Brendon_EDA

Brendon Stanley

2024-05-20

Exec Summary

- Centroid of "types" (triangles, big cirlces, and red circles) appear to relate to ability to predict correct centroid quadrant.
- More "types" in the correct quadrant helps to predict correct quadrant.
- Decent amount of per-person variability. Mixed model would be good.
- Appears on average, as `howManyCorr` increases so does odds of correct prediction
- The further the true centroid is from the center, the easier to predict the corect quadrent
- Visual inspection for longitudinal effect . . . inconclusive. Prob good to check though.

```
library(ggplot2)
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(zoo)
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
df = read.csv("/Users/brendo/repos/CSU/Visual-Inference/src/data/Wills_data_corrected.csv", header=TRUE

df$isCorrect = df$resp == df$corrQuad
df$subj = as.factor(df$subj)
df$currImg = as.factor(df$currImg)
df$resp = as.factor(df$resp)
df$corrQuad = as.factor(df$resp)
df$isColSame = as.logical(df$isColSame)
df$isPulseSame = as.logical(df$isPulseSame)
df$isSizeSame = as.logical(df$isSizeSame)
```

Check Total Correct

```
sum(df$isCorrect) / length(df$isCorrect)
```

```
## [1] 0.4095192
```

Check Total Correct when `isColSame` is TRUE

```
sum(df[df$isColSame == TRUE, ]$isCorrect) / length(df[df$isColSame == TRUE, ]$isCorrect)
```

```
## [1] 0.4492591
```

Check Total Correct when `isPulseSame` is TRUE

```
sum(df[df$isColSame == TRUE, ]$isPulseSame) / length(df[df$isColSame == TRUE, ]$isPulseSame)
```

```
## [1] 0.3879197
```

Check Total Correct when `isSizeSame` is TRUE

```
sum(df[df$isColSame == TRUE, ]$isSizeSame) / length(df[df$isColSame == TRUE, ]$isSizeSame)
```

```
## [1] 0.4033014
```

How many plots did each subject look at?

```
subjs = levels(df$subj)

for (i in 1:length(subjs)){
  subject_num = subjs[i]
  subject_df = df[df$subj == subject_num, ]
  print(paste(subject_num, nrow(subject_df)))
}
```

```
## [1] "1 200"
## [1] "2 200"
## [1] "3 200"
## [1] "4 200"
## [1] "5 200"
## [1] "6 200"
## [1] "7 200"
## [1] "8 200"
## [1] "9 200"
## [1] "10 200"
## [1] "11 200"
## [1] "12 200"
## [1] "13 200"
## [1] "14 200"
## [1] "15 200"
## [1] "16 200"
## [1] "17 200"
```

```
## [1] "18 200"
## [1] "19 200"
## [1] "20 200"
## [1] "21 200"
## [1] "22 200"
## [1] "23 200"
## [1] "24 200"
## [1] "25 200"
## [1] "26 200"
## [1] "27 200"
## [1] "28 200"
## [1] "29 200"
## [1] "30 200"
## [1] "31 200"
## [1] "32 200"
## [1] "33 200"
## [1] "34 200"
## [1] "35 200"
## [1] "36 200"
## [1] "37 200"
## [1] "38 200"
## [1] "39 200"
## [1] "40 200"
## [1] "41 200"
## [1] "42 200"
## [1] "43 200"
## [1] "44 200"
## [1] "45 200"
## [1] "46 200"
## [1] "47 200"
## [1] "48 200"
## [1] "49 200"
## [1] "50 200"
## [1] "51 200"
## [1] "52 200"
```

... I just can't do 52*200 in my head ... duch this is correct.

