Preliminary Analysis of RTC Data

Matthew Knowles

2023-03-12

1: Data Cleaning

We begin by converting to a binary variable by severity. Accidents with a severity of 1 stay as such, and others are converted to a 0.

```
accidents <- accidents %>%
    mutate(bin_severity = ifelse(.data$accident_severity == 1, 1, 0))
```

We now reduce the size of the dataset to contain only variables we care about.

```
sub_accidents <- accidents %>%
    select(
        "accident_index",
        "bin_severity",
        "day_of_week",
        "road_type",
        "speed_limit",
        "time",
        "light_conditions",
        "weather_conditions",
        "road_surface_conditions"
    )

sub_vehicles <- vehicles %>%
    select(
        "accident_index",
        "sex_of_driver",
        "age_band_of_driver"
    )

sub_casualties <- casualty %>%
    select(
        "accident_index",
        "sex_of_casualty",
        "age_band_of_casualty"
    )

df_sub <- merge.data.frame(sub_vehicles, sub_casualties, by = "accident_index")
df_sub_unique <- unique(df_sub)
df <- merge.data.frame(sub_accidents, df_sub_unique, by = "accident_index")
```

Before dealing with other variables in the dataset, we deal with time on its own. The times provided in the dataset are characters, for example of the form "12:14". There are many unique times in the dataset, which would cause issues when fitting models as these would all be treated as individual factors. We therefore split

the 24 hour day into 6 chunks of 4 hours, as this dramatically reduces the number of factors, but also gives good information about the relationship between time of day and accident severity.

The first thing to do is build a function to parse the times by splitting the character before the colon, and converting that to an integer. A series of if-statements are then checked to put the time into the correct category. This isn't the most computationally efficient method, but it does the job. We then apply this function to every time observation to create a new column of data called "time_group", and drop the original time variable from the data so it doesn't cause problems later on.

```
# Split time into 6 chunks of four hours
parse_times <- function(time){
    if (!is.character(time)) {
        stop("Time is not character")
    }

    split_time <- stringr::str_split(time, ":", simplify = TRUE)
    hour <- as.integer(split_time[1])
    if (hour < 4) return(1)
    if (hour < 8 ) return(2)
    if (hour < 12) return(3)
    if (hour < 16) return(4)
    ifelse(hour < 20, return(5), return(6))
}

df <- df %>%
    mutate(time_group = purrr::map_int(time, parse_times)) %>%
    select(-time)
```

We want all variables, except speed limit, to be factors. To achieve this we mutate across all integer columns and change the type to factor. However, we need speed limit to stay as an integer to make the model adaptable to non-standard speed limits. The "bin_severity" column needs to be a factor, so we ensure this by calling the as.factor function once more just to ensure it is infact a factor. By viewing the head of the data as a tibble we can check that the columns are of expected type.

```
df <- df %>%
    mutate(across(where(is.integer), as.factor))
df$speed_limit <- as.integer(df$speed_limit) #Speed limit goes back to integer
df$bin_severity <- as.factor(df$bin_severity)
head(tibble(df))
```

```
## # A tibble: 6 x 13
##   accident_index bin_s~1 day_o~2 road_~3 speed~4 light~5 weath~6 road_~7 sex_o~8
##   <chr>          <fct>   <fct>   <fct>     <int> <fct>   <fct>   <fct>   <fct>
## 1 2020010219808  0       3       6             2 1       9       9       2
## 2 2020010220496  0       2       6             2 1       1       1       1
## 3 2020010228005  0       4       6             3 4       1       2       3
## 4 2020010228006  0       4       6             3 4       1       1       1
## 5 2020010228011  0       4       6             3 4       1       1       1
## 6 2020010228012  0       4       2             2 4       1       1       1
## # ... with 4 more variables: age_band_of_driver <fct>, sex_of_casualty <fct>,
## #   age_band_of_casualty <fct>, time_group <fct>, and abbreviated variable
## #   names 1: bin_severity, 2: day_of_week, 3: road_type, 4: speed_limit,
## #   5: light_conditions, 6: weather_conditions, 7: road_surface_conditions,
## #   8: sex_of_driver
```

