Machine Learning Project

Contents

========== load caret and ggplot2

```
library(knitr)
opts_chunk$set(fig.width=8, fig.height=8,dpi=144)
require(ggplot2)
require(dplyr)
require(caret)
require(ranger)
```

load data

```
training<-read.csv('./data/pml-training.csv',stringsAsFactors=TRUE,na.strings = c("#DIV/0!","NA"))
dim(training)
```

```
## [1] 19622   160
```

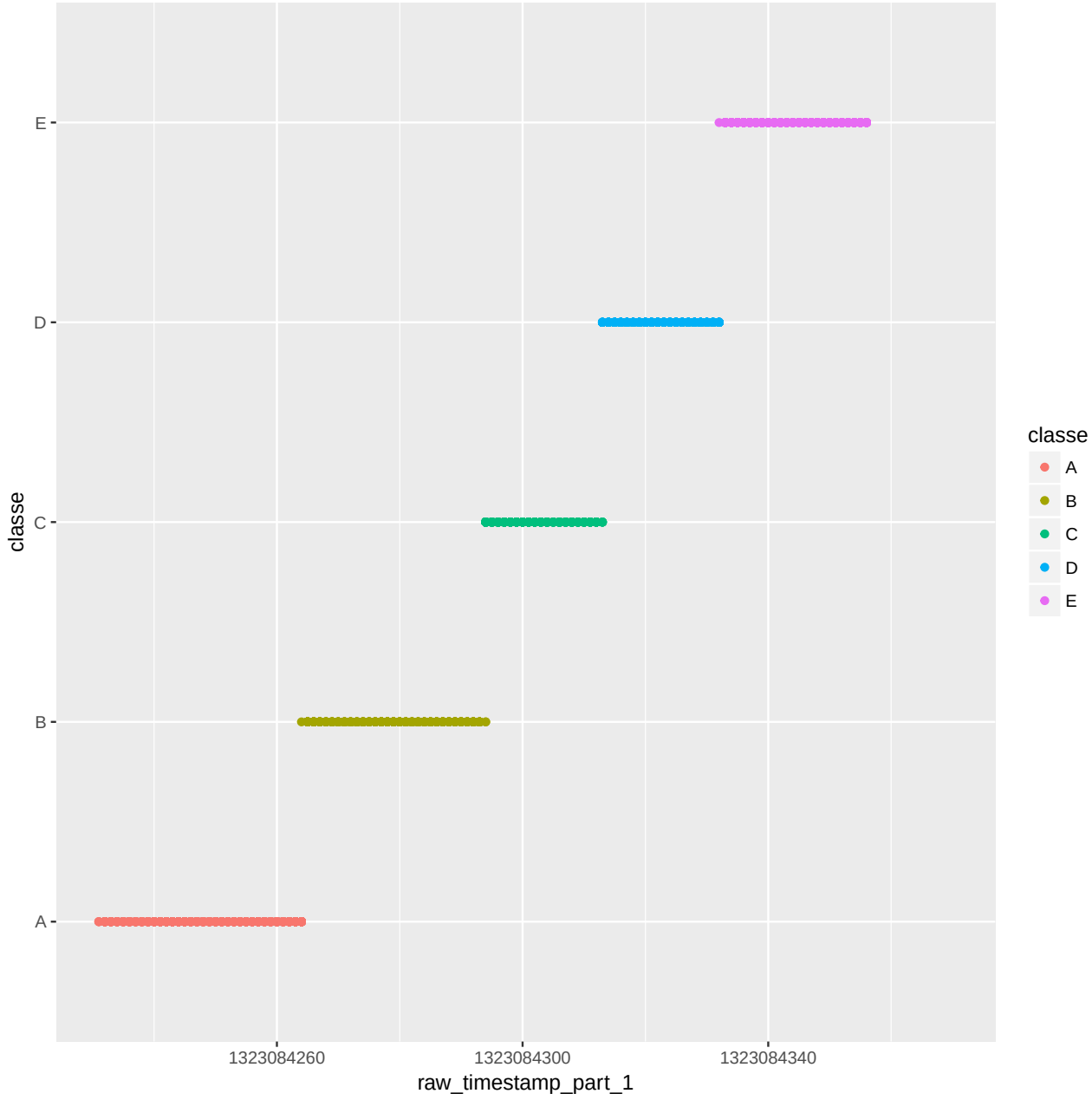
```
#str(training,list.len=ncol(training))
# training$cvtd_time<-strptime(training$cvtd_timestamp,format = "%d/%m/%Y %H:%M",list=FALSE)
```

