R ET ATT 15

Jake PC Desktop

2024-05-07

#Set a working directory, here's mine:

```
# Set working directory and read in all data for participants 01-07
#Windows:
setwd("F:/R ET ATT 15")
#Mac:
#setwd("/Volumes/Intel SSD/R ET ATT 15")
```

```
require(dplyr)
```

```
## Loading required package: dplyr
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
##
## 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
```

```
# Assign each psychopy output to dataframes:
P01_MD <- read.csv("P01_MD.csv")
P02_MD <- read.csv("P02_MD.csv")
P03_MD <- read.csv("P03_MD.csv")
P04_MD <- read.csv("P04_MD.csv")
P05_MD <- read.csv("P05_MD.csv")
P06_MD <- read.csv("P06_MD.csv")
P07_MD <- read.csv("P07_MD.csv")

# Add RECORDING_SESSION_LABEL column to each participant dataset
P01_MD <- mutate(P01_MD, RECORDING_SESSION_LABEL = "REAIM01")
P02_MD <- mutate(P02_MD, RECORDING_SESSION_LABEL = "REAIM02")
P03_MD <- mutate(P03_MD, RECORDING_SESSION_LABEL = "REAIM03")
P04_MD <- mutate(P04_MD, RECORDING_SESSION_LABEL = "REAIM04")
```

```
P05_MD <- mutate(P05_MD, RECORDING_SESSION_LABEL = "REAIM05")
P06_MD <- mutate(P06_MD, RECORDING_SESSION_LABEL = "REAIM06")
P07_MD <- mutate(P07_MD, RECORDING_SESSION_LABEL = "REAIM07")

#Assign saccade and full_trial reports to dataframes:
P01_P07_SACCADE <- read.csv("P01_P07_CUSTOM_IP_SACCADE.csv")
P01_P07_FULL_TRIAL <- read.csv("P01_P07_FULL_TRIAL_TR.csv")

#Organise them for continuity:
P01_P07_SACCADE <- P01_P07_SACCADE %>%
  arrange(RECORDING_SESSION_LABEL, TRIAL_INDEX)

P01_P07_FULL_TRIAL <- P01_P07_FULL_TRIAL %>%
  arrange(RECORDING_SESSION_LABEL, INDEX)
```

#Sort and rename dataframes and objects:

```
require(dplyr)


# Combine participant datasets into one dataframe
P01_P07_MD <- bind_rows(P01_MD, P02_MD, P03_MD, P04_MD, P05_MD, P06_MD, P07_MD)


#Filtering purely by NA RT's means we only get recorded trials, 540 per participant.
P01_P07_MD_TRIALS <- P01_P07_MD %>% filter(RT != "NA")


#Assign "true" trial index:
P01_P07_MD_TRIALS <- P01_P07_MD_TRIALS %>%
  group_by(RECORDING_SESSION_LABEL) %>%
  mutate(Trial_Number = row_number())

P01_P07_MD_TRIALS <- P01_P07_MD_TRIALS %>%
  group_by(RECORDING_SESSION_LABEL) %>%
  mutate(Previous_Target = lag(tx))


#Rename variables for ease and continuity:
P01_P07_MD_TRIALS <- P01_P07_MD_TRIALS %>%
  rename(
    POV = BlockType,
    TARGET_TARG = tx,
    TRT = RT,
    TRIAL_INDEX = Trial_Number,
    PREVIOUS_TARG = Previous_Target
  )


#Only need select objects from .EDF report
P01_P07_MD_TRIALS <- P01_P07_MD_TRIALS %>%
  select(RECORDING_SESSION_LABEL, TRIAL_INDEX, POV, TARGET_TARG, PREVIOUS_TARG, TRT)
```

#Cleaning the *FULL* Trial Dataset: *##For later use with matching previous trial target location:*

```r
# Cleaning the full trial dataset to use TRIAL_INDEX and create TRIAL_TYPE for true trial pairs

library(dplyr)

# Select necessary variables
clean_trial_lot <- P01_P07_FULL_TRIAL %>%
  select(RECORDING_SESSION_LABEL, INDEX, POV, TARGET)

# Remove undefined target locations
clean_trial_lot <- clean_trial_lot %>% filter(TARGET != "UNDEFINED")

# Sort by RECORDING_SESSION_LABEL and INDEX
clean_trial_lot <- clean_trial_lot %>%
  arrange(RECORDING_SESSION_LABEL, INDEX)

# Create columns for previous trial's target and index
clean_trial_lot <- clean_trial_lot %>%
  mutate(Previous_Target = lag(TARGET),
         Previous_Index = lag(INDEX))

# Determine trial type (same or different)
clean_trial_lot <- clean_trial_lot %>%
  mutate(Trial_Type = ifelse(TARGET == Previous_Target & INDEX == Previous_Index + 1,
                             "same",
                             ifelse(INDEX == Previous_Index + 1,
                                    "different",
                                    NA)))

# Rename the "INDEX" column to "TRIAL_INDEX"
clean_trial_lot <- clean_trial_lot %>%
  rename(TRIAL_INDEX = INDEX)
```

#Tidying the objects in the dataframe for visual inspection:

```r
# Create a new dataset with relevant data for analysis

library(dplyr)
library(car)
```

```
## Warning: package 'car' was built under R version 4.3.3
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
CLEAN_P01_P07 <- P01_P07_SACCADE %>%
  select(RECORDING_SESSION_LABEL, TRIAL_INDEX, POV, TRIAL_START_TIME, IP_START_TIME, CURRENT_SAC_MSG_LIS
         CURRENT_SAC_MSG_LIST_TIME, CURRENT_SAC_START_TIME, CURRENT_SAC_DURATION, CURRENT_SAC_AMPLITUDE
         CURRENT_SAC_DIRECTION, CURRENT_SAC_ANGLE, CURRENT_SAC_AVG_VELOCITY, CURRENT_SAC_START_TIME,
         CURRENT_SAC_END_TIME, CURRENT_SAC_START_X, CURRENT_SAC_START_Y, CURRENT_SAC_END_X, CURRENT_SAC
         CURRENT_SAC_END_INTEREST_AREA_LABEL, CURRENT_SAC_END_INTEREST_AREA_X_OFFSET,
         CURRENT_SAC_END_INTEREST_AREA_Y_OFFSET, NEXT_FIX_DURATION, NEXT_FIX_X, NEXT_FIX_Y,
         PREVIOUS_SAC_DURATION, PREVIOUS_SAC_AMPLITUDE, PREVIOUS_SAC_DIRECTION, PREVIOUS_FIX_MSG_LIST_TI
         PREVIOUS_FIX_MSG_LIST_TIME, TARGET_TARG)
```

#Clean CLEAN_P01_P07 and change name to keep both dataframes for checks:

```
# Main data cleaning chunk

# Remove trials with undefined target from P01_P07_SACCADE dataframe
CLEAN_P01_P07 <- CLEAN_P01_P07 %>% filter(TARGET_TARG != "UNDEFINED")

#Remove all trials with centre targets because they do (or should) not contain saccades.
CLEAN_P01_P07 <- CLEAN_P01_P07 %>% filter(TARGET_TARG != "0")

# Filter trials not starting in the center of the screen
center_x <- 960
center_y <- 720
max_distance <- 144

FILTERED_P01_P07 <- CLEAN_P01_P07 %>%
  filter(abs(CURRENT_SAC_START_X - center_x) <= max_distance &
         abs(CURRENT_SAC_START_Y - center_y) <= max_distance &
         sqrt((CURRENT_SAC_START_X - center_x)^2 + (CURRENT_SAC_START_Y - center_y)^2) <= max_distance)
```

#Main data cleaning chunk:

```
# Calculate SRT and filter trials based on latency criteria

FILTERED_P01_P07 <- FILTERED_P01_P07 %>%
  mutate(SRT = CURRENT_SAC_START_TIME - (IP_START_TIME - TRIAL_START_TIME)) %>%
  filter(SRT >= 100 & SRT <= 500)

# Calculate SRT and filter trials based on latency criteria

FILTERED_P01_P07 <- FILTERED_P01_P07 %>%
  mutate(SRT = CURRENT_SAC_START_TIME - (IP_START_TIME - TRIAL_START_TIME)) %>%
  filter(SRT >= 100 & SRT <= 500)

# Count the number of remaining trials for each participant
remaining_trials_per_participant <- FILTERED_P01_P07 %>%
  group_by(RECORDING_SESSION_LABEL) %>%
  summarize(remaining_trials = n_distinct(TRIAL_INDEX))

# Get the total number of remaining trials
total_remaining_trials <- sum(remaining_trials_per_participant$remaining_trials)
cat("Total number of trials remaining after filtering:", total_remaining_trials, "\n")
```

```
## Total number of trials remaining after filtering: 2366
```

#Create empty dataframe to populate with primary (first in this case) saccades:

```
# Create a new dataframe to store primary saccades
primary_saccades_df <- data.frame()

# Create vectors to store debugging information
direction_values <- vector()
target_values <- vector()
towards_target <- vector()

# Iterate through each participant
for (participant_label in unique(FILTERED_P01_P07$RECORDING_SESSION_LABEL)) {
  participant_data <- filter(FILTERED_P01_P07, RECORDING_SESSION_LABEL == participant_label)

  # Iterate through each trial index
  for (trial_index in unique(participant_data$TRIAL_INDEX)) {
    trial_data <- filter(participant_data, TRIAL_INDEX == trial_index)

    # Select the first row (saccade) in the trial
    first_saccade <- trial_data[1, ]

    # Store direction and target values for debugging
    direction_values <- c(direction_values, first_saccade$CURRENT_SAC_DIRECTION)
    target_values <- c(target_values, first_saccade$TARGET_TARG)

    # Check if saccade direction is towards the target
    if (first_saccade$CURRENT_SAC_DIRECTION == "LEFT" && first_saccade$TARGET_TARG == "60") {
      towards_target <- c(towards_target, TRUE)
    } else if (first_saccade$CURRENT_SAC_DIRECTION == "RIGHT" && first_saccade$TARGET_TARG == "-60") {
      towards_target <- c(towards_target, TRUE)
    } else {
      towards_target <- c(towards_target, FALSE)
    }

    # Add the first saccade to primary_saccades_df
    primary_saccades_df <- rbind(primary_saccades_df, first_saccade)
  }
}

# Calculate the proportion of trials where the primary saccade was in the direction of the target
prop_towards_target <- mean(towards_target) * 100  # Convert to percentage

# Print the debugging information
cat("Proportion of trials where the primary saccade was towards the target:", prop_towards_target, "%\n"
```

```
## Proportion of trials where the primary saccade was towards the target: 94.71682 %
```

```
# Sort the dataframe by participant and trial index for visual inspection
primary_saccades_df <- primary_saccades_df %>%
  arrange(RECORDING_SESSION_LABEL, TRIAL_INDEX)
```

#Clean the full trial dataframe and match the TRIAL_INDEX

```r
require(dplyr)
# Filter out rows with NA in Previous_Target
clean_trial_lot_filtered <- clean_trial_lot[!is.na(clean_trial_lot$Previous_Target), ]

matching_indices <- match(
  paste(primary_saccades_df$RECORDING_SESSION_LABEL, primary_saccades_df$TRIAL_INDEX),
  paste(clean_trial_lot_filtered$RECORDING_SESSION_LABEL, clean_trial_lot_filtered$TRIAL_INDEX)
)

previous_matching_indices <- match(
  paste(primary_saccades_df$RECORDING_SESSION_LABEL, primary_saccades_df$TRIAL_INDEX - 1),
  paste(clean_trial_lot_filtered$RECORDING_SESSION_LABEL, clean_trial_lot_filtered$TRIAL_INDEX)
)

primary_saccades_df$Trial_Type <- clean_trial_lot_filtered$Trial_Type[matching_indices]
primary_saccades_df$Previous_Target <- clean_trial_lot_filtered$TARGET[previous_matching_indices]

# Arrange the dataframe by participant and trial index
primary_saccades_df <- primary_saccades_df %>%
  arrange(RECORDING_SESSION_LABEL, TRIAL_INDEX)

# Filter out rows with NA in Previous_Target and Trial_Type from clean_trial_lot
clean_trial_lot_filtered <- clean_trial_lot[!is.na(clean_trial_lot$Previous_Target) & !is.na(clean_trial

# Filter out rows with NA in Previous_Target and Trial_Type from primary_saccades_df
primary_saccades_df_filtered <- primary_saccades_df[!is.na(primary_saccades_df$Previous_Target) & !is.na

# Merge the two dataframes based on RECORDING_SESSION_LABEL and TRIAL_INDEX
merged_df <- merge(primary_saccades_df_filtered, P01_P07_MD_TRIALS,
                   by = c("RECORDING_SESSION_LABEL", "TRIAL_INDEX"), all.x = TRUE)

# Replace NA values in TRT column with 0
merged_df$TRT[is.na(merged_df$TRT)] <- 0

# Select only the necessary columns from the merged dataframe
merged_df <- merged_df %>%
  select(RECORDING_SESSION_LABEL, TRIAL_INDEX, TRT)

# Merge the TRT column back into primary_saccades_df_filtered dataframe
primary_saccades_df_filtered <- merge(primary_saccades_df_filtered, merged_df,
                                      by = c("RECORDING_SESSION_LABEL", "TRIAL_INDEX"), all.x = TRUE)

# Reorder the dataframe by RECORDING_SESSION_LABEL, POV, and TRIAL_INDEX
primary_saccades_df_filtered <- primary_saccades_df_filtered %>%
  arrange(RECORDING_SESSION_LABEL, POV, TRIAL_INDEX)

tidy_PS_df <- primary_saccades_df_filtered %>%
  select(RECORDING_SESSION_LABEL, POV, TRIAL_INDEX, TARGET_TARG, Previous_Target, Trial_Type, SRT,
         TRT, CURRENT_SAC_DIRECTION, CURRENT_SAC_AMPLITUDE, CURRENT_SAC_DURATION, CURRENT_SAC_AVG_VELOCI
         CURRENT_SAC_START_TIME, CURRENT_SAC_END_TIME, CURRENT_SAC_START_X, CURRENT_SAC_END_X,
         CURRENT_SAC_START_Y, CURRENT_SAC_END_Y, CURRENT_SAC_END_INTEREST_AREA_LABEL, IP_START_TIME)

# Convert TRT from seconds to milliseconds
```

```r
tidy_PS_df$TRT <- tidy_PS_df$TRT * 1000

# Remove rows with TRT equal to 0
tidy_PS_df <- tidy_PS_df %>%
  filter(TRT >=200)

tidy_PS_df <- tidy_PS_df %>%
  filter(TRT <=3000)

# Check the head of the dataframe again to confirm the changes
#head(tidy_PS_df)
```

#Calculate Descriptive Statistics for Each Participant:

```r
# Calculate descriptive statistics for each participant
participant_stats <- tidy_PS_df %>%
  group_by(RECORDING_SESSION_LABEL) %>%
  summarize(
    Mean_SRT = mean(SRT),
    SD_SRT = sd(SRT),
    Mean_TRT = mean(TRT),
    SD_TRT = sd(TRT)
  )

# Print the calculated statistics for each participant
print("Descriptive Statistics for Each Participant:")
```

```
## [1] "Descriptive Statistics for Each Participant:"
```

```r
print(participant_stats)
```

```
## # A tibble: 7 x 5
##   RECORDING_SESSION_LABEL Mean_SRT SD_SRT Mean_TRT SD_TRT
##   <chr>                      <dbl>  <dbl>    <dbl>  <dbl>
## 1 REAIM01                     241.   68.3    1112.   337.
## 2 REAIM02                     264.   48.2    1098.   305.
## 3 REAIM03                     184.   35.1    1212.   348.
## 4 REAIM04                     234.   49.6    1169.   362.
## 5 REAIM05                     239.   70.1    1052.   316.
## 6 REAIM06                     185.   31.4    1057.   282.
## 7 REAIM07                     198.   41.2    1006.   320.
```

```r
# Split dataframe to calculate descriptive statistics for each condition on each level
st_same_df <- tidy_PS_df %>%
  filter(POV == "static", Trial_Type == "same")

st_diff_df <- tidy_PS_df %>%
  filter(POV == "static", Trial_Type == "different")

dy_same_df <- tidy_PS_df %>%
  filter(POV == "dynamic", Trial_Type == "same")
```

```r
dy_diff_df <- tidy_PS_df %>%
  filter(POV == "dynamic", Trial_Type == "different")

# Calculate combined descriptive statistics for all participants together for each condition
st_same_stats <- st_same_df %>%
  summarize(
    Mean_SRT = mean(SRT),
    SD_SRT = sd(SRT),
    Mean_TRT = mean(TRT),
    SD_TRT = sd(TRT)
  )

st_diff_stats <- st_diff_df %>%
  summarize(
    Mean_SRT = mean(SRT),
    SD_SRT = sd(SRT),
    Mean_TRT = mean(TRT),
    SD_TRT = sd(TRT)
  )

dy_same_stats <- dy_same_df %>%
  summarize(
    Mean_SRT = mean(SRT),
    SD_SRT = sd(SRT),
    Mean_TRT = mean(TRT),
    SD_TRT = sd(TRT)
  )

dy_diff_stats <- dy_diff_df %>%
  summarize(
    Mean_SRT = mean(SRT),
    SD_SRT = sd(SRT),
    Mean_TRT = mean(TRT),
    SD_TRT = sd(TRT)
  )

# Combine the statistics into a single data frame
combined_stats <- data.frame(
  Condition = c("Static - Same", "Static - Different", "Dynamic - Same", "Dynamic - Different"),
  Mean_SRT = c(st_same_stats$Mean_SRT, st_diff_stats$Mean_SRT, dy_same_stats$Mean_SRT, dy_diff_stats$Mea
  SD_SRT = c(st_same_stats$SD_SRT, st_diff_stats$SD_SRT, dy_same_stats$SD_SRT, dy_diff_stats$SD_SRT),
  Mean_TRT = c(st_same_stats$Mean_TRT, st_diff_stats$Mean_TRT, dy_same_stats$Mean_TRT, dy_diff_stats$Mea
  SD_TRT = c(st_same_stats$SD_TRT, st_diff_stats$SD_TRT, dy_same_stats$SD_TRT, dy_diff_stats$SD_TRT)
)

# Print the combined descriptive statistics
print("Combined Descriptive Statistics for All Participants for Each Condition:")
```

```
## [1] "Combined Descriptive Statistics for All Participants for Each Condition:"
```

```r
print(combined_stats)
```

```
##             Condition Mean_SRT   SD_SRT Mean_TRT   SD_TRT
```

```
## 1       Static - Same 208.4734 51.86112 1109.503 339.6067
## 2  Static - Different 206.3304 53.49897 1129.389 334.8941
## 3      Dynamic - Same 234.0988 62.10042 1095.994 312.6025
## 4 Dynamic - Different 233.9593 60.83833 1072.761 331.5108
```

```
as.data.frame(participant_stats)
```

```
##   RECORDING_SESSION_LABEL Mean_SRT   SD_SRT Mean_TRT   SD_TRT
## 1                 REAIM01 241.0156 68.26210 1112.149 336.7320
## 2                 REAIM02 264.3133 48.23889 1098.222 304.8153
## 3                 REAIM03 183.6974 35.07207 1211.894 347.8923
## 4                 REAIM04 233.8013 49.55921 1169.416 361.9129
## 5                 REAIM05 238.7287 70.07701 1052.308 316.4612
## 6                 REAIM06 185.3025 31.44478 1056.793 281.9196
## 7                 REAIM07 197.5000 41.21633 1006.477 320.0270
```

```
as.data.frame(combined_stats)
```

```
##             Condition Mean_SRT   SD_SRT Mean_TRT   SD_TRT
## 1       Static - Same 208.4734 51.86112 1109.503 339.6067
## 2  Static - Different 206.3304 53.49897 1129.389 334.8941
## 3      Dynamic - Same 234.0988 62.10042 1095.994 312.6025
## 4 Dynamic - Different 233.9593 60.83833 1072.761 331.5108
```

```
write.csv(participant_stats, "Participant Desc.csv", row.names = FALSE)
write.csv(combined_stats, "Combined Desc.csv", row.names = FALSE)
```

```
require(dplyr)
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

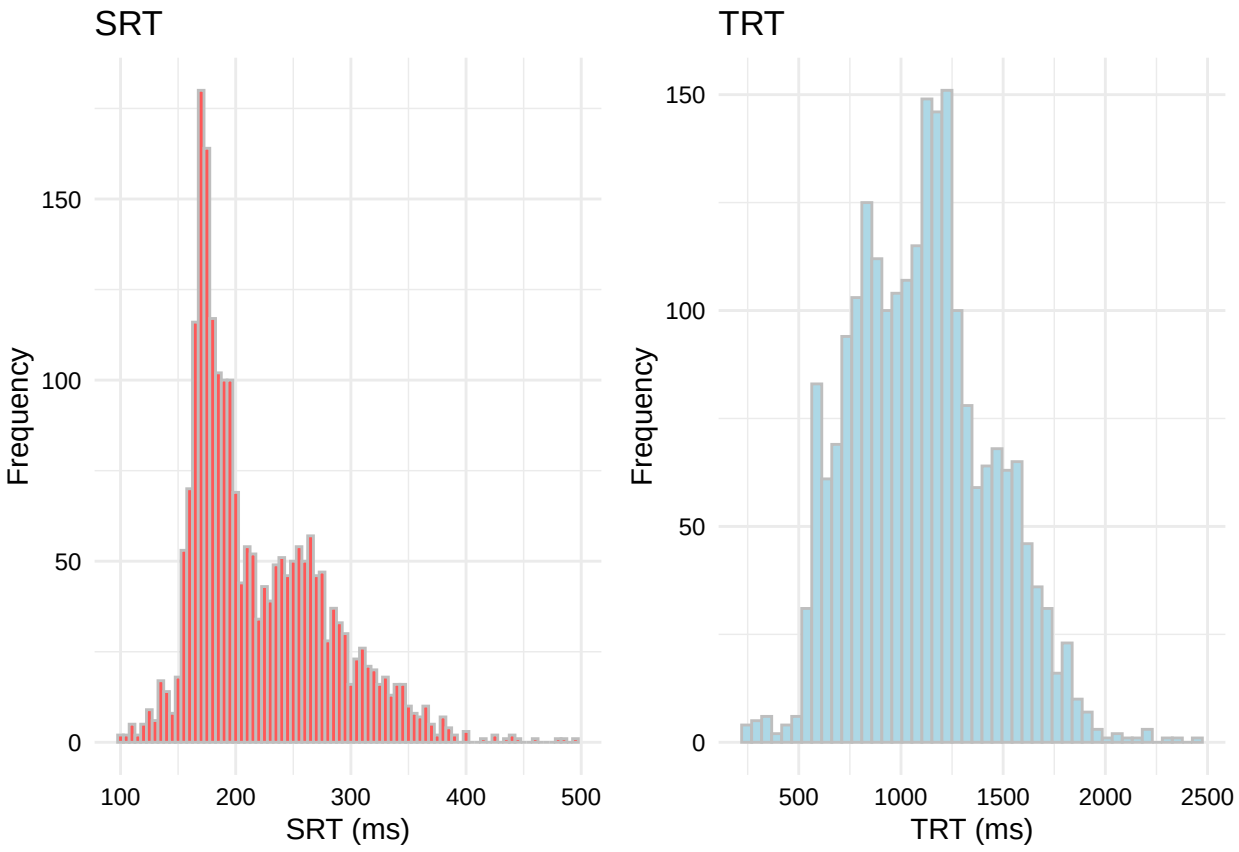
```
require(cowplot)
```

