Human Activity Recognition

The data set contains 561 features which were calculated from time series data from a waist sensor. The data represents 6 possible human activities, walking, walking up stairs, walking down stairs, sitting, standing, and lying (coded as numbers 1-6 respectively).

Loading packages:

In [49]:
```
library(dplyr)
library(ggplot2)
library(lattice)
library(stringr)
library(gridExtra)
library(caret)
library(rpart)
library(readr)
library(e1071)
options(repos='https://cran.cnr.berkeley.edu/')
install.packages('fastICA')
install.packages('klaR')
install.packages('kknn')
install.packages('gbm')

library(fastICA)
library(klaR)
library(kknn)
library(gbm)
```

...

Data Exploration

Loading Data:

In [26]:
```
X <- read_table('C:/Datasets/UCI HAR Dataset/train/X_train.txt', col_names=FALSE)
y <- read.csv('C:/Datasets/UCI HAR Dataset/train/y_train.txt', header = FALSE)
```

...

In [27]:
```
dim(as.matrix(X))
head(X,5)
```

7352 561

X1	X2	X3	X4	X5	X6	X7	X8	X9
0.2885845	-0.02029417	-0.1329051	-0.9952786	-0.9831106	-0.9135264	-0.9951121	-0.9831846	-0.923527
0.2784188	-0.01641057	-0.1235202	-0.9982453	-0.9753002	-0.9603220	-0.9988072	-0.9749144	-0.957686
0.2796531	-0.01946716	-0.1134617	-0.9953796	-0.9671870	-0.9789440	-0.9965199	-0.9636684	-0.977468
0.2791739	-0.02620065	-0.1232826	-0.9960915	-0.9834027	-0.9906751	-0.9970995	-0.9827498	-0.989302
0.2766288	-0.01656965	-0.1153619	-0.9981386	-0.9808173	-0.9904816	-0.9983211	-0.9796719	-0.990441

In [28]:
```
head(y,5)
```

V1
5
5
5
5
5

The response vector is an integer vector, which will be converted to a factor.

In [29]:
```
y[,1] <- factor(y[,1])
summary(y)
```

```
V1
1:1226
2:1073
3: 986
4:1286
5:1374
6:1407
```

Check for duplicates and missing values.

In [30]:
```
sum(duplicated(X))
```

0

In [31]:
```
sum(is.na(X))
sum(is.na(y))
```

0

0

The X matrix and y vector will be combined into a data frame for further processing.

