ping_analysis

Subject 4: Latency and capacity estimation for a network connection from asymmetric measurements

```
# install.packages("patchwork")
# install.packages("quantreg")
```

The original logs for this analysis were downloaded from these links:

http://mescal.imag.fr/membres/arnaud.legrand/teaching/2014/RICM4_EP_ping/liglab2.log.gz

http://mescal.imag.fr/membres/arnaud.legrand/teaching/2014/RICM4_EP_ping/stackoverflow.log.gz

```
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(patchwork)
library(quantreg)
```

```
## Loading required package: SparseM
```

Introduction

In this analysis, we will examine two datasets containing logs from pinging two different addresses. A simple and commonly used model for network performance assumes that the time T required to send a message depends primarily on its size S (in bytes) and two characteristics of the connection: the latency L (in seconds) and the capacity C(in bytes per second). We aim to estimate C and L for the network connections using linear regression.

Initial inspection

After an initial visual inspection, I noticed that not all log lines represent successful pings. Since failed pings are not relevant to our analysis, we begin by removing entries that do not conform to the expected structure.

```
lines <- readLines("liglab2.log", warn = FALSE)
paste("Original number of lines : ", length(lines))
```

```
## [1] "Original number of lines :  44413"
```

```r
lines <- lines[grepl("bytes from .* time=", lines)]
paste("Number of lines with successful pings: ", length(lines))
```

```
## [1] "Number of lines with successful pings:  44036"
```

Next, the remaining log lines are parsed to extract the information required for analysis and converted into a data frame with appropriate data types. Timestamps are stored as POSIXct, while ping latency (in milliseconds) and payload size are converted to numeric values. As before, any lines that do not represent successful pings are excluded.

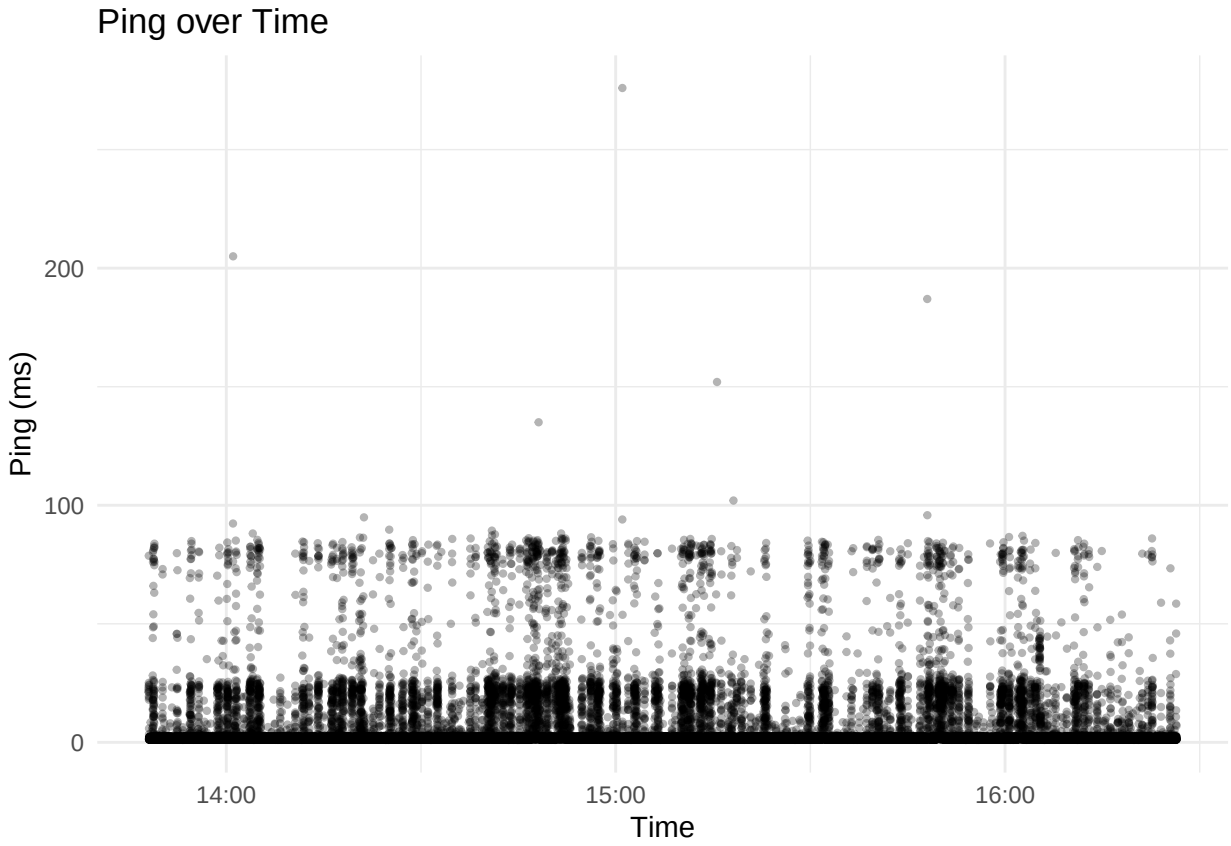
```r
parse_ping_log <- function(file) {
  lines <- readLines(file, warn = FALSE)

  lines <- lines[grepl("bytes from .* time=", lines)]

  timestamps <- as.numeric(sub("^\\[(\\d+\\.\\d+)\\].*", "\\1", lines))
  times <- as.POSIXct(timestamps, origin="1970-01-01", tz="UTC")
  size_bytes <- as.numeric(sub("^.*\\] ([0-9]+) bytes.*$", "\\1", lines))
  ping_ms <- as.numeric(sub("^.*time=([0-9.]+) ms.*$", "\\1", lines))

  data.frame(
    times = times,
    size_bytes = size_bytes,
    ping_ms = ping_ms,
    stringsAsFactors = FALSE
  )
}
```

```r
data <- parse_ping_log("liglab2.log")
str(data)
```

```
## 'data.frame':    44036 obs. of  3 variables:
##  $ times     : POSIXct, format: "2015-01-20 13:48:02" "2015-01-20 13:48:02" ...
##  $ size_bytes: num  665 1373 262 1107 1128 ...
##  $ ping_ms   : num  22.5 21.2 21.2 23.3 1.41 21.9 78.7 25.1 24 19.5 ...
```

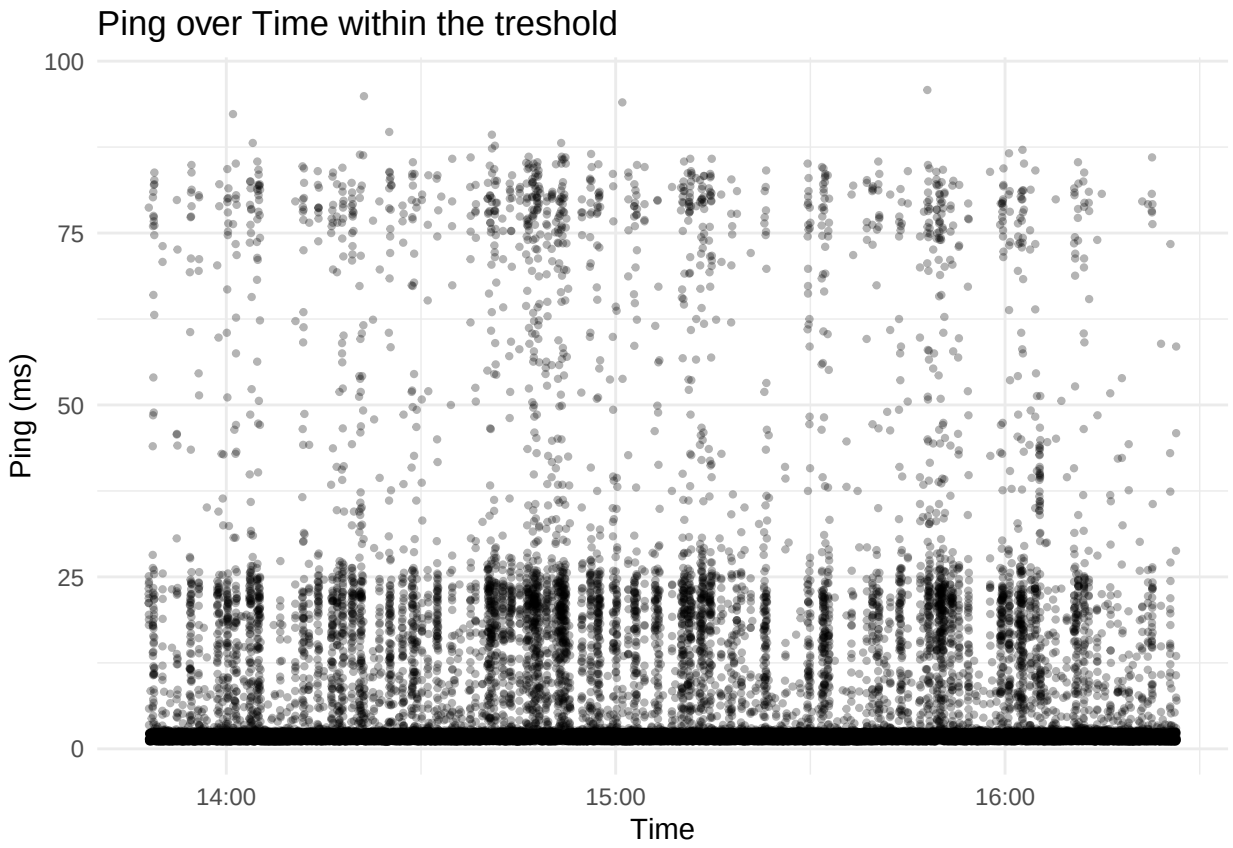
Now let's try to analyse the data visually by plotting it:

```r
ggplot(data, aes(x = times, y = ping_ms)) +
  geom_point(alpha = 0.3, size = 0.8) +
  labs(
    title = "Ping over Time",
    x = "Time",
    y = "Ping (ms)"
  ) +
  theme_minimal()
```



Several outliers are clearly visible, with the majority of ping times falling below 100 ms. To avoid influencing further results, values above this threshold are removed. The presence of these outliers may be attributed to various factors, most likely rapid and significant to network network conditions.

```r
data <- data[data$ping_ms <= 100, ]

ggplot(data, aes(x = times, y = ping_ms)) +
  geom_point(alpha = 0.3, size = 0.8) +
  labs(
    title = "Ping over Time within the treshold",
    x = "Time",
    y = "Ping (ms)"
  ) +
  theme_minimal()
```



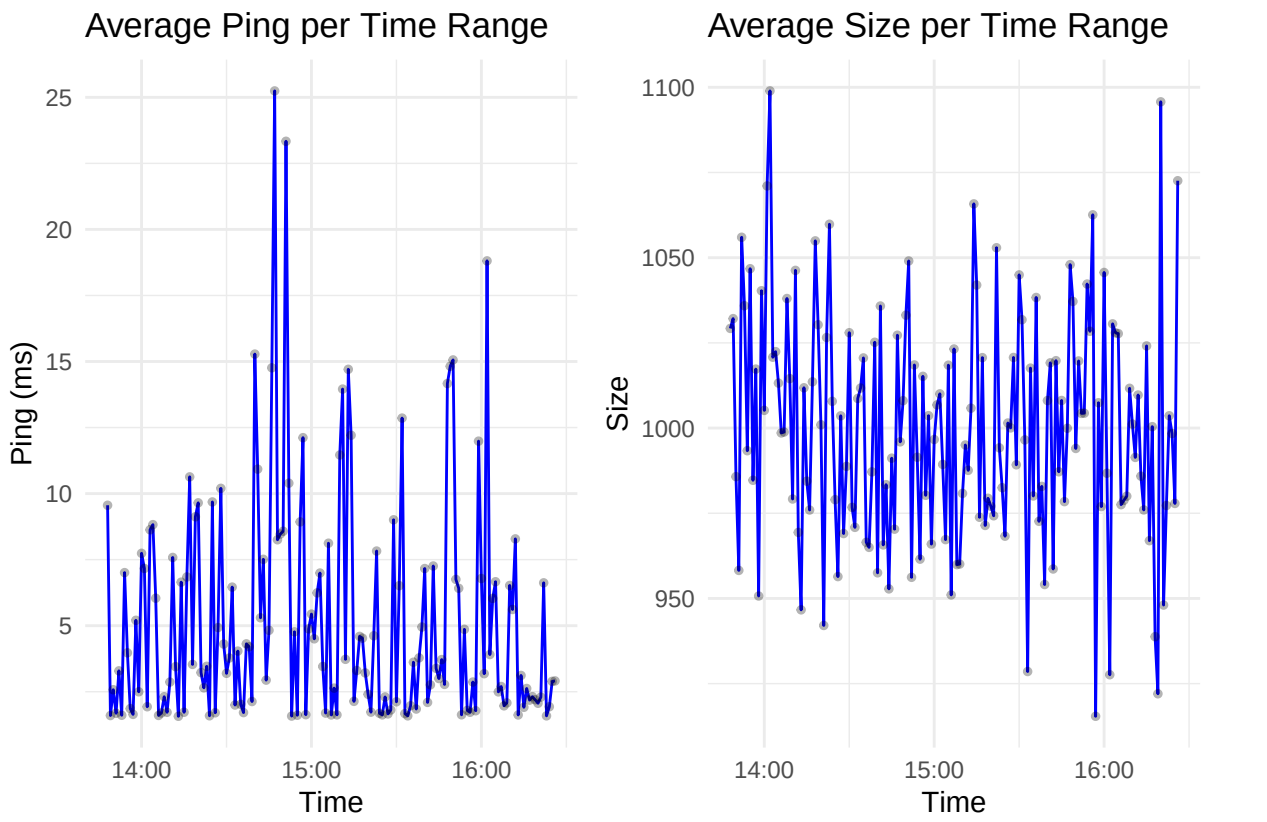
Evolution of transmission times in different time scales, and time ranges

In order to determine if the variations in transmission time can be explained only by the size of the payload, let's visualize the aggregated means of both transmission times and size of payload for specific timescales. I.e., let's visualize means for 20 min, 10 min, 1 min, etc.

