Case Two Report

RUNTIME:

Call Libraries

```
library(zoo)
```

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

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

```
library(varian)
```

```
## Loading required package: rstan
```

```
## Loading required package: StanHeaders
```

```
## Loading required package: ggplot2
```

```
## rstan (Version 2.21.2, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
library(tidyverse)
```

```
## -- Attaching packages ----------------------------------------------------------- tidyverse 1.3.0 --
```

```
## v tibble  3.0.3     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.4
```

```
## -- Conflicts -------------------------------------------------------------- tidyverse_conflicts() --
## x tidyr::extract() masks rstan::extract()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
## Loaded glmnet 4.0-2
library(FactoMineR)
library(factoextra)
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
library(MASS)

##
## Attaching package: 'MASS'
## The following object is masked from 'package:dplyr':
##
##     select
library(caret)

## Loading required package: lattice
##
## Attaching package: 'caret'
## The following object is masked from 'package:purrr':
##
##     lift
library(MLmetrics)

##
## Attaching package: 'MLmetrics'
## The following objects are masked from 'package:caret':
##
##     MAE, RMSE
## The following object is masked from 'package:base':
##
##     Recall
library(pROC)

## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
library(gridExtra)

##
## Attaching package: 'gridExtra'
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(grid)
library(broom)
library(knitr)
```

```
library(sjPlot)
library(RColorBrewer)
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

I. Introduction

II a. Data

Create Data and Feature Engineering

```
features_all_modalities = function(fn, win_size, shift) {
  df_all = read.csv(fn) %>%
    mutate(ACC_chest_3D = sqrt(ACC_chest_X^2+ACC_chest_Y^2+ACC_chest_Z^2)) %>%
    mutate(ACC_wrist_3D = sqrt(ACC_wrist_x^2+ACC_wrist_y^2+ACC_wrist_z^2))
  drops <- c("X", "Label", "subject")
  df = df_all[ , !(names(df_all) %in% drops)]
  replace_rows =  length(rollapply(df[,1], width = win_size*4, by = shift, FUN = mean, align = "left"))
  features_df <- data.frame(matrix(ncol = ncol(df)*4, nrow = replace_rows))
  new_names = sapply(1:length(df), function(c) {
    c(paste0(colnames(df)[c],"_mean"),
      paste0(colnames(df)[c],"_sd"),
      #paste0(colnames(df)[c],"_range"),
      paste0(colnames(df)[c],"_min"),
      paste0(colnames(df)[c],"_max")
      # paste0(colnames(df)[c],"_skew")
      )
  })
  colnames(features_df) = new_names



  for (c in 1:length(df)) {

    # finding mu
    mu_vals = rollapply(df[,c], width = win_size*4, by = shift, FUN = mean, align = "left")
    new_col_name = paste0(colnames(df)[c],"_mean")
    cindx = which(colnames(features_df)==new_col_name)
    features_df[, cindx] = mu_vals

    # finding sd
    sd_vals = rollapply(df[,c], width = win_size*4, by = shift, FUN = var, align = "left")
    new_col_name = paste0(colnames(df)[c],"_sd")
```

```
    cindx = which(colnames(features_df)==new_col_name)
    features_df[, cindx] = sqrt(sd_vals)

    # finding max
    max_vals = rollapply(df[,c], width = win_size*4, by = shift, FUN = max, align = "left")
    new_col_name = paste0(colnames(df)[c],"_max")
    cindx = which(colnames(features_df)==new_col_name)
    features_df[, cindx] = max_vals

    # finding min
    min_vals = rollapply(df[,c], width = win_size*4, by = shift, FUN = min, align = "left")
    new_col_name = paste0(colnames(df)[c],"_min")
    cindx = which(colnames(features_df)==new_col_name)
    features_df[, cindx] = min_vals

  }

 # make sure merge label, subject back into feature dataframe
  features_df$Label = df_all[1:replace_rows, "Label"]
  features_df$Subject = df_all[1:replace_rows, "subject"]
  return(features_df)
}
```

```
HR_calc = function(df, win_size, shift) {
  df <- df[rep(seq_len(nrow(df)), each = 4), ]
  mu_vals = rollapply(df, width = win_size*4, by = shift, FUN = mean, align = "left")
  sd_vals = rollapply(df, width = win_size*4, by = shift, FUN = sd, align = "left")
  min_vals = rollapply(df, width = win_size*4, by = shift, FUN = min, align = "left")
  max_vals = rollapply(df, width = win_size*4, by = shift, FUN = max, align = "left")

  return(list(mu = mu_vals, sd = sd_vals,
              min = min_vals, max = max_vals))
}
```

```
file_list <- list.files()
subject_data = file_list[grepl("df_S", file_list, fixed = TRUE)]
```

```
ALL_df = NULL
for (i in 1:length(subject_data)) {
  print(i)
  feature_data = features_all_modalities(subject_data[i], 5, 1) #CHANGE FILE PATH
  subject_no = gsub("\\..*","", sub('.*_', '', subject_data[i]))
  HR_fn = paste0("~/case-study-2/WESAD/", subject_no, "/", subject_no, "_E4_Data/HR.csv") #CHANGE FILEP
  HR_data = read.csv(HR_fn)[-1,]
  hr = HR_calc(as.data.frame(HR_data), 5, 1)

  df_hr = data.frame(matrix(unlist(hr), nrow= length(hr$mu),
                            ncol=4, byrow = F))
  colnames(df_hr) = c("hr_wrist_mu","hr_wrist_sd", "hr_wrist_min", "hr_wrist_max")#, "hr_wrist_range")
  df_hr$ID =seq.int(nrow(df_hr))

  feature_data = feature_data[-c(1:40),]
  feature_data$ID <- seq.int(nrow(feature_data))
```

```
  S_df = merge(feature_data, df_hr, by="ID")
  ALL_df = rbind(ALL_df, S_df)
}
```

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

```
ALL_df = ALL_df %>% filter(Label %in% c("2","3"))
write.csv(ALL_df, "master10000_df.csv")
```

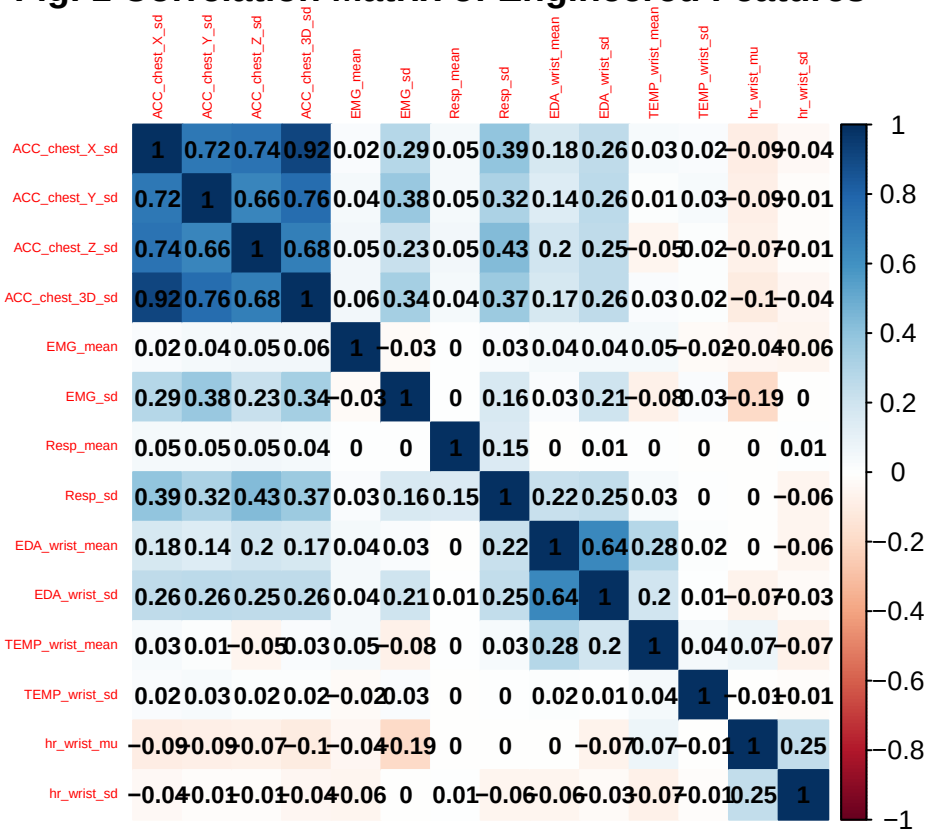
II b. EDA

EDA

```
ALL_df = read.csv("master_df.csv", header = T)[,-c(1:2)]

# get rid of chest EDA and temp and all min max (underdispersion)
remove_select_cov = ALL_df %>%
  dplyr::select(-contains("ACC_wrist")) %>%
  dplyr::select(-EDA_mean, -EDA_sd, -EDA_max, -EDA_min) %>%
  dplyr::select(-Temp_mean, -Temp_sd, -Temp_max, -Temp_min) %>%
  dplyr::select(-contains("BVP")) %>%
  dplyr::select(-contains("ECG")) %>%
  dplyr::select(-contains("min")) %>%
  dplyr::select(-contains("max")) %>%
  dplyr::select(-contains("ACC_chest_X_mean")) %>%
  dplyr::select(-contains("ACC_chest_Y_mean")) %>%
  dplyr::select(-contains("ACC_chest_Z_mean")) %>%
  dplyr::select(-contains("ACC_chest_3D_mean"))

# maybe reorder the variables
col_order <- c("ACC_chest_X_sd", "ACC_chest_Y_sd", "ACC_chest_Z_sd", "ACC_chest_3D_sd", "EMG_mean", "EM
remove_select_cov <- remove_select_cov[, col_order]

remove_select_cov$Subject <- factor(remove_select_cov$Subject, levels = c("S2", "S3","S4", "S5","S6", "

remove_select_cov$Label <- ifelse(remove_select_cov$Label=="2", 1, 0)

non_biological = c("Label", "Subject")
remove_select_cov_nb = remove_select_cov[ , !(names(remove_select_cov) %in% non_biological)]
```

