CSCI 385 - Third Deliverable

Kris Selvidge

12/14/2020

Introduction

As previously described, the domain of this project is network data and how we can use R to help visualize performance metrics related to a programming experiment developed for a CSU Chico Networking class. (Additional details on how my experiment was setup are featured in the first data set section further in this report.)

As a recap, in the first deliverable we explored and discovered the direct output (data set 1) from our programming experiment. Based on this examination, we refined and reran the experiment to deliver a revised data set for deliverable 2. From this new exploration and review we considered a few possible models and fully developed, trained and tested a simple yet strong linear model between two of our output variables. Finally, we captured results from our program using an independent network traffic capture program (Wireshark) and detailed the import process of these results (data set 2) from JSON format to R.

Following the last two deliverables, I now have three goals for this next phase of this project:

1.)While our model developed in the previous deliverable had strong results, both variables used were dependent variables in the experiment. Of the three independent variables we are more interested in modeling, we saw interesting results using chunk size that would required a slightly more advanced, logarithmic model. I will attempt to implement this here.

2.) Connect data set 2 to data set 1 and explore results, discover potential for modeling, and implementing those potentials.

3.) If the merge process and models from the previous goals are successful, I would like to rerun Wireshark to capture web browser performance and compare them with the experimental model.

Revised Data Set 1

While some changes were made between deliverable 1 and 2, the results of deliverable 2 were promising enough that I'd like to keep this set for our new modeling.

Loading Data Set 1 Subset Site 1: Amazon

```
amazon <- read_csv('D:\\fall2020\\TTH\\CSCI385 Data Science\\a2\\amazon.csv')
```

```
## Parsed with column specification:
## cols(
##   source = col_character(),
##   destination = col_character(),
##   protocol = col_character(),
##   chunkSize = col_double(),
```

```
##   tCount = col_double(),
##   fileSize = col_double(),
##   retransmits = col_double(),
##   connectTime = col_double(),
##   requestTime = col_double(),
##   receiveTime = col_double()
## )
```

Loading Data Set 1 Subset Site 1: Facebook

```
facebook <- read_csv('D:\\fall2020\\TTH\\CSCI385 Data Science\\a2\\facebook.csv')
```

```
## Parsed with column specification:
## cols(
##   source = col_character(),
##   destination = col_character(),
##   protocol = col_character(),
##   chunkSize = col_double(),
##   tCount = col_double(),
##   fileSize = col_double(),
##   retransmits = col_double(),
##   connectTime = col_double(),
##   requestTime = col_double(),
##   receiveTime = col_double()
## )
```

Loading Data Set 1 Subset Site 1: Google

```
google <- read_csv('D:\\fall2020\\TTH\\CSCI385 Data Science\\a2\\google.csv')
```

```
## Parsed with column specification:
## cols(
##   source = col_character(),
##   destination = col_character(),
##   protocol = col_character(),
##   chunkSize = col_double(),
##   tCount = col_double(),
##   fileSize = col_double(),
##   retransmits = col_double(),
##   connectTime = col_double(),
##   requestTime = col_double(),
##   receiveTime = col_double()
## )
```

Loading Data Set 1 Subset Site 1: Yahoo

```
yahoo <- read_csv('D:\\fall2020\\TTH\\CSCI385 Data Science\\a2\\yahoo.csv')
```

```
## Parsed with column specification:
## cols(
##   source = col_character(),
```

```
##   destination = col_character(),
##   protocol = col_character(),
##   chunkSize = col_double(),
##   tCount = col_double(),
##   fileSize = col_double(),
##   retransmits = col_double(),
##   connectTime = col_double(),
##   requestTime = col_double(),
##   receiveTime = col_double()
## )
```

Loading Data Set 1 Subset Site 1:Youtube

```
youtube <- read_csv('D:\\fall2020\\TTH\\CSCI385 Data Science\\a2\\youtube.csv')
```

```
## Parsed with column specification:
## cols(
##   source = col_character(),
##   destination = col_character(),
##   protocol = col_character(),
##   chunkSize = col_double(),
##   tCount = col_double(),
##   fileSize = col_double(),
##   retransmits = col_double(),
##   connectTime = col_double(),
##   requestTime = col_double(),
##   receiveTime = col_double()
## )
```

Combining all Data Set 1 Subset Sites

```
netdat <- bind_rows(amazon, facebook, google, yahoo, youtube)
head(netdat)
```

```
## # A tibble: 6 x 10
##   source destination protocol chunkSize tCount fileSize retransmits connectTime
##   <chr>  <chr>       <chr>        <dbl>  <dbl>    <dbl>       <dbl>       <dbl>
## 1 192.1~ www.amazon~ HTTP/1.0         4      7     1263           2    0.00084
## 2 192.1~ www.amazon~ HTTP/1.1         4      7     1263           2    0.000445
## 3 192.1~ www.amazon~ HTTP/1.0         5     12     1263           2    0.000437
## 4 192.1~ www.amazon~ HTTP/1.1         5     12     1263           2    0.000463
## 5 192.1~ www.amazon~ HTTP/1.0         6     13     1263           2    0.000448
## 6 192.1~ www.amazon~ HTTP/1.1         6     13     1263           2    0.000520
## # ... with 2 more variables: requestTime <dbl>, receiveTime <dbl>
```

Categorical Variables:

- 'source' - 'character' - Origin location requesting root file from destination. Note: In this phase of the experiement there is only one location (localhost) requesting data. This computer is my local HP Omen 17 laptop running Ubuntu 18.
- 'destination' - 'character' - Web server sending root file. These hosts are ten top and popular websites.

- ‘protocol’ - ‘character’ - Transfer protocol version HTTP (Hypertext Transfer Protocol) 1.0 or 1.1. Note that twitter.com does not respond to HTTP 1.1 requests.

Continuous Variables

- ‘chunkSize’ - ‘double’ - Size of each chunk of packet data in bytes. This value is the only continuous variable adjusted during the experiment. Values range from 4 to 1000.

(Note all variable values below were observed as a part of the experiment.)

- ‘tCount’ - ‘double’ - Number of completed HTML markup tags parsed within chunks. Tags markup content within documents and are opened with character ‘<’ and closed with character ‘>’.
- ‘fileSize’ - ‘double’ - Total root file size in bytes.
- ‘retransmits’ - ‘double’ - Number of packets smaller than chunkSize indicating lost bytes that required a retransmit of data.
- ‘connectTime’ - ‘double’ - Time in seconds to establish connection from source to destination.
- ‘requestTime’ - ‘double’ - Time in seconds to submit request from source to destination.
- ‘receiveTime’ - ‘double’ - Time in seconds to receive root file from destination to source.

Manipulated/Independent Variables

For our program experiment, we manually chose the following three variables as input to determine the resulting observations:

destination (amazon.com, facebook.com, google.com, yahoo.com, youtube.com) protocol (HTTP/1, HTTP/1.1) chunkSize (4:120)

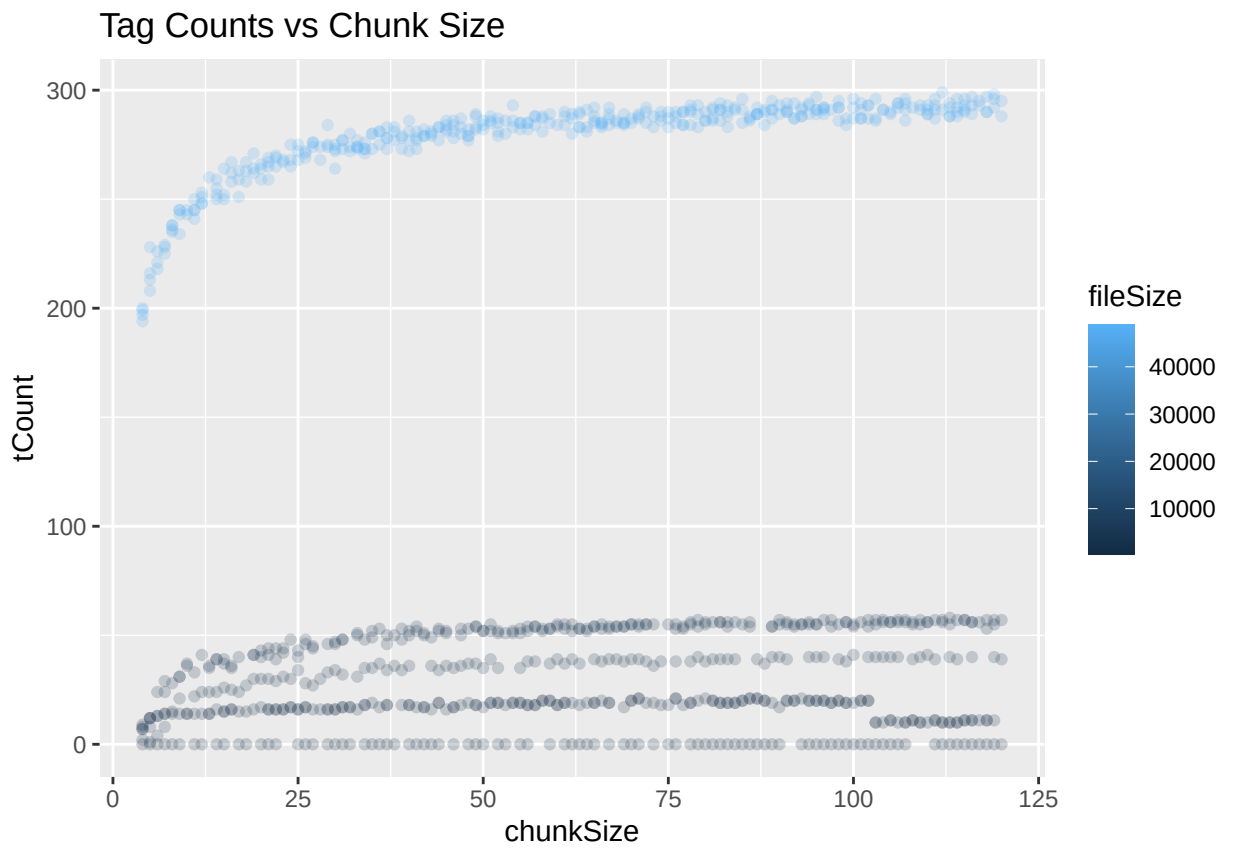
Relevant Data Analysis Recap

Below is a brief recap of the data we explored in the previous deliverable. I will use these for our new models.

Since I’ll be creatinng new models, I’ll set aside some test data here.

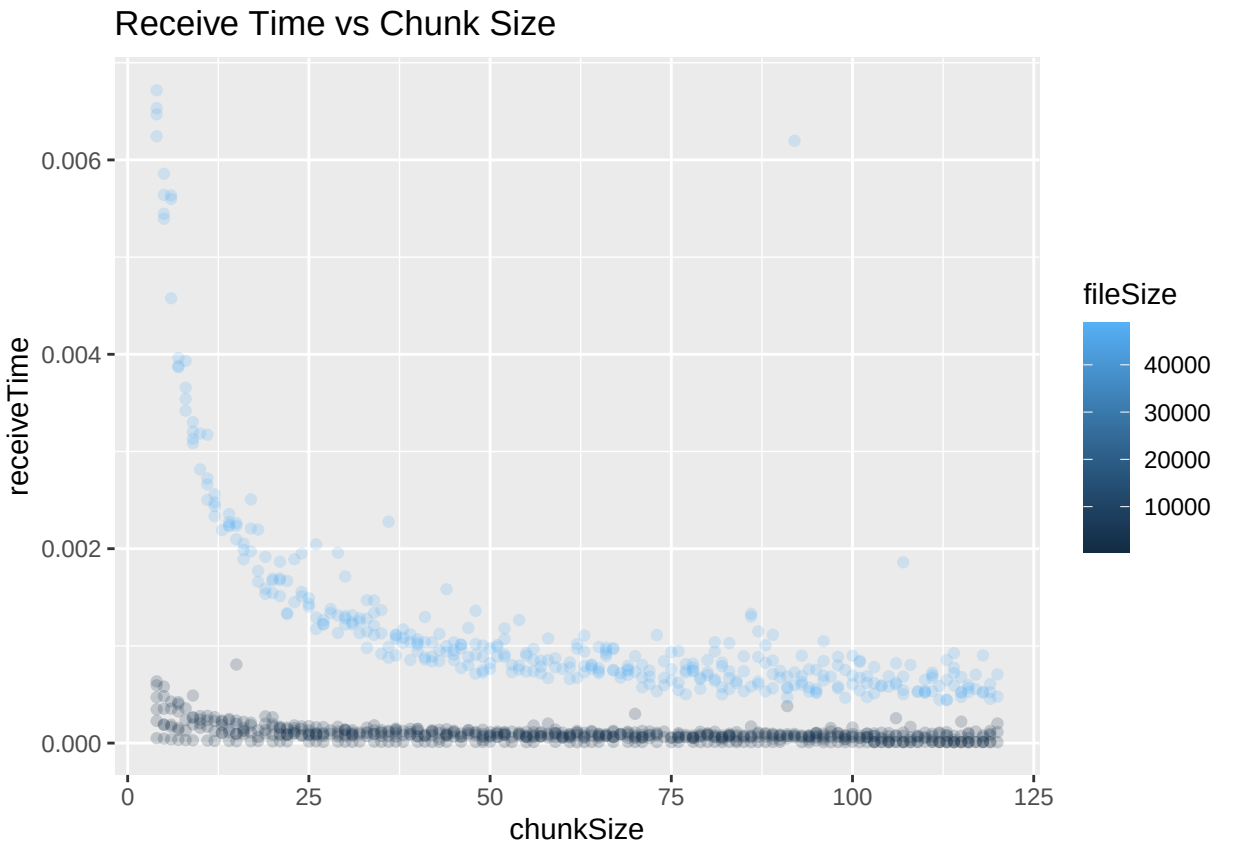
```
rest_rows <- as.vector(createDataPartition(netdat$chunkSize, p = 0.8, list = FALSE))
test <- netdat[-rest_rows, ]
rest <- netdat[rest_rows, ]
```

```
ggplot(data = rest) +
  geom_point(mapping = aes(x = chunkSize, y = tCount, color=fileSize), alpha = 0.2) +
  ggtitle("Tag Counts vs Chunk Size")
```



In the previous deliverable, we saw an increase of tag counts as chunk size increases. The relationship appears to be logarithmic.

```
ggplot(data = rest) +
  geom_point(mapping = aes(x = chunkSize, y = receiveTime,color = fileSize), alpha = 0.2) +
  ggtitle("Receive Time vs Chunk Size")
```



We also saw that larger files with smaller chunk sizes are connected to higher receiving times.

