**HEBB'S Lab**

Human Enhancement Brain and Body
Systems Science Laboratory

# Quick guide for pre-processing experiment data for motor tasks

Contents

1. Basics in behavioral data analysis (for motor tasks in this lab)

Behavioral experiment & analysis basic steps

1) Data collection (i.e., run experiments)

- Data are stored as text files: row=time, column= data type (position, velocity, etc.)

2) Preprocessing

This slide is about this

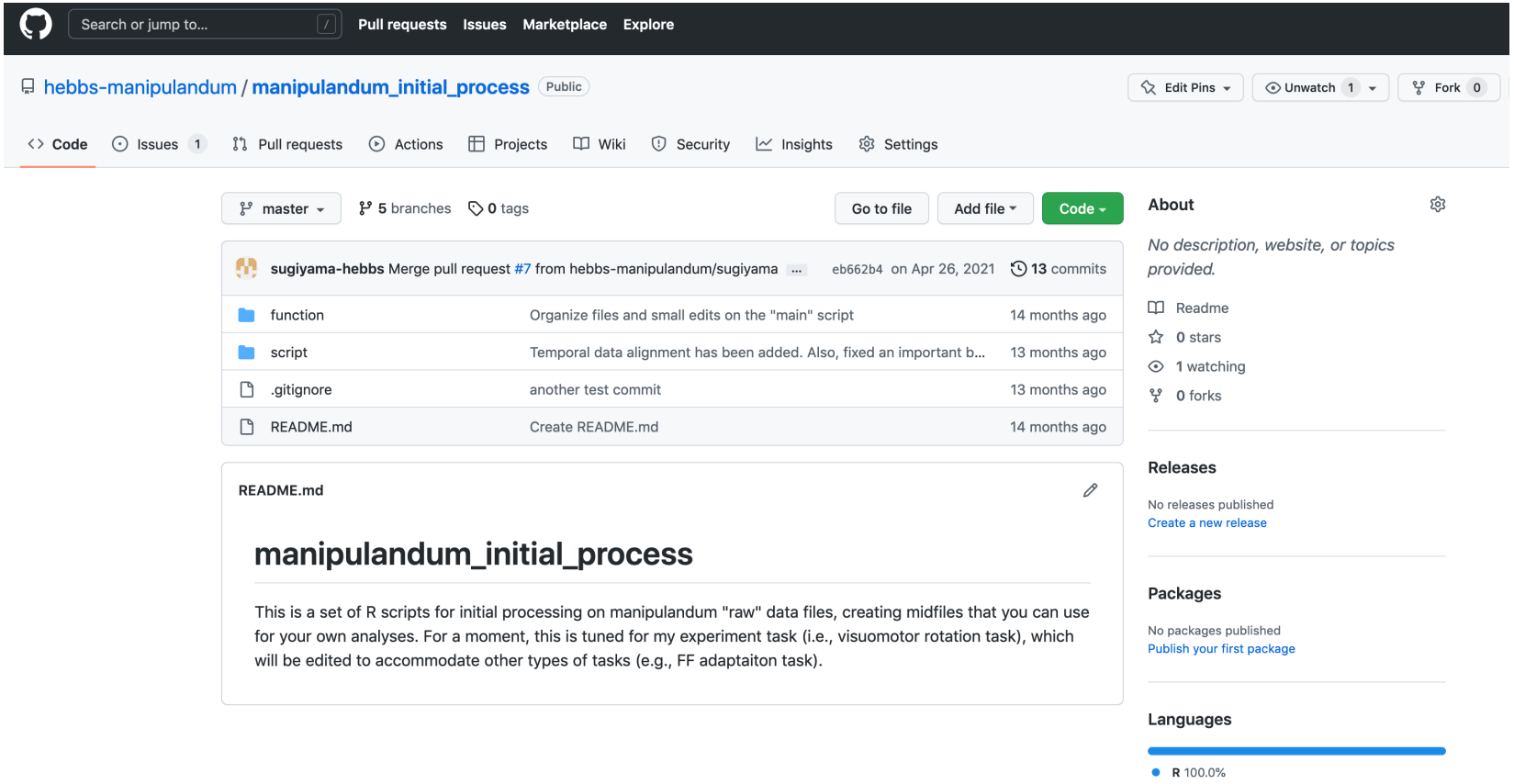
- Convert raw files into “midfiles”
  - Facilitate data handling (labeling data, organizing, etc.)
  - Initial processing
    - “zero-ing”, alignment, etc.
    - Smoothing, filtering
  - Data-size reduction (down-sampling)
    - Usually, 1000Hz is too high for our motor tasks (200Hz is typically good enough)

3) Analysis

- Data plot
- Stats
- etc.