```
cormatrix = cor(remove_select_cov_nb)
corrplot::corrplot(cormatrix, method="color", addCoef.col="black", tl.cex = 0.4, number.cex= 10/ncol(re
```

Fig. 1 Correlation Matrix of Engineered Features



From the correlation matrix (Fig. 1), we see that the `ACC` variables (variables related to motion) are highly correlated. We also see some moderate correlation between `Resp` and `EDA` variables, as well as between `EDA` and `TEMP` variables.

Fig. 2 Distribution of Mean Wrist EDA by State for All Sub

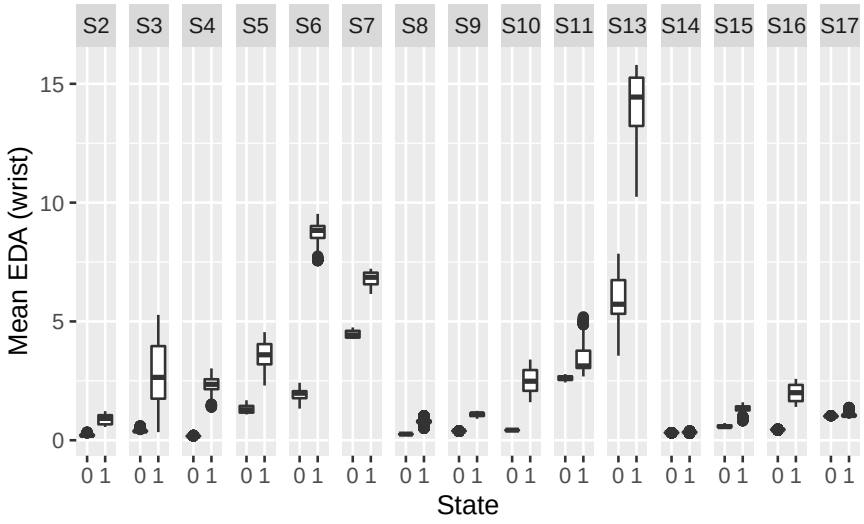


Fig. 3 Distribution of Mean Wrist Temperature by State fo

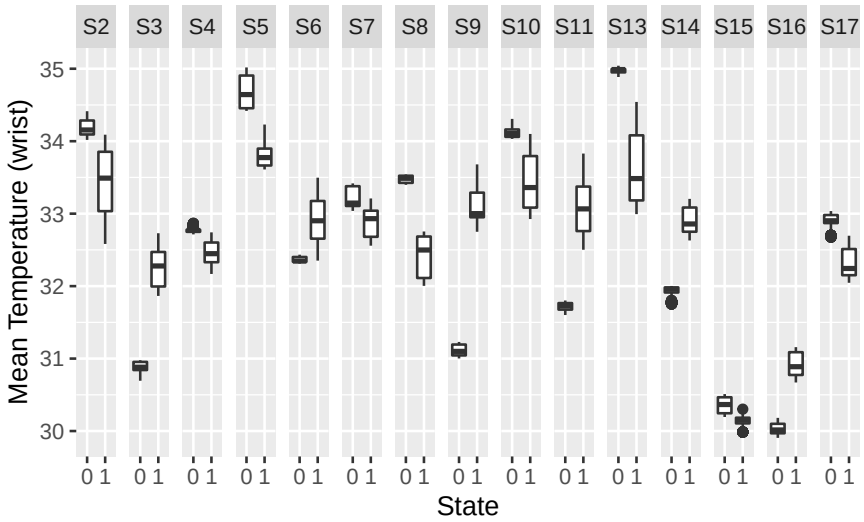
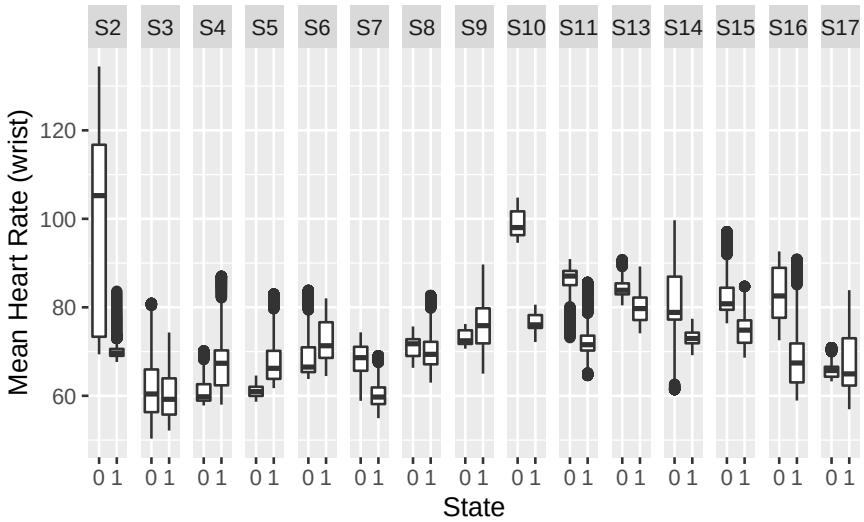


Fig. 4 Distribution of Mean Heart Rate by State for All Su



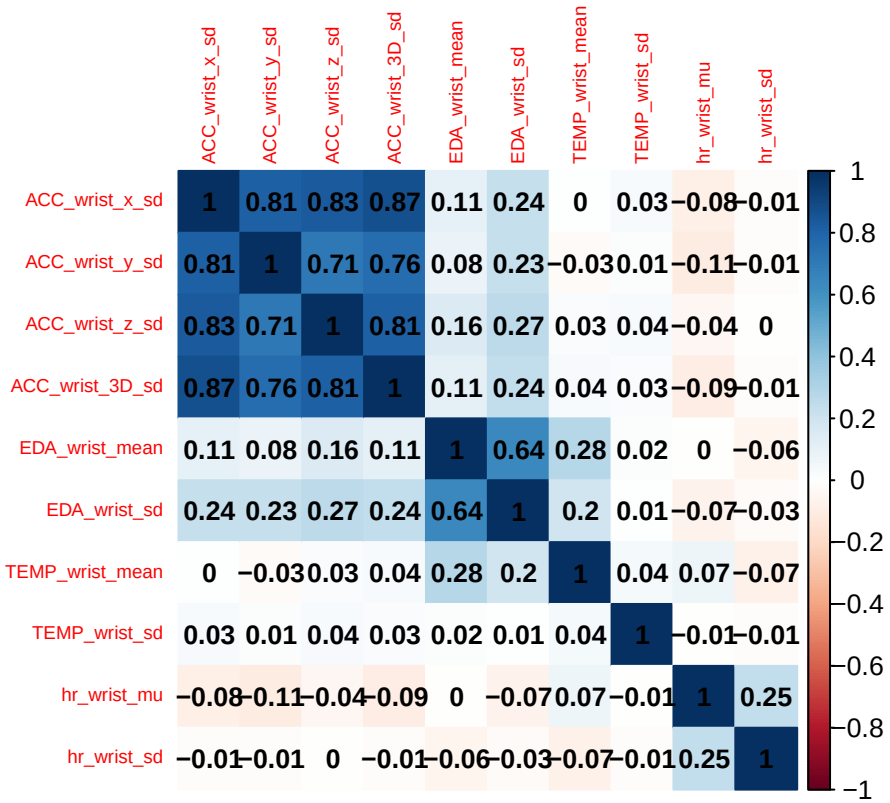
After examining boxplots of the

distribution of the physiological measures by state for each subject (see Appendix A.1 for all plots of this type), the variables that had the greatest difference in distribution between states and across subjects are `EDA_wrist_mean`, `Temp_wrist_mean`, and `hr_wrist_mu`. For all subjects, mean EDA values (Fig. 2) are higher in the stress state, and for subjects 3, 5, 6, 7, 10, 13, the difference in the distribution of EDA between the two states is more drastic then that of the other subjects. For `Temp_wrist_mean` (Fig. 3), we can see that for some subjects, their mean temperature tends to be higher in the stress state but for others, their mean temperatue tends to be lower when stressed. Likewise, for `hr_wrist_mu` (Fig. 4), we see that some subjects have higher average heart rate when stressed while others exhibit the opposite behavior. Among all of the other variables, it appears that the motion (`ACC`) variables tend to show a higher distribtion for each subject under the stress state, and the `EMG` and `Resp` variables tend not to vary much in distribution between states across subjects. In addition, it is important to note that for certain subjects (such as subject 13), there is no overlap in mean EDA values or mean Temperature values between the two states. This is likely to cause perfect separation in logistic regression (see Conclusion for a discussion on potential changes to the experimental design to deal with this issue).

III. Methods

Wrist Data Cleaning

. 5 Correlation Matrix of Engineered Features (Wrist Or



For the wrist only data, we see similar correlation structures as before (with all of the data). Motion variables are strongly correlated with each other, and there is some correlation between `EDA` and `TEMP` variables.

Wrist PCA

```
scaled_wrist_nb = scale(wrist_nb, center = TRUE, scale = TRUE)
res.pca <- PCA(scaled_wrist_nb, graph = FALSE, ncp = 4)
eig.val <- get_eigenvalue(res.pca)
eig.val
```

```
##        eigenvalue variance.percent cumulative.variance.percent
## Dim.1   3.5700222        35.700222                    35.70022
## Dim.2   1.6790330        16.790330                    52.49055
## Dim.3   1.2389191        12.389191                    64.87974
## Dim.4   1.0077038        10.077038                    74.95678
## Dim.5   0.8810815         8.810815                    83.76760
## Dim.6   0.7007898         7.007898                    90.77550
## Dim.7   0.3410337         3.410337                    94.18583
## Dim.8   0.2846091         2.846091                    97.03192
## Dim.9   0.1789510         1.789510                    98.82143
## Dim.10  0.1178566         1.178566                   100.00000
```