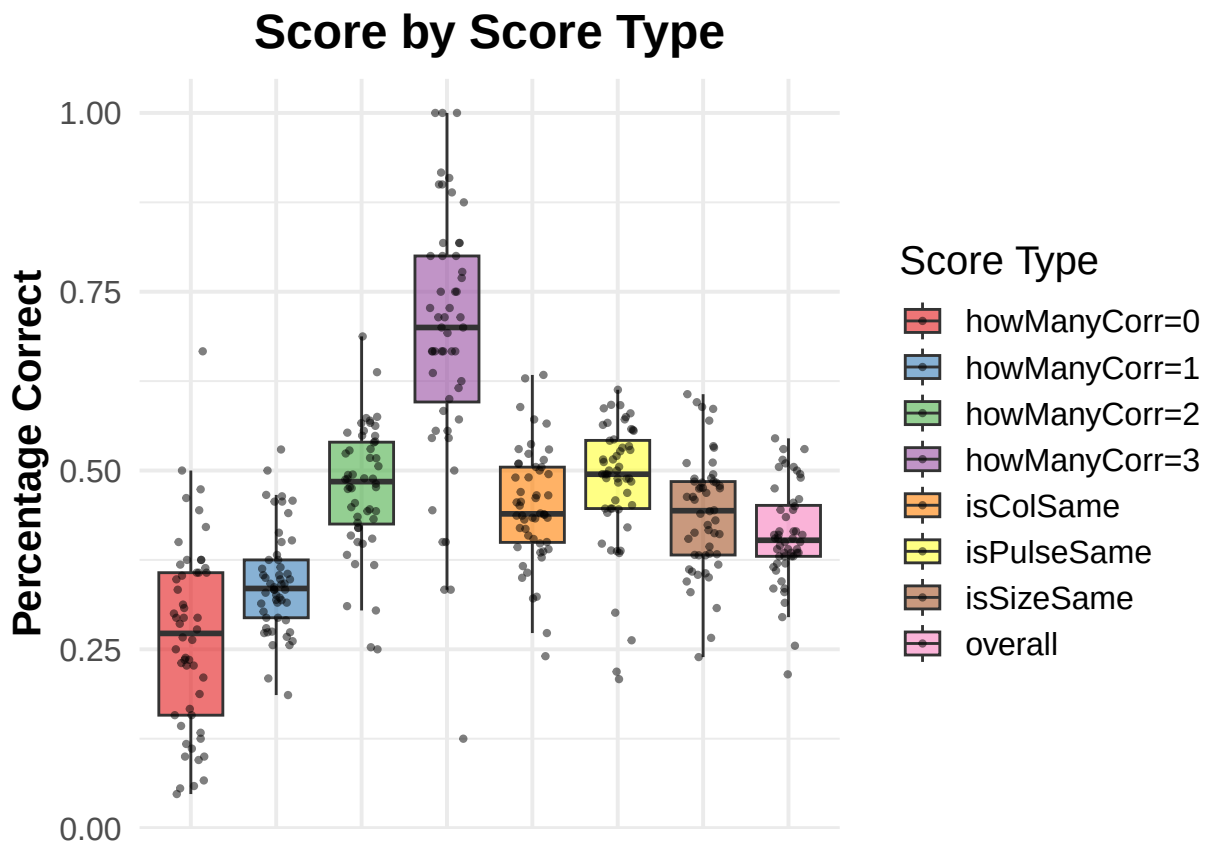
Now lets look at scores for each subject

```
subjs = levels(df$subj)

score_vector = c()

for (i in 1:length(subjs)){
  subject_num = subjs[i]
  subject_df = df[df$subj == subject_num, ]

  overall_score = sum(subject_df$isCorrect) / length(subject_df$isCorrect)

  score_is_col = sum(subject_df[subject_df$isColSame == TRUE, ]$isCorrect) / length(subject_df[subject_

  score_is_pulse = sum(subject_df[subject_df$isPulseSame == TRUE, ]$isCorrect) / length(subject_df[subj
```

```
  score_is_size = sum(subject_df[subject_df$isSizeSame == TRUE, ]$isCorrect) / length(subject_df[subjec

  score_zero_correct = sum(subject_df[subject_df$howManyCorr == 0, ]$isCorrect) / length(subject_df[sub

  score_one_correct = sum(subject_df[subject_df$howManyCorr == 1, ]$isCorrect) / length(subject_df[subj

  score_two_correct = sum(subject_df[subject_df$howManyCorr == 2, ]$isCorrect) / length(subject_df[subj

  score_three_correct = sum(subject_df[subject_df$howManyCorr == 3, ]$isCorrect) / length(subject_df[su

  score_vector = c(score_vector, overall_score, score_is_col, score_is_pulse, score_is_size, score_zero

}

score_df = data.frame(
  score = score_vector,
  score_type = rep(c("overall", "isColSame", "isSizeSame", "isPulseSame", "howManyCorr=0", "howManyCorr
  subject = rep(1:length(subjs), each = 8)
)

score_df$score_type = as.factor(score_df$score_type)
score_df$subject = as.factor(score_df$subject)
```

Score by Score Type

```
ggplot(score_df, aes(x = score_type, y = score, fill = score_type)) +
  geom_boxplot(outlier.shape = NA, alpha = 0.6) +
  geom_jitter(width = 0.2, size = 0.8, alpha = 0.5) +
  scale_fill_brewer(palette = "Set1") +
  labs(
    x = NULL,
    y = "Percentage Correct",
    title = "Score by Score Type",
    fill = "Score Type"
  ) +
  theme_minimal(base_size = 15) +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    axis.title.x = element_blank(),  # Remove x-axis title
    axis.text.x = element_blank(),  # Remove x-axis text
    axis.ticks.x = element_blank(),  # Remove x-axis ticks
    axis.title.y = element_text(face = "bold")
  )
```



