Practical Machine Learning Project

Tom Ritch

January 24, 2017

Synopsis

This report makes use of the dataset from a study titled “Qualitative Activity Recognition of Weight Lifting Exercises”.[1] Six male subjects, ages ranging 20 - 28 years, wore special athletic equipment fitted with accelerometers (weight belt, glove and armband). In addition, an accelerometer was placed on the dumbbell. The subjects performed dumbbell lifts correctly (class A) and incorrectly (classes B, C, D, E). This report will evaluate several machine learning models that attempt to predict the manner in which a human subject is performing a dumbbell lift exercise.

[1]Qualitative Activity Recognition of Weight Lifting Exercises (Velloso, Bulling, Gellersen, Ugulino, Fuks); ACM SIGCHI, 2013.

Exploratory Analysis

The dataset includes summary statistics calculated at a measurement boundary - these are very sparse and/or contain very high proportions of missing and/or error values. We note that columns 1:7 include an observation id, subject name, and time information. The information content in these is low, so they will be discarded.

```
div0 <- apply(data, 2, function(x) sum(x=="#DIV/0!", na.rm = T)>0)  # elimination vectors
blank <- apply(data, 2, function(x) sum(x=="", na.rm = T)>0)  # missing numbers scanning
NAs <- apply(data, 2, function(x) sum(is.na(x))>0 )  # NAs scanning
hiValue <- c(rep(TRUE, 7), rep(FALSE, length(names(data))-7))
elim <- names(data[, ( div0|blank|NAs|hiValue )])
elimVar <- c(); nC <- 5  # table of discarded variables for Backup Material
names <- c(paste0("` ", elim, " `"), rep("...",3)); nV <- length(names)
for(i in 1:nC) elimVar <- cbind(elimVar, names[seq(1,nV,nV/nC)[i] : (seq(1,nV,nV/nC)+(nV/nC-1))[i]])
data <- data[, !( div0|blank|NAs|hiValue )] # eliminate variables listed above
allVars <- names(data) # list of all variables
allPreds <- allVars[-grep("classe", allVars)] # list of all predictors
nzr <- nearZeroVar(data, saveMetrics = T)
if(any(!is.na(data)) & any(!nzr$zeroVar) & any(!nzr$nzv))
     cat("No missing values, no zero / near zero variance predictors remaining after eliminations")
```

```
## No missing values, no zero / near zero variance predictors remaining after eliminations
```

Summary statistics have been eliminated except for four total acceleration variables. Remaining predictors represent raw numeric data from sensors attached to the test subject forearm, arm, belt, and to the dumbbell. The 53 remaining variables are presented in the **Backup Material** section. We will not need to consider zero or near-zero variance predictors in our model development per the analysis above.

Comments on Exploratory Graphs, Sensor Data Correlation, Predictor Skewness

The reader is referred to **Exploratory Graphs** in **Backup Material** to explore characteristics of the training dataset. The complexity of the associations between response and predictors is displayed in **Exploratory Scatterplots - Arm Accelerometer, Magnetometer sensors**, which plots associations between `classe` response and arm mounted magnetometer and accelerometer sensors.

Is there significant collinearity among remaining predictors? Refer to the **Correlation map, ALL training predictors** in **Backup Material**. As shown, there exist mild to strong positive and negative correlations with sensor signals from the same location, but the author has chosen to not optimize further, and to err on the side of over-production of features to maximize the ranges presented to the machine learning algorithms discussed later in this report.

Skewness of remaining predictors is presented in **Skewness Analysis** in **Backup Material**.

Splitting

The dataset will be split 70/30 into training and testing datasets labeled `trn` and `tst` respectively. `trn` will be used to build and tune the model; `tst` will be used to estimate the model predictive performance. Variables in `trn` and `tst` are listed in **Backup Material**. `classe` (a factor) is the response.

```
set.seed(1);  inTrain <- createDataPartition(y = data$classe, p = 0.7, list = FALSE)
trn <- data[inTrain, ] # 70% to trn
tst <- data[-inTrain, ] # 30% to tst
cat("trn:", dim(trn)[1], "observations by", dim(trn)[2], "variables;  tst:",
    dim(tst)[1], "observations by", dim(tst)[2], "variables")
```

```
## trn: 13737 observations by 53 variables;  tst: 5885 observations by 53 variables
```

Model Development

The author fit five models for this supervised learning classification problem: (1) a CART decision tree, (2) a generalized boosted model, (3) a support vectors machine, (4) a random forest model invoked without train(), but for which tunable parameters were determined by way of the last model (5), a random forest model invoked within train(). Optimal tuning parameters are discussed below.