```
fviz_eig(res.pca, addlabels = TRUE, ylim = c(0, 50))
```

```
## Registered S3 methods overwritten by 'car':
##   method                          from
##   influence.merMod                lme4
##   cooks.distance.influence.merMod lme4
##   dfbeta.influence.merMod         lme4
##   dfbetas.influence.merMod        lme4
```

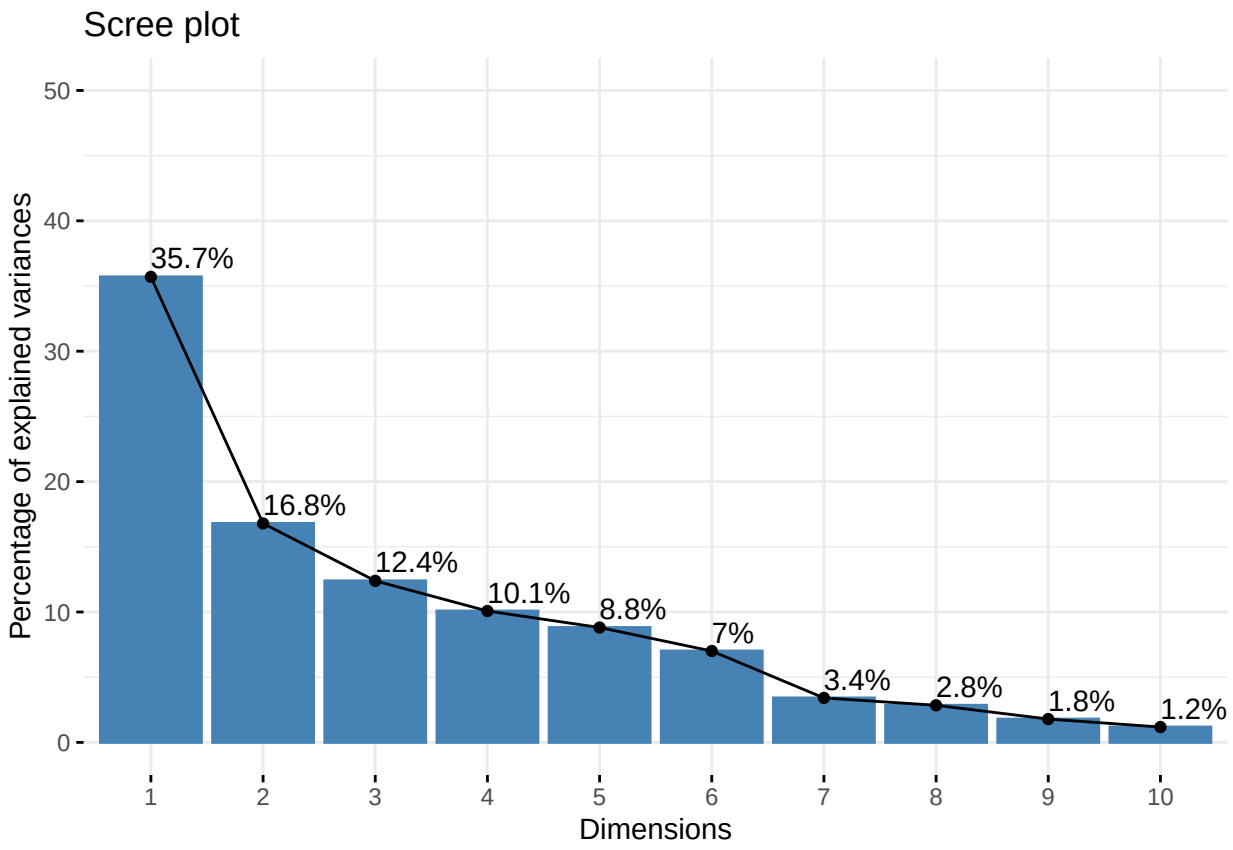
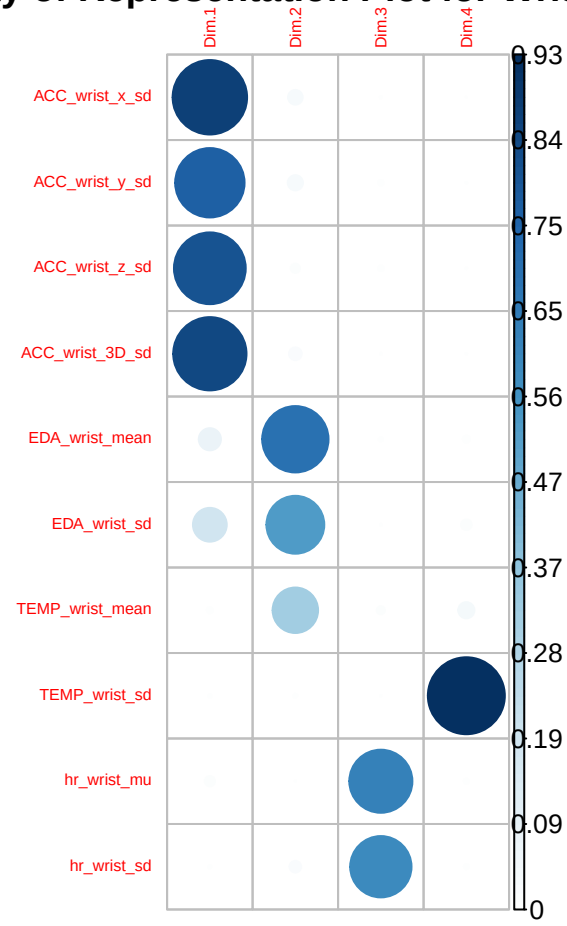
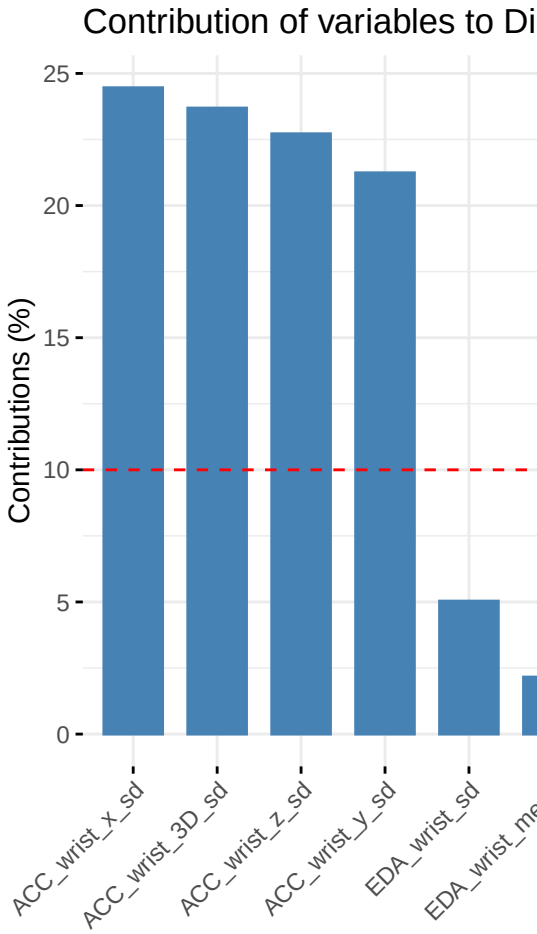
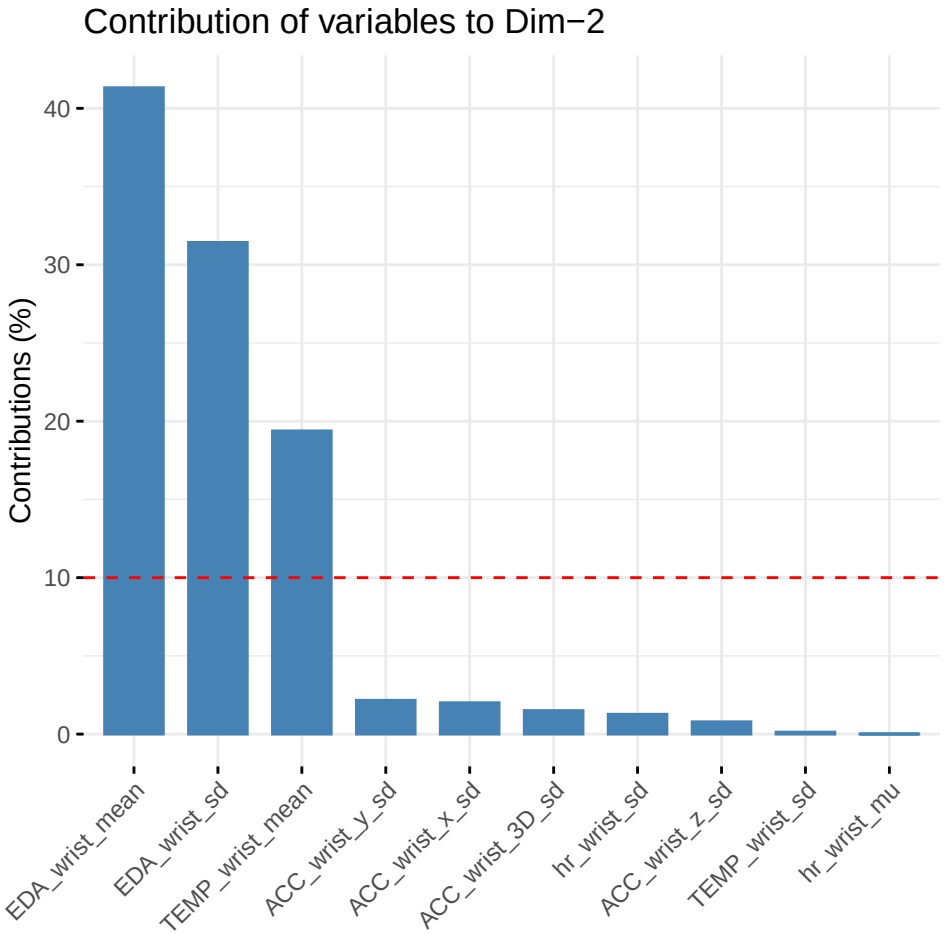
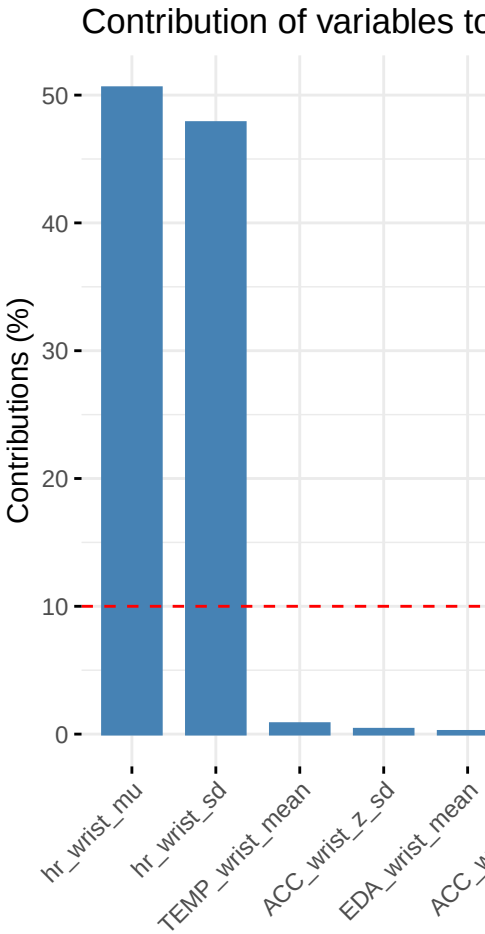


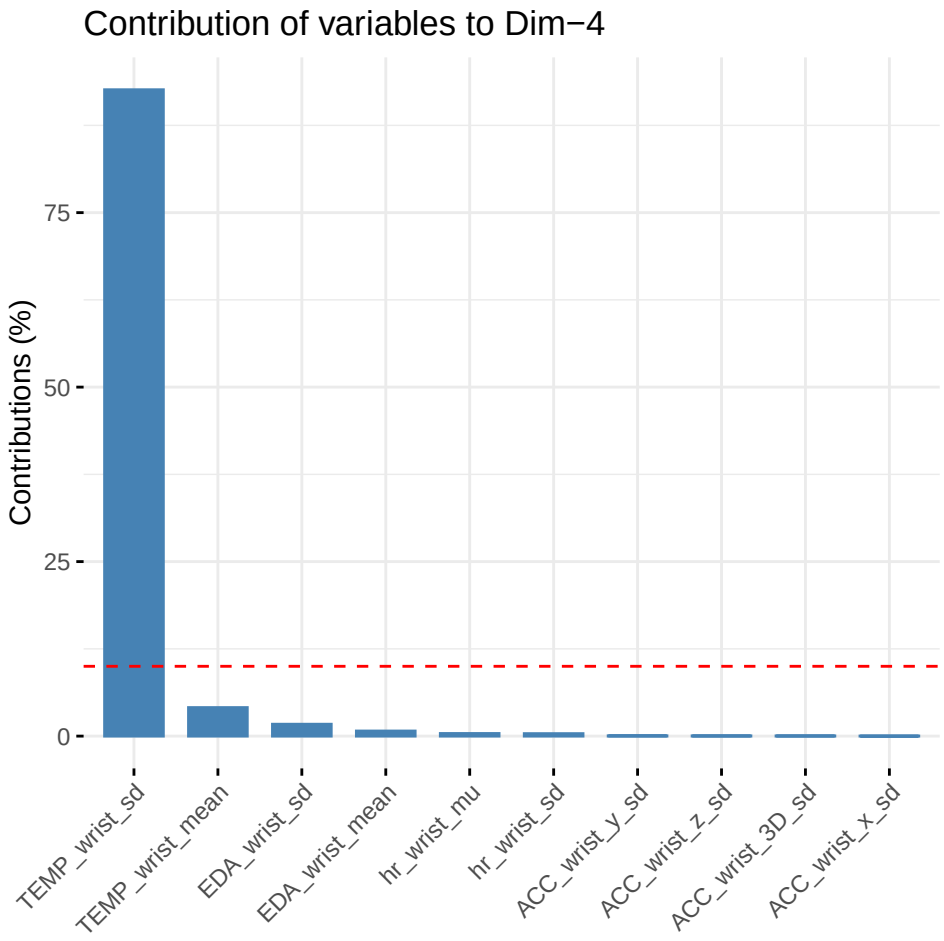
Fig. 6 Quality of Representation Plot for Wrist PCA






Contribution of variables to Dim−2
Contributions (%)
40
30
20
10
0
EDA_wrist_mean
EDA_wrist_sd
TEMP_wrist_mean
ACC_wrist_y_sd
ACC_wrist_x_sd
ACC_wrist_3D_sd
hr_wrist_sd
ACC_wrist_z_sd
TEMP_wrist_sd
hr_wrist_mu


Contribution of variables to
Contributions (%)
50
40
30
20
10
0
hr_wrist_mu
hr_wrist_sd
TEMP_wrist_mean
ACC_wrist_z_sd
EDA_wrist_mean



In the quality of representation plot above (Fig. 6 shows the importance of a principal component for a given observation), we can see that the principal components (with number of components being 4) are about variability in motion, dermal temperature and activity, heart rate, and variability in temperature.

```
wrist_pca_vals <- as.data.frame(res.pca$ind$coord)
wrist_pca_vals$Label = wrist_only$Label
wrist_pca_vals$Subject = wrist_only$Subject
```

Wrist Model

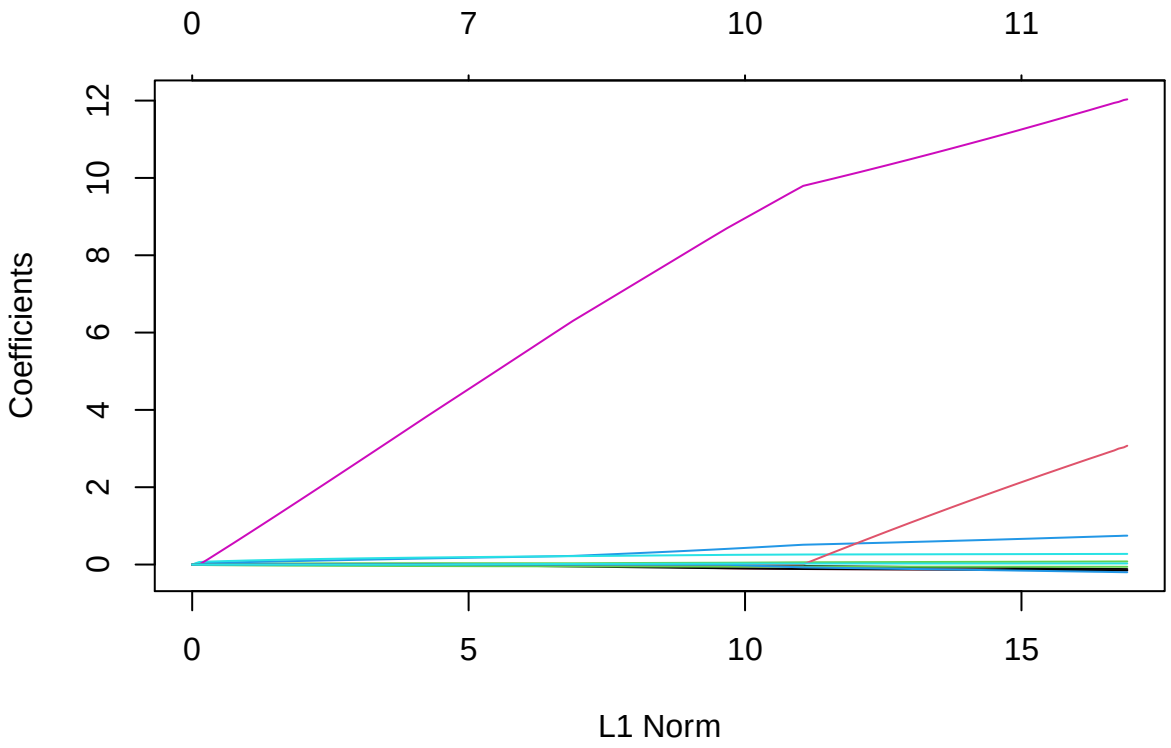
```
wrist_only_nsubj = wrist_only %>% dplyr::select(-Subject, -Label)
scaled_wrist_only = scale(wrist_only_nsubj, center=TRUE, scale = FALSE)
scaled_wrist_only = cbind(scaled_wrist_only, as.data.frame(wrist_only$Subject), as.data.frame(wrist_only
scaled_wrist_only = scaled_wrist_only %>%
  rename(Subject = `wrist_only$Subject`) %>%
  rename(Label = `wrist_only$Label`)

smp_size <- floor(0.75 * nrow(scaled_wrist_only))
train_ind <- sample(seq_len(nrow(scaled_wrist_only)), size = smp_size)
train_wrist <- scaled_wrist_only[train_ind, ]
test_wrist <- scaled_wrist_only[-train_ind, ]