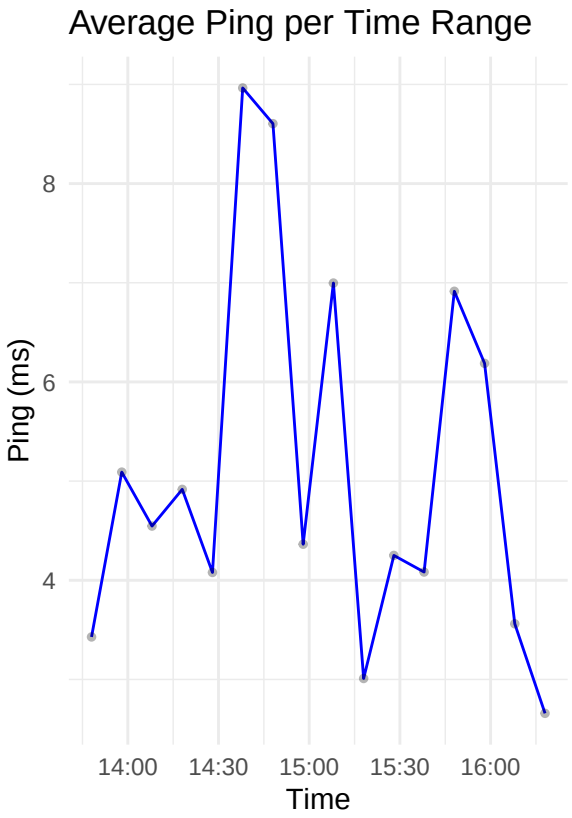
```
library(dplyr)
library(ggplot2)
library(patchwork)

plot_means_for_timerange <- function(data, timerange) {
  data_min <- data %>%
    mutate(time_min = as.POSIXct(cut(times, timerange))) %>%
    group_by(time_min) %>%
    summarise(avg_ping = mean(ping_ms, na.rm = TRUE),
              avg_size = mean(size_bytes, na.rm = TRUE))

  p1 <- ggplot(data_min, aes(x = time_min, y = avg_ping)) +
    geom_line(color = "blue") +
    geom_point(alpha = 0.3, size = 1) +
    labs(title = "Average Ping per Time Range",
         x = "Time",
         y = "Ping (ms)") +
    theme_minimal()


  p2 <- ggplot(data_min, aes(x = time_min, y = avg_size)) +
```

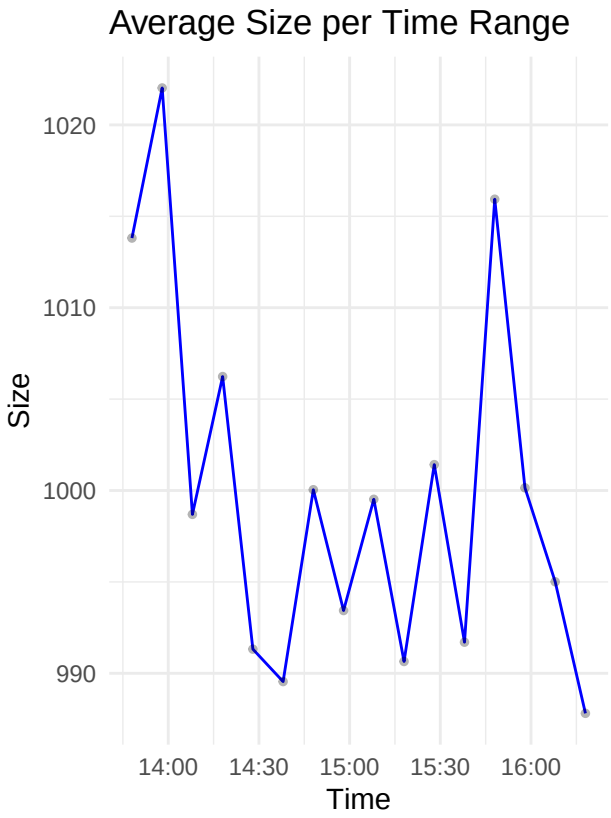
```
    geom_line(color = "blue") +
    geom_point(alpha = 0.3, size = 1) +
    labs(title = "Average Size per Time Range",
         x = "Time",
         y = "Size") +
    theme_minimal()

  p1 + p2
}
```

```
plot_means_for_timerange(data, "1 min")
```

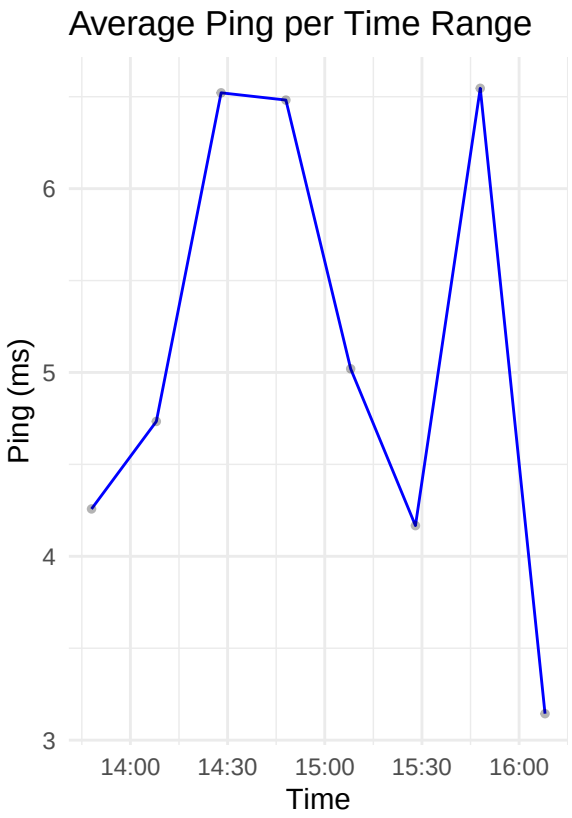


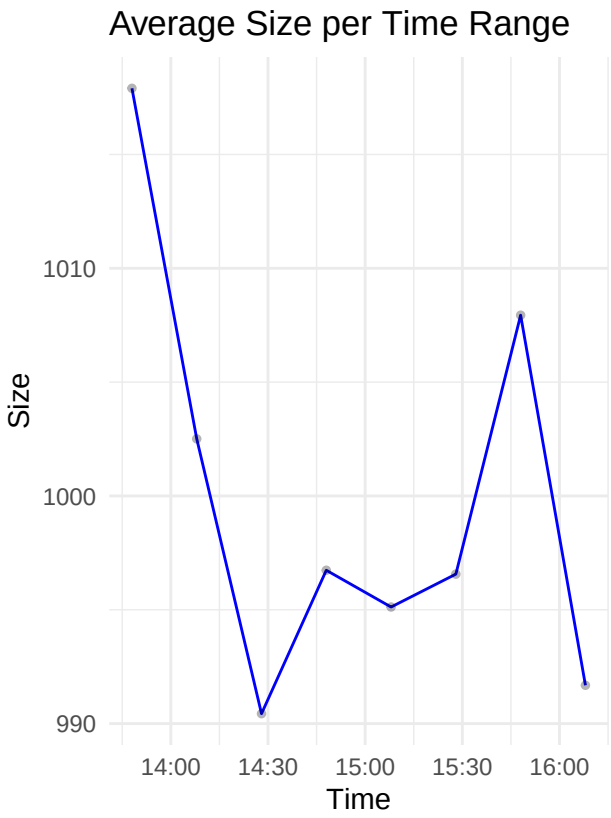

```
plot_means_for_timerange(data, "10 min")
```






```
plot_means_for_timerange(data, "20 min")
```





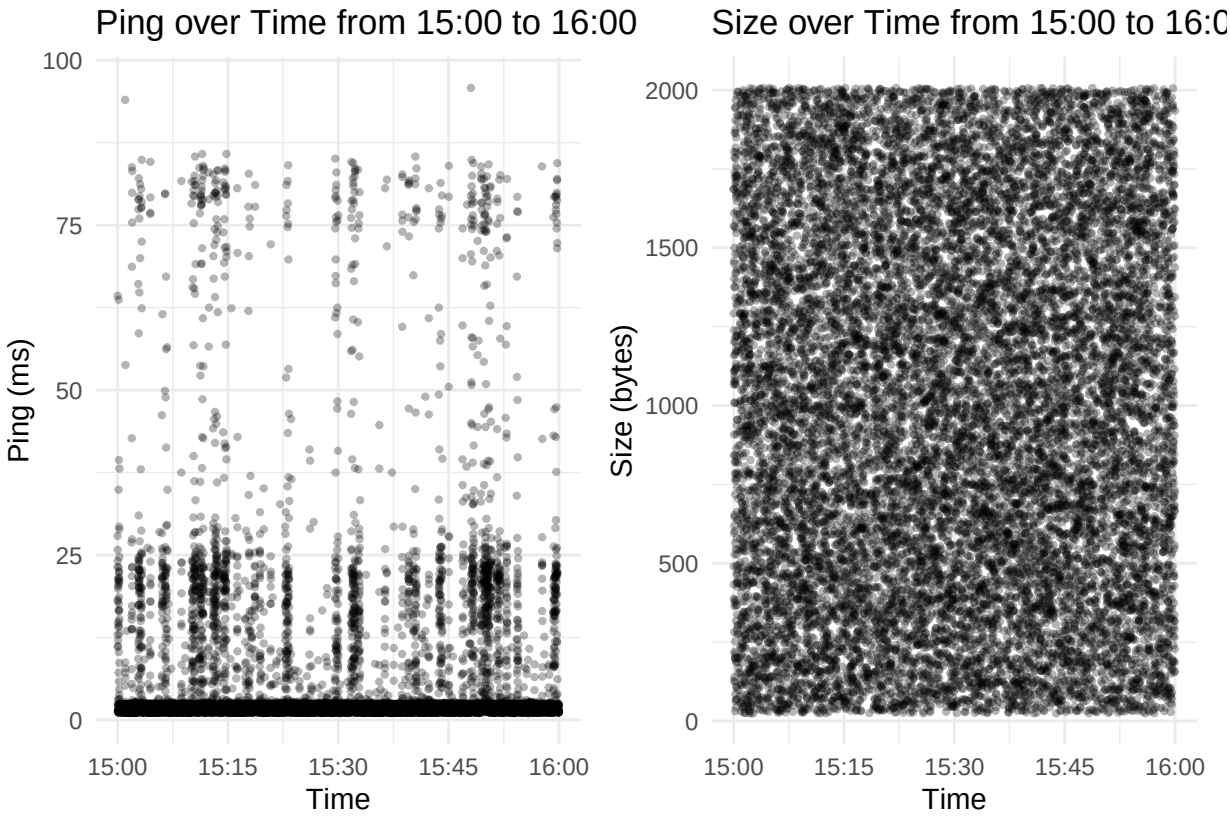
Based on the plots, we can see that transmission time cannot be determined only by the payload size, as we

see that spikes on the graphs do not visually seem to correlate with each other. More parameters for the model, such as time of day, could give better results, but since the dataset is limited to one specific day and no additional parameters are available, we cannot extend the model further.

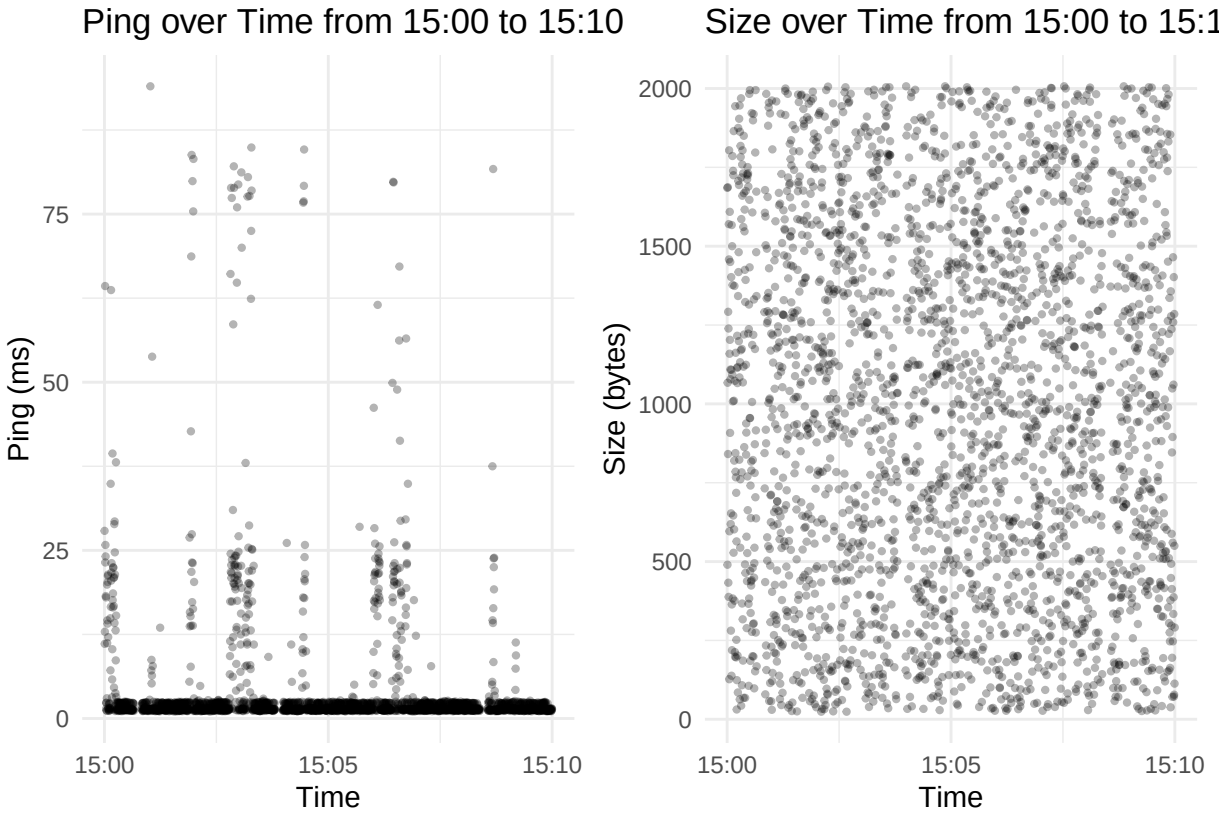
Now, let's try to plot the transmission times for specific shorter time spans to see if it also suggests that transmission time is not determined solely by payload size.

```
plot_for_timerange <- function(data, begin, end) {

  # Filter by time-of-day
  time_str <- format(data$times, "%H:%M")
  data_range <- data[time_str >= begin & time_str < end, ]

  p_ping <- ggplot(data_range, aes(x = times, y = ping_ms)) +
    geom_point(alpha = 0.3, size = 0.8) +
    labs(
      title = paste("Ping over Time from", begin, "to", end),
      x = "Time",
      y = "Ping (ms)"
    ) +
    theme_minimal()

  p_size <- ggplot(data_range, aes(x = times, y = size_bytes)) +
    geom_point(alpha = 0.3, size = 0.8) +
    labs(
      title = paste("Size over Time from", begin, "to", end),
      x = "Time",
      y = "Size (bytes)"
    ) +
    theme_minimal()

  p_ping + p_size
}
```