Decision Tree Model

We evaluate a CART decision tree, using train() method `rpart`. 52 variables were centered and scaled, of which 19 important variables were identified. Cross-validation parameters used: 10 folds, repeated 10 times.

```
set.seed(1); dTreeModel <- train(classe ~ ., method="rpart", data=trn, preProcess=c("center", "scal
e"))
dTreeModelPred <- predict(dTreeModel, newdata=tst, na.action = na.pass)  # predict w tst
cmDT <- confusionMatrix(dTreeModelPred, tst$classe)
```

Generalized Boosted Model

We now evaluate a Generalized Boosted Model tuned with train(). After some experimentation, the best tune was `n.trees` =150, `interaction.depth` =3, `shrinkage` =0.1. Cross-validation parameters: 10 folds, repeated 10 times.

```
set.seed(1); gbModel <- suppressMessages(train(classe ~ . , data=trn, method='gbm', metric='Accurac
y',
     preProcess=c("center", "scale"), verbose=F, trControl=trainControl(method="repeatedcv",
repeats=10)))
gbModelPred <- predict(gbModel, newdata=tst) # prediction with testing dataset
cmGB <- confusionMatrix(gbModelPred, tst$classe)
```

Support Vector Machine Model

We now evaluate a Support Vector Machine with Radial Basis Function Kernel model tuned with train(), `tuneLength=12` , and `method=svmRadial` . Fine tuning from multiple svm fits with sigma and C parameter ranges in expand.grid() identified these tuning parameters used in the calculation below. We avoided over-fitting with cross-validation (10 fold, repeated 10 times).

```
set.seed(1); gridSVM <- expand.grid(sigma=0.01228219, C=513)
svmModel <- train(classe ~ . , data = trn, method = "svmRadial", preProc = c("center", "scale"),
     tuneGrid = gridSVM, trControl = trainControl(method = "repeatedcv", repeats = 10))
svmModelPred <- predict(svmModel, newdata=tst) # prediction with testing dataset
cmSVM <- confusionMatrix(svmModelPred, tst$classe)
```

Random Forest Models

Two random forest models were created. The first (RF1) call was invoked as randomForest(), and performs well with default parameters. It was tuned after experimentation on RF2, specifically by using expand.grid() to hunt for `ntree` and `mtry` . Model features common to both RF1 and RF2 are `ntree=500` trees, `mtry=8` , and no pre-processing. In addition, we compensated for over-fitting on RF2 with cross-Validation (10 fold, 10 repeats).

```
# random forest model 1
set.seed(1)
RF1 <- randomForest(classe ~ ., trn, mtry=8, importance=T)
RF1Pred <- predict(RF1, newdata=tst)
cmRF1 <- confusionMatrix(RF1Pred, tst$classe)
# random forest model 2
grid <- expand.grid(.mtry=8)  # Grid Search for best tune mtry value
RF2 <- train(classe ~ . , data=trn, method="rf", metric='Accuracy', tuneGrid=grid, ntree=500,
     trControl=trainControl(method="repeatedcv", number=10, repeats=10, search="grid"))
RF2Pred <- predict(RF2, newdata=tst)
cmRF2 <- confusionMatrix(RF2Pred, tst$classe)
```

Model Comparisons

```
# table of 95% confidence intervals for model accuracy
accuracyTbl <- round(100*cmDT$overall[3:4],1)
accuracyTbl <- rbind(accuracyTbl, round(100*cmGB$overall[3:4],1))
accuracyTbl <- rbind(accuracyTbl, round(100*cmSVM$overall[3:4],1))
accuracyTbl <- rbind(accuracyTbl, round(100*cmRF1$overall[3:4],1))
accuracyTbl <- rbind(accuracyTbl, round(100*cmRF2$overall[3:4],1))
row.names(accuracyTbl) <- c('Decision Tree', 'Generalized Boosted', 'Support Vector Machine',
    'RandomForest 1', 'RandomForest 2')
htmlTable(accuracyTbl, caption='**Comparisons of model accuracy, 95% confidence intervals**')
```

Comparisons of model accuracy, 95% confidence intervals

	AccuracyLower	AccuracyUpper
Decision Tree	47.8	50.4
Generalized Boosted	95.9	96.8
Support Vector Machine	99.3	99.7
RandomForest 1	99.4	99.8
RandomForest 2	99.5	99.8

The last four models listed above were the highest performing of the five evaluated. For those four, we continue with a comparison of confusion matrices, out-of-box error rate, and out-of-sample error rate for the four high performing models. The confusion matrices are combined below.

```
# concatenate all confusion matrices
labs <- paste0("| ", rownames(cmSVM$table)); aln <- paste0(rep('lccccc',4))
hdr <- paste0('..',substring(labs, 3),'..')
tbl <- cbind(labs, cmGB$table, labs, cmSVM$table, labs, cmRF1$table, labs, cmRF2$table)
htmlTable(tbl, header=c('[ GB ]', hdr,'[ SVM ]', hdr, '[ RF1 ]', hdr,'[ RF2 ]', hdr),
    rnames=F, align=aln, header.align=aln, caption="**Confusion Matrices, Four Models**")
```

Confusion Matrices, Four Models

[GB]	..A..	..B..	..C..	..D..	..E..	[SVM]	..A..	..B..	..C..	..D..	..E..	[RF1]	..A..	..B..	..C..	..D..	..E..	[RF2]	..A..	..B..	..C..	..D..	..E..
| A	1640	39	0	1	1	| A	1673	3	0	0	0	| A	1671	2	0	0	0	| A	1672	2	0	0	0
| B	20	1075	39	3	13	| B	1	1131	7	0	0	| B	0	1135	7	0	0	| B	1	1136	7	0	0
| C	9	24	976	22	9	| C	0	4	1016	5	0	| C	1	2	1018	4	0	| C	1	1	1018	3	0
| D	4	1	9	929	7	| D	0	0	3	956	2	| D	0	0	1	959	3	| D	0	0	1	960	2
| E	1	0	2	9	1052	| E	0	1	0	3	1080	| E	2	0	0	1	1079	| E	0	0	0	1	1080