# 2. Organization of directory & Data

R script packet for pre-processing is availabe on Github



Directory organization

- (main directory)
  - xxx.Rproj – Rproject file (you open Rstudio)
  - data
    - rawdata: this is where you put text data file
    - processed: output directory
  - figure
  - function
  - script
    - “main” scripts are located here
    - sub_script: Main scripts call and use these

* My style is largely influenced by this tutorial: so check Lesson 1 and 2 (and more as needed)
https://pagepiccinini.com/r-course/

For each block, an experiment script makes 4 types of text file

One file per block

- data_load: “sequence” file used for the block
- data_param: key parameter of experiment task

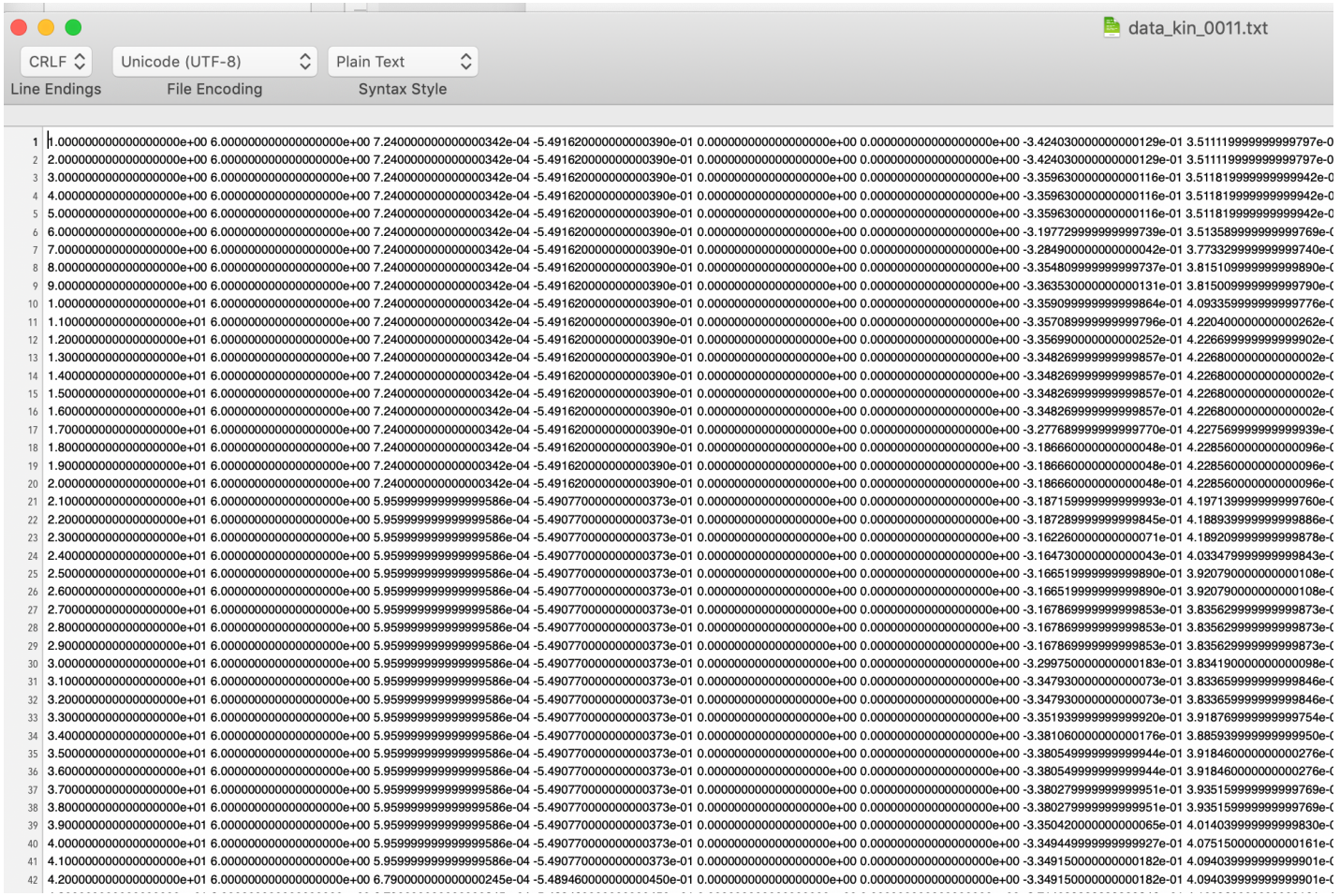
One file per trial

- data_point: some important single values per trial

  (e.g., xy-position at crossing, endpoint-angle)
- data_kin: kinematic data at each recording

kinematic data

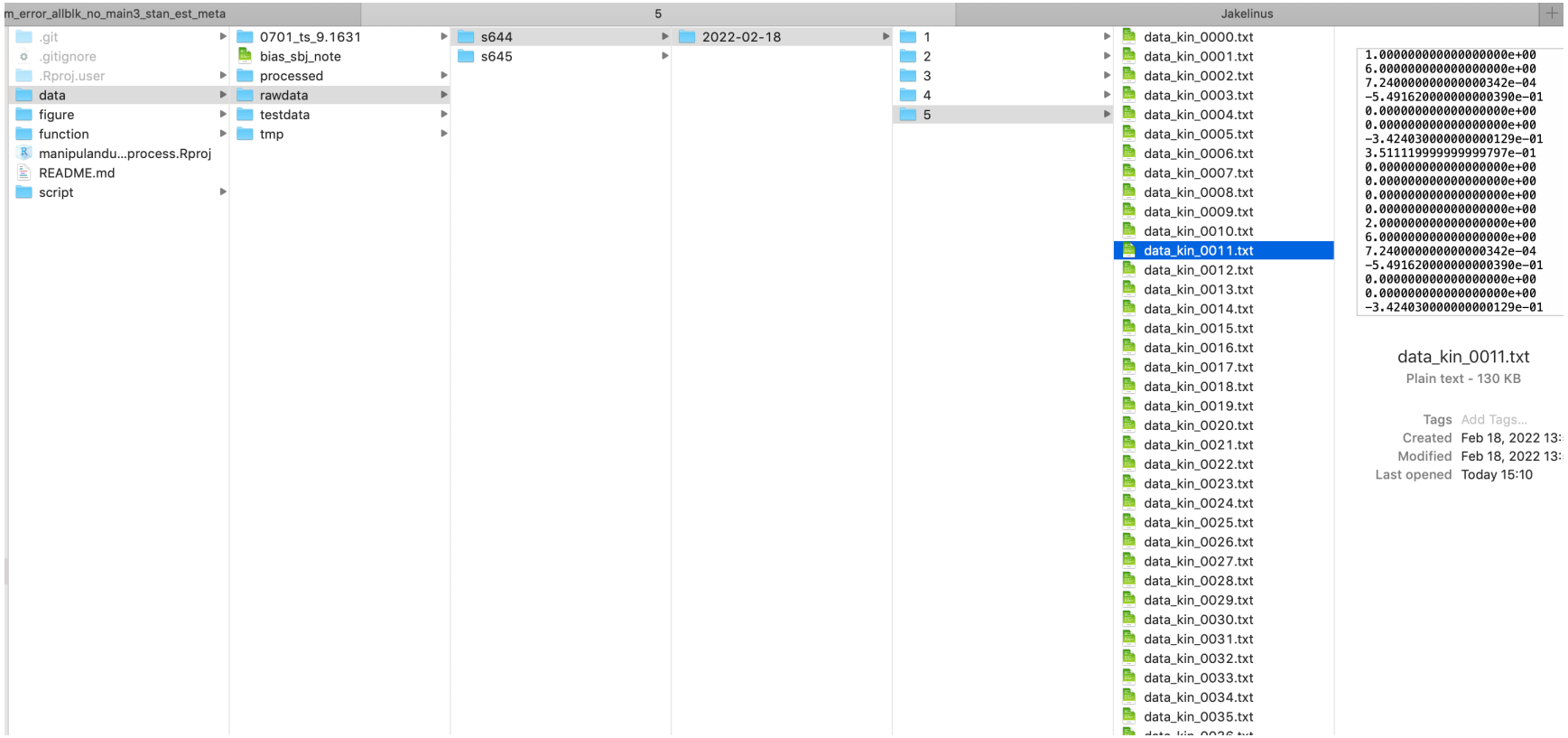
- row: time point (each recording)
- column: data type (x-position, y-position, etc.) * specified in an experiment script, and also labeled in pre-processing