Some observations of the data are -1. We don't want these in the data, so we trim the data down to remove any rows that contain -1 in any of the columns. This helps with run-time of the model as well due to the

size of the data before this is done. At this stage we also remove the accident index variable, as it was only needed for combining the three datasets, and isn't needed in fitting the GLM itself.

```
has.neg <- apply(df, 1, function(row) any(row == -1))
df <- df[-which(has.neg), ] %>%
    select(-accident_index)
```

Finally before fitting, split the data into two sets. A training and test set. The training set has been selected to contain a random sample of 80% of the original data, and the other 20% becomes the test set for later on. There is a of maths one could do to identify an optimal training data set size, but in this case we have selected an 80:20 split as a general rule of thumb.

```
df$id <- 1:nrow(df)
df_train <- df %>% sample_frac(0.8)
df_test <- anti_join(df, df_train, by = "id")
```

2: Fitting

We fit the glm to the training data. We specify that the response variable is binomial, and that we wish to use a logit link function.

```
fit <- glm(
    data = df_train,
    formula = bin_severity ~ .,
    family = binomial(link = "logit")
)
```

Let us take a look at the summary and plots of this model.

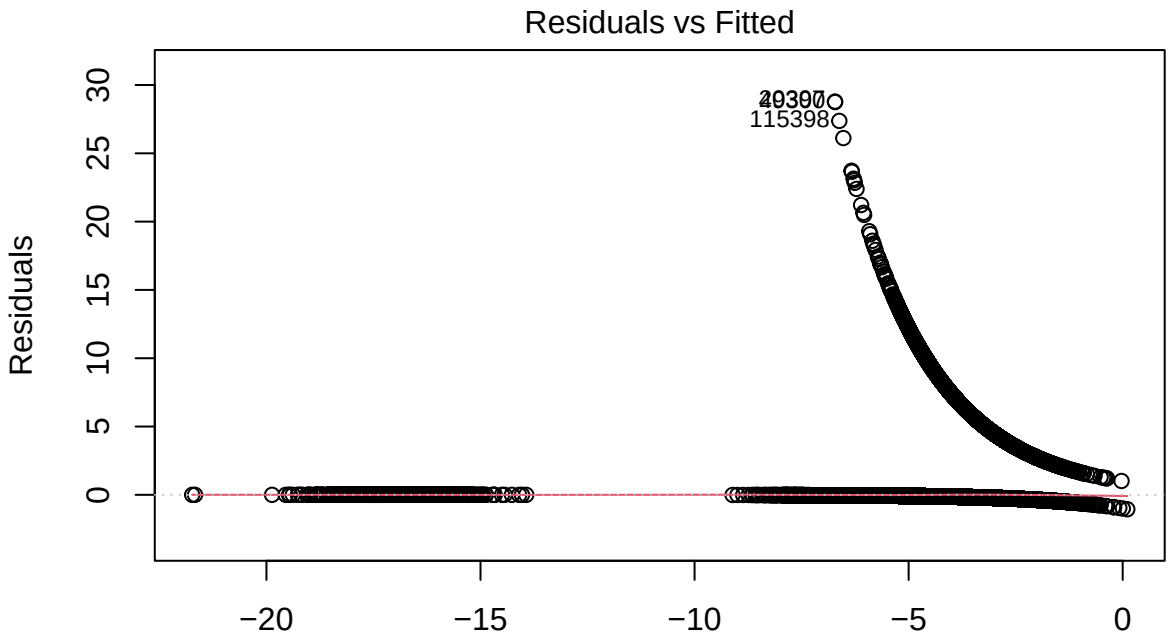
```
summary(fit)
```

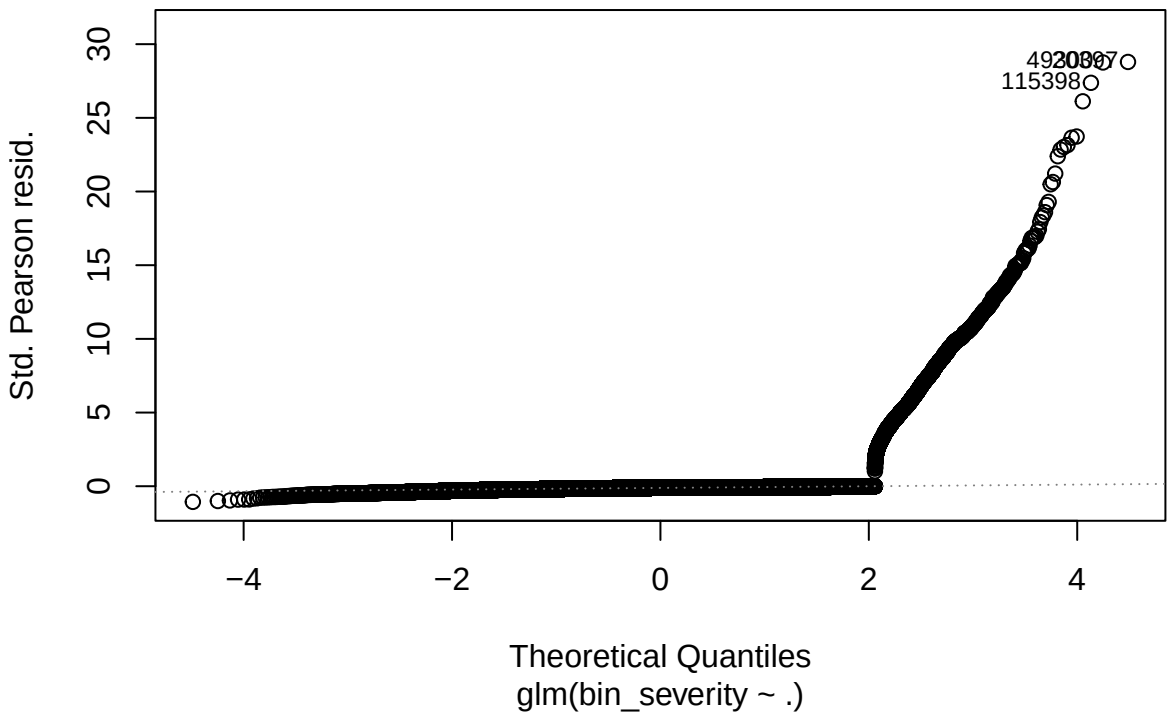
```
##
## Call:
## glm(formula = bin_severity ~ ., family = binomial(link = "logit"),
##     data = df_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2233  -0.2134  -0.1454  -0.1074   3.6664
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -1.795e+01  4.389e+02  -0.041 0.967380
## day_of_week2             -3.795e-01  7.877e-02  -4.817 1.45e-06 ***
## day_of_week3             -3.138e-01  7.682e-02  -4.084 4.42e-05 ***
## day_of_week4             -1.925e-01  7.401e-02  -2.601 0.009290 **
## day_of_week5             -6.602e-02  7.143e-02  -0.924 0.355372
## day_of_week6             -3.058e-01  7.397e-02  -4.134 3.57e-05 ***
## day_of_week7              4.584e-02  7.082e-02   0.647 0.517412
## road_type2               -3.719e-01  5.266e-01  -0.706 0.480057
## road_type3                7.472e-01  1.669e-01   4.476 7.59e-06 ***
## road_type6                1.366e+00  1.611e-01   8.474  < 2e-16 ***
## road_type7                2.341e-01  2.479e-01   0.944 0.344941
## road_type9                1.853e-01  4.778e-01   0.388 0.698138
## speed_limit               4.537e-01  1.560e-02  29.085  < 2e-16 ***
## light_conditions4         1.206e-02  7.376e-02   0.163 0.870152
```