Out-of-Box error rate model comparisons

```
# OOB is ratio of misclassified to total training observations in confusion matrix
sumTrainObserv <- dim(trn)[1]  # total training observations
GBConfusion <- confusionMatrix(predict(gbModel, newdata=trn), trn$classe) # Decision Tree
sumMisclassified.GB <- sumTrainObserv-sum(diag(GBConfusion$table))
OOB.GB <- round(sumMisclassified.GB / sumTrainObserv, 5)
SVMConfusion <- confusionMatrix(predict(svmModel, newdata=trn), trn$classe) # Support Vectors Machin
e
sumMisclassified.SVM <- sumTrainObserv-sum(diag(SVMConfusion$table))
OOB.SVM <- round(sumMisclassified.SVM / sumTrainObserv, 5)
RF1Confusion <- with(RF1, confusion[1:nrow(confusion),1:nrow(confusion)])  # RF1
sumMisclassified.RF1 <- sumTrainObserv-sum(diag(RF1Confusion))
OOB.RF1 <- round(sumMisclassified.RF1 / sumTrainObserv, 5)
sumMisclassified.RF2 <- sumTrainObserv-sum(diag(RF2$finalModel$confusion))  # RF2
OOB.RF2 <- round(sumMisclassified.RF2 / sumTrainObserv, 5)
# comparison table
htmlTable(c(OOB.GB, OOB.SVM, OOB.RF1, OOB.RF2),
     header=c('| Generalized Boost |', '| Support Vector Machine |', '| RandForest1 |', '| RandFores
t2 |'),
     caption='**Out-of-Box Error Rate Comparison, Four Models**', rnames=F)
```

Out-of-Box Error Rate Comparison, Four Models

| Generalized Boost ||	Support Vector Machine ||	RandForest1 ||	RandForest2 |
0.02701	0.00051	0.00517	0.00524

Refer to the table above - note that the two random forest methods yielded very accurate fits and similar results, but were out-performed on this training set by the support vector machine.

Out-of-Sample error rate model comparisons

```
# OOS is (1 - Accuracy) of fit to testing set observations
OOS.GB <- round(1-cmGB$overall[1],5); OOS.SVM <- round(1-cmSVM$overall[1],5)
OOS.RF1 <- round(1-cmRF1$overall[1],5); OOS.RF2 <- round(1-cmRF2$overall[1],5)
htmlTable(c(OOS.GB, OOS.SVM, OOS.RF1, OOS.RF2),
     header=c('| Generalized Boost |', '| Support Vector Machine |', '| RandForest1 |', '| RandFores
t2 |'),
     caption='**Out-of-Sample Error Rate Comparison, Four Models**', rnames=F)
```

Out-of-Sample Error Rate Comparison, Four Models

Generalized Boost	Support Vector Machine	RandForest1	RandForest2
0.03619	0.00493	0.00391	0.00323

Refer to the table above. Previously, the support vector machine out-performed based on OOB error rate, but when out-of-sample (OOS) rates are compared, the random forest models out-perform. So we see a slightly different result when the models are evaluated with the test dataset. Again, both random forest models deliver similar results, but the trained random forest (RF2) method outperforms slightly.

Final Test: Evaluate Performance

The `pml-testing.csv` dataset includes 20 observations that will used to evaluate predictive performance.

```
gbPred.FT <- predict(gbModel, newdata=final.test, na.action = na.pass)  # evaluate on `final.test`
svmPred.FT <- predict(svmModel, newdata=final.test)
RF1Pred.FT <- predict(RF1, newdata=final.test)
RF2Pred.FT <- predict(RF2, newdata=final.test)
eq <- ifelse(gbPred.FT==svmPred.FT & svmPred.FT==RF1Pred.FT & RF1Pred.FT==RF2Pred.FT, 'T', 'F')
space <- rep('...' , 20)
htmlTable(t(data.frame('GB'=gbPred.FT, 'SVM'=svmPred.FT, 'RF1'=RF1Pred.FT, 'RF2'=RF2Pred.FT,
     space, 'Equal'=eq, fix.empty.names=F)), header=c(paste0('Ob','0',(1:9)), paste0('Ob',(10:20))))
```

	Ob01	Ob02	Ob03	Ob04	Ob05	Ob06	Ob07	Ob08	Ob09	Ob10	Ob11	Ob12	Ob13	Ob14	Ob15	Ob16	Ob17	Ob18	Ob19	Ob20
GB	B	A	B	A	A	E	D	B	A	A	B	C	B	A	E	E	A	B	B	B
SVM	B	A	B	A	A	E	D	B	A	A	B	C	B	A	E	E	A	B	B	B
RF1	B	A	B	A	A	E	D	B	A	A	B	C	B	A	E	E	A	B	B	B
RF2	B	A	B	A	A	E	D	B	A	A	B	C	B	A	E	E	A	B	B	B
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Equal	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T

Refer to the table above. Each of the four highest-performing models yield identical predictions for the twenty observation test dataset.

