DSC_1105_FA4

Frances Aneth Rosales

2024-02-29

PLOTTING

Click to Hide/Show Plots

HIDE/SHOW

1

Create a histogram on the diamonds dataset, for example with ggplot() + geom_histogram(aes(x = carat), data = diamonds)

```
library(tidyverse)
## Warning: package 'tidyverse' was built under R version 4.2.3
## Warning: package 'tibble' was built under R version 4.2.3
## Warning: package 'tidyr' was built under R version 4.2.3
## Warning: package 'readr' was built under R version 4.2.3
## Warning: package 'purrr' was built under R version 4.2.3
## Warning: package 'dplyr' was built under R version 4.2.3
## Warning: package 'stringr' was built under R version 4.2.3
## Warning: package 'forcats' was built under R version 4.2.3
## Warning: package 'lubridate' was built under R version 4.2.3
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.1     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
```

```
## ✓ purrr     1.0.1
## ── Conflicts ─────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the  ]8;;http://conflicted.r-lib.org/ conflicted package ]8;;  to force all c
onflicts to become errors
library(car)
## Warning: package 'car' was built under R version 4.2.3
## Loading required package: carData
## Warning: package 'carData' was built under R version 4.2.3
##
## Attaching package: 'car'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
library(ggplot2)


library(readr)
mortality_file <- read_csv("C:/Users/asus/Documents/ALL FEU FILES/FEU FOLDER 6/DSC_110
5 Explo/FA4/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.
summary(mortality_file)
##     latitude     mortality_index   temperature
##  Min.   :50.00   Min.   : 525.0   Min.   :31.80
##  1st Qu.:53.75   1st Qu.: 711.8   1st Qu.:42.25
##  Median :57.50   Median : 858.0   Median :45.70
```

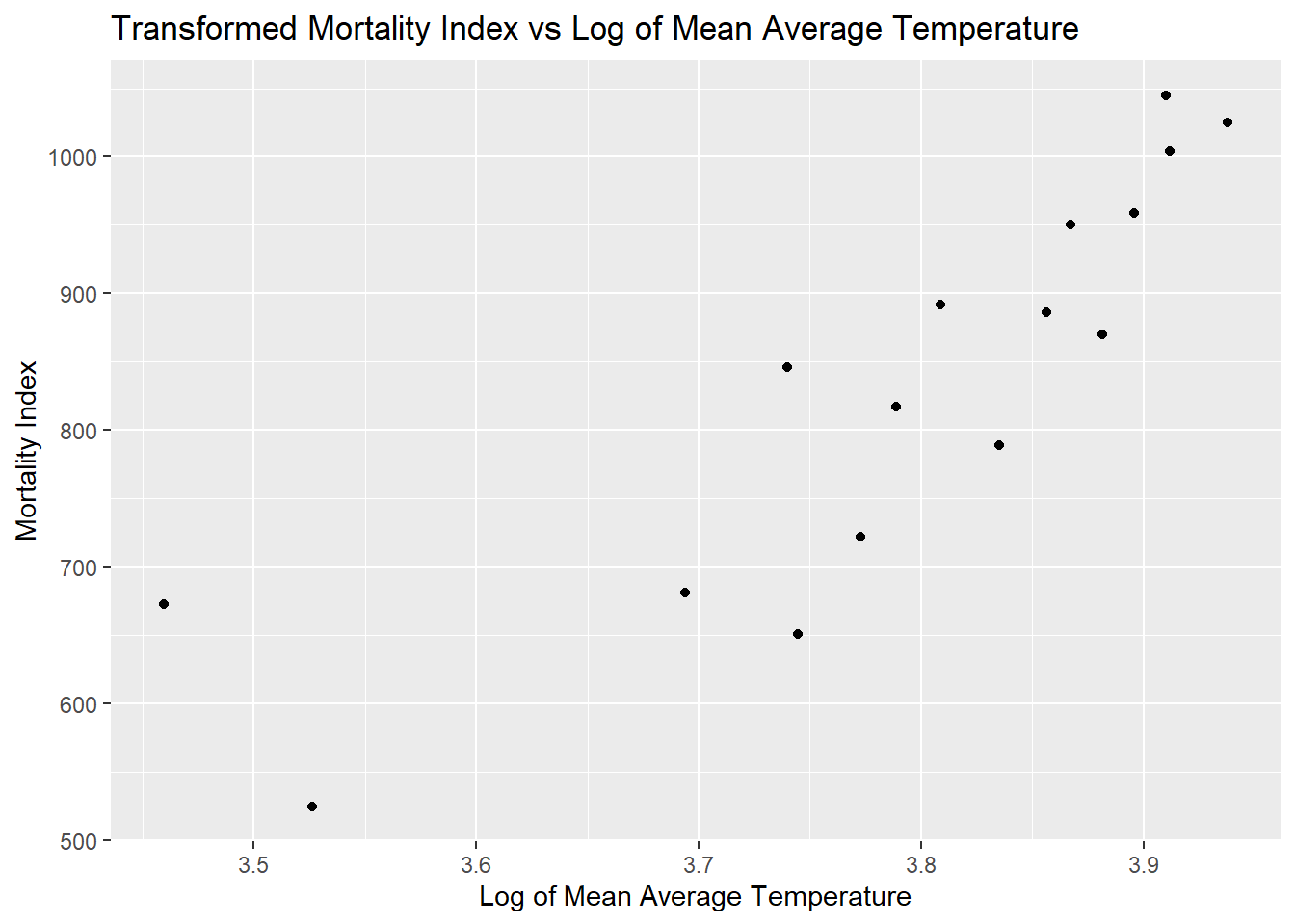
```
##  Mean   :58.12   Mean   : 833.4   Mean   :44.59
##  3rd Qu.:61.25   3rd Qu.: 952.2   3rd Qu.:48.67
##  Max.   :70.00   Max.   :1045.0   Max.   :51.30
```