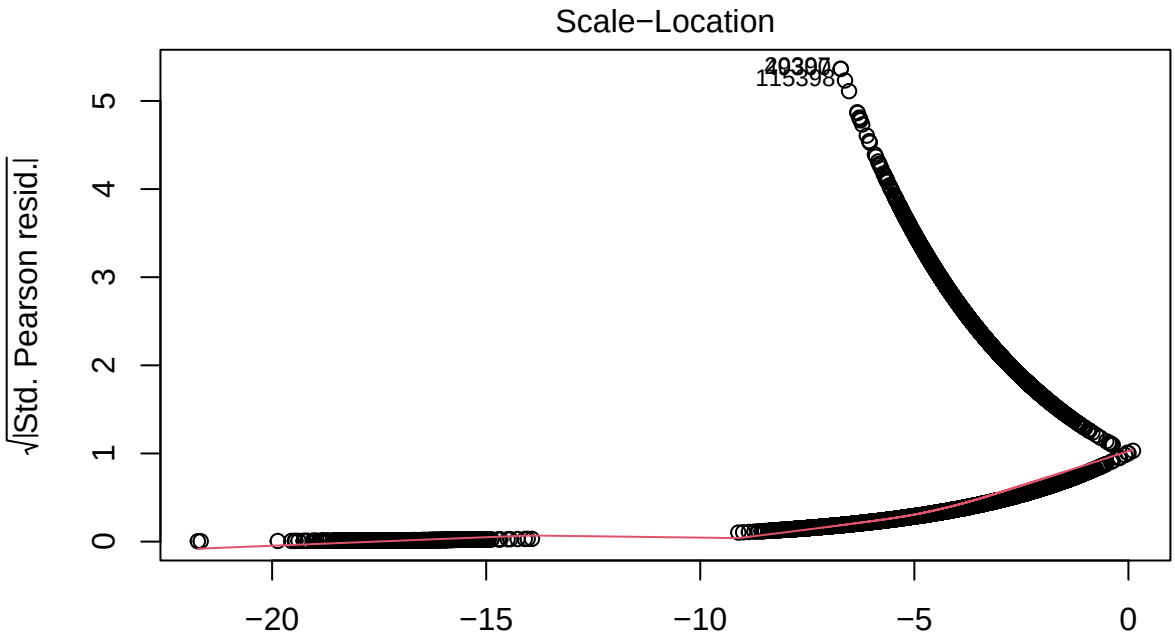
```
## light_conditions5         2.637e-01  1.959e-01   1.346 0.178280
## light_conditions6         3.896e-01  7.193e-02   5.416 6.10e-08 ***
## light_conditions7        -3.743e-01  2.029e-01  -1.845 0.065072 .
## weather_conditions2      -4.112e-01  7.487e-02  -5.493 3.96e-08 ***
## weather_conditions3       6.808e-01  3.748e-01   1.816 0.069331 .
## weather_conditions4       3.648e-01  1.239e-01   2.945 0.003232 **
## weather_conditions5      -1.948e-01  1.381e-01  -1.411 0.158338
## weather_conditions6      -1.234e+01  2.107e+02  -0.059 0.953286
## weather_conditions7       1.062e-01  1.825e-01   0.582 0.560627
## weather_conditions8       2.859e-01  1.175e-01   2.434 0.014934 *
## weather_conditions9      -5.755e-01  2.402e-01  -2.396 0.016569 *
## road_surface_conditions2  1.648e-01  5.118e-02   3.220 0.001283 **
## road_surface_conditions3 -2.447e+00  1.062e+00  -2.304 0.021224 *
## road_surface_conditions4 -6.798e-01  2.335e-01  -2.911 0.003606 **
## road_surface_conditions5  1.294e-02  3.510e-01   0.037 0.970587
## road_surface_conditions9 -1.089e+01  9.287e+01  -0.117 0.906640
## sex_of_driver2           -2.729e-01  5.166e-02  -5.282 1.28e-07 ***
## sex_of_driver3           -1.012e+00  3.056e-01  -3.313 0.000923 ***
## age_band_of_driver2       1.152e+01  4.389e+02   0.026 0.979057
## age_band_of_driver3       1.126e+01  4.389e+02   0.026 0.979526
## age_band_of_driver4       1.174e+01  4.389e+02   0.027 0.978665
## age_band_of_driver5       1.200e+01  4.389e+02   0.027 0.978180
## age_band_of_driver6       1.207e+01  4.389e+02   0.028 0.978059
## age_band_of_driver7       1.199e+01  4.389e+02   0.027 