**Analytic flow described in the main text.**

General work flow:  
- preporcess data  
- fourier analysis  
- principle component analysis  
- linear discriminant analysis

# **Importation of packages and defining functions**

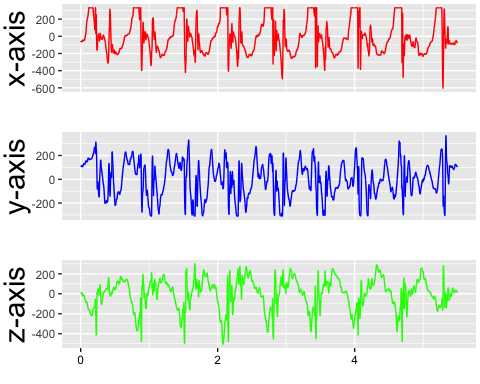
rm(list=ls())  
library(ggplot2)  
library(dplyr)  
library(gridExtra)  
library(MASS)  
setwd("/Users/MatsuiHiroshi/Dropbox/ongoing experiment/Acce")  
  
# define functions   
read\_data <- function(file){  
 fps <- 1/5000  
 d <- read.csv(file, skip=1)  
 colnames(d) <- c("x", "y", "z")  
   
   
 d$time <- seq(0, (nrow(d)-1) \* fps, by = fps)  
 return(d)  
}  
  
get\_info <- function(file){  
 N <- nrow(read.csv(file, skip=1))  
 R <- strsplit(file, "\_") %>% unlist  
 R[3] <- strsplit(R[3], fixed=TRUE, split=".")[[1]][1]  
 rows <- data.frame(rep(R, N) %>% matrix(., ncol=3, byrow=TRUE))  
 colnames(rows) <- c("cond", "trial", "id")  
 return(rows)  
}  
  
get\_freq <- function(d){  
 sampling = nrow(d)   
 n = 0:(sampling-1)   
 samplefreq = 5000   
 f = n\*samplefreq/sampling  
   
 spec\_x = abs(fft(d$x))   
 spec\_y = abs(fft(d$y))  
 spec\_z = abs(fft(d$z))  
 return(cbind(spec\_x, spec\_y, spec\_z, f))  
}  
  
select\_data = function(d){  
 inp <- dplyr::select(d, spec\_x, spec\_y, spec\_z) %>% reshape2::melt()  
 return(inp[,2])  
}

# **Importation and sorting of data**

# import of data  
d <- list.files(pattern=".csv") %>% lapply(., read\_data) %>% do.call(rbind,.)  
info <- list.files(pattern=".csv") %>% lapply(., get\_info) %>% do.call(rbind,.)  
d <- cbind(d, info)  
d$ser <- as.integer(interaction(d$cond, d$trial, d$id, drop=TRUE))  
  
# sort and centerize data   
d = d[order(d$ser),]  
d$x <- unlist(tapply(d$x, d$ser, function(x) x - mean(x)))  
d$y <- unlist(tapply(d$y, d$ser, function(x) x - mean(x)))  
d$z <- unlist(tapply(d$z, d$ser, function(x) x - mean(x)))

# **Example of raw data**

grid.arrange(ggplot(d[d$ser==3,]) +   
 geom\_line(aes(x=time, y=x), colour="red") +   
 theme(axis.text.x = element\_text(colour="white"),  
 axis.ticks.x = element\_line(colour="white"),  
 axis.title.x = element\_text(colour="white"),  
 axis.title.y = element\_text(size=rel(2)),  
 plot.margin = unit(c(0.1,.1,0,0.1), "cm") ) + ylab("x-axis"),  
 ggplot(d[d$ser==3,]) +   
 geom\_line(aes(x=time, y=y), colour="blue") +   
 theme(axis.text.x = element\_text(colour="white"),  
 axis.ticks.x = element\_line(colour="white"),  
 axis.title.x = element\_text(colour="white"),  
 axis.title.y = element\_text(size=rel(2)),  
 plot.margin = unit(c(0.1,.1,0,0.1), "cm") ) + ylab("y-axis"),  
 ggplot(d[d$ser==3,]) +   
 geom\_line(aes(x=time, y=z), colour="green") +   
 theme(axis.text.x = element\_text(colour="black"),  
 axis.ticks.x = element\_line(colour="black"),  
 axis.title.x = element\_text(colour="white"),  
 axis.title.y = element\_text(size=rel(2)),  
 plot.margin = unit(c(0.1,.1,0,0.1), "cm") ) + ylab("z-axis")  
 )



# **Fourier analysis**

d = d[d$time <= min(tapply(d$time, d$ser, max)),]  
d = split(d, d$ser) %>% lapply(., get\_freq) %>% do.call(rbind,.) %>% cbind(d,. )  
inp = split(d, d$ser) %>% lapply(., select\_data) %>% do.call(rbind,.)