Timestamps

classe is correlated with time:

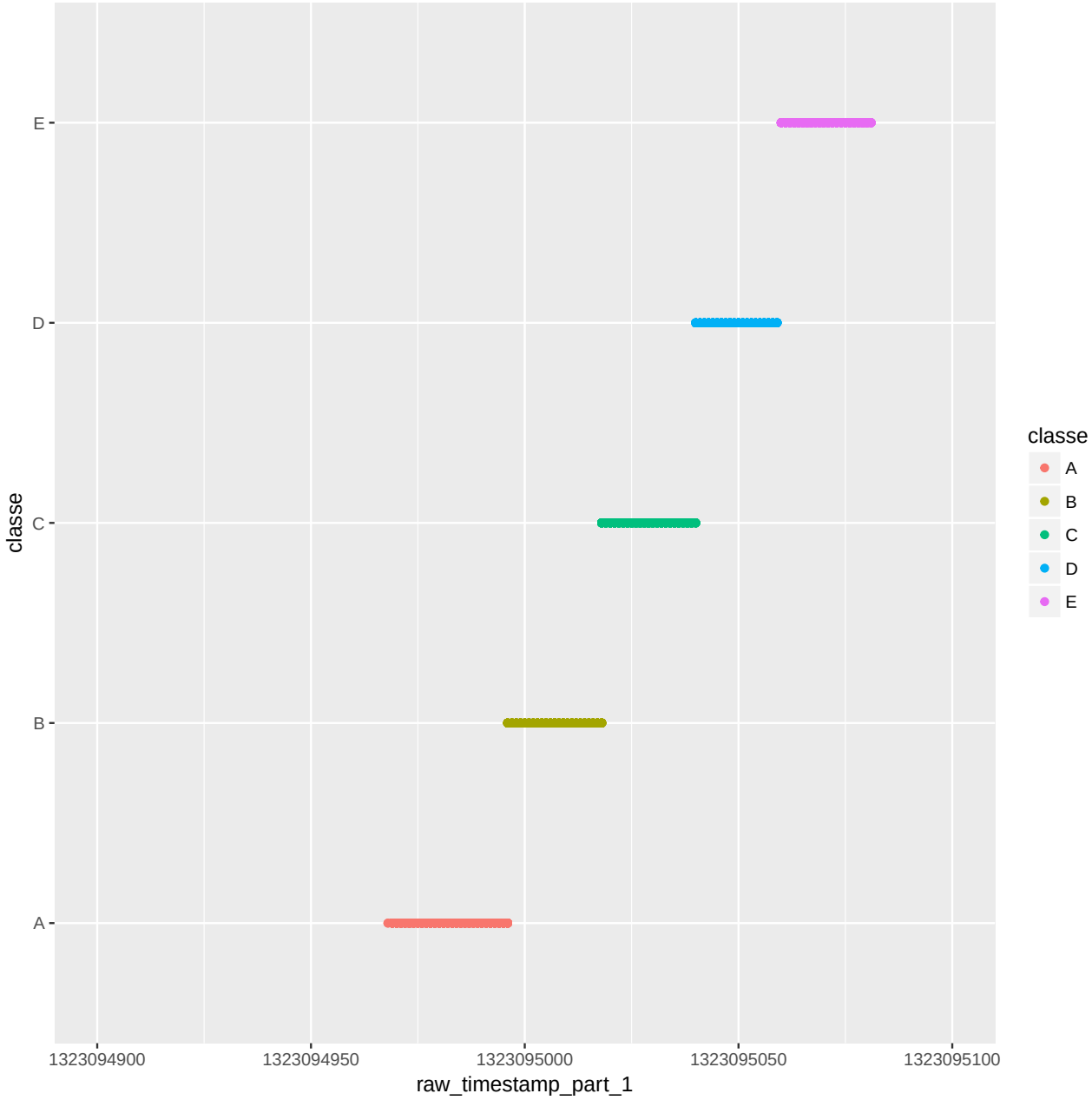
```
h<- ggplot(data=training,aes(x=raw_timestamp_part_1,y=classe,colour=classe)) + geom_point()
print(h + scale_x_continuous(limits=c(1323084231,1323084370)))
```

```
## Warning: Removed 16510 rows containing missing values (geom_point).
```




```
print(h + scale_x_continuous(limits=c(1323094900,1323095100)))
```

```
## Warning: Removed 17012 rows containing missing values (geom_point).
```



will want to strip irrelevant info, user, time, window stuff from training data.

