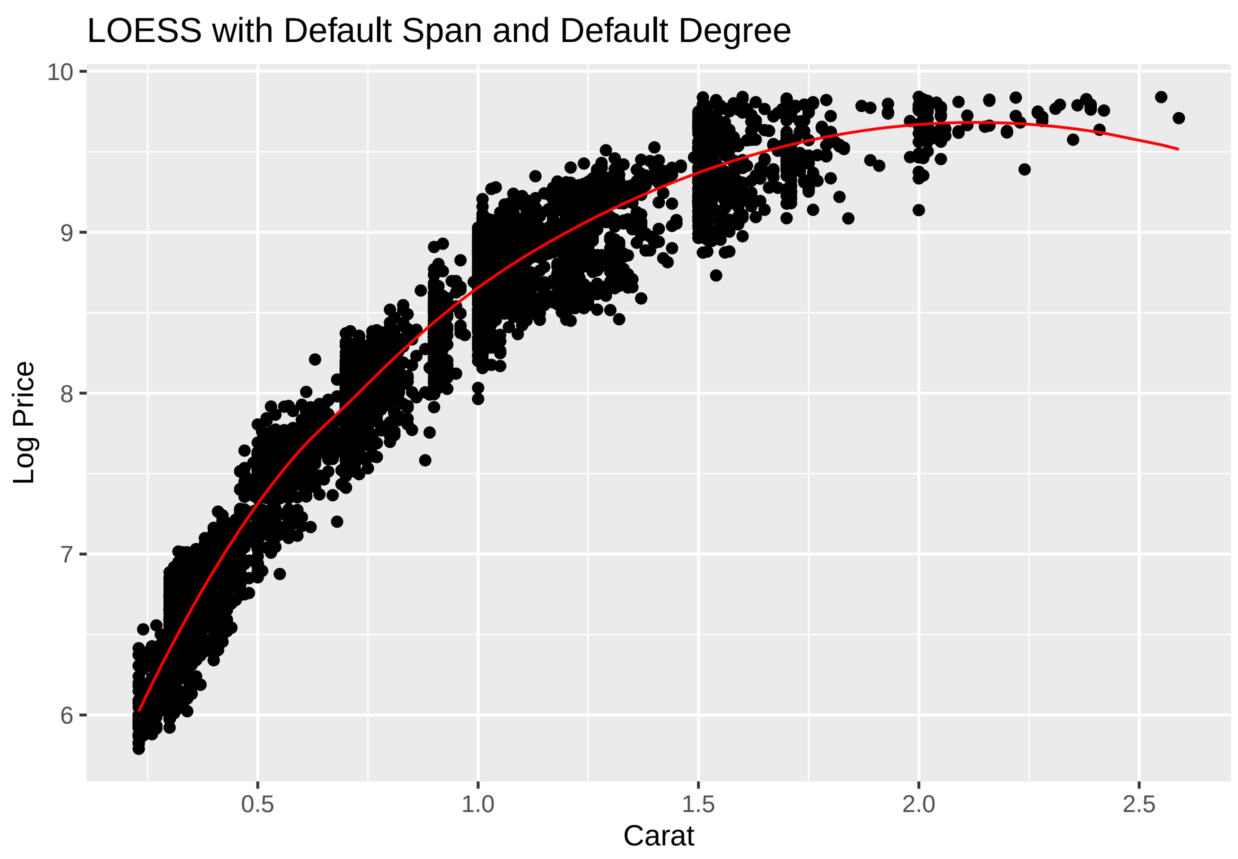
Since the plot of the (untransformed) mortality index vs. temperature is hollow upward, I move down the ladder of Box-Cox transformations. Setting tau = -2 straightens out the relationship. Going further down the ladder (tau = -3) makes the relationship hollow downward, so tau = -2 is my choice for a transformation.