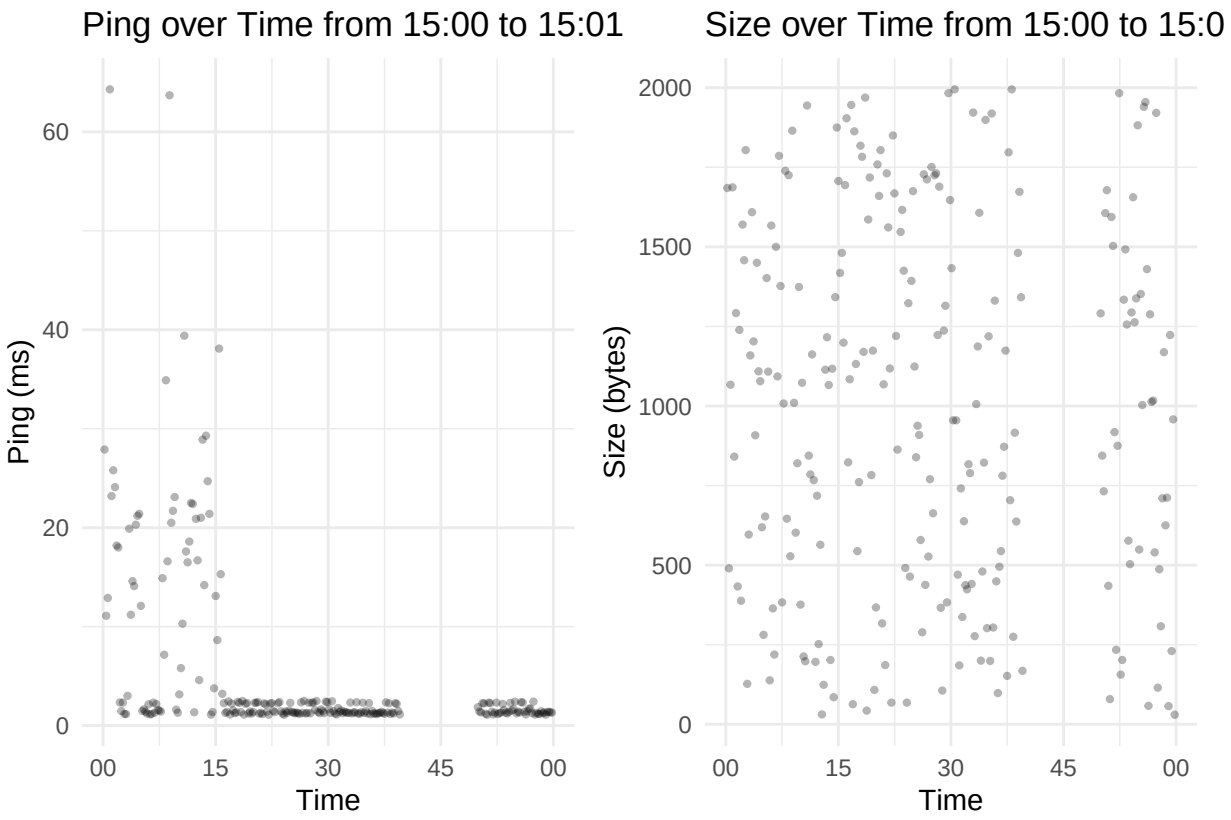
```
plot_for_timerange(data, "15:00", "16:00")
```




```
plot_for_timerange(data, "15:00", "15:10")
```



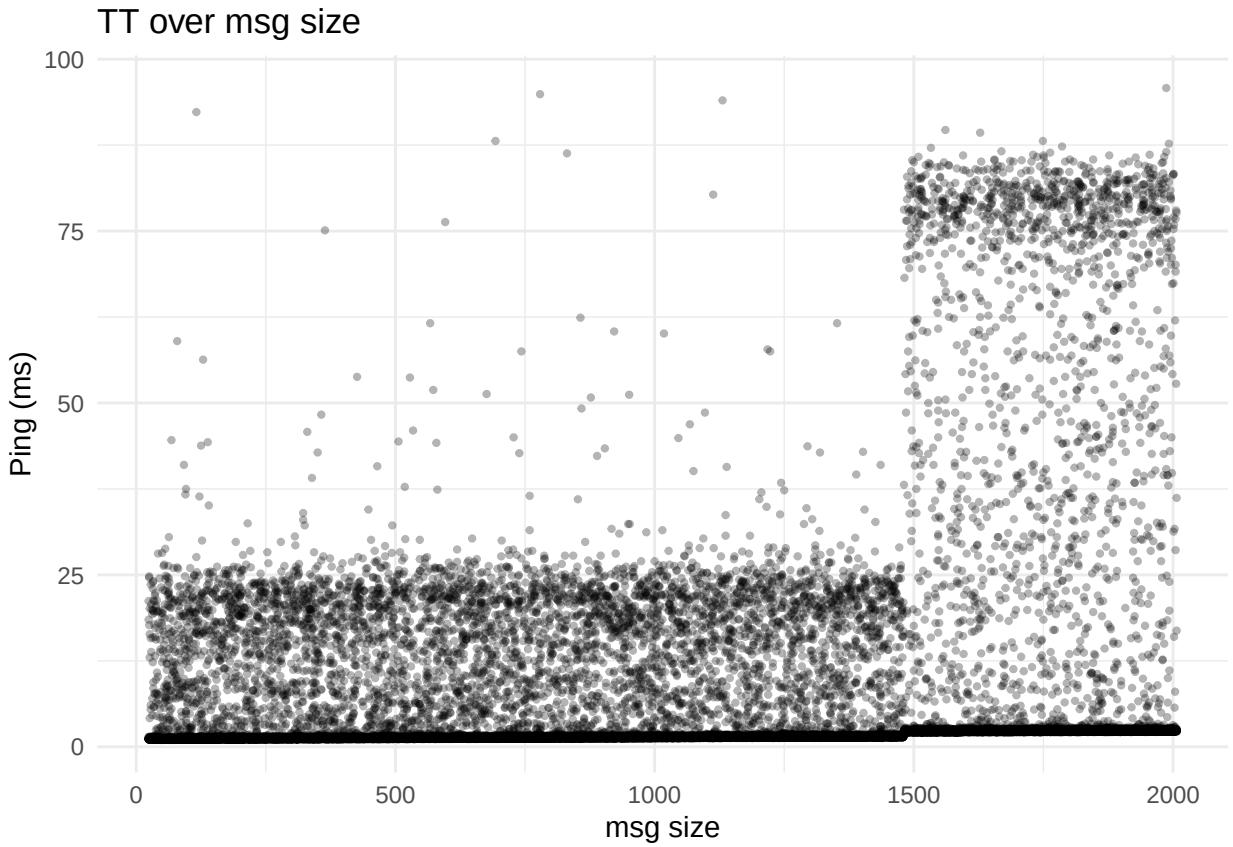
```
plot_for_timerange(data, "15:00", "15:01")
```



We see that even for shorter time spans, the distribution of payload sizes seems to be uniform, but the transmission times vary significantly between specific moments in time, with clear visual spikes in certain models. This also confirms that the model in which transmission time depends solely on payload size might be too simplified.

Transmission Time over the size of payload

```
ggplot(data, aes(x = size_bytes, y = ping_ms)) +
  geom_point(alpha = 0.3, size = 0.8) +
  labs(
    title = "TT over msg size",
    x = "msg size",
    y = "Ping (ms)"
  ) +
  theme_minimal()
```



We see that there is a clear difference between the distribution of transmission times for payloads larger than a specific point around 1500 bytes and for smaller payloads. After investigation, it is clear that this is caused by payload fragmentation. On a standard Ethernet network, the MTU (Maximum Transmission Unit) is 1500 bytes, and if any packet is larger, it will be fragmented, causing longer transmission times. The ping log shows the payload size without headers. The specific size of the ping payload that will divide the classes can be calculated as: 1500 - 20 (IP) - 8 (ICMP) = 1472 bytes. We split the dataset into two classes: one with payloads above 1472 - data_fragmented, and one with payloads below 1472 - data_unfragmented.

```
threshold <- 1472

data_unfragmented <- data[data$size_bytes <= threshold, ]
data_fragmented <- data[data$size_bytes > threshold, ]

str(data_fragmented)
```

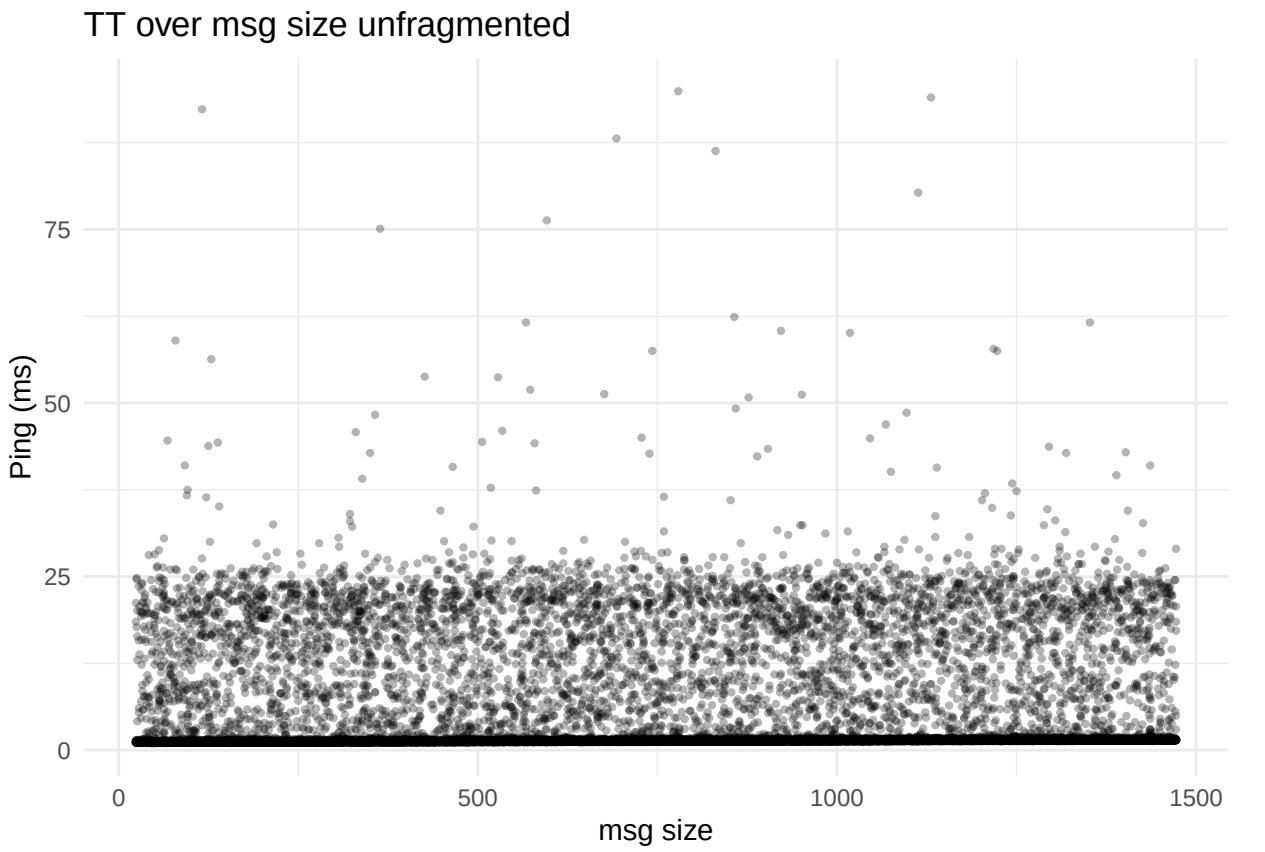
```
## 'data.frame':    11528 obs. of  3 variables:
##  $ times     : POSIXct, format: "2015-01-20 13:48:03" "2015-01-20 13:48:05" ...
##  $ size_bytes: num  1759 1843 1511 1510 1966 ...
##  $ ping_ms   : num  78.7 2.31 2.18 2.17 2.2 2.19 2.29 2.14 2.1 2.23 ...
```

```
str(data_unfragmented)
```

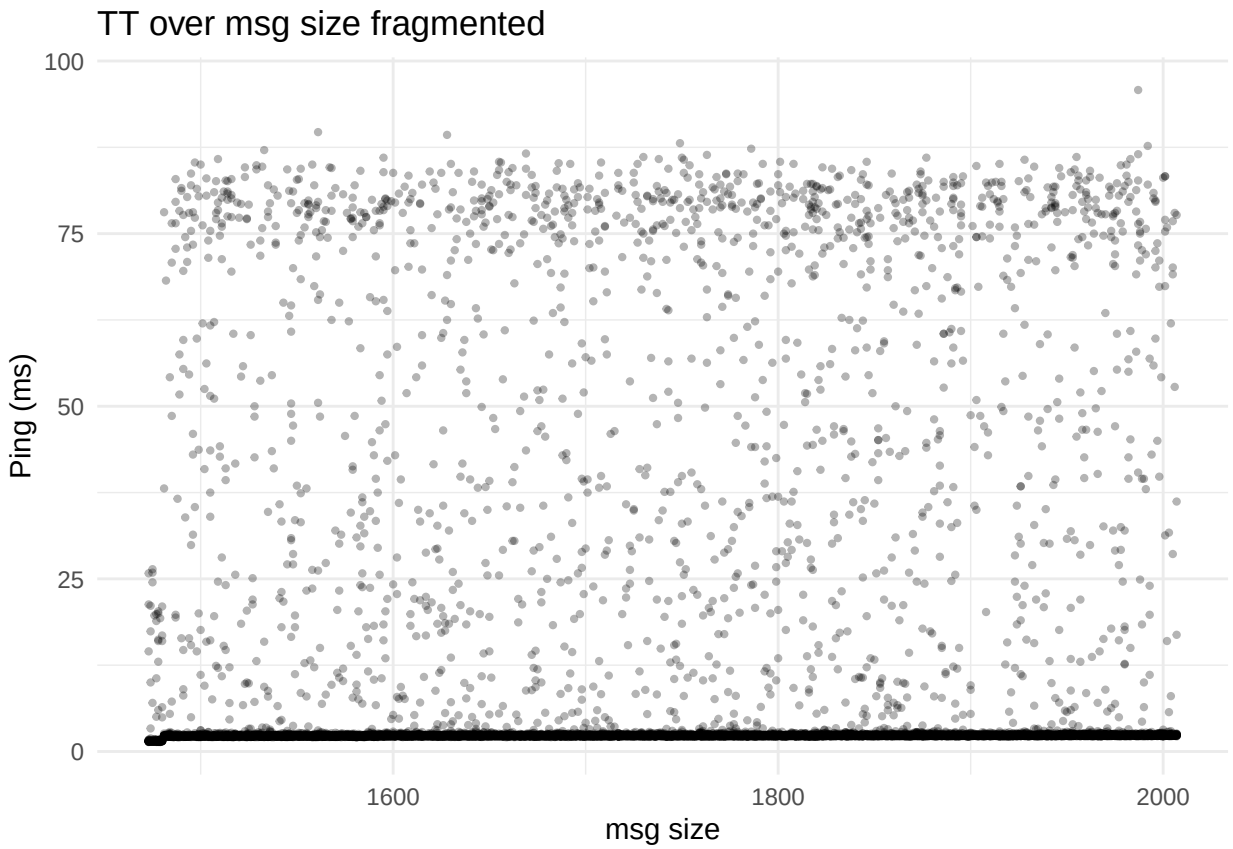
```
## 'data.frame':    32502 obs. of  3 variables:
##  $ times     : POSIXct, format: "2015-01-20 13:48:02" "2015-01-20 13:48:02" ...
##  $ size_bytes: num  665 1373 262 1107 1128 ...
##  $ ping_ms   : num  22.5 21.2 21.2 23.3 1.41 21.9 25.1 24 19.5 18 ...
```

```
ggplot(data_unfragmented, aes(x = size_bytes, y = ping_ms)) +
  geom_point(alpha = 0.3, size = 0.8) +
```

```
  labs(
    title = "TT over msg size unfragmented",
    x = "msg size",
    y = "Ping (ms)"
  ) +
  theme_minimal()
```




```
ggplot(data_fragmented, aes(x = size_bytes, y = ping_ms)) +
  geom_point(alpha = 0.3, size = 0.8) +
  labs(
    title = "TT over msg size fragmented",
    x = "msg size",
    y = "Ping (ms)"
  ) +
  theme_minimal()
```



After inspecting both plots, we see that now within one class, the distribution of transmission times seems to be more consistent.