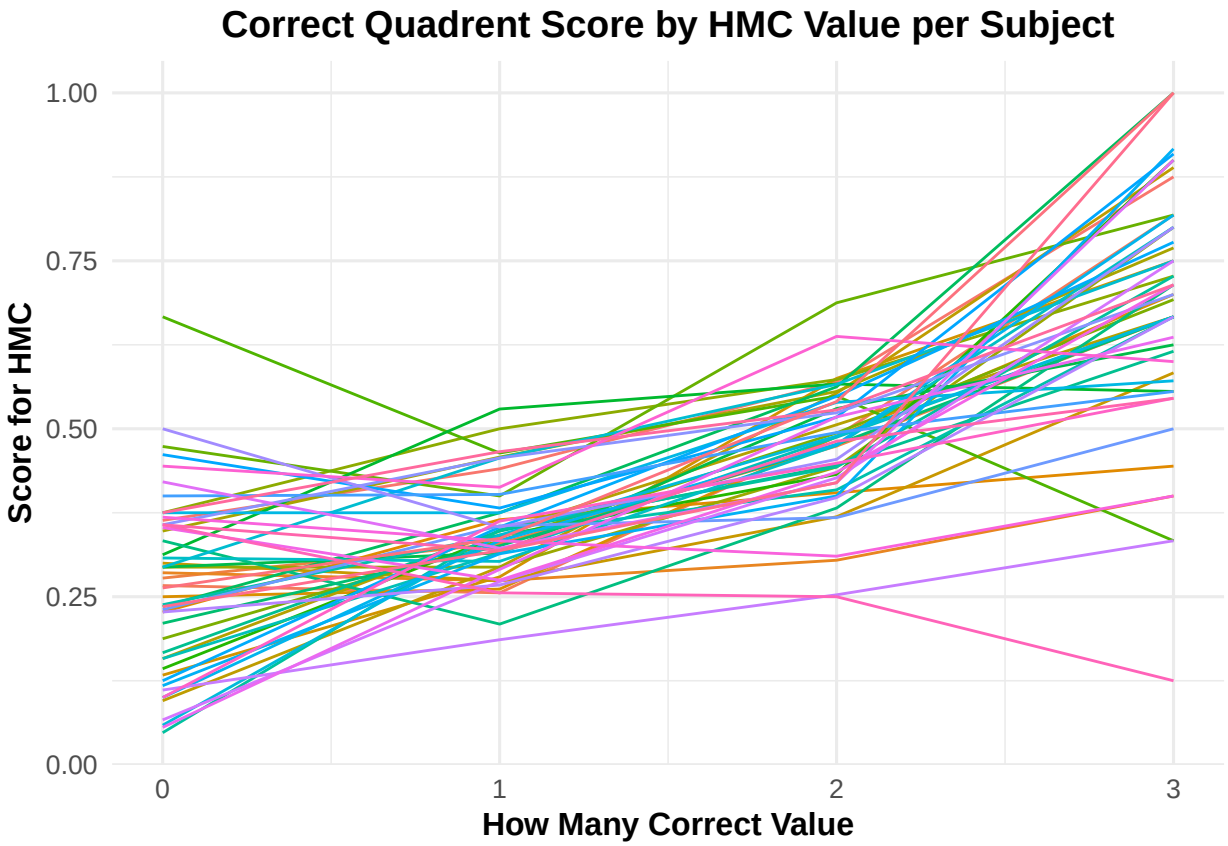
Per Person Effect of `howManycorr`

```
subjs = levels(df$subj)

store = c()

for (i in 1:length(subjs)){
  subject_num = subjs[i]
  subject_df = df[df$subj == subject_num, ]

  s_df_hmc0 = subject_df[subject_df$howManyCorr == 0, ]$isCorrect
  s_df_hmc1 = subject_df[subject_df$howManyCorr == 1, ]$isCorrect
  s_df_hmc2 = subject_df[subject_df$howManyCorr == 2, ]$isCorrect
  s_df_hmc3 = subject_df[subject_df$howManyCorr == 3, ]$isCorrect

  hmc0 = sum(s_df_hmc0) / length(s_df_hmc0)
  hmc1 = sum(s_df_hmc1) / length(s_df_hmc1)
  hmc2 = sum(s_df_hmc2) / length(s_df_hmc2)
  hmc3 = sum(s_df_hmc3) / length(s_df_hmc3)

  store = c(store, hmc0, hmc1, hmc2, hmc3)
}

hmc_df = data.frame(
```

```
  score = store,
  hmc = rep(0:3, length(subjs)),
  subject = rep(1:length(subjs), each = 4)
)

hmc_df$subject = as.factor(hmc_df$subject)
```