# OUR MODEL FOR WRIST
wrist.full = glm(Label ~. -Subject, family = "binomial", data = train_wrist)
```

```
# summary(wrist.full)
#
# pred.test = wrist.full %>% predict(test_wrist, type= "response")
# pred.test = ifelse(pred.test>0.5, 1,0)
#
# mean(test_wrist$Label==pred.test)
```

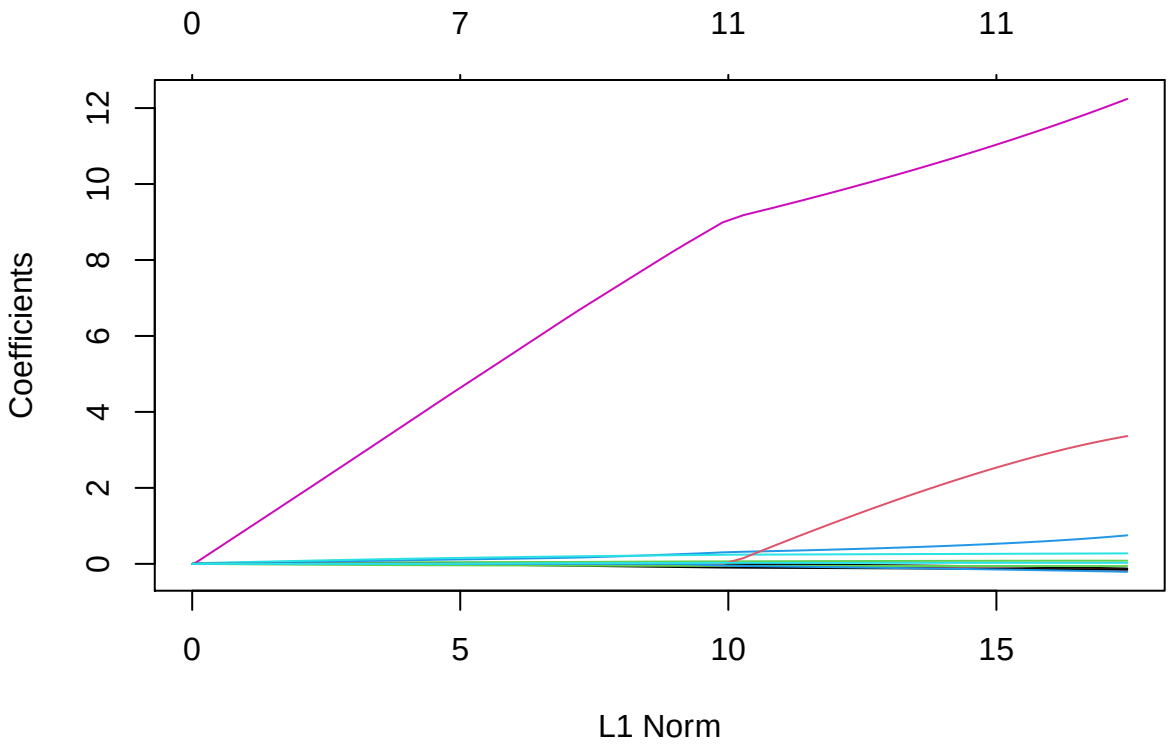
```
# stepwise
step.wrist.model = wrist.full %>% stepAIC(trace=FALSE)
summary(step.wrist.model)
```

```
##
## Call:
## glm(formula = Label ~ ACC_wrist_x_sd + ACC_wrist_y_sd + ACC_wrist_z_sd +
##     ACC_wrist_3D_sd + EDA_wrist_mean + EDA_wrist_sd + TEMP_wrist_mean +
##     hr_wrist_mu + hr_wrist_sd, family = "binomial", data = train_wrist)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.7400  -0.9264   0.1459   0.8409   2.0653
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)      1.667489   0.026424   63.10  < 2e-16 ***
## ACC_wrist_x_sd  -0.137729   0.008103  -17.00  < 2e-16 ***
## ACC_wrist_y_sd  -0.161818   0.006344  -25.51  < 2e-16 ***
## ACC_wrist_z_sd   0.087019   0.006000   14.50  < 2e-16 ***
## ACC_wrist_3D_sd  0.793297   0.021369   37.12  < 2e-16 ***
## EDA_wrist_mean   0.284959   0.008532   33.40  < 2e-16 ***
## EDA_wrist_sd    11.659561   0.609005   19.14  < 2e-16 ***
## TEMP_wrist_mean -0.190817   0.009584  -19.91  < 2e-16 ***
## hr_wrist_mu     -0.053886   0.001332  -40.46  < 2e-16 ***
## hr_wrist_sd     -0.255107   0.061035   -4.18 2.92e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 60795  on 46622  degrees of freedom
## Residual deviance: 43312  on 46613  degrees of freedom
## AIC: 43332
##
## Number of Fisher Scoring iterations: 7
```

```
#convert training data to matrix format
train_wrist_x = train_wrist %>%
  dplyr::select(-Label) %>%
  mutate(Subject = as.factor(Subject))

x <- data.matrix(train_wrist_x)
y <- train_wrist$Label
lasso.fit = glmnet(x, y, alpha = 1, family = "binomial")
plot(lasso.fit)
```




```
elasticnet.fit = glmnet(x, y, alpha = 0.5, family = "binomial")
plot(elasticnet.fit)
```



Wrist K-Fold CV

```
# set.seed(123)
#
# kfoldcv_wrist_full = function(df) {
#
#   actual_values = c()
#   predicted_values = c()
#   fitted_probs = c()
#
#   # fold_accuracies = c()
#   # fold_f1 = c()
#
#   fold_no = c()
#   subject = c()
#
#   print("at folds")
#   folds <- createFolds(factor(df$Subject), k = 5)
#
#   print("past folds")
#   for (i in 1:5) {
#   #train and test
#     col = paste0("Fold", i)
#     indx = unlist(folds[col])
#
#   train<- df[-indx,]
#   print(nrow(train))
#   test<- df[indx, ]
#   print(nrow(test))
#
#
#   # run model
#   wrist.full = glm(Label ~. -Subject, family = "binomial", data = train)
# #summary(wrist.full)
#
# pred.test = wrist.full %>% predict(test, type= "response")
# fitted_probs = c(fitted_probs, pred.test)
# pred.test = ifelse(pred.test>0.5, 1,0)
# predicted_values = c(predicted_values, pred.test)
# actual_values = c(actual_values, test$Label)
#
#
#   subject = c(subject, test$Subject)
#   fold_no = c(fold_no, rep(i, nrow(test)))
#
#
#
#   #acc1 = mean(test1$Label==pred.test)
#   #fold_accuracies = c(fold_accuracies, acc1)
#
#   # add function to compute f1
#   # blah
# }
```

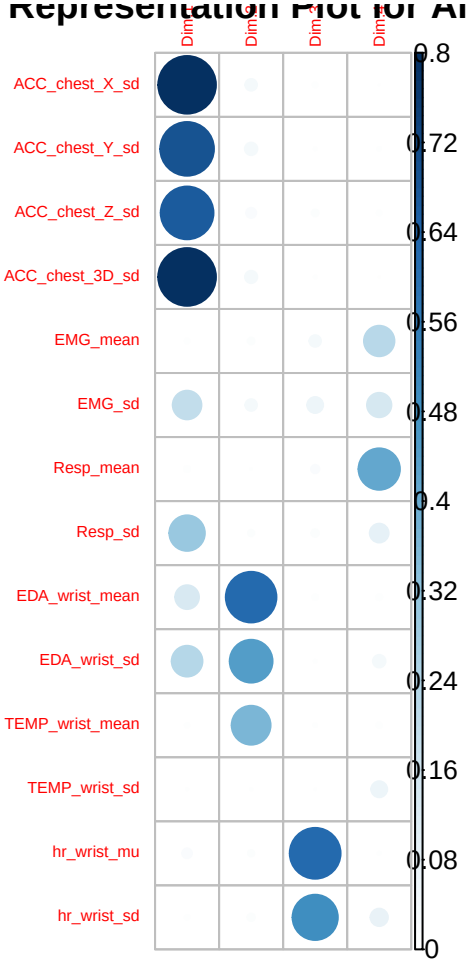
```
#   #train on training set
#   #test on testing set
#   #predict labels
#   #find accuracy and f1, add to fold_accuracies and fold_f1
#   #put predicted values in predicted_values
#   #put actual values in actual_values
#
#   cvoutput = data.frame(cbind(actual_values, predicted_values, fitted_probs,
#         fold_no, subject))
#
#   return(cvoutput)
# }
#
# wristcvoutput = kfoldcv_wrist_full(scaled_wrist_only)
```

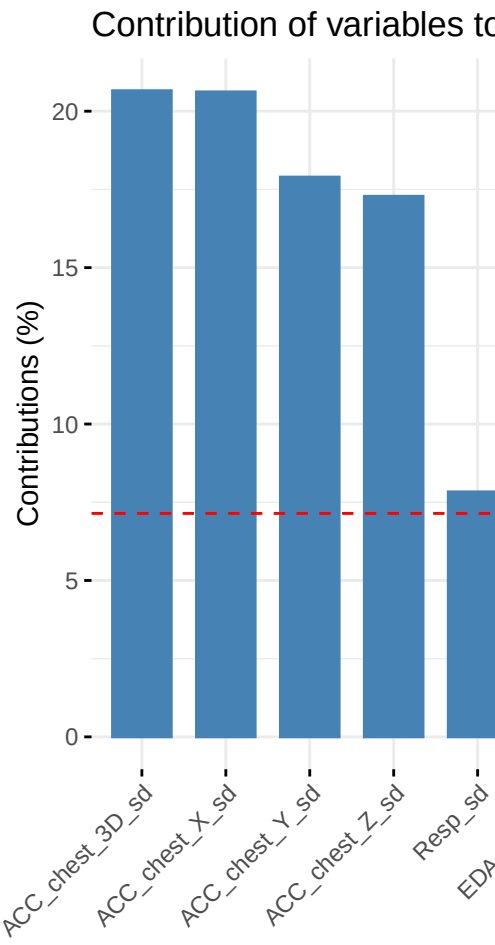
```
# fold1 = wristcvoutput %>%
#   filter(subject=="S11")
#
# mean(fold1$actual_values==fold1$predicted_value)
```

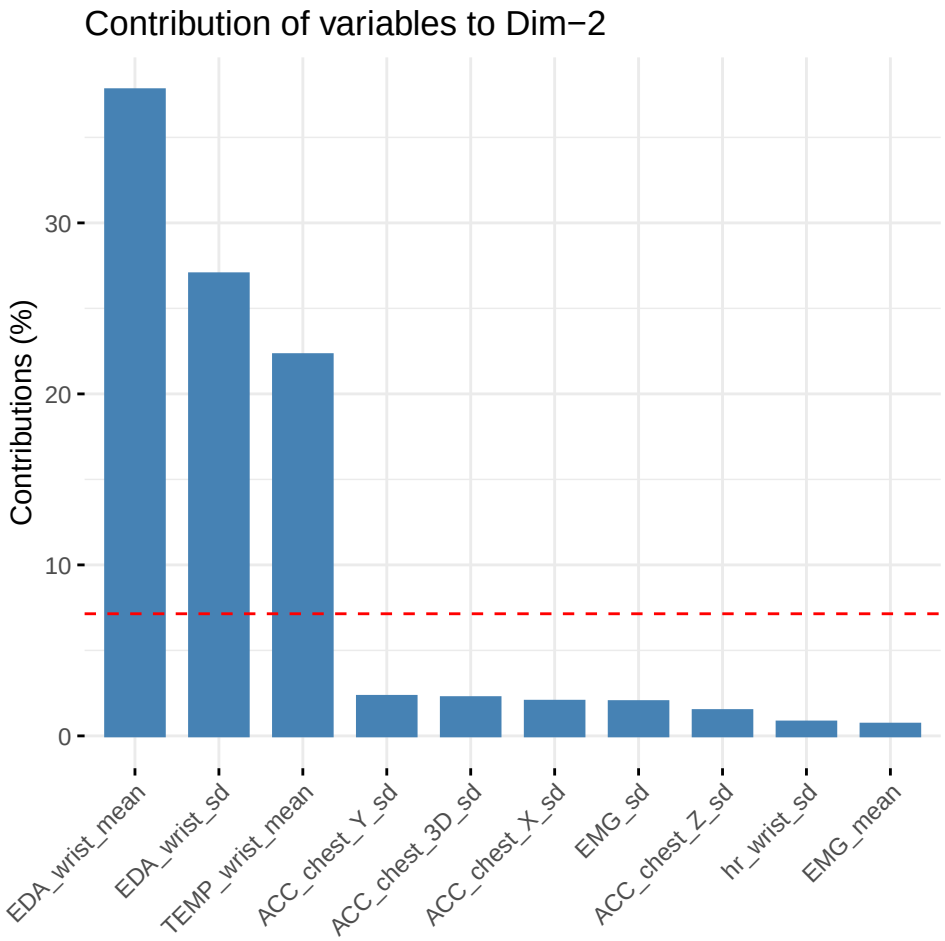
Combo PCA

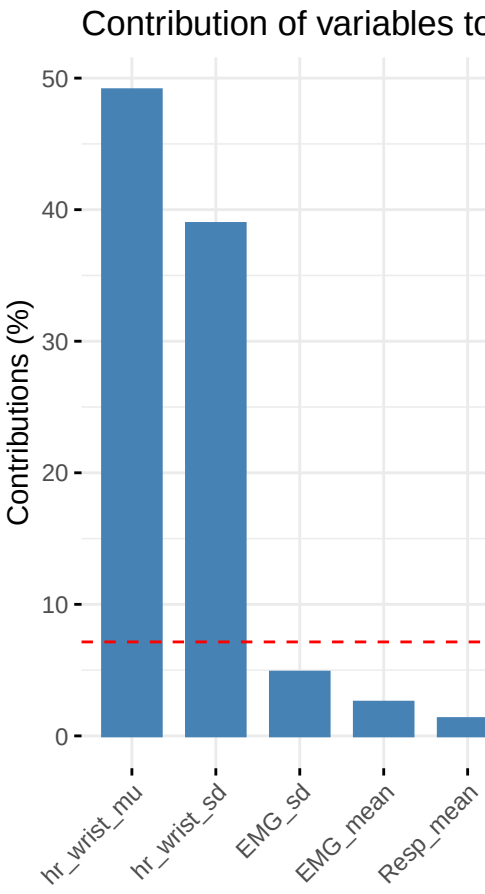
```
scaled_select_cov_nb = scale(remove_select_cov_nb, center = TRUE, scale = TRUE)