0.978208
## age_band_of_driver8       1.209e+01  4.389e+02   0.028 0.978020
## age_band_of_driver9       1.222e+01  4.389e+02   0.028 0.977786
## age_band_of_driver10      1.198e+01  4.389e+02   0.027 0.978233
## age_band_of_driver11      1.219e+01  4.389e+02   0.028 0.977840
## sex_of_casualty2         -3.781e-01  4.683e-02  -8.075 6.76e-16 ***
## sex_of_casualty9         -1.151e+01  1.696e+03  -0.007 0.994586
## age_band_of_casualty2    -5.451e-01  2.623e-01  -2.078 0.037718 *
## age_band_of_casualty3    -3.467e-01  2.297e-01  -1.509 0.131251
## age_band_of_casualty4    -3.924e-01  1.892e-01  -2.074 0.038078 *
## age_band_of_casualty5    -2.506e-01  1.816e-01  -1.380 0.167572
## age_band_of_casualty6    -2.840e-01  1.755e-01  -1.618 0.105679
## age_band_of_casualty7    -7.166e-02  1.770e-01  -0.405 0.685523
## age_band_of_casualty8    -5.534e-02  1.774e-01  -0.312 0.755056
## age_band_of_casualty9     1.021e-01  1.791e-01   0.570 0.568795
## age_band_of_casualty10    5.109e-01  1.856e-01   2.753 0.005897 **
## age_band_of_casualty11    1.163e+00  1.840e-01   6.319 2.63e-10 ***
## time_group2              -8.534e-01  1.081e-01  -7.895 2.91e-15 ***
## time_group3              -1.231e+00  1.090e-01 -11.300  < 2e-16 ***
## time_group4              -1.126e+00  1.048e-01 -10.745  < 2e-16 ***
## time_group5              -1.162e+00  9.296e-02 -12.498  < 2e-16 ***
## time_group6              -4.476e-01  9.097e-02  -4.920 8.66e-07 ***
## id                        1.862e-06  4.529e-07   4.111 3.94e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 26947  on 138720  degrees of freedom
## Residual deviance: 24014  on 138661  degrees of freedom
## AIC: 24134
```

