MusicMapping: Online Experiments in Auditory Motor-Mapping

Instructions:

1. Enter path to reference sounds (approximately line 30)
2. Enter path to data (approximately line 81)
3. Click 'Run All' in the upper right hand corner of R

```
rm(list = ls())
library(jsonlite)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
## v readr     2.1.5
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter()  masks stats::filter()
## x purrr::flatten() masks jsonlite::flatten()
## x dplyr::lag()     masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

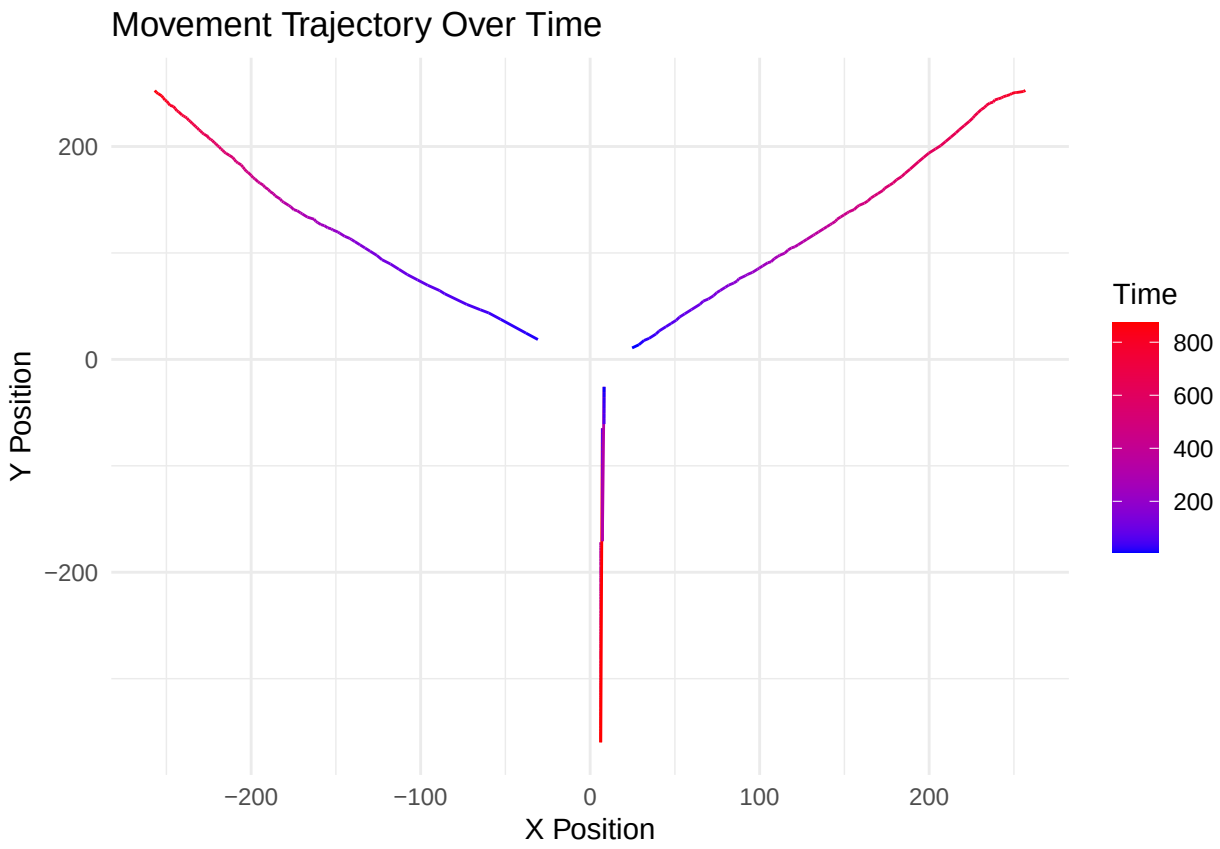
Load in reference data

```
start_x <- 833.5   # example value
start_y <- 539.5   # example value

######
#1 ENTER PATH TO REFERENCE SOUNDS
reference_data <- read.csv("/Users/katie/Documents/workspace/Kinetic-Muse/Data/KatieSoundReference.csv")

reference_data <- reference_data %>%
  mutate(Hand_Path_JSON = gsub("'", "\"", Hand_Path))

