Dual Task

Eileen Petros

7/12/2022

Constants

```r
# Enums
WALK <- 1
VSTROOP <- 2
ASTROOP <- 3

# All in seconds
# TASK_DURATION <- 2
# VSTROOP_INTERVAL <- 1
# ASTROOP_INTERVAL <- 1
# WALK_TRIALS <- 1
# VSTROOP_TRAILS <- 1
# ASTROOP_TRIALS <- 1

TASK_DURATION <- 9
VSTROOP_INTERVAL <- 1.5
ASTROOP_INTERVAL <- 1.5
WALK_TRIALS <- 1
VSTROOP_TRAILS <- 1
ASTROOP_TRIALS <- 1

VSTROOP_STIMULI_COUNT <- TASK_DURATION / VSTROOP_INTERVAL
ASTROOP_STIMULI_COUNT <- TASK_DURATION / ASTROOP_INTERVAL

EVENT_COUNT <- WALK_TRIALS + VSTROOP_STIMULI_COUNT + ASTROOP_STIMULI_COUNT
# StartTrials, WalkTrial, VStroopTrial, AStroopTrial
# TODO change counts to 1, 6, 12, 12
# Depends on interstimulus duration
EVENT_A <- "# EVENT A - COUNT 1" #start treadmill
EVENT_B <- "# EVENT B - COUNT 1" #walk
EVENT_C <- "# EVENT C - COUNT 2" #vstroop
EVENT_D <- "# EVENT D - COUNT 2" #astroop

filePath <- "./Data/SL_dual0006.csv"
```

Data Import and Cleaning

```r
data <- read.csv(filePath)
data <- as_tibble(data)

# Remove separation lines
data <- data %>% filter(Time != "#")

# Extract summary rows and remove from data
summary <- data %>% slice_tail(n = EVENT_COUNT)
data <- data %>% slice(1:(n() - EVENT_COUNT))
```

Detect Stomp

```r
# https://newbedev.com/r-tidyverse-how-to-change-column-data-type-using-pipes-with-least-typing
data$FP1.ForY <- data$FP1.ForY %>% as.double()
data$FP2.ForY <- data$FP2.ForY %>% as.double()

stompTime <- data %>% filter(FP1.ForY == max(FP1.ForY, na.rm = TRUE))
(stompTime$Time <- stompTime$Time %>% as.double())
```

```
## [1] 13.35000 13.35333 13.35667
```

```r
(stompTime$FP1.ForY)
```

```
## [1] 1445.963 1445.963 1445.963
```

```r
stompTime2 <- data %>% filter(FP2.ForY == max(FP2.ForY, na.rm = TRUE))
(stompTime2$Time <- stompTime2$Time %>% as.double())
```

```
## [1] 13.81000 13.81333 13.81667
```

```r
(stompTime$FP2.ForY)
```

```
## [1] 8.983367 8.983367 8.983367
```

Subset by Stimulus using D-Flow Event Breaks

```r
# Find event detection rows
eventBreaks <- data %>%  with(which(is.na(FP1.CopX)))
eventBreaks <- eventBreaks %>% sort()

# break into sets for each event
sets <- list()
# Ignore first grouping from recording data to
for (i in 2:(EVENT_COUNT)) {
  # Offset to exclude labels themselves
  start <- (eventBreaks[i] + 1)
  end <- (eventBreaks[i + 1] - 1)
  varName <- paste0("set", i - 1)

  # Don't create indidiual set vars
  #assign(varName, data[start:end,])
  sets[[i - 1]] <- data[start:end,]
}

# Last set
```

```r
start <- eventBreaks[length(eventBreaks)] + 1
#assign(paste0("set", EVENT_COUNT), data[start :nrow(data),])
sets[[EVENT_COUNT]] <- data[start:nrow(data),]
```

Final Data Cleaning with All Text Removed

```r
data$Time <- as.double(data$Time)
```

```
## Warning: NAs introduced by coercion
```