1. Using the Mortality by Latitude data Download Mortality by Latitude data, make a plot of mortality index against mean average temperature. Is it hollow up or hollow down? Try to identify a transformation of one of the variables that will straighten out the relationship, and make a plot of the residuals to check for any remaining patterns.

Using the most common transformation, the log.

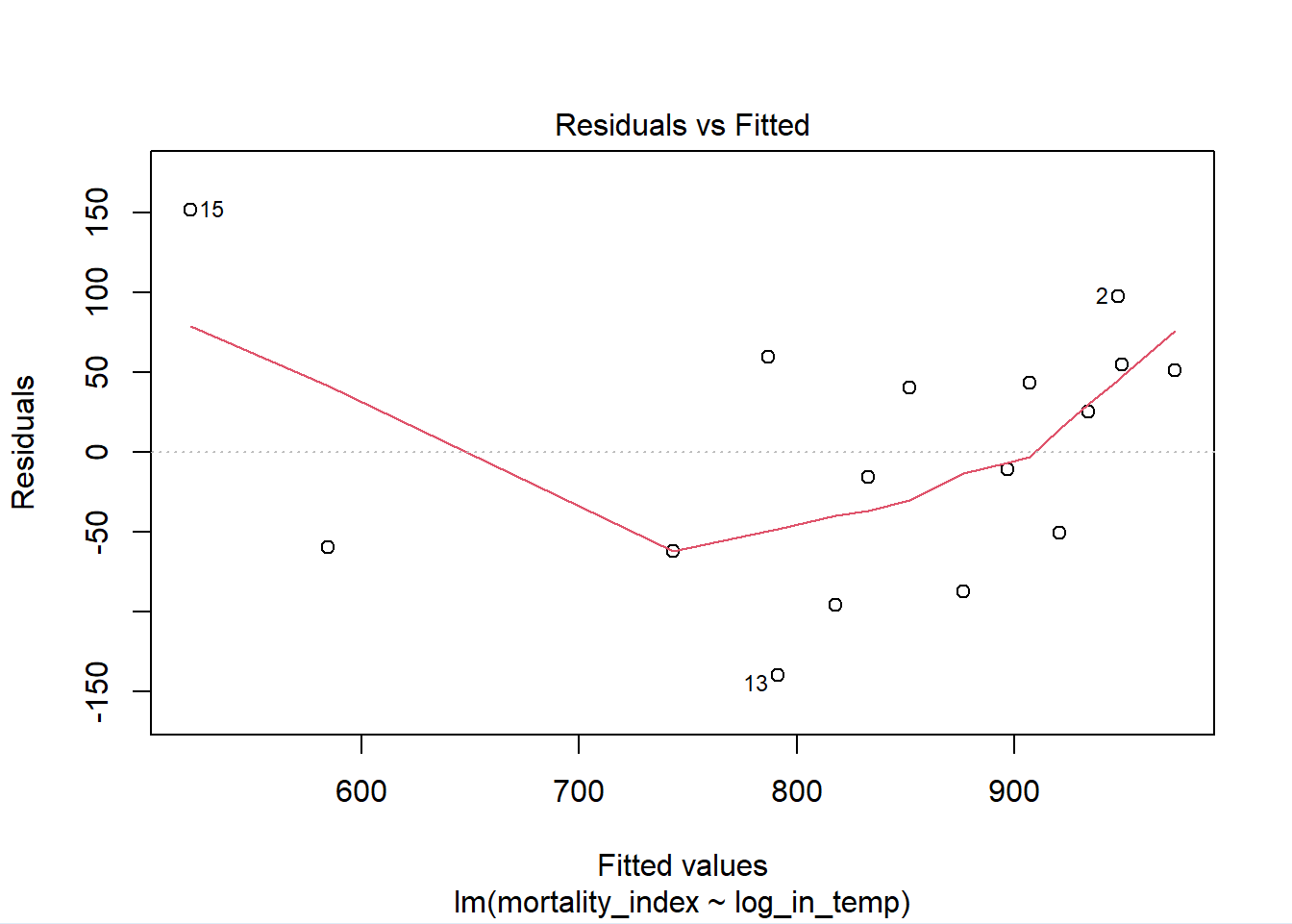
The main reason we gave was that it often made positive data more normal. Taking logs amounts to changing the units of the data in such a way that equal differences now mean equal multiplicative factors. This simplifies the interpretation of the measurement scale because addition is easier than multiplication. Some statisticians will go as far as to recommend log transforming positive data by default, though by the end of Cleveland's chapter 2, we'll see an example where that backfires.

```
log_in_temp <- log(mortality_file$temperature)


ggplot(mortality_file, aes(x = log_in_temp, y = mortality_index)) +
  geom_point() +
  labs(title = "Transformed Mortality Index vs Log of Mean Average Temperature",
       x = "Log of Mean Average Temperature",
       y = "Mortality Index")
```



Straight Line

```
model <- lm(mortality_index ~ log_in_temp, data = mortality_file)
plot(model, which = 1)
```