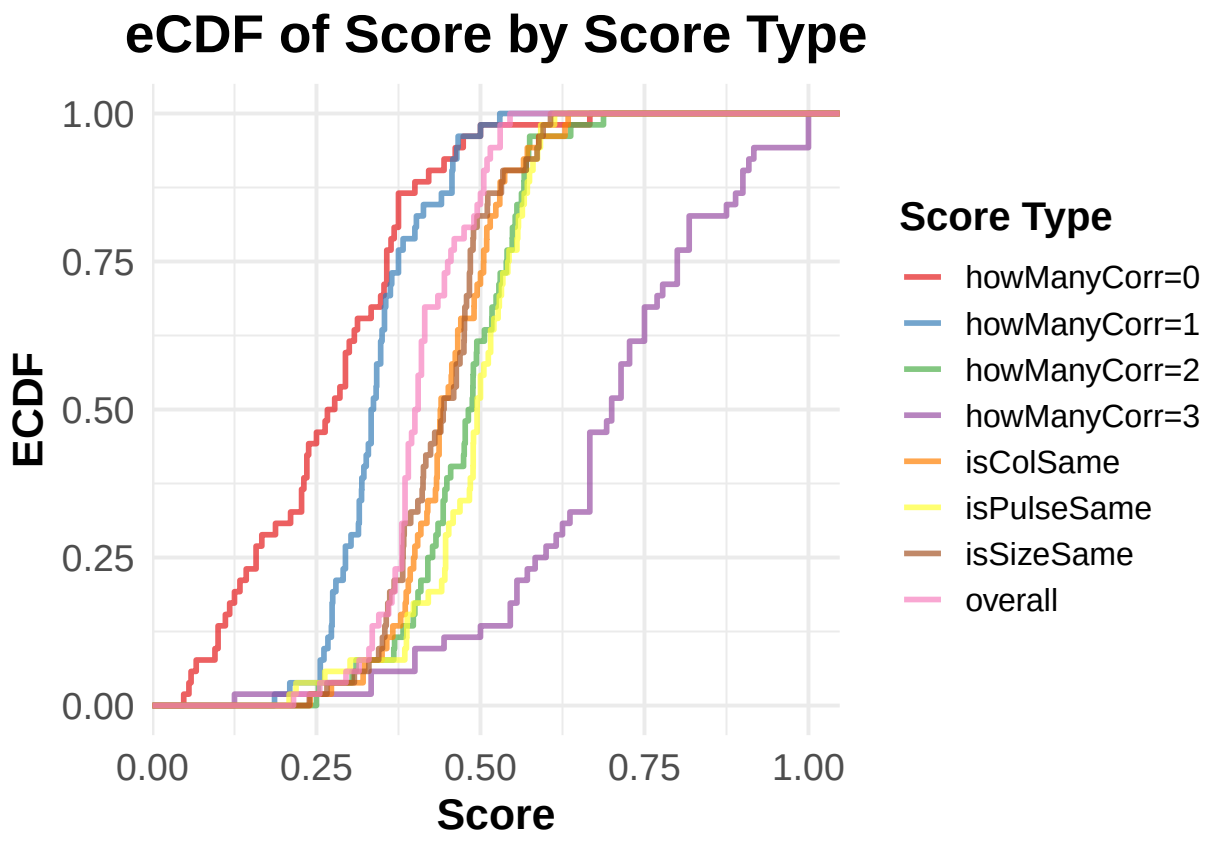
```
ggplot(hmc_df, aes(x=hmc, y=score, color=subject)) +
  geom_line(size=0.5) +
  # scale_color_brewer(palette = "Set1") +
  theme_minimal(base_size = 12) +
    labs(
    x = "How Many Correct Value",
    y = "Score for HMC",
    title = "Correct Quadrent Score by HMC Value per Subject",
  ) +
  theme(
  legend.position = "none",  # Remove legend if it becomes too cluttered
  plot.title = element_text(hjust = 0.5, face = "bold"),
  axis.title.x = element_text(face = "bold"),
  axis.title.y = element_text(face = "bold")
  )
```

```
## 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.
```



eCDF of Score by Type

```
ggplot(score_df, aes(x = score, color = score_type, fill = score_type)) +
  stat_ecdf(geom = "step", size = 1, alpha = 0.7) +
  labs(
    x = "Score",
    y = "ECDF",
    title = "eCDF of Score by Score Type",
    fill = "Score Type",
    color = "Score Type"
  ) +
  scale_fill_brewer(palette = "Set1") +
  scale_color_brewer(palette = "Set1") +
  theme_minimal(base_size = 15) +
  theme(
    legend.position = "right",
    legend.title = element_text(face = "bold"),
    legend.text = element_text(size = 12),
    plot.title = element_text(hjust = 0.5, face = "bold", size = 20),
    axis.title.x = element_text(face = "bold", size = 16),
    axis.title.y = element_text(face = "bold", size = 16),
    axis.text = element_text(size = 14)  # Increase axis text size
  )
```