Calculate Average Belt Speed for each Stimulus and Pull Sitmulus Info

```r
walkSpeeds <- c()
vStroopSpeeds <- c()
aStroopSpeeds <- c()

vStroopContent <- c()
vStroopCharacteristic <- c()

aStroopContent <- c()
aStroopCharacteristic <- c()

for (i in 1:EVENT_COUNT){
  currentSet <- as_tibble(sets[[i]])
  task <- first(currentSet$Task)

  # With character entries in time removed, convert col to double
  currentSet$Time <- as.double(currentSet$Time)
  sets[[i]] <- currentSet

  # Compute average speeds
  avgSpeed <- mean(currentSet$LeftBelt.Speed)

  if(task == WALK){
    walkSpeeds <- c(walkSpeeds, avgSpeed)
  }
  else if (task == VSTROOP){
    vStroopSpeeds <- c(vStroopSpeeds, avgSpeed)
    # Use first, these values are the same for the entire set
    vStroopContent <- c(vStroopContent, first(currentSet$Stimulus.Content))
    vStroopCharacteristic <- c(vStroopCharacteristic,
                               first(currentSet$Stimulus.Characteristic))
  }
  else if (task == ASTROOP){
    aStroopSpeeds <- c(aStroopSpeeds, avgSpeed)
    aStroopContent <- c(aStroopContent, first(currentSet$Stimulus.Content))
    aStroopCharacteristic <- c(aStroopCharacteristic,
                               first(currentSet$Stimulus.Characteristic))
  }
  else {
    print("Invlaid task type")
  }
}

walkSpeeds
```

```
## [1] 1.286082
```

```
vStroopSpeeds
```

```
## [1] 1.289661 1.330956 1.310103 1.267568 1.227065 1.266615
```

```
aStroopSpeeds
```

```
## [1] 1.265573 1.192783 1.129033 1.166725 1.172383 1.148375
```

Single Speed Value for Each Task Type

```
(mean(walkSpeeds))
```

```
## [1] 1.286082
```

```
(mean(vStroopSpeeds))
```

```
## [1] 1.281995
```

```
(mean(aStroopSpeeds))
```

```
## [1] 1.179145
```

Cognitive Performance

```
print("VStroop Correct Answer: 1 = Red, 2 = Green, 3= Blue, 4 = Yellow, 5 = Purple,
      6 = Orange, 7 = Gray, 8 = White, 9 = Black")
```

```
## [1] "VStroop Correct Answer: 1 = Red, 2 = Green, 3= Blue, 4 = Yellow, 5 = Purple, \n      6 = Orange
```

```
(vStroopCharacteristic)
```

```
## [1] 9 9 6 4 1 3
```

```
print("AStroop Correct Answer: -0.3 = LOW, 0.3 = HIGH")
```

```
## [1] "AStroop Correct Answer: -0.3 = LOW, 0.3 = HIGH"
```

```
(aStroopCharacteristic)
```

```
## [1] -0.3 -0.3  0.3 -0.3  0.3 -0.3
```

```
print("Results")
```

```
## [1] "Results"
```

```
vStroopResults <- c(1,1,1,1,1,1)
aStroopResults <- c(1,0,1,1,1,0)

(vStroopResults)
```

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

```r
(vStroopPercent <- sum(vStroopResults) / length(vStroopResults))
```

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

```r
(aStroopResults)
```

```
## [1] 1 0 1 1 1 0
```

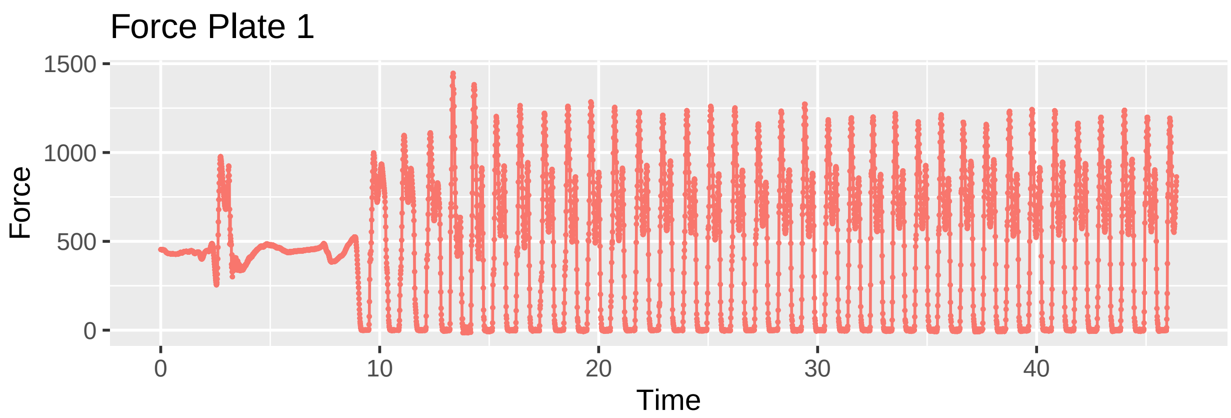
```r
(aStroopPercent <- sum(aStroopResults) / length(aStroopResults))
```