In [32]:
```
df <- as.data.frame(X)
df$y <- y
```

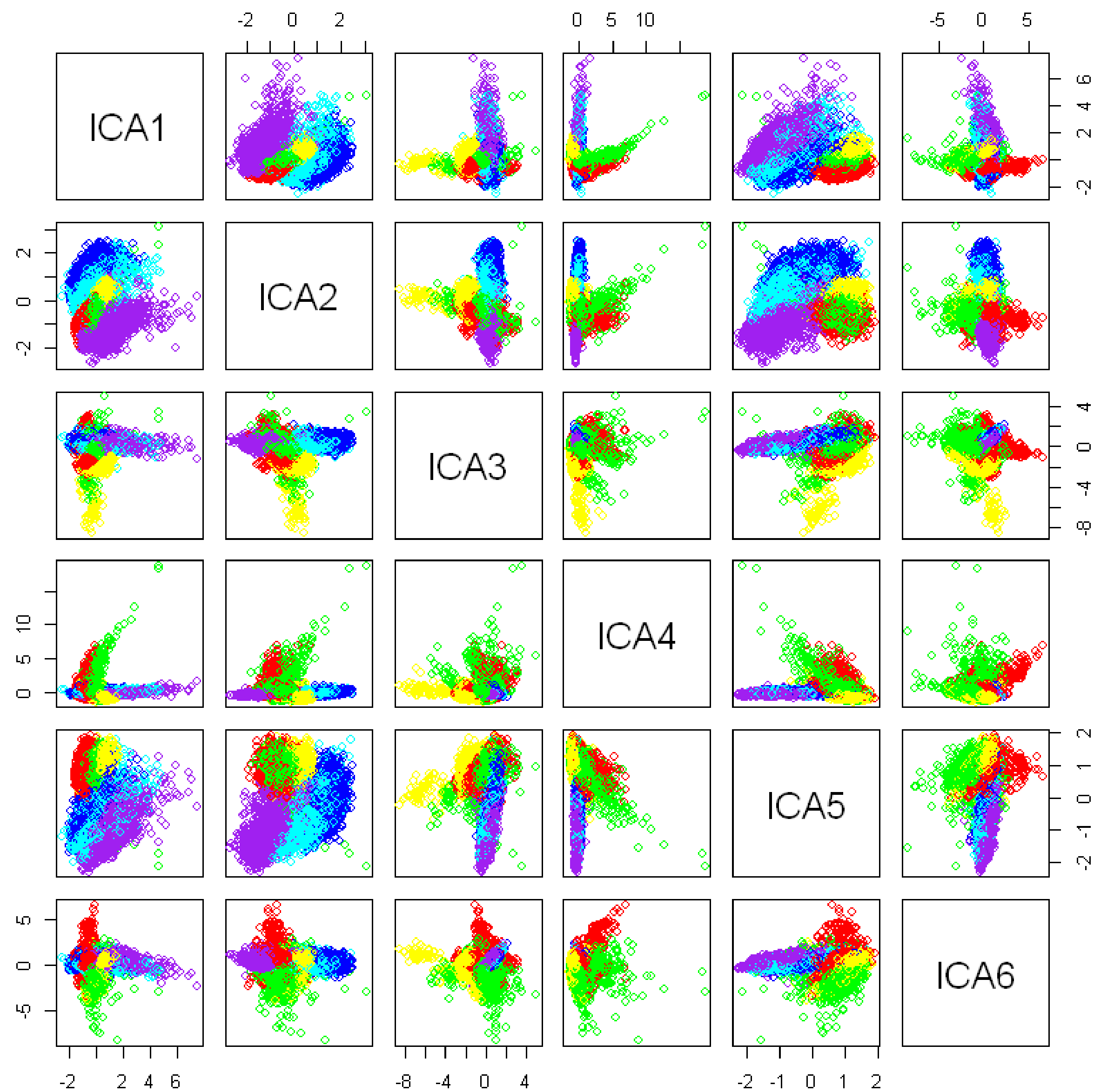
Since the features are not readily interpretable, further summarizing of the data will not provide much insight. Instead, the data will be visualized in pairwise plots.

Data Visualization

Since the data contains so many features, and the features in themselves already are not so easily interpretable, visualization will be performed by first using ICA to extract independent components.

In [47]:
```
ICA <- preProcess(X,method='ica',n.comp=6)
Xica <- predict(ICA, X)
```