tidying

write a function to do this, as we will need to tidy the test data also cvtd timestamp -> to date $ max_roll_belt
max_picth_belt min_roll_belt
min_pitch_belt amplitude_roll_belt
A bunch of others

set things to missing, impute the missing values

Then procede

```r
na_list<-sapply(training,function(x) { sum(is.na(x))})
good<- na_list < 1900
nona<-training[good]
# remove name, time, window info
slimmed<-nona[,-(1:7)]
str(slimmed)
```

```
## 'data.frame':    19622 obs. of  53 variables:
##  $ roll_belt           : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt    : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x        : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y        : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x        : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y        : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z        : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x       : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y       : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z       : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm           : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x         : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x         : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y         : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z         : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x        : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y        : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z        : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ roll_dumbbell       : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell      : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell        : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
```

```
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 ...
```

some exploratory plots

glm

We'll try a glm (binomial) model (logit). Maybe this does a bunch of models? make sure we do the preprocessing within the `train` statement, or else it messes up cross validation (We should split the data into folds and conduct PCA within each fold, rather than transforming via PCA and then splitting).

basic tree

```
treefit<-train(classe ~ . , data=slimmed, method='rpart')
treefit
```

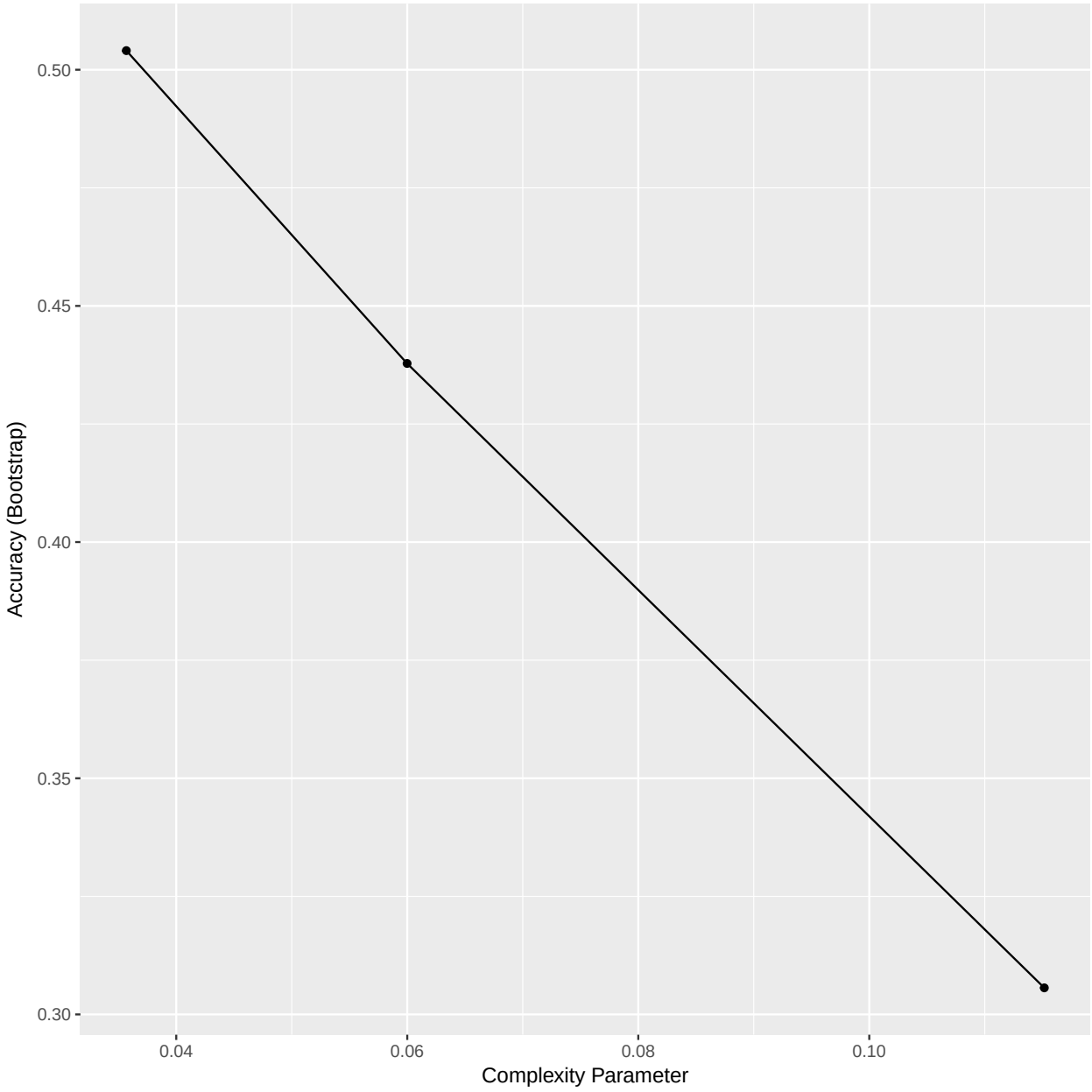
```
## CART
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 19622, 19622, 19622, 19622, 19622, 19622, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03567868  0.5040516  0.35265886
##   0.05998671  0.4378180  0.24575285
##   0.11515454  0.3056263  0.03397579
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03567868.
```

```
treefit$finalModel
```

```
## n= 19622
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 19622 14042 A (0.28 0.19 0.17 0.16 0.18)
```

```
##    2) roll_belt< 130.5 17977 12411 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -33.95 1578    10 A (0.99 0.0063 0 0 0) *
##      5) pitch_forearm>=-33.95 16399 12401 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 439.5 13870  9953 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 123.5 8643  5131 A (0.41 0.18 0.18 0.17 0.061) *
##         21) roll_forearm>=123.5 5227  3500 C (0.077 0.18 0.33 0.23 0.18) *
##       11) magnet_dumbbell_y>=439.5 2529  1243 B (0.032 0.51 0.043 0.22 0.19) *
##    3) roll_belt>=130.5 1645    14 E (0.0085 0 0 0 0.99) *
```