```
## Loading required package: cowplot
```

```
## Warning: package 'cowplot' was built under R version 4.3.3
```

```
# Define the hexadecimal color code for presentation yellow
"black" <- "#ffcc01"

# Histogram of SRT
ALL_SRT_PLOT <- ggplot(tidy_PS_df, aes(x = SRT)) +
                geom_histogram(binwidth = 5, fill = "#FF5757", color = "grey") +
                labs(title = "SRT", x = "SRT (ms)", y = "Frequency") +
                theme_minimal() +
                theme(axis.title.x = element_text(color = "black"),
                      axis.title.y = element_text(color = "black"),
```

```
                      axis.text = element_text(color = "black"),
                      plot.title = element_text(color = "black", size = 13))

# Scatterplot of SRT vs Frequency
ALL_SRT_SCATTER <- ggplot(tidy_PS_df, aes(x = SRT)) +
  geom_point(stat = "count", color = "#FF5757", position = position_jitter(width = 5, height = 0)) +
  labs(title = "SRT", x = "SRT (ms)", y = "Frequency") +
  theme_minimal() +
  theme(axis.title.x = element_text(color = "black"),
        axis.title.y = element_text(color = "black"),
        axis.text = element_text(color = "black"),
        plot.title = element_text(color = "black", size = 13))

# Histogram of TRT
ALL_TRT_PLOT <- ggplot(tidy_PS_df, aes(x = TRT)) +
                geom_histogram(binwidth = 49, fill = "lightblue", color = "grey") +
                labs(title = "TRT", x = "TRT (ms)", y = "Frequency") +
                theme_minimal() +
                theme(axis.title.x = element_text(color = "black"),
                      axis.title.y = element_text(color = "black"),
                      axis.text = element_text(color = "black"),
                      plot.title = element_text(color = "black", size = 13))

# Scatterplot of Rounded TRT vs Frequency
TRT_ROUND <- tidy_PS_df %>%
  mutate(Rounded_TRT = round(TRT)) %>%
  group_by(Rounded_TRT) %>%
  summarize(Frequency = n())

ALL_TRT_SCATTER <- ggplot(TRT_ROUND, aes(x = Rounded_TRT, y = Frequency)) +
  geom_point(color = "purple", position = position_jitter(width = 0.1, height = 0), alpha = 0.5) +
  geom_jitter(position = position_jitter(width = 0.1, height = 0), color = "purple", alpha = 0.5) +
  labs(title = "TRT", x = "Rounded TRT (ms)", y = "Frequency") +
  scale_y_continuous(breaks = seq(0, max(TRT_ROUND$Frequency), by = 1)) +
  theme_minimal() +
  theme(axis.title.x = element_text(color = "black"),
        axis.title.y = element_text(color = "black"),
        axis.text = element_text(color = "black"),
        plot.title = element_text(color = "black", size = 13))

# Arrange plots in a single panel
panel <- plot_grid(ALL_SRT_PLOT, ALL_TRT_PLOT, ncol = 2)

# Display the panel
panel
```



#Normality (shapiro wilks tests):

```
# Shapiro-Wilk tests
ALL_SRT_NORM <- shapiro.test(tidy_PS_df$SRT)
ALL_TRT_NORM <- shapiro.test(tidy_PS_df$TRT)

# Print Shapiro-Wilk test results
print("Shapiro Test for All SRT's:")
```

```
## [1] "Shapiro Test for All SRT's:"
```

```
print(ALL_SRT_NORM)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  tidy_PS_df$SRT
## W = 0.92494, p-value < 2.2e-16
```

```
print("Shapiro Test for All TRT's:")
```

```
## [1] "Shapiro Test for All TRT's:"
```

```r
print(ALL_TRT_NORM)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  tidy_PS_df$TRT
## W = 0.98992, p-value = 1.792e-11
```

```r
# Load the required libraries
library(car)

#C variance stands for condition variance. Measuring differences in variance across SRT
#between trial types in each of the POV conditions. Can't test interaction between condition
#in this function.
bartlett_c_variance_static <- bartlett.test(SRT ~ Trial_Type,
                                            data = tidy_PS_df,
                                            subset = (POV == "static"))
print(bartlett_c_variance_static)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  SRT by Trial_Type
## Bartlett's K-squared = 0.47273, df = 1, p-value = 0.4917
```

```r
bartlett_c_variance_dynamic <- bartlett.test(SRT ~ Trial_Type,
                                             data = tidy_PS_df,
                                             subset = (POV == "dynamic"))
print(bartlett_c_variance_dynamic)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  SRT by Trial_Type
## Bartlett's K-squared = 0.19605, df = 1, p-value = 0.6579
```

```r
bartlett_c_variance_static2 <- bartlett.test(TRT ~ Trial_Type,
                                            data = tidy_PS_df,
                                            subset = (POV == "static"))
print(bartlett_c_variance_static2)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  TRT by Trial_Type
## Bartlett's K-squared = 0.096601, df = 1, p-value = 0.7559
```

```r
bartlett_c_variance_dynamic2 <- bartlett.test(TRT ~ Trial_Type,
                                             data = tidy_PS_df,
                                             subset = (POV == "dynamic"))
print(bartlett_c_variance_dynamic2)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  TRT by Trial_Type
## Bartlett's K-squared = 1.5705, df = 1, p-value = 0.2101
```

#For estimated start values in glmer functions:

```
# Calculate mean and median SRT for each level of Trial_Type
mean_SRT <- aggregate(SRT ~ Trial_Type * POV, data = tidy_PS_df, FUN = mean)
#median_SRT <- aggregate(SRT ~ Trial_Type * POV, data = tidy_PS_df, FUN = median)

print(mean_SRT)
```

```
##   Trial_Type     POV      SRT
## 1  different dynamic 233.9593
## 2       same dynamic 234.0988
## 3  different  static 206.3304
## 4       same  static 208.4734
```

```
#print(median_SRT)

sd(tidy_PS_df$SRT)
```

```
## [1] 58.68065
```

#Model of SRT with density plot

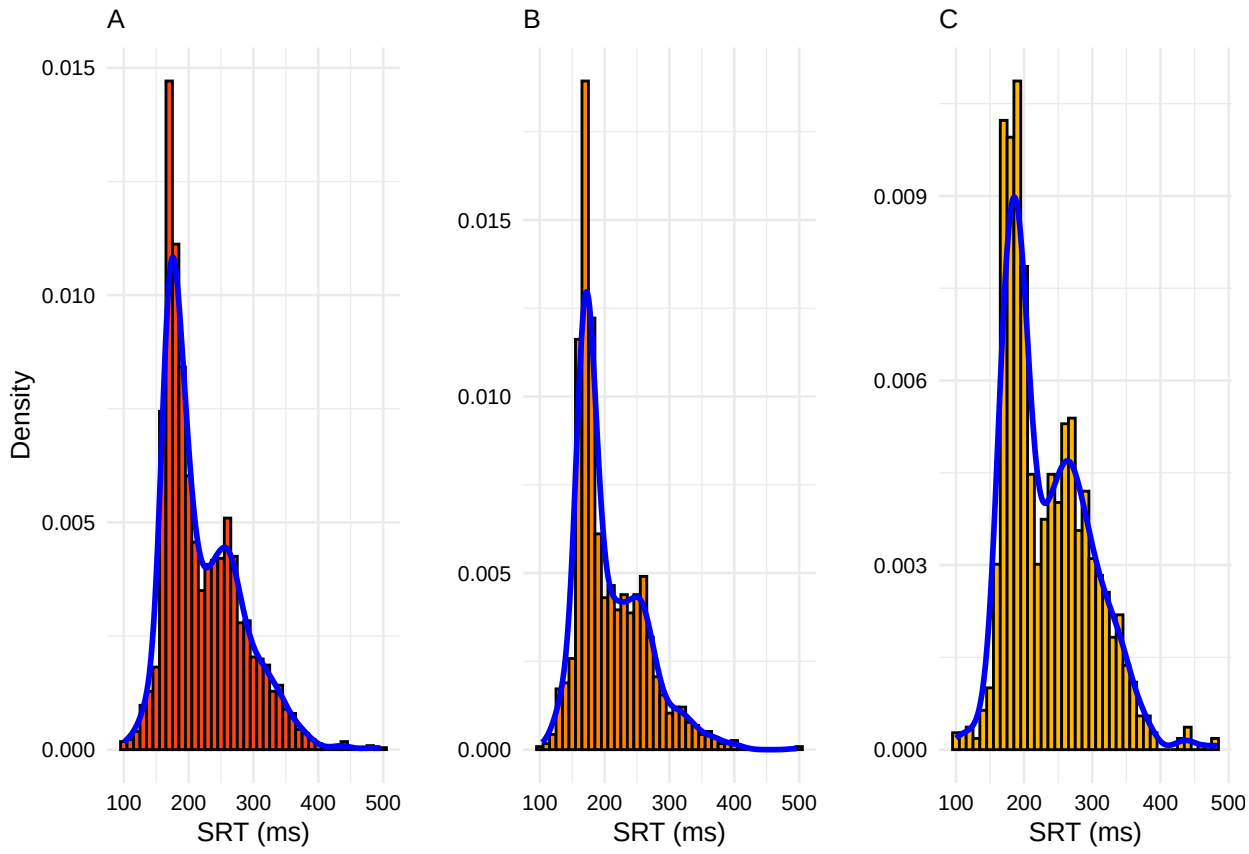
```
require(ggplot2)
require(dplyr)
require(cowplot)

STAT <- tidy_PS_df %>% filter(POV != "dynamic")
DYN <- tidy_PS_df %>% filter(POV != "static")

# Histogram and Density Plot of SRT
SRT_histogram <- ggplot(tidy_PS_df, aes(x = SRT)) +
  geom_histogram(binwidth = 10, fill = "#ff4517", color = "black", aes(y = ..density..)) +
  geom_density(color = "blue", size = 1, alpha = 0.5) +
  labs(title = "A",
       x = "SRT (ms)", y = "Density") +
  theme_minimal(base_size = 10) +
  theme(axis.title.x = element_text(color = "black"),
        axis.title.y = element_text(color = "black"),
        axis.text = element_text(color = "black"),
        plot.title = element_text(color = "black", size = 10))
```

```
## 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.
```

```r
#Histogram of only static data with dist. curve
SRT_STAT_histogram <- ggplot(STAT, aes(x = SRT)) +
  geom_histogram(binwidth = 10, fill = "#ff8400", color = "black", aes(y = ..density..)) +
  geom_density(color = "blue", size = 1, alpha = 0.5) +
  labs(title = "B",
       x = "SRT (ms)", y = "") +
  theme_minimal(base_size = 10) +
  theme(axis.title.x = element_text(color = "black"),
        axis.title.y = element_text(color = "black"),
        axis.text = element_text(color = "black"),
        plot.title = element_text(color = "black", size = 10))

#Histogram of only dynamic data with dist. curve
SRT_DYN_histogram <- ggplot(DYN, aes(x = SRT)) +
  geom_histogram(binwidth = 10, fill = "#ffb700", color = "black", aes(y = ..density..)) +
  geom_density(color = "blue", size = 1, alpha = 0.5) +
  labs(title = "C",
       x = "SRT (ms)", y = "") +
  theme_minimal(base_size = 10) +
  theme(axis.title.x = element_text(color = "black"),
        axis.title.y = element_text(color = "black"),
        axis.text = element_text(color = "black"),
        plot.title = element_text(color = "black", size = 10))

SRT_PANEL<-plot_grid(SRT_histogram, SRT_STAT_histogram, SRT_DYN_histogram, ncol = 3, align = "w", rel_w
```

```
## Warning: The dot-dot notation (‘..density..‘) was deprecated in ggplot2 3.4.0.
## i Please use ‘after_stat(density)‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```r
SRT_PANEL
```

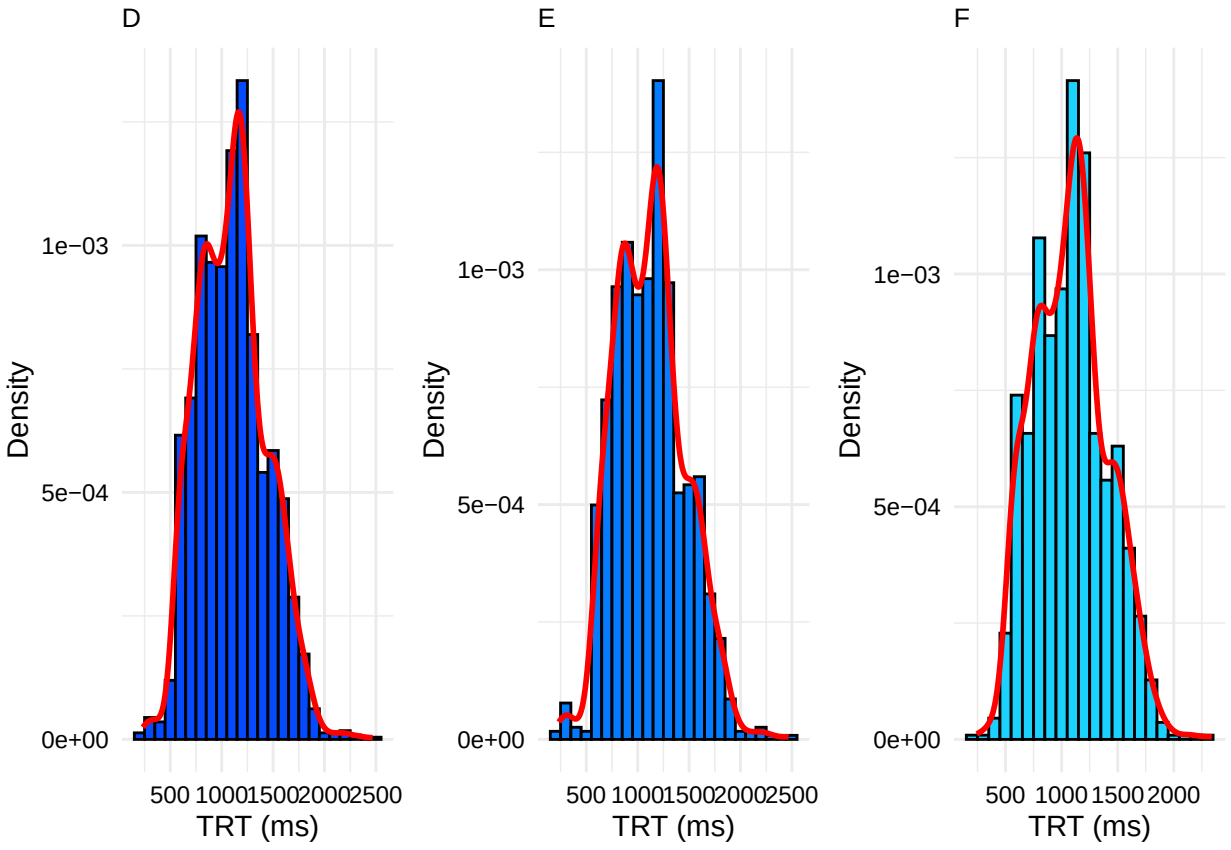



```
ggsave("SRT_PANEL.png", plot = SRT_PANEL, width = 9, height = 3)
```

#Model of TRT with density plot

```
require(ggplot2)
require(dplyr)
require(cowplot)

STAT <- tidy_PS_df %>% filter(POV != "dynamic")
DYN <- tidy_PS_df %>% filter(POV != "static")

# Histogram and Density Plot of TRT
TRT_histogram <- ggplot(tidy_PS_df, aes(x = TRT)) +
  geom_histogram(binwidth = 100, fill = "#054bfd", color = "black", aes(y = ..density..)) +
  geom_density(color = "red", size = 1) +
  labs(title = "D",
       x = "TRT (ms)", y = "Density") +
  theme_minimal() +
  theme(axis.title.x = element_text(color = "black"),
        axis.title.y = element_text(color = "black"),
        axis.text = element_text(color = "black"),
        plot.title = element_text(color = "black", size = 10))

#Histogram of only static data with dist. curve
TRT_STAT_histogram <- ggplot(STAT, aes(x = TRT)) +
```

```
  geom_histogram(binwidth = 100, fill = "#057cfd", color = "black", aes(y = ..density..)) +
  geom_density(color = "red", size = 1) +
  labs(title = "E",
       x = "TRT (ms)", y = "Density") +
  theme_minimal() +
  theme(axis.title.x = element_text(color = "black"),
        axis.title.y = element_text(color = "black"),
        axis.text = element_text(color = "black"),
        plot.title = element_text(color = "black", size = 10))

#Histogram of only dynamic data with dist. curve
TRT_DYN_histogram <- ggplot(DYN, aes(x = TRT)) +
  geom_histogram(binwidth = 100, fill = "#1bd1fd", color = "black", aes(y = ..density..)) +
  geom_density(color = "red", size = 1) +
  labs(title = "F",
       x = "TRT (ms)", y = "Density") +
  theme_minimal() +
  theme(axis.title.x = element_text(color = "black"),
        axis.title.y = element_text(color = "black"),
        axis.text = element_text(color = "black"),
        plot.title = element_text(color = "black", size = 10))

TRT_PANEL<-plot_grid(TRT_histogram, TRT_STAT_histogram, TRT_DYN_histogram, ncol = 3, align = "w", rel_w