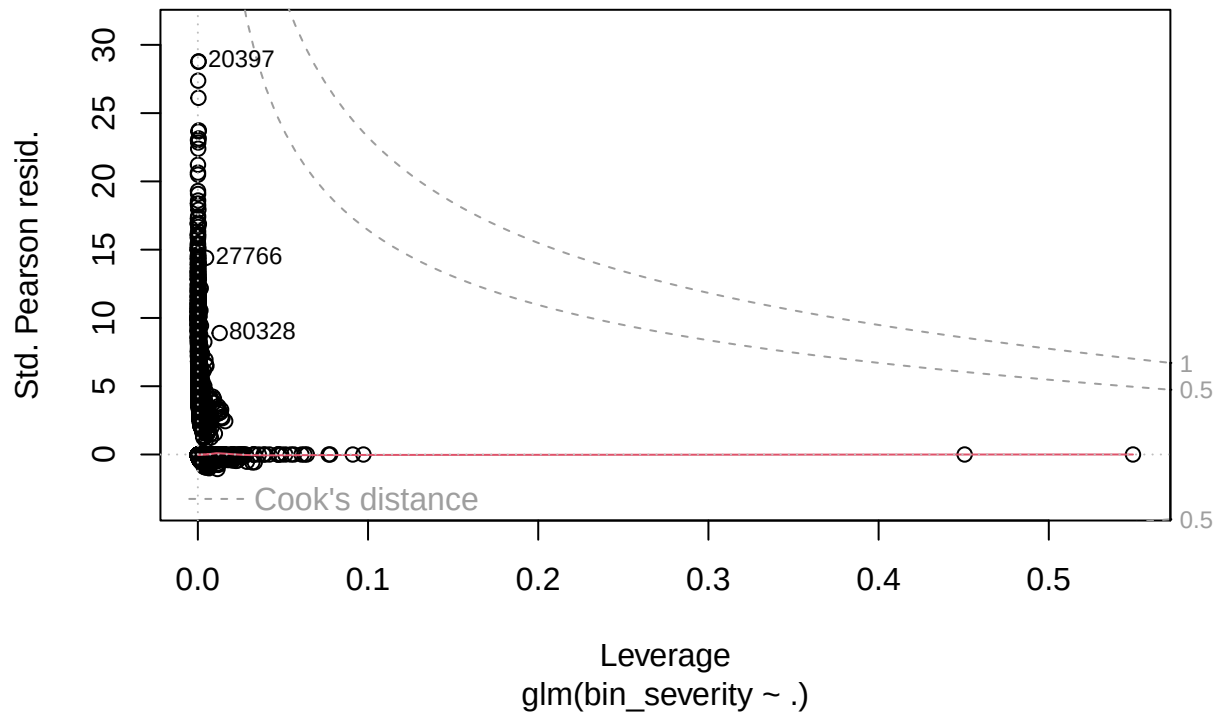
```
##
## Number of Fisher Scoring iterations: 15
```

```
plot(fit)
```