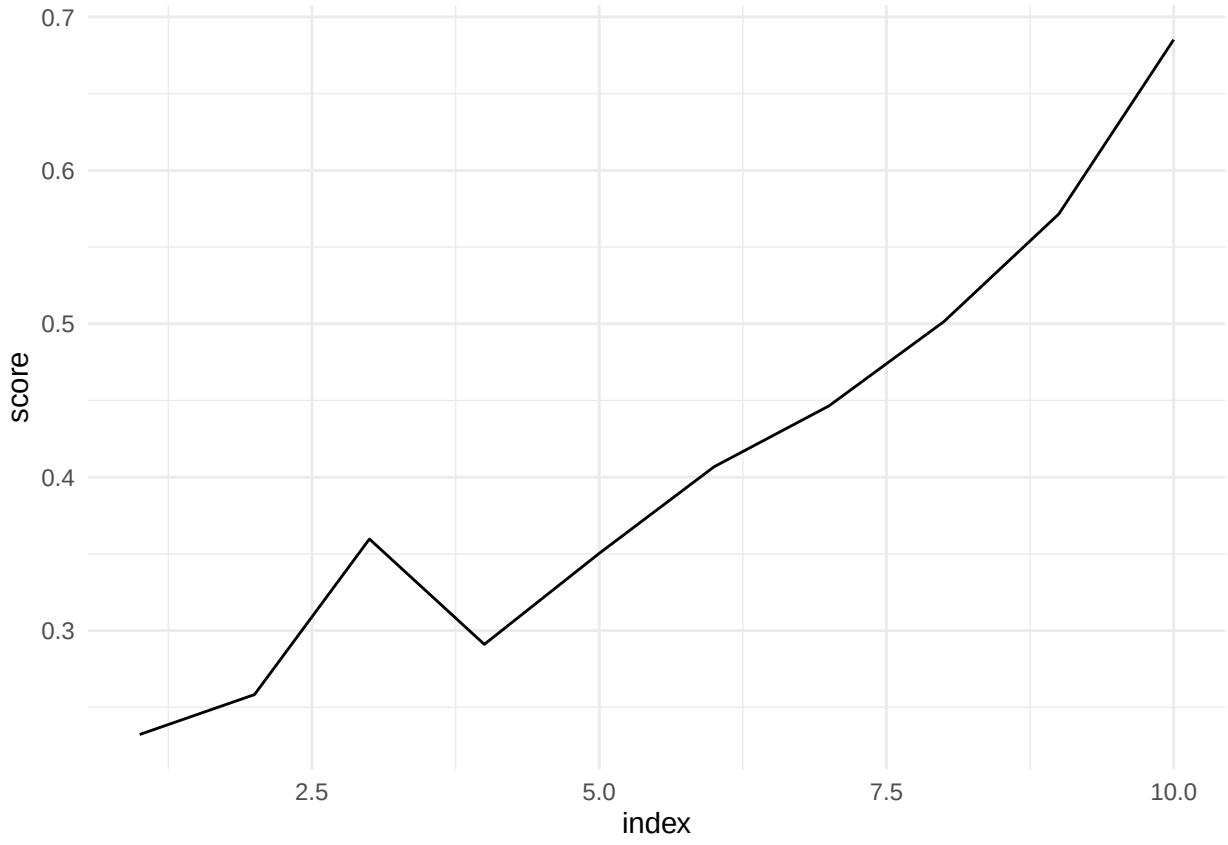
Effects of `distToMiddle` on Overall Score

```
q = quantile(df$distToMiddle, c(0, 0.1, .2, .3, .4, .5, .6, .7, .8, .9, 1))

index = c()
score = c()

for (i in 2:length(q)-1){
  lower = q[i]
  upper = q[i+1]

  temp = df[df$distToMiddle > lower & df$distToMiddle <= upper, ]$isCorrect
  t_score = sum(temp) / length(temp)
  index = c(index, i)
  score = c(score, t_score)
}

dtm_score_df = data.frame(
  score = score,
  index = index
)
q
```

```
##          0%         10%         20%         30%         40%         50%
## 0.004419553 0.038019707 0.060839468 0.077533188 0.091055108 0.106303687
```

```
##         60%         70%         80%         90%        100%
## 0.124544064 0.143062520 0.162151914 0.190197830 0.287484807
```

```
ggplot(dtm_score_df, aes(x=index, y=score)) +
  geom_line() +
  theme_minimal()
```