Linear Regression

For both classes lets try to fit linear regression and calculate L and C, for vizualization of regression line we plot only Y axis to max value 5, to make it more readable.

```
summarize_regression <- function(data, maxY) {
  reg <- lm(data=data, ping_ms~size_bytes)
  summary(reg)

  coef_size <- coef(reg)["size_bytes"]
  inv_coef <- 1 / coef_size


  print(summary(reg))
  print(paste("C: ", inv_coef))
  print(paste("L: ", coef(reg)["(Intercept)"]))

  ggplot(data, aes(x = size_bytes, y = ping_ms)) +
    geom_point(alpha = 0.5) +
    geom_smooth(method = "lm", se = TRUE, color = "blue", fill = "lightblue", alpha = 0.3) +
    labs(
      title = "Ping vs Payload Size with Regression Line",
      x = "Payload Size (bytes)",
      y = "Ping (ms)"
    ) +
```

```
    ylim(0, maxY) +
    theme_minimal()
}
```

```
summarize_regression(data_unfragmented, 5)
```

```
##
## Call:
## lm(formula = ping_ms ~ size_bytes, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.456 -2.226 -2.168 -2.067 91.390
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.254e+00  6.874e-02  47.339  < 2e-16 ***
## size_bytes  3.281e-04  8.117e-05   4.043  5.3e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.097 on 32500 degrees of freedom
## Multiple R-squared:  0.0005026,  Adjusted R-squared:  0.0004718
## F-statistic: 16.34 on 1 and 32500 DF,  p-value: 5.298e-05
##
## [1] "C:  3047.38424466126"
## [1] "L:  3.25388045145047"
```

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

```
## Warning: Removed 4357 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 4357 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

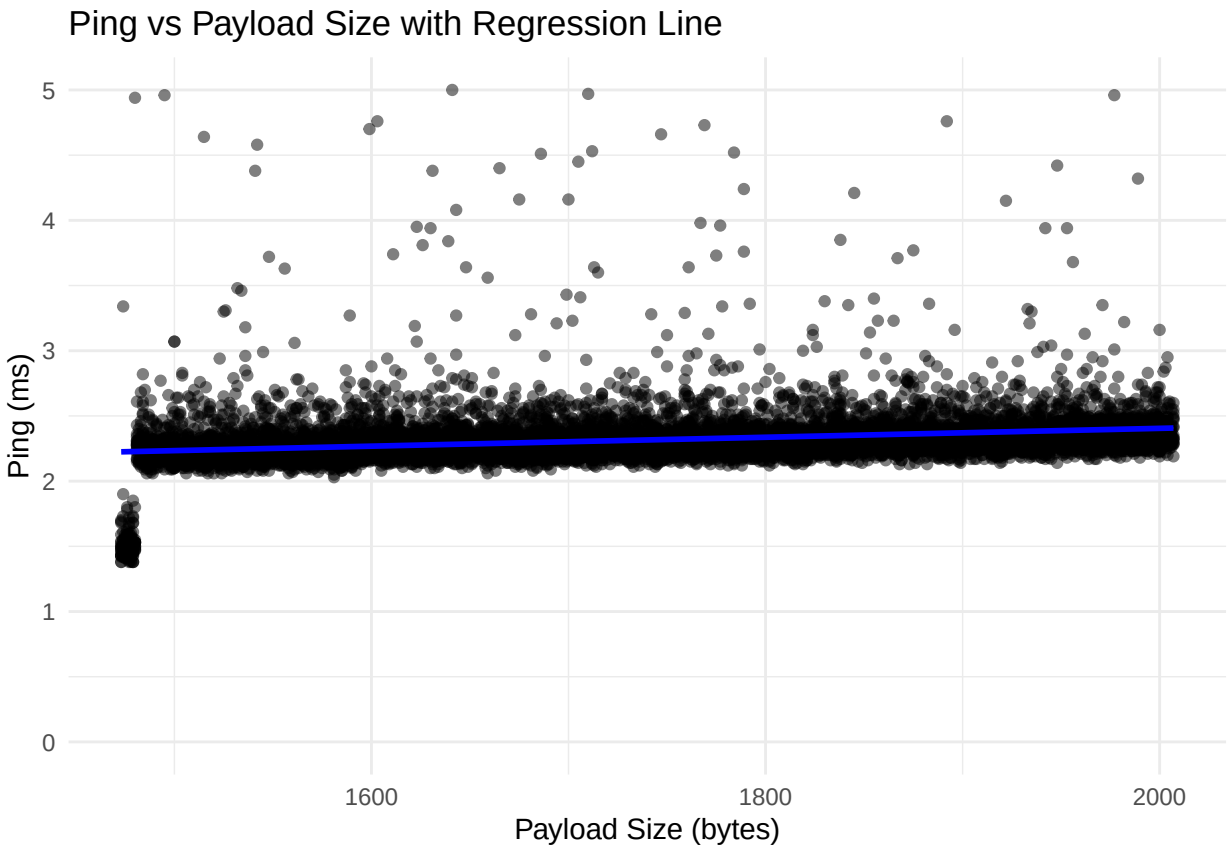

```
summarize_regression(data_fragmented, 5)
```

```
##
## Call:
## lm(formula = ping_ms ~ size_bytes, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.368 -7.693 -7.221 -6.753 85.309
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.891681   2.167113   1.796   0.0726 .
## size_bytes  0.003321   0.001240   2.679   0.0074 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.51 on 11526 degrees of freedom
## Multiple R-squared:  0.0006221,  Adjusted R-squared:  0.0005354
## F-statistic: 7.175 on 1 and 11526 DF,  p-value: 0.007404
##
## [1] "C:  301.083002019366"
## [1] "L:  3.89168091761965"
```

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

```
## Warning: Removed 1706 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 1706 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



In both cases, from the regression summary we can see that the model does not fit the data well. By analyzing the residuals, we observe that for both classes the model mostly underpredicts: the residual quantiles are around -2 for the unfragmented class and around -7 for the fragmented class. The maximum overprediction reaches approximately 90, suggesting the presence of outliers that are not explained by payload size.

Looking at the R-squared values, we see they are 0.006 and 0.005 — extremely small for both classes. This indicates that payload size explains almost none of the variation in transmission times.

The variability not related to payload size is so strong and asymmetric that a simple linear regression does not fit properly, and the estimated parameter values are likely unreliable. This analysis can be extended further.

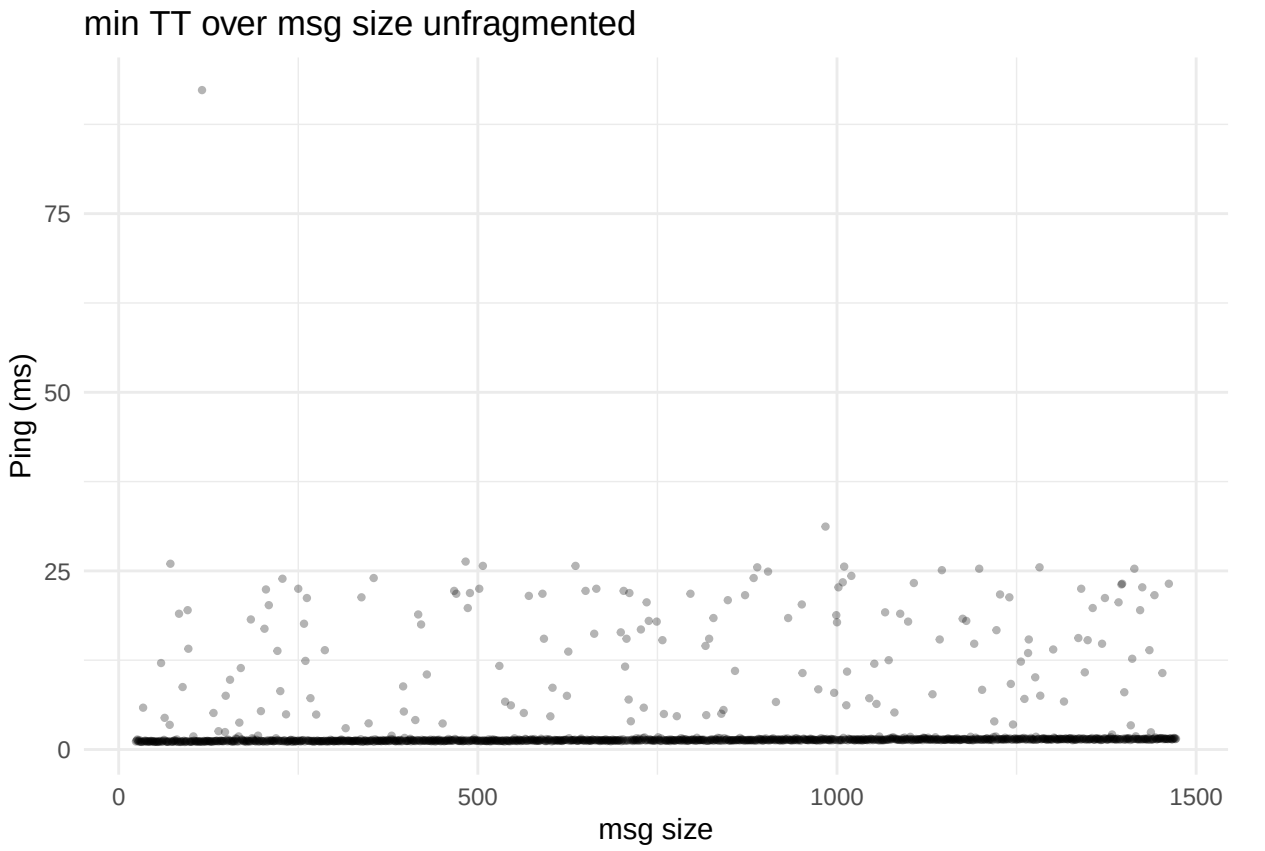
Different approaches to regression

As the variability is so strong and asymmetric that the normal linear regression does not fir properly. We could consider only the smallest times for each payload size and perform linear regression only on the subset of dataset.

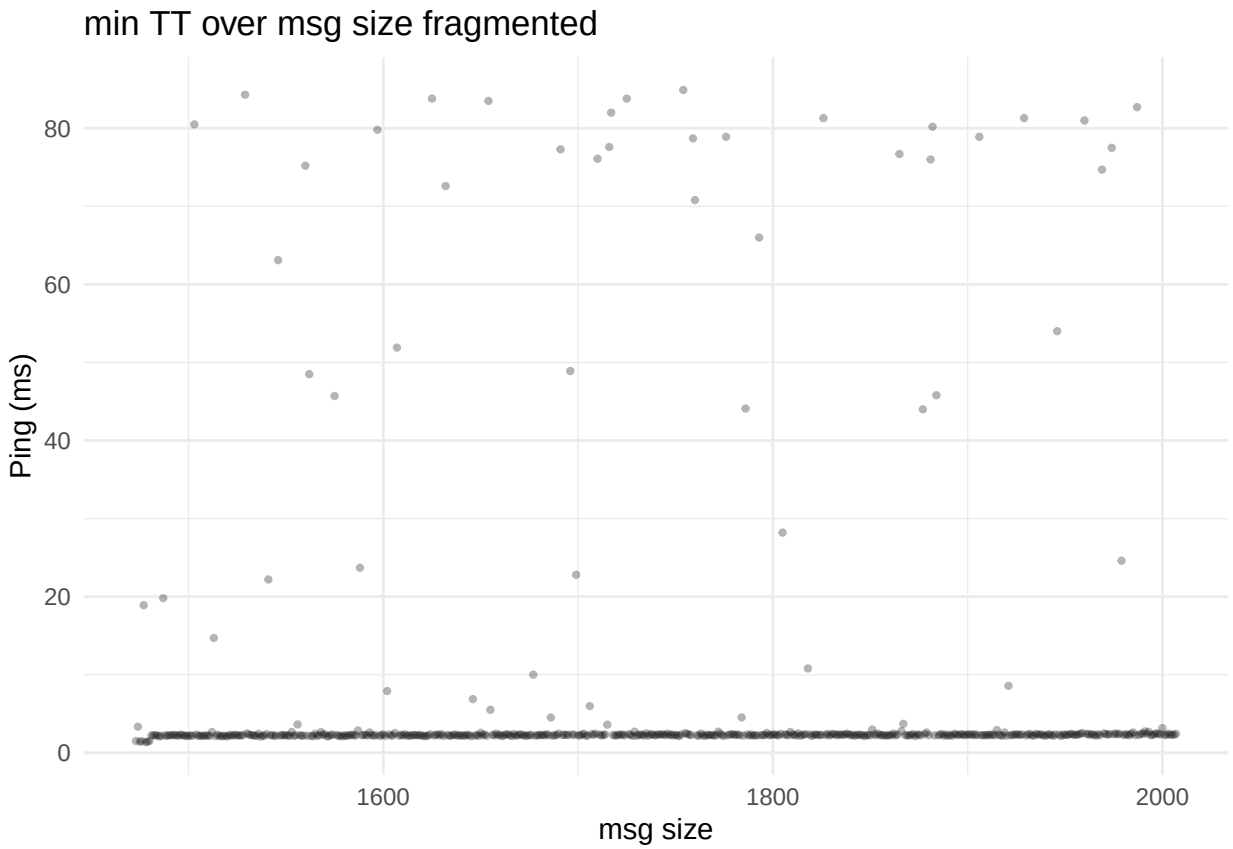
```
data_unfragmented_min <- data_unfragmented %>%
  group_by(size_bytes) %>%
  slice_min(times, n = 1) %>%
  ungroup()


data_fragmented_min <- data_fragmented %>%
  group_by(size_bytes) %>%
  slice_min(times, n = 1) %>%
```

```
  ungroup()
```

```
ggplot(data_unfragmented_min, aes(x = size_bytes, y = ping_ms)) +
  geom_point(alpha = 0.3, size = 0.8) +
  labs(
    title = "min TT over msg size unfragmented",
    x = "msg size",
    y = "Ping (ms)"
  ) +
  theme_minimal()
```




```
ggplot(data_fragmented_min, aes(x = size_bytes, y = ping_ms)) +
  geom_point(alpha = 0.3, size = 0.8) +
  labs(
    title = "min TT over msg size fragmented",
    x = "msg size",
    y = "Ping (ms)"
  ) +
  theme_minimal()
```



Lets try to fit linear regression here:

