Linear Model
Pro SX and MX

1 Setup

Let's connect to the database and get the gate drop times.

```
db_path <- './_data/gate_drop.db'
query_path <-  './notebooks/R_Notebooks/main_query.sql'
convenience_routines <- './notebooks/R_Notebooks/convenience_routines.R'
source(convenience_routines, local = knitr::knit_global())

src_db <- DBI::dbConnect(RSQLite::SQLite(),db_path)
main_query <- paste(readLines(query_path), collapse='\n')
main_dat <- DBI::dbGetQuery(conn = src_db, statement = main_query)
DBI::dbDisconnect(conn = src_db)
```

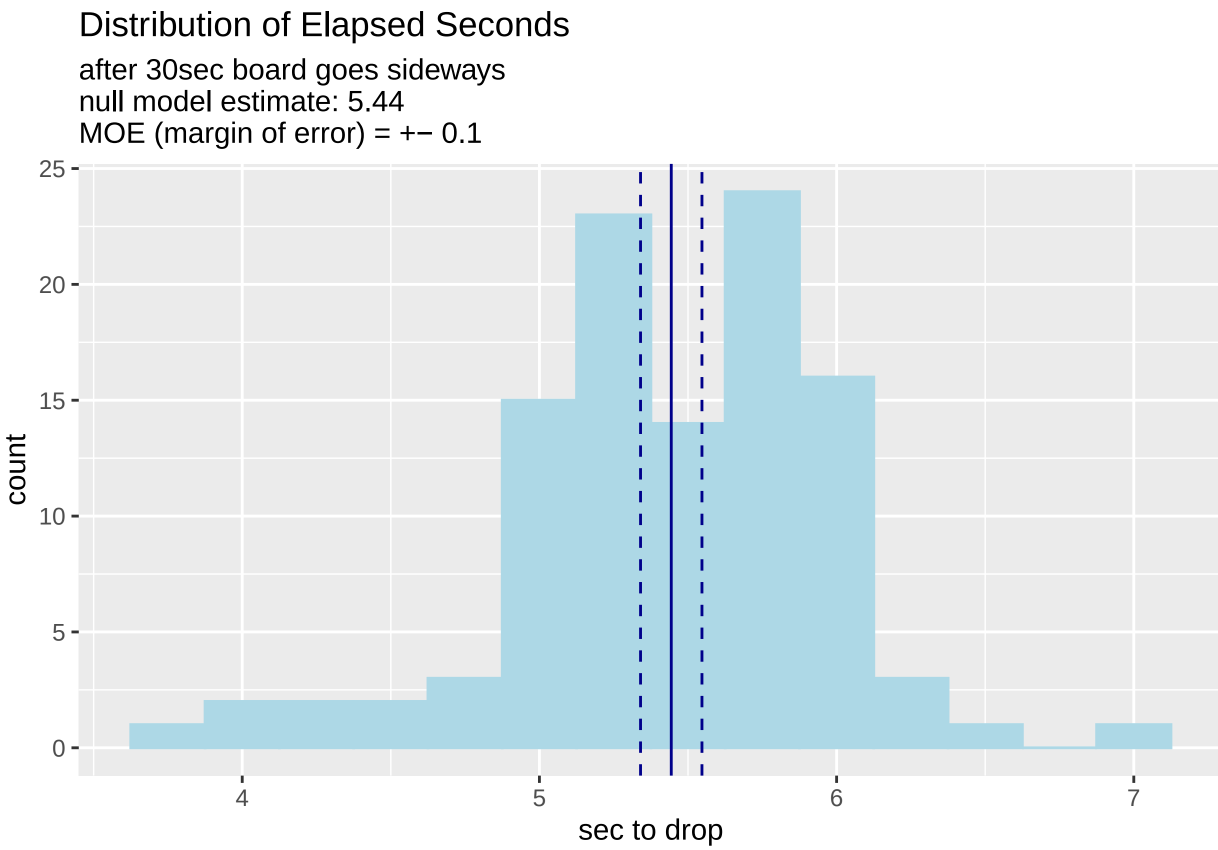
This analysis includes 107 separate observations from 14 Pro SX rounds.