Expanded version of this report: http://rpubs.com/tomritch/Machine_Learning_Weightlifting_Exercises (http://rpubs.com/tomritch/Machine_Learning_Weightlifting_Exercises)

Backup Material

Correlation map, ALL training predictors

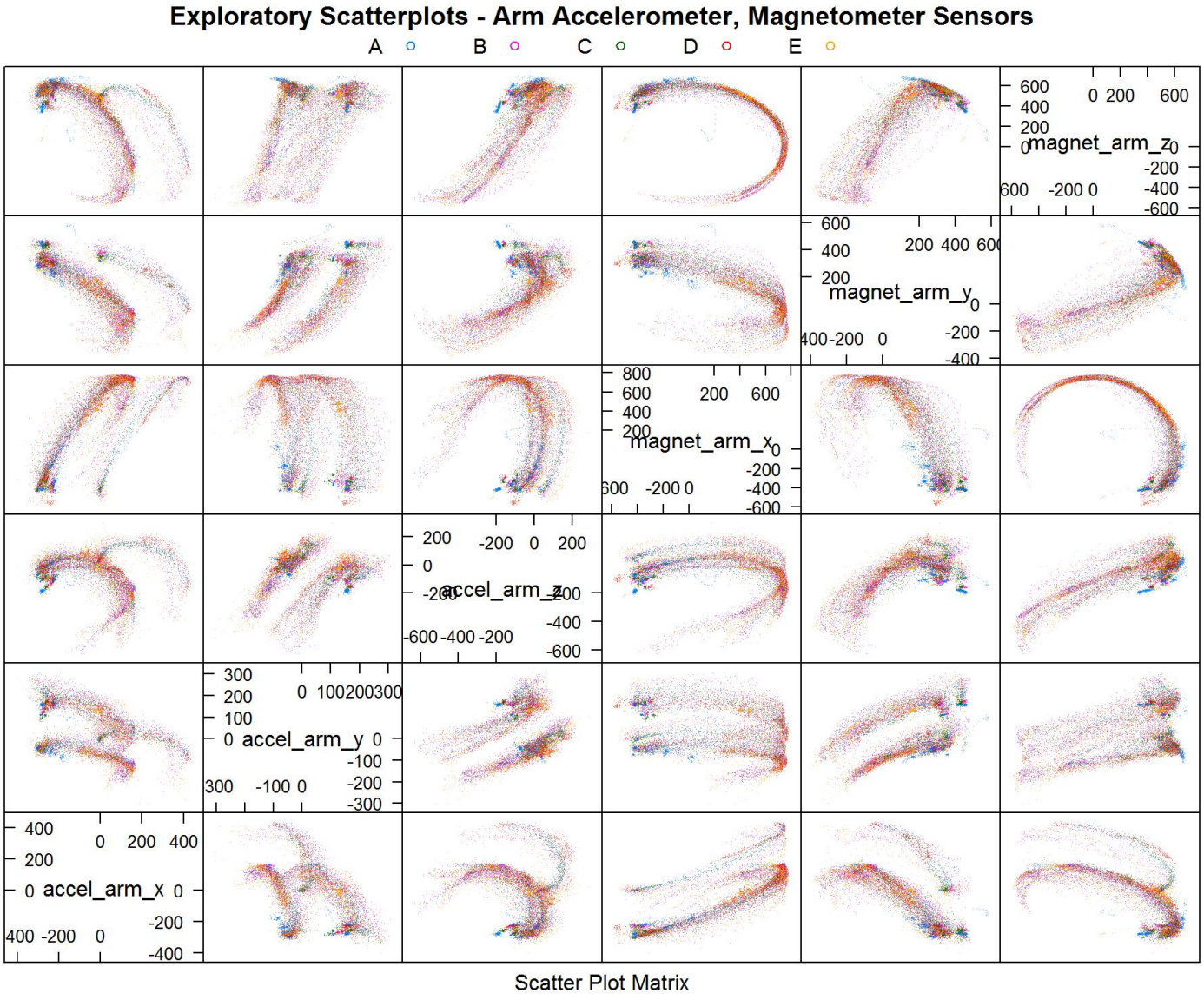
Exploratory Analysis referred the reader to the correlation plots in this section. Plots are provided for correlations for (1) all training dataset predictors below.

```
# correlation maps for ALL training predictors, plot order: first principal component order
corrplot(cor(trn[, allPreds]), tl.cex=.8, tl.col="firebrick", order="FPC",
         method="ellipse", type="full", sig.level=0.05, tl.pos="lt", cl.pos="n")
```