Rolling Mean

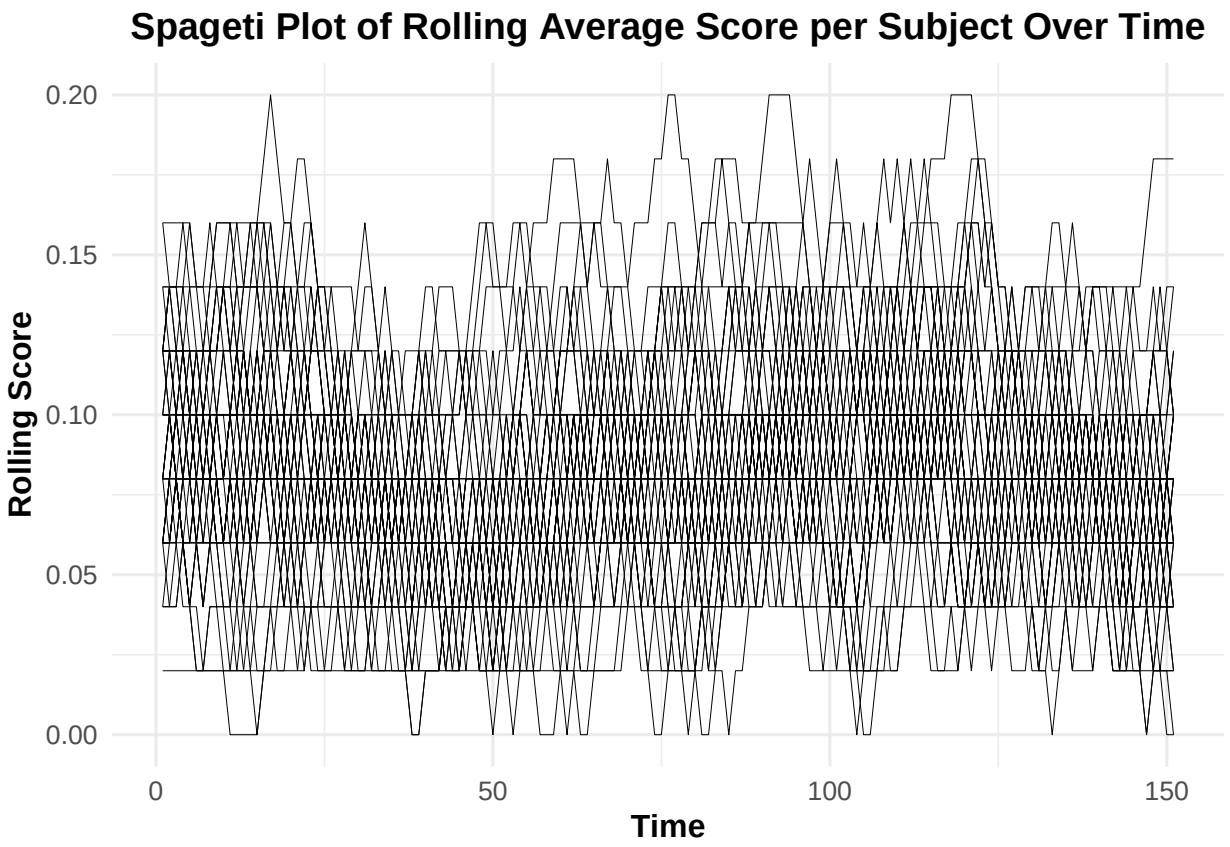
```
subjs = levels(df$subj)
smoothing_chunck = 50
samples_per_subj = 200
smoothing_samples = (samples_per_subj - smoothing_chunck) + 1

store = c()

for (i in 1:length(subjs)){
  subject_num = subjs[i]
  sorted_df = df[df$subj == subject_num, ] %>% arrange(X)
  is_correct = sorted_df$isCorrect


  temp_store = c()

  for (j in 1:smoothing_samples) {
```

```
    roll_chunck = sum(is_correct[j:(j+9)]) / smoothing_chunck
    temp_store = c(temp_store, roll_chunck)
  }

  store = c(store, temp_store)

}

rolling_mean_df = data_frame(
  rolling_mean = store,
  sample = rep(1:smoothing_samples, length(subjs)),
  subject = rep(1:length(subjs), each = smoothing_samples)
)
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## i Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
# rolling_mean_df$sample = as.factor(rolling_mean_df$sample)
# rolling_mean_df$subject = as.factor(rolling_mean_df$subject)
```

```
df_small = rolling_mean_df
df_small$subject = as.factor(df_small$subject)

ggplot(df_small, aes(x=sample, y=rolling_mean, group=subject)) +
  geom_line(size=0.15) +
  theme_minimal(base_size = 12) +
    labs(
    x = "Time",
    y = "Rolling Score",
    title = "Spageti Plot of Rolling Average Score per Subject Over Time",
  ) +
  theme(
  legend.position = "none",  # Remove legend if it becomes too cluttered
  plot.title = element_text(hjust = 0.5, face = "bold"),
  axis.title.x = element_text(face = "bold"),
  axis.title.y = element_text(face = "bold")
  )
```



- This was really not helpful.