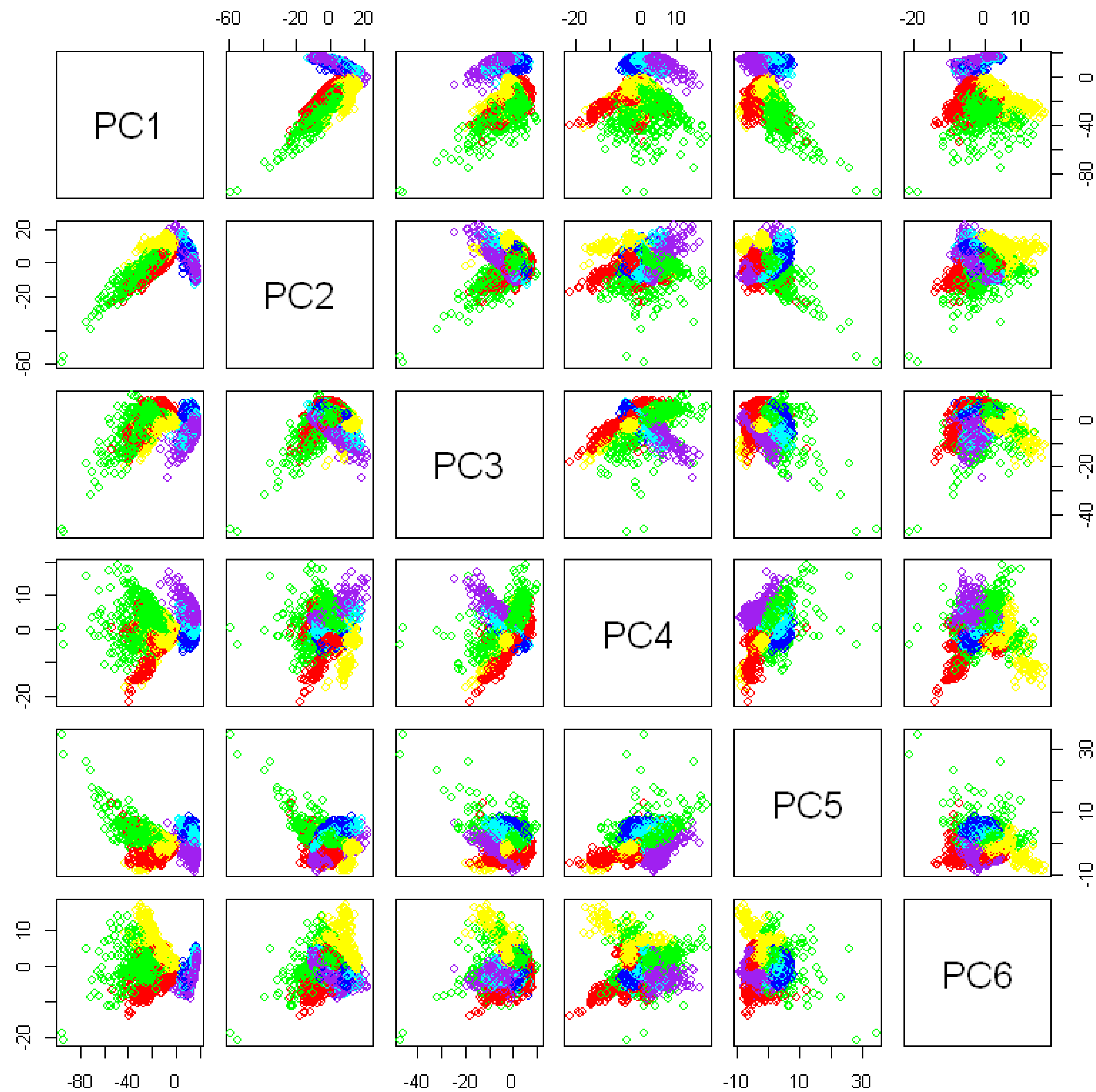
In [48]:
```
color <- character(7352)
colors <- c('red','yellow','green','cyan','blue','purple')
for(i in 1:6){
    color[y$V1 == paste(i)]<-colors[i]
}
pairs(Xica[,1:6],col=color)
```



For comparison, a similar plot is generated using PCA.

In [45]:
```
PCA <- preProcess(X,method='pca')
Xpca <- predict(PCA, X)
```

In [46]:
```
pairs(Xpca[,1:6],col=color)
```



Under ICA and PCA coordinates, the first three activity levels are sometimes separated from the last three levels. This makes sense because the first three involve walking (straight, up stairs, and down stairs) and the last three are stationary activities (sitting, standing, lying).

Both techniques separate the groups to a small degree, but there is still significant overlap. Modeling will be performed using the full set of predictors if possible.