Exploratory Graphs

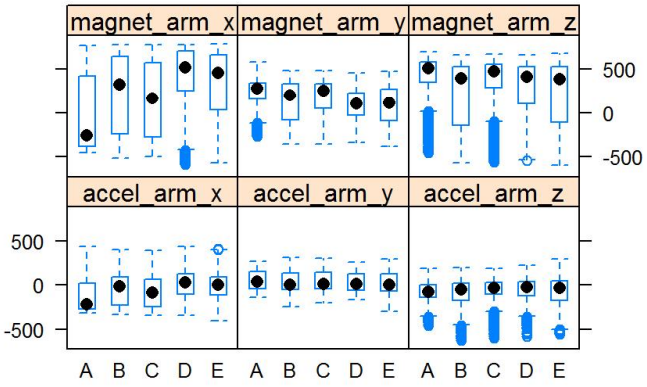
Exploratory Scatterplots - Arm Accelerometer, Magnetometer sensors

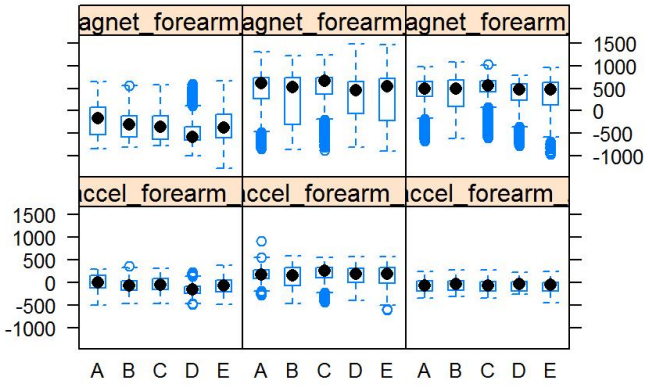
```
what <- (grepl("accel_arm", allPreds)|grepl("magnet_arm", allPreds))&!grepl("total", allPreds)
featurePlot(x=trn[,what], y=trn$classe, plot = "pairs", auto.key = list(columns = 5),
    pch='.', alpha=.5, cex=.2, aspect=.75, main="Exploratory Scatterplots - Arm Accelerometer, Magne
tometer Sensors")
```

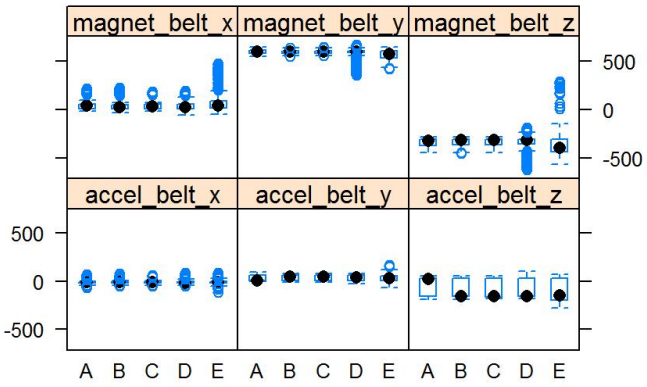


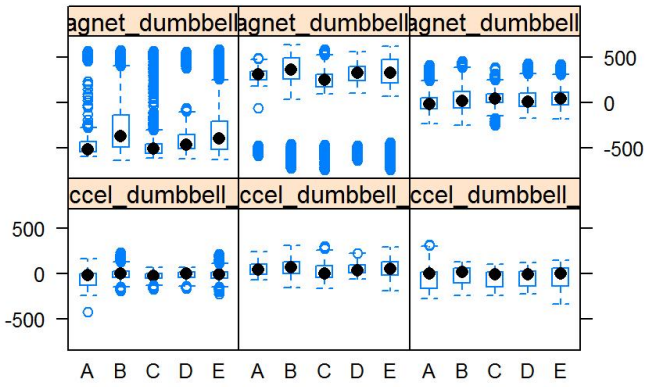
Exploratory Boxplots - Accelerometer, Magnetometer sensors

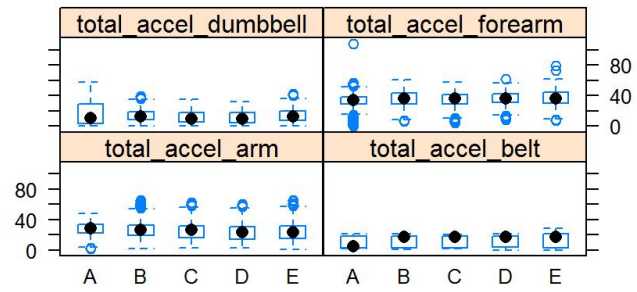
```
locs <- c("_arm", "_forearm", "_belt", "_dumbbell")  # sensor locations
plotIt <- function(feat) featurePlot(x=trn[ , feat], y=trn$classe, "box", label=NULL)
featGrep <- function(loc, f1, f2, f3, not)
      grepl(loc,allPreds)&!grepl(not,allPreds)&(grepl(f1,allPreds)|grepl(f2,allPreds)|grepl(f3,allPr
eds))
p1 <- plotIt(featGrep(locs[1], "magnet_", "accel_","accel_","total_accel_"))
p2 <- plotIt(featGrep(locs[2], "magnet_", "accel_","accel_","total_accel_"))
p3 <- plotIt(featGrep(locs[3], "magnet_", "accel_","accel_","total_accel_"))
p4 <- plotIt(featGrep(locs[4], "magnet_", "accel_","accel_","total_accel_"))
p5 <- plotIt(grepl("total_accel_", allPreds))
grid.arrange(p1, p2, p3, p4, p5, heights=c(3.75,3.75,3))
```