```
model <- lm(box_cox(mortality_index,-2) ~ temperature, data=data)
data <- data %>%
  mutate(residuals = residuals(model),
         fitted.values = fitted.values(model))
ggplot(data=data, aes(x=fitted.values, y=residuals)) +
  geom_point() +
  geom_hline(yintercept=0, col="black") +
  geom_smooth(method="loess", formula="y~x", col="red", se=F) +
  xlab("Fitted Values") +
  ylab("Residuals") +
  ggtitle("Residual Plot")
```

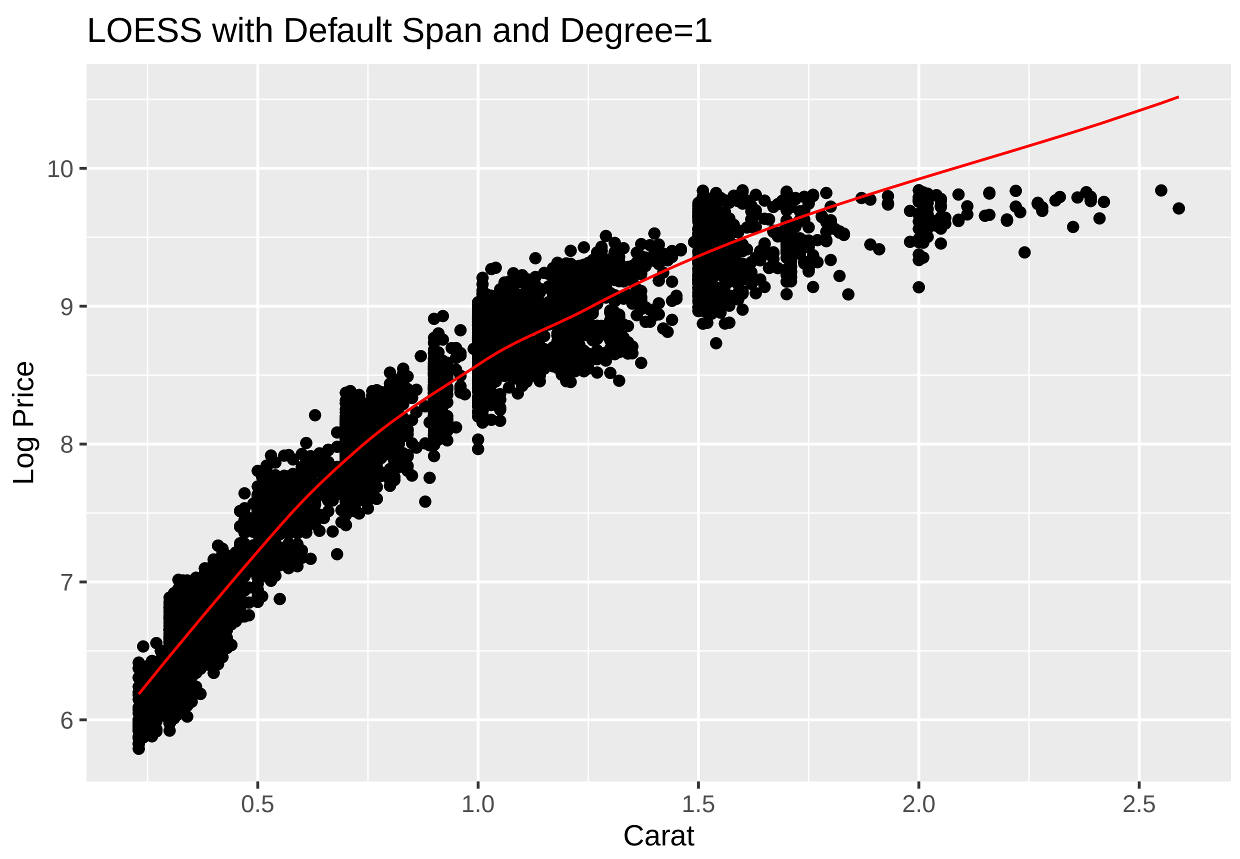


Outside of the two outlier points (at low levels of the fitted values), the residual plot when y is transformed using tau = -2 is a null plot. This means that the assumption of linearity is met. However, the residuals form a funnel shape, which indicates heteroskedasticity.