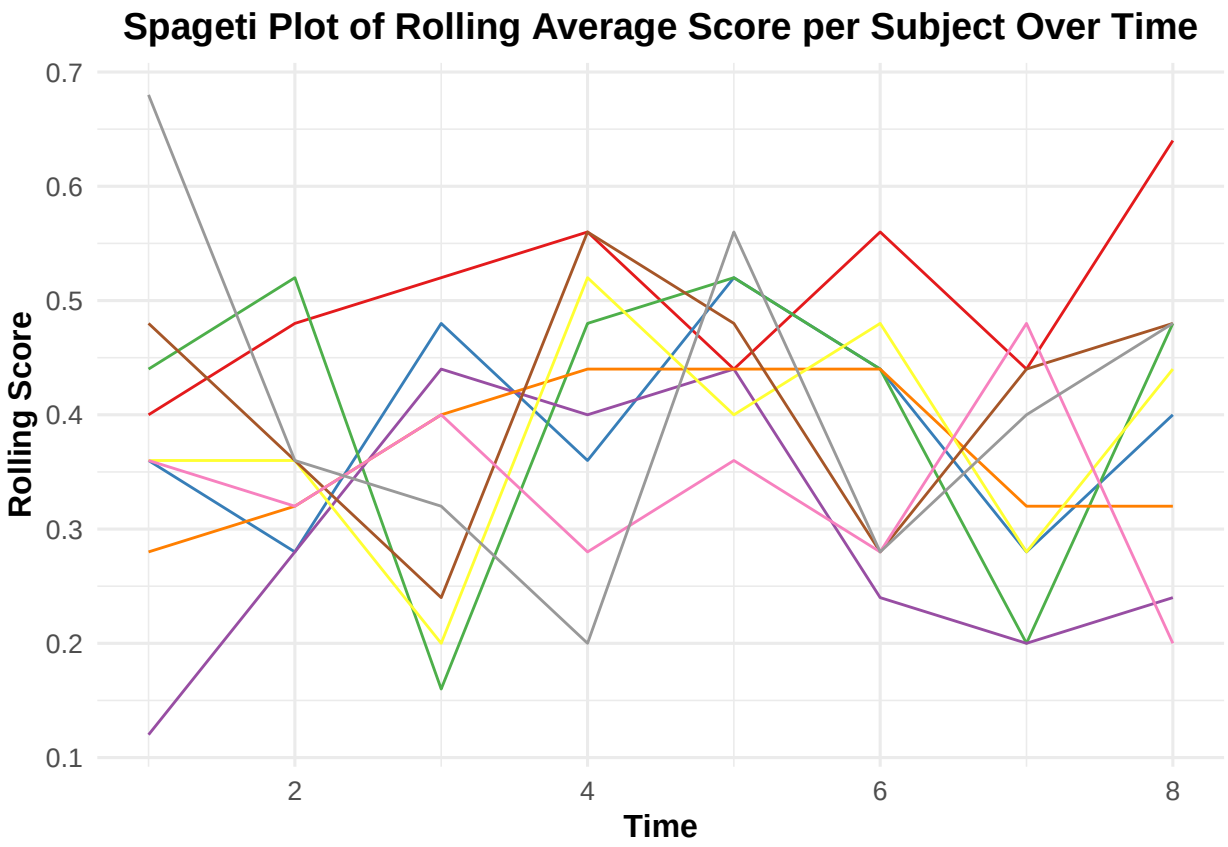
Trying again but looking at average for discrete 25 sample chunks. So % right in 1-25, 26-50, etc

```
subjs = levels(df$subj)
smoothing_chunck = 25  # Must be a factor of <samples_per_subj>
samples_per_subj = 200
smoothing_samples = samples_per_subj / smoothing_chunck

store = c()

for (i in 1:length(subjs)){
  subject_num = subjs[i]
  sorted_df = df[df$subj == subject_num, ] %>% arrange(X)
  is_correct = sorted_df$isCorrect


  temp_store = c()

  for (j in 1:smoothing_samples){
    chunk = is_correct[(1 + ((j-1) * 25)):(j*25)]
    t_score = sum(chunk) / smoothing_chunck
    temp_store = c(temp_store, t_score)
  }

  store = c(store, temp_store)
```

```
}

rolling_mean_df = data_frame(
  chunk_mean = store,
  sample = rep(1:smoothing_samples, length(subjs)),
  subject = rep(1:length(subjs), each = smoothing_samples)
)
```