New Models for Data Set 1

In this third deliverable we'll compare two different new models. We saw what looked like a logarithmic relationship in both the chunk size vs the tag count and also the chunk time vs the receiving time.

As in our previous modeling, we'll start by taking data we separated from the test group into two new groups: validation and training. Our test set removed 20% of our data, and now we will put 60% into validation and the remaining 20% into test. We want 60% of our remaining 80%, so we will split this off again into 25% and 75%.

```
set.seed(1234)
train_rows <- as.vector(createDataPartition(rest$chunkSize, p = 0.75, list = FALSE))
validate <- rest[-train_rows, ]
train <- rest[train_rows, ]
```

I'm going to make a series of models using different formulas and compare the goodness of fit and graphs to see how they compare.

```
model <- lm(receiveTime ~ chunkSize, data = train)
predictions <- add_predictions(validate, model)
R2(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.1288796
```

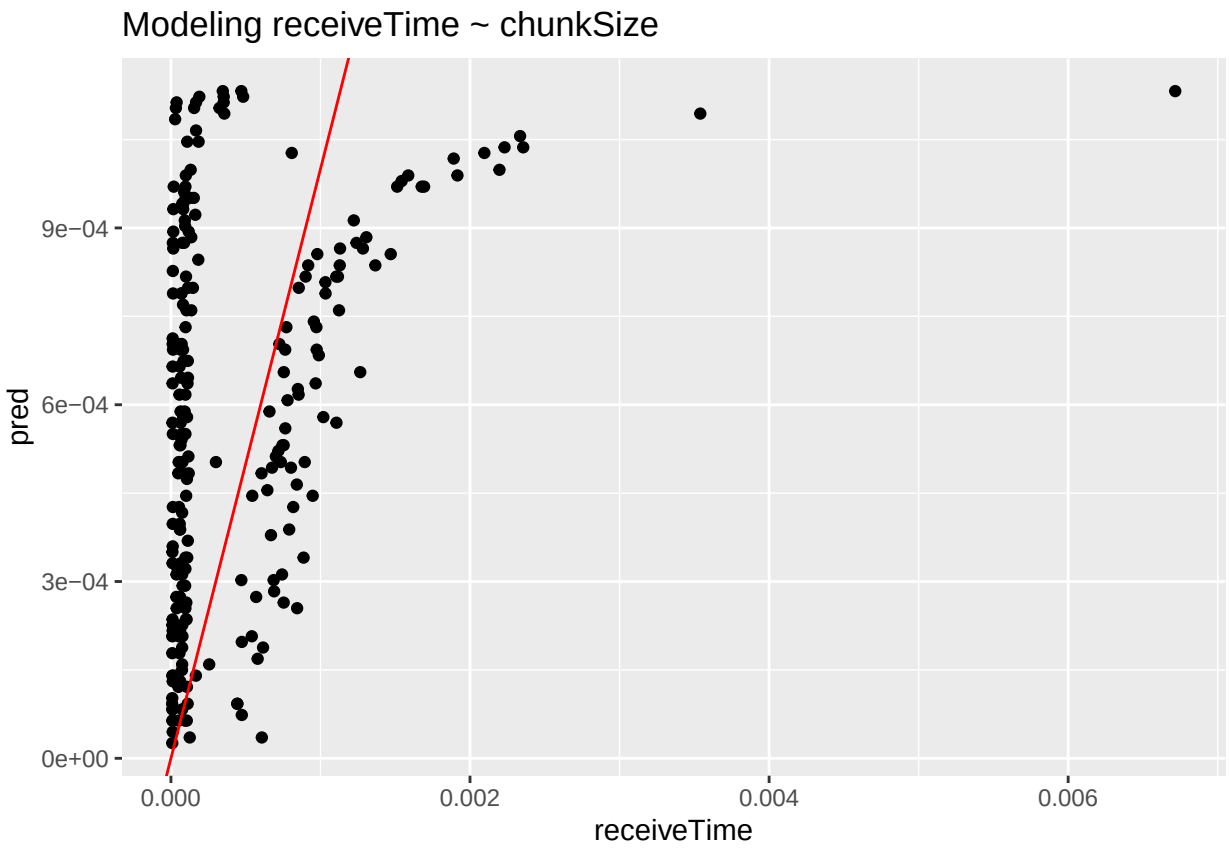
```
MAE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0004730555
```

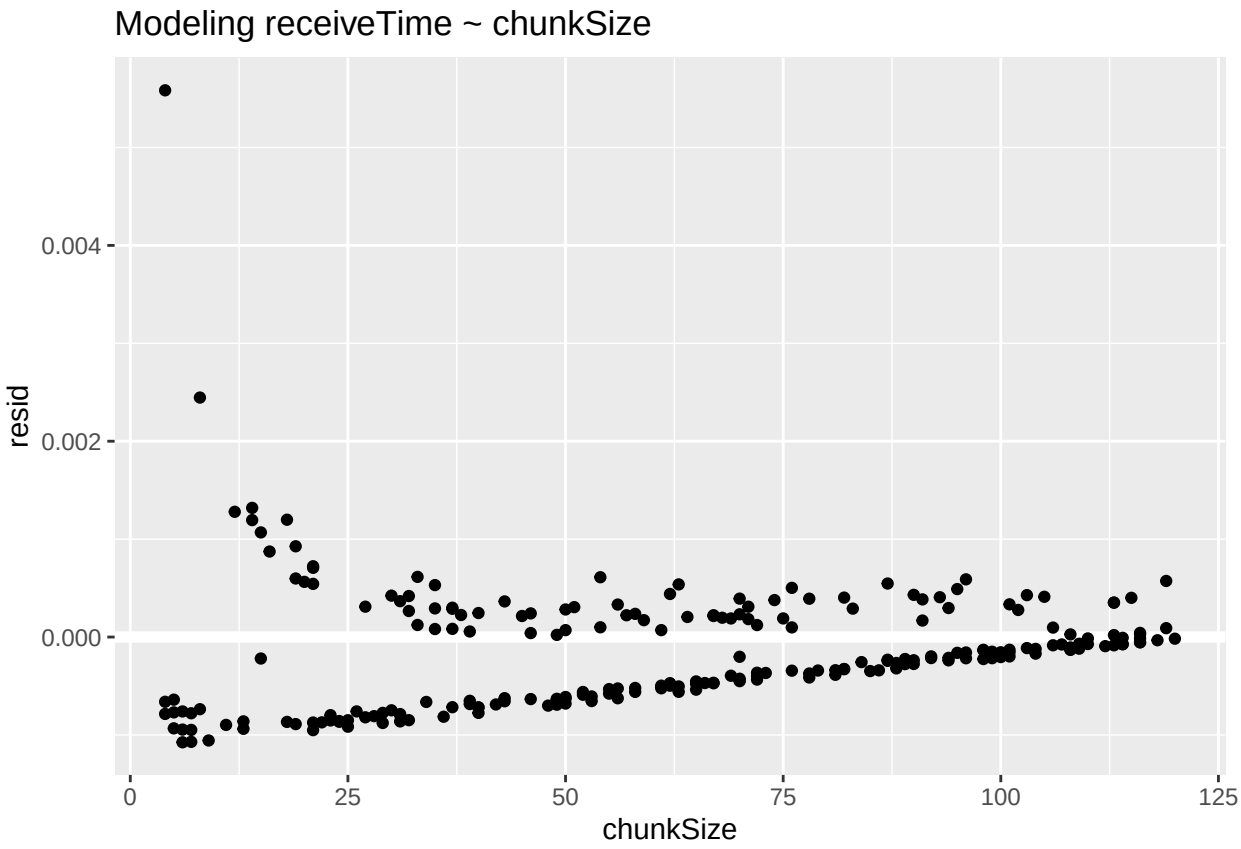
```
RMSE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0006625477
```

```
ggplot(data = predictions, mapping = aes(x = receiveTime, y = pred)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  ggtitle("Modeling receiveTime ~ chunkSize")
```




```
resids <- add_residuals(validate, model)
ggplot(data = resids, mapping = aes(x = chunkSize, y = resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  ggtitle("Modeling receiveTime ~ chunkSize")
```




```
rm(predictions)
rm(resids)

model <- lm(receiveTime ~ fileSize, data = train)
predictions <- add_predictions(validate, model)
R2(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.4988362
```

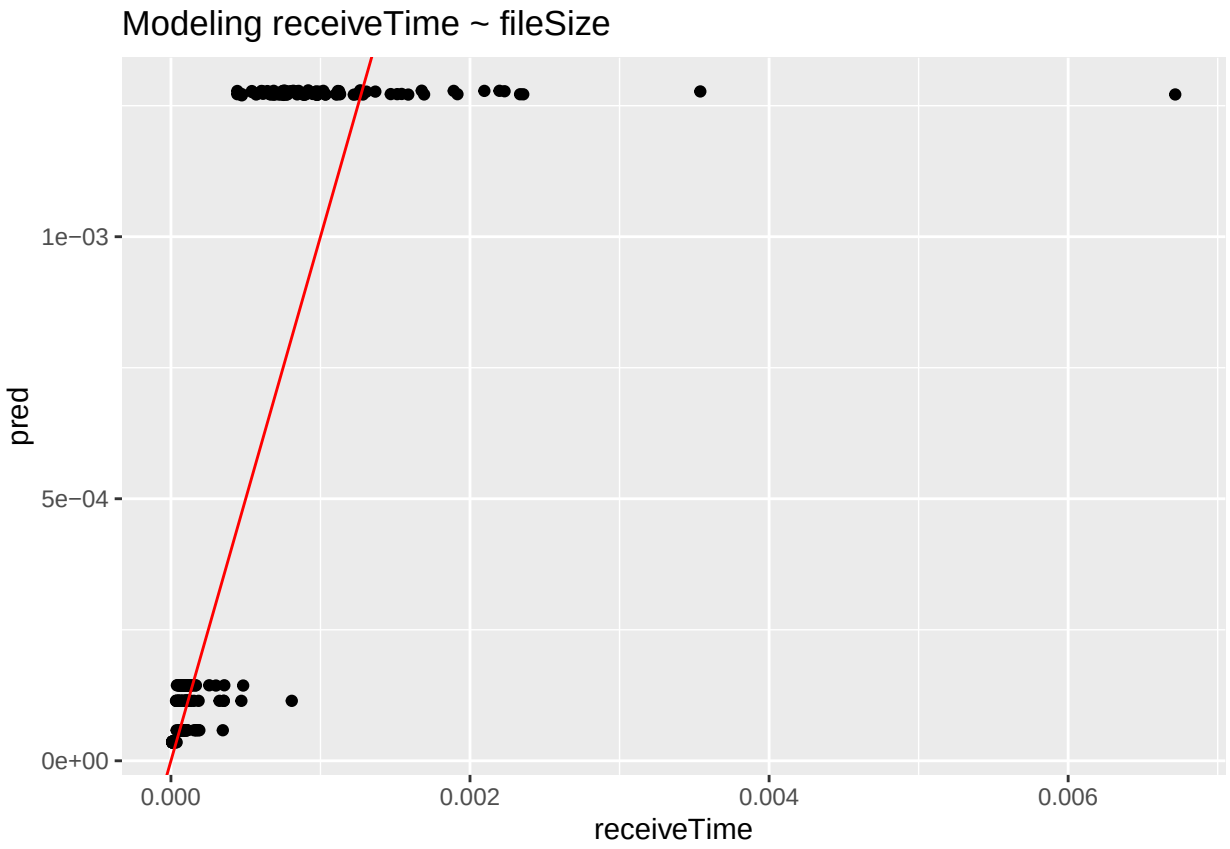
```
MAE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0002215804
```