2 Histograms

Now, let's do some fundamental analysis to get the margin of error for all gate drop times.

```
t_data <- main_dat |>  tibble::as_tibble()

# Drop extra columns we don't need to keep
t_data <- t_data |> dplyr::select (-one_of(c("row_id","Date", "Venue", "Round","Comments")))

# make the logistic variables numeric and replace NA with 0
t_data <- t_data |> dplyr::mutate_if(is.logical, as.numeric)
t_data <- t_data |> dplyr::mutate_all(funs(replace_na(.,0)))

t_moe_factor <- qt(0.025, nrow(t_data), lower.tail = FALSE)
t_moe <- t_moe_factor * sd(t_data$`sec to drop`) / sqrt(nrow(t_data))
```

```
null_model <- mean(t_data$`sec to drop`)
lower_ci <- null_model - t_moe
upper_ci <- null_model + t_moe

gp1 <- ggplot(data = t_data, aes(x= `sec to drop`))+geom_histogram(binwidth = 0.25,color="light blue", f
    geom_vline(aes(xintercept = null_model), color = "dark blue")+
    geom_vline(aes(xintercept = lower_ci), color = "dark blue", linetype = "dashed")+
    geom_vline(aes(xintercept = upper_ci), color = "dark blue", linetype = "dashed")+
    labs(title = "Distribution of Elapsed Seconds",
         subtitle = paste("after 30sec board goes sideways",
                          "\nnull model estimate:", round(mean(t_data$`sec to drop`), 2),
                          "\nMOE (margin of error) = +-", round(t_moe, 2))) + scale_x_continuous()
gp1
```


Distribution of Elapsed Seconds
after 30sec board goes sideways
null model estimate: 5.44
MOE (margin of error) = +− 0.1
count
sec to drop
0
5
10
15
20
25
4
5
6
7

3 MLR Model

Let's build a linear model to predict the gate drop times.

```
t_data_src_cols <- colnames(t_data)
t_data_src_cols <- t_data_src_cols[!t_data_src_cols == 'sec to drop'] #stage this for later when we mak

t_model <- lm(`sec to drop` ~ . , data = t_data)
print(summary(t_model))
```

```
##
## Call:
## lm(formula = `sec to drop` ~ ., data = t_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5160 -0.2893  0.0229  0.4297  1.3200
##
## Coefficients: (2 not defined because of singularities)
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    6.025472   0.644596   9.348  2.9e-15 ***
## `SX Futures`   0.120000   0.420658   0.285    0.776
## `250 SX East`  0.001335   0.345284   0.004    0.997
## `250 SX West`  0.114100   0.340349   0.335    0.738
## `450 SX`      -0.015472   0.334897  -0.046    0.963
## `450 MX`             NA         NA      NA       NA
## `250 MX`             NA         NA      NA       NA
## Heat          -0.629703   0.560379  -1.124    0.264
## LCQ           -0.653994   0.569210  -1.149    0.253
## Final         -0.585472   0.560703  -1.044    0.299
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5508 on 99 degrees of freedom
## Multiple R-squared:  0.02537,    Adjusted R-squared:  -0.04355
## F-statistic: 0.3681 on 7 and 99 DF,  p-value: 0.9188
```

```
t_beta <- round(coefficients(t_model),2)
print(t_beta)
```

```
##   (Intercept)  `SX Futures` `250 SX East` `250 SX West`      `450 SX`
##          6.03          0.12          0.00          0.11         -0.02
##      `450 MX`      `250 MX`          Heat           LCQ         Final
##            NA            NA         -0.63         -0.65         -0.59
```

Here's our fitted model:

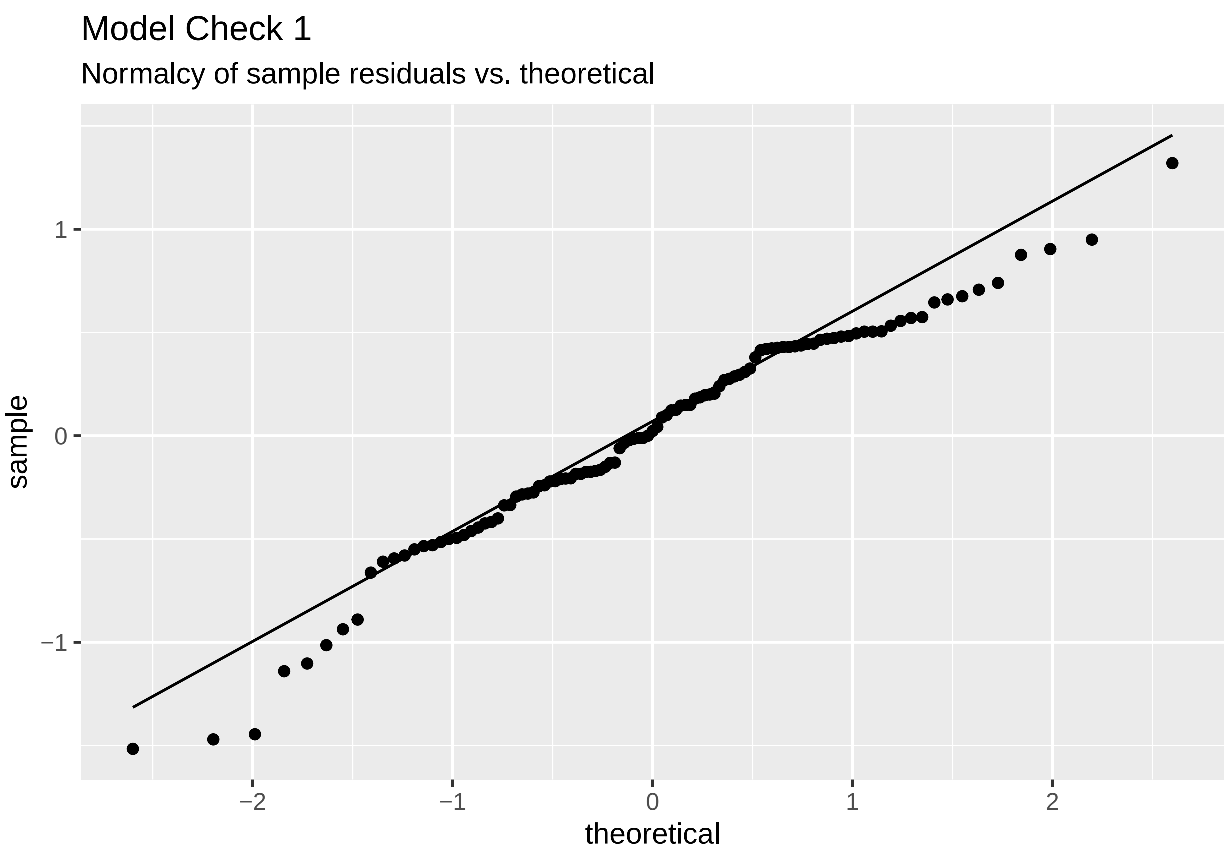
$\hat{y}$ = 6.03 + 0.12x_1 + 0x_2 + 0.11x_3 + -0.02x_4 + NAx_5 + NAx_6 + -0.63x_7 + -0.65x_8 + -0.59 x_9

3.1 Model Checking

3.1.1 Normalcy

Let's do some basic model checking. First, let's check to see if the data follows the normal distribution. We can do this by evaluating the ranked residuals against the theoretical ideal corresponding point in the normal cumulative distribution function. If the ranked residuals generally follow the theoretical line, then we can declare the data follows the normal distribution.