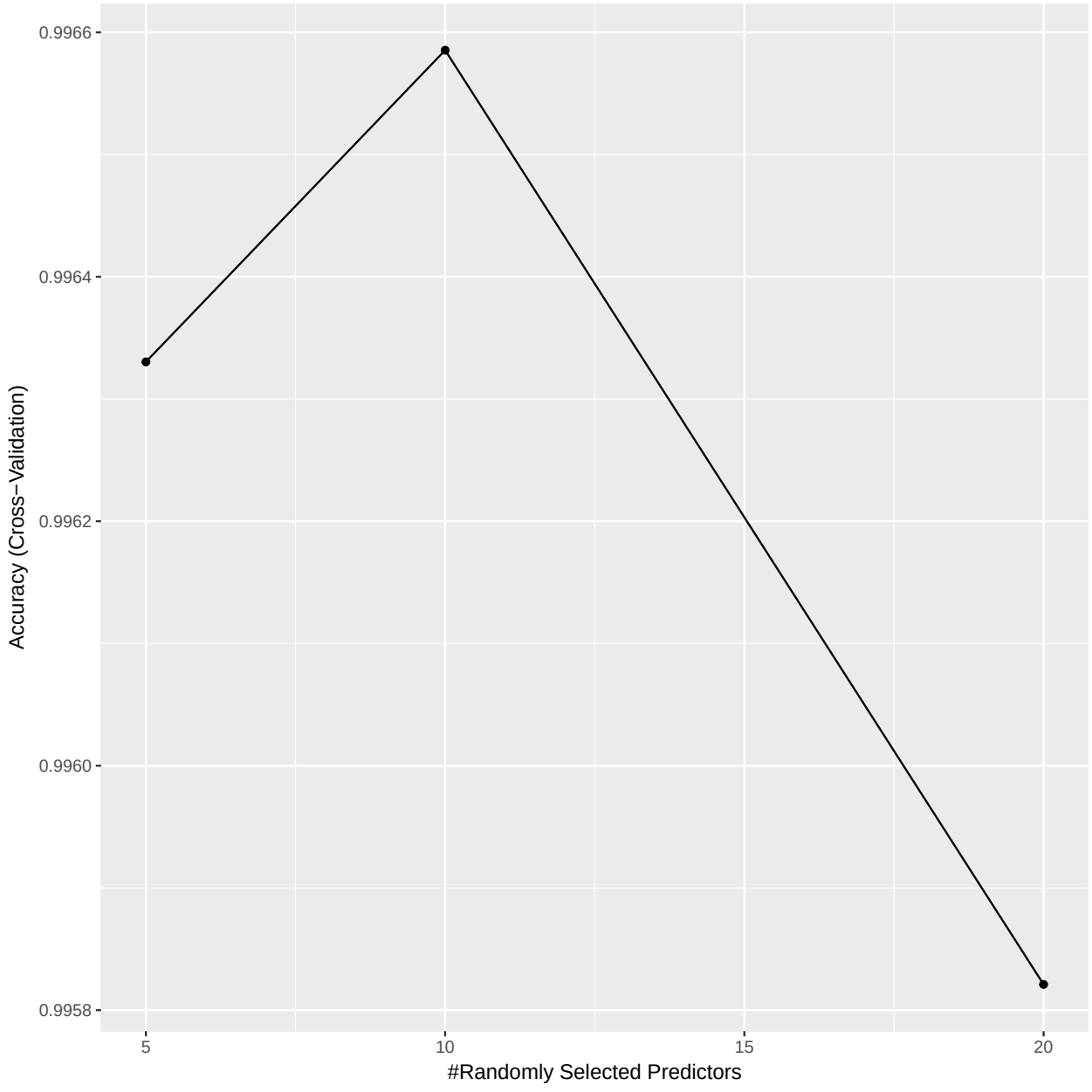
```
ggplot(treefit)
```



random forest

```
set.seed(5074491)
# tr_c<-trainControl(method='repeatedcv',number=20,repeats=5,allowParallel=TRUE)
tr_c<-trainControl(method='cv',number=10,allowParallel=TRUE)

# 20 fold cross validation, repeated 5 times
# can also select 'oob' for out of bag
# model_ranger_nopca<-train(classe~.,data=slim2,method='ranger',trControl=tr_c)
model_ranger<-train(classe ~ .  ,data=slimmed,method='ranger',trControl=tr_c,tuneGrid = expand.grid(mtry
ggplot(model_ranger)
```




```
model_ranger
```

```
## Random Forest
```

```
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 17660, 17660, 17660, 17658, 17661, 17661, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    5    0.9963304  0.9953580
##   10    0.9965854  0.9956806
##   20    0.9958210  0.9947137
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 10.
```