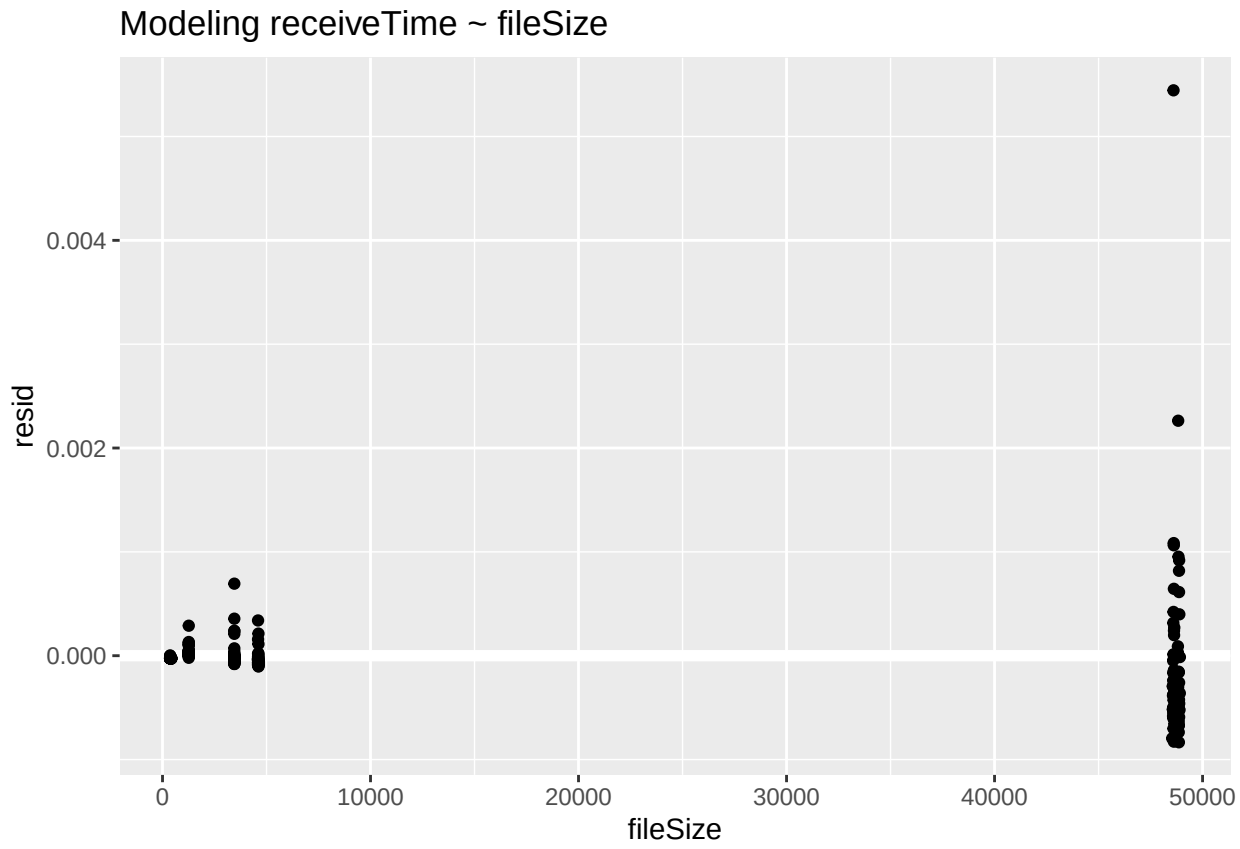
```
RMSE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.000497053
```

```
ggplot(data = predictions, mapping = aes(x = receiveTime, y = pred)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  ggtitle("Modeling receiveTime ~ fileSize")
```




```
resids <- add_residuals(validate, model)
ggplot(data = resids, mapping = aes(x = fileSize, y = resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  ggtitle("Modeling receiveTime ~ fileSize")
```




```
rm(predictions)
rm(resids)

model <- lm(receiveTime ~ log(chunkSize), data = train)
predictions <- add_predictions(validate, model)
R2(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.1430096
```

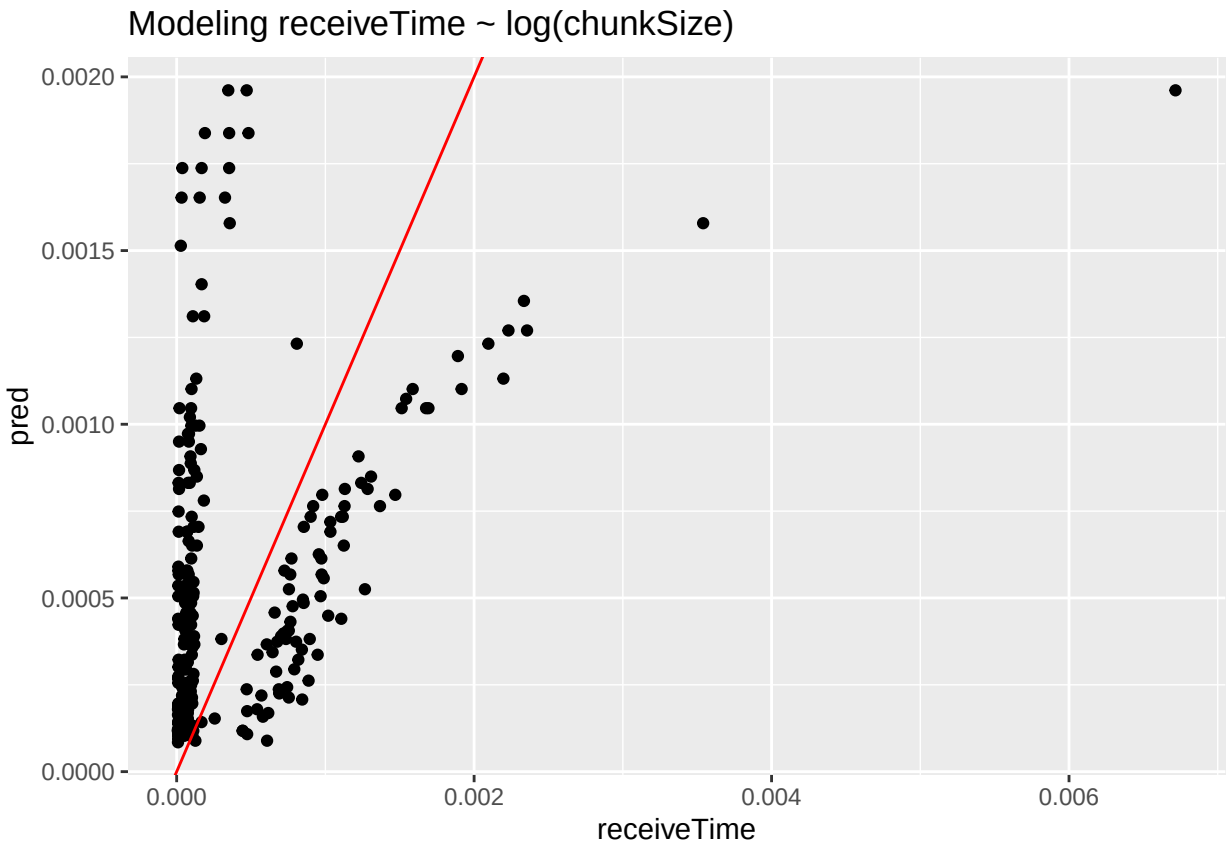
```
MAE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0004903447
```

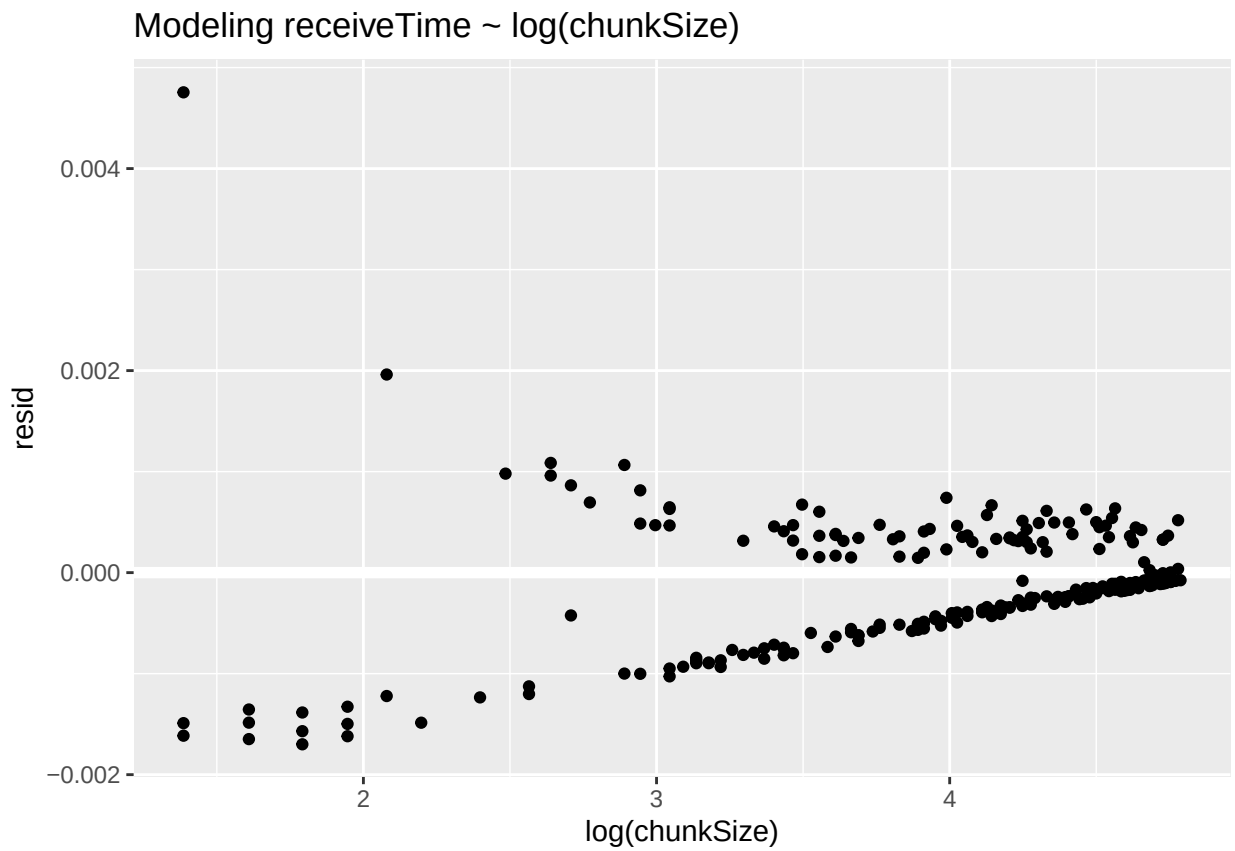
```
RMSE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0006771855
```

```
ggplot(data = predictions, mapping = aes(x = receiveTime, y = pred)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  ggtitle("Modeling receiveTime ~ log(chunkSize)")
```




```
resids <- add_residuals(validate, model)
ggplot(data = resids, mapping = aes(x = log(chunkSize), y = resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  ggtitle("Modeling receiveTime ~ log(chunkSize)")
```




```
rm(predictions)
rm(resids)

model <- lm(receiveTime ~ log(fileSize), data = train)
predictions <- add_predictions(validate, model)
R2(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.4314912
```

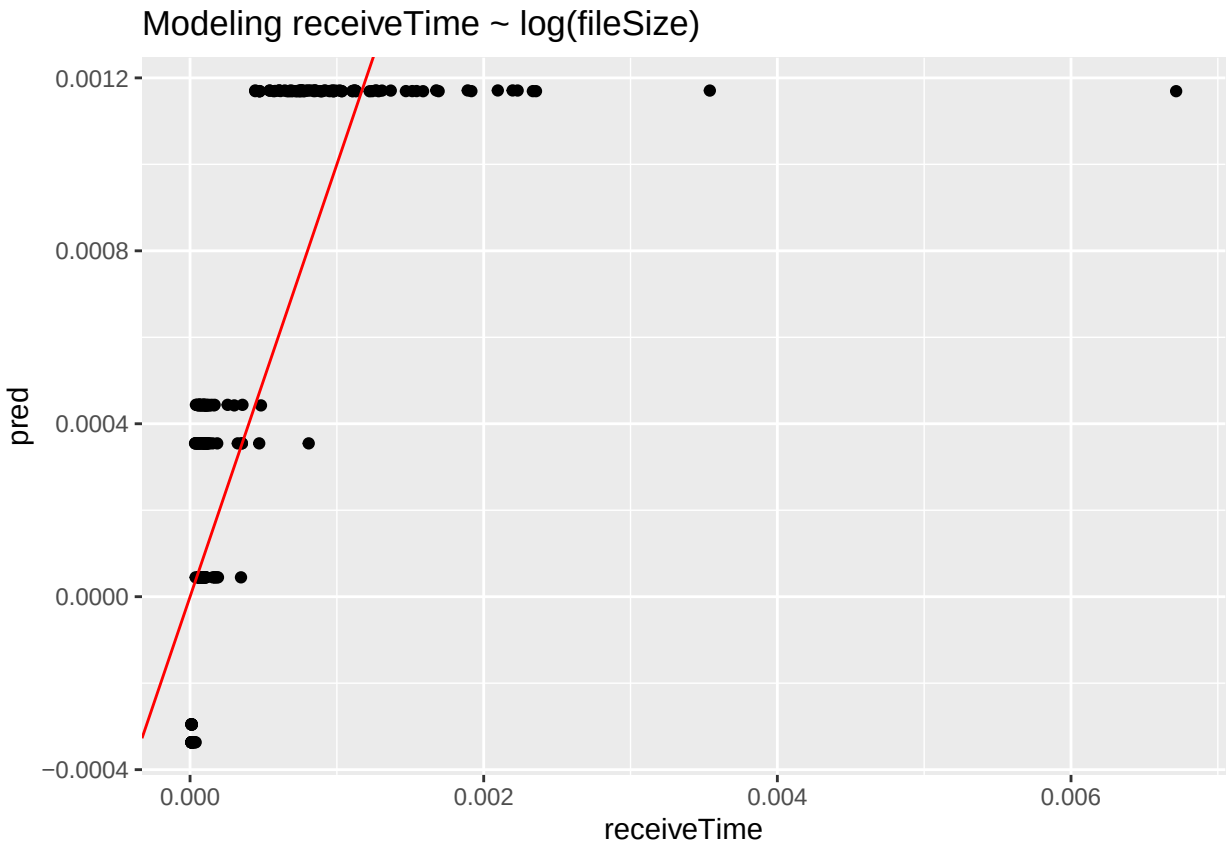
```
MAE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0003250275
```