TRT_PANEL
```



```
ggsave("TRT_PANEL.png", plot = TRT_PANEL, width = 9, height = 3)
```

#Taking the log of SRT was considered, and log start values are still used in functional glmer's:

```
library(dplyr)

# Mutate to create a new variable with the logarithm of SRT
tidy_PS_df <- tidy_PS_df %>%
  mutate(log_SRT = log(SRT))

STAT <- STAT %>%
  mutate(log_SRT = log(SRT))

DYN <- DYN %>%
  mutate(log_SRT = log(SRT))
# Calculate mean and median SRT for each level of Trial_Type
mean_log_SRT <- aggregate(log_SRT ~ Trial_Type * POV, data = tidy_PS_df, FUN = mean)

print(mean_log_SRT)
```

```
##   Trial_Type     POV  log_SRT
## 1  different dynamic 5.422760
## 2       same dynamic 5.422335
## 3  different  static 5.299361
## 4       same  static 5.312180
```

```
sd(tidy_PS_df$log_SRT)
```

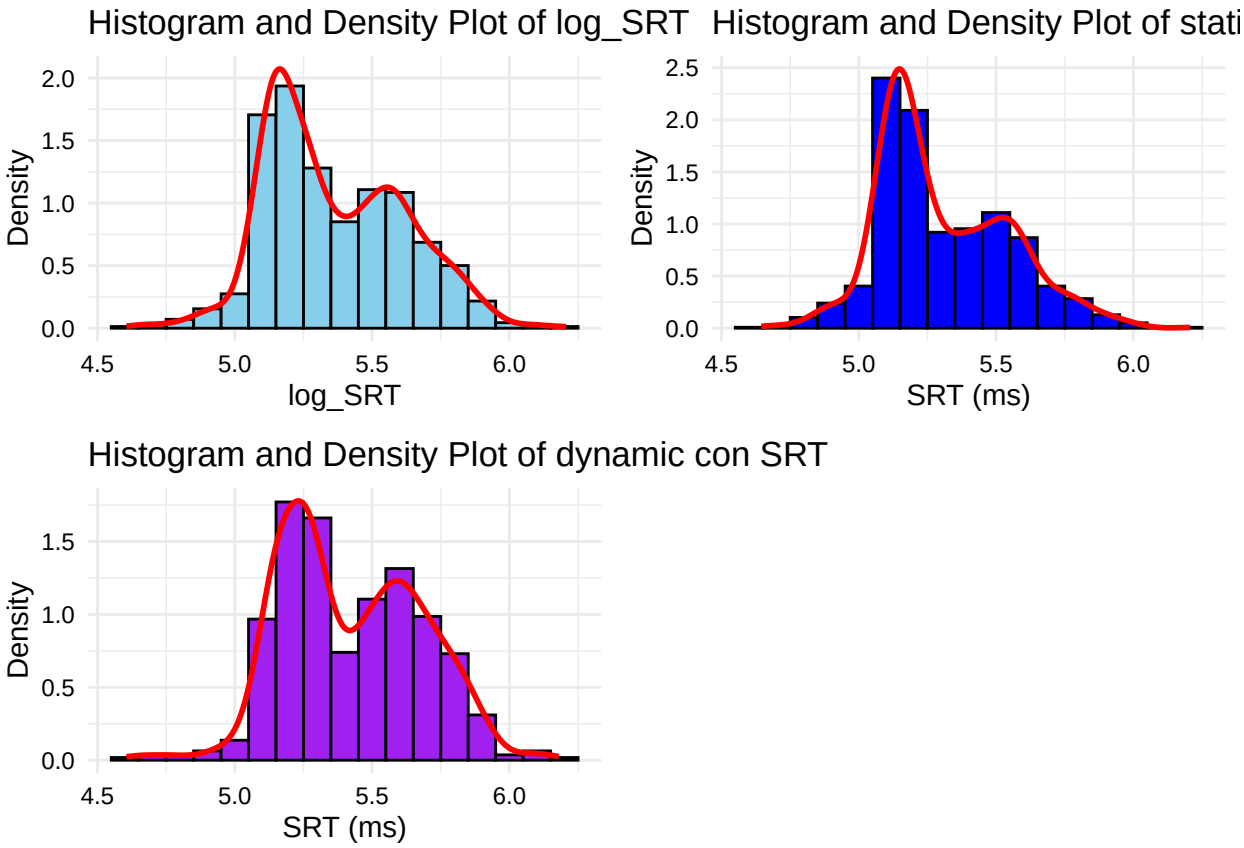
```
## [1] 0.2527194
```

```
#Again, we need estimated start values, but for the log_SRT now.
```

#Here are plots of the log_SRT for context:

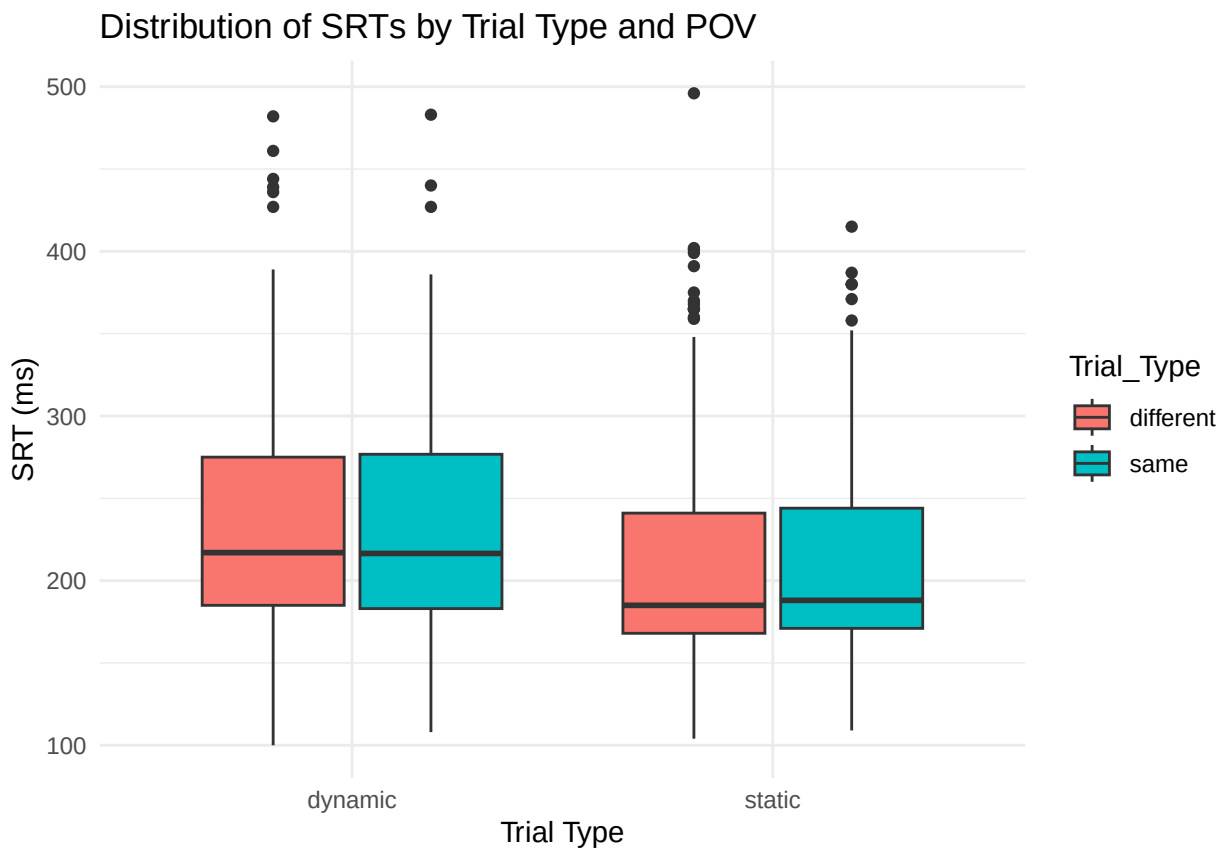
```
require(ggplot2)
require(dplyr)


# Histogram and Density Plot of SRT
log_SRT_histogram <- ggplot(tidy_PS_df, aes(x = log_SRT)) +
  geom_histogram(binwidth = 0.1, fill = "skyblue", color = "black", aes(y = ..density..)) +
  geom_density(color = "red", size = 1) +
  labs(title = "Histogram and Density Plot of log_SRT",
       x = "log_SRT", y = "Density") +
  theme_minimal() +
  theme(axis.title.x = element_text(color = "black"),
        axis.title.y = element_text(color = "black"),
        axis.text = element_text(color = "black"),
        plot.title = element_text(color = "black", size = 13))

log_SRT_STAT_histogram <- ggplot(STAT, aes(x = log_SRT)) +
```

```
  geom_histogram(binwidth = 0.1, fill = "blue", color = "black", aes(y = ..density..)) +
  geom_density(color = "red", size = 1) +
  labs(title = "Histogram and Density Plot of static con SRT",
       x = "SRT (ms)", y = "Density") +
  theme_minimal() +
  theme(axis.title.x = element_text(color = "black"),
        axis.title.y = element_text(color = "black"),
        axis.text = element_text(color = "black"),
        plot.title = element_text(color = "black", size = 13))

log_SRT_DYN_histogram <- ggplot(DYN, aes(x = log_SRT)) +
  geom_histogram(binwidth = 0.1, fill = "purple", color = "black", aes(y = ..density..)) +
  geom_density(color = "red", size = 1) +
  labs(title = "Histogram and Density Plot of dynamic con SRT",
       x = "SRT (ms)", y = "Density") +
  theme_minimal() +
  theme(axis.title.x = element_text(color = "black"),
        axis.title.y = element_text(color = "black"),
        axis.text = element_text(color = "black"),
        plot.title = element_text(color = "black", size = 13))

plot_grid(log_SRT_histogram, log_SRT_STAT_histogram, log_SRT_DYN_histogram)
```

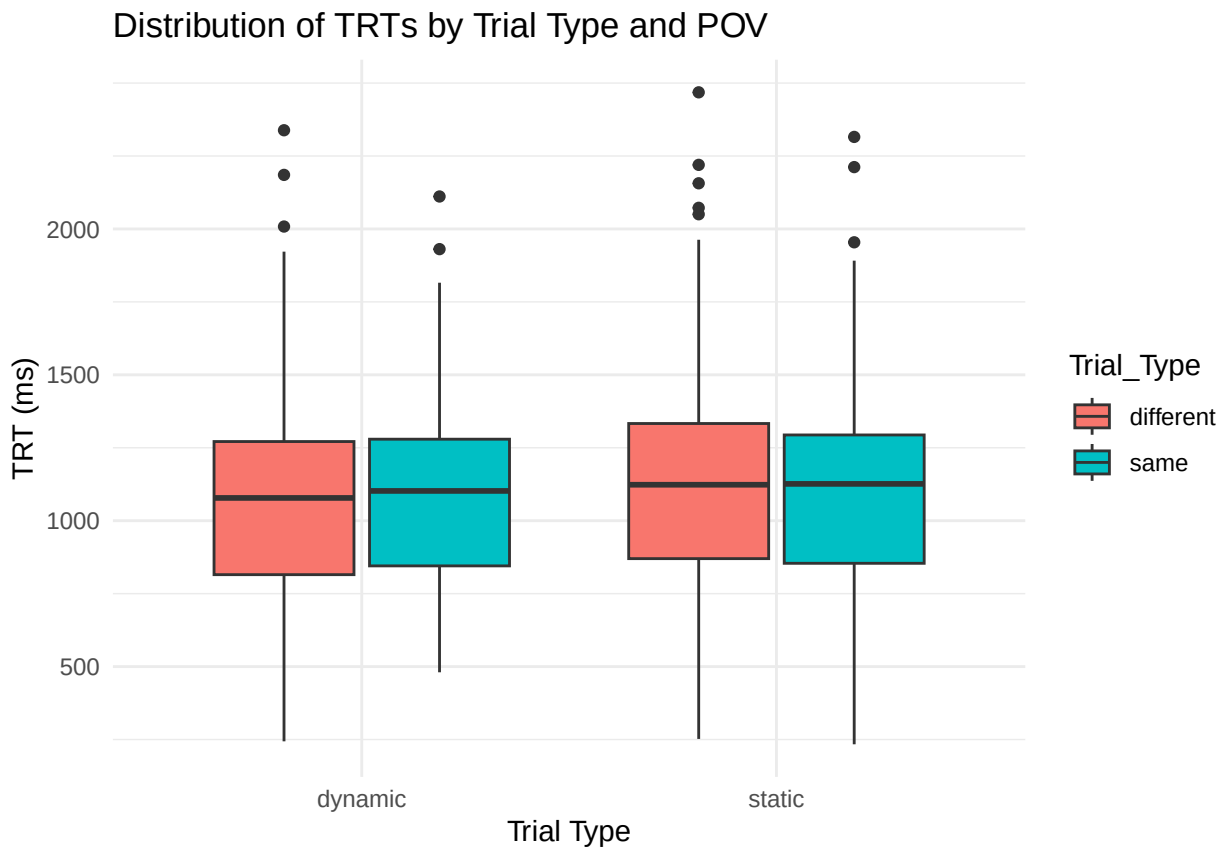



```
library(ggplot2)
```

```r
# Create box plot for SRTs grouped by the four conditions
ggplot(tidy_PS_df, aes(x = POV, y = SRT, fill = Trial_Type)) +
  geom_boxplot() +
  labs(x = "Trial Type", y = "SRT (ms)", fill = "Trial_Type") +
  ggtitle("Distribution of SRTs by Trial Type and POV") +
  theme_minimal()
```

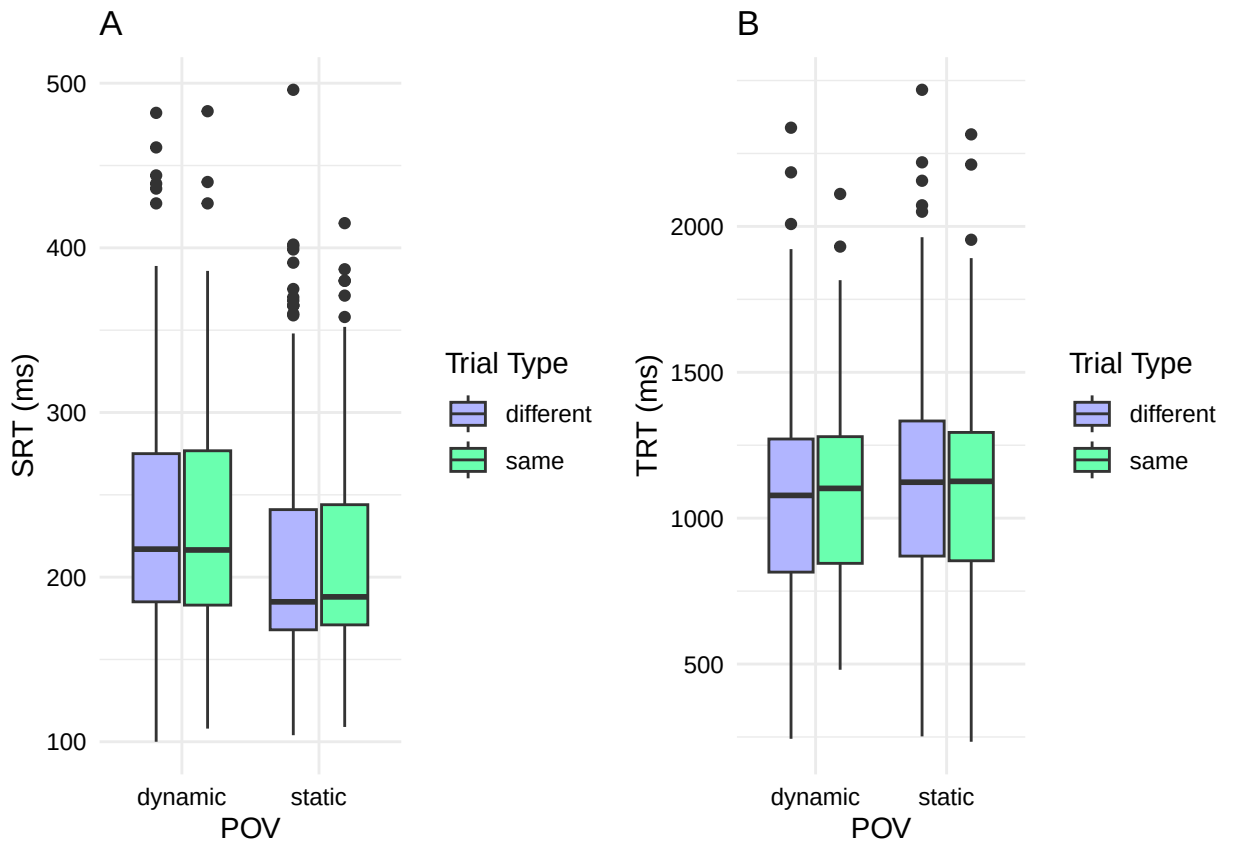



```r
# Create box plot for TRTs grouped by the four conditions
ggplot(tidy_PS_df, aes(x = POV, y = TRT, fill = Trial_Type)) +
  geom_boxplot() +
  labs(x = "Trial Type", y = "TRT (ms)", fill = "Trial_Type") +
  ggtitle("Distribution of TRTs by Trial Type and POV") +
  theme_minimal() +
  scale_y_continuous(breaks = seq(0, max(tidy_PS_df$TRT), by = 500))
```




```
library(ggplot2)

# Create box plot for SRTs grouped by the four conditions
SRT_BOX <- ggplot(tidy_PS_df, aes(x = POV, y = SRT, fill = Trial_Type)) +
            geom_boxplot() +
            scale_fill_manual(values = c("same" = "#6affaf", "different" = "#b1b5ff")) +
            labs(x = "POV", y = "SRT (ms)", fill = "Trial Type") +
            ggtitle("A") +
            theme_minimal() +
            theme(axis.title.x = element_text(color = "black"),
                  axis.title.y = element_text(color = "black"),
                  axis.text = element_text(color = "black"),
                  plot.title = element_text(color = "black", size = 13))

# Create box plot for TRTs grouped by the four conditions
TRT_BOX <- ggplot(tidy_PS_df, aes(x = POV, y = TRT, fill = Trial_Type)) +
            geom_boxplot() +
            scale_fill_manual(values = c("same" = "#6affaf", "different" = "#b1b5ff")) +
            labs(x = "POV", y = "TRT (ms)", fill = "Trial Type") +
            ggtitle("B") +
            theme_minimal() +
            scale_y_continuous(breaks = seq(0, max(tidy_PS_df$TRT), by = 500)) +
            theme(axis.title.x = element_text(color = "black"),
                  axis.title.y = element_text(color = "black"),
                  axis.text = element_text(color = "black"),
                  plot.title = element_text(color = "black", size = 13))
```

```
BOX_PANEL<-plot_grid(SRT_BOX, TRT_BOX, ncol = 2, align = "w", rel_widths = c(1, 1, 1))

BOX_PANEL
```




```
ggsave("BOX_PANEL.png", plot = BOX_PANEL, width = 9, height = 3)
```

#Echo of all considered exploratory models (now *DEFUNCT*):

```
#(Model1: Raw SRT, raw start values and IG dist. with canonical link):
#mean_ST_SAME <- mean(tidy_PS_df$SRT[tidy_PS_df$Trial_Type == "same" & tidy_PS_df$POV == "static"])
#mean_ST_DIFF <- mean(tidy_PS_df$SRT[tidy_PS_df$Trial_Type == "different" & tidy_PS_df$POV == "static"]
#mean_DY_SAME <- mean(tidy_PS_df$SRT[tidy_PS_df$Trial_Type == "same" & tidy_PS_df$POV == "dynamic"])
#mean_DY_DIFF <- mean(tidy_PS_df$SRT[tidy_PS_df$Trial_Type == "different" & tidy_PS_df$POV == "dynamic"

# Fit GLMM with specified start values
#model1 <- glmer(SRT ~ Trial_Type * POV + (1 | RECORDING_SESSION_LABEL),
#                data = tidy_PS_df,
#                family = inverse.gaussian(link = "1/mu^2"),
#                start = list(theta = 58.9,
#                             fixef = c(ST_SAME = mean_ST_SAME,
#                                       ST_DIFF = mean_ST_DIFF,
#                                       DY_SAME = mean_DY_SAME,
#                                       DY_DIFF = mean_DY_DIFF)),
```