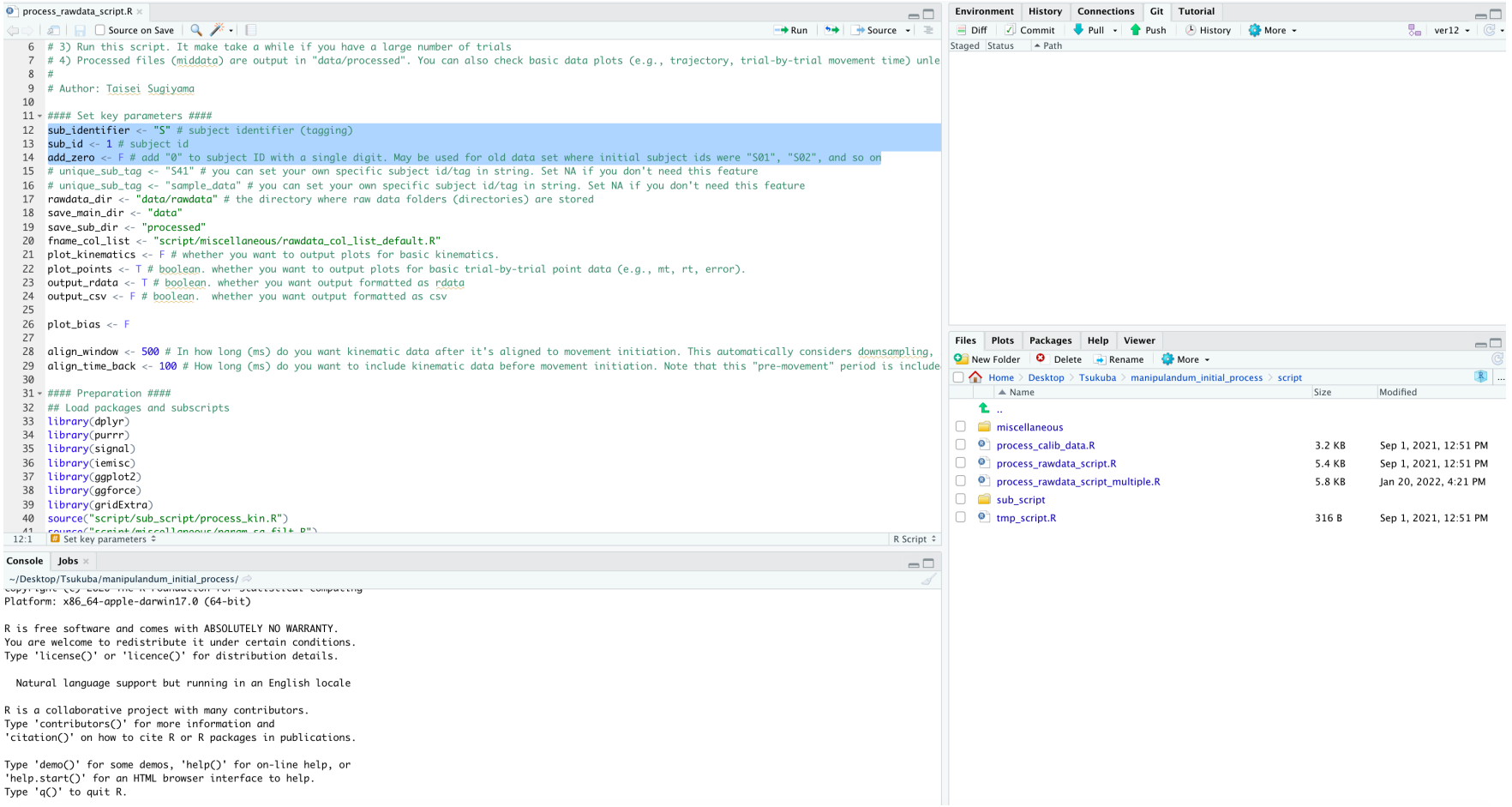
# 3. Procedure (quick manual)