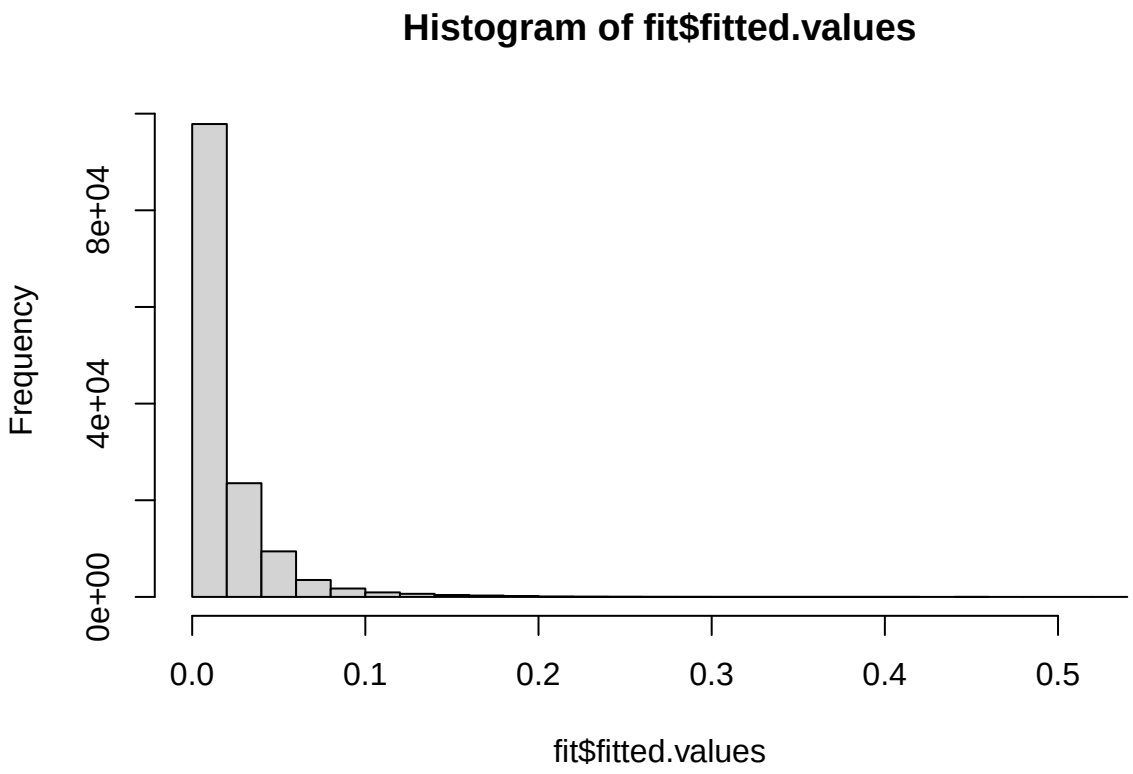






This histogram shows the distribution of fitted values from the training data.

```
hist(fit$fitted.values)
```



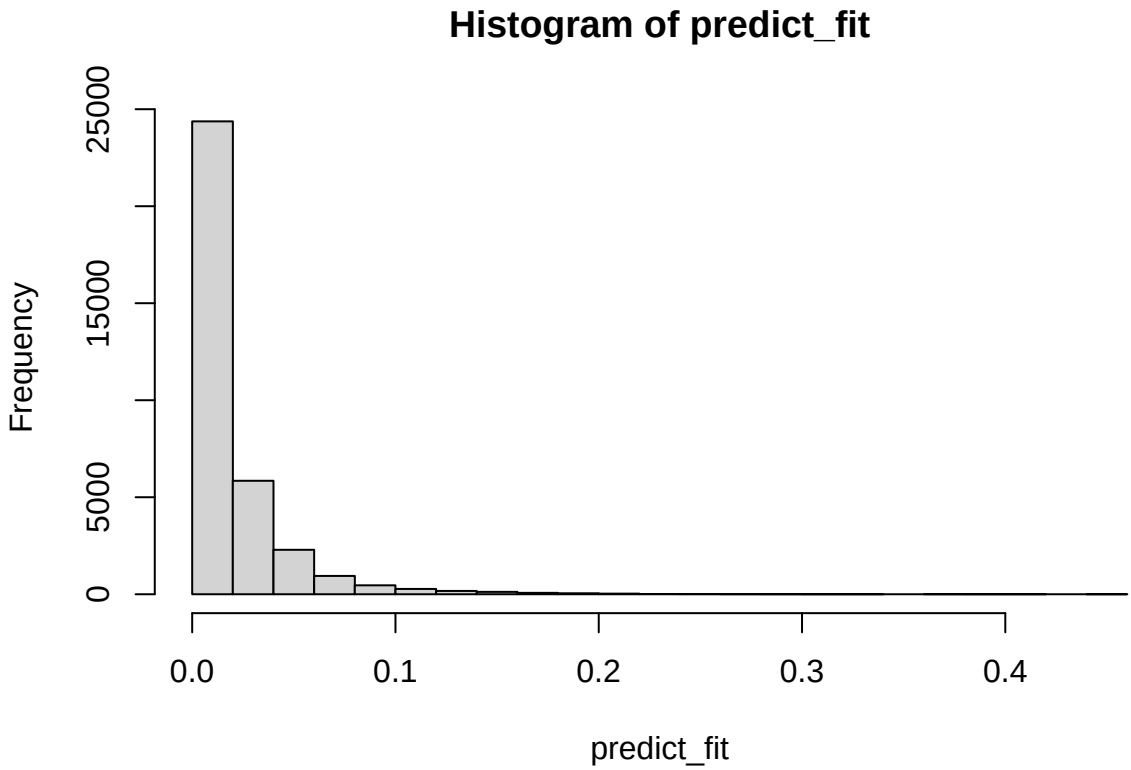
3: Predicting Unseen Values

We can use the predict function to see how well the model predicts the response on unseen data.

```
predict_fit <- predict(
    fit,
    newdata = df_test,
    type = "response"
)
```

This histogram shows the distribution of fitted values from the test data.

```
hist(predict_fit)
```



Notice the shape of this histogram and the previous one are identical.