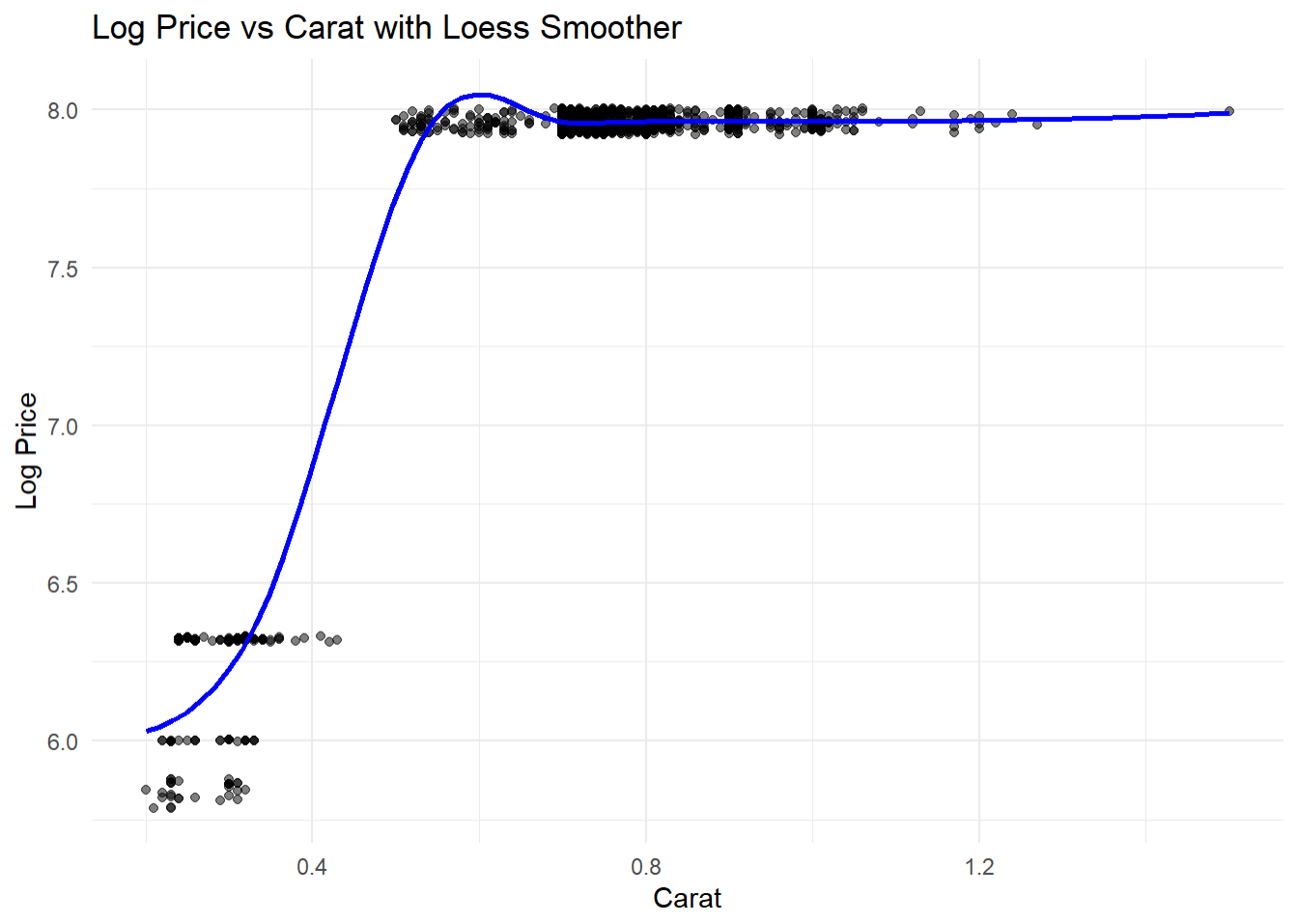


Analyzing the ploting, the index of temperature increases, therefore, **Hollow Up** .

2. Using the same subset of the diamonds dataset, make a plot of log price as a function of carat with a loess smoother. Try several values for the span and degree arguments and comment briefly about your choice.

```
library(ggplot2)
sample_diamonds <- diamonds[1:1500, ]
ggplot(sample_diamonds, aes(x = carat, y = log(price))) +
  geom_point(alpha = 0.5) +
```