# Parse the JSON
reference_data <- reference_data %>%
  rowwise() %>%
```

```
  mutate(positions_list = list(fromJSON(Hand_Path_JSON)$positions)) %>%
  ungroup()

reference_data_long <- reference_data %>%
  unnest(cols = positions_list) %>%
  mutate(x = x - start_x,
         y = start_y - y)


# Plot the trajectories colored by Target.Angle as before
p1 <- ggplot(reference_data_long, aes(x = x, y = y, group = Trial.Number, color = factor(Target.Angle))
  geom_path() +
  labs(title = "Movement Trajectories by Trial",
       x = "X Position",
       y = "Y Position",
       color = "Target Angle") +
  geom_point(aes(x = 0, y = 0), color = "black", size = 4) +
  theme_minimal()

# Print the trajectory plot
print(p1)
```


```
reference_data_long %>%
  filter(Trial.Number %in% c(12, 19, 20)) %>%
  ggplot(aes(x = x, y = y, color = time, group = Target.Angle)) +
  geom_line() +
  scale_color_gradient(low = "blue", high = "red") +
```

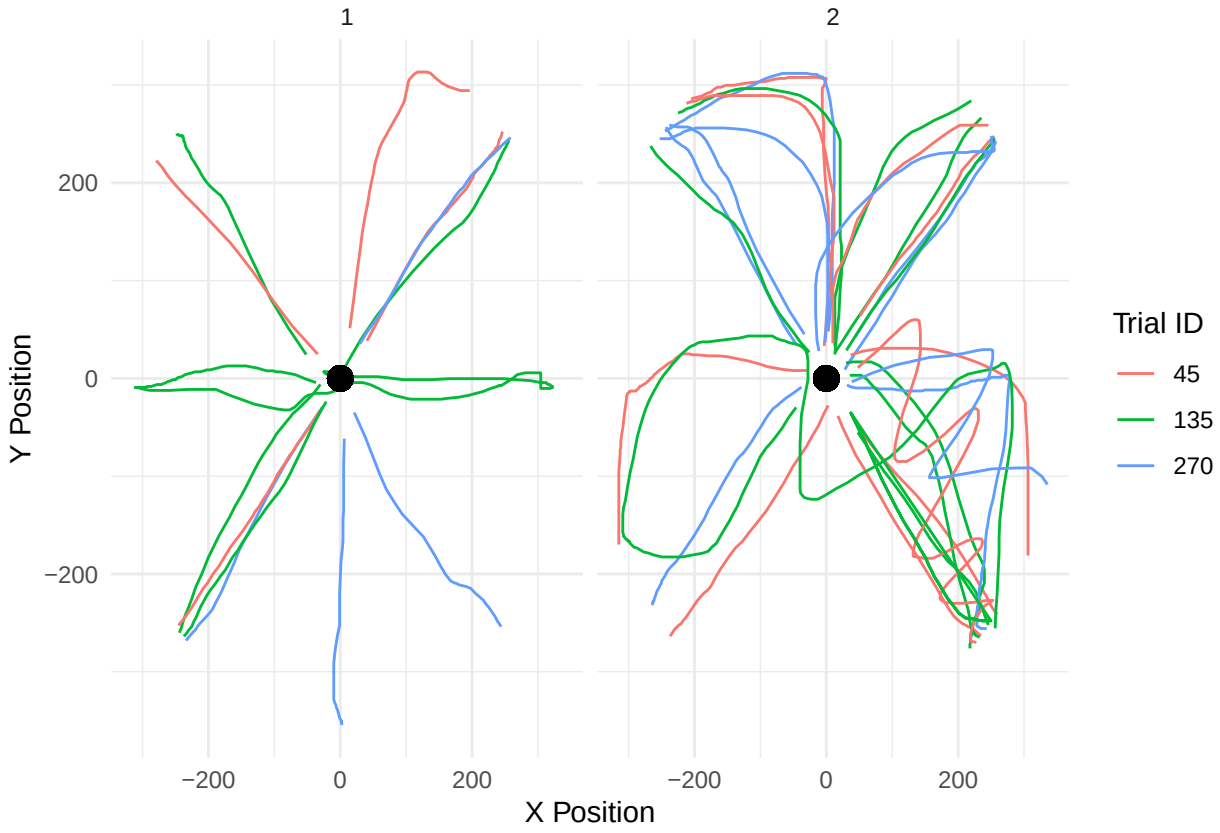
```
  labs(title = "Movement Trajectory Over Time",
       x = "X Position",
       y = "Y Position",
       color = "Time") +
  theme_minimal()
```



Load in participant data