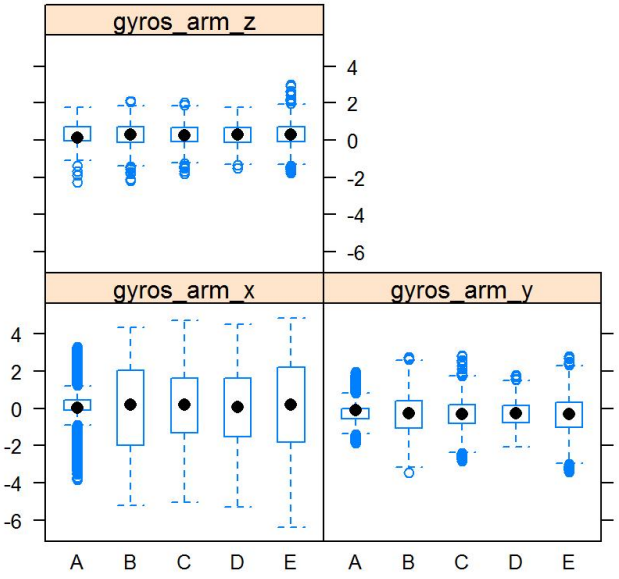


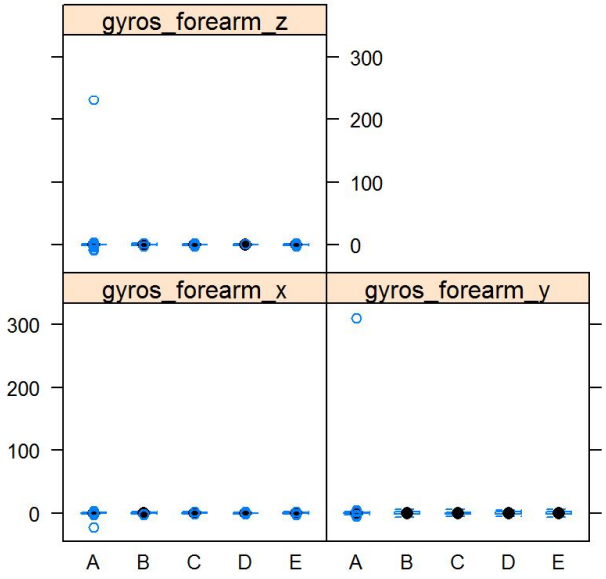


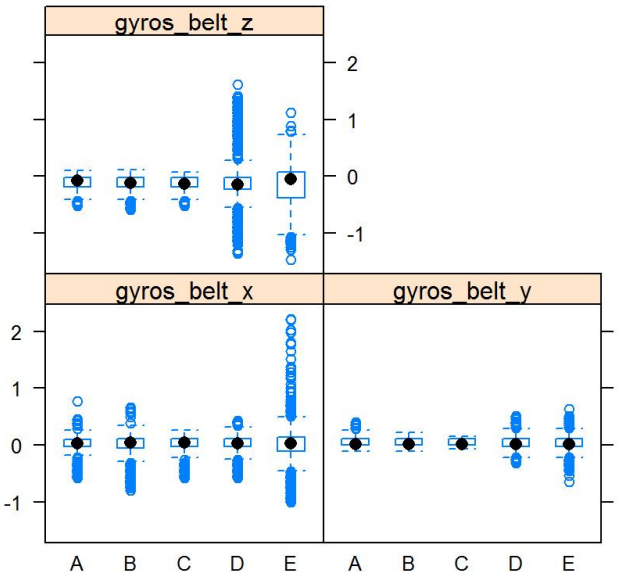


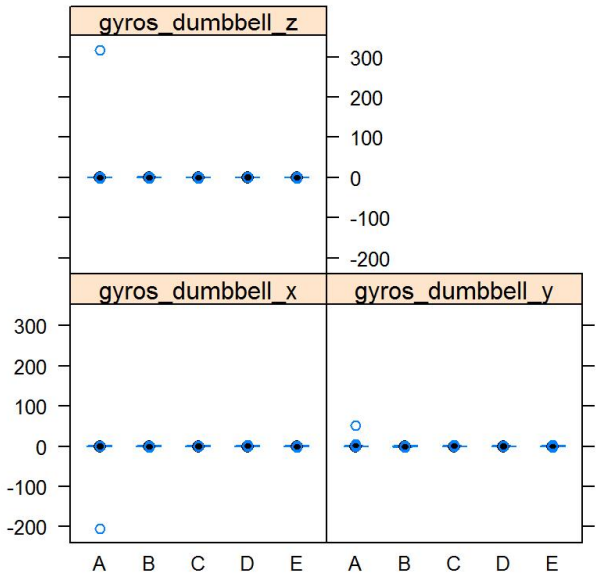
Exploratory Boxplots - Gyroscopic sensors

```
p6 <- plotIt(featGrep(locs[1], "gyros_", "gyros_","gyros_","total_accel_"))
p7 <- plotIt(featGrep(locs[2], "gyros_", "gyros_","gyros_","total_accel_"))
p8 <- plotIt(featGrep(locs[3], "gyros_", "gyros_","gyros_","total_accel_"))
p9 <- plotIt(featGrep(locs[4], "gyros_", "gyros_","gyros_","total_accel_"))
grid.arrange(p6, p7, p8, p9)
```