```
#               control = glmerControl(optimizer = "bobyqa"))

#summary(model1)

#(Model 1.5: Model1 but with estimated start values):
#model1.5 <- glmer(SRT ~ Trial_Type * POV + (1 | RECORDING_SESSION_LABEL),
#               data = tidy_PS_df,
#               family = inverse.gaussian(link = "1/mu^2"),
#               start = list(theta = 58.9,
#                            fixef = c(ST_SAME = 220,
#                                      ST_DIFF = 220,
#                                      DY_SAME = 220,
#                                      DY_DIFF = 220)),
#               control = glmerControl(optimizer = "bobyqa"))

#summary(model1.5)

#(Model1.75: Transformation attempt to centralise.
#log_SRT, log start values, IG dist. with canonical link):
#model1.75 <- glmer(log_SRT ~ Trial_Type * POV + (1 | RECORDING_SESSION_LABEL),
#               data = tidy_PS_df,
#               family = inverse.gaussian(link = "1/mu^2"),
#               #More research required on this.s
#               start = list(theta = 0.25, fixef = c(ST_SAME = 5.36, ST_DIFF = 5.36, DY_SAME = 5.36, DY
#               control = glmerControl(optimizer = "bobyqa"))

#summary(model1.75)

#(Model2: raw SRT, log start values, IG dist. with log link)
#(functional, but defunct):
#model2 <- glmer(SRT ~ Trial_Type * POV + (1 | RECORDING_SESSION_LABEL),
#               data = tidy_PS_df,
#               family = inverse.gaussian(link = "log"),
#               #More research required on this.s
#               start = list(theta = 0.25, fixef = c(ST_SAME = 5.31, ST_DIFF = 5.29, DY_SAME = 5.42, DY
#               control = glmerControl(optimizer = "bobyqa"))

#summary(model2)

#(Model3: Model 2 but with unbiased start values)
#(Again, functional but defunct):
#model3 <- glmer(SRT ~ Trial_Type * POV + (1 | RECORDING_SESSION_LABEL),
#               data = tidy_PS_df,
#               family = inverse.gaussian(link = "log"),
#               #More research required on this.s
#               start = list(theta = 0.25, fixef = c(ST_SAME = 5.36, ST_DIFF = 5.36, DY_SAME = 5.36, DY
#               control = glmerControl(optimizer = "bobyqa"))

#summary(model3)

#(Model4: Model 3 but with log_SRT rather than raw SRT)
#(Functional with warning of bad fit):
#model4 <- glmer(log_SRT ~ Trial_Type * POV + (1 | RECORDING_SESSION_LABEL),
```

```
#                data = tidy_PS_df,
#                family = inverse.gaussian(link = "log"), #Changing link from individual to log still wo
#                #More research required on this.s
#                start = list(theta = 0.25, fixef = c(ST_SAME = 5.36, ST_DIFF = 5.36, DY_SAME = 5.36, DY
#                control = glmerControl(optimizer = "bobyqa"))

#summary(model4)

#(Model 5: raw SRT, log start values, identity link {virtually null link,
#no manipulation of curve fit} functional.
#Best contender thus far.
#Outputs in ms rather than log_ms)
#model5 <- glmer(SRT ~ Trial_Type * POV + (1 | RECORDING_SESSION_LABEL),
#                data = tidy_PS_df,
#                family = inverse.gaussian(link = "identity"), #Changing link from individual to log sti
#                #More research required on this.s
#                start = list(theta = 0.25, fixef = c(ST_SAME = 5.36, ST_DIFF = 5.36, DY_SAME = 5.36, DY
#                control = glmerControl(optimizer = "bobyqa"))

#summary(model5)
```

#Factorise main effects prior to relevelling for more logical structure of models pre-comparison:

```
# Convert Trial_Type and POV variables to factors with appropriate levels
tidy_PS_df$Trial_Type <- factor(tidy_PS_df$Trial_Type, levels = c("same", "different"))
tidy_PS_df$POV <- factor(tidy_PS_df$POV, levels = c("static", "dynamic"))

# Check the levels of both Trial_Type and POV variables
levels(tidy_PS_df$Trial_Type)
```

```
## [1] "same"      "different"
```

```
levels(tidy_PS_df$POV)
```

```
## [1] "static"  "dynamic"
```

```
# Remove unused levels from Trial_Type and POV
tidy_PS_df$Trial_Type <- droplevels(tidy_PS_df$Trial_Type)
tidy_PS_df$POV <- droplevels(tidy_PS_df$POV)

levels(tidy_PS_df$Trial_Type)
```

```
## [1] "same"      "different"
```

```
levels(tidy_PS_df$POV)
```

```
## [1] "static"  "dynamic"
```

#Setting up models for comparison based on model5 being best fit thus far:

```r
#AIC weighting/model comparisons:

################################################################################
#Relevelling so that the model summary shows "static" as POV and "same" as reference category:
tidy_PS_df$POV_LVL <- relevel(tidy_PS_df$POV, "static")
tidy_PS_df$Trial_Type_LVL <- relevel(tidy_PS_df$Trial_Type, "same")
################################################################################
levels(tidy_PS_df$Trial_Type)
```

```
## [1] "same"      "different"
```

```r
levels(tidy_PS_df$POV)
```

```
## [1] "static"  "dynamic"
```

```r
require(lme4)
```

```
## Loading required package: lme4
```

```
## Warning: package 'lme4' was built under R version 4.3.3
```

```
## Loading required package: Matrix
```

```
## Warning: package 'Matrix' was built under R version 4.3.3
```

```r
#Recode glmer so that leveLled objects are called:
modelA <- glmer(SRT ~ Trial_Type_LVL * POV_LVL + (1 | RECORDING_SESSION_LABEL),
                data = tidy_PS_df,
                family = inverse.gaussian(link = "identity"),
                #More research required on this.s
                start = list(theta = 0.25, fixef = c(ST_SAME = 5.36, ST_DIFF = 5.36, DY_SAME = 5.36, DY_
                control = glmerControl(optimizer = "bobyqa"))

summary(modelA)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: inverse.gaussian  ( identity )
## Formula: SRT ~ Trial_Type_LVL * POV_LVL + (1 | RECORDING_SESSION_LABEL)
##    Data: tidy_PS_df
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##  23493.1  23527.4 -11740.5  23481.1     2251
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.6123 -0.6713 -0.1928  0.5370  6.5887
##
## Random effects:
```

```
##  Groups                   Name        Variance  Std.Dev.
##  RECORDING_SESSION_LABEL (Intercept) 1.007e+02 10.03584
##  Residual                             2.106e-04  0.01451
## Number of obs: 2257, groups:  RECORDING_SESSION_LABEL, 7
##
## Fixed effects:
##                                      Estimate Std. Error t value Pr(>|z|)
## (Intercept)                           218.886     13.803  15.858  < 2e-16 ***
## Trial_Type_LVLdifferent                -2.433      2.534  -0.960    0.337
## POV_LVLdynamic                         24.681      3.363   7.340 2.14e-13 ***
## Trial_Type_LVLdifferent:POV_LVLdynamic  3.116      4.025   0.774    0.439
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) Tr_T_LVL POV_LV
## Trl_Typ_LVL -0.110
## POV_LVLdynm -0.080  0.524
## T_T_LVL:POV  0.069 -0.626   -0.832
```

```
modelB <- glmer(SRT ~ Trial_Type_LVL + POV_LVL + (1 | RECORDING_SESSION_LABEL),
                data = tidy_PS_df,
                family = inverse.gaussian(link = "identity"),
                #More research required on this.s
                start = list(theta = 0.25, fixef = c(ST_SAME = 5.36, ST_DIFF = 5.36, DY_SAME = 5.36)),
                control = glmerControl(optimizer = "bobyqa"))

summary(modelB)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: inverse.gaussian  ( identity )
## Formula: SRT ~ Trial_Type_LVL + POV_LVL + (1 | RECORDING_SESSION_LABEL)
##    Data: tidy_PS_df
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##  23491.7  23520.3 -11740.8  23481.7     2252
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.6114 -0.6722 -0.1952  0.5397  6.5614
##
## Random effects:
##  Groups                   Name        Variance  Std.Dev.
##  RECORDING_SESSION_LABEL (Intercept) 1.006e+02 10.03157
##  Residual                             2.105e-04  0.01451
## Number of obs: 2257, groups:  RECORDING_SESSION_LABEL, 7
##
## Fixed effects:
##                          Estimate Std. Error t value Pr(>|z|)
## (Intercept)               218.030     14.032  15.538   <2e-16 ***
## Trial_Type_LVLdifferent   -1.217      1.972  -0.617    0.537
## POV_LVLdynamic            26.861      1.868  14.378   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) T_T_LV
## Trl_Typ_LVL -0.083
## POV_LVLdynm -0.042 -0.012
```

```
modelC <- glmer(SRT ~ Trial_Type_LVL + (1 | RECORDING_SESSION_LABEL),
                data = tidy_PS_df,
                family = inverse.gaussian(link = "identity"),
                start = list(theta = 0.25, fixef = c(ST_SAME = 5.36, ST_DIFF = 5.36)),
                control = glmerControl(optimizer = "bobyqa"))

summary(modelC)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: inverse.gaussian  ( identity )
## Formula: SRT ~ Trial_Type_LVL + (1 | RECORDING_SESSION_LABEL)
##    Data: tidy_PS_df
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##  23691.4  23714.3 -11841.7  23683.4     2253
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.4589 -0.6757 -0.2096  0.4799  5.7462
##
## Random effects:
##  Groups                   Name        Variance  Std.Dev.
##  RECORDING_SESSION_LABEL (Intercept) 1.059e+02 10.29176
##  Residual                             2.314e-04  0.01521
## Number of obs: 2257, groups:  RECORDING_SESSION_LABEL, 7
##
## Fixed effects:
##                           Estimate Std. Error t value Pr(>|z|)
## (Intercept)              230.6367    13.1446  17.546   <2e-16 ***
## Trial_Type_LVLdifferent  -0.8404     2.0856  -0.403    0.687
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## Trl_Typ_LVL -0.087
```

```
modelD <- glmer(SRT ~ POV_LVL + (1 | RECORDING_SESSION_LABEL),
                data = tidy_PS_df,
                family = inverse.gaussian(link = "identity"),
                start = list(theta = 0.25, fixef = c(ST_SAME = 5.36, ST_DIFF = 5.36)),
                control = glmerControl(optimizer = "bobyqa"))

summary(modelD)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: inverse.gaussian  ( identity )
## Formula: SRT ~ POV_LVL + (1 | RECORDING_SESSION_LABEL)
##    Data: tidy_PS_df
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##  23490.1  23512.9 -11741.0  23482.1     2253
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.6121 -0.6697 -0.1890  0.5354  6.5298
##
## Random effects:
##  Groups                   Name        Variance  Std.Dev.
##  RECORDING_SESSION_LABEL (Intercept) 1.006e+02 10.03044
##  Residual                             2.105e-04  0.01451
## Number of obs: 2257, groups:  RECORDING_SESSION_LABEL, 7
##
## Fixed effects:
##                Estimate Std. Error t value Pr(>|z|)
## (Intercept)     217.192     13.690   15.87   <2e-16 ***
## POV_LVLdynamic   26.846      1.868   14.37   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## POV_LVLdynm -0.049
```

```
modelE <- glmer(SRT ~ 1 + (1 | RECORDING_SESSION_LABEL),
                data = tidy_PS_df,
                family = inverse.gaussian(link = "identity"),
                start = list(theta = 0.25, fixef = c(ST_SAME = 5.36)),
                control = glmerControl(optimizer = "bobyqa"))

summary(modelE)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: inverse.gaussian  ( identity )
## Formula: SRT ~ 1 + (1 | RECORDING_SESSION_LABEL)
##    Data: tidy_PS_df
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##  23689.6  23706.7 -11841.8  23683.6     2254
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -2.4596 -0.6728 -0.2137  0.4754  5.7281
##
## Random effects:
##  Groups                   Name        Variance  Std.Dev.
##  RECORDING_SESSION_LABEL (Intercept) 1.059e+02 10.29140
##  Residual                             2.314e-04  0.01521
## Number of obs: 2257, groups:  RECORDING_SESSION_LABEL, 7
##
## Fixed effects:
##             Estimate Std. Error t value Pr(>|z|)
## (Intercept)   230.05      13.64   16.86   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

#AIC Weighting for stepped model (subtractive from modelA-modelE):

```
require(AICcmodavg)
```

```
## Loading required package: AICcmodavg
```

```
## Warning: package 'AICcmodavg' was built under R version 4.3.3
```

```
##
## Attaching package: 'AICcmodavg'
```

```
## The following object is masked from 'package:lme4':
##
##     checkConv
```

```
# Calculate AIC weights with custom model names
AIC_weights <- aictab(list(modelA, modelB, modelC, modelD, modelE),
                      modnames = c("Model A", "Model B", "Model C", "Model D", "Model E"))

# Print the AIC weights table
print(AIC_weights)
```

```
##
## Model selection based on AICc:
##
##         K     AICc Delta_AICc AICcWt Cum.Wt        LL
## Model D 4 23490.08       0.00   0.60   0.60 -11741.03
## Model B 5 23491.71       1.63   0.27   0.87 -11740.84
## Model A 6 23493.13       3.05   0.13   1.00 -11740.54
## Model E 3 23689.56     199.49   0.00   1.00 -11841.78
## Model C 4 23691.41     201.33   0.00   1.00 -11841.69
```

```
#ModelD best fit as super significant differences between POV tasks.
#Expected, they are different tasks altogether. Not part of hypothesis
#Interestingly, model B is both predictors as main effects
#Without the interaction. This may be viable. Continuing with
#ModelB and ModelA assumed best fits
```

#Interaction plots (1) - plotting predicted SRT:

```
require(sjPlot)
```

```
## Loading required package: sjPlot
```

```
## Warning: package 'sjPlot' was built under R version 4.3.3
```

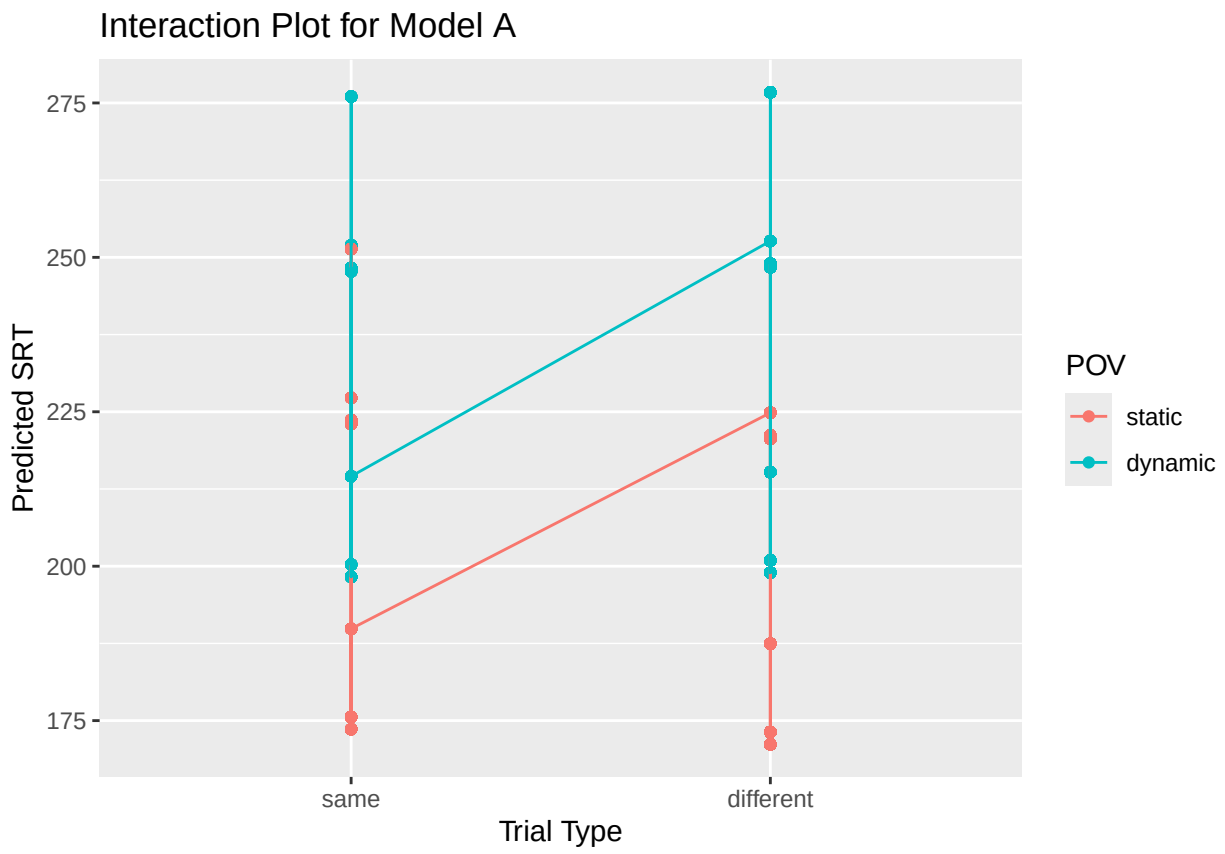
```
##
## Attaching package: 'sjPlot'
```

```
## The following objects are masked from 'package:cowplot':
##
##     plot_grid, save_plot
```

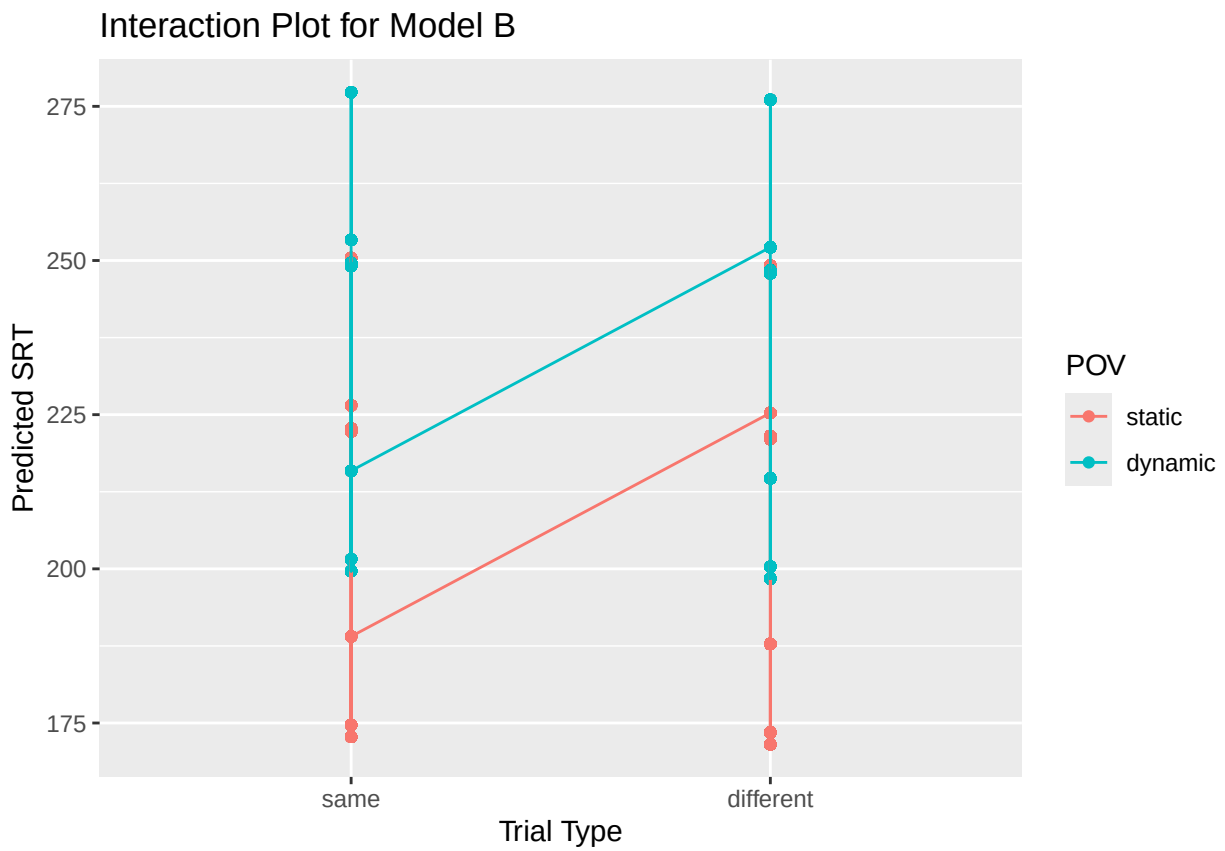
```
require(sjstats)
```

```
## Loading required package: sjstats
```

```
# Predicted values for model A
predicted_A <- predict(modelA, type = "response")

# Predicted values for model B
predicted_B <- predict(modelB, type = "response")

# Create a data frame with the original data and predicted values
plot_data_A <- data.frame(SRT = tidy_PS_df$SRT, Predicted_A = predicted_A)
plot_data_B <- data.frame(SRT = tidy_PS_df$SRT, Predicted_B = predicted_B)

# Load required packages
library(ggplot2)

# Combine predicted values with original data
plot_data_A <- cbind(tidy_PS_df, predicted_A)
plot_data_B <- cbind(tidy_PS_df, predicted_B)

# Plot interaction for model A
ggplot(plot_data_A, aes(x = Trial_Type, y = predicted_A, color = POV)) +
  geom_point() +
  geom_line(aes(group = POV)) +
  labs(title = "Interaction Plot for Model A",
       x = "Trial Type", y = "Predicted SRT")
```