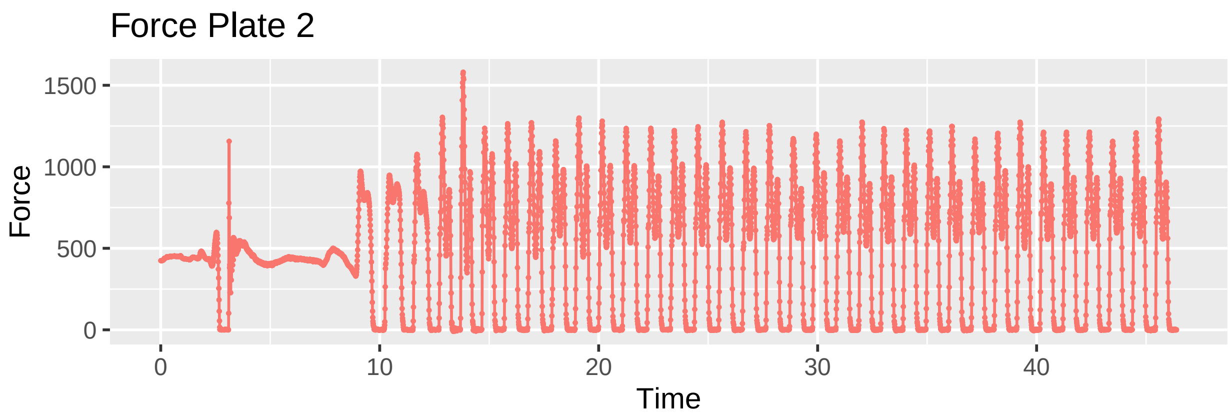
```
## [1] 0.6666667
```

Visualizing Forces

```r
# dataToPlot <- sets[[6]]
# ggplot(data = dataToPlot, mapping = aes(x = Time, y = FP1.ForY)) +
#   geom_line() +
#   geom_point() +
#   xlab("Time") +
#   ylab("Force (N)")

forcePlate1Plot <- ggplot(data = data, mapping =
                            aes(x = Time, y = FP1.ForY, color = "Red")) +
  geom_line() +
  geom_point(size = 0.25) +
  labs(title = "Force Plate 1", x = "Time", y = "Force") +
  theme(legend.position = "none")

forcePlate2Plot <- ggplot(data = data, mapping =
                            aes(x = Time, y = FP2.ForY, color = "Blue")) +
  geom_line() +
  geom_point(size = 0.25) +
  labs(title = "Force Plate 2", x = "Time", y = "Force") +
  theme(legend.position =  "none")

plot_grid(forcePlate1Plot, forcePlate2Plot, ncol = 1, vjust = 0.5)
```