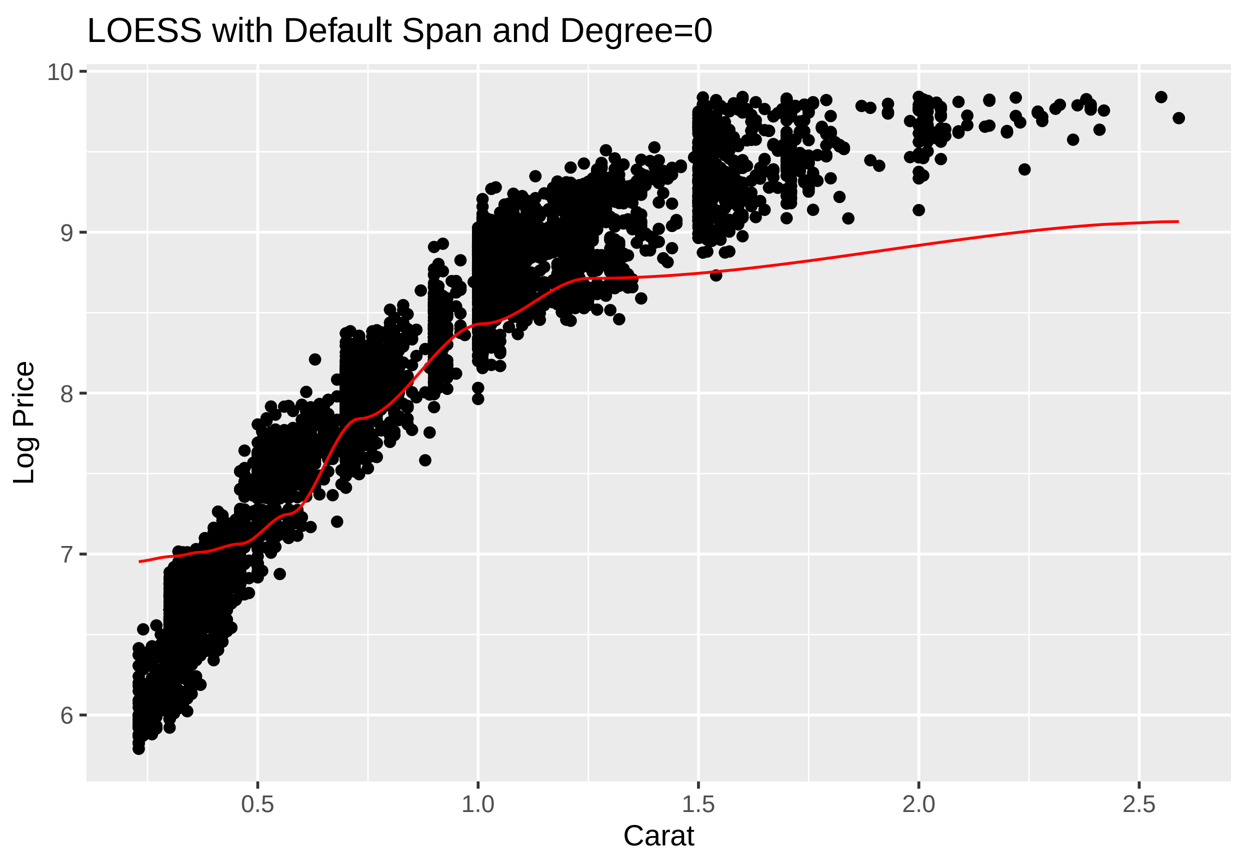
```r
## 2
data(diamonds)
diamonds <- diamonds %>%
  filter(clarity == "VS1") %>%
  mutate(log.price = log(price))

loess.model <- loess(log.price ~ carat, data=diamonds)
ggplot(augment(loess.model, data=diamonds), aes(x=carat, y=.fitted)) +
  geom_point(aes(y=log.price)) +
  geom_line(color="red") +
  xlab("Carat") +
  ylab("Log Price") +
  ggtitle("LOESS with Default Span and Default Degree")
```



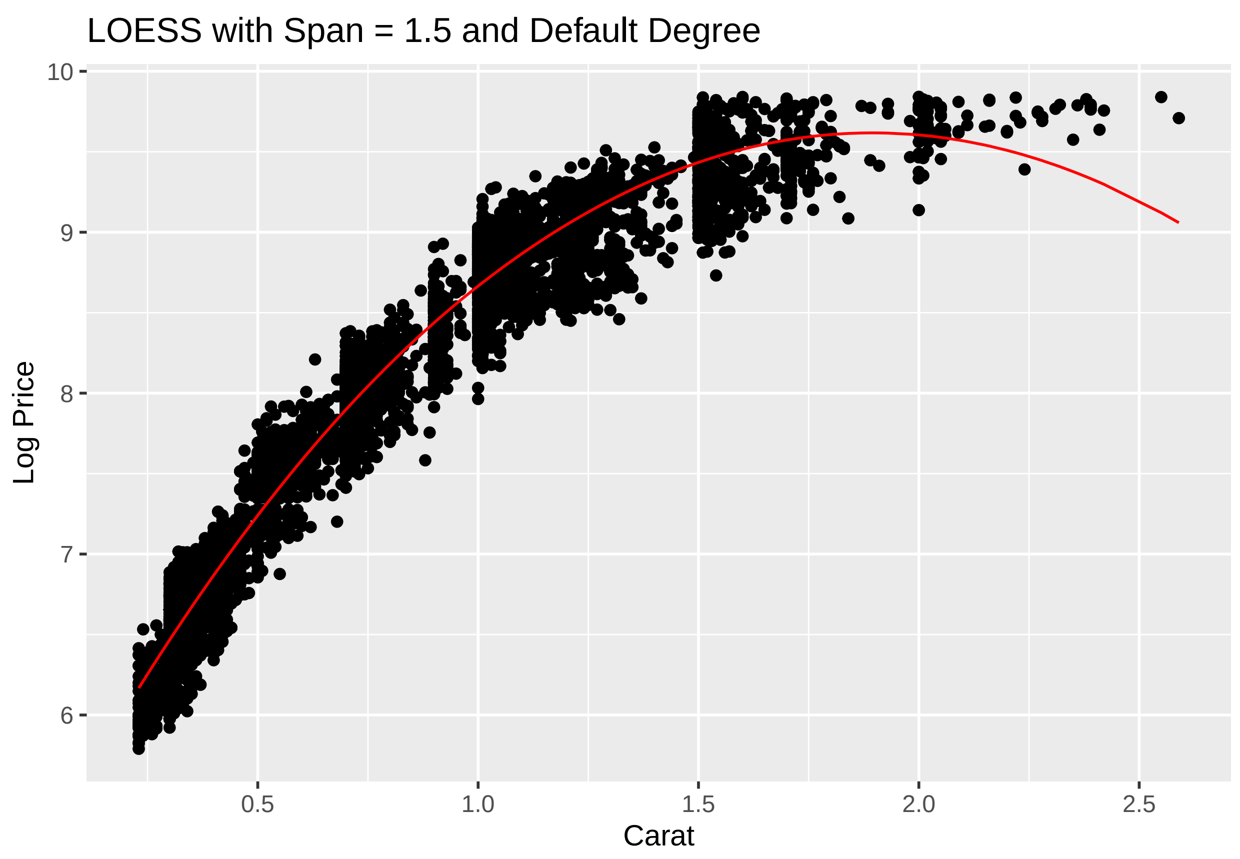
```r
loess.model <- loess(log.price ~ carat, data=diamonds, degree=1)
ggplot(augment(loess.model, data=diamonds), aes(x=carat, y=.fitted)) +
  geom_point(aes(y=log.price)) +
  geom_line(color="red") +
  xlab("Carat") +
  ylab("Log Price") +
  ggtitle("LOESS with Default Span and Degree=1")
```



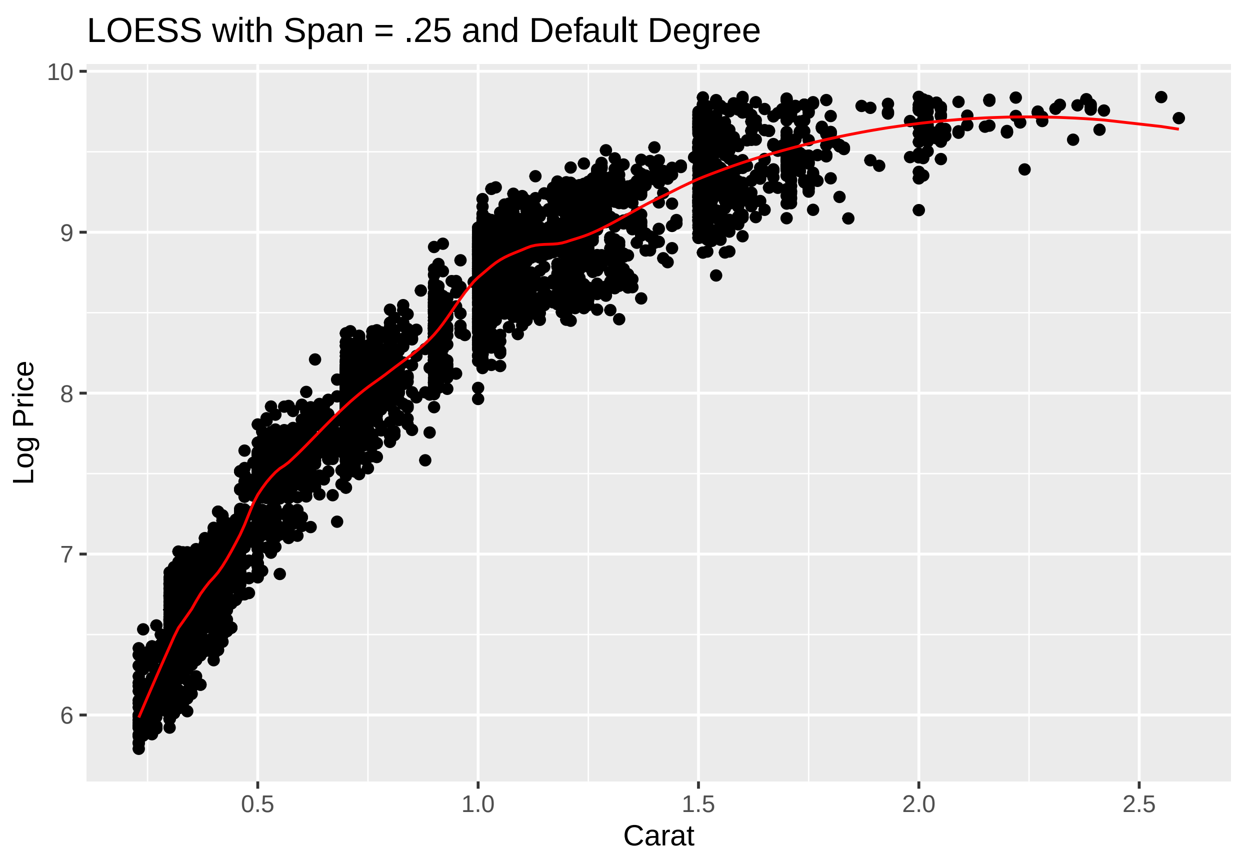
```
loess.model <- loess(log.price ~ carat, data=diamonds, degree=0)