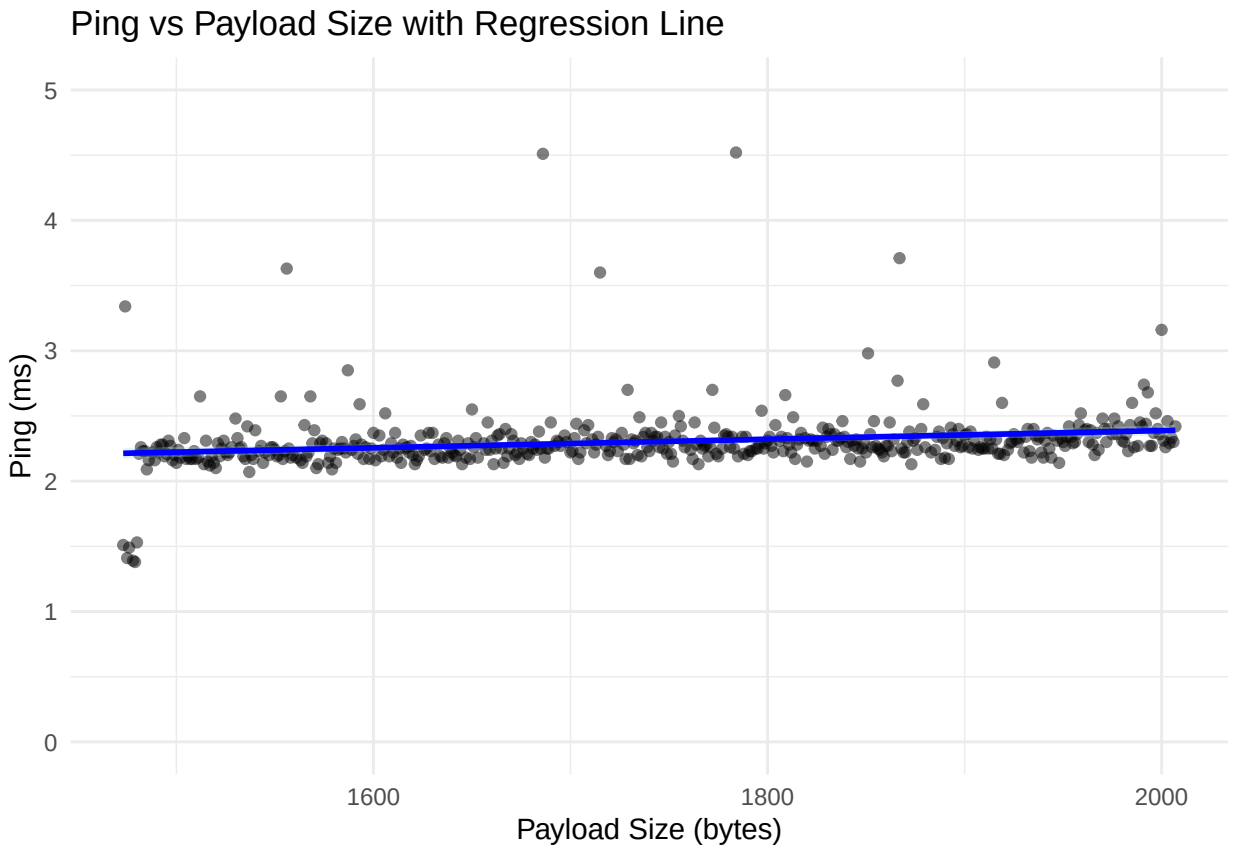
```
summarize_regression(data_fragmented_min, 5)
```

```
##
## Call:
## lm(formula = ping_ms ~ size_bytes, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.709 -5.310 -4.953 -4.635 77.544
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.020152   8.696537   0.347    0.729
## size_bytes  0.002472   0.004978   0.497    0.620
##
## Residual standard error: 17.78 on 533 degrees of freedom
## Multiple R-squared:  0.0004623,  Adjusted R-squared:  -0.001413
## F-statistic: 0.2465 on 1 and 533 DF,  p-value: 0.6197
##
## [1] "C:  404.559947271093"
## [1] "L:  3.02015198082882"
```

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

```
## Warning: Removed 51 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 51 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



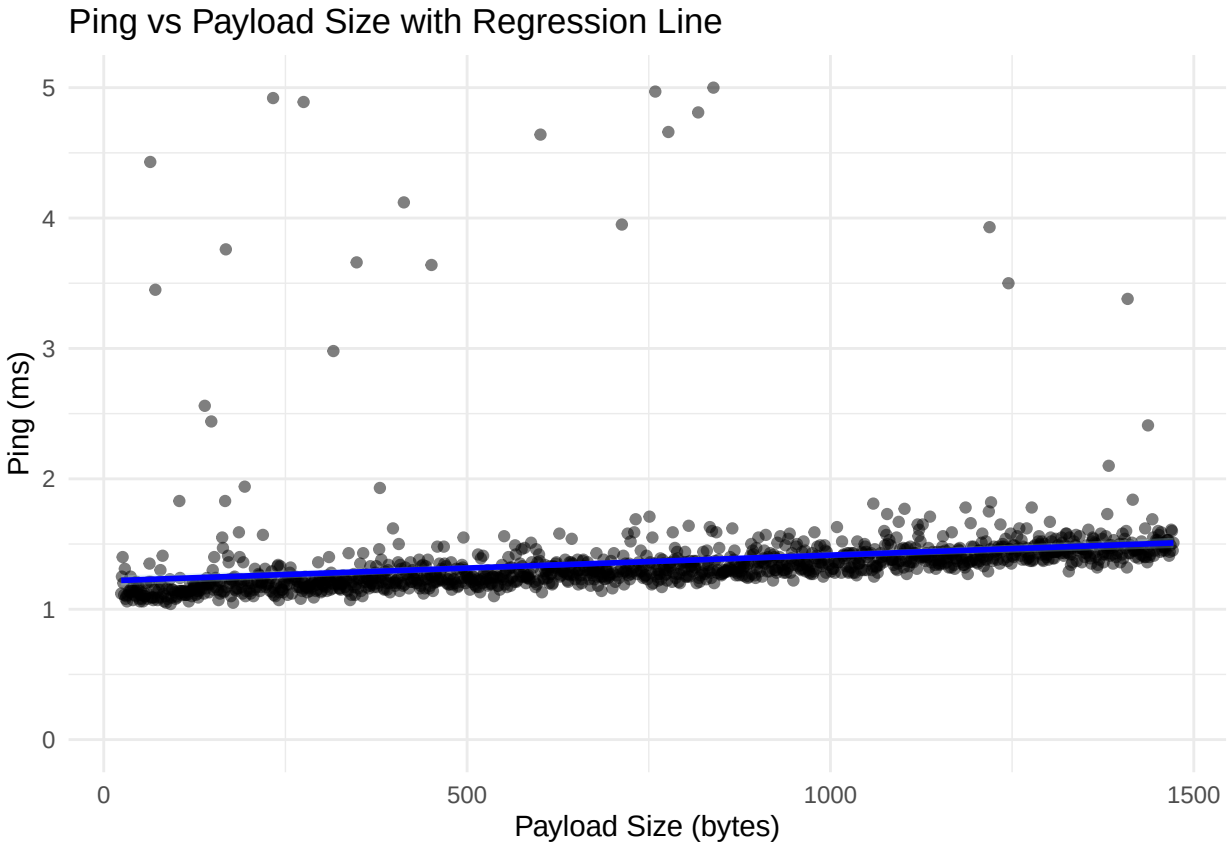

```
summarize_regression(data_unfragmented_min, 5)
```

```
##
## Call:
## lm(formula = ping_ms ~ size_bytes, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.933 -1.657 -1.499 -1.345 89.826
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.4043827  0.2884012   8.337   <2e-16 ***
## size_bytes  0.0006026  0.0003365   1.791   0.0735 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.358 on 1447 degrees of freedom
## Multiple R-squared:  0.002211,   Adjusted R-squared:  0.001522
## F-statistic: 3.207 on 1 and 1447 DF,  p-value: 0.07354
##
## [1] "C:  1659.37350095321"
## [1] "L:  2.40438274422379"
```

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

```
## Warning: Removed 141 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 141 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



After analyzing the regression on this subset, we can see that the fit is improved in this case. For both classes, the residuals are smaller, with quantiles around -1.5 for unfragmented and -4.9 for fragmented data. The model still tends to underpredict, but the performance is better overall. The R-squared values have also improved for both classes—especially for unfragmented, where it is around 0.002, compared to approximately 0.0004 for fragmented. Although these values are still quite small, the increase in variance explained suggests that in this case, payload size accounts for more of the variance in transmission times

Quantile Regression

Quantile regression is a regression method that works even when the errors are skewed and not normally distributed, as is the case here. Previously, using traditional linear regression, we tended to underestimate the response variable. Quantile regression is used to estimate the conditional quantiles of Y given X.

For a specific quantile tau, it fits a line that predicts that quantile of the response. For example, for tau=0.5(the median), over- and under-predictions are weighted equally — making it a robust version of linear regression.

For other quantiles, the weighting shifts: the 0.25 quantile gives more weight to over-predictions, while the 0.75 quantile gives more weight to under-predictions.

Since our previous models tended to underpredict, we might see a better fit for lower quantiles, for example, tau=0.25.

```r
summarize_qregression <- function(data, tau = 0.5, maxY) {
  reg <- rq(ping_ms ~ size_bytes, data = data, tau = tau)

  coef_size <- coef(reg)["size_bytes"]
  inv_coef <- 1 / coef_size

  print(summary(reg))
  cat("C (1/coef_size):", inv_coef, "\n")
  cat("L (intercept):", coef(reg)["(Intercept)"], "\n")

  residuals <- data$ping_ms - predict(reg, newdata = data)

  res_quantiles <- quantile(residuals, probs = c(0, 0.25, 0.5, 0.75, 1))
  cat("Residual quantiles (min, 25%, median, 75%, max):\n")
  print(res_quantiles)

  new_data <- data.frame(size_bytes = seq(min(data$size_bytes),
                                          max(data$size_bytes), length.out = 100))
  new_data$ping_pred <- predict(reg, newdata = new_data)

  ggplot(data, aes(x = size_bytes, y = ping_ms)) +
    geom_point(alpha = 0.5) +
    geom_line(data = new_data, aes(x = size_bytes, y = ping_pred),
              color = "red", size = 1) +
    labs(
      title = paste("Ping vs Payload Size with Quantile Regression (tau =", tau, ")"),
      x = "Payload Size (bytes)",
      y = "Ping (ms)"
    ) +
    ylim(0, maxY) +
    theme_minimal()
}
```

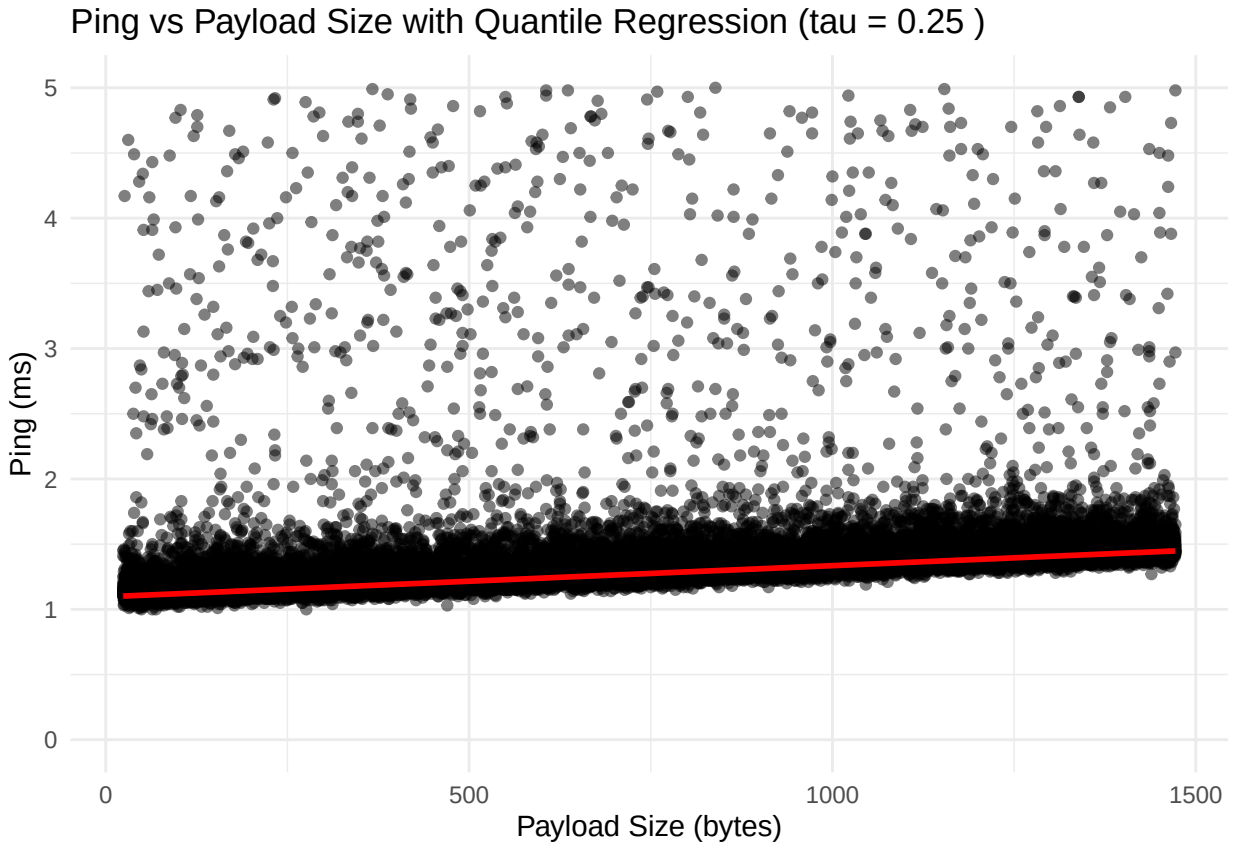
```r
summarize_qregression(data_unfragmented, 0.25, 5)
```

```
##
## Call: rq(formula = ping_ms ~ size_bytes, tau = tau, data = data)
##
## tau: [1] 0.25
##
## Coefficients:
##             Value      Std. Error t value    Pr(>|t|)
## (Intercept)    1.09598    0.00098 1118.81830    0.00000
## size_bytes     0.00024    0.00000  204.14578    0.00000
## C (1/coef_size): 4183.333
## L (intercept): 1.095976
## Residual quantiles (min, 25%, median, 75%, max):
##          0%         25%         50%         75%        100%
## -0.17832669  0.00000000  0.04900398  0.14799801 93.61780876
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
```

```
## generated.
```