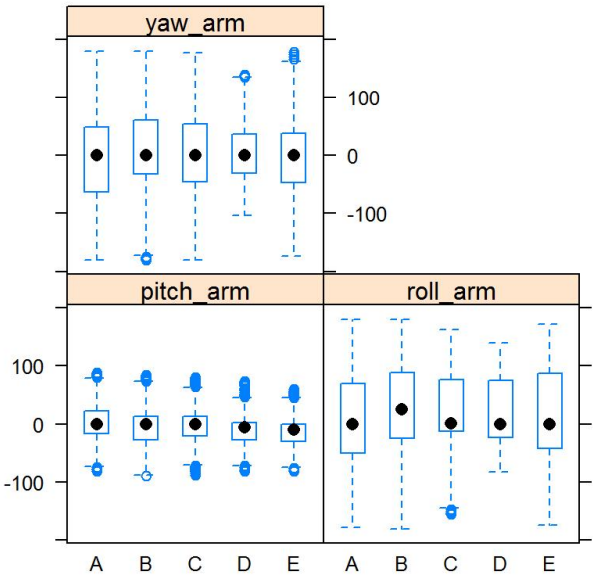


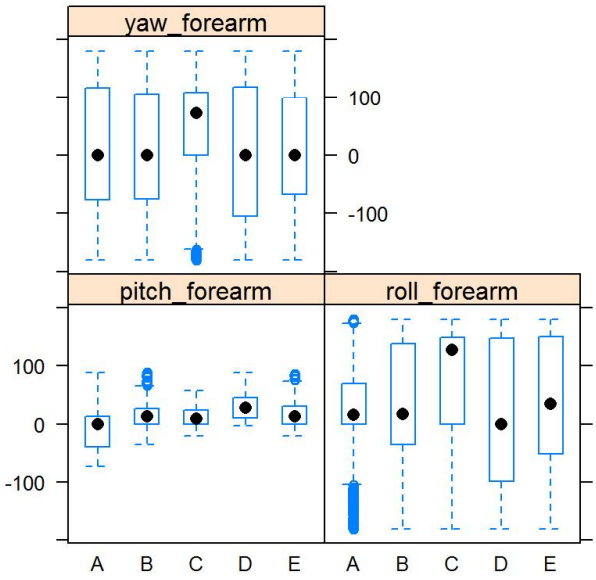


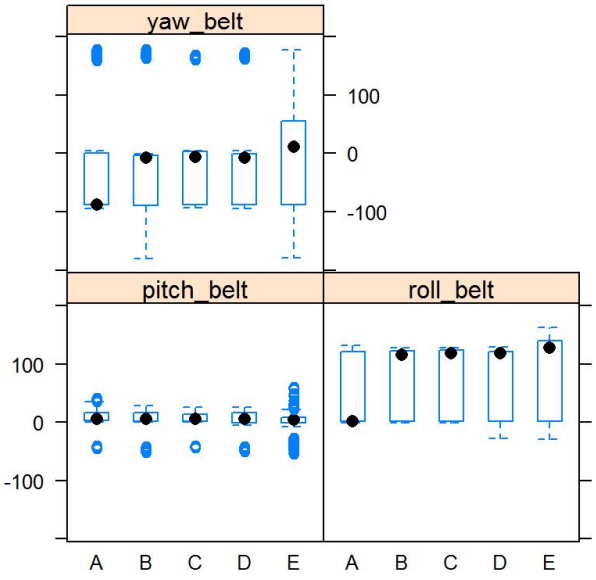


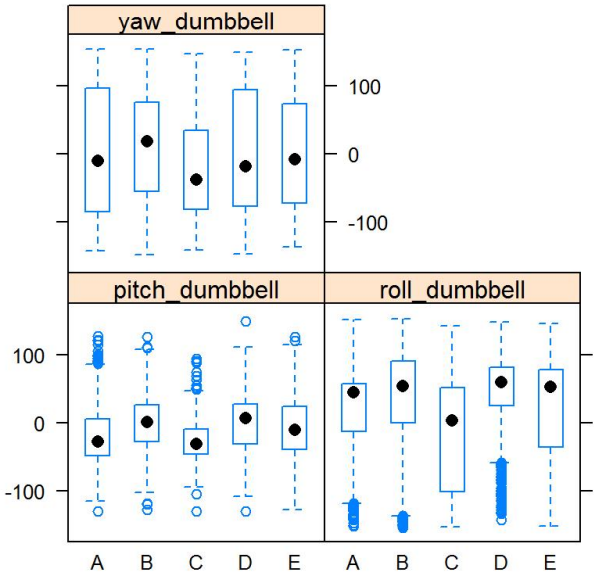
Exploratory Boxplots - Roll/Pitch/Yaw sensors

```
p10 <- plotIt(featGrep(locs[1], "roll_", "pitch_","yaw_","total_accel_"))
p11 <- plotIt(featGrep(locs[2], "roll_", "pitch_","yaw_","total_accel_"))
p12 <- featurePlot(trn[ , allPreds[featGrep(locs[3], "roll_", "pitch_","yaw_",
      "total_accel_")]], trn$classe, "box", label=NULL) # workaround
p13 <- plotIt(featGrep(locs[4], "roll_", "pitch_","yaw_","total_accel_"))
grid.arrange(p10, p11, p12, p13)
```