res.pca <- PCA(scaled_select_cov_nb, graph = FALSE, ncp = 4)
eig.val <- get_eigenvalue(res.pca)
#eig.val
#fviz_eig(res.pca, addlabels = TRUE, ylim = c(0, 50))
```

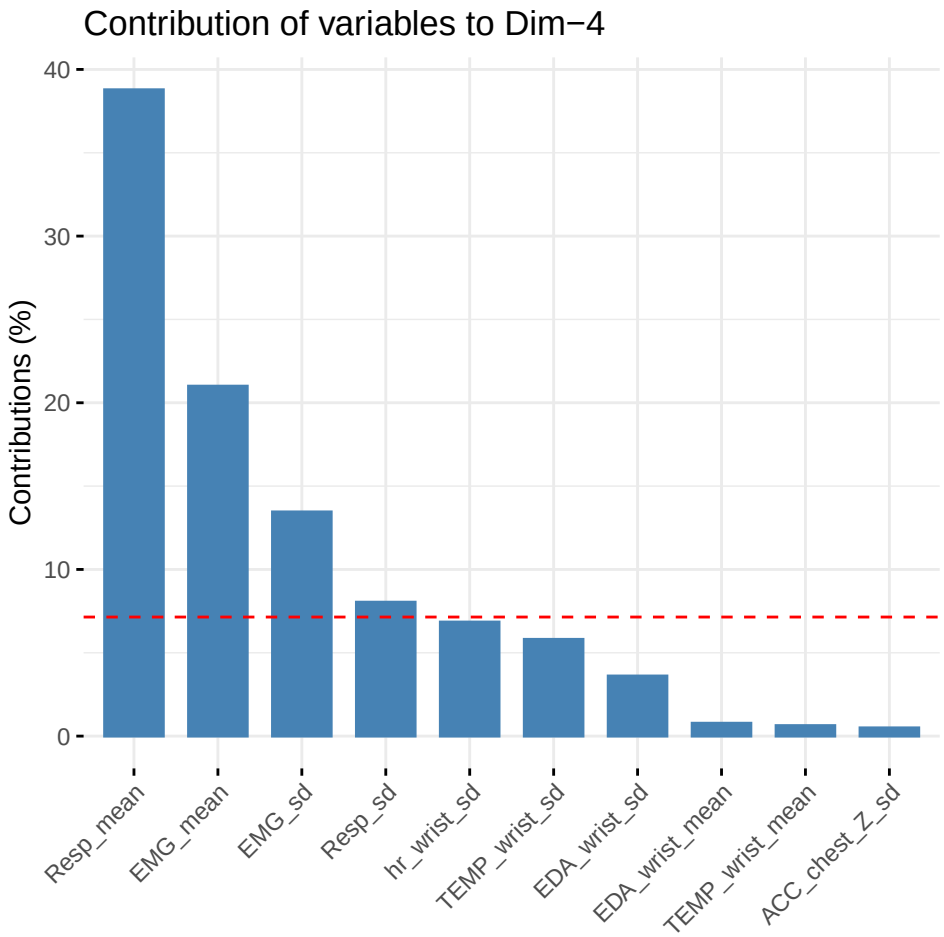
Fig. 7 Quality of Representation Plot for All Data PCA






Contribution of variables to Dim−2
Contributions (%)
0
10
20
30
EDA_wrist_mean
EDA_wrist_sd
TEMP_wrist_mean
ACC_chest_Y_sd
ACC_chest_3D_sd
ACC_chest_X_sd
EMG_sd
ACC_chest_Z_sd
hr_wrist_sd
EMG_mean


Contribution of variables to
Contributions (%)
0
10
20
30
40
50
hr_wrist_mu
hr_wrist_sd
EMG_sd
EMG_mean
Resp_mean



From the plot above (Fig. 7), we see that the principal components are variability in movement (Dim.1), dermal temperature and activity (Dim.2), heart rate (Dim.3), and Neuro-muscular acitivity (Dim.4).

```
pca_vals <- as.data.frame(res.pca$ind$coord)
pca_vals$Label = remove_select_cov$Label
pca_vals$Subject = remove_select_cov$Subject
```

Combo Model

```
smp_size <- floor(0.75 * nrow(pca_vals))
train_ind <- sample(seq_len(nrow(pca_vals)), size = smp_size)
train_pca <- pca_vals[train_ind, ]
test_pca <- pca_vals[-train_ind, ]

# COMBO PCA MODEL model without Subject  INTERPRET MAIN EFFECTS
pca.model = glm(Label ~ .-Subject, family = binomial(link="logit"), data=train_pca)

# summary(pca.model)
#
# pred.test = pca.model %>% predict(test_pca, type= "response")
# pred.test = ifelse(pred.test>0.5, 1,0)
#
# mean(test_pca$Label==pred.test)
```

Table 1: Estimate and Confidence Intervals for Coefficients in the All Data Model

	Estimate	SE	2.5%	97.5%
(Intercept)	1.10	0.01	1.07	1.13
Dim.1	1.00	0.01	0.97	1.02
Dim.2	0.32	0.01	0.30	0.35
Dim.3	-0.35	0.01	-0.38	-0.33
Dim.4	-0.28	0.01	-0.31	-0.25

```
est = round(summary(pca.model)$coefficients[,1],2)
SE = round(summary(pca.model)$coefficients[,2],2)
ci = round(confint(pca.model),2)
```

```
## Waiting for profiling to be done...
```

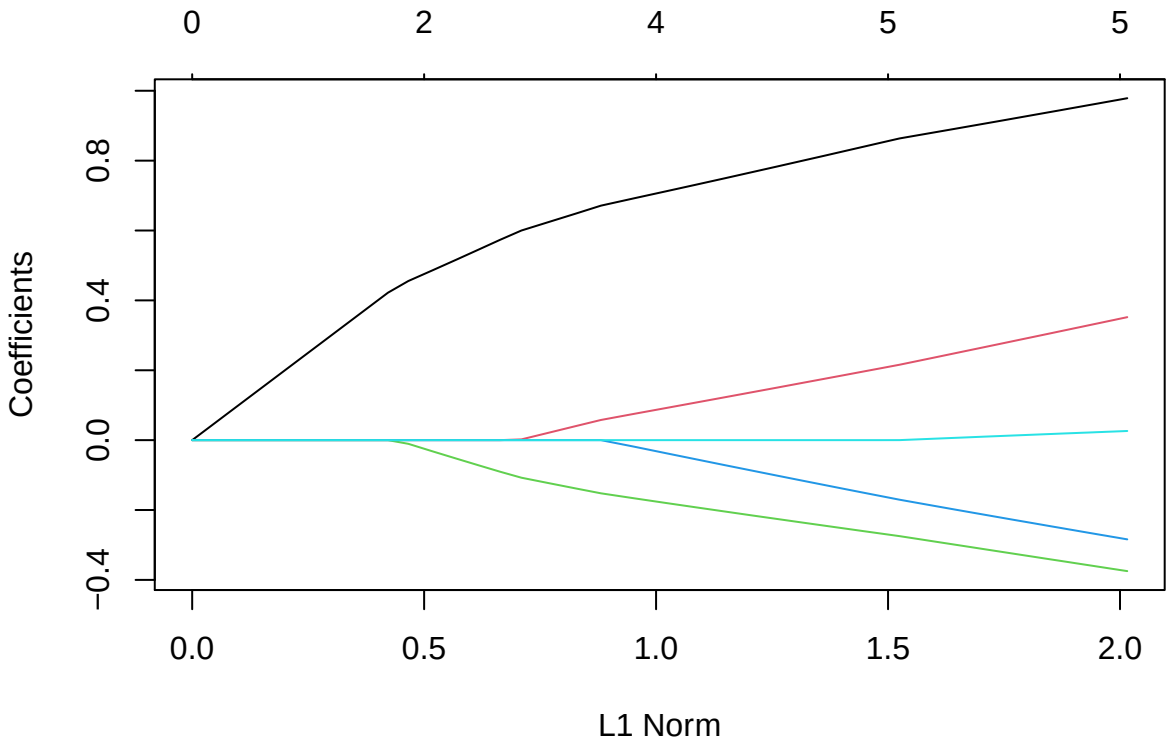
```
combo_model_tab = data.frame(cbind(est,SE,ci))
colnames(combo_model_tab) = c("Estimate", "SE", "2.5%", "97.5%")
kable(combo_model_tab, caption = "Estimate and Confidence Intervals for Coefficients in the All Data Mo
```

A increase of one in the standard deviation for movement and dermal temperature and activity is associated with a multiplicative increase in the odds that the state is stress. A increase of one in the standard deviation of heart rate and neuro-muscular activity is associated with a multiplicative decrease in the odds that the state is stress.

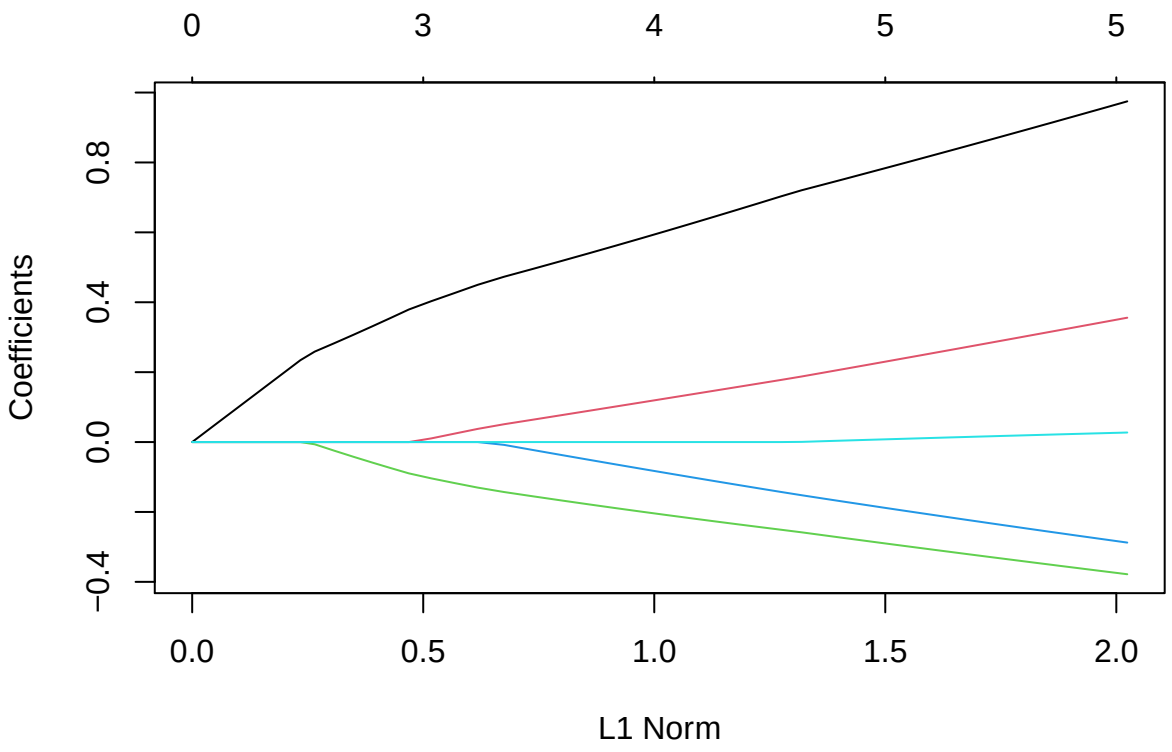
```
# stepwise
step.wrist.model = pca.model %>% stepAIC(trace=FALSE)
summary(step.wrist.model)
```