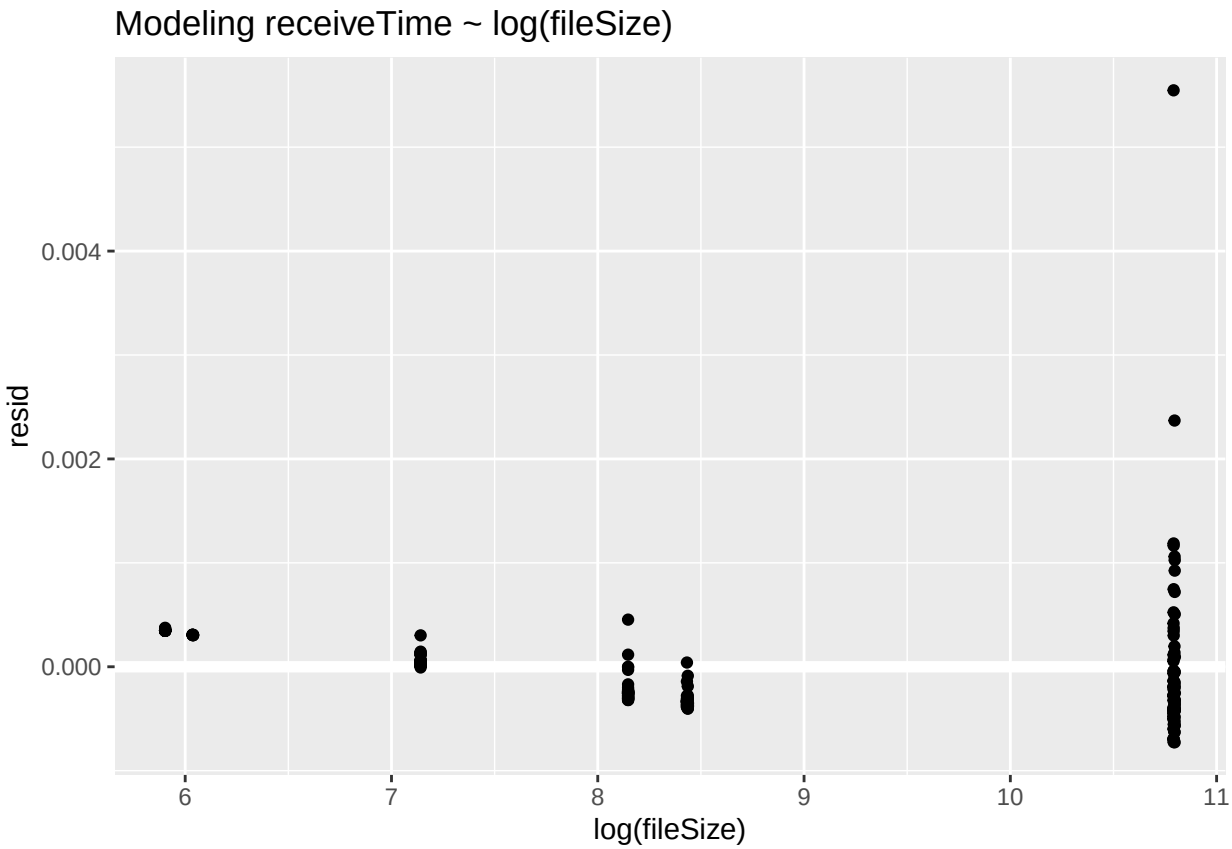
```
RMSE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.000531085
```

```
ggplot(data = predictions, mapping = aes(x = receiveTime, y = pred)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  ggtitle("Modeling receiveTime ~ log(fileSize)")
```




```
resids <- add_residuals(validate, model)
ggplot(data = resids, mapping = aes(x = log(fileSize), y = resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  ggtitle("Modeling receiveTime ~ log(fileSize)")
```




```
rm(predictions)
rm(resids)

model <- lm(receiveTime ~ chunkSize + fileSize, data = train)
predictions <- add_predictions(validate, model)
R2(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.5721617
```

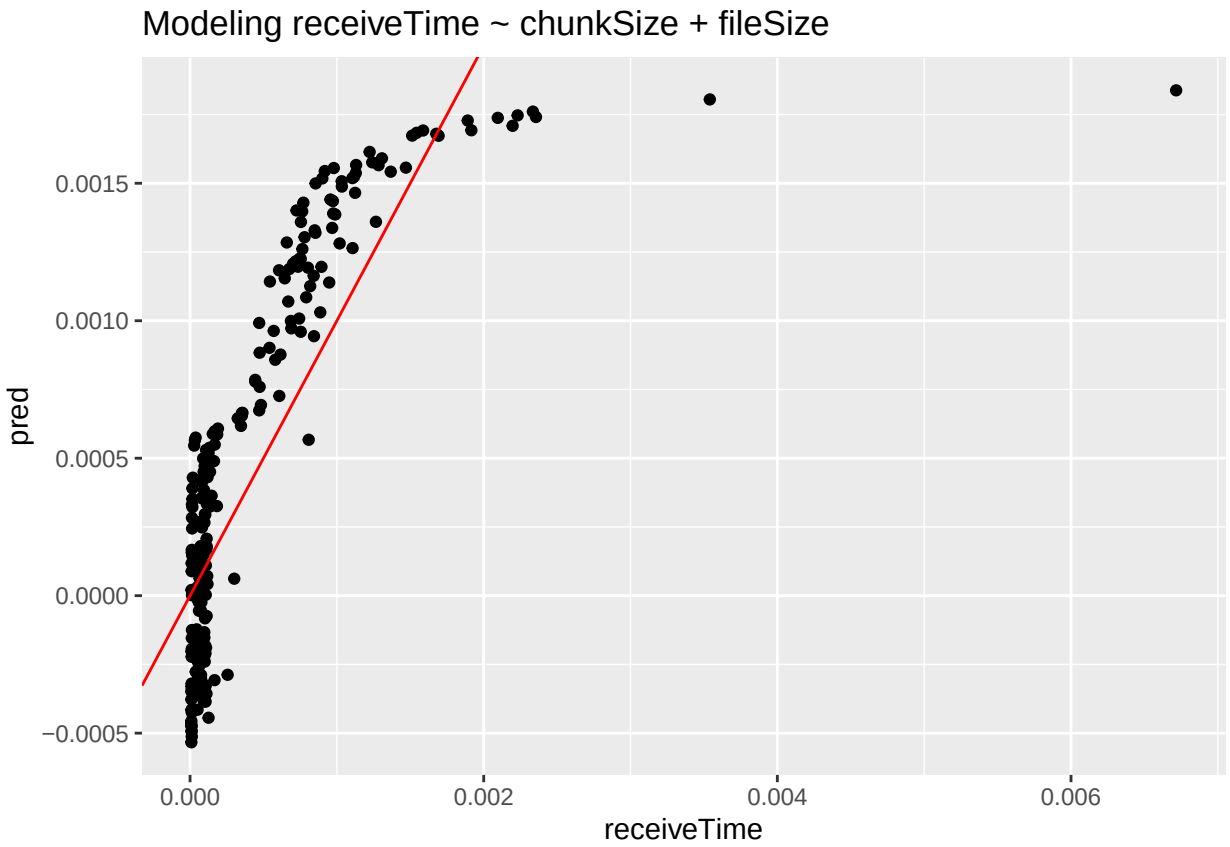
```
MAE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0003296183
```

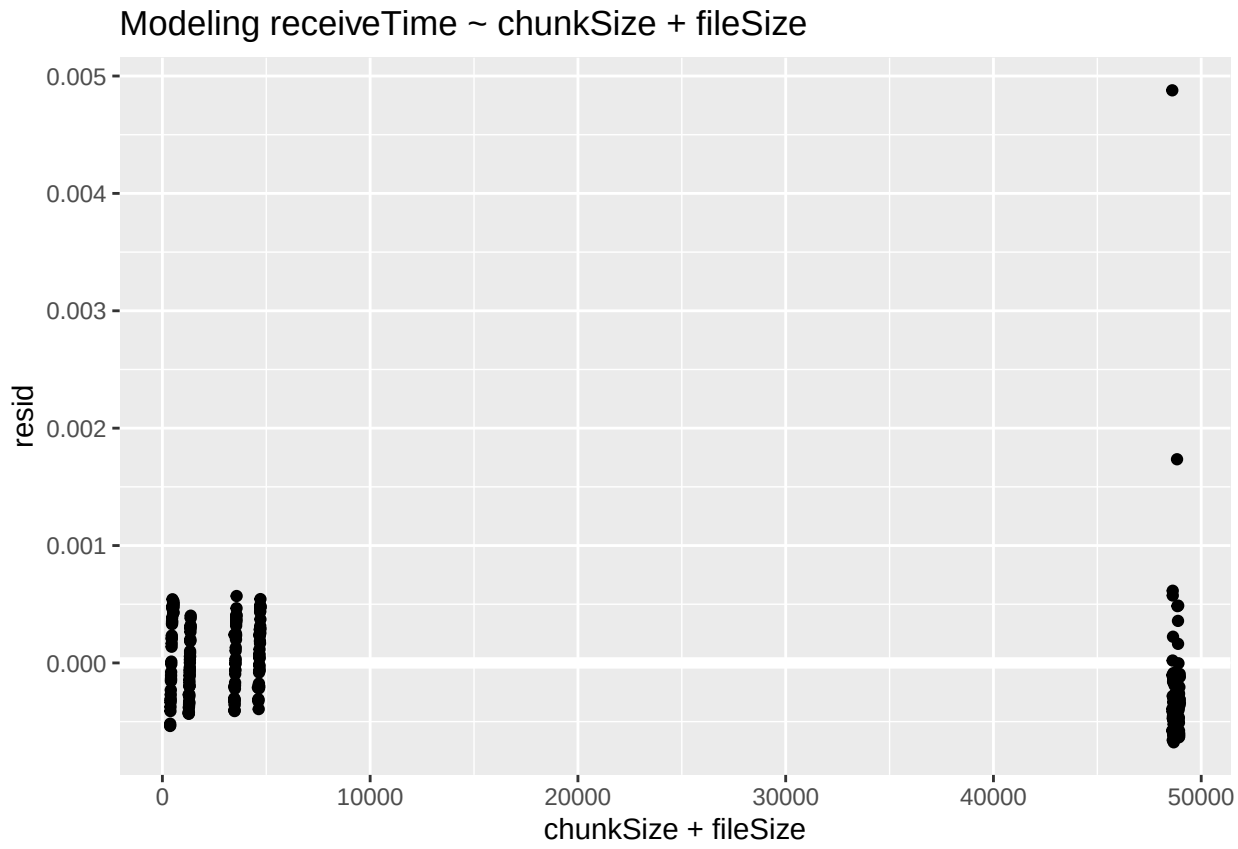
```
RMSE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0004822998
```

```
ggplot(data = predictions, mapping = aes(x = receiveTime, y = pred)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  ggtitle("Modeling receiveTime ~ chunkSize + fileSize")
```




```
resids <- add_residuals(validate, model)
ggplot(data = resids, mapping = aes(x = chunkSize + fileSize, y = resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  ggtitle("Modeling receiveTime ~ chunkSize + fileSize")
```




```
rm(predictions)
rm(resids)

model <- lm(receiveTime ~ chunkSize * fileSize, data = train)
predictions <- add_predictions(validate, model)
R2(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.7070837
```