ggplot(augment(loess.model, data=diamonds), aes(x=carat, y=.fitted)) +
  geom_point(aes(y=log.price)) +
  geom_line(color="red") +
  xlab("Carat") +
  ylab("Log Price") +
  ggtitle("LOESS with Default Span and Degree=0")
```



```
loess.model <- loess(log.price ~ carat, data=diamonds, span=1.5)
ggplot(augment(loess.model, data=diamonds), aes(x=carat, y=.fitted)) +
  geom_point(aes(y=log.price)) +
  geom_line(color="red") +
  xlab("Carat") +
  ylab("Log Price") +
  ggtitle("LOESS with Span = 1.5 and Default Degree")
```

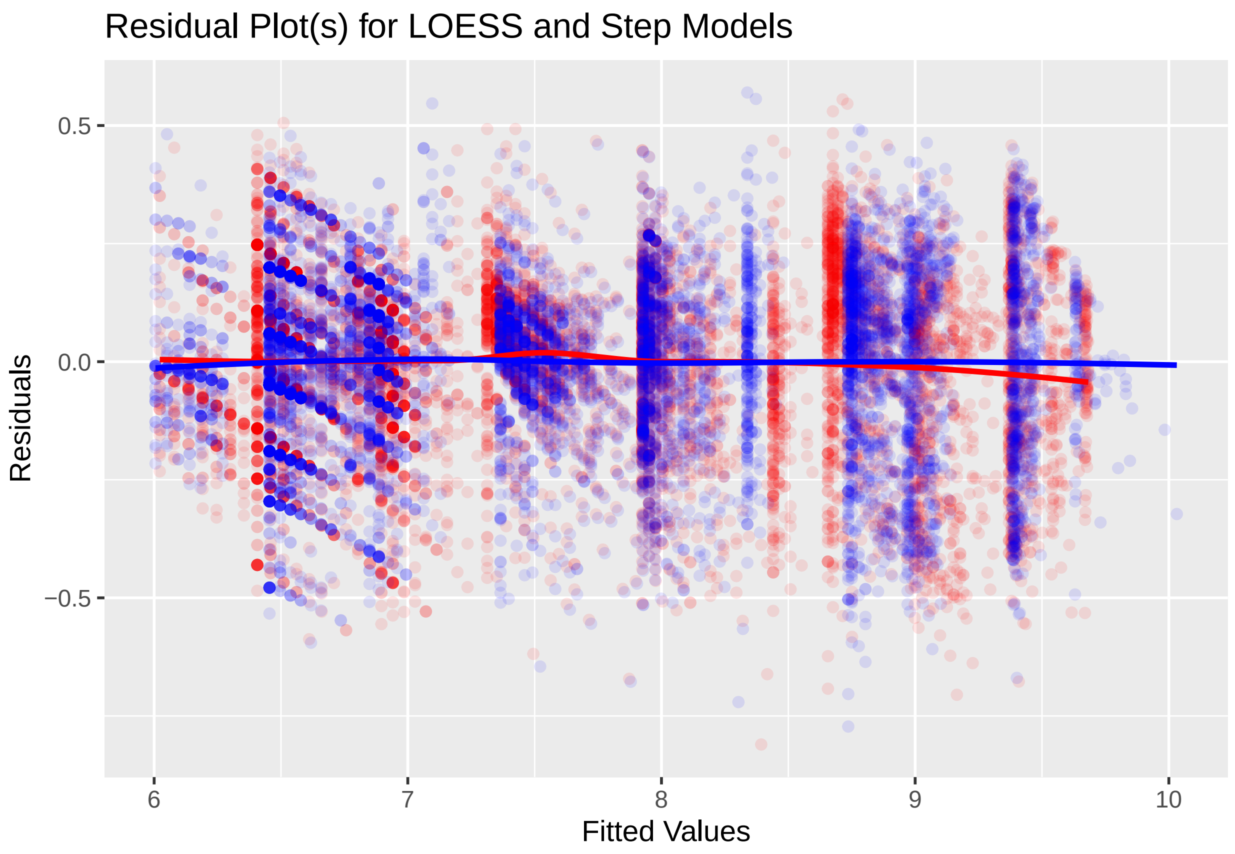


```
loess.model <- loess(log.price ~ carat, data=diamonds, span=.25)
ggplot(augment(loess.model, data=diamonds), aes(x=carat, y=.fitted)) +
  geom_point(aes(y=log.price)) +
  geom_line(color="red") +
  xlab("Carat") +
  ylab("Log Price") +
  ggtitle("LOESS with Span = .25 and Default Degree")
```