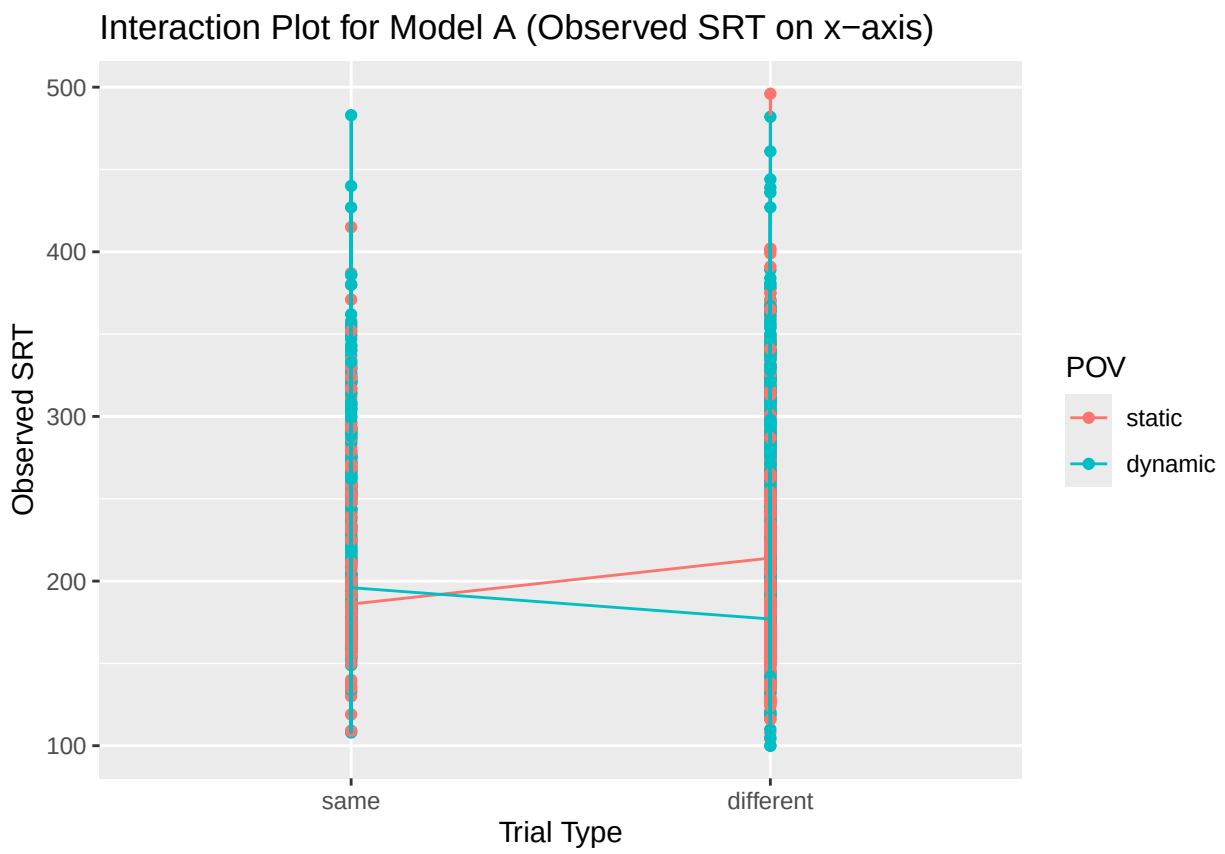



```
# Plot interaction for model B
ggplot(plot_data_B, aes(x = Trial_Type, y = predicted_B, color = POV)) +
  geom_point() +
  geom_line(aes(group = POV)) +
  labs(title = "Interaction Plot for Model B",
       x = "Trial Type", y = "Predicted SRT")
```

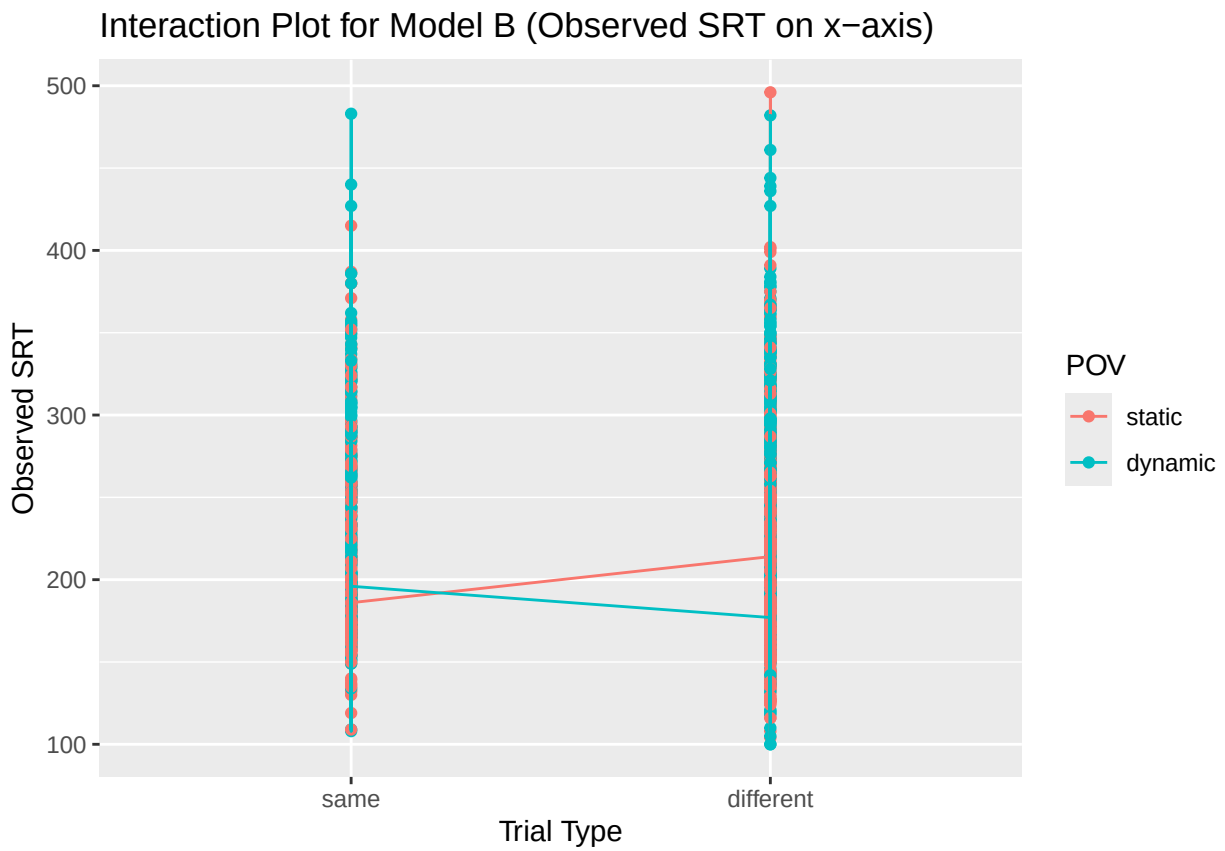


#Interaction plots (2) - plotting observed SRT:

```
require(sjPlot)
require(sjstats)
# Predicted values for model A
predicted_A <- predict(modelA, type = "response")

# Predicted values for model B
predicted_B <- predict(modelB, type = "response")

# Create a data frame with the original data and predicted values
plot_data_A <- data.frame(SRT = tidy_PS_df$SRT, Predicted_A = predicted_A)
plot_data_B <- data.frame(SRT = tidy_PS_df$SRT, Predicted_B = predicted_B)

# Load required packages
library(ggplot2)

# Combine predicted values with original data
plot_data_A <- cbind(tidy_PS_df, predicted_A)
plot_data_B <- cbind(tidy_PS_df, predicted_B)

# Plot interaction for model A with observed SRT on x-axis
ggplot(plot_data_A, aes(x = Trial_Type, y = SRT, color = POV)) +
  geom_point() +
  geom_line(aes(group = POV)) +
  labs(title = "Interaction Plot for Model A (Observed SRT on x-axis)",
       x = "Trial Type", y = "Observed SRT")
```



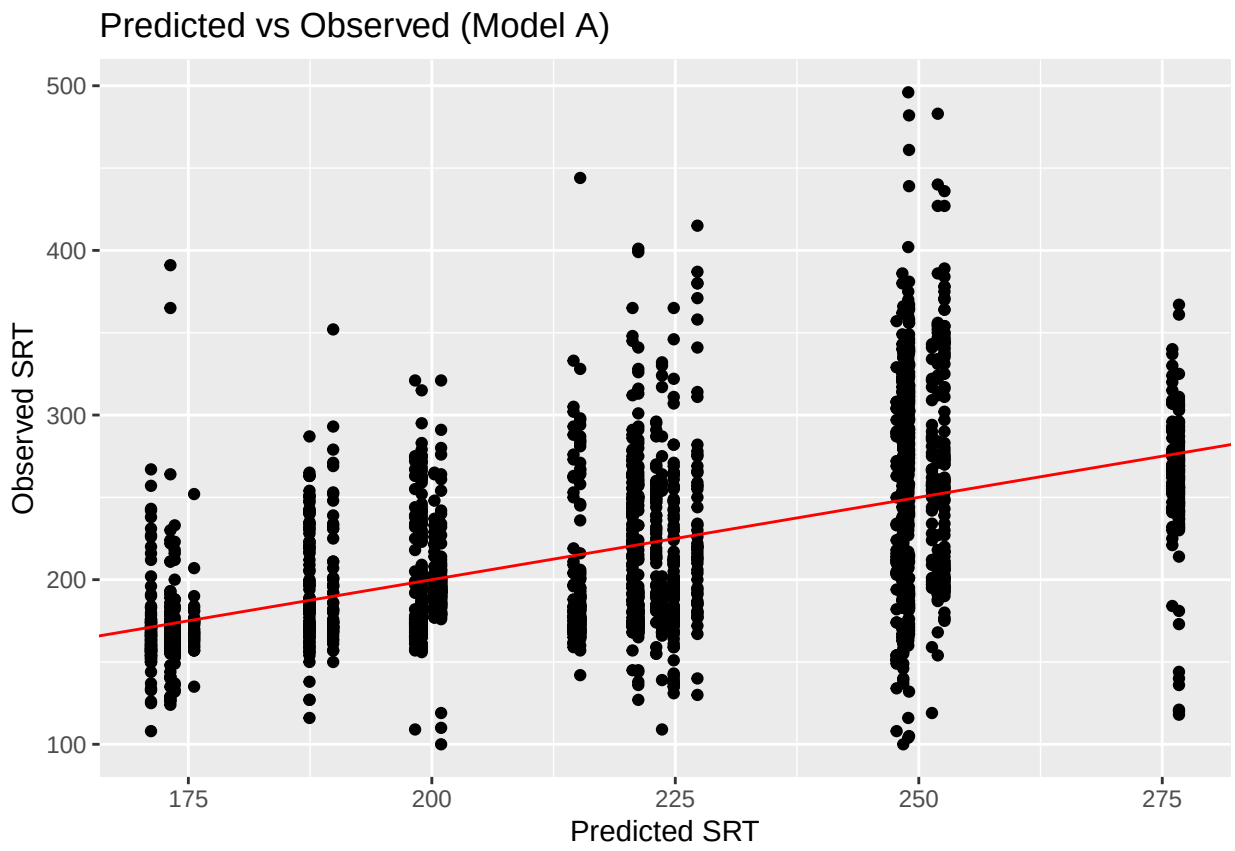

```
# Plot interaction for model B with observed SRT on x-axis
ggplot(plot_data_B, aes(x = Trial_Type, y = SRT, color = POV)) +
  geom_point() +
  geom_line(aes(group = POV)) +
  labs(title = "Interaction Plot for Model B (Observed SRT on x-axis)",
       x = "Trial Type", y = "Observed SRT")
```



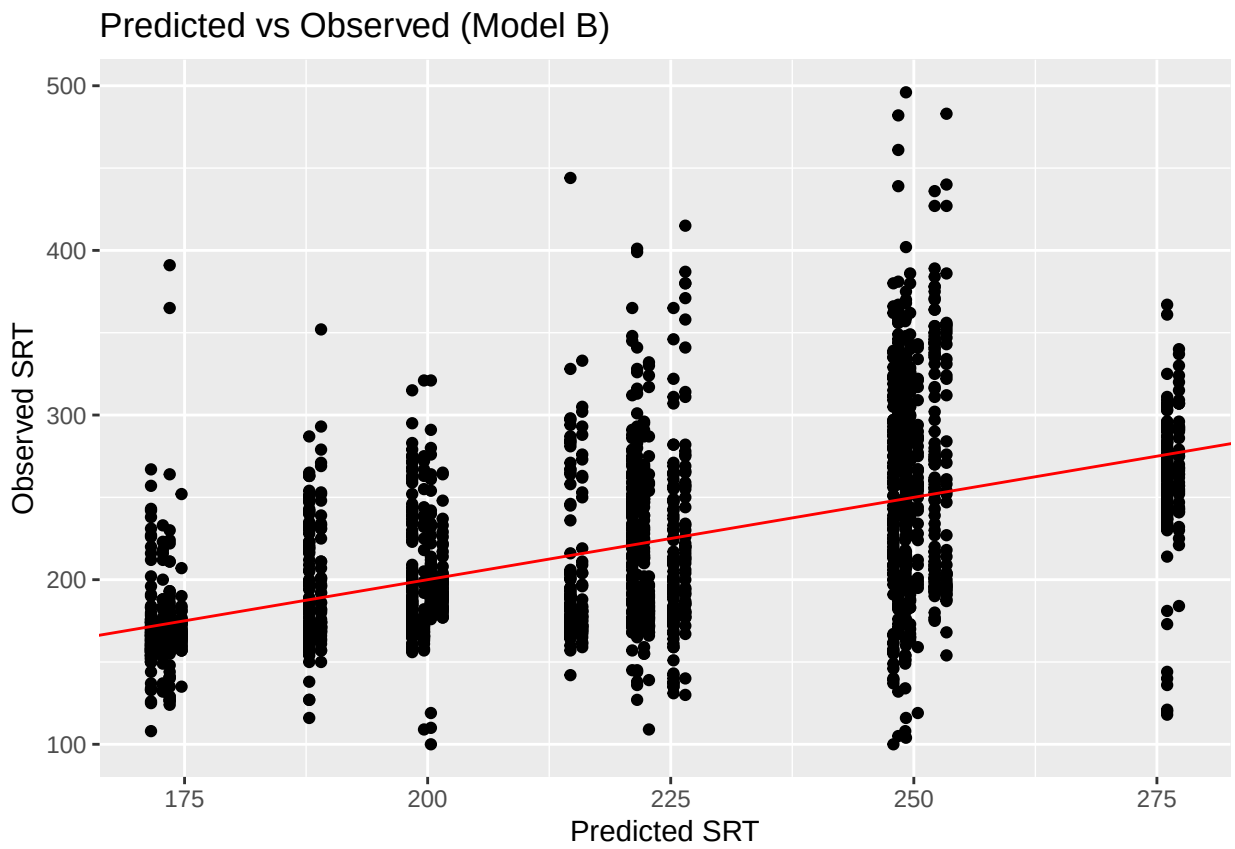

```
#Opposite of IOR effect? Not plotted backwards surely...
#Maybe factor levelling has something to do with this?! Be sure to ask,
#thats a big deal if so. Might be simply renaming objects,
#which ruins interpretation of glmers.
```

#Predicted vs. Observed (1) - Scatter plots:

```
predicted_A <- predict(modelA, type = "response")

# Calculate predicted values for model B
predicted_B <- predict(modelB, type = "response")

# Create data frames with observed and predicted values
plot_data_A <- data.frame(Observed = tidy_PS_df$SRT, Predicted = predicted_A)
plot_data_B <- data.frame(Observed = tidy_PS_df$SRT, Predicted = predicted_B)

# Plot predicted vs observed for model A
ggplot(plot_data_A, aes(x = Predicted, y = Observed)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +  # Add a 45-degree line
  labs(title = "Predicted vs Observed (Model A)",
       x = "Predicted SRT",
       y = "Observed SRT")
```



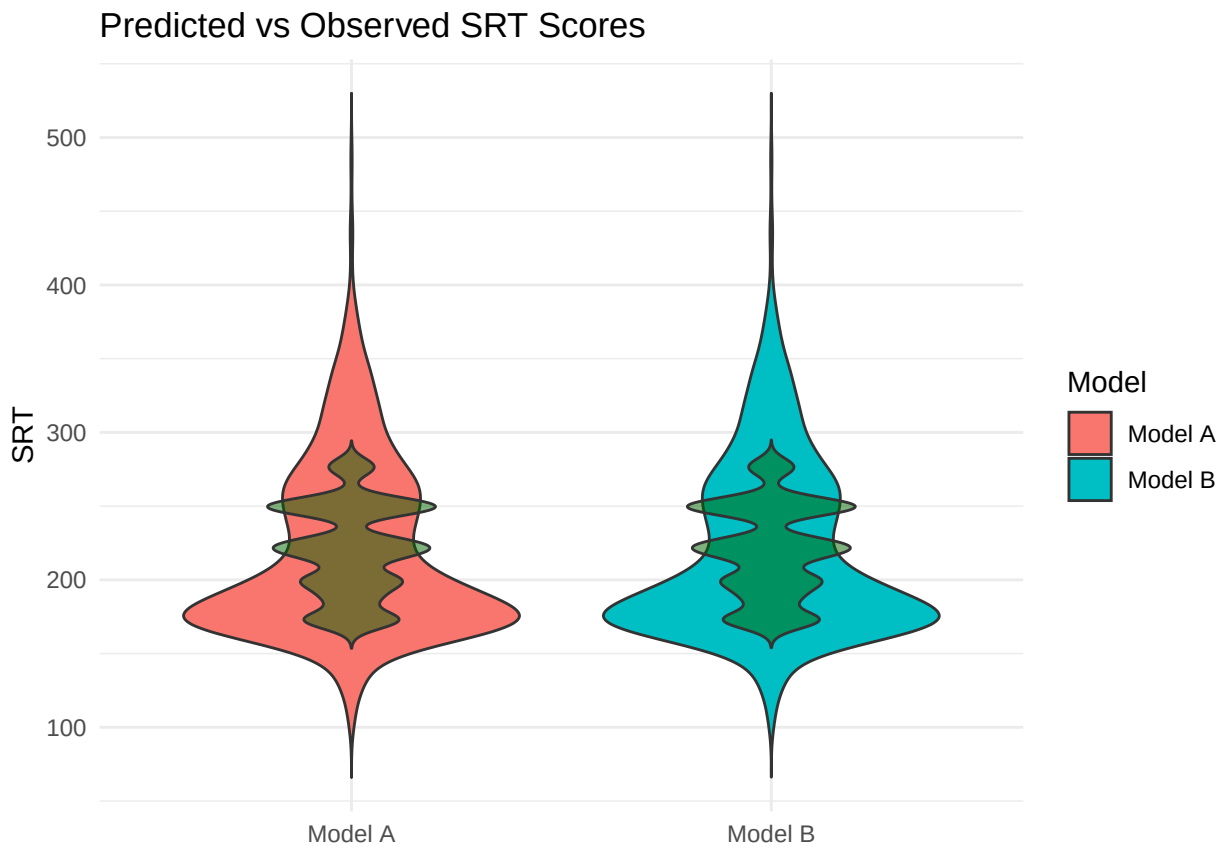

```r
# Plot predicted vs observed for model B
ggplot(plot_data_B, aes(x = Predicted, y = Observed)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +  # Add a 45-degree line
  labs(title = "Predicted vs Observed (Model B)",
       x = "Predicted SRT",
       y = "Observed SRT")
```