1. Put experiment data under data/rawdata

The example shows s644 (subject ID), which participated on Feb 18, 2022 and performed 5 blocks



2. Open Rproject, open “script/process_rawdata_script.R” set appropriate parameters, and run.
    - You may run other scripts under the same directory if you know what they are.
    - Depending on your experiment program, you may need to modify column labels, number of columns, etc.



(optional) Check process_rawdata_script.R and the subscripts used in it to understand what processing is done, especially on kinematic data

process_rawdata_script.R | read_files.R

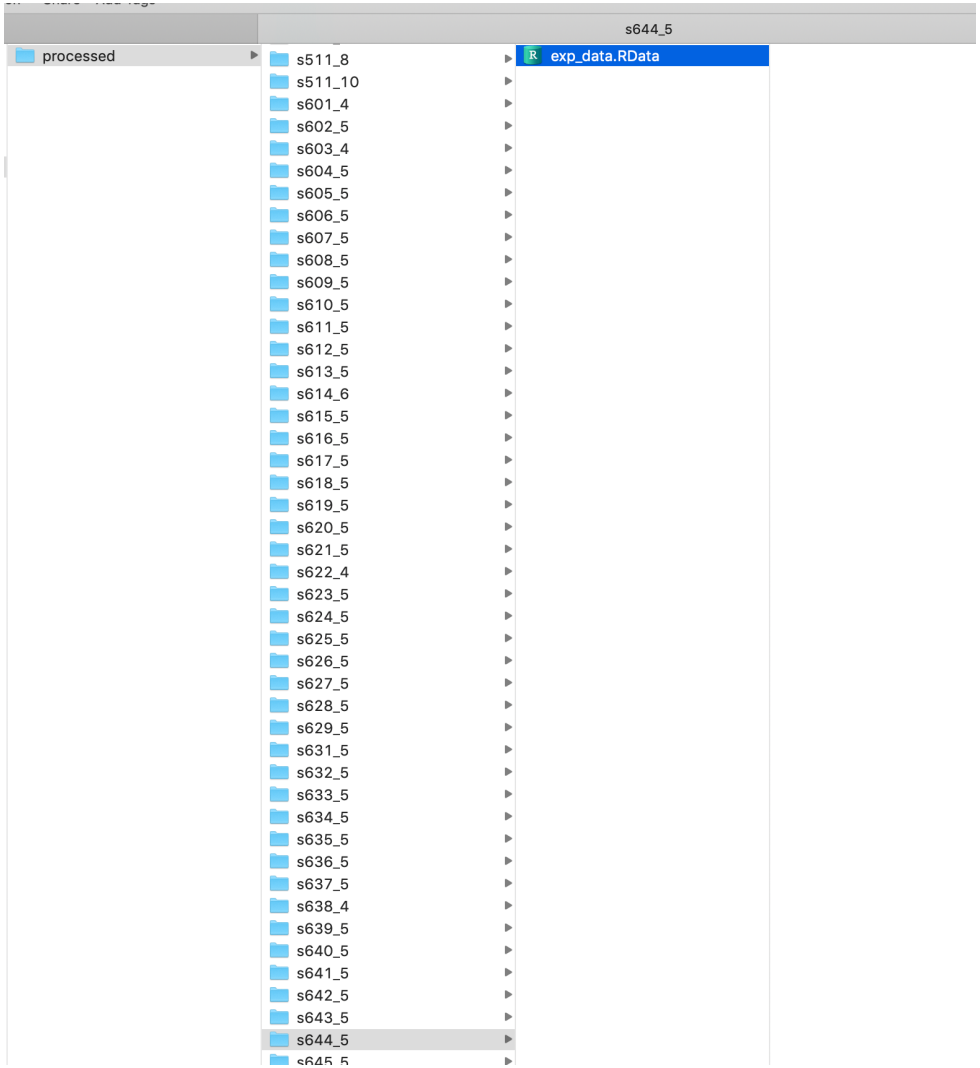
```r
names(tgt_raw) <- c(tgt_col,"blk_tri")

param_raw <- data.frame(read.table(fname_param, header = T, row.names = NULL)) # block-wide parameters

# single-point data
point_raw <- map(fname_point,function(fname){
  tri_filename <- gsub("^.*/","",fname) # remove file path
  tri_num <-   as.numeric(gsub("[^0-9]","",tri_filename))+1 # extract trial number (starting from 0, so add 1)
  return_df <- data.frame(read.table(fname, header = F, row.names = NULL)) %>%
    mutate(blk_tri = tri_num) # get block trial number, which is in filename (starting from 0, so add 1)
}) %>%
  reduce(rbind)

colnames(point_raw) <- c(point_col,"blk_tri")

# kinematic data
task_center_x <- param_raw$center_pos_x
task_center_y <- param_raw$center_pos_y

kin_raw <- map(fname_kin,function(fname){

  tri_filename <- gsub("^.*/","",fname) # remove file path
  tri_num <-   as.numeric(gsub("[^0-9]","",tri_filename))+1 # extract trial number (starting from 0, so add 1)
  return_df_raw <- data.frame(read.table(fname, header = F, row.names = NULL)) # %>%

  colnames(return_df_raw) <- kin_col

  return_df_raw2 <- return_df_raw %>%
    mutate(x = (x - task_center_x), y = (y - task_center_y)) # zero-ing with respect to the center of task space.

  return_df <- process_kin(return_df_raw2,reduce_hz,reduce_hz_rate) %>%
    mutate(blk_tri = tri_num)

}) %>%
  reduce(rbind)
```

```r