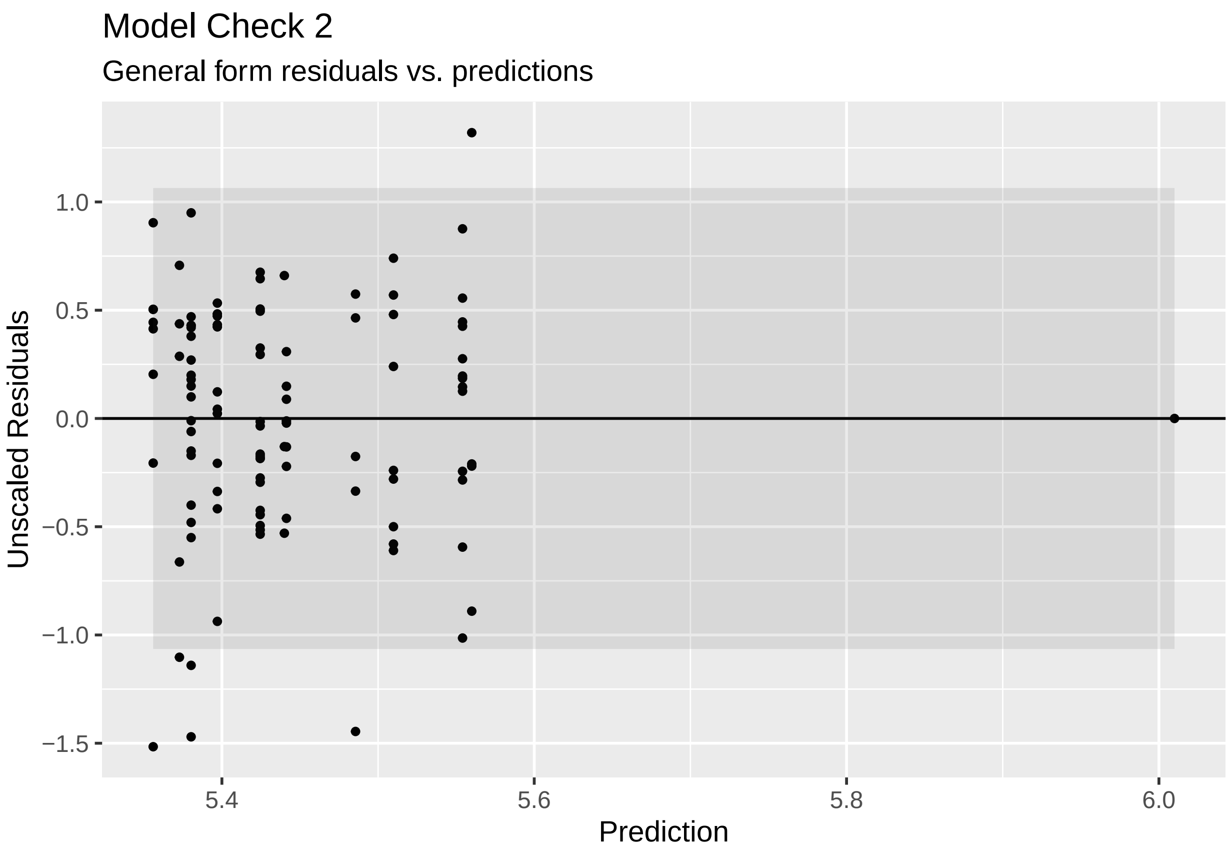
```
unscaled_residual <- tibble(t_model$residuals)

t_data <- t_data |>bind_cols(unscaled_residual)
t_length <- length(t_data)
colnames(t_data)[t_length] <- 'unscaled_residual'

t_qq_plot <- ggplot(data = t_data, aes(sample = unscaled_residual)) +stat_qq() + stat_qq_line() +
    labs(title =  "Model Check 1", subtitle = "Normalcy of sample residuals vs. theoretical", y = "sampl
t_qq_plot
```



These gate drop times are close to the theoretical normal distribution. However, it is worth noting the data is a little light-tailed on the lower-valued residuals and a little heavy-tailed on the higher-valued residuals. It's unlikely, though this will negatively impact our predictions.

3.1.2 Residuals

```
t_predicted_response <- tibble(t_model$fitted.values)
highbound <- 2 * sd(t_data$unscaled_residual)
lowbound <- -2 * sd(t_data$unscaled_residual)

t_data <- t_data |>bind_cols(t_predicted_response)
t_length <- length(t_data)
colnames(t_data)[t_length]  <- 't_prediction'
q <- ggplot(data = t_data, aes(y = `unscaled_residual`, x = `t_prediction` )) + geom_point(size=1) +
    labs(title =  "Model Check 2", subtitle = "General form residuals vs. predictions", y = "Unscaled Re
    geom_ribbon(aes(ymin = lowbound, ymax = highbound), alpha = 0.1)
q
```



Above, the shaded region indicates $\pm 2\sigma$ of the unscaled residuals against each of the predictions. Given that most of the predictions fit within the shaded area, we can say this model passes this check as well.

4 Predictions

Let's make some predictions for the 250 and 450 main events!

```
pred_250_main_SX_east <- c(0,1,0,0,0,0,0,0,1)
pred_250_main_SX_west <- c(0,0,1,0,0,0,0,0,1)
pred_450_main_SX <- c(0,0,0,1,0,0,0,0,1)
p_dat <- tibble()
p_dat <- rbind(p_dat, pred_250_main_SX_east, pred_250_main_SX_west, pred_450_main_SX)
names(p_dat) <- t_data_src_cols

a_prediction <- predict(t_model, p_dat, interval = 'confidence')
```

Table 1: Main Event Predictions

Class	Estimate	Lower Bound	Upper Bound
250 SX East	5.44	5.17	5.71
250 SX West	5.55	5.31	5.79
450 SX	5.42	5.22	5.63