Modeling

Linear Discriminant Analysis

In [52]:
```
LDAmodel <- train(X, y$V1, method = 'lda',trControl =trainControl(method='repeatedcv
LDAmodel
```

```
Linear Discriminant Analysis
7352 samples
561 predictor

    6 classes: '1', '2', '3', '4', '5', '6'
No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 5882, 5881, 5882, 5882, 5881, 5884, ...
Resampling results:

  Accuracy   Kappa
  0.9807763  0.9768605
```

Logistic Regression

In [54]:
```
Logisticmodel <- train(X, y$V1, method = 'multinom', MaxNWts = 4000, trControl =trai
Logisticmodel
```

```
Penalized Multinomial Regression

7352 samples
 561 predictor
   6 classes: '1', '2', '3', '4', '5', '6'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 5880, 5883, 5883, 5882, 5880, 5883, ...
Resampling results across tuning parameters:

decay  Accuracy   Kappa
0e+00  0.9473618  0.9366134
1e-04  0.9728876  0.9673573
1e-01  0.9844500  0.9812820

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was decay = 0.1.
```

Support Vector Machine

SVM and kNN need scaled features.

In [55]:
```
scaling <- preProcess(X, method='scale')
scaledX <- predict(scaling, X)
```

In [56]:
```
SVMLinmodel <- train(scaledX, y$V1, method = 'svmLinear', trControl =trainControl(me
SVMLinmodel
```

```
"Setting row names on a tibble is deprecated."Warning message:
"Setting row names on a tibble is deprecated."Warning message:
"Setting row names on a tibble is deprecated."Warning message:
"Setting row names on a tibble is deprecated."

Support Vector Machines with Linear Kernel

7352 samples
 561 predictor
   6 classes: '1', '2', '3', '4', '5', '6'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 5882, 5881, 5882, 5881, 5882, 5882, ...
Resampling results:

  Accuracy   Kappa
  0.9828163  0.979316

Tuning parameter 'C' was held constant at a value of 1
```

In [57]:
```
SVMRadmodel <- train(scaledX, y$V1, method = 'svmRadial', trControl =trainControl(me
SVMRadmodel
```

```
Support Vector Machines with Radial Basis Function Kernel

7352 samples
 561 predictor
   6 classes: '1', '2', '3', '4', '5', '6'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 5882, 5882, 5881, 5882, 5881, 5882, ...
Resampling results across tuning parameters:

  C     Accuracy   Kappa
  0.25  0.9584704  0.9500229
  0.50  0.9721173  0.9664414
  1.00  0.9778298  0.9733157

Tuning parameter 'sigma' was held constant at a value of 0.001997289
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.001997289 and C = 1.
```

k Nearest Neighbors

In [60]:
```
kNNmodel <- train(scaledX, y$V1, method = 'knn', trControl =trainControl(method='rep
kNNmodel
```

```
k-Nearest Neighbors

7352 samples
 561 predictor
   6 classes: '1', '2', '3', '4', '5', '6'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 5881, 5882, 5883, 5881, 5881, 5883, ...
Resampling results across tuning parameters:

  k Accuracy Kappa
5 0.9609630 0.9529978
7 0.9568827 0.9480809
9 0.9555670 0.9464954

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 5.
```

Naive Bayes

In [61]:
```
NBmodel <- train(X, y$V1, method = 'nb',trControl =trainControl(method='repeatedcv',
NBmodel
```

```
Naive Bayes

7352 samples
 561 predictor
   6 classes: '1', '2', '3', '4', '5', '6'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 5881, 5882, 5881, 5883, 5881, 5882, ...
Resampling results across tuning parameters:

   usekernel  Accuracy   Kappa
   FALSE      0.7209910  0.6653259
   TRUE       0.7704948  0.7242201

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
```

Decision Tree

In [62]:

```
Treemodel <- train(X, y$V1, method = 'rpart',trControl =trainControl(method='repeate
Treemodel
```

```
CART

7352 samples
 561 predictor
   6 classes: '1', '2', '3', '4', '5', '6'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 5882, 5883, 5882, 5880, 5881, 5882, ...
Resampling results across tuning parameters:

  cp         Accuracy   Kappa
  0.1663583  0.6133321  0.5297484
  0.2062237  0.4557399  0.3328328
  0.2311186  0.2784373  0.1093237

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.1663583.
```