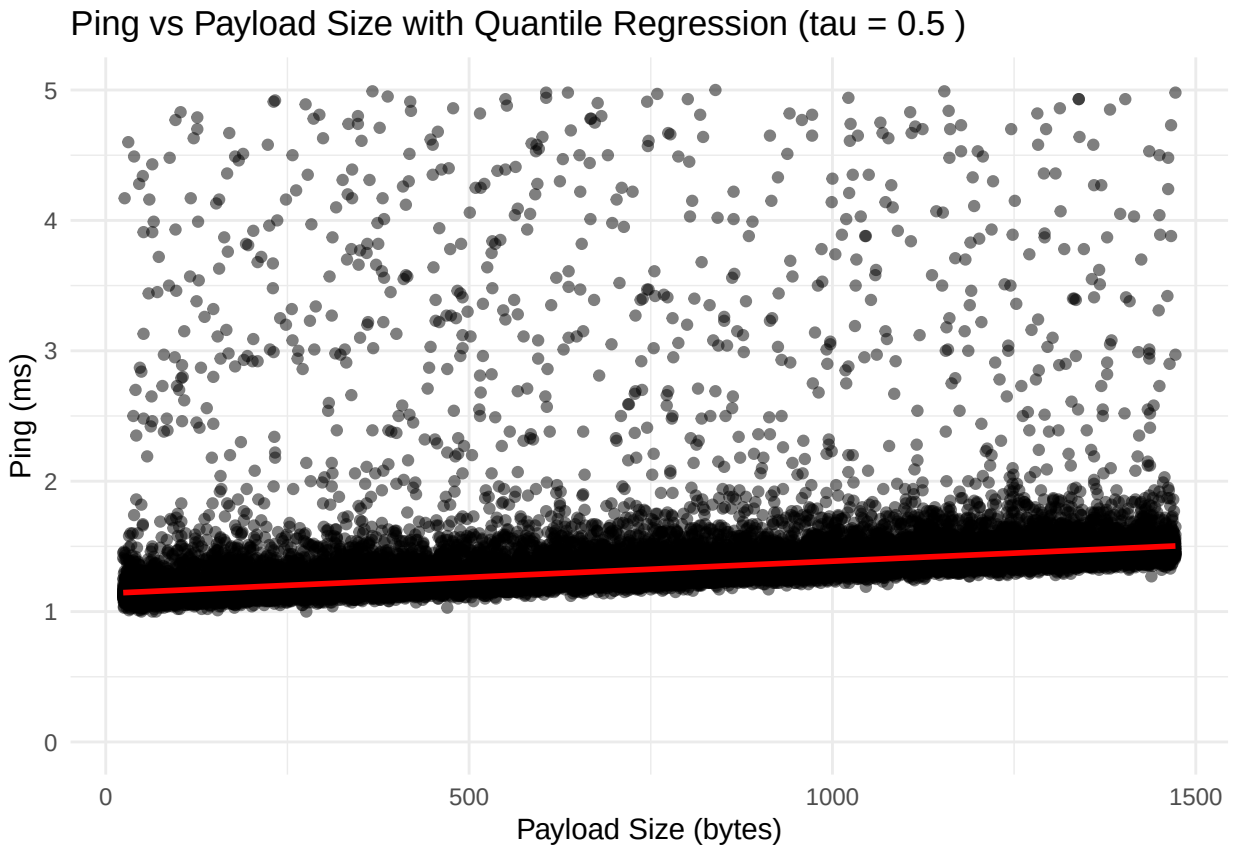
```
## Warning: Removed 4357 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
summarize_qregression(data_unfragmented, 0.50, 5)
```

```
##
## Call: rq(formula = ping_ms ~ size_bytes, tau = tau, data = data)
##
## tau: [1] 0.5
##
## Coefficients:
##             Value     Std. Error t value   Pr(>|t|)
## (Intercept)   1.13918   0.00120  945.88413   0.00000
## size_bytes    0.00025   0.00000  169.35078   0.00000
## C (1/coef_size): 4045
## L (intercept): 1.139184
## Residual quantiles (min, 25%, median, 75%, max):
##            0%           25%           50%           75%          100%
## -2.253770e-01 -4.925216e-02  2.220446e-16  9.876700e-02  9.356823e+01
```

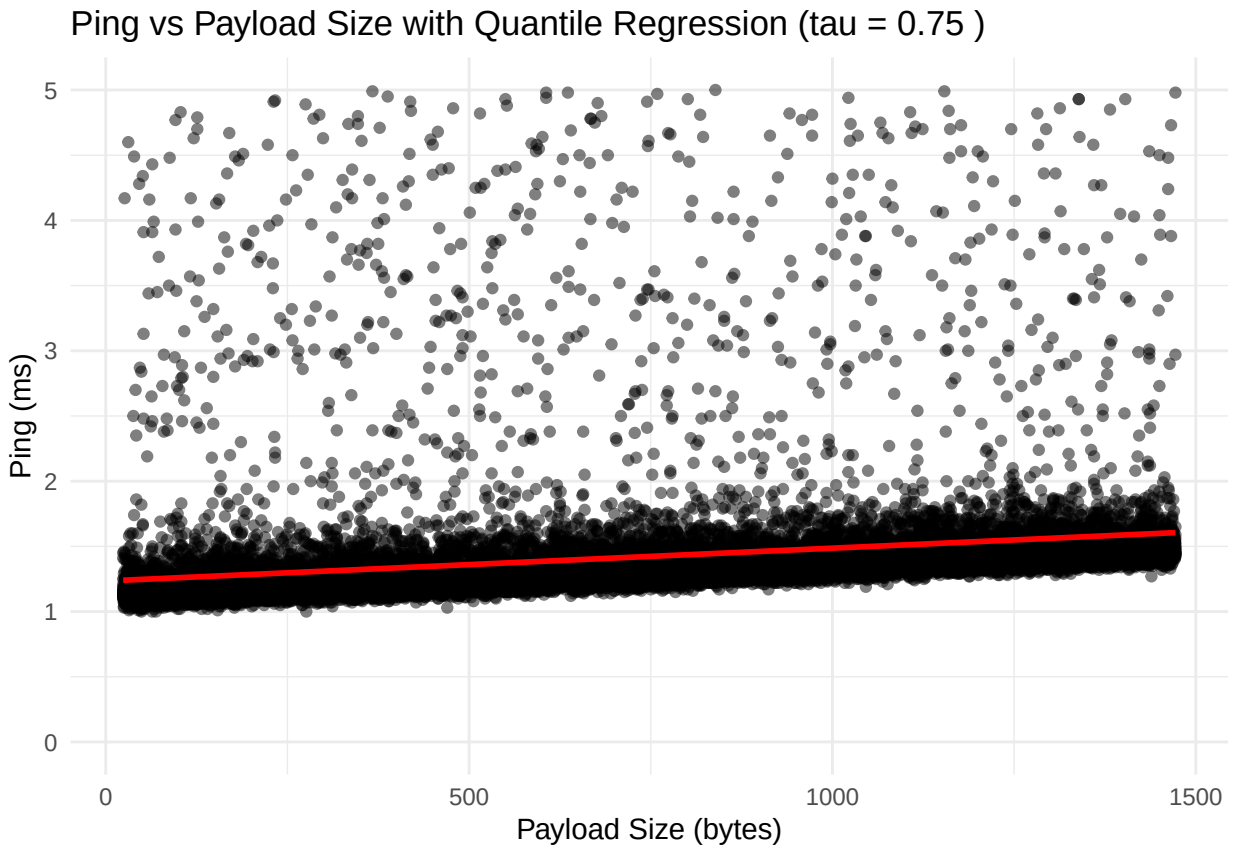
```
## Warning: Removed 4357 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
summarize_qregression(data_unfragmented, 0.75, 5)
```

```
##
## Call: rq(formula = ping_ms ~ size_bytes, tau = tau, data = data)
##
## tau: [1] 0.75
##
## Coefficients:
##             Value     Std. Error t value   Pr(>|t|)
## (Intercept)   1.23430   0.00515  239.50616   0.00000
## size_bytes    0.00025   0.00001   42.95345   0.00000
## C (1/coef_size): 3968.182
## L (intercept): 1.234296
## Residual quantiles (min, 25%, median, 75%, max):
##            0%           25%           50%           75%          100%
## -3.269301e-01 -1.480298e-01 -9.861397e-02 -1.145475e-05  9.346939e+01
```

```
## Warning: Removed 4357 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



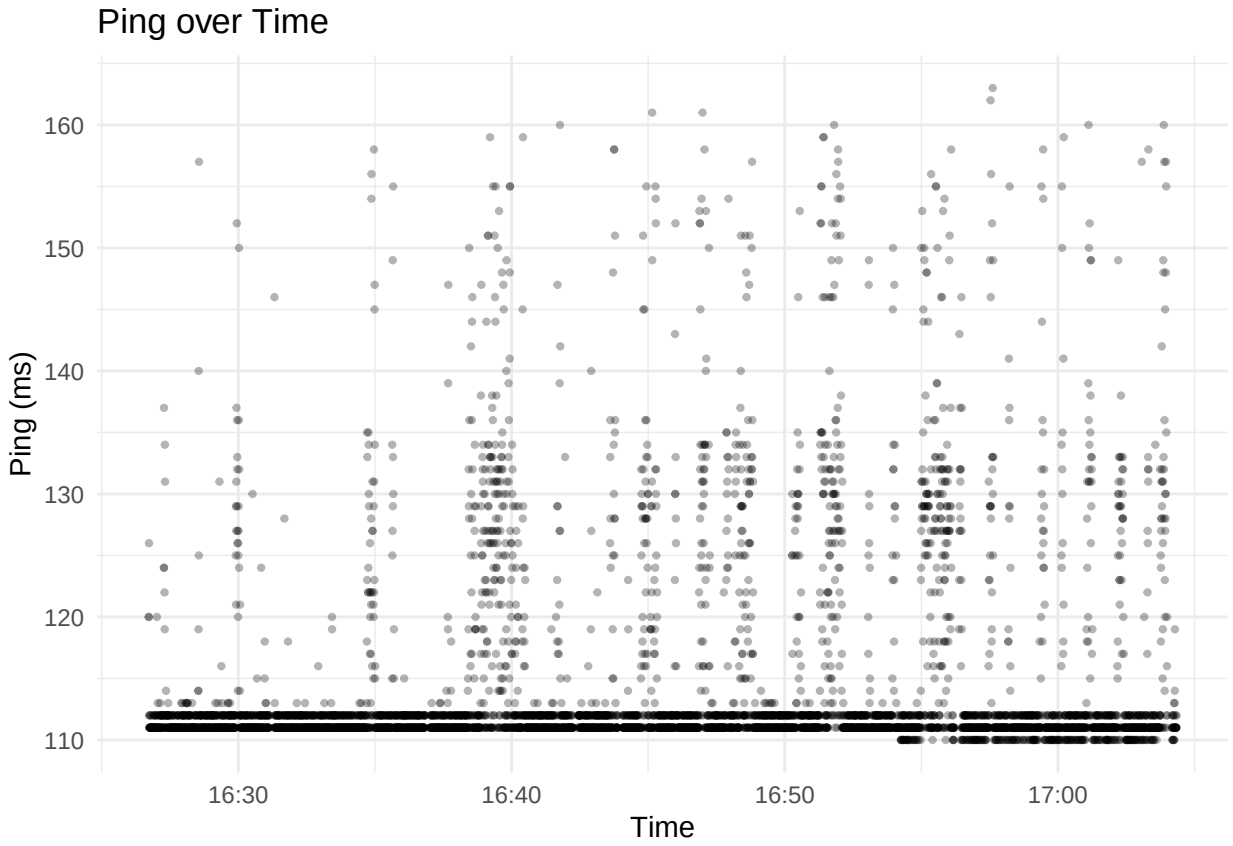
It seems that we have the best fit for tau=0.5. Based on the comparison of the residual quantiles, this choice provides a better fit for the data overall.

Stackoverflow dataset