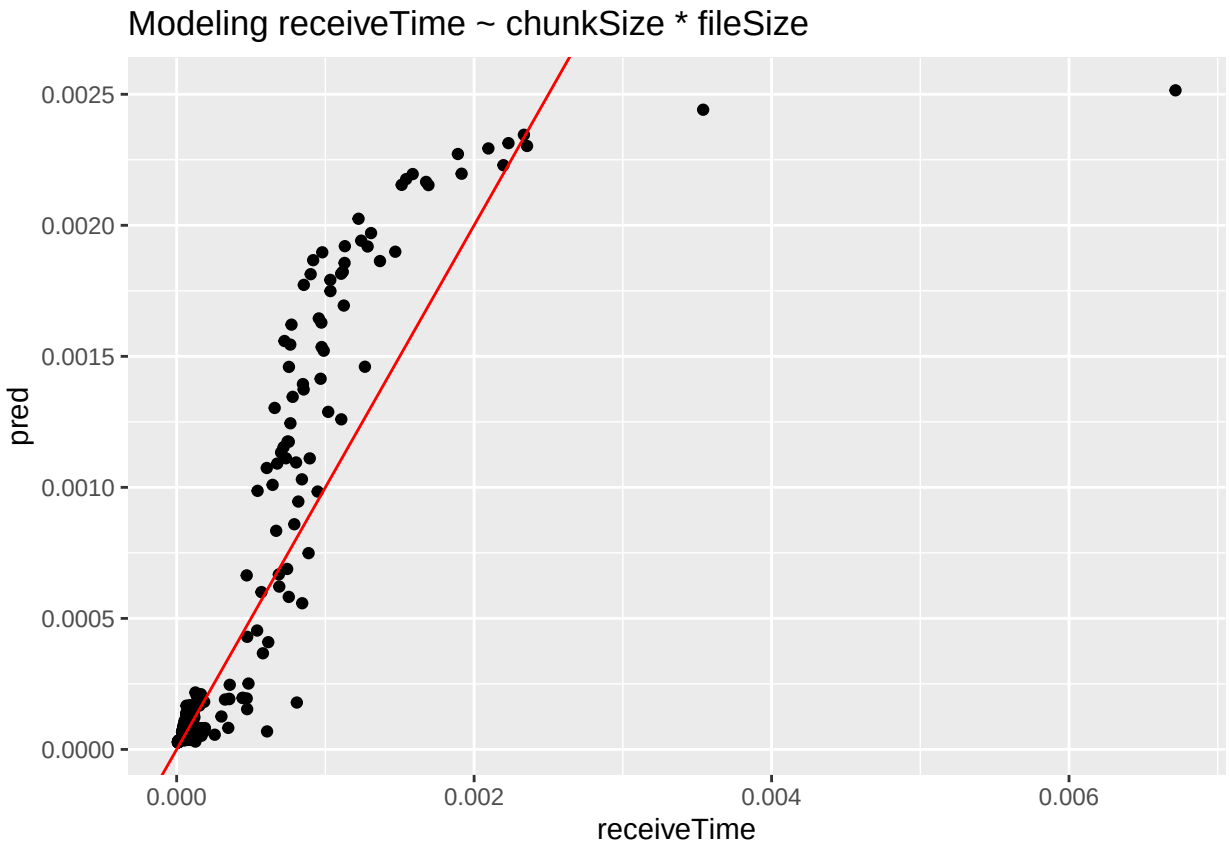
```
MAE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0001937808
```

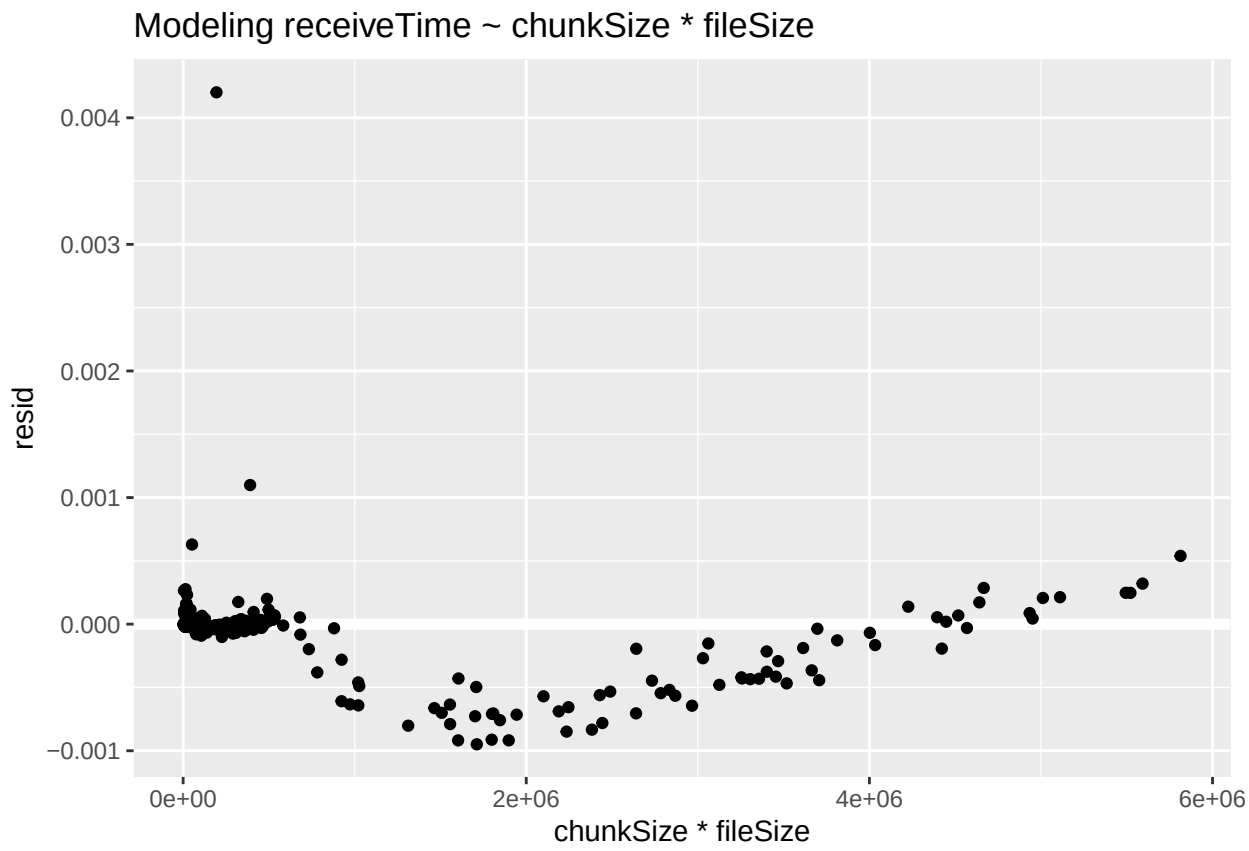
```
RMSE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0004114285
```

```
ggplot(data = predictions, mapping = aes(x = receiveTime, y = pred)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  ggtitle("Modeling receiveTime ~ chunkSize * fileSize")
```




```
resids <- add_residuals(validate, model)
ggplot(data = resids, mapping = aes(x = chunkSize * fileSize, y = resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  ggtitle("Modeling receiveTime ~ chunkSize * fileSize")
```




```
rm(predictions)
rm(resids)

model <- lm(receiveTime ~ fileSize * log(chunkSize), data = train)
predictions <- add_predictions(validate, model)
R2(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.856895
```

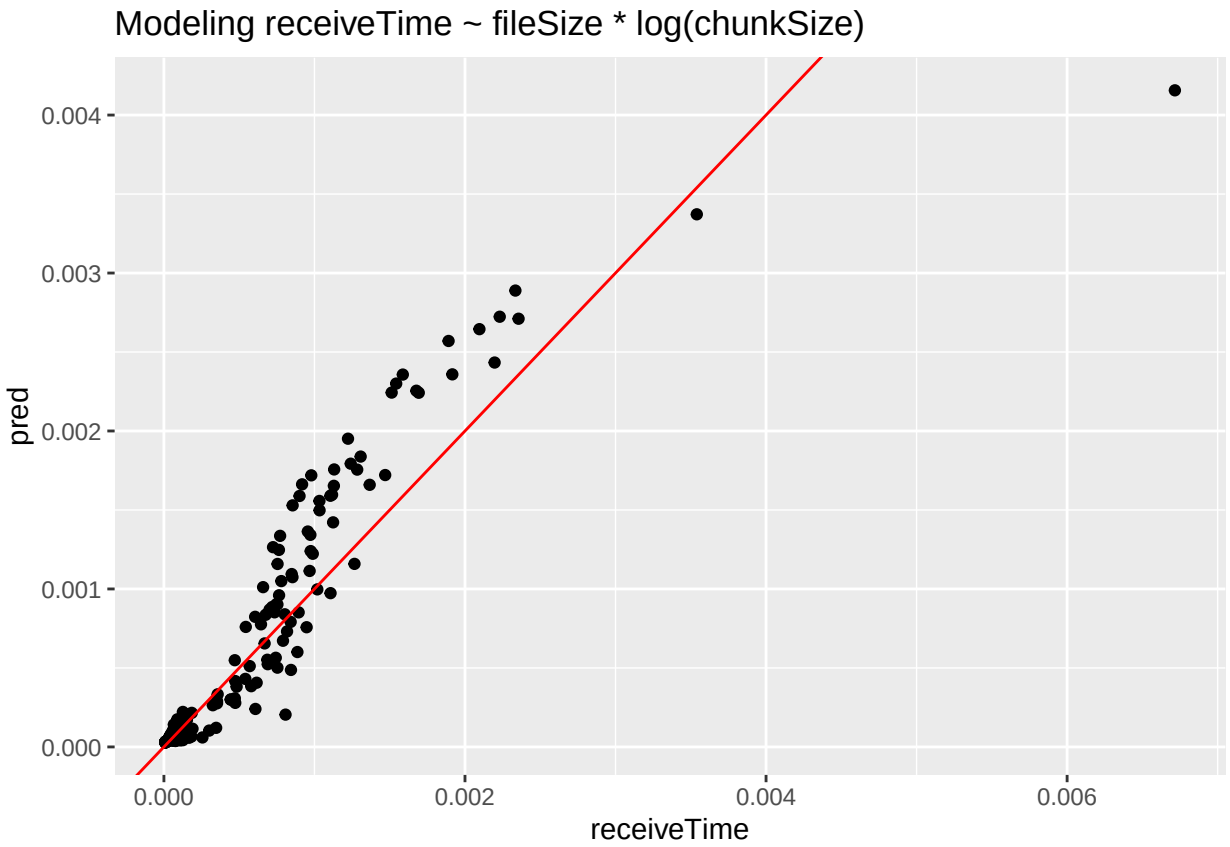
```
MAE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0001453502
```

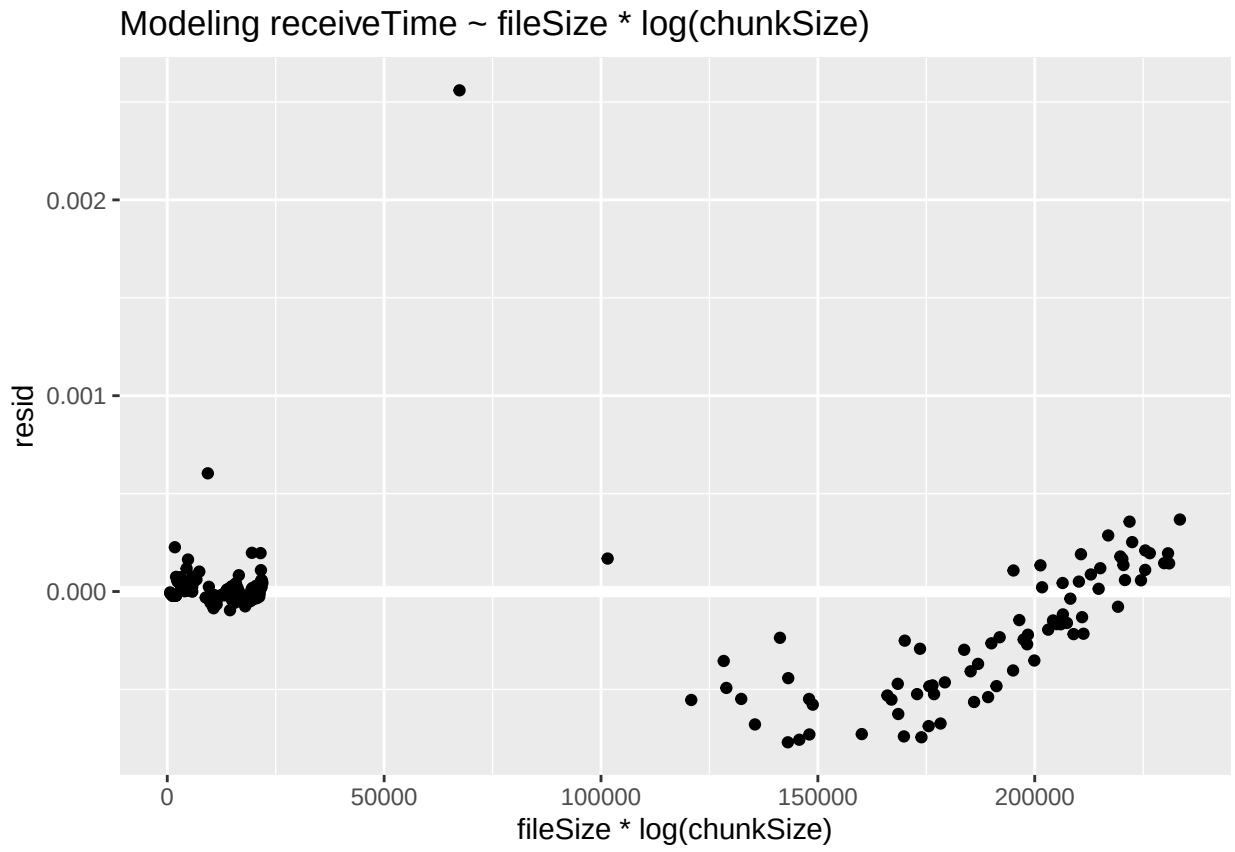
```
RMSE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0002876056
```

```
ggplot(data = predictions, mapping = aes(x = receiveTime, y = pred)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  ggtitle("Modeling receiveTime ~ fileSize * log(chunkSize)")
```




```
resids <- add_residuals(validate, model)
ggplot(data = resids, mapping = aes(x = fileSize * log(chunkSize), y = resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  ggtitle("Modeling receiveTime ~ fileSize * log(chunkSize)")
```




```
rm(predictions)
rm(resids)

model <- lm(receiveTime ~ chunkSize * log(fileSize), data = train)
predictions <- add_predictions(validate, model)
R2(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.6439318
```

