Course Project: Write Up

To begin I imported the training and test data sets. I then split the training set into a set with which I would train my model and one I would use for cross validation. Because the training set had so many data points, nearly 20,000, I portioned only 20% into the training subset and left 80% for validation. After some preliminary analysis I saw most of the variables were incomplete so I only included variables which were complete. This still left me with over 50 variables which I considered more than enough to create a sufficient model.

```
library(caret)
```

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

```
#import training and test data sets

testing <- read.csv("pml-testing.csv")
training <- read.csv("pml-training.csv")

#create a smaller more managable training set with 20% of training data.  The rest will be used to cr
oss validate.
part <- createDataPartition(training$classe,p=.2,list=F)
training.small <- training[part,-c(1:7,12:36,50:59,69:83,87:101,103:112,125:139,141:150),]
testing.small <- training[-part,-c(1:7,12:36,50:59,69:83,87:101,103:112,125:139,141:150),]
```

In order to preprocess and train the data I converted all but the classe variable into numeric classes. I then centered and scaled my variables and used a random forest method to create a model. I then validated the model with the remaining portion of the training set.

```
#Convert all but classe column to numeric data type
for (i in 1:52){
  training.small[,i] <- as.numeric(training.small[,i])
  }
#Create Model with preprocessing and evaluate on remainder of training set.
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.1
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
ModelFit <- train(training.small$classe~.,
                  method="rf",
                  preProcess=c("center","scale"),
                  data=training.small)
confusionMatrix(testing.small$classe,
                predict(ModelFit,testing.small))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 4417   30   11    3    3
##          B   93 2865   76    3    0
##          C    0   84 2609   44    0
##          D    0    6   74 2488    4
##          E    0   21   12   13 2839
## 
## Overall Statistics
## 
##                Accuracy : 0.9696
##                  95% CI : (0.9668, 0.9722)
##     No Information Rate : 0.2874
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9615
##  Mcnemar's Test P-Value : 3.339e-16
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9794   0.9531   0.9378   0.9753   0.9975
## Specificity            0.9958   0.9864   0.9901   0.9936   0.9964
## Pos Pred Value         0.9895   0.9434   0.9532   0.9673   0.9841
## Neg Pred Value         0.9917   0.9889   0.9866   0.9952   0.9995
## Prevalence             0.2874   0.1915   0.1773   0.1625   0.1813
## Detection Rate         0.2814   0.1825   0.1662   0.1585   0.1809
## Detection Prevalence   0.2844   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9876   0.9698   0.9640   0.9845   0.9970
```

I played with several preprocessing options many of wich made no difference or made prediction slightly worse. I ended with only centering and scaling the variables. I chose to use a random forest model because it is a model that produces highly accurate results when it has enough data, which I had, and it is good at classification.
The kappa statistic is the realitive accuracy the model adds compared to a random model. I estimated the out

of sample error rate to be 1 minus the kappa statistic, 0.0339. I knew that the out of sample error rate must be higher than the in sample error rate which was 0.0268.
Finally I used my model to make predictions on the test data set.

```
test.predictions <- predict(ModelFit,testing)
```