Skewness Analysis

```
skew <- function(x) round(abs(skewness(x)),2);  skew <- apply(trn[,allPreds], 2, skew)
skew <- skew[order(-skew)]
predSkew <- data.frame(Predictor=paste0("` ", names(skew), " `"), Skew=skew)
row.names(predSkew) <- NULL; n <- dim(predSkew)[1]
htmlTable(cbind(predSkew[1:(n/4),], predSkew[(n/4+1):(2*n/4),],
    predSkew[(2*n/4+1):(3*n/4),], predSkew[(3*n/4+1):n,]), align.header=rep('l',8),
    rnames=F, align=rep('l',8),  caption="***Sorted Predictor Skew***")
```

Sorted Predictor Skew

Predictor	Skew	Predictor	Skew	Predictor	Skew	Predictor	Skew
gyros_dumbbell_z	114.78	accel_belt_x	0.96	magnet_arm_y	0.49	gyros_arm_z	0.17
gyros_dumbbell_x	108.99	yaw_belt	0.9	accel_dumbbell_x	0.47	accel_belt_y	0.15
gyros_forearm_z	102.34	magnet_dumbbell_z	0.88	roll_forearm	0.46	magnet_arm_x	0.14
gyros_forearm_y	54.58	accel_arm_z	0.86	accel_forearm_z	0.44	gyros_belt_z	0.11
gyros_dumbbell_y	36.04	magnet_forearm_y	0.74	accel_arm_x	0.36	gyros_arm_y	0.11
gyros_forearm_x	2.65	roll_dumbbell	0.73	accel_dumbbell_y	0.36	yaw_arm	0.1
magnet_belt_y	2.22	accel_forearm_y	0.65	gyros_arm_x	0.29	accel_dumbbell_z	0.1
magnet_dumbbell_y	1.74	magnet_forearm_x	0.65	yaw_forearm	0.27	gyros_belt_y	0.09
magnet_dumbbell_x	1.69	total_accel_dumbbell	0.61	accel_forearm_x	0.23	accel_arm_y	0.09
magnet_belt_x	1.44	total_accel_forearm	0.57	yaw_dumbbell	0.22	total_accel_arm	0.07
magnet_forearm_z	1.22	gyros_belt_x	0.54	magnet_belt_z	0.19	total_accel_belt	0.03
magnet_arm_z	1.15	pitch_dumbbell	0.52	pitch_arm	0.19	roll_belt	0.02
pitch_belt	1	pitch_forearm	0.52	roll_arm	0.18	accel_belt_z	0.02

List of Remaining Variables Post-Eliminations

The variable elimination process described in the **Exploratory Analysis** section yielded the following variables remaining for model development, including the outcome `classe`.

```
tbl<-c(); nC <- 6; names <- c(paste0("` ", allVars[order(allVars)], " `"), rep("...",1))
nV <- length(names)
for(i in 1:nC)
    tbl <- cbind(tbl, c(names[seq(1, nV, nV/nC)[i] : (seq(1, nV, nV/nC)+(nV/nC-1))[i]]))
htmlTable::htmlTable(tbl, align='llllll', header=c(rep("",6)),
             caption="***Remaining variables in dataset (alphabetical order)***")
```

Remaining variables in dataset (alphabetical order)

accel_arm_x	accel_forearm_x	gyros_belt_z	magnet_arm_z	magnet_forearm_z	total_accel_arm
accel_arm_y	accel_forearm_y	gyros_dumbbell_x	magnet_belt_x	pitch_arm	total_accel_belt
accel_arm_z	accel_forearm_z	gyros_dumbbell_y	magnet_belt_y	pitch_belt	total_accel_dumbbell
accel_belt_x	classe	gyros_dumbbell_z	magnet_belt_z	pitch_dumbbell	total_accel_forearm
accel_belt_y	gyros_arm_x	gyros_forearm_x	magnet_dumbbell_x	pitch_forearm	yaw_arm
accel_belt_z	gyros_arm_y	gyros_forearm_y	magnet_dumbbell_y	roll_arm	yaw_belt
accel_dumbbell_x	gyros_arm_z	gyros_forearm_z	magnet_dumbbell_z	roll_belt	yaw_dumbbell
accel_dumbbell_y	gyros_belt_x	magnet_arm_x	magnet_forearm_x	roll_dumbbell	yaw_forearm
accel_dumbbell_z	gyros_belt_y	magnet_arm_y	magnet_forearm_y	roll_forearm	...