```
######
#2 ENTER PATH TO DATA YOU WANT TO ANALYZE
my_data <- read.csv("/Users/katie/Documents/workspace/Kinetic-Muse/Data/today2.csv")

my_data <- my_data %>%
  mutate(Hand_Path_JSON = gsub("'", "\"", Hand_Path))

# Parse the JSON
my_data <- my_data %>%
  rowwise() %>%
  mutate(positions_list = list(fromJSON(Hand_Path_JSON)$positions)) %>%
  ungroup()

my_data_long <- my_data %>%
  unnest(cols = positions_list) %>%
  mutate(x = x - Start_X,
         y = Start_Y - y,
         Phase = ifelse(Trial.Number <= 12, 1, 2))
```

```r
ggplot(my_data_long , aes(x = x, y = y, group = Trial.Number, color = factor(Target.Angle))) +
  geom_path() +
  labs(x = "X Position",
       y = "Y Position",
       color = "Trial ID") +
  geom_point(aes(x = 0, y = 0), color = "black", size = 4) +
  theme_minimal() +
  facet_wrap(.~Phase)
```



Calculate error of each trial relative to reference

```r
# Define old and new screen parameters (make sure these are set as needed)
old_screen_width <- 1667
old_screen_height <- 1079
old_start_x <- 833.5
old_start_y <- 539.5

new_screen_width <- unique(my_data$Screen_Width)
new_screen_height <- unique(my_data$Screen_Height)
new_start_x <- unique(my_data$Start_X)
new_start_y <- unique(my_data$Start_Y)

# Initialize a results data frame to store the output
results <- data.frame(
  Trial.Number = integer(),
  Target.Angle = numeric(),
  MeanError = numeric(),
  stringsAsFactors = FALSE
```

```
)

# Get all unique trial numbers from my_data
all_trials <- unique(my_data$Trial.Number)

for (ti in all_trials) {
  # Extract the target angle for this trial
  this_angle <- unique(my_data$Target.Angle[my_data$Trial.Number == ti])

  # Select the appropriate reference trajectory based on the angle
  if (this_angle == 45) {
    ref_trajectory <- reference_data_long[reference_data_long$Trial.Number == 19, ]
  } else if (this_angle == 135) {
    ref_trajectory <- reference_data_long[reference_data_long$Trial.Number == 20, ]
  } else if (this_angle == 270) {
    ref_trajectory <- reference_data_long[reference_data_long$Trial.Number == 12, ]
  } else {
    # If there's an angle not covered by these conditions, skip this trial
    next
  }

  # Scale the reference trajectory to the new monitor dimensions and start position
  ref_trajectory_scaled <- ref_trajectory
  ref_trajectory_scaled$x <- ((ref_trajectory_scaled$x - old_start_x) / old_screen_width) * new_screen_
  ref_trajectory_scaled$y <- ((ref_trajectory_scaled$y - old_start_y) / old_screen_height) * new_screen_

  # Define common time points from the reference
  common_times <- ref_trajectory$time

  # Interpolate new trajectory to reference time points
  new_x_interp <- approx(
    x = my_data_long$time[my_data_long$Trial.Number == ti],
    y = my_data_long$x[my_data_long$Trial.Number == ti],
    xout = common_times
  )$y

  new_y_interp <- approx(
    x = my_data_long$time[my_data_long$Trial.Number == ti],
    y = my_data_long$y[my_data_long$Trial.Number == ti],
    xout = common_times
  )$y

  # Compute error (Euclidean distance) at each time point using the scaled reference trajectory
  error_values <- sqrt((ref_trajectory_scaled$x - new_x_interp)^2 + (ref_trajectory_scaled$y - new_y_int

  # Compute a summary metric, e.g., mean error
  mean_error <- mean(error_values, na.rm = TRUE)

  # Append results for this trial
  results <- rbind(
    results,
    data.frame(
      Trial.Number = ti,
```

```
      Target.Angle = this_angle,
      MeanError = mean_error,
      stringsAsFactors = FALSE
    )
  )
}
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```