# principle component analysis

res <- prcomp(inp)  
d\_pc <- data.frame(res$x)  
d\_pc$cond <- tapply(as.character(d$cond), d$ser, function(x)x[1])  
d\_pc$id <- factor(tapply(as.character(d$id), d$ser, function(x)x[1]), levels=c("yb","hm"))

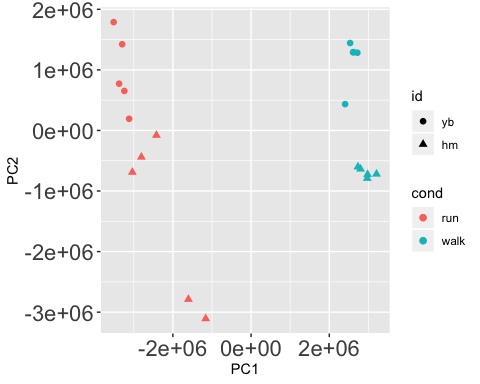
**The cumulative variance explained is as follows:**

## PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8 PC9   
## 0.55456 0.67394 0.76553 0.81719 0.85627 0.88133 0.90203 0.92066 0.93674   
## PC10 PC11 PC12 PC13 PC14 PC15 PC16 PC17 PC18   
## 0.95145 0.96562 0.97544 0.98401 0.98916 0.99285 0.99568 0.99740 0.99894   
## PC19 PC20   
## 1.00000 1.00000

We arbitrarily used only PC1 and PC2.

# **Visualization of PC scores:**

ggplot(d\_pc) +   
 geom\_point(aes(x=PC1, y=PC2, colour=cond, shape=id), size=2) +   
 theme(axis.text = element\_text(size=rel(1.5)))



# **Linear discriminant analysis**

Leave-one-out cross validation was used to check the accuracy.

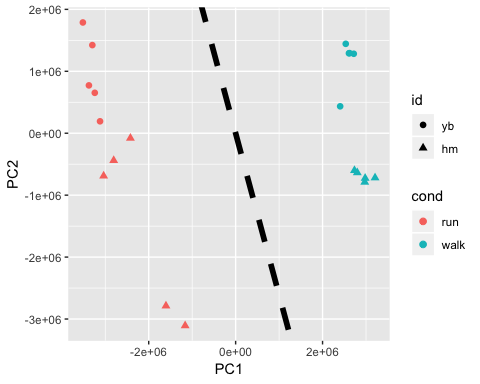
**Distinguishing the conditions (walk or run):**

fit <- lda(cond ~ PC1 + PC2, data=d\_pc, CV=TRUE)  
ct <- table(d\_pc$cond, fit$class)  
p = sum(diag(prop.table(ct)))   
print(paste("The accuracy is", p \* 100, "%"))

## [1] "The accuracy is 100 %"

**Visualizing boundary:**

coeff = lda(cond ~ PC1 + PC2, data=d\_pc, CV=FALSE)  
gmean = coeff$prior %\*% coeff$means  
const = as.numeric(gmean %\*% coeff$scaling)  
z <- as.matrix(d\_pc[, 1:2]) %\*% coeff$scaling - const  
slope = - coeff$scaling[1] / coeff$scaling[2]  
intercept = const / coeff$scaling[2]  
  
ggplot(d\_pc) +   
 geom\_point(aes(x=PC1, y=PC2, colour=cond, shape=id), size=2) +   
 geom\_abline(slope = slope, intercept = intercept, lwd=2, lty=2)



**Distinguishing the subjects (YB or HM):**

fit <- lda(id ~ PC1 + PC2, data=d\_pc, CV=TRUE)  
ct <- table(d\_pc$id, fit$class)  
p = sum(diag(prop.table(ct)))   
print(paste("The accuracy is", p \* 100, "%"))

## [1] "The accuracy is 95 %"

**Visualizing boundary:**

coeff = lda(id ~ PC1 + PC2, data=d\_pc, CV=FALSE)  
gmean = coeff$prior %\*% coeff$means  
const = as.numeric(gmean %\*% coeff$scaling)  
z <- as.matrix(d\_pc[, 1:2]) %\*% coeff$scaling - const  
slope = - coeff$scaling[1] / coeff$scaling[2]  
intercept = const / coeff$scaling[2]  
  
ggplot(d\_pc) +   
 geom\_point(aes(x=PC1, y=PC2, colour=cond, shape=id), size=2) +   
 geom\_abline(slope = slope, intercept = intercept, lwd=2, lty=2)