```
data <- parse_ping_log("stackoverflow.log")
str(data)
```

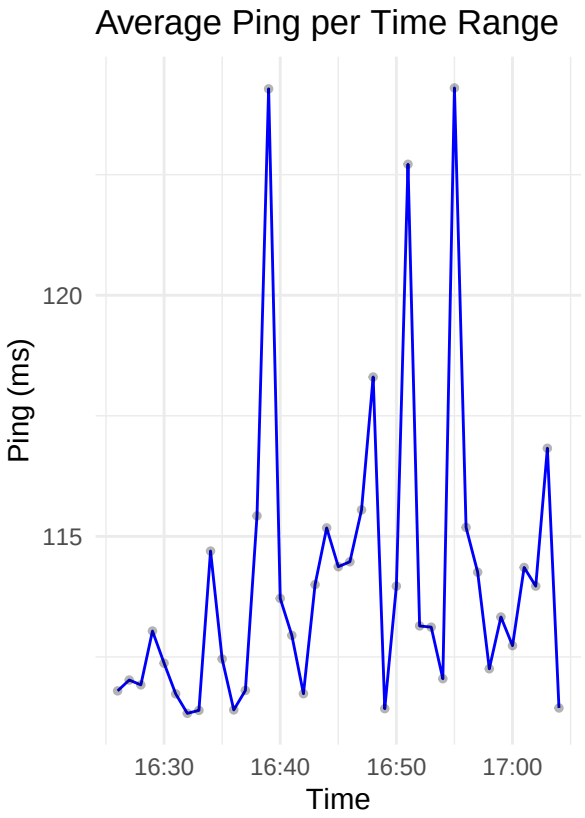
```
## 'data.frame':    6824 obs. of  3 variables:
##  $ times     : POSIXct, format: "2015-01-20 16:26:43" "2015-01-20 16:26:43" ...
##  $ size_bytes: num  1257 454 775 1334 83 ...
##  $ ping_ms   : num  120 120 126 112 111 111 112 111 111 111 ...
```

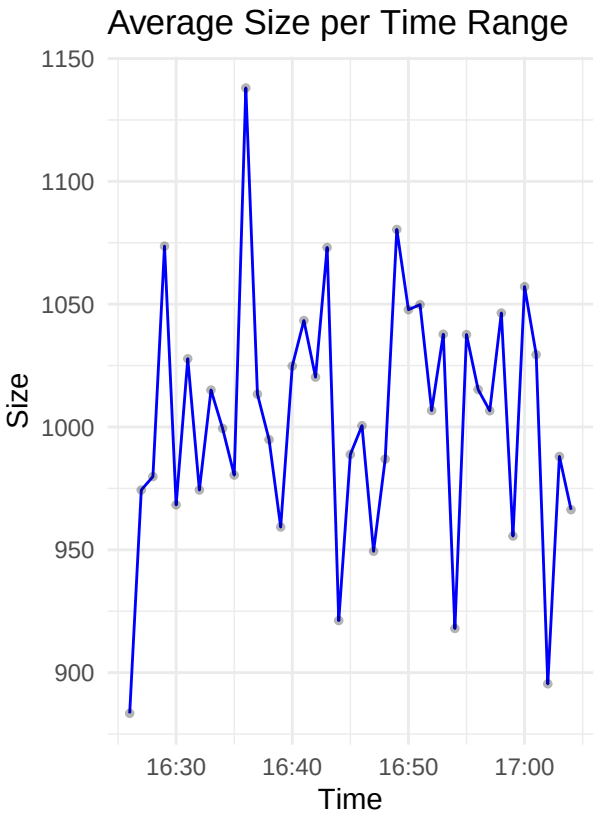
```
ggplot(data, aes(x = times, y = ping_ms)) +
  geom_point(alpha = 0.3, size = 0.8) +
  labs(
    title = "Ping over Time",
    x = "Time",
    y = "Ping (ms)"
  ) +
  theme_minimal()
```



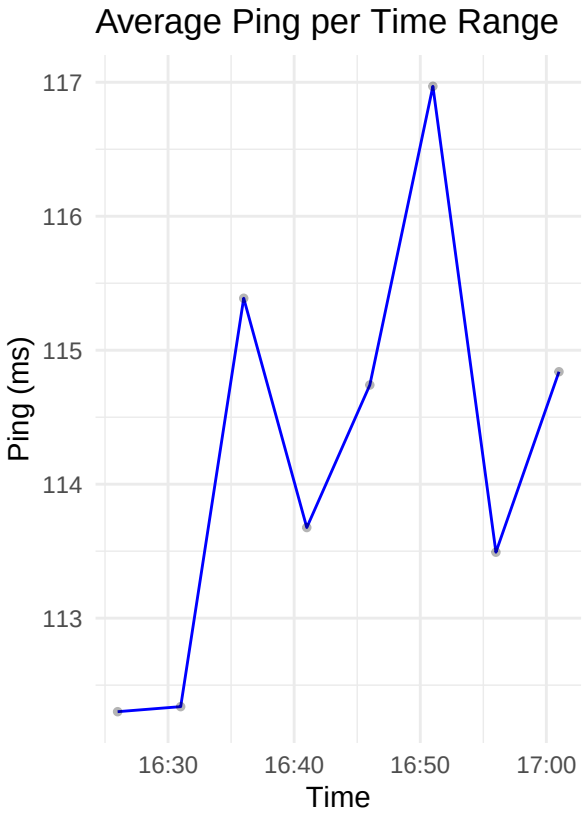
By visually analyzing the data, we can see that the transmission times are larger in this case. No obvious outliers are visible.

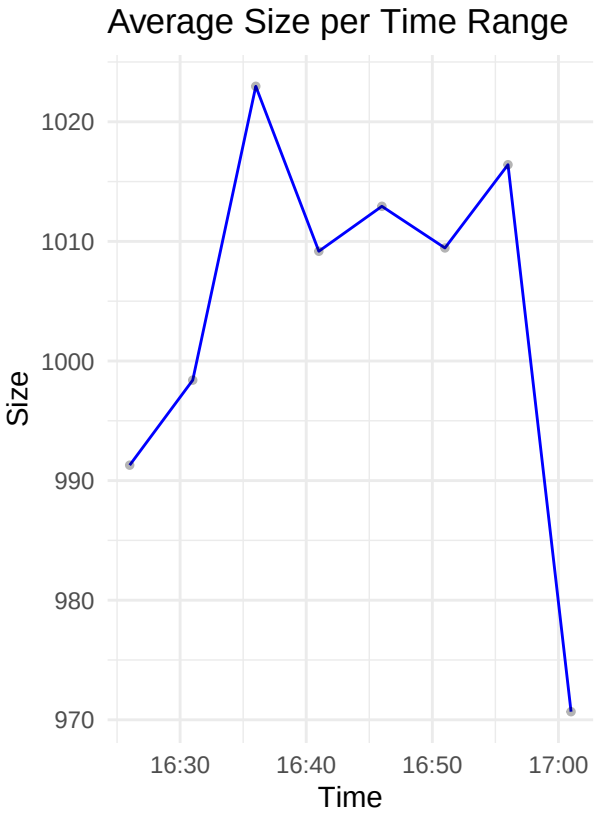
```
plot_means_for_timerange(data, "1 min")
```



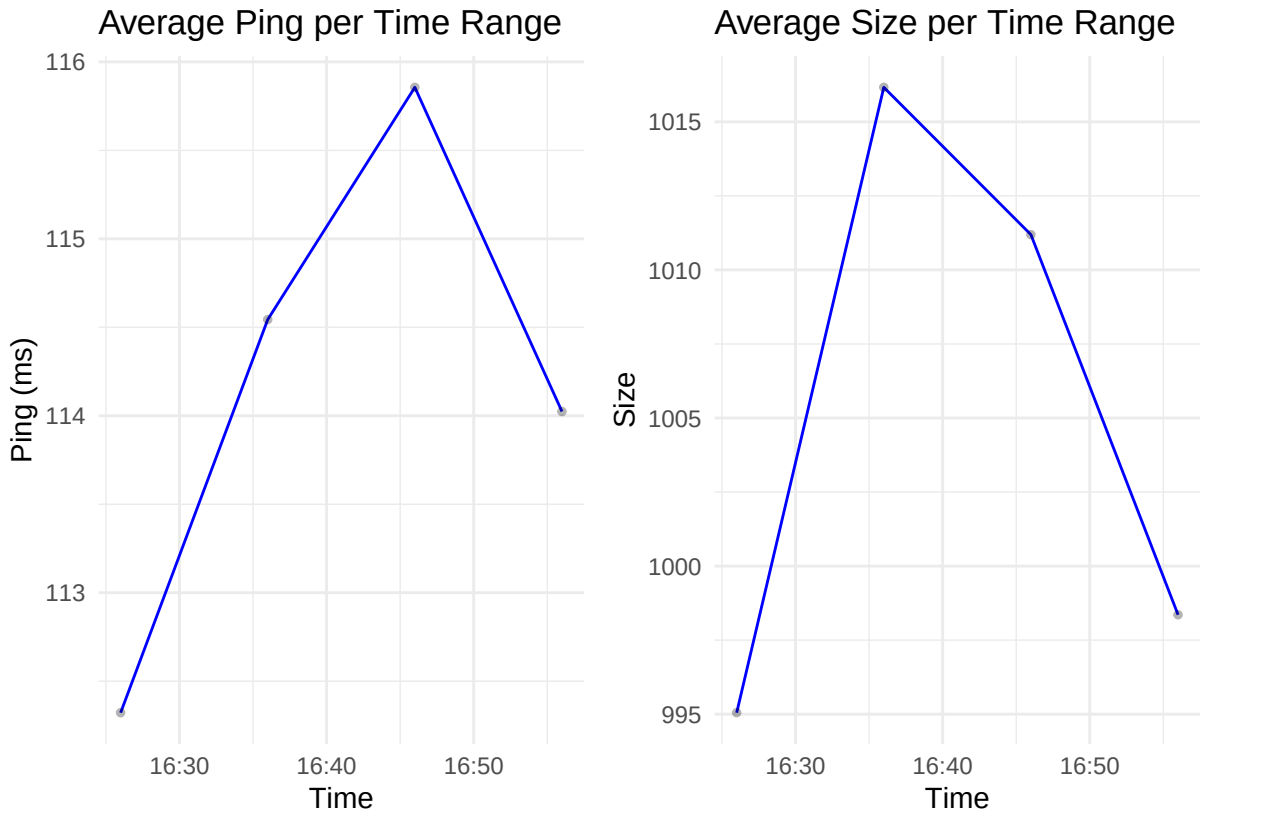



```
plot_means_for_timerange(data, "5 min")
```



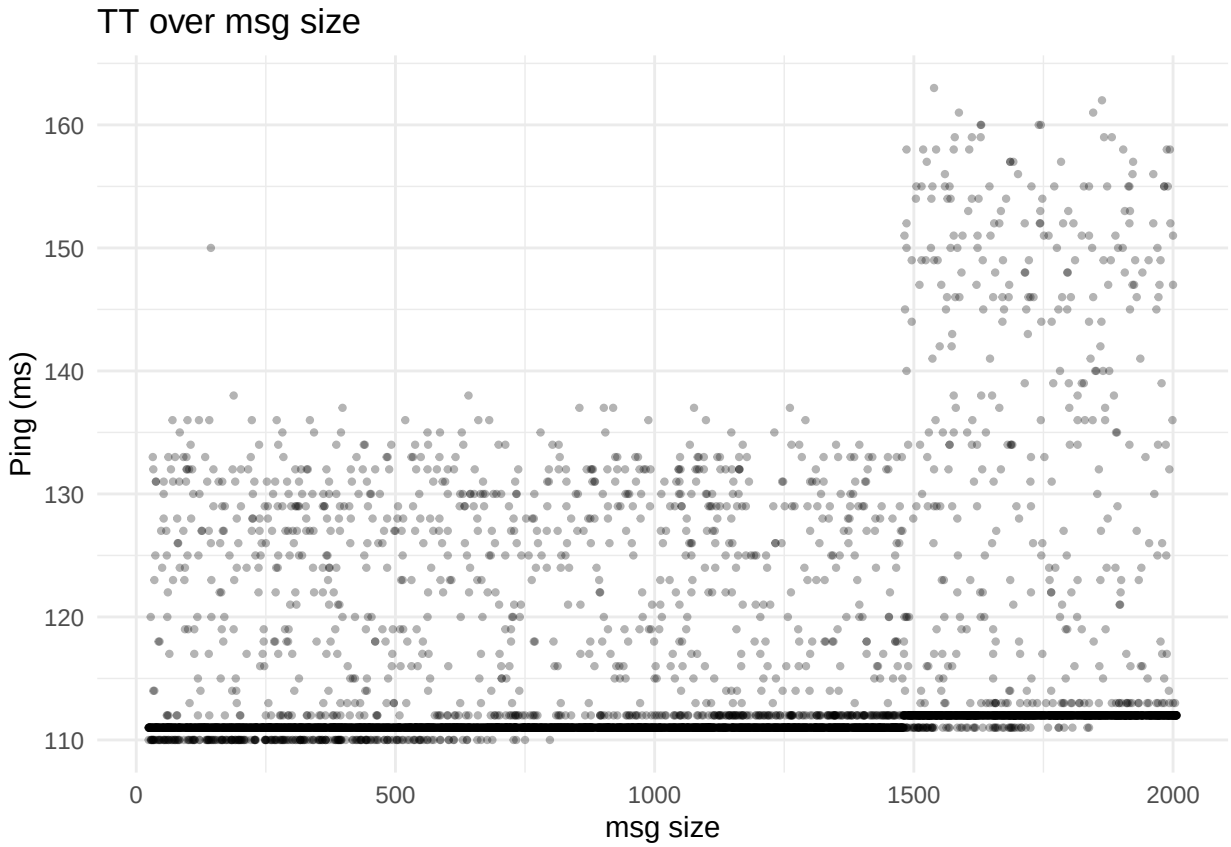


```
plot_means_for_timerange(data, "10 min")
```



Visually, it appears that payload size has a stronger correlation with transmission time than before. The line plots also show more similar shapes.

```
ggplot(data, aes(x = size_bytes, y = ping_ms)) +
  geom_point(alpha = 0.3, size = 0.8) +
  labs(
    title = "TT over msg size",
    x = "msg size",
    y = "Ping (ms)"
  ) +
  theme_minimal()
```



Here, we also see that the dataset should be split into two classes, which appears to be caused by the same factor: fragmented and unfragmented packets.

```
threshold <- 1472

data_unfragmented <- data[data$size_bytes <= threshold, ]
data_fragmented <- data[data$size_bytes > threshold, ]

str(data_fragmented)
```

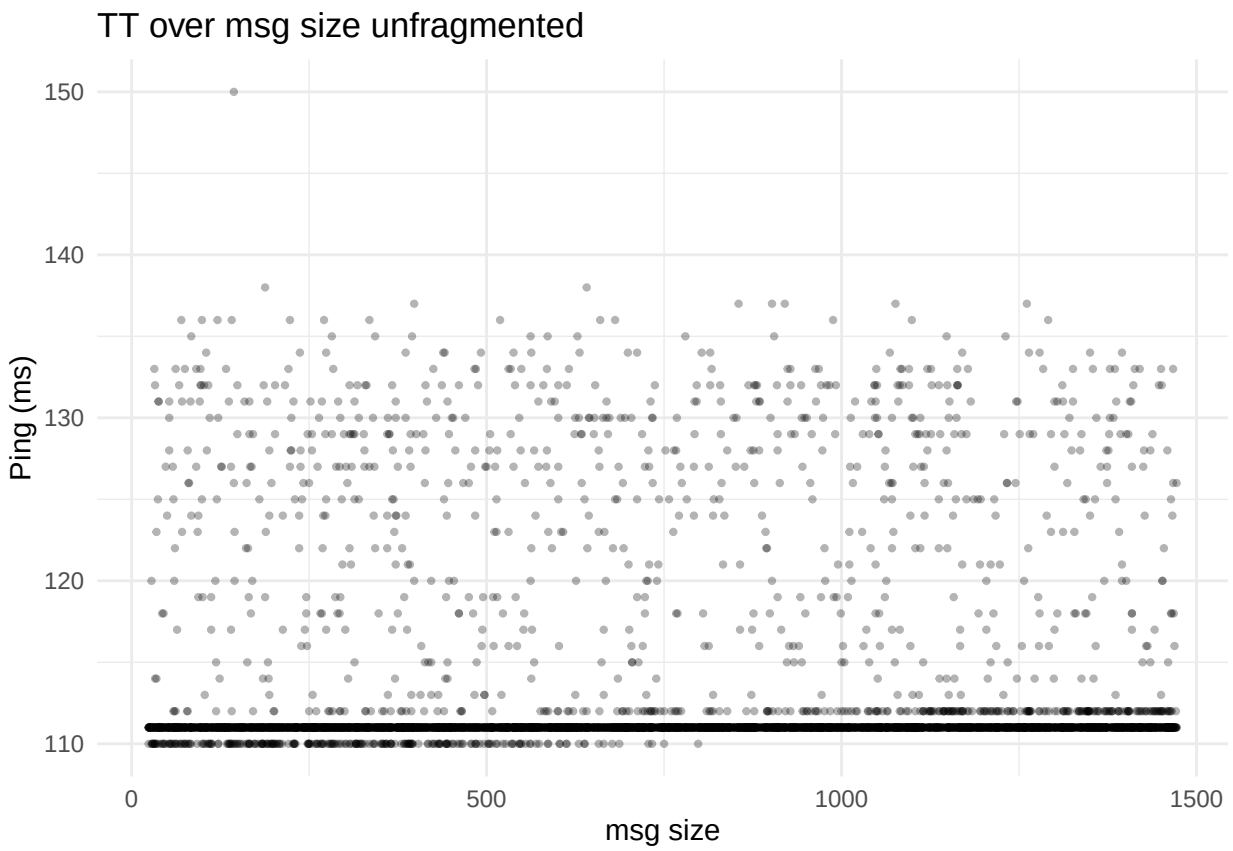
```
## 'data.frame':    1840 obs. of  3 variables:
##  $ times     : POSIXct, format: "2015-01-20 16:26:45" "2015-01-20 16:26:46" ...
##  $ size_bytes: num  1577 1714 1598 1619 1655 ...
##  $ ping_ms   : num  112 112 112 112 112 112 112 112 120 112 ...
```