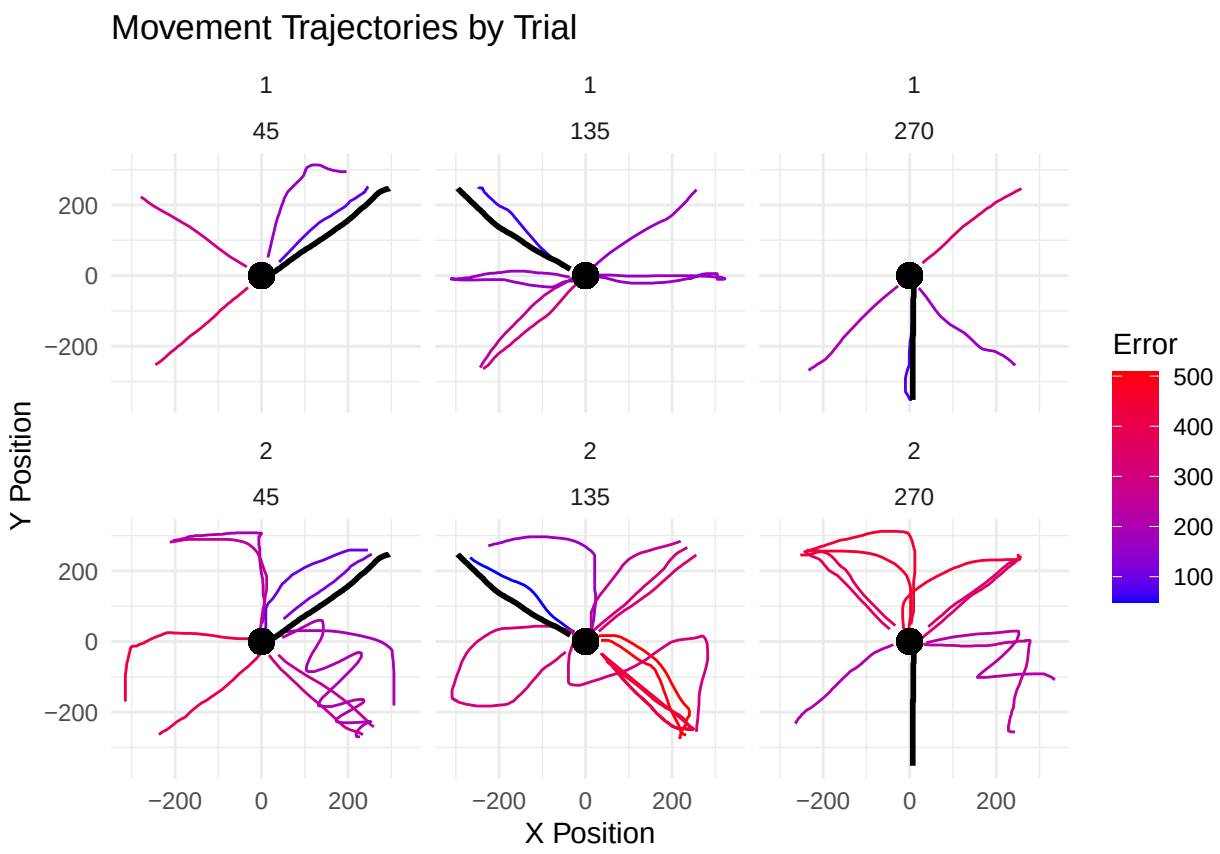
```
# Print the resulting data frame
print(results)
```

```
##    Trial.Number Target.Angle MeanError
## 1             1           45  87.59654
## 2             2          270  75.76205
## 3             3          135  66.04564
## 4             4           45 160.71278
## 5             5          270 203.37198
## 6             6          135 243.55878
## 7             7           45 354.21668
## 8             8          270 161.72977
## 9             9          135 162.15890
## 10           10           45 281.47404
## 11           11          270 333.76032
## 12           12          135 296.25437
## 13           13           45 271.85779
## 14           14          270 372.09990
## 15           15          135 313.12350
## 16           16           45 388.48477
## 17           17          270 373.53002
## 18           18          135  49.95770
## 19           19           45 122.89654
## 20           20          270 342.62099
## 21           21          135 450.55476
## 22           22           45 252.13409
## 23           23          270 348.02688
## 24           24          135 334.77180
## 25           25           45 200.77769
## 26           26          270 445.13993
## 27           27          135 164.44099
## 28           28           45 250.37938
## 29           29          270 452.59831
## 30           30          135 251.56261
## 31           31           45  98.34731
## 32           32          270 442.72572
## 33           33          135 507.92346
## 34           34           45 184.17981
## 35           35          270 257.47961
## 36           36          135 500.45120
## 37           37           45 415.85209
## 38           38          270 203.77418
## 39           39          135 308.22570
```