I choose the default span of 0.75 and the default degree of 2 because these parameters balance capturing the patterns in the data without overfitting to the data. The degree parameters can be 0, 1, or 2 (default). Setting the degree to 1 or 0 results in a LOESS smoother that misses portions of the data. Reducing the default span to 0.25 overfits to the data. Doubling the span to 1.5 results in a smoother that does not fully capture the patterns in the data.

```
## 3.
loess.model <- loess(log.price ~ carat, data=diamonds)
diamonds <- diamonds %>%
  mutate(loess.residuals = residuals(loess.model),
         loess.fitted.values = fitted.values(loess.model))

step = function(x, step_position) {
    return(ifelse(x >= step_position, 0, 1))
}
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 = diamonds)
diamonds <- diamonds %>%
  mutate(step.residuals = residuals(lm.steps),
        step.fitted.values = fitted.values(lm.steps))

ggplot(diamonds) +
  geom_point(aes(x=loess.fitted.values, y=loess.residuals), col="red", alpha=.1) +
  geom_point(aes(x=step.fitted.values, y=step.residuals), col="blue", alpha=.1) +
  geom_smooth(aes(x=loess.fitted.values, y=loess.residuals), col="red", method="loess", formula="y~x", s
  geom_smooth(aes(x=step.fitted.values, y=step.residuals), col="blue", method="loess", formula="y~x", se
  ylab("Residuals") +
  xlab("Fitted Values") +
  ggtitle("Residual Plot(s) for LOESS and Step Models")
```



Both models are very faithful to the data, as LOESS smoothers on the residual plot for both models are almost perfect horizontal lines with y-intercepts of 0. However, the step model is slightly better in this case because the residuals decrease away from 0 every so slightly at the upper range of the fitted values for the LOESS model.