#Predicted vs. Observed (2) - Violin Plots:

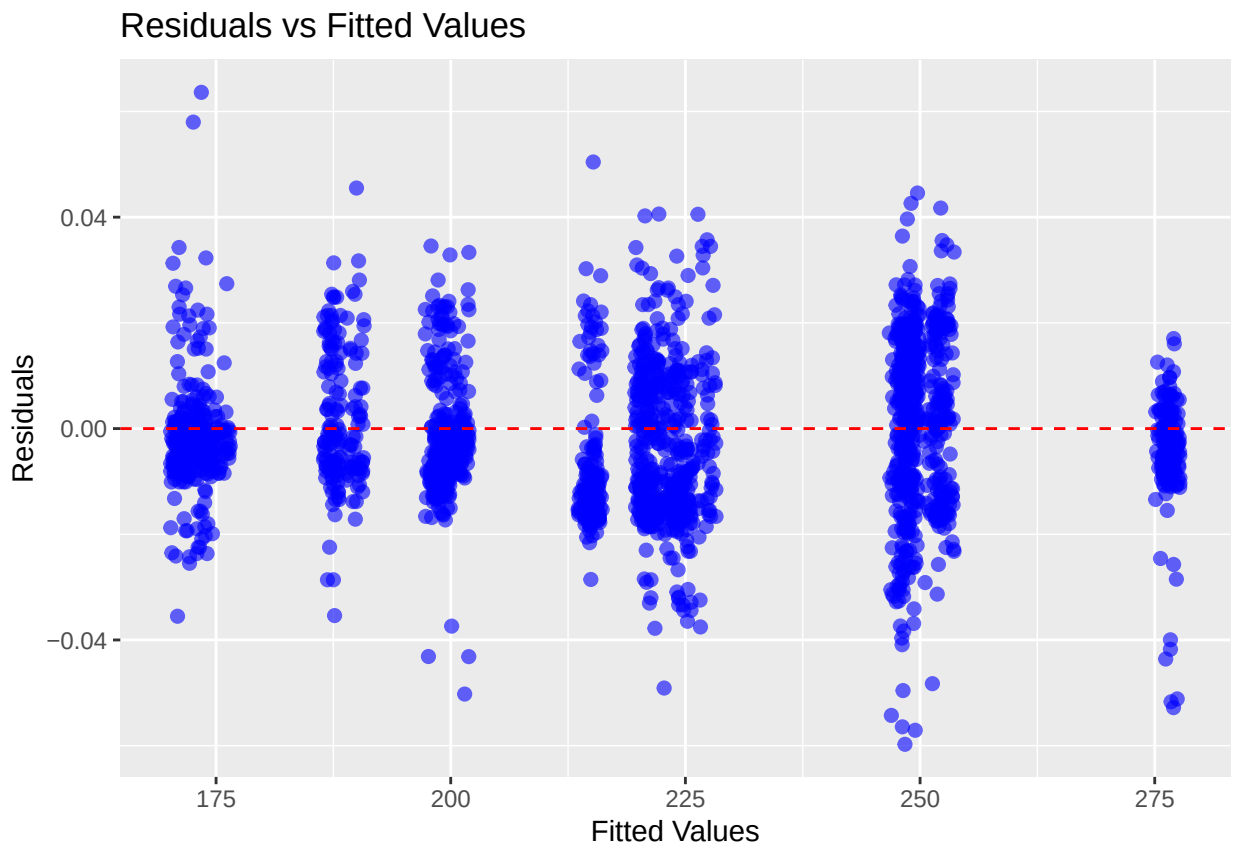
```
require(sjPlot)
require(sjstats)
require(ggplot2)
# Predicted values for model A
predicted_A <- predict(modelA, type = "response")

# Predicted values for model B
predicted_B <- predict(modelB, type = "response")

# Create a data frame with the original data and predicted values
plot_data_A <- data.frame(SRT = tidy_PS_df$SRT, Predicted_A = predicted_A)
plot_data_B <- data.frame(SRT = tidy_PS_df$SRT, Predicted_B = predicted_B)

# Rename columns in plot_data_B to match plot_data_A
colnames(plot_data_B) <- colnames(plot_data_A)

# Combine predicted values with original data
plot_data_combined <- rbind(
  cbind(plot_data_A, Model = "Model A"),
  cbind(plot_data_B, Model = "Model B")
)

# Plot observed SRT scores with predicted SRT scores as separate violins
ggplot(plot_data_combined, aes(x = Model, y = SRT, fill = Model)) +
  geom_violin(trim = FALSE, width = 0.8) +
```

```
  geom_violin(data = plot_data_combined, aes(x = Model, y = Predicted_A),
              fill = "darkgreen", trim = FALSE, width = 0.4, alpha = 0.5) +
  labs(title = "Predicted vs Observed SRT Scores",
       x = NULL, y = "SRT") +
  theme_minimal()
```



#Residual vs fitted for model A:

```
library(ggplot2)
library(lme4)

# Extract fitted values and residuals from the model
fitted_values <- fitted(modelA)
residuals <- resid(modelA)

# Create a data frame with fitted values and residuals
residuals_df <- data.frame(Fitted_Values = fitted_values, Residuals = residuals)

# Create a scatter plot with ggplot2
ggplot(residuals_df, aes(x = Fitted_Values, y = Residuals)) +
  geom_point(position = position_jitter(width = 1), color = "blue", size = 2, alpha = 0.6) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(x = "Fitted Values", y = "Residuals",
       title = "Residuals vs Fitted Values")
```



#Now the same, but for TRT:

```
require(lme4)

#USING LOG START VALUES AGAIN.
#UPDATED FOR TRT OBVIOUSLY
#BUT RAW-TRT = PIRLS LOOP!!!
#FOR MORE COMPLEX MODELS...

#Recode glmer so that levelled objects are called:
modelF <- glmer(TRT ~ Trial_Type_LVL * POV_LVL + (1 | RECORDING_SESSION_LABEL),
                data = tidy_PS_df,
                family = inverse.gaussian(link = "identity"),
                #More research required on this.s
                start = list(theta = 5.80, fixef = c(ST_SAME = 7.01, ST_DIFF = 7.01, DY_SAME = 7.01, DY_
                control = glmerControl(optimizer = "bobyqa"))

summary(modelF)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: inverse.gaussian  ( identity )
## Formula: TRT ~ Trial_Type_LVL * POV_LVL + (1 | RECORDING_SESSION_LABEL)
##    Data: tidy_PS_df
## Control: glmerControl(optimizer = "bobyqa")
##
```

```
##      AIC      BIC   logLik deviance df.resid
##  32654.3  32688.7 -16321.2  32642.3     2251
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.6341 -0.7872 -0.0150  0.6367  4.6648
##
## Random effects:
##  Groups                   Name        Variance  Std.Dev.
##  RECORDING_SESSION_LABEL (Intercept) 1.188e+03 34.461931
##  Residual                             7.949e-05  0.008916
## Number of obs: 2257, groups:  RECORDING_SESSION_LABEL, 7
##
## Fixed effects:
##                                         Estimate Std. Error t value Pr(>|z|)
## (Intercept)                             1112.182     30.448  36.527   <2e-16 ***
## Trial_Type_LVLdifferent                   17.863     17.828   1.002   0.3164
## POV_LVLdynamic                            -6.622     19.476  -0.340   0.7338
## Trial_Type_LVLdifferent:POV_LVLdynamic   -47.195     22.100  -2.136   0.0327 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) Tr_T_LVL POV_LV
## Trl_Typ_LVL -0.256
## POV_LVLdynm -0.210  0.314
## T_T_LVL:POV  0.147 -0.529   -0.683
```

```
modelG <- glmer(TRT ~ Trial_Type_LVL + POV_LVL + (1 | RECORDING_SESSION_LABEL),
                data = tidy_PS_df,
                family = inverse.gaussian(link = "identity"),
                #More research required on this.s
                start = list(theta = 5.80, fixef = c(ST_SAME = 7.01, ST_DIFF = 7.01, DY_SAME = 7.01)),
                control = glmerControl(optimizer = "bobyqa"))

summary(modelG)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: inverse.gaussian  ( identity )
## Formula: TRT ~ Trial_Type_LVL + POV_LVL + (1 | RECORDING_SESSION_LABEL)
##    Data: tidy_PS_df
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##  32654.4  32683.0 -16322.2  32644.4     2252
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.6251 -0.7887 -0.0130  0.6392  4.7469
##
## Random effects:
##  Groups                   Name        Variance  Std.Dev.
##  RECORDING_SESSION_LABEL (Intercept) 1.183e+03 34.394886
```

```
##  Residual                             7.948e-05  0.008915
## Number of obs: 2257, groups:  RECORDING_SESSION_LABEL, 7
##
## Fixed effects:
##                         Estimate Std. Error t value Pr(>|z|)
## (Intercept)             1128.988     27.207  41.496  < 2e-16 ***
## Trial_Type_LVLdifferent   -6.349     15.627  -0.406  0.68452
## POV_LVLdynamic           -39.445     14.042  -2.809  0.00497 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) T_T_LV
## Trl_Typ_LVL -0.201
## POV_LVLdynm -0.203  0.049
```

```
modelH <- glmer(TRT ~ Trial_Type_LVL + (1 | RECORDING_SESSION_LABEL),
                data = tidy_PS_df,
                family = inverse.gaussian(link = "identity"),
                start = list(theta = 331, fixef = c(ST_SAME = 1102, ST_DIFF = 1102)),
                control = glmerControl(optimizer = "bobyqa"))

summary(modelH)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: inverse.gaussian  ( identity )
## Formula: TRT ~ Trial_Type_LVL + (1 | RECORDING_SESSION_LABEL)
##    Data: tidy_PS_df
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##  32659.4  32682.2 -16325.7  32651.4     2253
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.6315 -0.7677 -0.0103  0.6487  4.9526
##
## Random effects:
##  Groups                   Name        Variance  Std.Dev.
##  RECORDING_SESSION_LABEL (Intercept) 1.199e+03 34.626731
##  Residual                             7.969e-05  0.008927
## Number of obs: 2257, groups:  RECORDING_SESSION_LABEL, 7
##
## Fixed effects:
##                         Estimate Std. Error t value Pr(>|z|)
## (Intercept)             1109.205     25.311  43.822   <2e-16 ***
## Trial_Type_LVLdifferent   -5.149     16.721  -0.308    0.758
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## Trl_Typ_LVL -0.287
```

```r
modelI <- glmer(TRT ~ POV_LVL + (1 | RECORDING_SESSION_LABEL),
                data = tidy_PS_df,
                family = inverse.gaussian(link = "identity"),
                start = list(theta = 331, fixef = c(ST_SAME = 1102, ST_DIFF = 1102)),
                control = glmerControl(optimizer = "bobyqa"))

summary(modelI)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: inverse.gaussian  ( identity )
## Formula: TRT ~ POV_LVL + (1 | RECORDING_SESSION_LABEL)
##    Data: tidy_PS_df
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##  32652.6  32675.5 -16322.3  32644.6     2253
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.6277 -0.7830 -0.0139  0.6457  4.7264
##
## Random effects:
##  Groups                   Name        Variance  Std.Dev.
##  RECORDING_SESSION_LABEL (Intercept) 1.181e+03 34.365085
##  Residual                             7.947e-05  0.008915
## Number of obs: 2257, groups:  RECORDING_SESSION_LABEL, 7
##
## Fixed effects:
##                Estimate Std. Error t value Pr(>|z|)
## (Intercept)     1124.48      26.42  42.557  < 2e-16 ***
## POV_LVLdynamic   -39.28      13.50  -2.909  0.00362 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## POV_LVLdynm -0.149
```

```r
modelJ <- glmer(TRT ~ 1 + (1 | RECORDING_SESSION_LABEL),
                data = tidy_PS_df,
                family = inverse.gaussian(link = "identity"),
                start = list(theta = 331, fixef = c(ST_SAME = 1102)),
                control = glmerControl(optimizer = "bobyqa"))

summary(modelJ)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: inverse.gaussian  ( identity )
## Formula: TRT ~ 1 + (1 | RECORDING_SESSION_LABEL)
##    Data: tidy_PS_df
```

```
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##  32657.5  32674.6 -16325.7  32651.5     2254
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.6336 -0.7689 -0.0099  0.6467  4.9342
##
## Random effects:
##  Groups                   Name        Variance  Std.Dev.
##  RECORDING_SESSION_LABEL (Intercept) 1.197e+03 34.602895
##  Residual                             7.968e-05  0.008926
## Number of obs: 2257, groups:  RECORDING_SESSION_LABEL, 7
##
## Fixed effects:
##             Estimate Std. Error t value Pr(>|z|)
## (Intercept)  1105.62      28.99   38.14   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#Interesting finding interaction wise
#In model F (Most complex for TRT)
#Must ask Jason about that...
```

#AIC Weighting for stepped model (subtractive from modelF-modelJ):

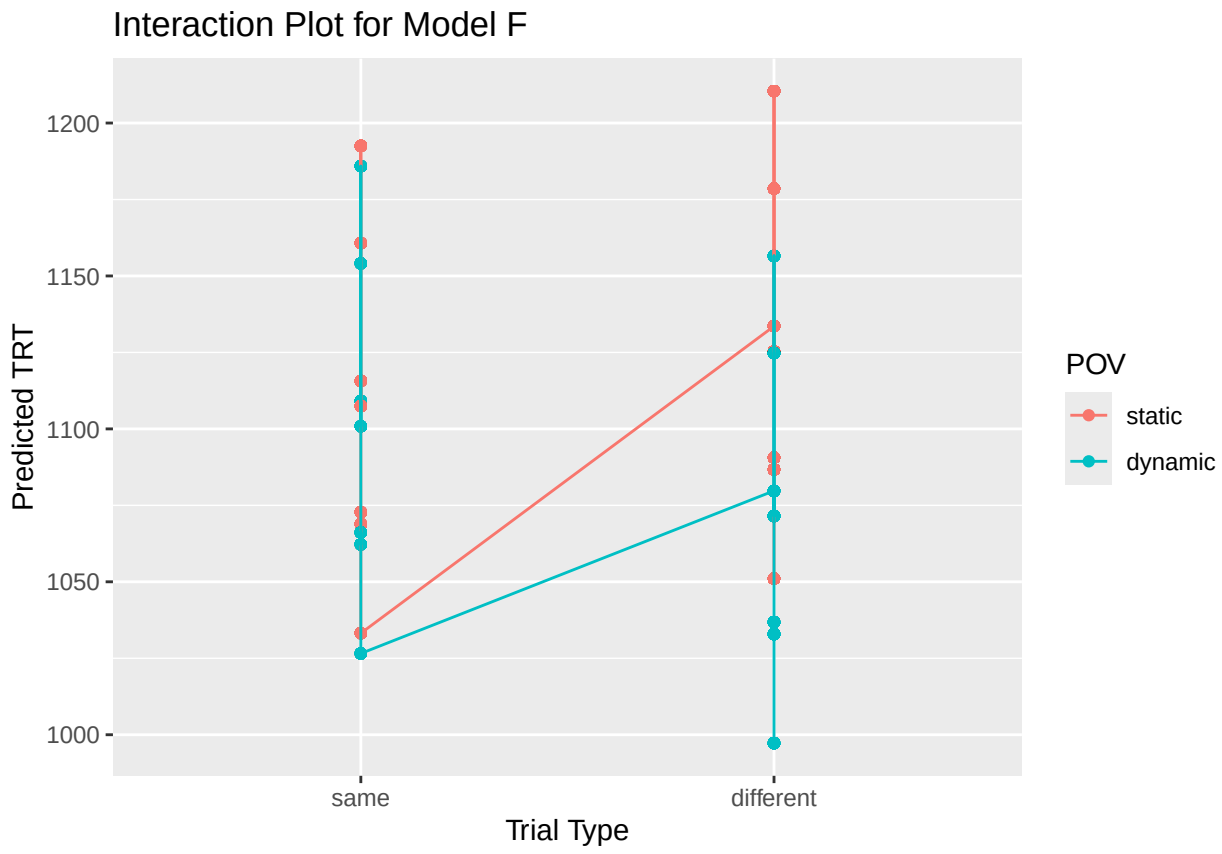
```
require(AICcmodavg)

# Calculate AIC weights with custom model names
AIC_weights_TRT <- aictab(list(modelF, modelG, modelH, modelI, modelJ),
                          modnames = c("Model F", "Model G", "Model H", "Model I", "Model J"))

# Print the AIC weights table
print(AIC_weights_TRT)
```

```
##
## Model selection based on AICc:
##
##         K     AICc Delta_AICc AICcWt Cum.Wt        LL
## Model I 4 32652.61       0.00   0.52   0.52 -16322.30
## Model F 6 32654.38       1.77   0.21   0.73 -16321.17
## Model G 5 32654.46       1.86   0.20   0.94 -16322.22
## Model J 3 32657.46       4.85   0.05   0.98 -16325.73
## Model H 4 32659.37       6.76   0.02   1.00 -16325.68
```

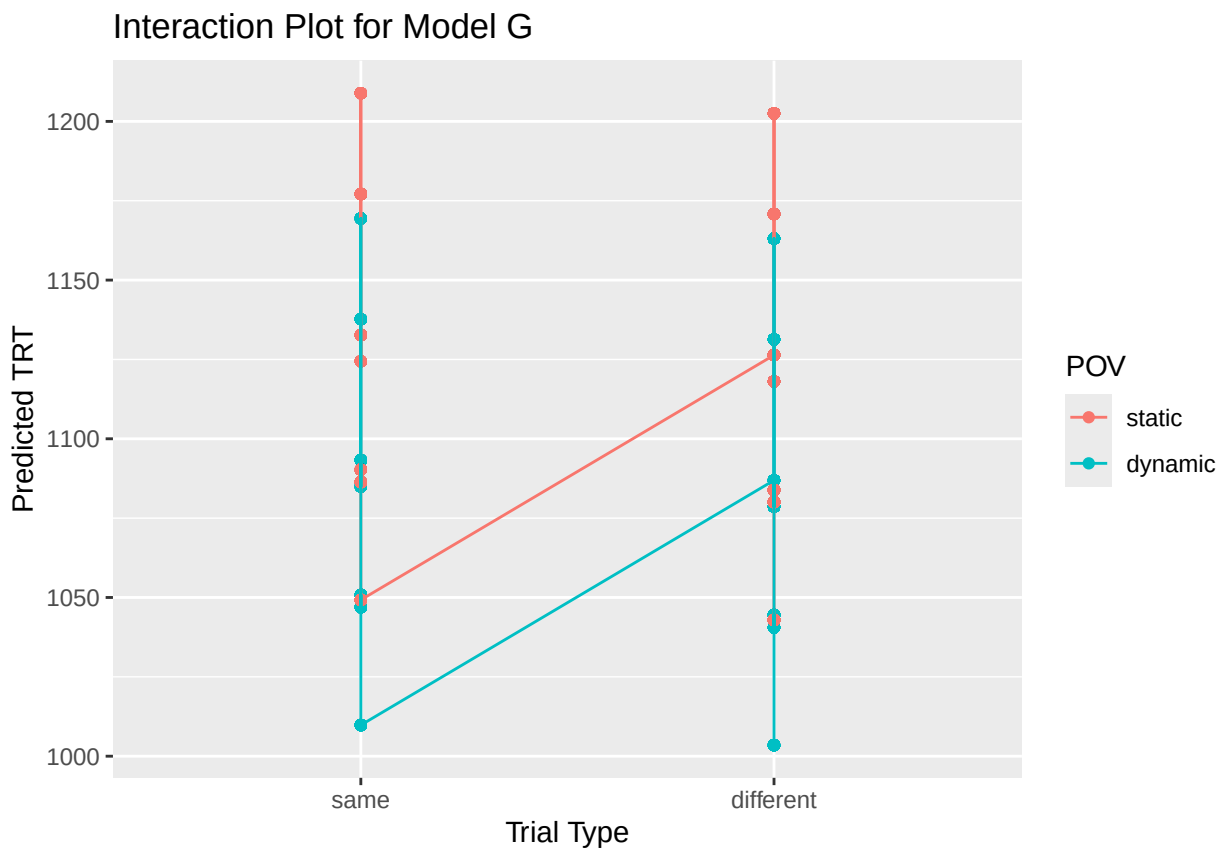
```
#Not quite the same weighting as SRT in terms of AIC, so continue with model F and G
#(TRT ~ Both predictors as interaction or main effects respectively).
#Actually a better fit with interaction present!
```