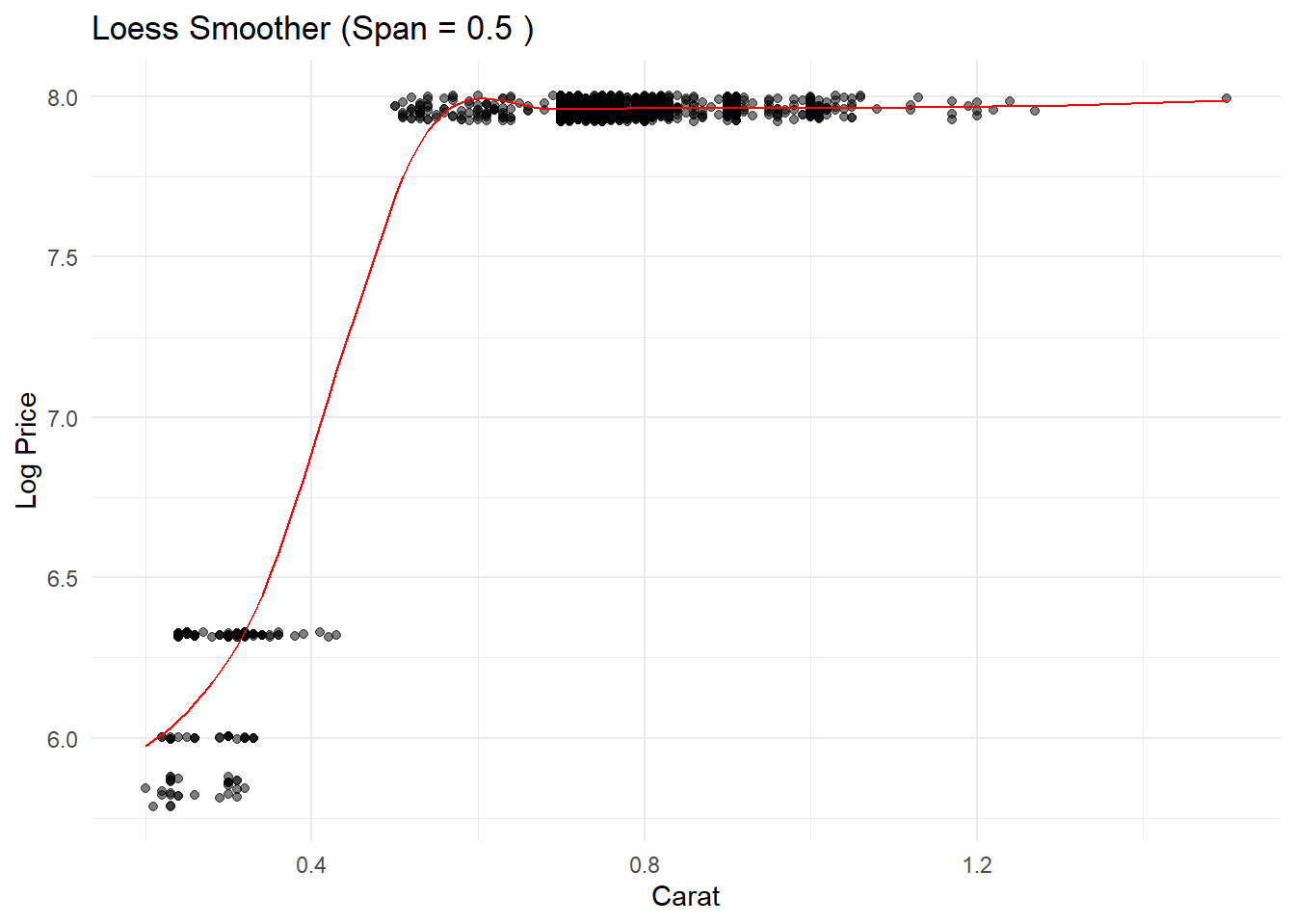
```
  geom_smooth(method = "loess", se = FALSE, span = 0.3, color = "blue") +
  labs(title = "Log Price vs Carat with Loess Smoother",
       x = "Carat",
       y = "Log Price") +
  theme_minimal()
## `geom_smooth()` using formula = 'y ~ x'
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : pseudoinverse used at 0.71
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : neighborhood radius 0.01
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : reciprocal condition number 0
```