```
##
## Call:
## glm(formula = Label ~ (Dim.1 + Dim.2 + Dim.3 + Dim.4 + Subject) -
##     Subject, family = binomial(link = "logit"), data = train_pca)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.7853  -0.8520   0.3303   0.8058   1.9831
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.10143    0.01490   73.90   <2e-16 ***
## Dim.1        0.99741    0.01178   84.68   <2e-16 ***
## Dim.2        0.32336    0.01273   25.41   <2e-16 ***
## Dim.3       -0.35209    0.01196  -29.43   <2e-16 ***
## Dim.4       -0.27894    0.01349  -20.67   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 60795  on 46622  degrees of freedom
## Residual deviance: 45272  on 46618  degrees of freedom
## AIC: 45282
##
## Number of Fisher Scoring iterations: 6
```

```r
#convert training data to matrix format
train_pca_x = train_pca %>%
  dplyr::select(-Label) %>%
  mutate(Subject = as.factor(Subject))

x <- data.matrix(train_pca_x)
y <- train_pca$Label
lasso.fit = glmnet(x, y, alpha = 1, family = "binomial")
plot(lasso.fit)
```




```r
elasticnet.fit = glmnet(x, y, alpha = 0.5, family = "binomial")
plot(elasticnet.fit)
```



Combo K-Fold CV

```
set.seed(123)

kfoldcv_no_subject = function(df) {

  actual_values = c()
  predicted_values = c()
  fitted_probs = c()

  # fold_accuracies = c()
  # fold_f1 = c()

  fold_no = c()
  subject = c()

  #print("at folds")
  folds <- createFolds(factor(df$Subject), k = 5)

  #print("past folds")
  for (i in 1:5) {
  #train and test
    col = paste0("Fold", i)
    indx = unlist(folds[col])

  train<- df[-indx,]
  #print(nrow(train))
  test<- df[indx, ]
  #print(nrow(test))
```

```
  # run model
pca.model = glm(Label ~ .-Subject, family = binomial(link="logit"), data=train)
#summary(wrist.full)

pred.test = pca.model %>% predict(test, type= "response")
fitted_probs = c(fitted_probs, pred.test)
pred.test = ifelse(pred.test>0.5, 1,0)
predicted_values = c(predicted_values, pred.test)
actual_values = c(actual_values, test$Label)


  subject = c(subject, test$Subject)
  fold_no = c(fold_no, rep(i, nrow(test)))



  #acc1 = mean(test1$Label==pred.test)
  #fold_accuracies = c(fold_accuracies, acc1)

  # add function to compute f1
  # blah
}
  #train on training set
  #test on testing set
  #predict labels
  #find accuracy and f1, add to fold_accuracies and fold_f1
  #put predicted values in predicted_values
  #put actual values in actual_values

  cvoutput = data.frame(cbind(actual_values, predicted_values, fitted_probs,
        fold_no, subject))

  return(cvoutput)
}

combocvoutput = kfoldcv_no_subject(pca_vals)
wristcvoutput = kfoldcv_no_subject(wrist_pca_vals)
```

```
combo_acc_per_fold = combocvoutput %>%
  group_by(fold_no) %>%
  summarize(accuracy = mean(actual_values==predicted_values))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
combo_f1_per_fold = combocvoutput %>%
  group_by(fold_no) %>%
  summarize(f1 = F1_Score(actual_values, predicted_values, positive = NULL))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
wrist_acc_per_fold = wristcvoutput %>%
  group_by(fold_no) %>%
  summarize(accuracy = mean(actual_values==predicted_values))
```

Table 2: Accuracy and F1 for All Data and Wrist Models

	All	Wrist
Accuracy	0.81	0.74
F1	0.73	0.59

Table 3: Confusion Matrices for All Data (left) and Wrist (right) Models

	Actual Amusement	Actual Stress
Predicted Amusement	15948	5238
Predicted Stress	6352	34626

	Actual Amusement	Actual Stress
Predicted Amusement	11608	5605
Predicted Stress	10692	34259

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
wrist_f1_per_fold = wristcvoutput %>%
  group_by(fold_no) %>%
  summarize(f1 = F1_Score(actual_values, predicted_values, positive = NULL))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
# REWMEMEVR TO ROUDN
acc_f1_data <- matrix(c(round(mean(combo_acc_per_fold$accuracy),2),
                  round(mean(wrist_acc_per_fold$accuracy),2),
                  round(mean(combo_f1_per_fold$f1),2),
                  round(mean(wrist_f1_per_fold$f1),2)),ncol=2,byrow=TRUE)
colnames(acc_f1_data) <- c("All","Wrist")
rownames(acc_f1_data) <- c("Accuracy","F1")
acc_f1_tab <- as.table(acc_f1_data)
kable(acc_f1_tab, caption = "Accuracy and F1 for All Data and Wrist Models")
```

```
# add row and col names or get rid of kable
combo_cm = as.table(confusionMatrix(as.factor(combocvoutput$predicted_values), as.factor(combocvoutput$a
colnames(combo_cm) = c("Actual Amusement", "Actual Stress")
rownames(combo_cm) = c("Predicted Amusement", "Predicted Stress")
#kable(combo_cm, caption = "Confusion Matrix for All Data Model")

wrist_cm = as.table(confusionMatrix(as.factor(wristcvoutput$predicted_values), as.factor(wristcvoutput$a
colnames(wrist_cm) = c("Actual Amusement", "Actual Stress")
rownames(wrist_cm) = c("Predicted Amusement", "Predicted Stress")
#kable(wrist_cm, caption = "Confusion Matrix for Wrist Data Model")

# kable(wrist_cm) %>%
#   kable_styling(full_width = FALSE, position = "float_left",
#                 title = "Confusion Matrix for All Data Model")
# kable(combo_cm) %>%
#   kable_styling(full_width = FALSE, position = "left",
#                 title = "Confusion Matrix for Wrist Data Model")

knitr::kable(list(combo_cm, wrist_cm), caption = "Confusion Matrices for All Data (left) and Wrist (rig
```

In Table 1 (and in the confusion matrices Table 2), we see the average accuracy and F1 results of 5-folds

cross validation for a model using only wrist data and a model using both chest and wrist data. The model using both types of sensor data has higher accuracy and F1.

IV. Results

Final Model

```
# HOPEFULLY THE RIGHT ONE model with no main effect for Subject but with interactions
pca.model.interact = glm(Label ~ Dim.1+Dim.2+Dim.3+Dim.4+Dim.1:Subject
                  +Dim.2:Subject
                  +Dim.3:Subject
                  +Dim.4:Subject
                 , family = binomial(link="logit"), data=pca_vals)

summary(pca.model.interact)
```

```
##
## Call:
## glm(formula = Label ~ Dim.1 + Dim.2 + Dim.3 + Dim.4 + Dim.1:Subject +
##     Dim.2:Subject + Dim.3:Subject + Dim.4:Subject, family = binomial(link = "logit"),
##     data = pca_vals)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.6026  -0.2067   0.0238   0.4043   3.1446
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        2.75764    0.04035  68.345  < 2e-16 ***
## Dim.1              0.31351    0.08606   3.643 0.000270 ***
## Dim.2             -8.43530    0.41164 -20.492  < 2e-16 ***
## Dim.3             -1.60165    0.07041 -22.746  < 2e-16 ***
## Dim.4              0.40475    0.08485   4.770 1.84e-06 ***
## Dim.1:SubjectS3    1.27153    0.09933  12.801  < 2e-16 ***
## Dim.1:SubjectS4    1.74004    0.09940  17.506  < 2e-16 ***
## Dim.1:SubjectS5    4.43791    0.20646  21.495  < 2e-16 ***
## Dim.1:SubjectS6    5.26245    0.36695  14.341  < 2e-16 ***
## Dim.1:SubjectS7    5.00461    0.22440  22.302  < 2e-16 ***
## Dim.1:SubjectS8    1.99914    0.09962  20.069  < 2e-16 ***
## Dim.1:SubjectS9    0.61708    0.09100   6.781 1.19e-11 ***
## Dim.1:SubjectS10   1.02093    0.10497   9.726  < 2e-16 ***
## Dim.1:SubjectS11   3.25439    0.20597  15.801  < 2e-16 ***
## Dim.1:SubjectS13   0.18959    0.08811   2.152 0.031420 *
## Dim.1:SubjectS14   1.43407    0.09598  14.942  < 2e-16 ***
## Dim.1:SubjectS15   1.09561    0.09425  11.624  < 2e-16 ***
## Dim.1:SubjectS16   2.03518    0.11104  18.328  < 2e-16 ***
## Dim.1:SubjectS17   0.29241    0.09340   3.131 0.001745 **
## Dim.2:SubjectS3   12.02063    0.41988  28.629  < 2e-16 ***
## Dim.2:SubjectS4   12.58803    0.43595  28.875  < 2e-16 ***
## Dim.2:SubjectS5   16.02306    0.60414  26.522  < 2e-16 ***
## Dim.2:SubjectS6   33.04259    1.70286  19.404  < 2e-16 ***
## Dim.2:SubjectS7    6.56933    0.46842  14.025  < 2e-16 ***
## Dim.2:SubjectS8    8.62471    0.43988  19.607  < 2e-16 ***
```

```
## Dim.2:SubjectS9  12.13523    0.42372  28.640  < 2e-16 ***
## Dim.2:SubjectS10 10.85709    0.45676  23.770  < 2e-16 ***
## Dim.2:SubjectS11 20.90306    0.69993  29.865  < 2e-16 ***
## Dim.2:SubjectS13  8.30807    0.41217  20.157  < 2e-16 ***
## Dim.2:SubjectS14 10.48645    0.42848  24.473  < 2e-16 ***
## Dim.2:SubjectS15  8.94947    0.41727  21.448  < 2e-16 ***
## Dim.2:SubjectS16  9.48707    0.41735  22.732  < 2e-16 ***
## Dim.2:SubjectS17  9.70360    0.42230  22.978  < 2e-16 ***
## Dim.3:SubjectS3   0.85803    0.10300   8.331  < 2e-16 ***
## Dim.3:SubjectS4   1.56018    0.09992  15.615  < 2e-16 ***
## Dim.3:SubjectS5   6.08450    0.28912  21.045  < 2e-16 ***
## Dim.3:SubjectS6   3.30071    0.32375  10.195  < 2e-16 ***
## Dim.3:SubjectS7  -0.67064    0.23745  -2.824 0.004738 **
## Dim.3:SubjectS8   1.31896    0.13318   9.904  < 2e-16 ***
## Dim.3:SubjectS9   2.82131    0.11328  24.906  < 2e-16 ***
## Dim.3:SubjectS10 -1.41101    0.10794 -13.072  < 2e-16 ***
## Dim.3:SubjectS11  0.61919    0.18121   3.417 0.000633 ***
## Dim.3:SubjectS13 -0.91241    0.10440  -8.740  < 2e-16 ***
## Dim.3:SubjectS14 -0.97794    0.15333  -6.378 1.79e-10 ***
## Dim.3:SubjectS15  0.30891    0.10075   3.066 0.002169 **
## Dim.3:SubjectS16  1.42710    0.09288  15.365  < 2e-16 ***
## Dim.3:SubjectS17  3.58877    0.10381  34.572  < 2e-16 ***
## Dim.4:SubjectS3  -1.58708    0.10145 -15.644  < 2e-16 ***
## Dim.4:SubjectS4  -1.83509    0.12602 -14.562  < 2e-16 ***
## Dim.4:SubjectS5  -2.03910    0.16427 -12.413  < 2e-16 ***
## Dim.4:SubjectS6   0.01843    0.43488   0.042 0.966200
## Dim.4:SubjectS7   1.92784    0.26413   7.299 2.90e-13 ***
## Dim.4:SubjectS8  -2.03017    0.11333 -17.914  < 2e-16 ***
## Dim.4:SubjectS9  -0.60533    0.09967  -6.073 1.25e-09 ***
## Dim.4:SubjectS10 -0.10198    0.11314  -0.901 0.367421
## Dim.4:SubjectS11 -1.79669    0.17621 -10.196  < 2e-16 ***
## Dim.4:SubjectS13 -0.92594    0.09385  -9.866  < 2e-16 ***
## Dim.4:SubjectS14 -0.09012    0.10881  -0.828 0.407515
## Dim.4:SubjectS15 -0.45976    0.09844  -4.670 3.01e-06 ***
## Dim.4:SubjectS16 -1.72068    0.12083 -14.240  < 2e-16 ***
## Dim.4:SubjectS17 -0.50899    0.09819  -5.184 2.18e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 81147  on 62163  degrees of freedom
## Residual deviance: 31344  on 62103  degrees of freedom
## AIC: 31466
##
## Number of Fisher Scoring iterations: 12
```

```
pred.test = pca.model.interact %>% predict(pca_vals, type= "response")
pca_vals$fitted = pred.test
pred.test = ifelse(pred.test>0.5, 1,0)
pca_vals$predicted = pred.test

#
# mean(test_pca$Label==pred.test)
```

```
#
# F1_Score(test_pca$Label, pred.test, positive = NULL)
# confusionMatrix(as.factor(pred.test), as.factor(test_pca$Label))
```

```
set.seed(123)

kfoldcv_final = function(df) {

  actual_values = c()
  predicted_values = c()
  fitted_probs = c()

  # fold_accuracies = c()
  # fold_f1 = c()

  fold_no = c()
  subject = c()

  print("at folds")
  folds <- createFolds(factor(df$Subject), k = 5)

  print("past folds")
  for (i in 1:5) {
  #train and test
    col = paste0("Fold", i)
    indx = unlist(folds[col])

  train<- df[-indx,]
  print(nrow(train))
  test<- df[indx, ]
  print(nrow(test))


  # run model
pca.model.interact = glm(Label ~ Dim.1+Dim.2+Dim.3+Dim.4+Dim.1:Subject
                  +Dim.2:Subject
                  +Dim.3:Subject
                  +Dim.4:Subject
                 , family = binomial(link="logit"), data=train)
#summary(wrist.full)

pred.test = pca.model.interact %>% predict(test, type= "response")
fitted_probs = c(fitted_probs, pred.test)
pred.test = ifelse(pred.test>0.5, 1,0)
predicted_values = c(predicted_values, pred.test)
actual_values = c(actual_values, test$Label)


  subject = c(subject, test$Subject)
  fold_no = c(fold_no, rep(i, nrow(test)))



  #acc1 = mean(test1$Label==pred.test)
```

```
  #fold_accuracies = c(fold_accuracies, acc1)

  # add function to compute f1
  # blah
}
  #train on training set
  #test on testing set
  #predict labels
  #find accuracy and f1, add to fold_accuracies and fold_f1
  #put predicted values in predicted_values
  #put actual values in actual_values

  cvoutput = data.frame(cbind(actual_values, predicted_values, fitted_probs,
        fold_no, subject))

  return(cvoutput)
}

finalcvoutput = kfoldcv_final(pca_vals)
```

```
## [1] "at folds"
## [1] "past folds"
## [1] 49729
## [1] 12435
## [1] 49732
## [1] 12432
## [1] 49732
## [1] 12432
## [1] 49731
## [1] 12433
## [1] 49732
## [1] 12432
```

```
#MALAVI ADD FOR SUBJECT

final_acc_per_fold = finalcvoutput %>%
  group_by(fold_no) %>%
  summarize(accuracy = mean(actual_values==predicted_values))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
final_f1_per_fold = finalcvoutput %>%
  group_by(fold_no) %>%
  summarize(f1 = F1_Score(actual_values, predicted_values, positive = NULL))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
final_acc_per_subj = finalcvoutput %>%
  group_by(subject) %>%
  summarize(accuracy = mean(actual_values==predicted_values))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
final_f1_per_subj = finalcvoutput %>%
  group_by(subject) %>%
  summarize(f1 = F1_Score(actual_values, predicted_values, positive = NULL))
```