#Interaction plots (3) - plotting predicted TRT:

```r
require(sjPlot)
require(sjstats)
# Predicted values for model A
predicted_F <- predict(modelF, type = "response")

# Predicted values for model B
predicted_G <- predict(modelG, type = "response")

# Create a data frame with the original data and predicted values
plot_data_F <- data.frame(TRT = tidy_PS_df$TRT, Predicted_F = predicted_F)
plot_data_G <- data.frame(TRT = tidy_PS_df$TRT, Predicted_G = predicted_G)

# Load required packages
library(ggplot2)

# Combine predicted values with original data
plot_data_F <- cbind(tidy_PS_df, predicted_F)
plot_data_G <- cbind(tidy_PS_df, predicted_G)

# Plot interaction for model A
ggplot(plot_data_F, aes(x = Trial_Type, y = predicted_F, color = POV)) +
  geom_point() +
  geom_line(aes(group = POV)) +
  labs(title = "Interaction Plot for Model F",
       x = "Trial Type", y = "Predicted TRT")
```



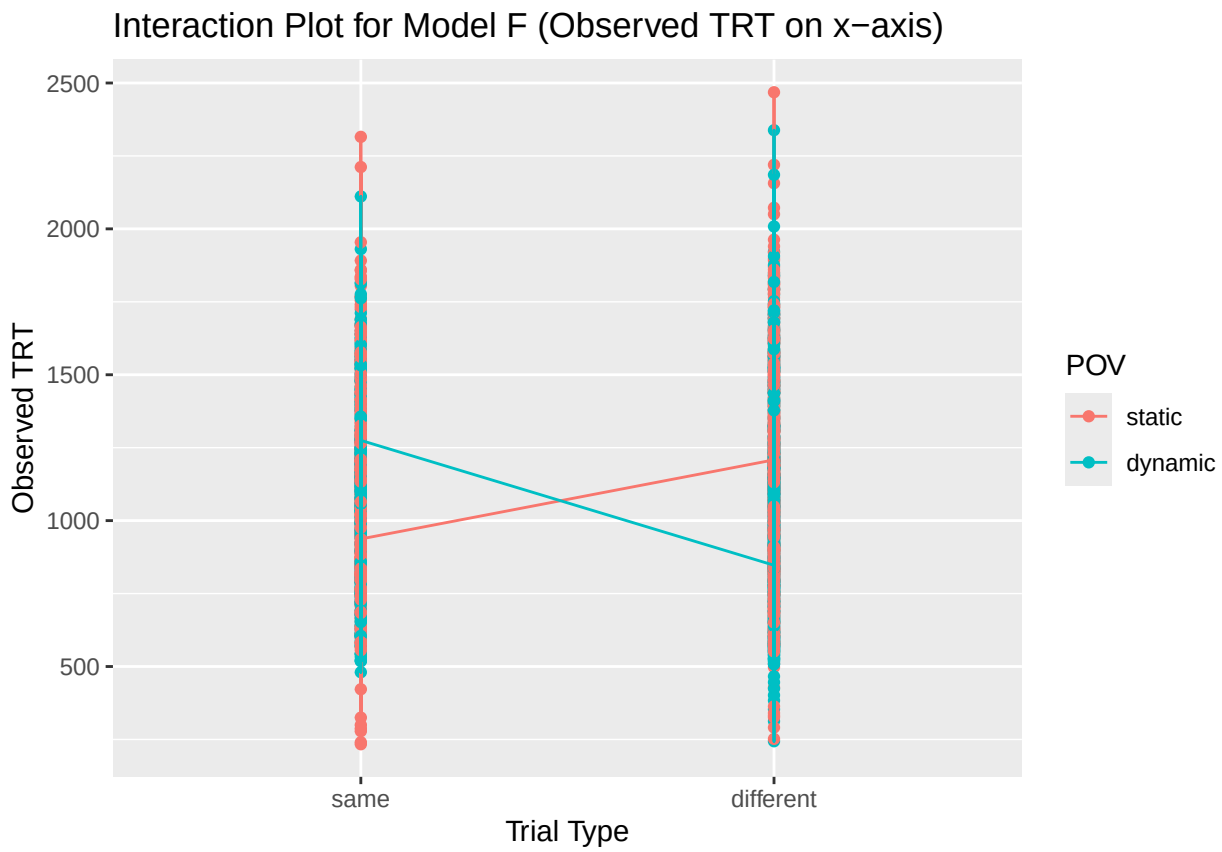
```r
# Plot interaction for model B
ggplot(plot_data_G, aes(x = Trial_Type, y = predicted_G, color = POV)) +
  geom_point() +
  geom_line(aes(group = POV)) +
  labs(title = "Interaction Plot for Model G",
       x = "Trial Type", y = "Predicted TRT")
```



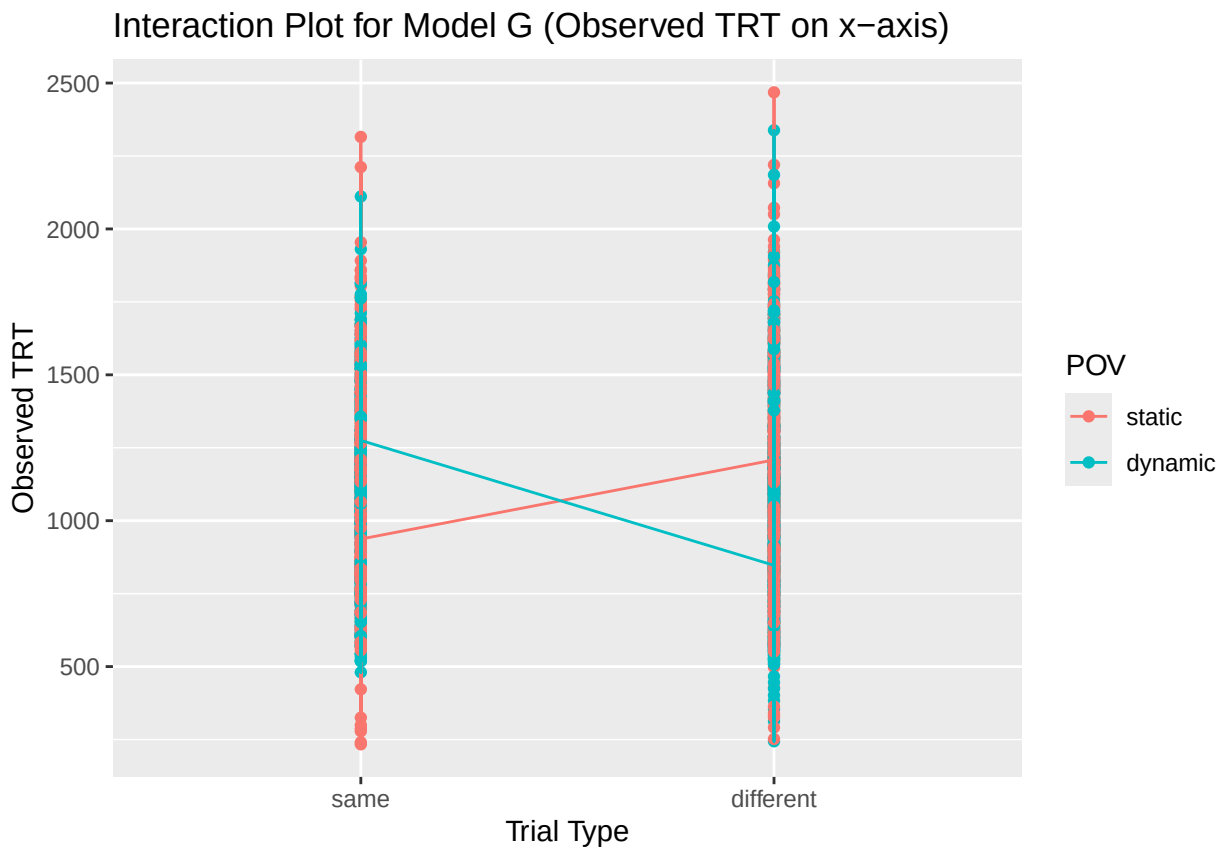
#Interaction plots (4) - plotting observed TRT:

```r
require(sjPlot)
require(sjstats)
# Predicted values for model A
predicted_F <- predict(modelF, type = "response")

# Predicted values for model B
predicted_G <- predict(modelG, type = "response")

# Create a data frame with the original data and predicted values
plot_data_F <- data.frame(TRT = tidy_PS_df$TRT, Predicted_F = predicted_F)
plot_data_G <- data.frame(TRT = tidy_PS_df$TRT, Predicted_G = predicted_G)

# Load required packages
library(ggplot2)

# Combine predicted values with original data
plot_data_F <- cbind(tidy_PS_df, predicted_F)
```

```r
plot_data_G <- cbind(tidy_PS_df, predicted_G)

# Plot interaction for model A with observed SRT on x-axis
ggplot(plot_data_F, aes(x = Trial_Type, y = TRT, color = POV)) +
  geom_point() +
  geom_line(aes(group = POV)) +
  labs(title = "Interaction Plot for Model F (Observed TRT on x-axis)",
       x = "Trial Type", y = "Observed TRT")
```

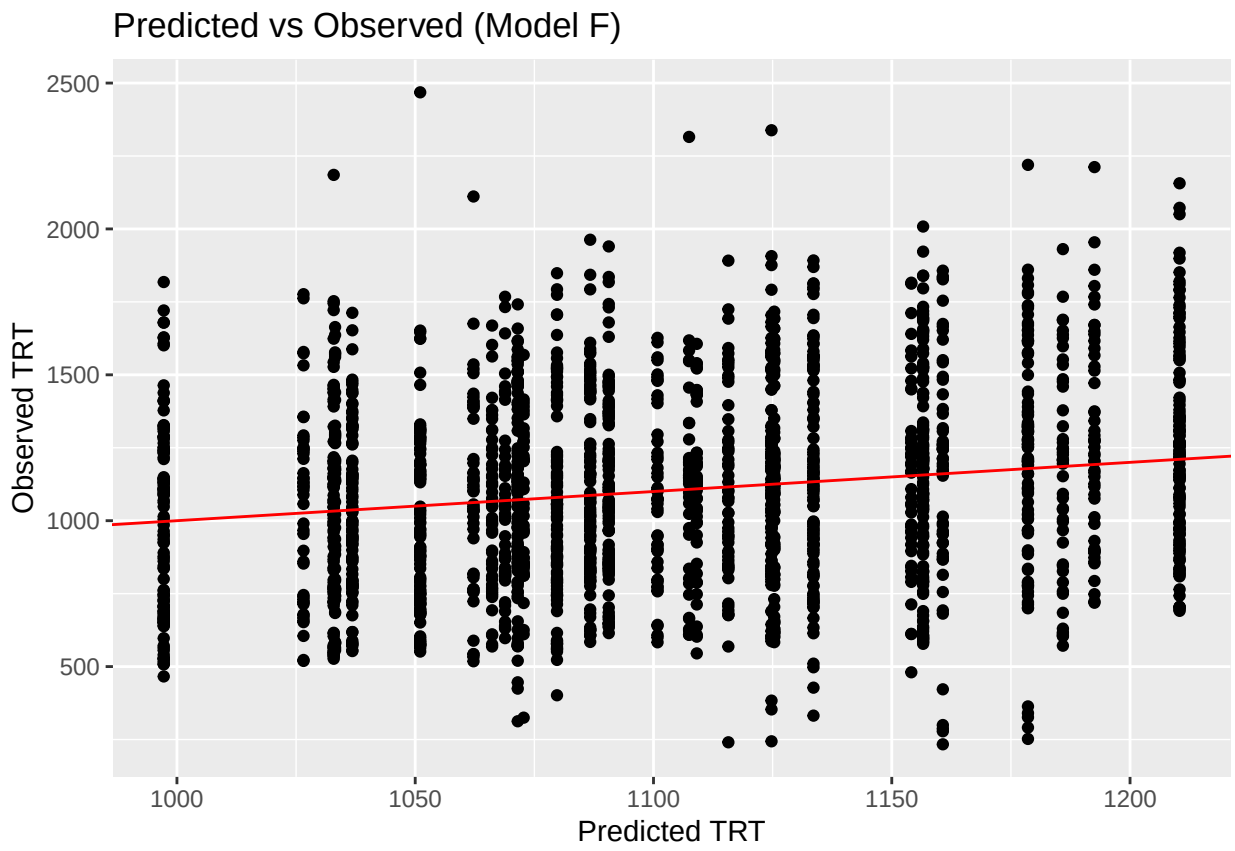



```r
# Plot interaction for model B with observed SRT on x-axis
ggplot(plot_data_G, aes(x = Trial_Type, y = TRT, color = POV)) +
  geom_point() +
  geom_line(aes(group = POV)) +
  labs(title = "Interaction Plot for Model G (Observed TRT on x-axis)",
       x = "Trial Type", y = "Observed TRT")
```

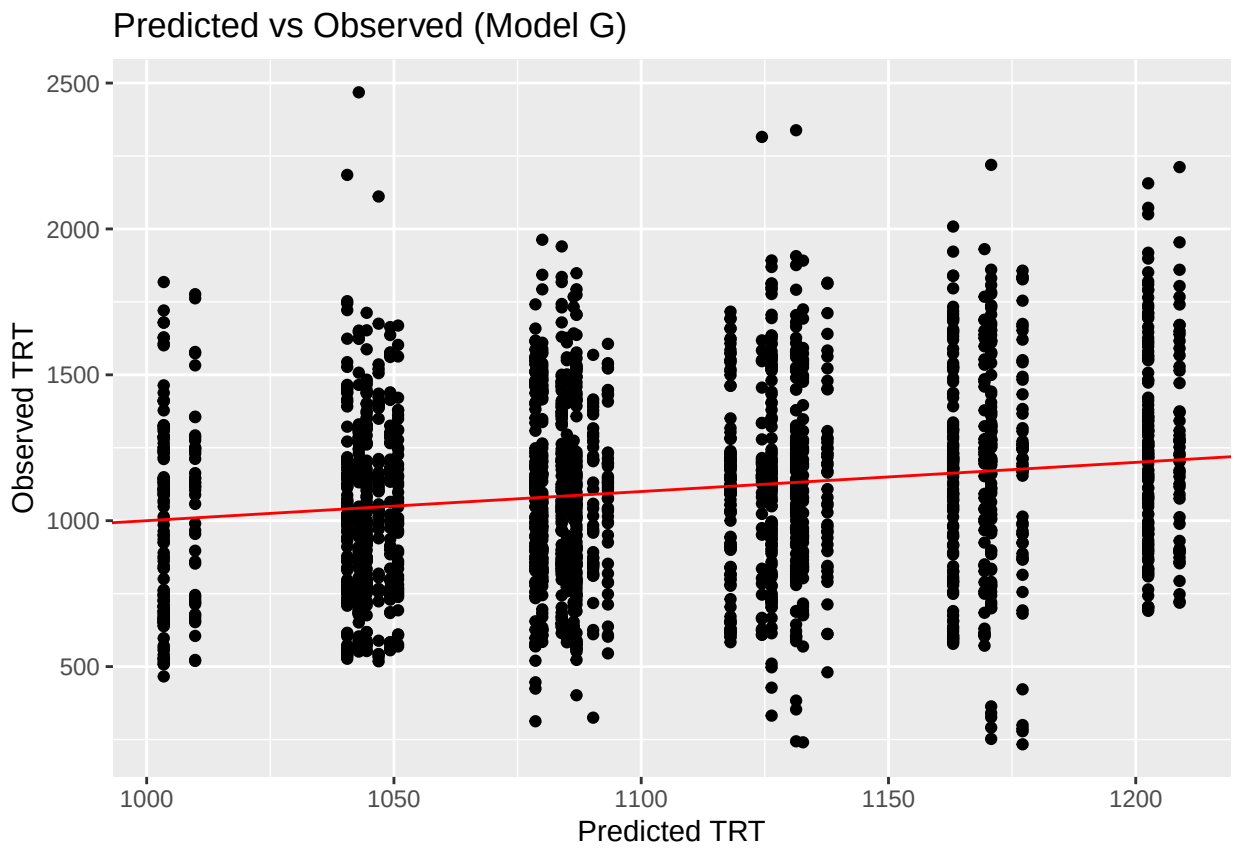


#Predicted vs. Observed (3) - Scatter plots:

```r
predicted_F <- predict(modelF, type = "response")

# Calculate predicted values for model B
predicted_G <- predict(modelG, type = "response")

# Create data frames with observed and predicted values
plot_data_F <- data.frame(Observed = tidy_PS_df$TRT, Predicted = predicted_F)
plot_data_G <- data.frame(Observed = tidy_PS_df$TRT, Predicted = predicted_G)

# Plot predicted vs observed for model A
ggplot(plot_data_F, aes(x = Predicted, y = Observed)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +  # Add a 45-degree line
  labs(title = "Predicted vs Observed (Model F)",
       x = "Predicted TRT",
       y = "Observed TRT")
```



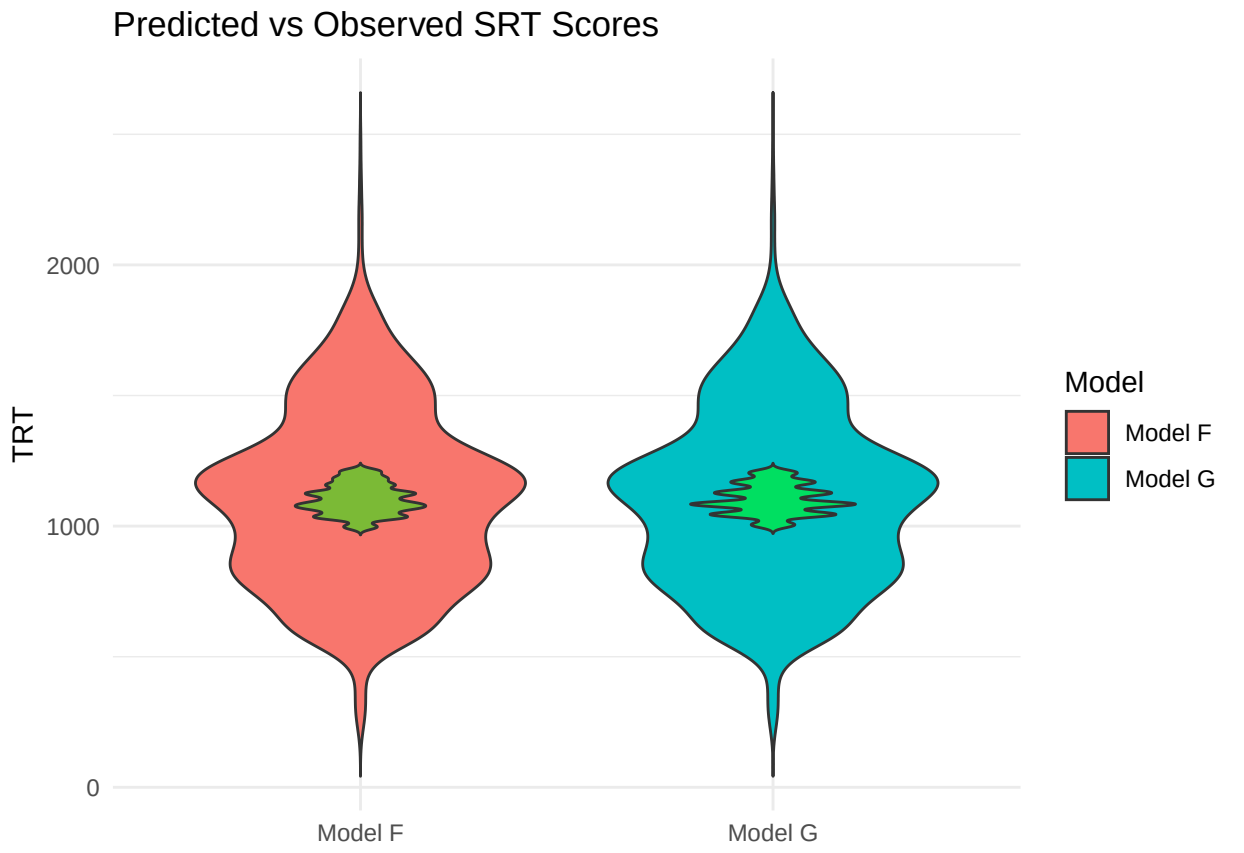

```r
# Plot predicted vs observed for model B
ggplot(plot_data_G, aes(x = Predicted, y = Observed)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +  # Add a 45-degree line
  labs(title = "Predicted vs Observed (Model G)",
       x = "Predicted TRT",
       y = "Observed TRT")
```