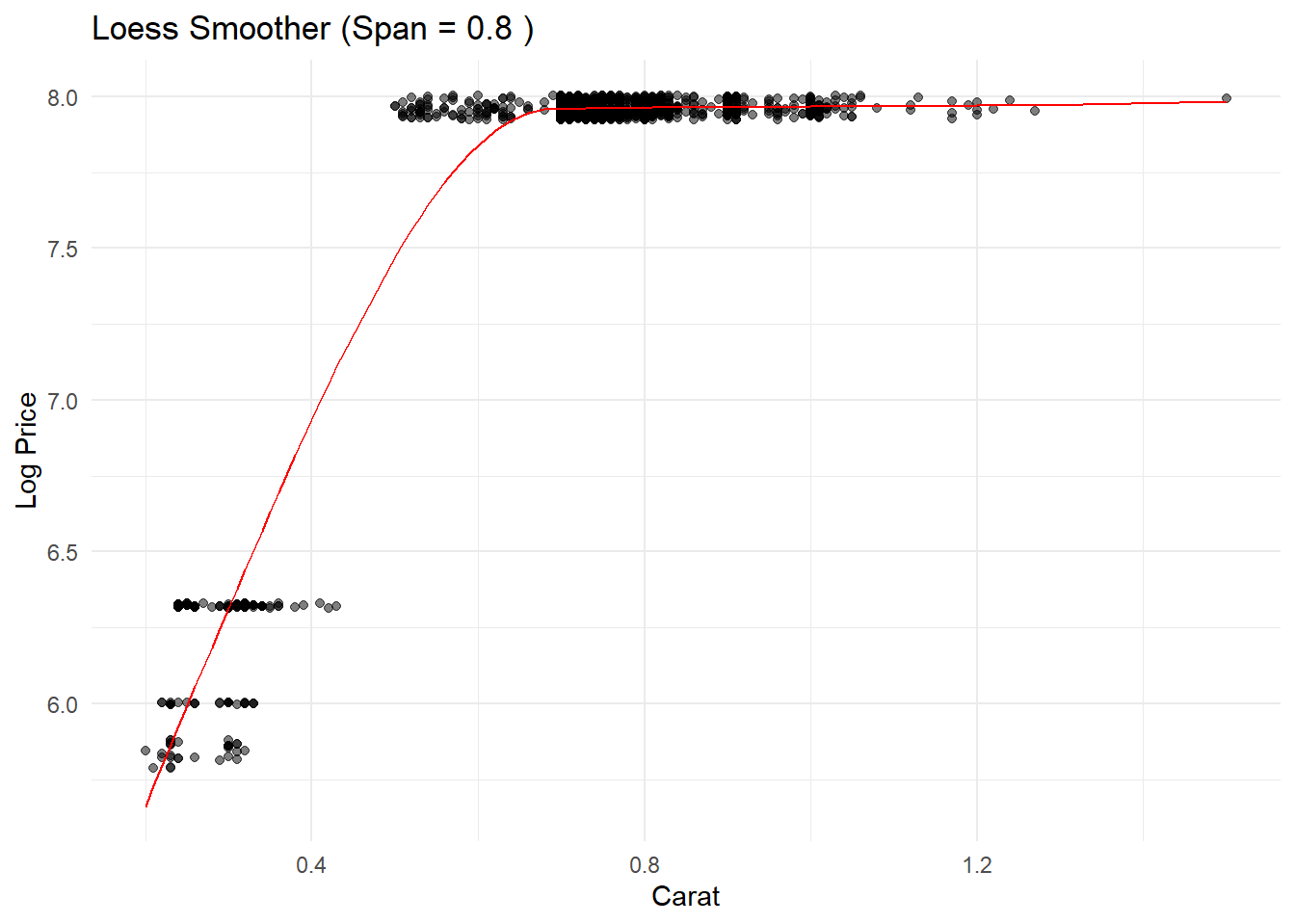
Trying Values 0.5,0.8,0.9

```
subset_diamonds <- diamonds[1:1500, ]
```