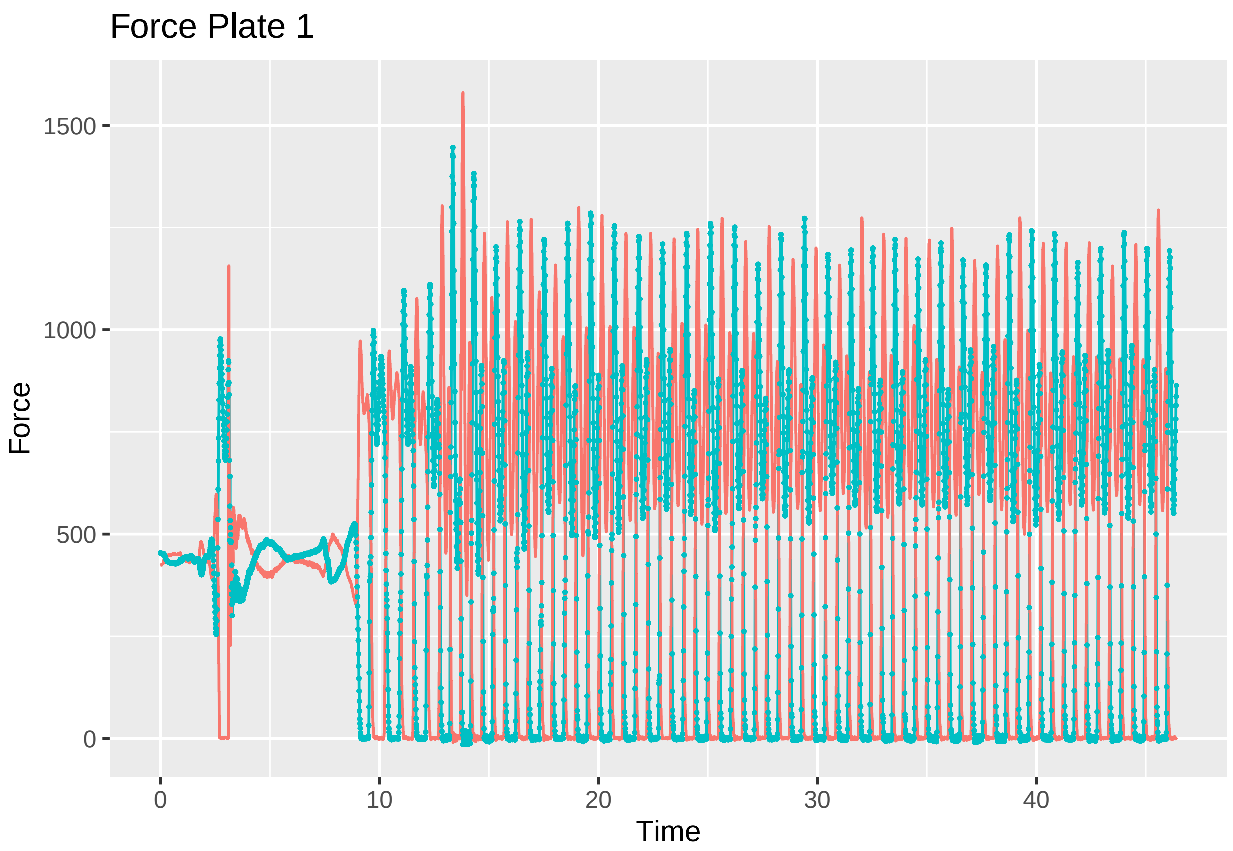
```
## Warning: Removed 14 row(s) containing missing values (geom_path).

## Warning: Removed 14 rows containing missing values (geom_point).

## Warning: Removed 14 row(s) containing missing values (geom_path).

## Warning: Removed 14 rows containing missing values (geom_point).
```





```r
ggplot(data = data, mapping = aes(x = Time, y = FP1.ForY, color = "Red")) +
  geom_line() +
  geom_point(size = 0.25) +
  geom_line(mapping = aes(y = FP2.ForY, color = "Blue")) +
  geom_point(size = 0.25) +
  labs(title = "Force Plate 1", x = "Time", y = "Force") +
  theme(legend.position = "none")
```

```
## Warning: Removed 14 row(s) containing missing values (geom_path).

## Warning: Removed 14 rows containing missing values (geom_point).

## Warning: Removed 14 row(s) containing missing values (geom_path).

## Warning: Removed 14 rows containing missing values (geom_point).
```



```
# Flip axis labesl
# theme(axis.text.x = element_text(angle = 90, vjust = 1, hjust = 1))

# ggplot(mm, aes(x = times, y = value)) + geom_line(aes(color = variable)) +
#       facet_grid(variable ~ ., scales = "free_y") + theme(legend.position = "none")
```