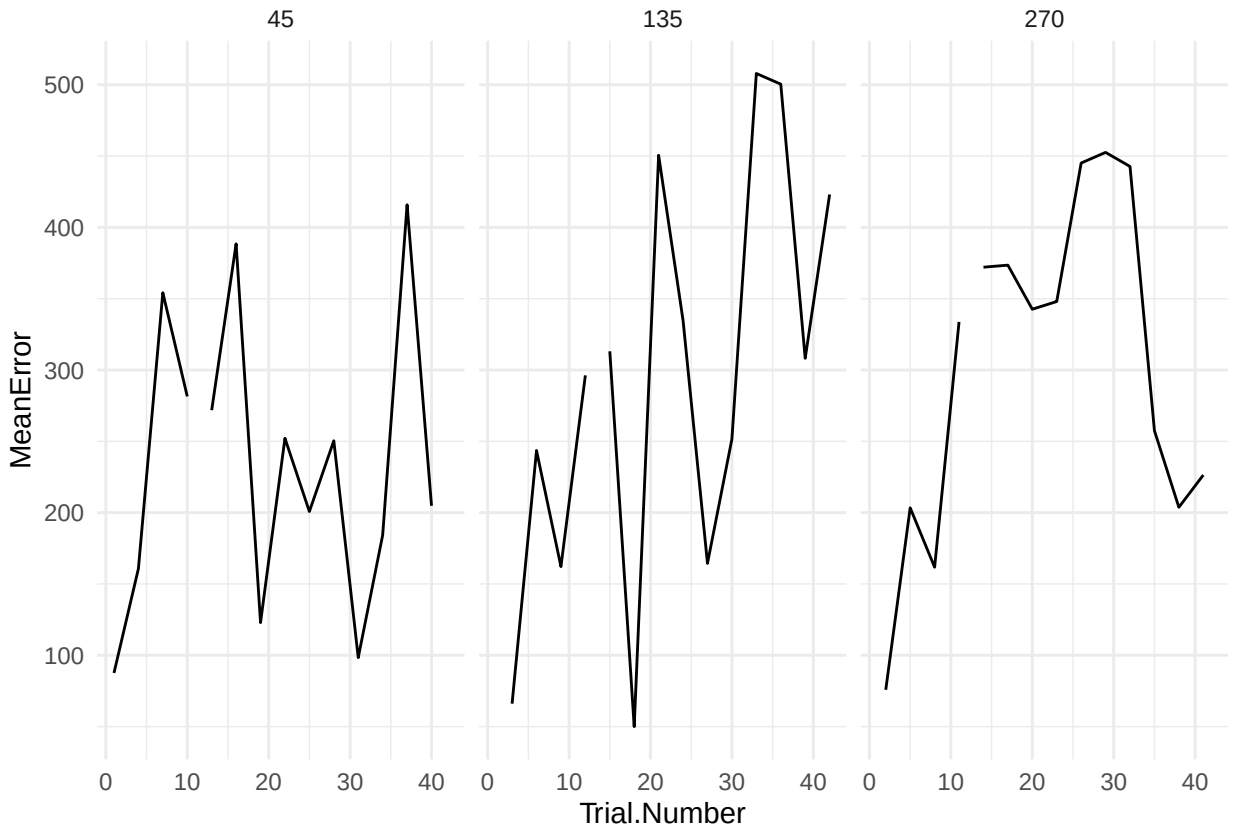
```
## 40           40           45 204.80623
## 41           41          270 226.38239
## 42           42          135 423.06510
```

```r
my_data_long_mod <- full_join(my_data_long, results, by = c("Trial.Number", "Target.Angle"))

# Filter out the reference trajectories
reference_data_trunc <- reference_data_long %>%
  filter(Trial.Number %in% c(12, 19, 20))

# Create a rescaled version of reference_data_trunc
reference_data_trunc_scaled <- reference_data_trunc %>%
  mutate(
    x = ((x - old_start_x) / old_screen_width) * new_screen_width + new_start_x,
    y = ((y - old_start_y) / old_screen_height) * new_screen_height + new_start_y
  )

# Plot using the rescaled reference trajectories
ggplot(my_data_long_mod, aes(x = x, y = y, group = Trial.Number, color = MeanError)) +
  geom_path() +
  labs(title = "Movement Trajectories by Trial",
       x = "X Position",
       y = "Y Position",
       color = "Error") +
  geom_point(aes(x = 0, y = 0), color = "black", size = 4) +
  theme_minimal() +
  facet_wrap(Phase~Target.Angle) +
  scale_color_gradient(low = "blue", high = "red") +
  geom_path(data = reference_data_trunc_scaled,
            aes(x = x, y = y, group = Trial.Number),
            inherit.aes = FALSE,
            color = "black",
            linetype = "solid",
            size = 1)
```