```
str(data_unfragmented)
```

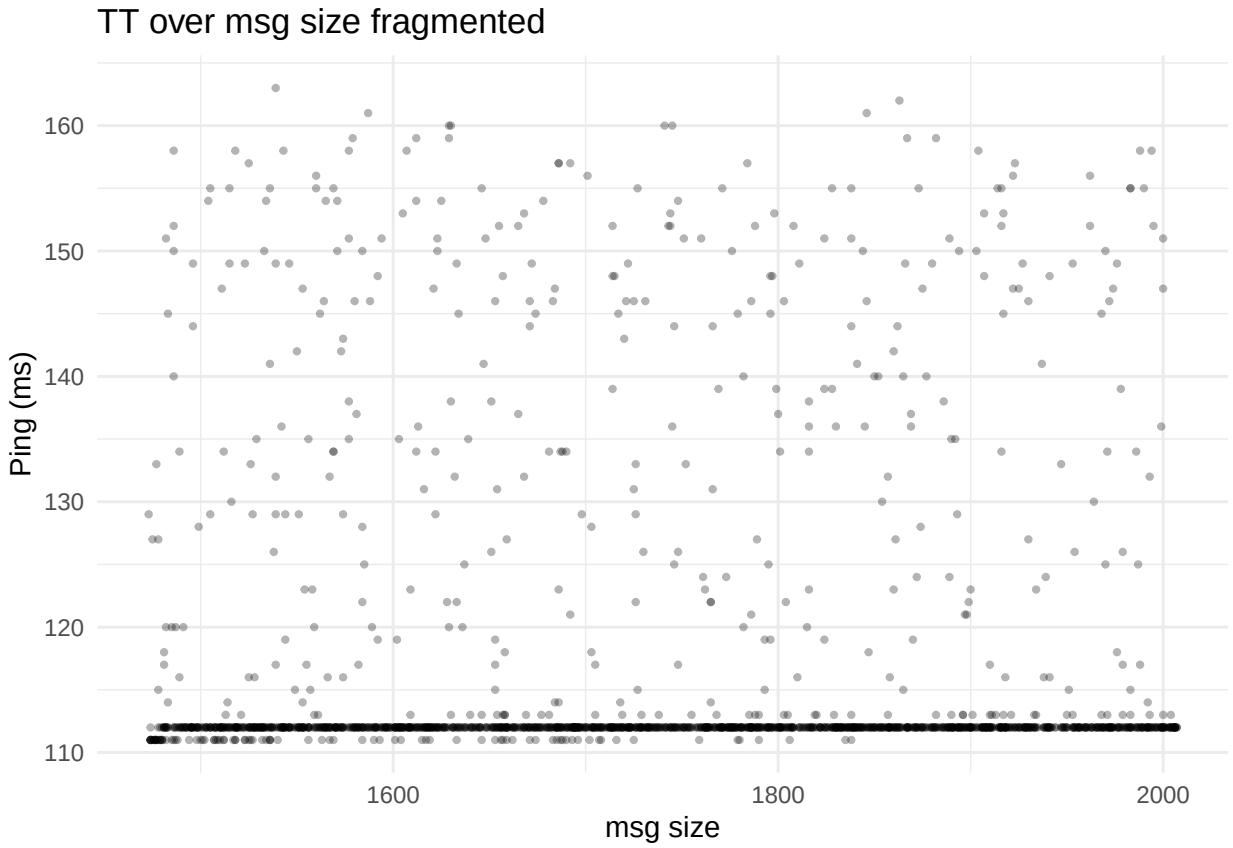
```
## 'data.frame':    4984 obs. of  3 variables:
##  $ times     : POSIXct, format: "2015-01-20 16:26:43" "2015-01-20 16:26:43" ...
##  $ size_bytes: num  1257 454 775 1334 83 ...
##  $ ping_ms   : num  120 120 126 112 111 111 111 111 111 111 ...
```

```
ggplot(data_unfragmented, aes(x = size_bytes, y = ping_ms)) +
  geom_point(alpha = 0.3, size = 0.8) +
  labs(
    title = "TT over msg size unfragmented",
    x = "msg size",
    y = "Ping (ms)"
  ) +
```

```
  theme_minimal()
```




```
ggplot(data_fragmented, aes(x = size_bytes, y = ping_ms)) +
  geom_point(alpha = 0.3, size = 0.8) +
  labs(
    title = "TT over msg size fragmented",
    x = "msg size",
    y = "Ping (ms)"
  ) +
  theme_minimal()
```




```
summarize_regression(data_unfragmented, 200)
```

```
##
## Call:
## lm(formula = ping_ms ~ size_bytes, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.261 -2.271 -2.254 -2.235 36.763
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.132e+02  1.660e-01 682.251   <2e-16 ***
## size_bytes  3.785e-05  1.956e-04   0.193    0.847
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.817 on 4982 degrees of freedom
## Multiple R-squared:  7.514e-06,  Adjusted R-squared:  -0.0001932
## F-statistic: 0.03744 on 1 and 4982 DF,  p-value: 0.8466
##
## [1] "C:  26423.2150499968"
## [1] "L:  113.231152911716"
```

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


```
summarize_regression(data_fragmented, 200)
```

```
##
## Call:
## lm(formula = ping_ms ~ size_bytes, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.154 -4.988 -4.783 -4.578 45.919
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 118.802924   3.088401  38.467   <2e-16 ***
## size_bytes   -0.001119   0.001774  -0.631    0.528
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.82 on 1838 degrees of freedom
## Multiple R-squared:  0.0002164,  Adjusted R-squared:  -0.0003275
## F-statistic: 0.3979 on 1 and 1838 DF,  p-value: 0.5282
##
## [1] "C:  -893.809793711346"
## [1] "L:  118.802923541362"
```

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

Residual errors for both classes indicate underprediction in both cases. The R^2 values are very low, especially for unfragmented packets, which, as in the previous analysis, suggests that variability in transmission times is barely explained by packet size. For fragmented packets, the coefficient for size in bytes is negative, implying that larger packets appear to correspond to shorter transmission times. This counterintuitive result indicates an even poorer fit for this class, which is reasonable given that, for larger sites like StackOverflow, transmission times are influenced by many factors and network conditions, leading to higher variability.

Lets try to also do regression on a subset where only the lowest transmission time for eahc paylaod size is used.

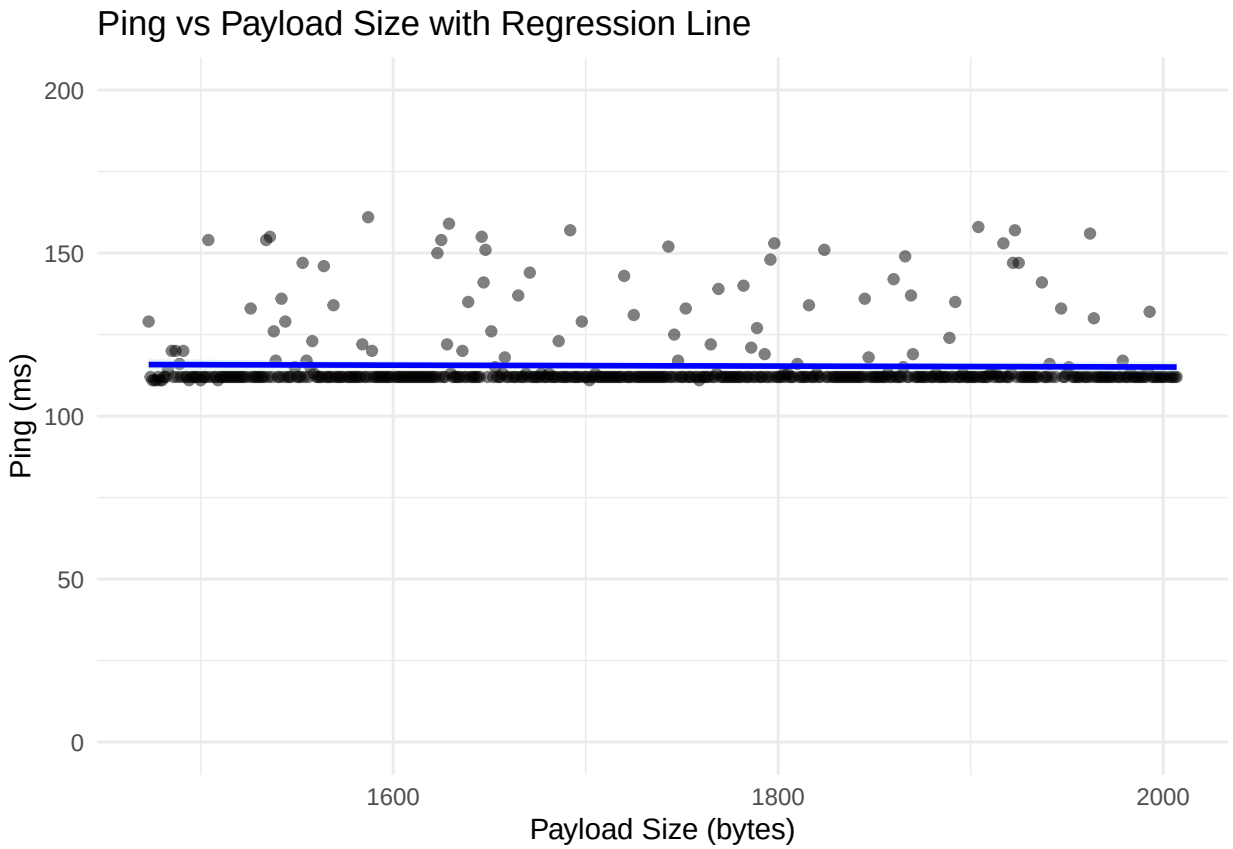
```
data_unfragmented_min <- data_unfragmented %>%
  group_by(size_bytes) %>%
  slice_min(times, n = 1) %>%
  ungroup()


data_fragmented_min <- data_fragmented %>%
  group_by(size_bytes) %>%
  slice_min(times, n = 1) %>%
  ungroup()
```

```
summarize_regression(data_fragmented_min, 200)
```

```
##
## Call:
## lm(formula = ping_ms ~ size_bytes, data = data)
##
## Residuals:
```

```
##    Min     1Q Median     3Q    Max
## -4.797 -3.569 -3.333 -3.088 45.362
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 117.895402   4.896109  24.079   <2e-16 ***
## size_bytes   -0.001423   0.002805  -0.507    0.612
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.749 on 512 degrees of freedom
## Multiple R-squared:  0.000502,   Adjusted R-squared:  -0.00145
## F-statistic: 0.2572 on 1 and 512 DF,  p-value: 0.6123
##
## [1] "C:  -702.939075424752"
## [1] "L:  117.895402276957"
```

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

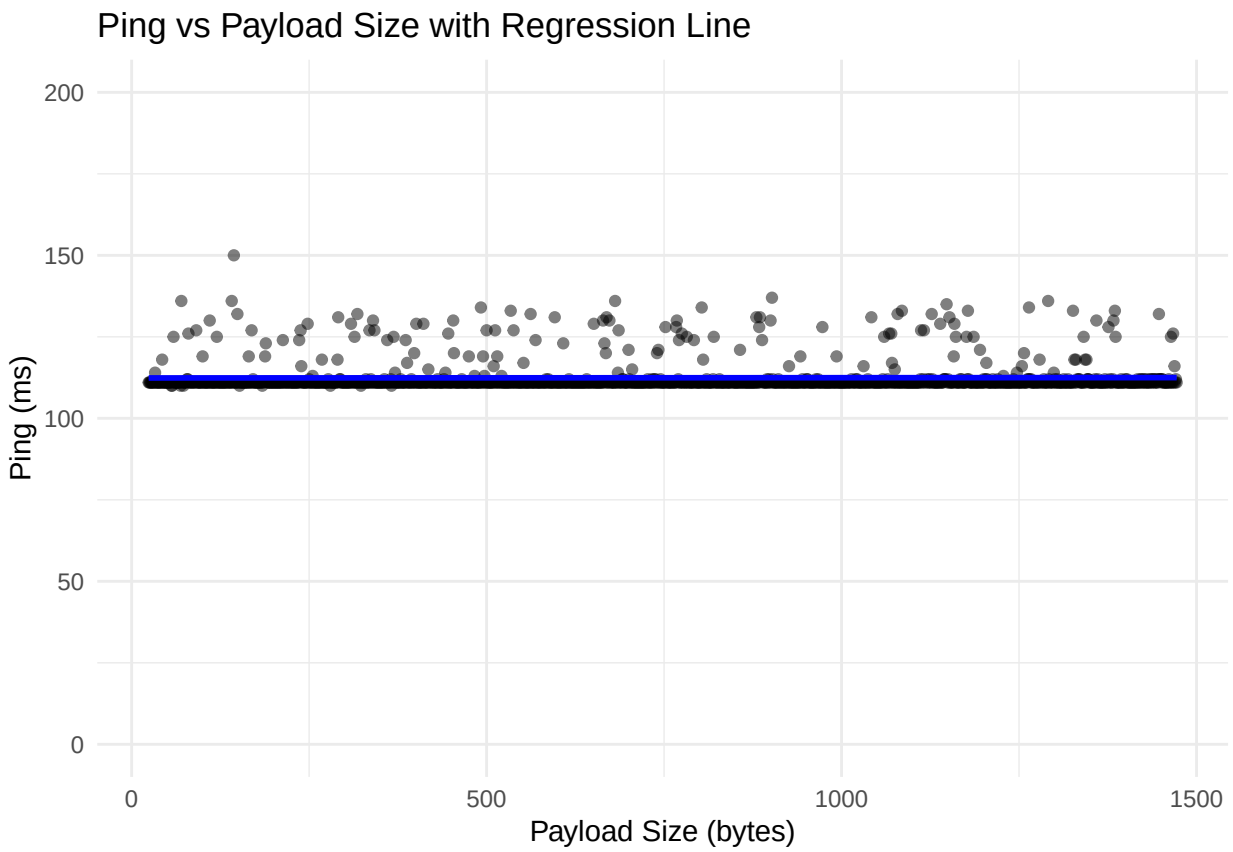



```
summarize_regression(data_unfragmented_min, 200)
```

```
##
## Call:
## lm(formula = ping_ms ~ size_bytes, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.411 -1.456 -1.426 -1.395 37.605
```

```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.124e+02  2.508e-01 448.160   <2e-16 ***
## size_bytes  7.106e-05  2.928e-04   0.243    0.808
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.601 on 1400 degrees of freedom
## Multiple R-squared:  4.207e-05,  Adjusted R-squared:  -0.0006722
## F-statistic: 0.0589 on 1 and 1400 DF,  p-value: 0.8083
##
## [1] "C:  14073.0816490408"
## [1] "L:  112.3848948251"
```

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



Performing regression on this subset improved the fit slightly, as suggested by the residual values. The R-squared is also slightly higher, but the improvement is minimal. Overall, we can conclude that, especially for this dataset, the model fit remains poor.

Conclusions

The analysis suggests that the chosen model, in which transmission time depends solely on packet size, latency and capacity, is too simple and does not correspond closely to the values observed in the datasets. This may be due to the high variance in the data, which arises from the specifics of network measurements, transmission times are heavily influenced by current network conditions and infrastructure, both of which can change rapidly and significantly, having a major impact on the final transmission time. The error for our model does

not seem to ahve normal ditribution thats why quanitile regression seem to solve to heva eay better fit.