process_kin <- function(df, reduce_hz, reduce_hz_rate){

# pos_data_tri <- subset(pos_data, trial == tri)

x <- df$x # raw x position
y <- df$y # raw y position

## SG filtering (smoothing & derivatives).
# x
sx <- sgolayfilt(x, p = order, n = framelen, m = 0, ts = 1/sample_rate) # smooth
svx <- sgolayfilt(x, p = order, n = framelen, m = 1, ts = 1/sample_rate) # velocity
sax <- sgolayfilt(x, p = order, n = framelen, m = 2, ts = 1/sample_rate) # acceleration
sjx <- sgolayfilt(x, p = order, n = framelen, m = 3, ts = 1/sample_rate) # jerk

# y
sy <- sgolayfilt(y, p = order, n = framelen, m = 0, ts = 1/sample_rate) # smooth
svy <- sgolayfilt(y, p = order, n = framelen, m = 1, ts = 1/sample_rate) # velocity
say <- sgolayfilt(y, p = order, n = framelen, m = 2, ts = 1/sample_rate) # acceleration
sjy <- sgolayfilt(y, p = order, n = framelen, m = 3, ts = 1/sample_rate) # jerk

vx <- df$vx # robot x-velocity
vy <- df$vy # robot y-velocity
state <- df$state # trial state


if (reduce_hz) {
  return_df <- data.frame(tstep = 1:length(x), state, x, y, vx, vy, sx, sy, svx, svy, sax, say, sjx, sjy, fx = df$fs_x, fy = df$fs_y)  %>%
    # dplyr::filter(tstep %% red_rate == 1) # reduce sample Hz
    dplyr::filter(tstep %% reduce_hz_rate == 1) # reduce sample Hz
} else{
  return_df <- data.frame(tstep = 1:length(x), state, x, y, vx, vy, sx, sy, svx, svy, sax, say, sjx, sjy, fx = df$fs_x, fy = df$fs_y)
}
```

3. Copy/move output file to your analysis packet

- Midfile is saved as Rdata for each block (or csv if you specify it in process_raw_data_script.R)
- It containts a list with 4 data frames
  - point: “point” values for each trials (e.g., endpoint angle)
  - tgt: sequence file data
  - kin: kinematic data
  - param: block parameter values (e.g., target distance)

Use this for subsequent analysis (plot, stats, etc.)



3. Copy/move output file to your analysis packet

My analysis packets have the same organization of directory, so familiarizing yourself with it helps if you also use them (but feel free to eventually develop your own style of organizing)