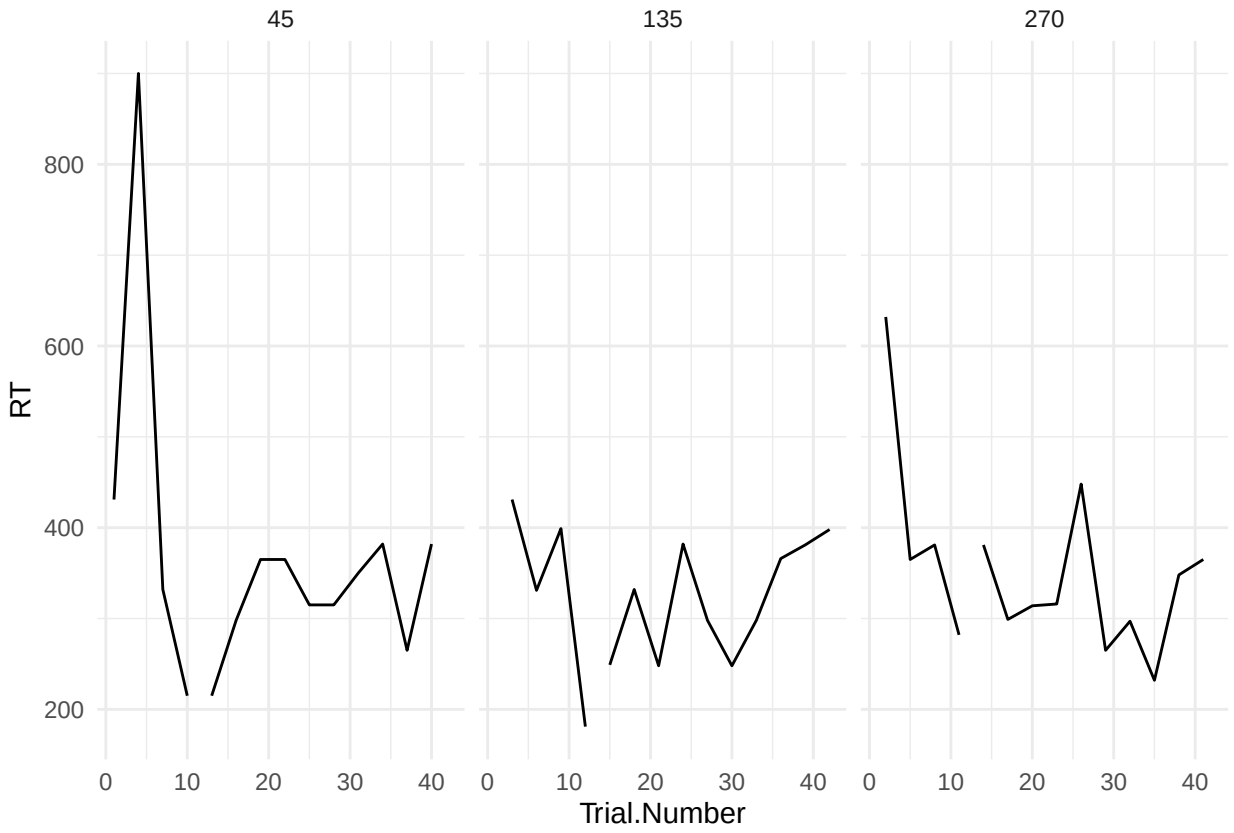
```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



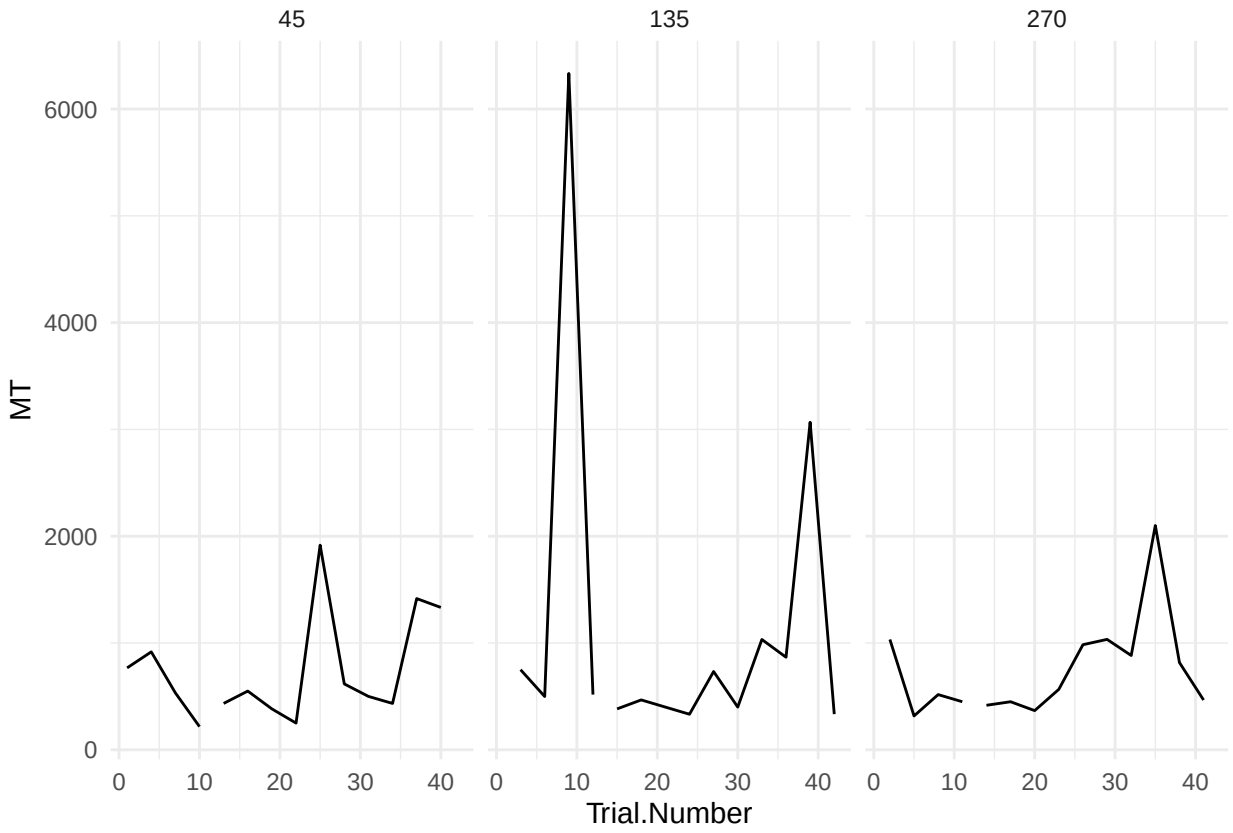

```
my_data_long_mod %>%
  distinct(Trial.Number, Target.Angle, MeanError, Phase) %>%
  ggplot(aes(x = Trial.Number, y = MeanError, group = Phase)) +
  geom_line() + theme_minimal() + facet_wrap(.~Target.Angle)
```




```
my_data_long_mod %>%
  distinct(Trial.Number, Target.Angle, RT, Phase) %>%
  ggplot(aes(x = Trial.Number, y = RT, group = Phase)) +
  geom_line() + theme_minimal() + facet_wrap(.~Target.Angle)
```




```
my_data_long_mod %>%
  distinct(Trial.Number, Target.Angle, MT, Phase) %>%
  ggplot(aes(x = Trial.Number, y = MT, group = Phase)) +
  geom_line() + theme_minimal() + facet_wrap(.~Target.Angle)
```


45
135
270
6000
4000
2000
0
MT
0
10
20
30
40
Trial.Number