Table 4: Confusion Matrix for Final Model

	Actual Amusement	Actual Stress
Predicted Amusement	11608	5605
Predicted Stress	10692	34259

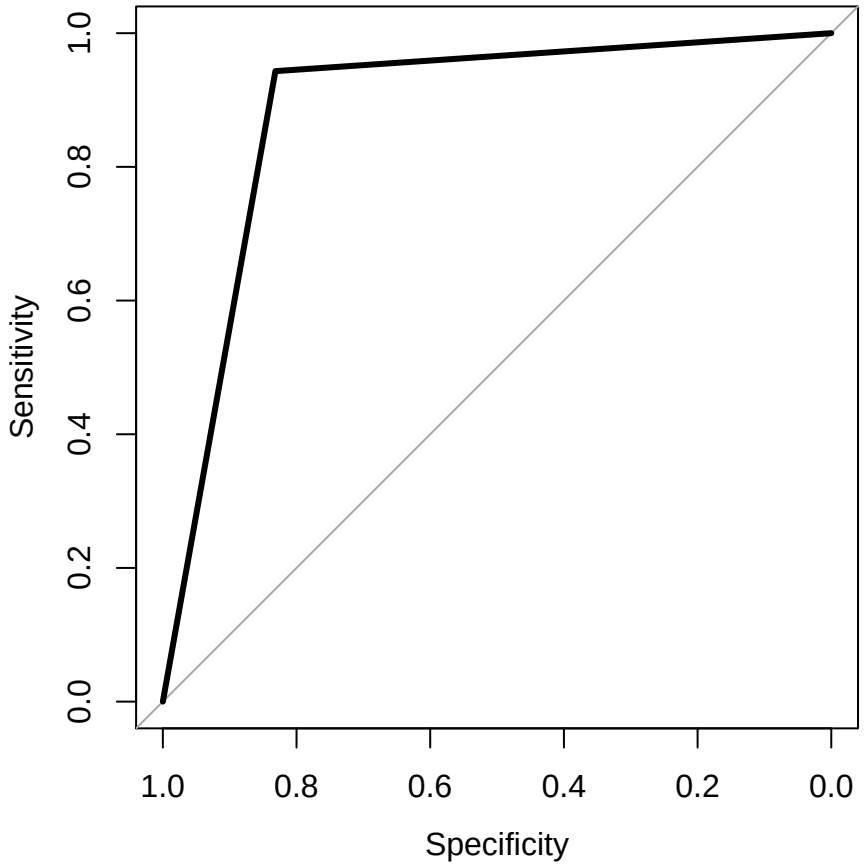
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
# REWMEMEVR TO ROUDN
# acc_f1_data <- matrix(c(mean(combo_acc_per_fold$accuracy),
#                     mean(wrist_acc_per_fold$accuracy),
#                     "", "",
#                     mean(combo_f1_per_fold$f1),
#                     mean(wrist_f1_per_fold$f1)),ncol=2,byrow=TRUE)
# # colnames(acc_f1_data) <- c("Combo","Wrist")
# # rownames(acc_f1_data) <- c("Accuracy","F1")
# acc_f1_tab <- as.table(acc_f1_data)
# library(broom)
# library(knitr)
# kable(acc_f1_tab, caption = "title")
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

Fig. 8 ROC Plot for Final Model



The ROC curve (Fig. 8) shows that our final model (using both chest and wrist data and including interactions for Subject) discriminates

moderately well between the stress and amusement states. Specifically, the area under the curve is 0.8874232.

Interaction Plot

```
# ggplot(data = pca_vals %>% filter(Subject %in% c("S4", "S9", "S13", "S17")), aes(x = Dim.1, y = predi
#   #geom_point() +
#   geom_jitter(width = 0.2, height = 0.1, size = 0.3) +
#   #geom_smooth(method = 'glm', method.args = list(family = 'binomial'),
#  #              size = 2, se = F, aes(color = as.factor(Subject))) +
#   geom_smooth(method = 'glm', method.args = list(family = 'binomial'),
#               size = 0.5, se = F, fullrange = T, aes(color = as.factor(Subject))) +
#   #geom_dl(aes(label = Subject), method = list(dl.combine("first.points", "last.points")))+
#   labs(color = "Subject", y = "Pr(Amusement State)") +
#   ylim(0,1)
#
# ggplot(data = pca_vals , aes(x = Dim.1, y = predicted, color = as.factor(Subject))) +
#   #geom_point() +
#   geom_jitter(width = 0.2, height = 0.1, size = 0.3) +
#   #geom_smooth(method = 'glm', method.args = list(family = 'binomial'),
#  #              size = 2, se = F, aes(color = as.factor(Subject))) +
#   geom_smooth(method = 'glm', method.args = list(family = 'binomial'),
#               size = 0.5, se = F, fullrange = T, aes(color = as.factor(Subject))) +
#   geom_dl(aes(label = Subject), method = list(dl.combine("first.points", "last.points")))+
#   labs(color = "Subject", y = "Pr(Amusement State)") +
#   ylim(0,1)
```

```
## Warning in RColorBrewer::brewer.pal(n, pal): n too large, allowed maximum for palette Set1 is 9
## Returning the palette you asked for with that many colors
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```

```
## Scale for 'y' is already present. Adding another scale for 'y', which will
## replace the existing scale.
```

```
## Warning in RColorBrewer::brewer.pal(n, pal): n too large, allowed maximum for palette Set1 is 9
## Returning the palette you asked for with that many colors
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
## Scale for 'y' is already present. Adding another scale for 'y', which will
## replace the existing scale.
```

Fig. 9 Marginal Effects of Motion

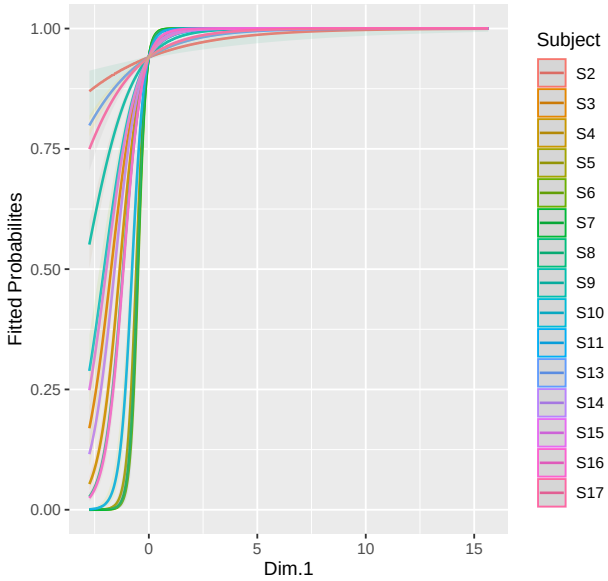
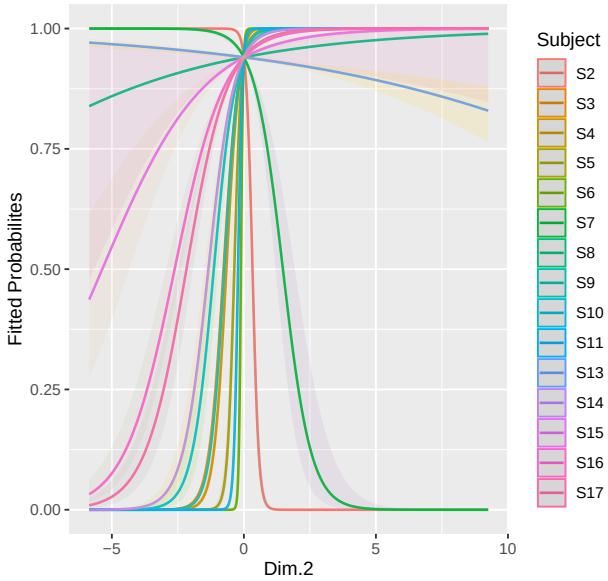


Fig. 10 Marginal Effects of Dermal Temperature and Act



As the variation in motion increases, the fitted probabilities (where stress is 1) increases similarly for all subjects except for S13 and S2. We note for S13 and S2 that as the variability of motion becomes gerater, the fitted probability of stress changes increase by a smaller amount. However, all subject exhibit a similar increasing trend. This behavior can be corroborated by Table [][][] in Appendix [][][]. As the variation in mean dermal temperature and activity increases, the fitted probabilities (where stress is 1) increases similarly for all subjects except for S2, S7, and S13. For these three subjects, the fitted probabilities decrease This behavior can be corroborated by Table [][][] in Appendix [][][].