#Predicted vs. Observed (4) - Violin Plots:

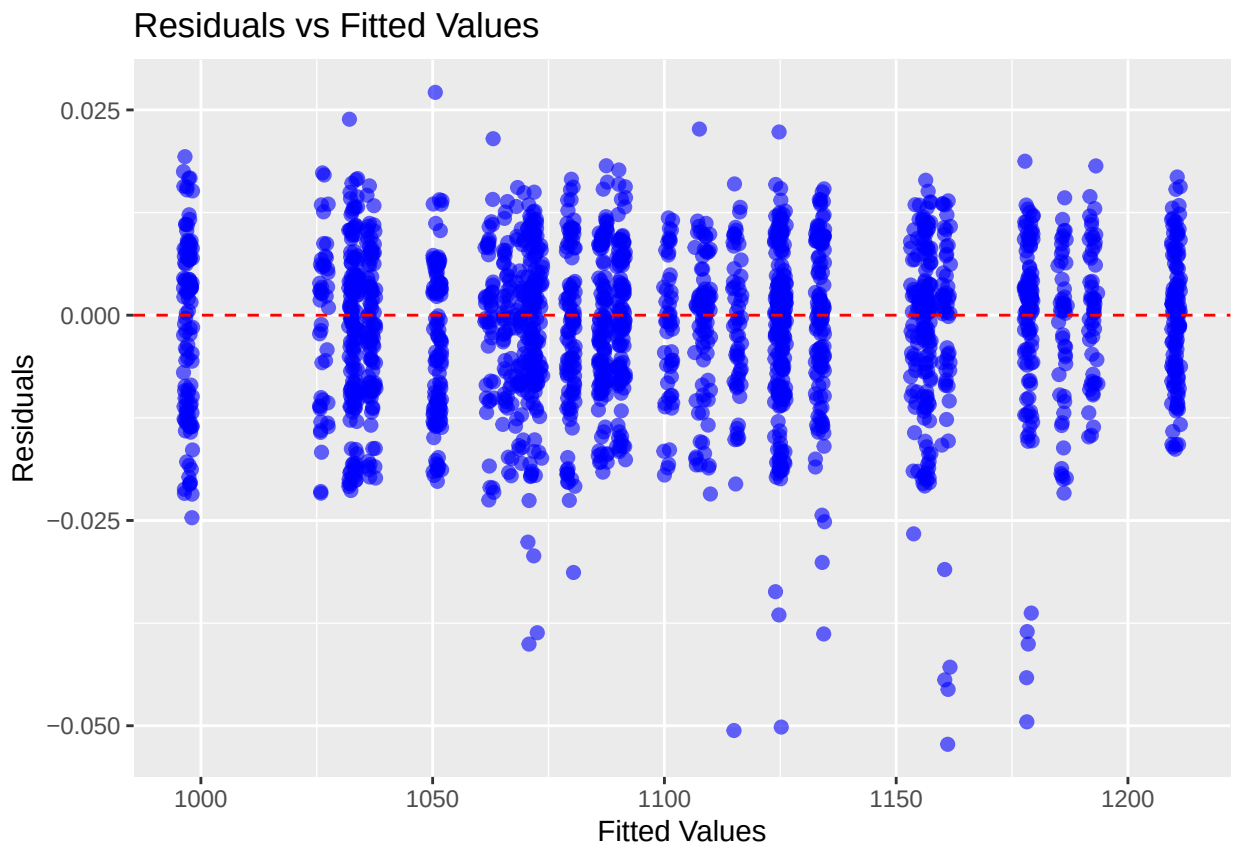
```
require(sjPlot)
require(sjstats)
# Predicted values for model A
predicted_F <- predict(modelF, type = "response")

# Predicted values for model B
predicted_G <- predict(modelG, type = "response")

# Create a data frame with the original data and predicted values
plot_data_F <- data.frame(TRT = tidy_PS_df$TRT, Predicted_F = predicted_F)
plot_data_G <- data.frame(TRT = tidy_PS_df$TRT, Predicted_G = predicted_G)

# Rename columns in plot_data_G to match plot_data_F
colnames(plot_data_G) <- colnames(plot_data_F)

# Combine predicted values with original data
plot_data_combined2 <- rbind(
  cbind(plot_data_F, Model = "Model F"),
  cbind(plot_data_G, Model = "Model G")
)

# Plot observed SRT scores with predicted SRT scores as separate violins
ggplot(plot_data_combined2, aes(x = Model, y = TRT, fill = Model)) +
  geom_violin(trim = FALSE, width = 0.8) +
```

```
  geom_violin(data = plot_data_combined2, aes(x = Model, y = Predicted_F), fill = "green", trim = FALSE
  labs(title = "Predicted vs Observed SRT Scores",
       x = NULL, y = "TRT") +
  theme_minimal()
```



#Residual vs fitted for model F:

```
library(ggplot2)
library(lme4)

# Extract fitted values and residuals from the model
fitted_values <- fitted(modelF)
residuals <- resid(modelF)

# Create a data frame with fitted values and residuals
residuals_df <- data.frame(Fitted_Values = fitted_values, Residuals = residuals)

# Create a scatter plot with ggplot2
ggplot(residuals_df, aes(x = Fitted_Values, y = Residuals)) +
  geom_point(position = position_jitter(width = 1), color = "blue", size = 2, alpha = 0.6) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(x = "Fitted Values", y = "Residuals",
       title = "Residuals vs Fitted Values")
```



#Investigate conditional counts after all cleaning complete. Imbalance expected because 3 types of trials, so "different" more common than "same."

```
library(dplyr)

# Count occurrences of Trial_Type in the STAT dataframe
STAT_counts <- STAT %>% count(Trial_Type)

# Count occurrences of Trial_Type in the DYN dataframe
DYN_counts <- DYN %>% count(Trial_Type)

# View the counts
print(STAT_counts)
```

```
##   Trial_Type   n
## 1  different 805
## 2       same 357
```

```
print(DYN_counts)
```

```
##   Trial_Type   n
## 1  different 761
## 2       same 334
```

```r
allcounts <- tidy_PS_df %>% count(Trial_Type)
print(allcounts)
```

```
##   Trial_Type    n
## 1       same  691
## 2  different 1566
```

#Reminder summaries of best fitting models:

```r
summary(modelB)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: inverse.gaussian  ( identity )
## Formula: SRT ~ Trial_Type_LVL + POV_LVL + (1 | RECORDING_SESSION_LABEL)
##    Data: tidy_PS_df
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##  23491.7  23520.3 -11740.8  23481.7     2252
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.6114 -0.6722 -0.1952  0.5397  6.5614
##
## Random effects:
##  Groups                   Name        Variance  Std.Dev.
##  RECORDING_SESSION_LABEL (Intercept) 1.006e+02 10.03157
##  Residual                             2.105e-04  0.01451
## Number of obs: 2257, groups:  RECORDING_SESSION_LABEL, 7
##
## Fixed effects:
##                          Estimate Std. Error t value Pr(>|z|)
## (Intercept)               218.030     14.032  15.538   <2e-16 ***
## Trial_Type_LVLdifferent   -1.217      1.972  -0.617    0.537
## POV_LVLdynamic            26.861      1.868  14.378   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) T_T_LV
## Trl_Typ_LVL -0.083
## POV_LVLdynm -0.042 -0.012
```

```r
summary(modelF)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: inverse.gaussian  ( identity )
## Formula: TRT ~ Trial_Type_LVL * POV_LVL + (1 | RECORDING_SESSION_LABEL)
##    Data: tidy_PS_df
```

```
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##  32654.3  32688.7 -16321.2  32642.3     2251
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.6341 -0.7872 -0.0150  0.6367  4.6648
##
## Random effects:
##  Groups                   Name        Variance  Std.Dev.
##  RECORDING_SESSION_LABEL (Intercept) 1.188e+03 34.461931
##  Residual                             7.949e-05  0.008916
## Number of obs: 2257, groups:  RECORDING_SESSION_LABEL, 7
##
## Fixed effects:
##                                       Estimate Std. Error t value Pr(>|z|)
## (Intercept)                           1112.182     30.448  36.527   <2e-16 ***
## Trial_Type_LVLdifferent                 17.863     17.828   1.002   0.3164
## POV_LVLdynamic                          -6.622     19.476  -0.340   0.7338
## Trial_Type_LVLdifferent:POV_LVLdynamic  -47.195     22.100  -2.136   0.0327 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) Tr_T_LVL POV_LV
## Trl_Typ_LVL -0.256
## POV_LVLdynm -0.210  0.314
## T_T_LVL:POV  0.147 -0.529   -0.683
```

#Post hoc testing for TRT:

```
STAT$Trial_Type <- factor(STAT$Trial_Type, levels = c("same", "different"))
STAT$POV <- factor(STAT$POV, levels = c("static", "dynamic"))
DYN$Trial_Type <- factor(DYN$Trial_Type, levels = c("same", "different"))
DYN$POV <- factor(DYN$POV, levels = c("static", "dynamic"))

STAT$POV_LVL <- relevel(STAT$POV, "static")
STAT$Trial_Type_LVL <- relevel(STAT$Trial_Type, "same")
DYN$POV_LVL <- relevel(DYN$POV, "static")
DYN$Trial_Type_LVL <- relevel(DYN$Trial_Type, "same")
```

```
require(lme4)

modelF1 <- glmer(TRT ~ Trial_Type_LVL + (1 | RECORDING_SESSION_LABEL),
                 data = STAT,
                 family = inverse.gaussian(link = "log"),
                 start = list(theta = 5.80, fixef = c(ST_SAME = 7.01, ST_DIFF = 7.01)),
                 control = glmerControl(optimizer = "bobyqa"))

summary(modelF1)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
```

```
##   Approximation) [glmerMod]
##  Family: inverse.gaussian  ( log )
## Formula: TRT ~ Trial_Type_LVL + (1 | RECORDING_SESSION_LABEL)
##    Data: STAT
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##  16865.0  16885.2  -8428.5  16857.0     1158
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.6788 -0.7797 -0.0143  0.6406  4.9420
##
## Random effects:
##  Groups                   Name        Variance  Std.Dev.
##  RECORDING_SESSION_LABEL (Intercept) 1.529e-03 0.039106
##  Residual                             7.653e-05 0.008748
## Number of obs: 1162, groups:  RECORDING_SESSION_LABEL, 7
##
## Fixed effects:
##                          Estimate Std. Error t value Pr(>|z|)
## (Intercept)               7.01012    0.04047 173.203   <2e-16 ***
## Trial_Type_LVLdifferent  0.01579    0.02065   0.765    0.445
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## Trl_Typ_LVL -0.352
```

```
modelF2 <- glmer(TRT ~ Trial_Type_LVL + (1 | RECORDING_SESSION_LABEL),
                 data = DYN,
                 family = inverse.gaussian(link = "log"),
                 start = list(theta = 5.80, fixef = c(DY_SAME = 7.01, DY_DIFF = 7.01)),
                 control = glmerControl(optimizer = "bobyqa"))

summary(modelF2)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: inverse.gaussian  ( log )
## Formula: TRT ~ Trial_Type_LVL + (1 | RECORDING_SESSION_LABEL)
##    Data: DYN
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##  15799.0  15819.0  -7895.5  15791.0     1091
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.5731 -0.8096  0.0089  0.6206  3.6979
##
## Random effects:
##  Groups                   Name        Variance  Std.Dev.
```

```
##  RECORDING_SESSION_LABEL (Intercept) 9.536e-04 0.03088
##  Residual                             8.298e-05 0.00911
## Number of obs: 1095, groups:  RECORDING_SESSION_LABEL, 7
##
## Fixed effects:
##                          Estimate Std. Error t value Pr(>|z|)
## (Intercept)               6.99950    0.03018 231.905   <2e-16 ***
## Trial_Type_LVLdifferent -0.02554    0.02144  -1.191    0.234
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## Trl_Typ_LVL -0.496
```

```
#Model F1 failed to converge at the max with an identity link but converged
#with a log link. Seems that significance at the TRT intercept level was a
#type 1 error, likely induced by the differences at the POV level. There are no
#significant differences within condition at the trial type level, indicating
#no large discrepancies of reaction time.
```

#Create AIC tables ready for transfer to write-up:

```
as.data.frame(AIC_weights)
```

```
##   Modnames K     AICc Delta_AICc     ModelLik       AICcWt        LL    Cum.Wt
## 4  Model D 4 23490.08   0.000000 1.000000e+00 6.022280e-01 -11741.03 0.6022280
## 2  Model B 5 23491.71   1.628847 4.428947e-01 2.667236e-01 -11740.84 0.8689516
## 1  Model A 6 23493.13   3.050139 2.176059e-01 1.310484e-01 -11740.54 1.0000000
## 5  Model E 3 23689.56 199.485577 4.811243e-44 2.897465e-44 -11841.78 1.0000000
## 3  Model C 4 23691.41 201.330792 1.912379e-44 1.151688e-44 -11841.69 1.0000000
```

```
as.data.frame(AIC_weights_TRT)
```

```
##   Modnames K     AICc Delta_AICc   ModelLik     AICcWt        LL    Cum.Wt
## 4  Model I 4 32652.61   0.000000 1.00000000 0.51783163 -16322.30 0.5178316
## 1  Model F 6 32654.38   1.766890 0.41335636 0.21404900 -16321.17 0.7318806
## 2  Model G 5 32654.46   1.855442 0.39545383 0.20477850 -16322.22 0.9366591
## 5  Model J 3 32657.46   4.854349 0.08828593 0.04571725 -16325.73 0.9823764
## 3  Model H 4 32659.37   6.760819 0.03403351 0.01762363 -16325.68 1.0000000
```

```
# For AIC weights table
# Rename columns
names(AIC_weights) <- c("Model Name", "Degrees of Freedom", "AICc", "Delta AICc", "Model Likelihood", "

# For AIC weights TRT table
# Rename columns
names(AIC_weights_TRT) <- c("Model Name", "Number of Parameters", "AICc", "Delta AICc", "Model Likeliho

library(dplyr)

# Assuming your dataframes are named AIC_weights and AIC_weights_TRT
```

```r
AIC_SRT <- AIC_weights %>%
  mutate_if(is.numeric, ~round(., digits = 2))

AIC_TRT <- AIC_weights_TRT %>%
  mutate_if(is.numeric, ~round(., digits = 2))

# Print the rounded dataframes
print(AIC_SRT)
```

```
##
## Model selection based on AICc:
##
##         Degrees of Freedom     AICc Delta AICc AICc Weight Log Likelihood
## Model D                  4 23490.08       0.00        0.60      -11741.03
## Model B                  5 23491.71       1.63        0.27      -11740.84
## Model A                  6 23493.13       3.05        0.13      -11740.54
## Model E                  3 23689.56     199.49        0.00      -11841.78
## Model C                  4 23691.41     201.33        0.00      -11841.69
```

```r
print(AIC_TRT)
```

```
##
## Model selection based on AICc:
##
##         Number of Parameters     AICc Delta AICc AICc Weight Log Likelihood
## Model I                    4 32652.61       0.00        0.52      -16322.30
## Model F                    6 32654.38       1.77        0.21      -16321.17
## Model G                    5 32654.46       1.86        0.20      -16322.22
## Model J                    3 32657.46       4.85        0.05      -16325.73
## Model H                    4 32659.37       6.76        0.02      -16325.68
```

```r
write.csv(AIC_SRT, "AIC_WEIGHTS_SRT.csv", row.names = FALSE)
write.csv(AIC_TRT, "AIC_WEIGHTS_TRT.csv", row.names = FALSE)
```

#Calculating SOA

```r
require(dplyr)

tidy_PS_df <- tidy_PS_df %>%
  group_by(RECORDING_SESSION_LABEL) %>%
  mutate(PREVIOUS_IP_START = lag(IP_START_TIME))

tidy_PS_df <- tidy_PS_df %>%
  group_by(RECORDING_SESSION_LABEL) %>%
  mutate(SOA = IP_START_TIME - PREVIOUS_IP_START)

STAT2 <- tidy_PS_df %>% filter(POV != "dynamic")
DYN2 <- tidy_PS_df %>% filter(POV != "static")
```

#Calculating SOA for each trial and preparing the data for plotting:

```r
# Load the necessary libraries
require(dplyr)
require(ggplot2)

# Convert SOA to seconds
tidy_PS_df <- tidy_PS_df %>%
  mutate(SOA = SOA / 1000)

# Define the threshold for unreasonable SOA values
upper_threshold <- 30

# Split the data into each condition and remove negative and unreasonable SOA values
STAT2 <- tidy_PS_df %>% filter(POV != "dynamic" & SOA >= 0 & SOA <= upper_threshold)
DYN2 <- tidy_PS_df %>% filter(POV != "static" & SOA >= 0 & SOA <= upper_threshold)

# Add a trial number for each condition
STAT2 <- STAT2 %>%
  group_by(RECORDING_SESSION_LABEL) %>%
  mutate(Trial = row_number())

DYN2 <- DYN2 %>%
  group_by(RECORDING_SESSION_LABEL) %>%
  mutate(Trial = row_number())

# Combine the data for plotting
combined_data <- bind_rows(STAT2, DYN2) %>%
  mutate(Condition = ifelse(POV == "dynamic", "Dynamic", "Static"))

# Function to identify outliers
identify_outliers <- function(data) {
  Q1 <- quantile(data$SOA, 0.25)
  Q3 <- quantile(data$SOA, 0.75)
  IQR <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR
  upper_bound <- Q3 + 1.5 * IQR
  data %>%
    mutate(Outlier = ifelse(SOA < lower_bound | SOA > upper_bound, "Outlier", "Non-Outlier"))
}

# Identify outliers in the combined data
combined_data <- identify_outliers(combined_data)

# Define shapes for conditions
condition_shapes <- c('Static' = 16, 'Dynamic' = 17)
```

#Plot SOA throughout the experiment:

```r
require(ggplot2)

# Plot the data with different shapes for conditions and colors for participants
p <- ggplot(combined_data, aes(x = Trial, y = SOA, shape = Condition, color = RECORDING_SESSION_LABEL)) +
  geom_jitter(width = 0.4, height = 0.2) +  # Add jitter with horizontal spread
  scale_shape_manual(values = condition_shapes) +
```

```
  scale_color_brewer(palette = "Set1", name = "Participant") +
  scale_fill_manual(values = c('Non-Outlier' = 'black', 'Outlier' = NA)) +  # No fill for outliers
  scale_alpha_manual(values = c('Non-Outlier' = 0.9, 'Outlier' = 0.8)) +
  labs(title = "Stimulus Onset Asynchrony (SOA) by Trial",
       x = "Trial Progression (left to right)",
       y = "SOA (seconds)") +
  theme_minimal() +
  theme(axis.text.x = element_text(color = "black"), # Set x-axis label color
        legend.position = "right",
        legend.title.align = 1,  # Center the legend title
        axis.title.x = element_text(margin = margin(t = 20)),  # Adjust x-axis title margin
        axis.title.y = element_text(margin = margin(r = 20))) +  # Adjust y-axis title margin
  scale_x_continuous(limits = c(0, 150), breaks = seq(0, 150, by = 150))  # Set x-axis limits and breaks
```

```
## Warning: The ‘legend.title.align‘ argument of ‘theme()‘ is deprecated as of ggplot2
## 3.5.0.
## i Please use theme(legend.title = element_text(hjust)) instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
# Save the plot as a PNG
ggsave("SOA_plot.png", plot = p, width = 8, height = 6, units = "in")
```

```
## Warning: No shared levels found between ‘names(values)‘ of the manual scale and the
## data’s fill values.
```

```
## Warning: No shared levels found between ‘names(values)‘ of the manual scale and the
## data’s alpha values.
```

```
## Warning: Removed 130 rows containing missing values or values outside the scale range
## (‘geom_point()‘).
```