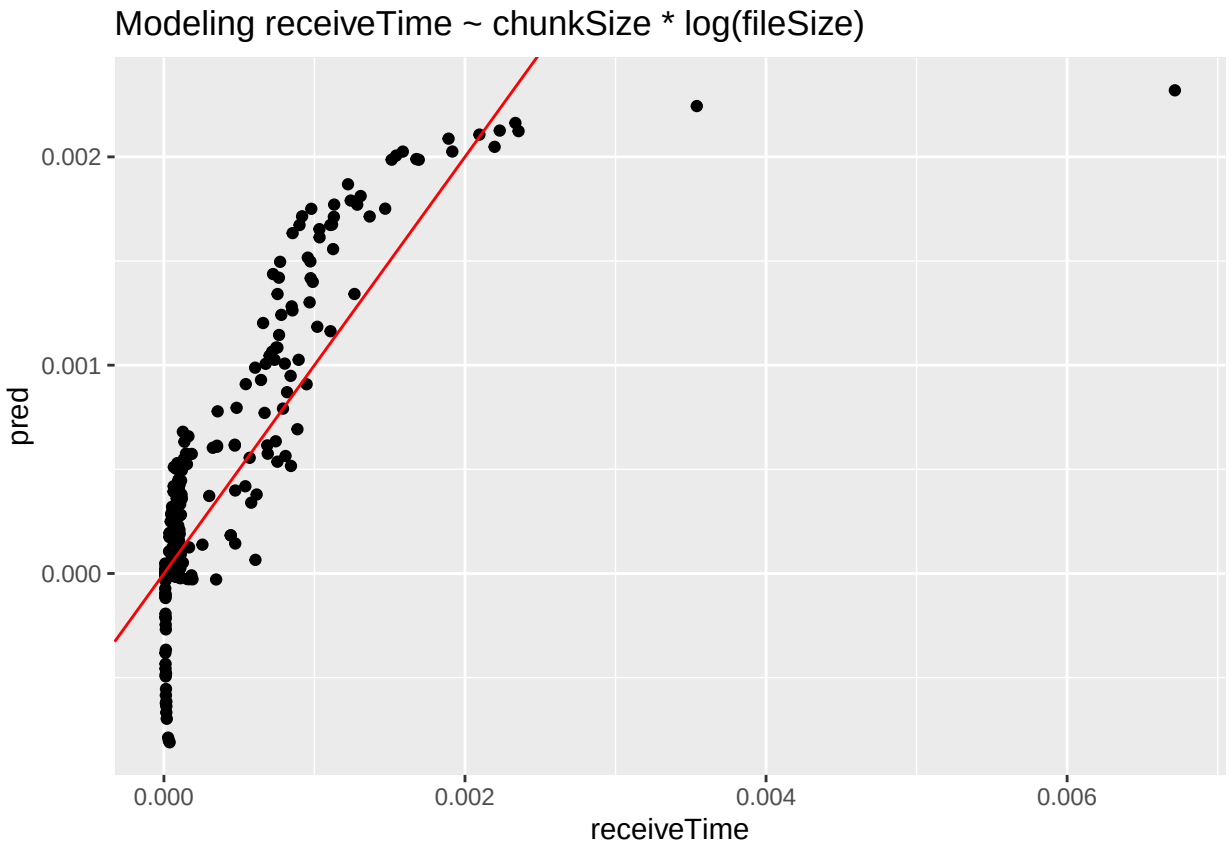
```
MAE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0002789324
```

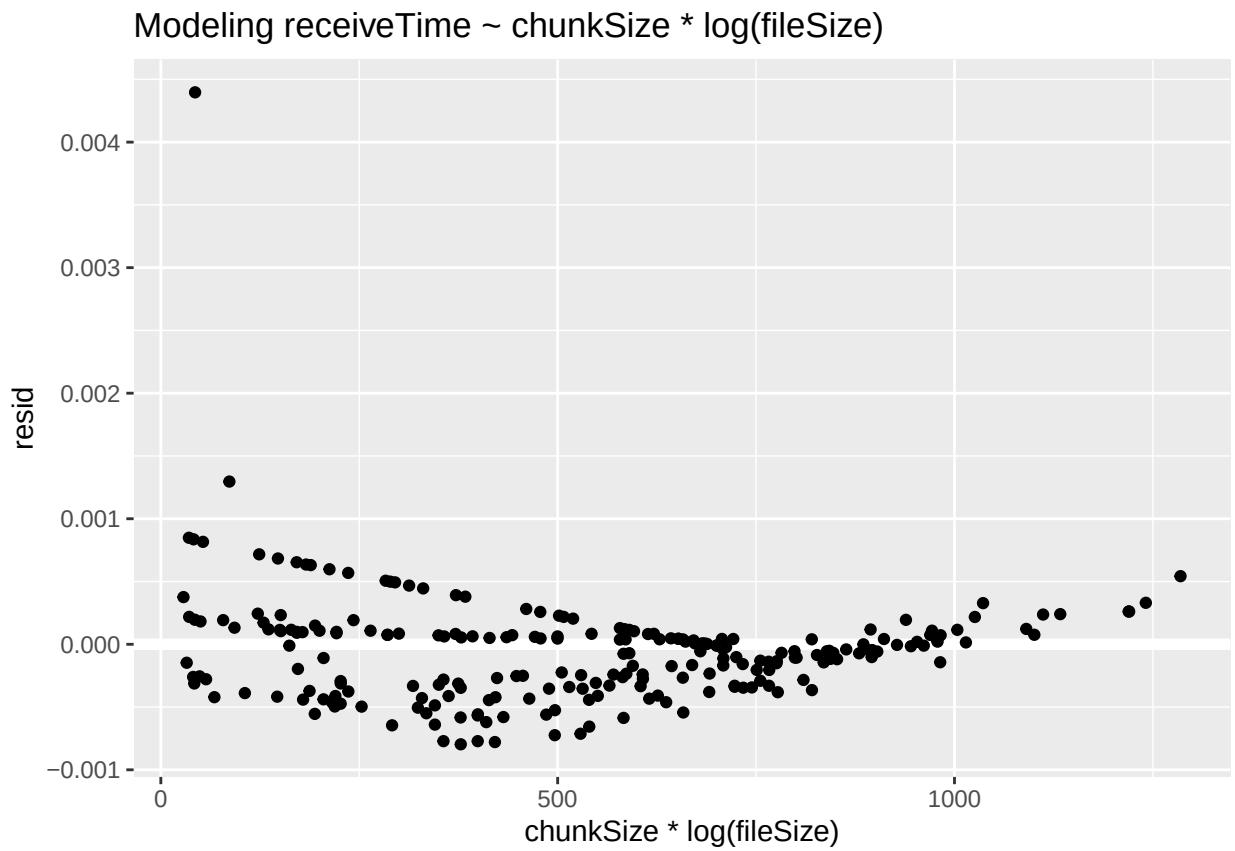
```
RMSE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0004455841
```

```
ggplot(data = predictions, mapping = aes(x = receiveTime, y = pred)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  ggtitle("Modeling receiveTime ~ chunkSize * log(fileSize)")
```




```
resids <- add_residuals(validate, model)
ggplot(data = resids, mapping = aes(x = chunkSize * log(fileSize), y = resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  ggtitle("Modeling receiveTime ~ chunkSize * log(fileSize)")
```




```
rm(predictions)
rm(resids)

model <- lm(receiveTime ~ log(chunkSize) + log(fileSize), data = train)
predictions <- add_predictions(validate, model)
R2(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.5472321
```

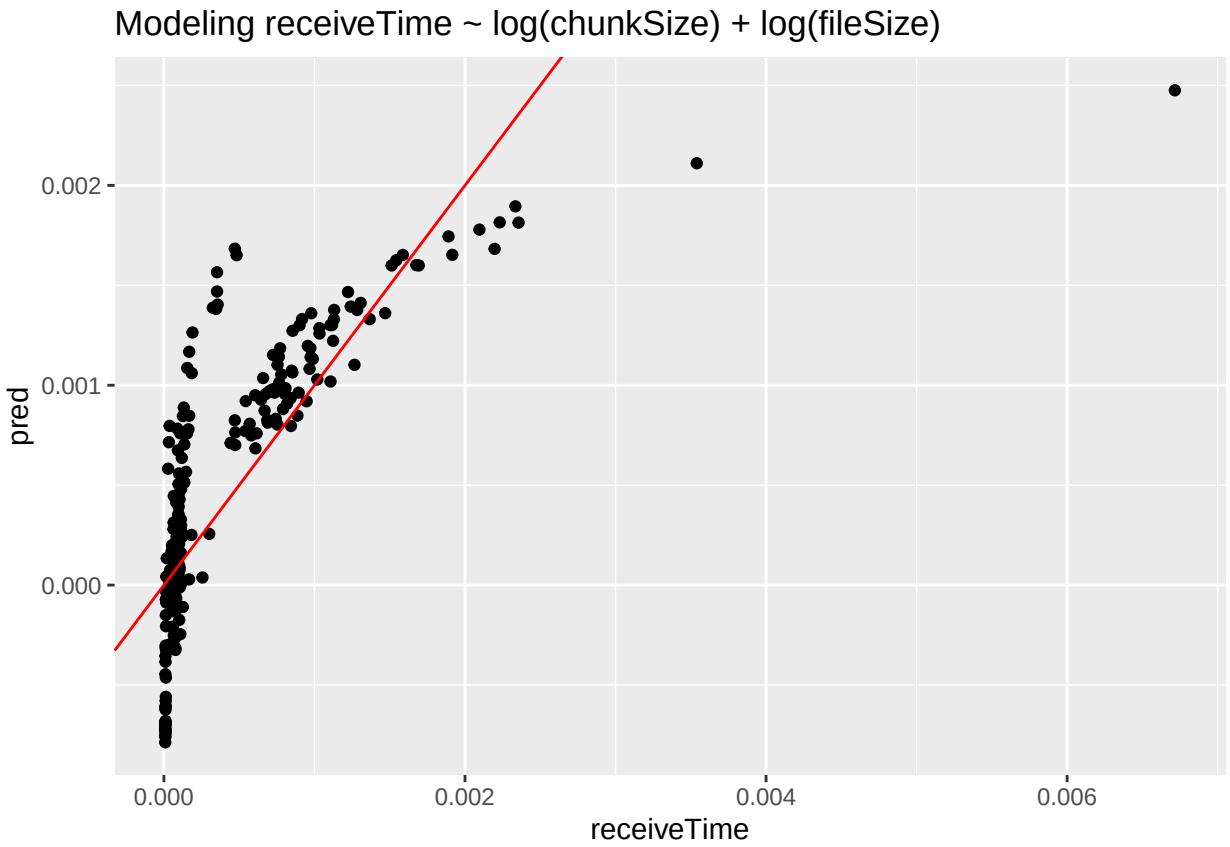
```
MAE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0003287526
```