```
p3 = plot_model(pca.model.interact, type = "pred", terms = c("Dim.3 [all]", "Subject")) +
  scale_fill_manual(values = mycolors)  +
  scale_y_continuous(name="Fitted Probabilites", limits=c(0, 1)) +
  labs(title = "Fig. 11 Marginal Effects of Heart Rate")
```

```
## Warning in RColorBrewer::brewer.pal(n, pal): n too large, allowed maximum for palette Set1 is 9
## Returning the palette you asked for with that many colors
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```

```
## Scale for 'y' is already present. Adding another scale for 'y', which will
## replace the existing scale.
```

```
p4 = plot_model(pca.model.interact, type = "pred", terms = c("Dim.4 [all]", "Subject")) +
  scale_fill_manual(values = mycolors)  +
  scale_y_continuous(name="Fitted Probabilites", limits=c(0, 1)) +
  labs(title = "Fig. 12 Marginal Effects of Neuro-Muscular Activity")
```

```
## Warning in RColorBrewer::brewer.pal(n, pal): n too large, allowed maximum for palette Set1 is 9
## Returning the palette you asked for with that many colors
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
## Scale for 'y' is already present. Adding another scale for 'y', which will
## replace the existing scale.
```

```
grid.arrange(p3, p4, nrow = 1)
```

Fig. 11 Marginal Effects of Heart Rate

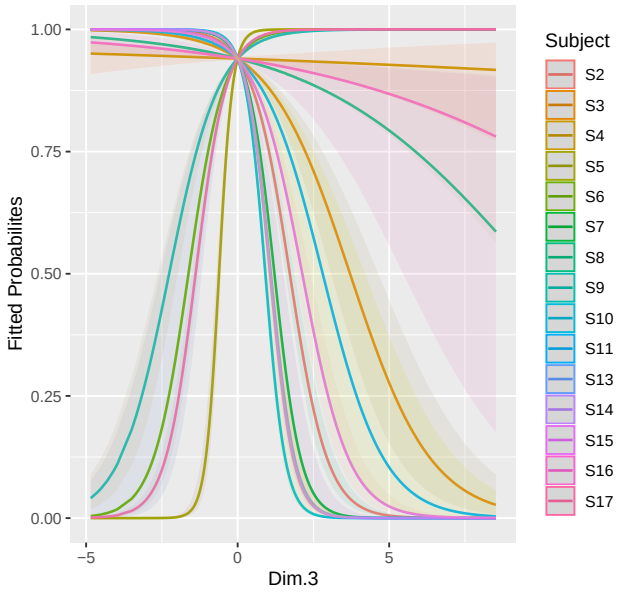


Fig. 12 Marginal Effects of Neuro−Muscular Activity

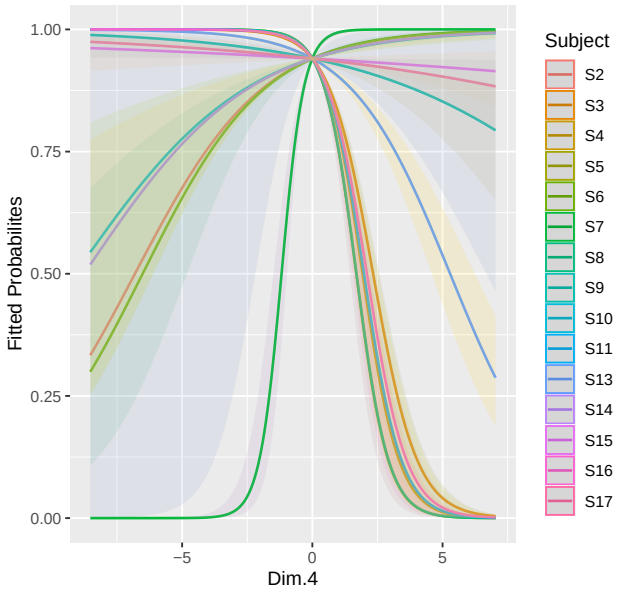


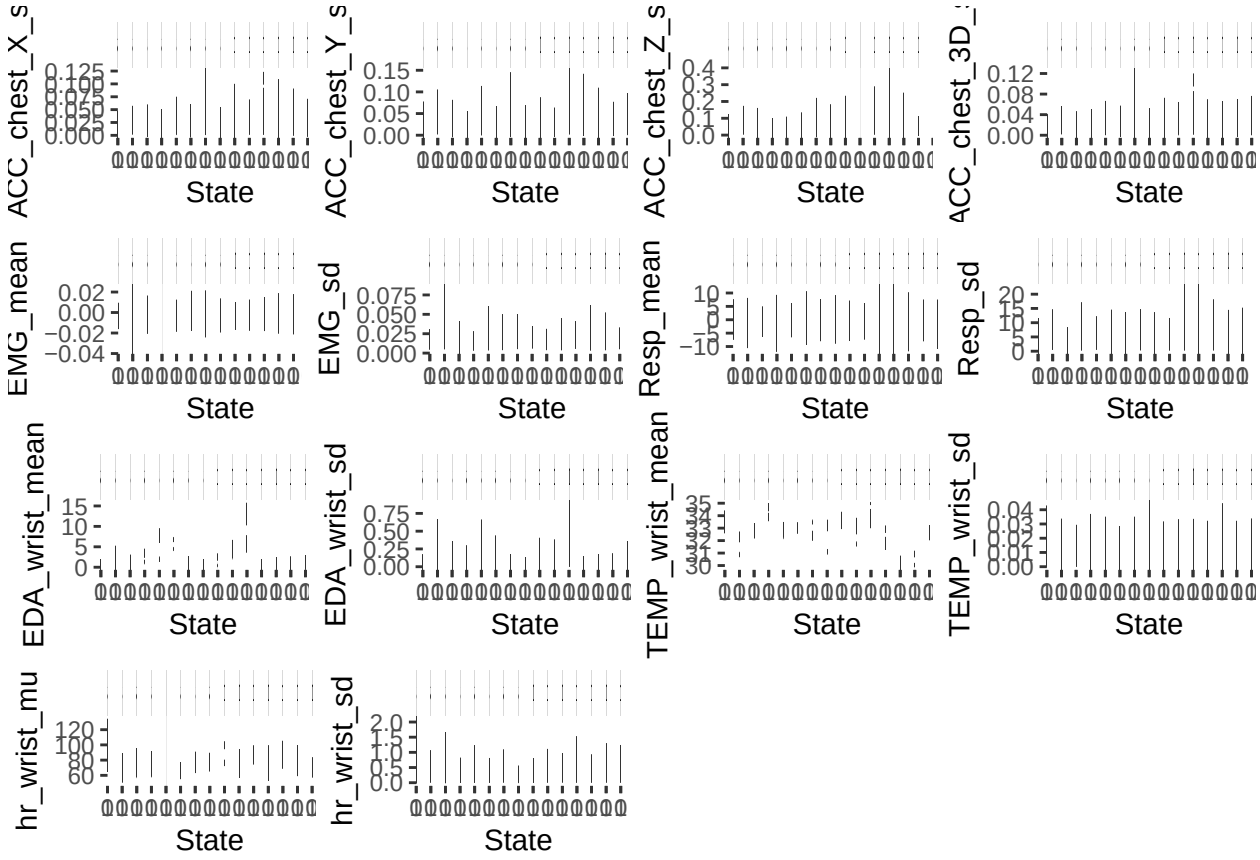
Figure ___ shows how as heart rate and variance in heart rate increases, the fitted probabilities of stress decreases similarly for all subjects except for S5, S6, S9, and S17. For these four subjects, the fitted probabilities increase. This behavior can be corroborated by Table [][][] in Appendix [][][], which show positive coefficients for the interaction between subject and the third principal component for these subjects. This positive cofficient indicates that, for these subjects, as heart rate and variance in heart rate increases, the odds of being stressed increase. Though the remaining subjects exhibit a seemingly "unnatural" trend, with the probability of stress *decreasing* with an increased heart rate, research has found that anxiety can be linked to a slowing heart rate as well.

Figure ___ shows how as neurological and muscular activity increase, the fitted probabilities of stress decrease similarly for all subjects barring S2, S6, S7, S10, and s14.For these five subjects, the fitted probabilities increase. As can be seen in Table _____ in Appendix _____, the coefficients for the interaction term between subject and the fourth principal component are positive for these subjects. For these individuals, as neuro-muscular activities, such as breathing, increasem the odds of being stressed increase as well.

Appendix

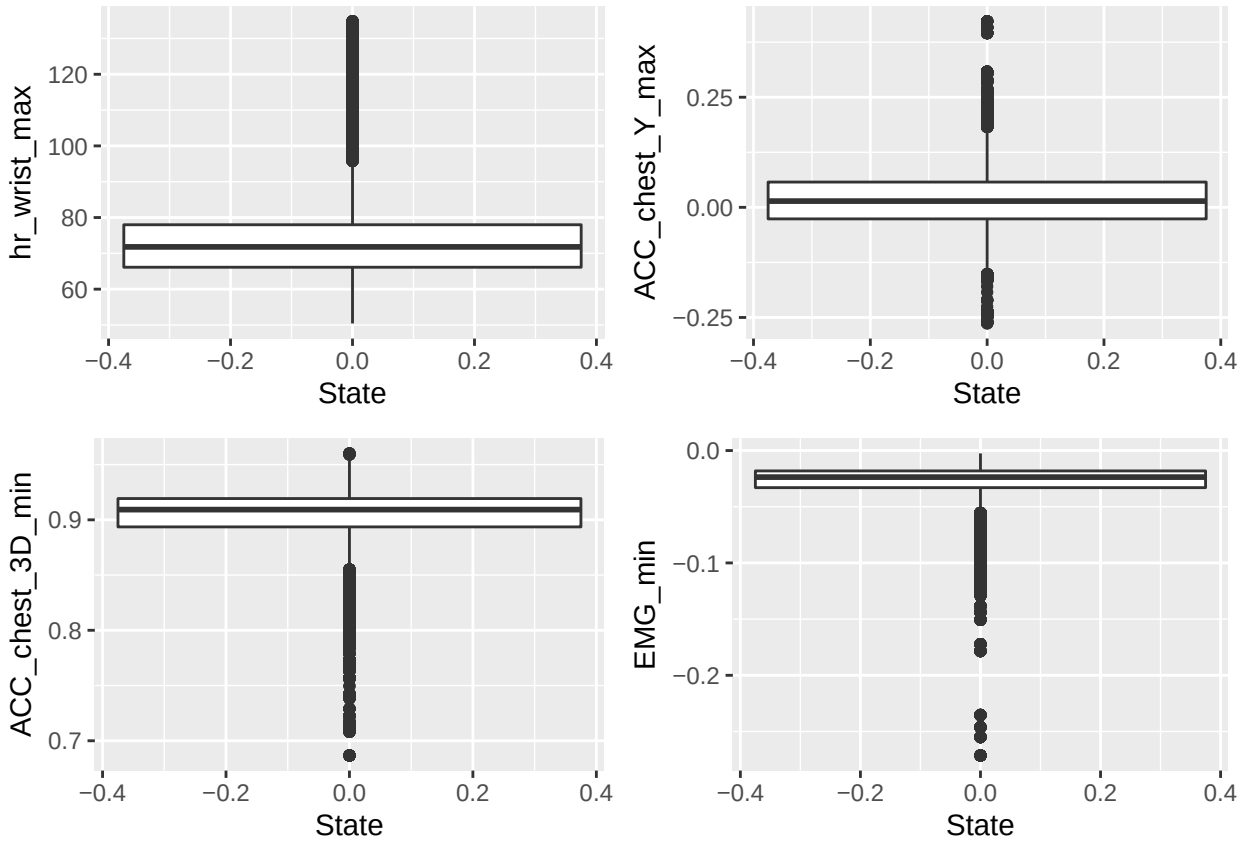
A.1

```
varvslab<- function(var) {
  ggplot(remove_select_cov, aes_(y=as.name(var), x=factor(remove_select_cov$Label), group = factor(remo
    labs(x ="State", y = as.name(var))
}
do.call(grid.arrange, lapply(names(remove_select_cov_nb), varvslab))
```



A.2

```
minmax<- function(var) {
  ggplot(ALL_df, aes_(y=as.name(var))) + geom_boxplot() +
    labs(x ="State", y = as.name(var))
}

p=lapply(c("hr_wrist_max", "ACC_chest_Y_max", "ACC_chest_3D_min", "EMG_min"), minmax)
do.call(grid.arrange, p)
```



Extra Functions

```
# IBI_data_3 = read.csv("~/case-study-2/WESAD/S3/S3_E4_Data/IBI.csv")
#
# RMSSD_calc = function(df, peaks) {
#   # for (i in 1:nrow(df)) {
#   #   if (i <= nrow(df)-peaks)
#   vals = rollapply(df[,2], width = peaks, by = 1, FUN = rmssd, align = "left")
#   #}
#
#   return(vals)
# }
#
# hrv = RMSSD_calc(IBI_data_3, 3)
# IBI_data_3[1:length(hrv),3] = hrv
# colnames(IBI_data_3)[3] = "HRV"
# IBI_data_3 = na.omit(IBI_data_3)
```

```
# hz_format = function(df, win_size) {
#   mu_vals = NULL
#   sd_vals = NULL
#
#   for (i in 1:nrow(df)) {
#     #if (i <= nrow(df)-win_size)
#     nexttime = df[i,1] + win_size
#       # browser()
```

```
#
#     #print(nexttime)
#     if (i+1 <= nrow(df)) {
#
#     for (j in i+1:nrow(df)) {
#            # print(df[j,1])
#       if (!is.na(df[j,1])) {
#       #   print(df[j,1]==nexttime)
#       # if (df[j,1]==nexttime) {
#       #   stoprow = j
#       # }
#         # print(i)
#         # print(j)
#         # print(df[j,1])
#         # print(nexttime)
#       if ((df[j,1] >= nexttime) & (df[j-1,1]<nexttime)) {
#         stoprow = j-1
#       }
#       }
#     }
#     mu_vals = c(mu_vals, mean(df[i:stoprow, 3]))
#     sd_vals = c(sd_vals, sqrt(var(df[i:stoprow, 3])))
#
#     }
#   }
#
#   return(list(hrv_mu = mu_vals, hrv_sd = sd_vals))
# }

# test = hz_format(IBI_data_3, 5)
```