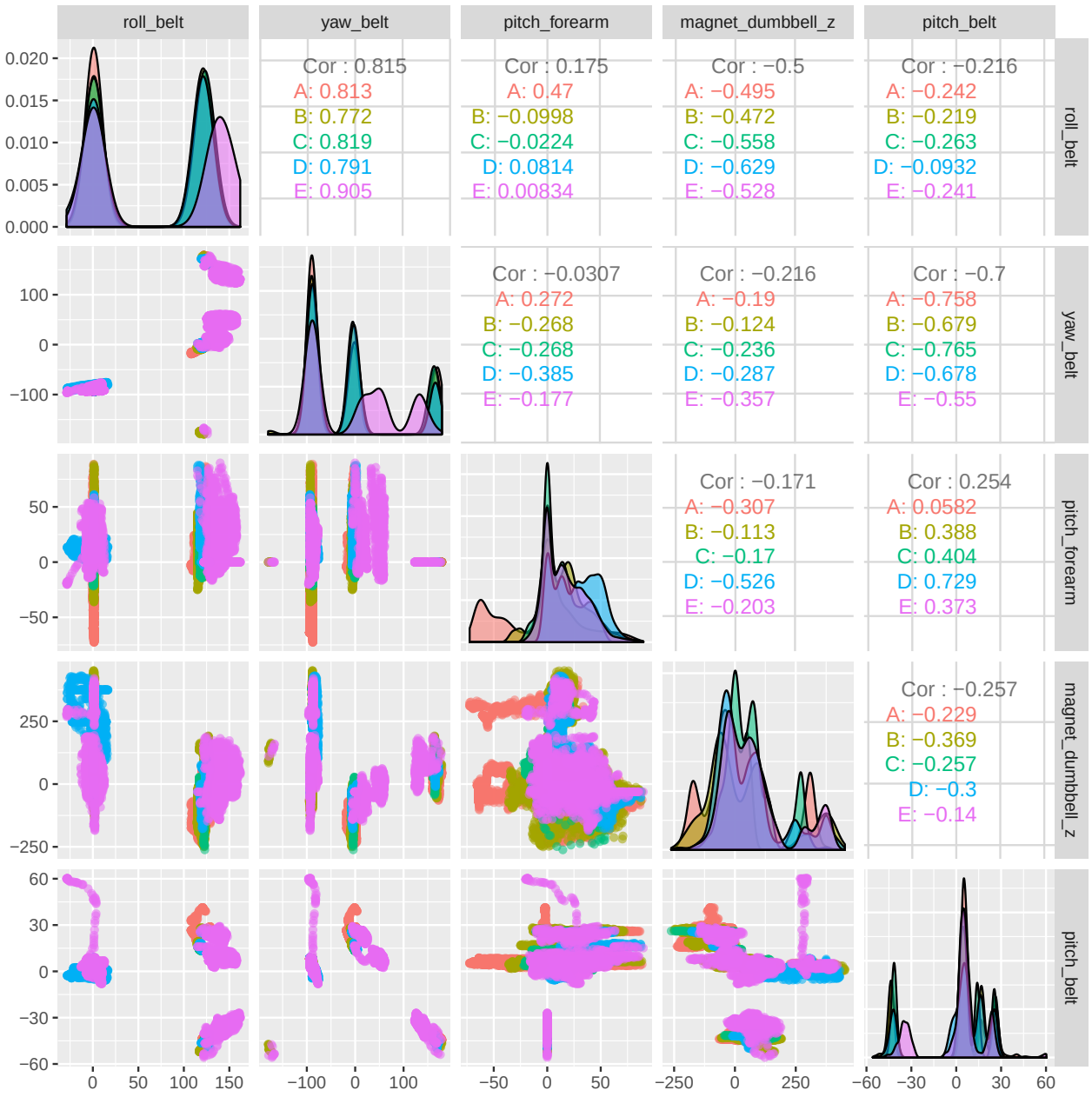
```
model_ranger$finalModel
```

```
## Ranger result
##
## Call:
##  ranger(.outcome ~ ., data = x, mtry = param$mtry, write.forest = TRUE,      probability = classProbs
##
## Type:                             Classification
## Number of trees:                  500
## Sample size:                      19622
## Number of independent variables:  52
## Mtry:                             10
## Target node size:                 1
## Variable importance mode:         impurity
## OOB prediction error:             0.29 %
```

```
# tr_c<-trainControl(method="cv",preProcOptions = list(thresh = 0.8))
# model_glm<-train(classe~.,data=slimmed,method='glm',family='binomial',train_control=tr_c,preProcess=c
# summary(model_glm)
```

Important variables

```
moose<-varImp(model_ranger)
mostimport <- order(moose$importance$Overall,decreasing=TRUE)
h<-ggpairs(data=slimmed,columns=mostimport[1:5],mapping=ggplot2::aes(colour = classe,alpha=0.1))
print(h)
```



A second random forest, this time using repeated cross validation, and maybe some more variable samplings

```
set.seed(5074491)
# tr_c<-trainControl(method='repeatedcv',number=20,repeats=5,allowParallel=TRUE)
tr_c<-trainControl(method='repeatedcv',number=10,repeats=3,allowParallel=TRUE)

# 20 fold cross validation, repeated 5 times
# can also select 'oob' for out of bag
# model_ranger_nopca<-train(classe~.,data=slim2,method='ranger',trControl=tr_c)
model_ranger2<-train(classe ~ .  ,data=slimmed,method='ranger',trControl=tr_c,tuneGrid = expand.grid(mt
```

ranger2 stuff

```
{r ranger2stuff,cache=TRUE,dependson='ranger2' }‘ ggplot(model_ranger2) model_ranger2
model_ranger2$finalModel # tr_c<-trainControl(method="cv",preProcOptions = list(thresh =
```

```
0.8)) # model_glm<-train(classe~.,data=slimmed,method='glm',family='binomial',train_control=tr_c,prePro
# summary(model_glm) varImp(model_ranger2)
```

Actually PCA

use some PCA preprocessing, maybe with another random forest

also look into svm

maybe one other model (gbm?) boosted decision tree?