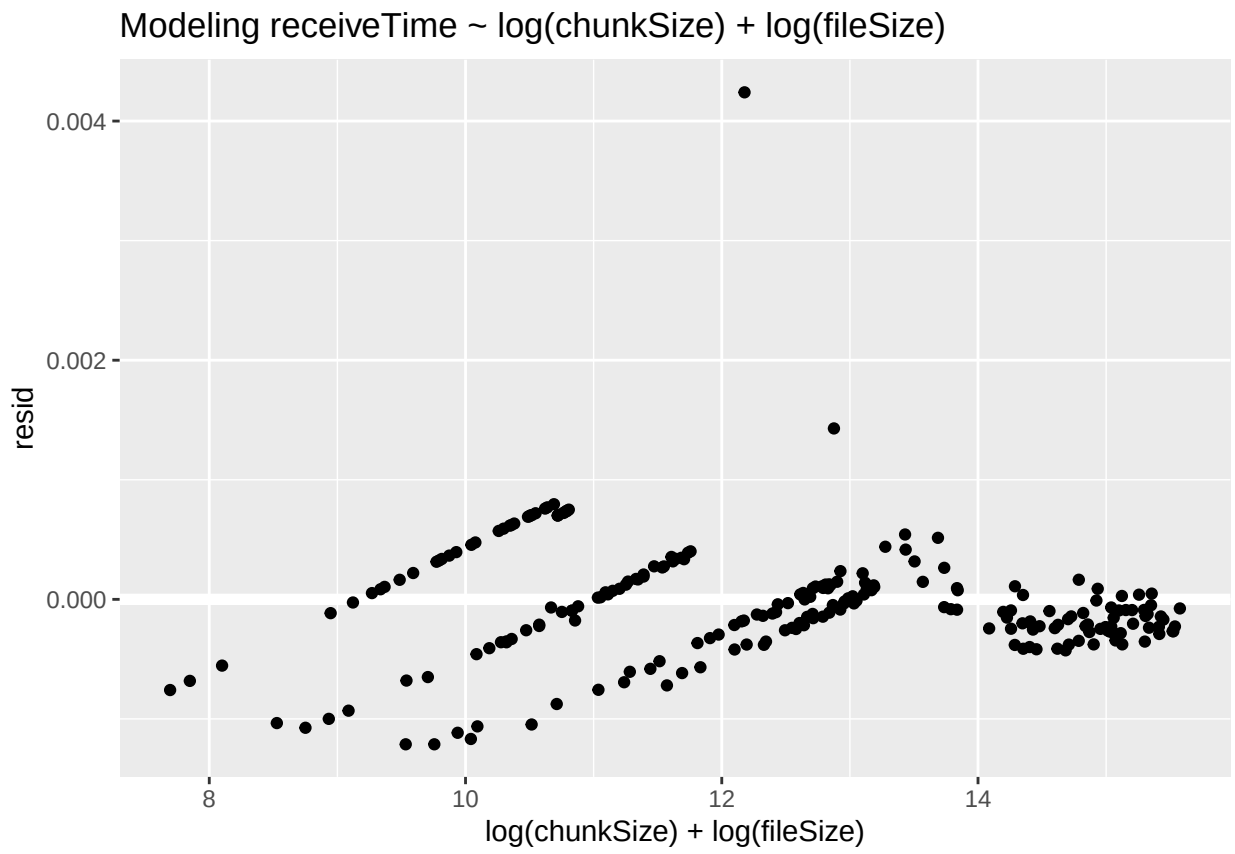
```
RMSE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0005003731
```

```
ggplot(data = predictions, mapping = aes(x = receiveTime, y = pred)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  ggtitle("Modeling receiveTime ~ log(chunkSize) + log(fileSize)")
```




```
resids <- add_residuals(validate, model)
ggplot(data = resids, mapping = aes(x = log(chunkSize) + log(fileSize), y = resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  ggtitle("Modeling receiveTime ~ log(chunkSize) + log(fileSize)")
```




```
rm(predictions)
rm(resids)

model <- lm(receiveTime ~ log(chunkSize) * log(fileSize), data = train)
predictions <- add_predictions(validate, model)
R2(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.7705344
```

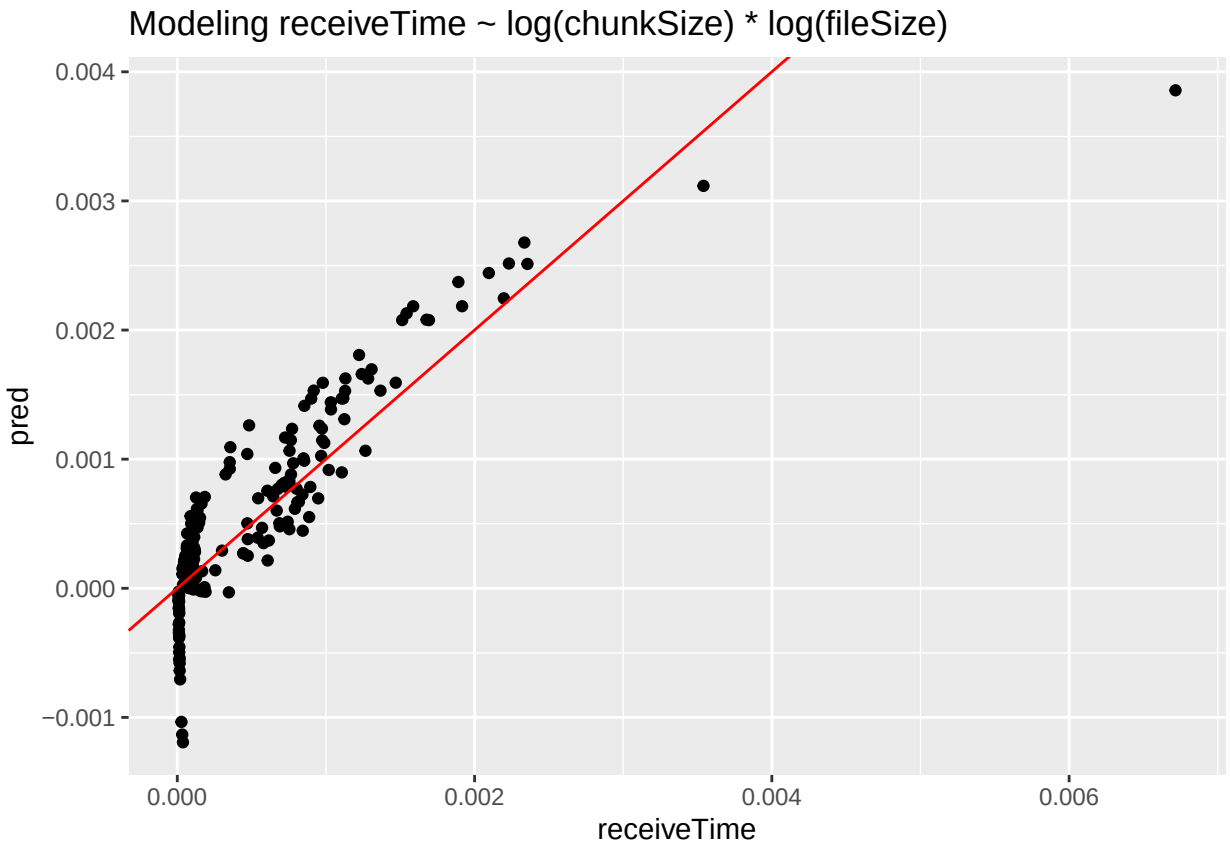
```
MAE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0002369809
```

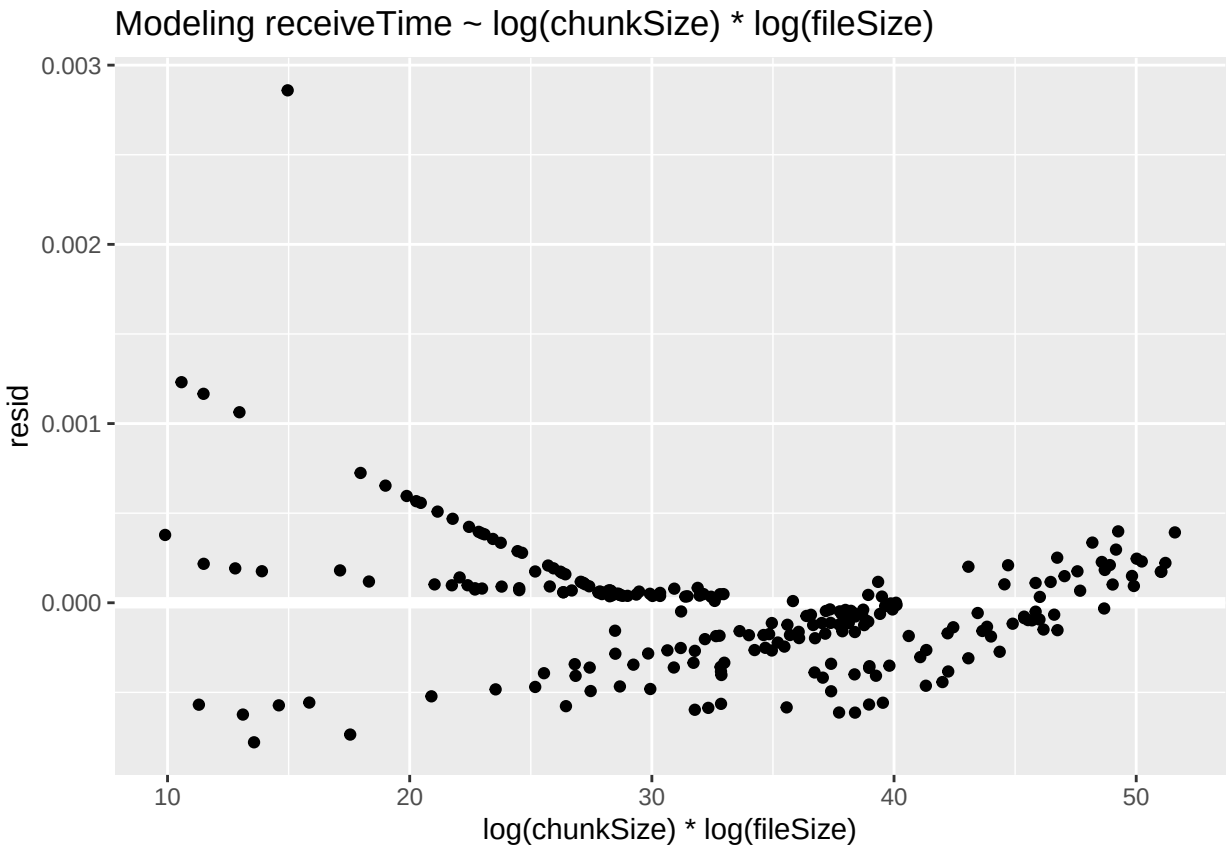
```
RMSE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0003576178
```

```
ggplot(data = predictions, mapping = aes(x = receiveTime, y = pred)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  ggtitle("Modeling receiveTime ~ log(chunkSize) * log(fileSize)")
```




```
resids <- add_residuals(validate, model)
ggplot(data = resids, mapping = aes(x = log(chunkSize) * log(fileSize), y = resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  ggtitle("Modeling receiveTime ~ log(chunkSize) * log(fileSize)")
```




```
rm(predictions)
rm(resids)
```

From reviewing all of the above model comparisons, the model with the best goodness of fit values, prediction graph views and residuals was:

receiveTime ~ fileSize * log(chunkSize)

Using this model let's try a cross validation training approach.

```
train.control <- trainControl(method = "cv", number = 5)
model <- train(receiveTime ~ fileSize * log(chunkSize), data = rest, method = "lm", trControl = train.c
summary(model)
```

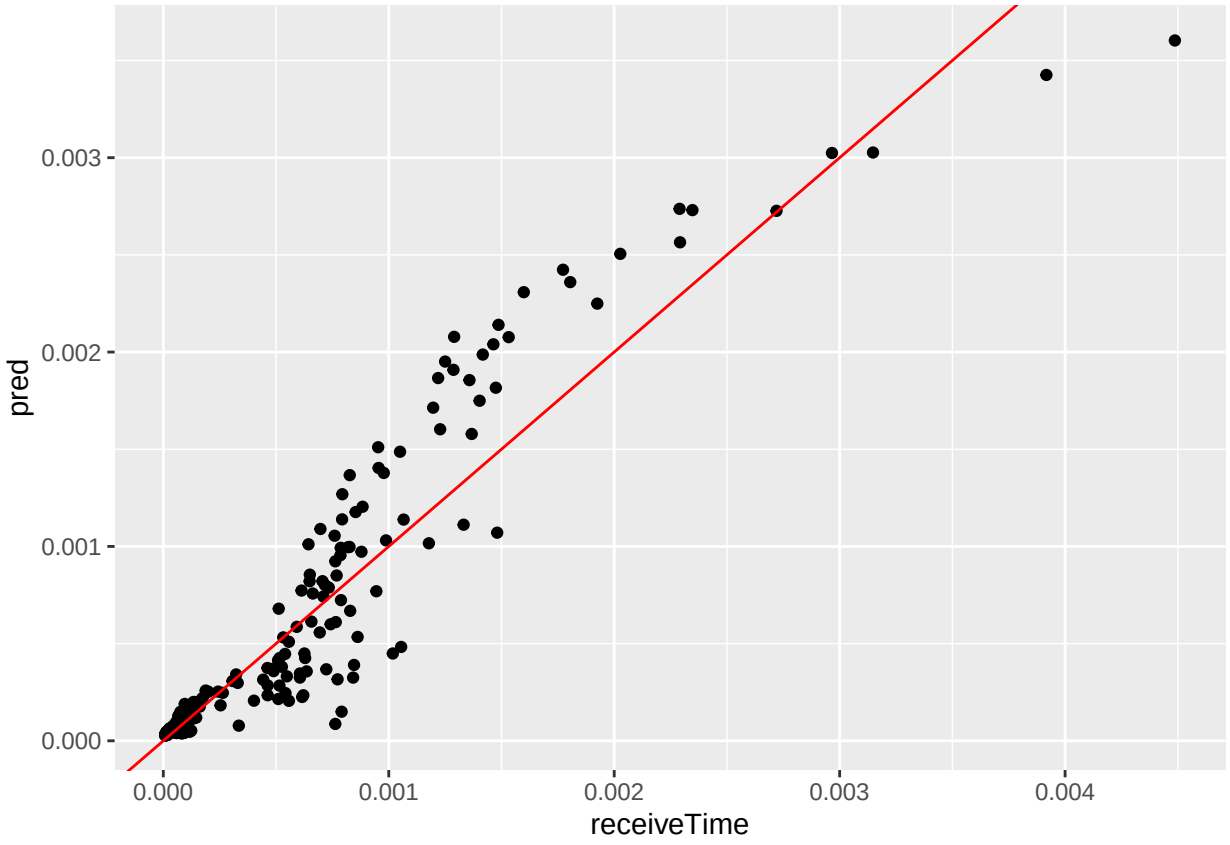
```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0008048 -0.0000505 -0.0000089  0.0000365  0.0056902
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 4.116e-05  8.152e-05   0.505    0.614
## fileSize                    1.151e-07  2.648e-09  43.485   <2e-16 ***
```

```
## `log(chunkSize)`          -2.981e-06  2.048e-05  -0.146    0.884
## `fileSize:log(chunkSize)` -2.328e-08  6.672e-10 -34.895   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0003678 on 932 degrees of freedom
## Multiple R-squared:  0.8265, Adjusted R-squared:  0.8259
## F-statistic:  1480 on 3 and 932 DF,  p-value: < 2.2e-16
```