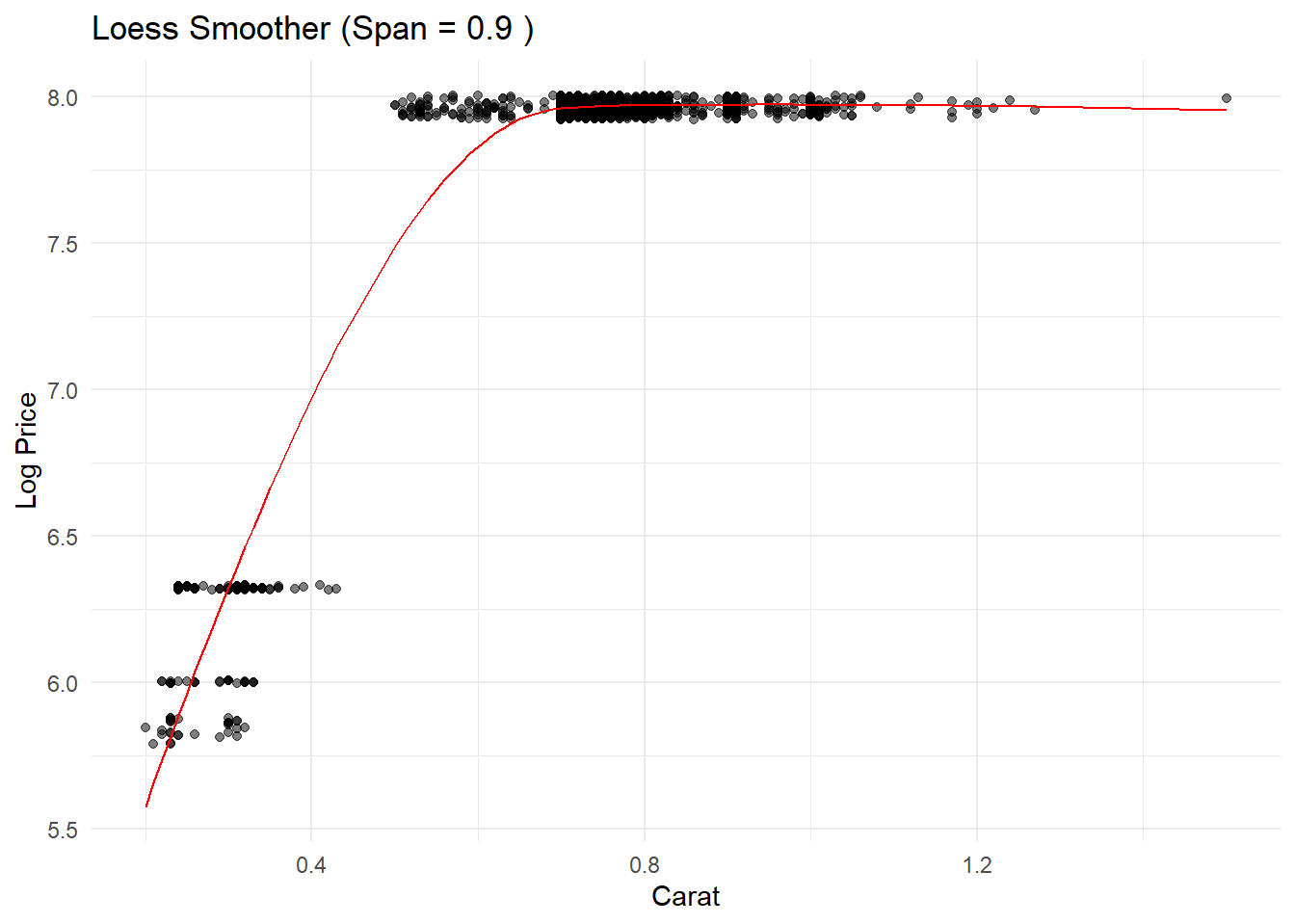
```
# Vector of values for span
span_values <- c( 0.5,0.8,0.9)


# Shortened code
for (span in span_values) {
  loess_model <- loess(log(price) ~ carat, data = subset_diamonds, span = span, degree = 2)
  loess_data <- cbind(subset_diamonds, predicted_values = predict(loess_model, newdata = data.frame(carat = subset_diamonds$carat)))
  p <- ggplot(loess_data, aes(x = carat, y = log(price))) +
    geom_point(alpha = 0.5) +
    geom_line(aes(y = predicted_values), color = "red") +
    labs(title = paste("Loess Smoother (Span =", span, ")"), x = "Carat", y = "Log Price") +
    theme_minimal()
  cat("For span =", span, ": Good balance between smoothing and capturing local patterns.\n")
  print(p)
}
## For span = 0.5 : Good balance between smoothing and capturing local patterns.
```




```
## For span = 0.8 : Good balance between smoothing and capturing local patterns.
```