In [64]:

```
Grid = expand.grid(cp=c(0.0001, 0.001, 0.01, 0.1))
Treemodel <- train(X, y$V1, method = 'rpart',trControl =trainControl(method='repeate
Treemodel
```

```
CART

7352 samples
 561 predictor
   6 classes: '1', '2', '3', '4', '5', '6'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 5881, 5881, 5883, 5881, 5882, 5883, ...
Resampling results across tuning parameters:

  cp     Accuracy   Kappa
  1e-04  0.9402862  0.9281200
  1e-03  0.9395601  0.9272425
  1e-02  0.8884664  0.8657260
  1e-01  0.8820278  0.8579325

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 1e-04.
```

In [66]:
```
Grid = expand.grid(cp=c(0.000001, 0.00001, 0.0001))
Treemodel <- train(X, y$V1, method = 'rpart',trControl =trainControl(method='repeate
Treemodel
```

```
"Setting row names on a tibble is deprecated."

CART

7352 samples
 561 predictor
   6 classes: '1', '2', '3', '4', '5', '6'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 5882, 5882, 5883, 5880, 5881, 5881, ...
Resampling results across tuning parameters:

  cp      Accuracy   Kappa
  1e-06   0.9391547  0.9267585
  1e-05   0.9391547  0.9267585
  1e-04   0.9396533  0.9273593

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 1e-04.
```

Random Forest

In [71]:
```
RFmodel <- train(X, y$V1, method = 'rf',trControl =trainControl(method='repeatedcv',
RFmodel
```

```
"Setting row names on a tibble is deprecated."

Random Forest

7352 samples
 561 predictor
   6 classes: '1', '2', '3', '4', '5', '6'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 5881, 5883, 5881, 5882, 5881, 5882, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
    2   0.9670848  0.9603772
   33   0.9797345  0.9756063
  561   0.9693515  0.9631070

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 33.
```

All the models that involve hyperplanes of separation perform better in cross validation than models that do not assume linear decision boundaries. This is probably due to the large number of features, which causes overfitting in those models. The models with hyplerplanes of separation have high accuracy in cross validation, which indicate minimal overfitting even with the large number of features. LDA, linear SVM, and logistic regression have similar cross validation accuracy, but LDA is faster. Therefore LDA will be chosen as the final model.

In [74]:
```
Xtest <- read_table('C:/Datasets/UCI HAR Dataset/test/X_test.txt', col_names=FALSE)
ytest <- read.csv('C:/Datasets/UCI HAR Dataset/test/y_test.txt', header = FALSE)
ytest[,1] <- factor(ytest[,1])
dftest <- as.data.frame(Xtest)
dftest$y <- ytest
```

...

In [78]:
```
ypred <- factor(predict(LDAmodel,Xtest))
confusionMatrix(ypred,ytest[,1])
```

```
Confusion Matrix and Statistics

          Reference
Prediction   1   2   3   4   5   6
         1 490  11   1   0   0   0
         2   6 460  14   1   0   0
         3   0   0 405   0   0   0
         4   0   0   0 434  22   0
         5   0   0   0  56 510   0
         6   0   0   0   0   0 537

Overall Statistics

               Accuracy : 0.9623
                 95% CI : (0.9548, 0.9689)
    No Information Rate : 0.1822
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9547
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
Sensitivity            0.9879   0.9766   0.9643   0.8839   0.9586   1.0000
Specificity            0.9951   0.9915   1.0000   0.9910   0.9768   1.0000
Pos Pred Value         0.9761   0.9563   1.0000   0.9518   0.9011   1.0000
Neg Pred Value         0.9975   0.9955   0.9941   0.9771   0.9908   1.0000
Prevalence             0.1683   0.1598   0.1425   0.1666   0.1805   0.1822
Detection Rate         0.1663   0.1561   0.1374   0.1473   0.1731   0.1822
Detection Prevalence   0.1703   0.1632   0.1374   0.1547   0.1921   0.1822
Balanced Accuracy      0.9915   0.9841   0.9821   0.9375   0.9677   1.0000
```

The LDA model is fairly accurate. Most of the confusion is between walking/walking upstairs/wallking downstairs and between standing and sitting. This is understandable because walking is similar to walking on stairs and because the orientation of the waist is similar when standing and sitting.