Finally let's run this model against the test data and review the results.

```
predictions <- add_predictions(test, model)

ggplot(data = predictions, mapping = aes(x = receiveTime, y = pred)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red")
```




```
R2(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.896987
```

```
MAE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0001477194
```

```
RMSE(predictions$pred, predictions$receiveTime)
```

```
## [1] 0.0002437521
```

Modeling and Testing Observations:

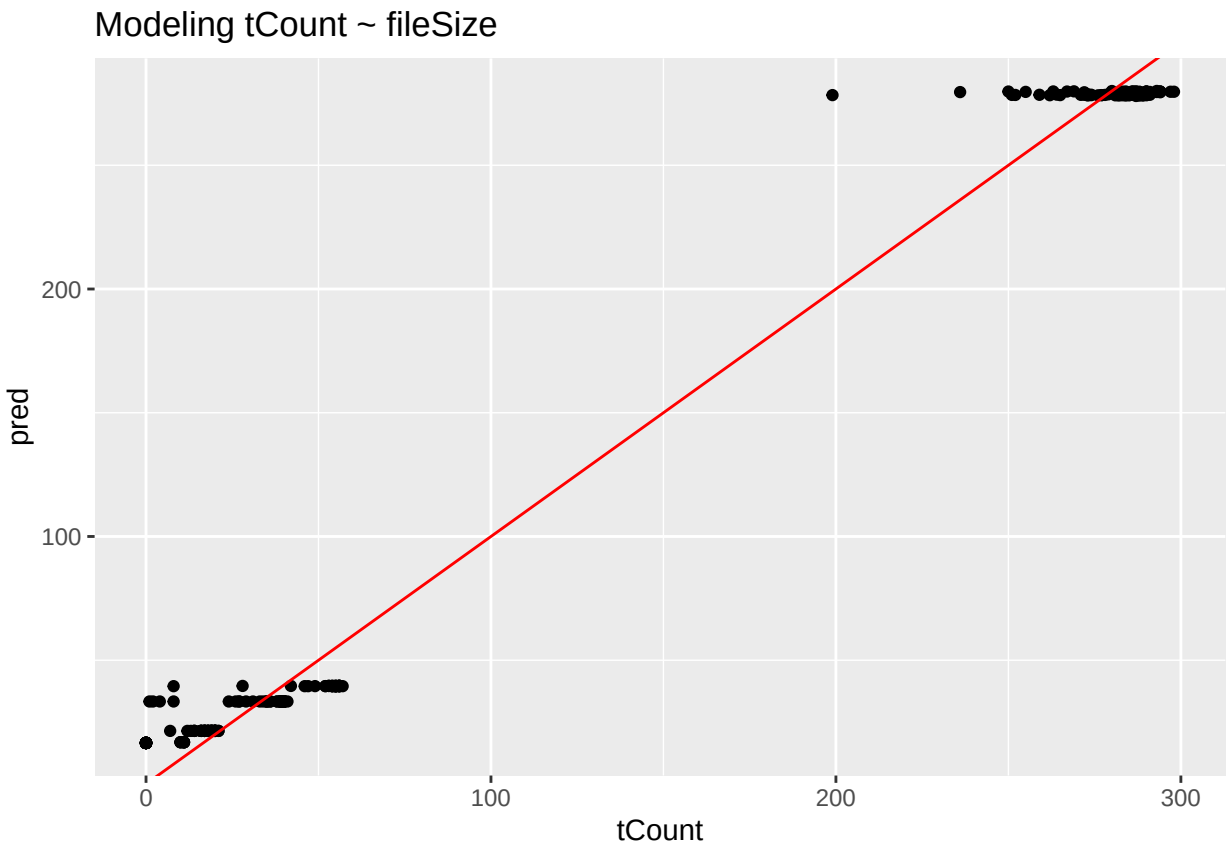
All goodness of fit measurements scored very well. This is the model I will use when rewriting future networking programs.

Revisiting Deliverable #2 Model

Now that we've developed an improved formula for predicting receiving time, let's see if we can use a similar formula for tag counts and compare this to the previous model made in Deliverable #2.

As a recap, here is the previous model: tCount ~ fileSize

```
rm(predictions)
model <- lm(tCount ~ fileSize, data = train)
predictions <- add_predictions(validate, model)
ggplot(data = predictions, mapping = aes(x = tCount, y = pred)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  ggtitle("Modeling tCount ~ fileSize")
```



```
R2(predictions$pred, predictions$tCount)
```

```
## [1] 0.9886148
```

```
MAE(predictions$pred, predictions$tCount)
```

```
## [1] 10.2067
```

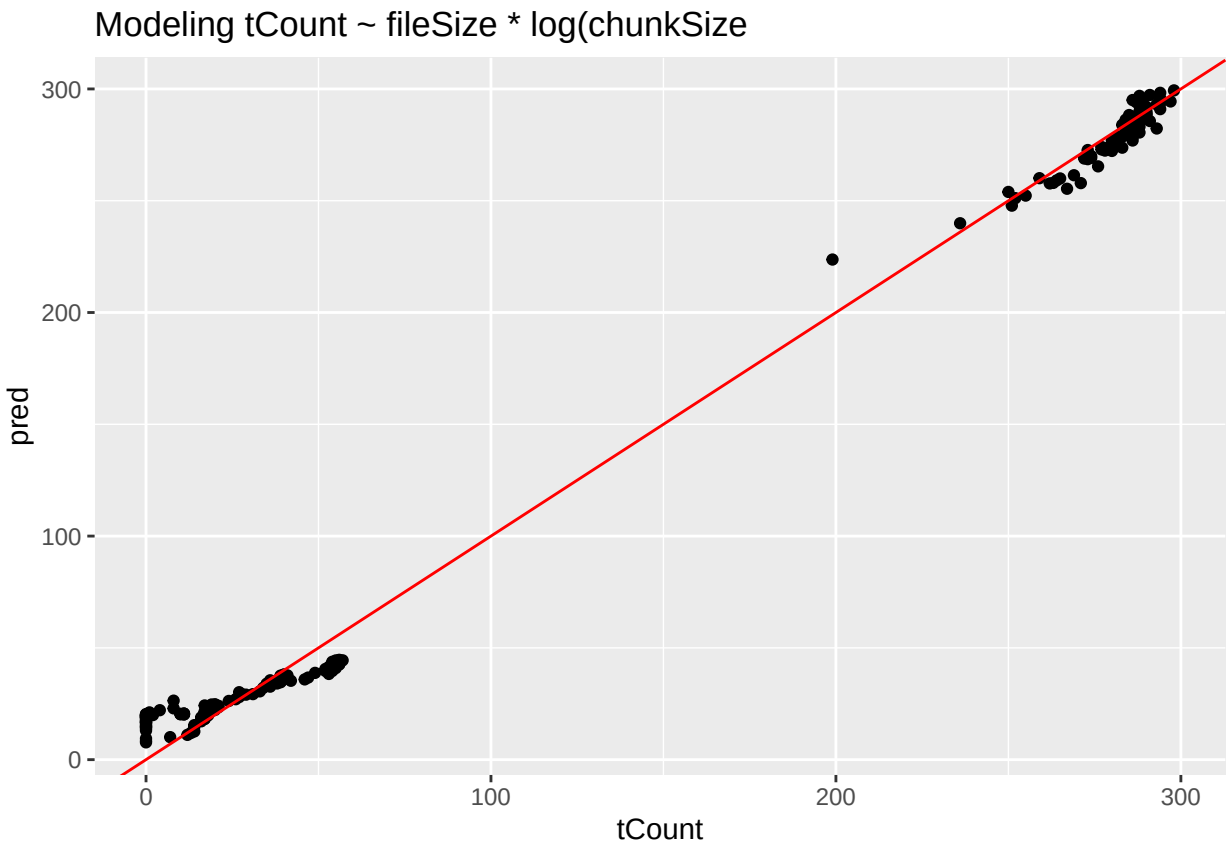
```
RMSE(predictions$pred, predictions$tCount)
```

```
## [1] 13.13917
```

```
rm(predictions)
```

Now I'll apply the same formula from above for receiving time to tag count:

```
model <- lm(tCount ~ fileSize * log(chunkSize), data = train)
predictions <- add_predictions(validate, model)
ggplot(data = predictions, mapping = aes(x = tCount, y = pred)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  ggtitle("Modeling tCount ~ fileSize * log(chunkSize")
```



```r
R2(predictions$pred, predictions$tCount)
```

```
## [1] 0.9947862
```

```r
MAE(predictions$pred, predictions$tCount)
```

```
## [1] 6.943947
```

```r
RMSE(predictions$pred, predictions$tCount)
```

```
## [1] 9.074035
```

As shown above, despite the very strong model from the second deliverable, by applying this new formula we have created an even more precise set of predictions. All goodness of fit variables have improved, and the prediction graph looks better.

Data Set 2

Unfortunately in the time allotted I was unable to build a satisfactory cross reference model of Data Set 2 Wireshark captures with Data Set 1 raw experimental output. I will continue to work on this and if there is more progress prior to the final report then I will include this.

On a more positive note, I reviewed the previous deliverable with my Networking professor at CSU Chico (Dr. Zhaohong Wang) and he was very excited about the results. After reviewing these results along wth my experimental code, he offered me a research assistant position in the spring to analyze and suggest improvements for network traffic design for a local emergency broadcast system. I am extraordinarily thankful for this opportunity and I owe it to what I learned in CSCI 385. I plan to continue both this final project for Data Science to incorporate Wireshark JSON output and apply my new knowledge of R in my future projects.

Data Science Questions

How does the size of chunk parsing impact response time? Using chunk size along with file size we were able to create a fair model to predict response time. The implication for this is that we should be able to incorporate this knowledge to improve processing performance.

Larger chunk sizes cause an increased demand in both processing speed and memory requirements. The default processing chunk size buffer for many routers and PC network adapters is 512. The file size value is included in the http header, and now that we have a better understanding of the relationship response time we can attempt to improve chunk size adjustments. In the future we can rewrite our experiment to capture fileSize in the headers so we can use it as an independent variable alongside chunk size. Including these and increasing the number of sources, destinations and observations we can revisit and attempt to improve our receiving time prediction model further.

Can we predict the number of tags that a website's root file based on the total file size? Our previous deliverable demonstrated this with file size alone, but with the addition of chunk size we can now predict this with even better precision.

Ethical Implications

Opacity: While the data collection is relatively straightforward and my code is available for inspection to recreate my experiment, an understanding of C programming and network functionality is required for scrutiny.

Scale: Although my experiment can be reran to include other protocols and a much larger number of websites, the R modeling here would need to be adjusted and reconsidered for such a larger scale.

Damage: Benchmarks results could be misinterpreted if compared to newer versions of protocols for web file transfer. HTTP/2 was introduced in 2015 and is currently used by 7% of web browsers. Unlike HTTP/1.0 and 1.1, this new protocol allows synchronous communication through web sockets for two way simulanteous communication. HTTP/3 is an upcoming standard that is currently only in use by Safari 14 introduced in September 2020, though over 10 million websites have implemented support for this newer standard. Unlike HTTP/2 and HTTP 1.0 and 1.1, HTTP/3 uses a new transport protocol called QUIC (general purpose transfer protocol) instead of the previous TCP (transfer control protocol). Future experiments using modern browser test could include HTTP/2 and HTTP/3 comparisons.