Untitled

This project is to identify the key columns in a data set and then create a machine learning solution so that we can assign people to classes based on the information collected.

Loads the basic data, I am also loading the sample data for the project as well as the submission section

```
library(caret)
```

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

```
library(corrgram)
```

```
## Warning: package 'corrgram' was built under R version 3.2.2
```

```
data <- read.csv("C:/Users/mark/Downloads/pml-training.csv", header = TRUE)
test <- read.csv("C:/Users/mark/Downloads/pml-testing.csv", header = TRUE)
```

IDENTIFYING COLUMNS TO USE:

The data set has a number of columns that are not numeric or composed mostly of nulls or invalid formulas. The following code sets all nulls to a character string and then creates a vector of values, false for columns that are not numeric and true for those that are. Even after this we have a number of columns dealing with the nature of the data that we do not want to analyses like timestamps and descriptors for the people. This code then sets those value to null.

The last step changes them into a vector so we can use it for filtering our data.

```
### removes non numeric Columns to
c_training <- data
c_training[is.na(c_training)] <- 'A'

## only returns columns of all valid entries columns with characters and blanks are set to false
columns <- lapply(c_training, is.numeric)
## Not all numeric columns are wanted, descriptive columns have been removed
columns$X <- FALSE
columns$raw_timestamp_part_1 <- FALSE
columns$raw_timestamp_part_2 <- FALSE
columns$num_window <- FALSE


## Changes the results to a column so we can filter our data on it
cl <- c(do.call("cbind",columns))
```

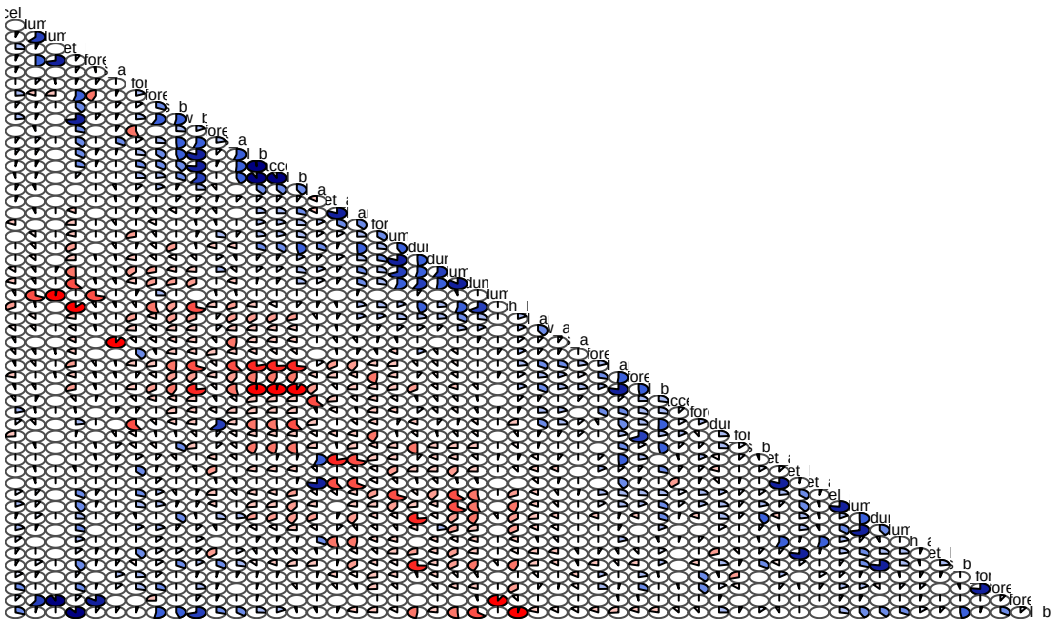
DATA CORRELATION:

One way to look at our data is to create a correlation matrix, this shows how the data moves together in our case it is shown as a pie chart and how full the pie is shows how related to items are.

```
 ## Create a corrilation matrix to test relationships

cor_data <- data[,cl]
cor_matrix <- cor( cor_data ,y=NULL)

##Print the matrix, did not print out the upper pannel and set the lower panel to show pie charts
corrgram(cor_matrix, order=TRUE, lower.panel=panel.pie, text.panel=panel.txt,upper.panel=NULL,
         main="CORRELATION MATRIX DATA")
```



FINAL PREP FROR MODEL:

Looking at the matrix lets us see that while some fields are highly correlated, we have almost not fields with correlations higher than .9.

Before creating our training and test data sets we need add in the classe column that we are predicting on. To allow for consistent rerunning of the report I am setting a seed for the random number generator before creating the vector that identifies the training and test sets.

```
columns$classe <- TRUE

## Changes the results to a column so we can filter our data on it
col <- c(do.call("cbind",columns))
set.seed(3425)
inTrain <- createDataPartition(y=data$classe,p=.8,list=FALSE)

training <- data[inTrain,col]
```

```
testing <- data[-inTrain,col]
homework  <- test[,col]
```

RUNNING THE MODEL:

For this project I choose to use the caret package random forest process limited to 5 layers.

```
modTree <- train(classe ~ ., data = training,method="rf",number = 5)
```

```
## Loading required package: randomForest
```

```
## Warning: package 'randomForest' was built under R version 3.2.2
```

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

```
## Look at model
```

This created the following model 27 with an accuracy of 0.9904396

```
modTree
```

```
## Random Forest
##
## 15699 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 15699, 15699, 15699, 15699, 15699, 15699, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9892384  0.9863841  0.001573305  0.001988798
##   27    0.9904396  0.9879052  0.001536555  0.001942861
##   52    0.9844481  0.9803242  0.004300379  0.005442785
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
summary(modTree$finalModel )
```

```
##                 Length Class      Mode
## call               5  -none-     call
## type               1  -none-     character
## predicted      15699  factor     numeric
## err.rate        3000  -none-     numeric
## confusion         30  -none-     numeric
## votes          78495  matrix     numeric
## oob.times      15699  -none-     numeric
```

```
## classes             5  -none-     character
## importance         52  -none-     numeric
## importanceSD        0  -none-     NULL
## localImportance     0  -none-     NULL
## proximity           0  -none-     NULL
## ntree               1  -none-     numeric
## mtry                1  -none-     numeric
## forest             14  -none-     list
## y               15699  factor     numeric
## test                0  -none-     NULL
## inbag               0  -none-     NULL
## xNames             52  -none-     character
## problemType         1  -none-     character
## tuneValue           1  data.frame list
## obsLevels           5  -none-     character
```

CROSS VALIDATION:

I then tested it on the testing sample I had excluded from the training set.

as you can see from the table the model we extremely accurate when predicting the values found in the test set.

In the table all the values on the lop left to lower right diagnal were correctly assigned while those that were on this line were assigned to an incorrect group.

With 2901 correct and 22 incorrect answers out 2923 of the model accuracy of 0.9924735

```
##predict on testing set
pred <- predict(modTree,testing);
testing$predRigh <- pred==testing$classe

table(pred,testing$classe)
```

```
##
## pred    A    B    C    D    E
##    A 1116    8    0    0    0
##    B    0  748    4    0    0
##    C    0    3  680    2    3
##    D    0    0    0  640    1
##    E    0    0    0    1  717
```

SUBMISSION:

Lastly I used the created model on the homework dataset to produce the output for the class project.

I then used the supplied R code for the summary assignment to print out the 20 answers.

```
pred <- predict(modTree,homework);
pred
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```