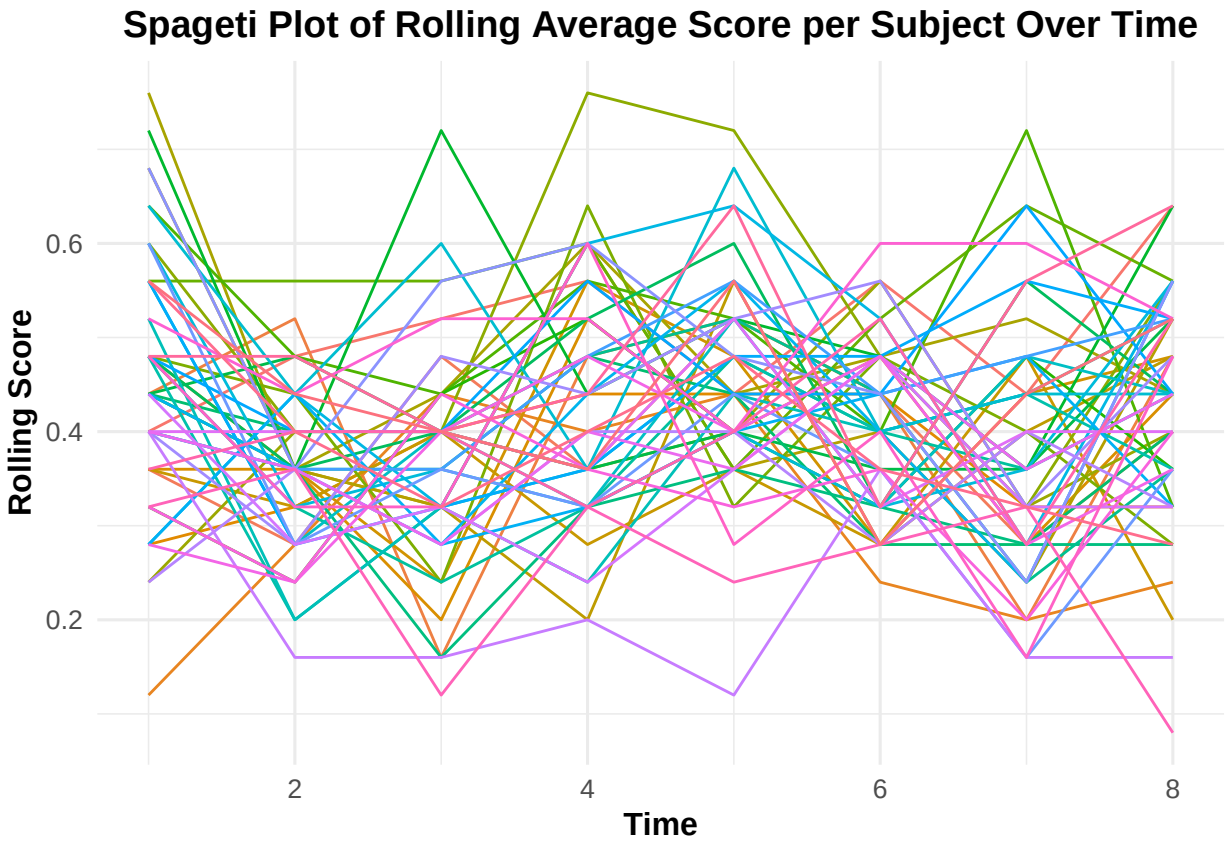
Just looking at the first 9 for easy visual

```
df_small = rolling_mean_df[rolling_mean_df$subject <= 9, ]
df_small$subject = as.factor(df_small$subject)

ggplot(df_small, aes(x=sample, y=chunk_mean, color=subject)) +
  geom_line(size=0.5) +
  scale_color_brewer(palette = "Set1") +
  theme_minimal(base_size = 12) +
    labs(
    x = "Time",
    y = "Rolling Score",
    title = "Spageti Plot of Rolling Average Score per Subject Over Time",
  ) +
  theme(
  legend.position = "none",  # Remove legend if it becomes too cluttered
  plot.title = element_text(hjust = 0.5, face = "bold"),
  axis.title.x = element_text(face = "bold"),
  axis.title.y = element_text(face = "bold")
  )
```



Okay now all subjs

```
df_small = rolling_mean_df
df_small$subject = as.factor(df_small$subject)

ggplot(df_small, aes(x=sample, y=chunk_mean, color=subject)) +
  geom_line(size=0.5) +
  # scale_color_brewer(palette = "Set1") +
  theme_minimal(base_size = 12) +
    labs(
    x = "Time",
    y = "Rolling Score",
    title = "Spageti Plot of Rolling Average Score per Subject Over Time",
  ) +
  theme(
  legend.position = "none",  # Remove legend if it becomes too cluttered
  plot.title = element_text(hjust = 0.5, face = "bold"),
  axis.title.x = element_text(face = "bold"),
  axis.title.y = element_text(face = "bold")
  )
```


Spageti Plot of Rolling Average Score per Subject Over Time
Rolling Score
0.6
0.4
0.2
2
4
6
8
Time