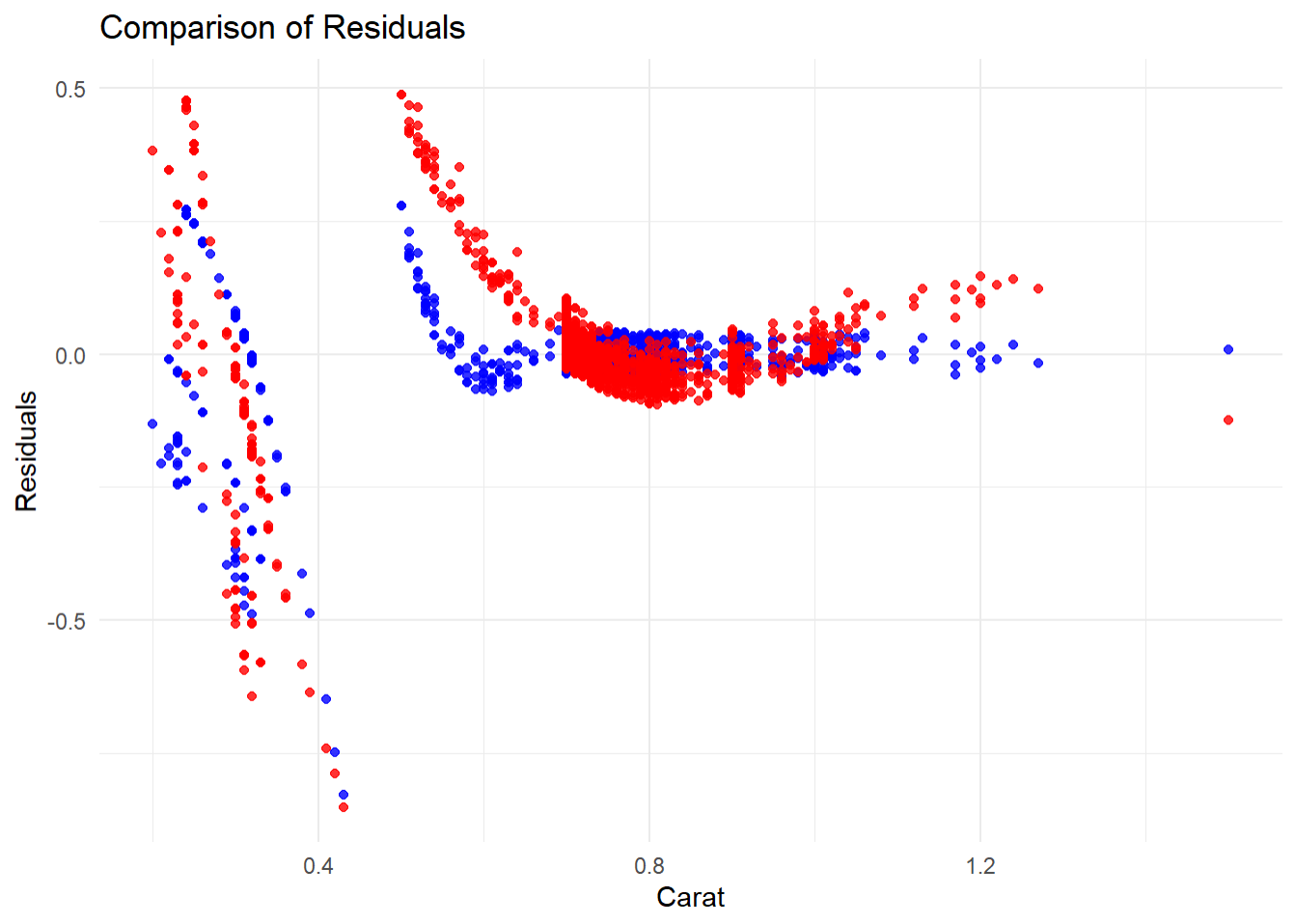

```
## For span = 0.9 : Good balance between smoothing and capturing local patterns.
```



3. Compare the fit of the loess smoother to the fit of the polynomial + step function regression using a plot of the residuals in the two models. Which one is more faithful to the data?

```
library(ggplot2)

subset_diamonds <- diamonds[1:1500, ]

loess_model <- loess(log(price) ~ carat, data = subset_diamonds, span = 0.5)

lm_model <- lm(log(price) ~ poly(carat, 3) + cut, data = subset_diamonds)
```

```r
subset_diamonds$residuals_loess <- residuals(loess_model)
subset_diamonds$residuals_lm <- residuals(lm_model)

ggplot(subset_diamonds, aes(x = carat)) +
  geom_point(aes(y = residuals_loess), color = "blue", alpha = 0.8) +
  geom_point(aes(y = residuals_lm), color = "red", alpha = 0.8) +
  labs(title = "Comparison of Residuals",
       x = "Carat",
       y = "Residuals") +
  theme_minimal()
```




```r
#
```

Analysis

As shown, the residual of for Loess Model for BLUE POINT and Poly Model for RED POINT do not spread together on some point, however as shown in the plotting **loess data** spreads more and constantly touches the 0.0 deviate which is a potential for a shortcomings, thus **more faithful**.