Assignment

Hans

April 9, 2018

Load the data

Data is in quite a weird format. Let us apply some changes to the data such that observations correspond to rows and features correspond to columns (i.e. translate it to a tidy format)

```
train <-  read_delim("Train_call.txt",delim="\t") %>%
  mutate(chrom_loc = as.factor(paste0("C",Chromosome,"_",Start,"_",End)))
```

```
## Parsed with column specification:
## cols(
##   .default = col_integer()
## )
```

```
## See spec(...) for full column specifications.
```

```
metadata <- train %>% subset(select=c(Chromosome, Start, End, Nclone, chrom_loc))

train <- train %>% gather(Sample,"measurements", which(grepl("Array",names(train)))) %>%
  subset(select=c(Sample, measurements, chrom_loc)) %>% spread(chrom_loc, measurements)

train <- read_delim("Train_clinical.txt", delim="\t") %>%
  mutate(Subgroup = as.factor(Subgroup)) %>%
  merge(train)  %>% as.data.frame()
```

```
## Parsed with column specification:
## cols(
##   Sample = col_character(),
##   Subgroup = col_character()
## )
```

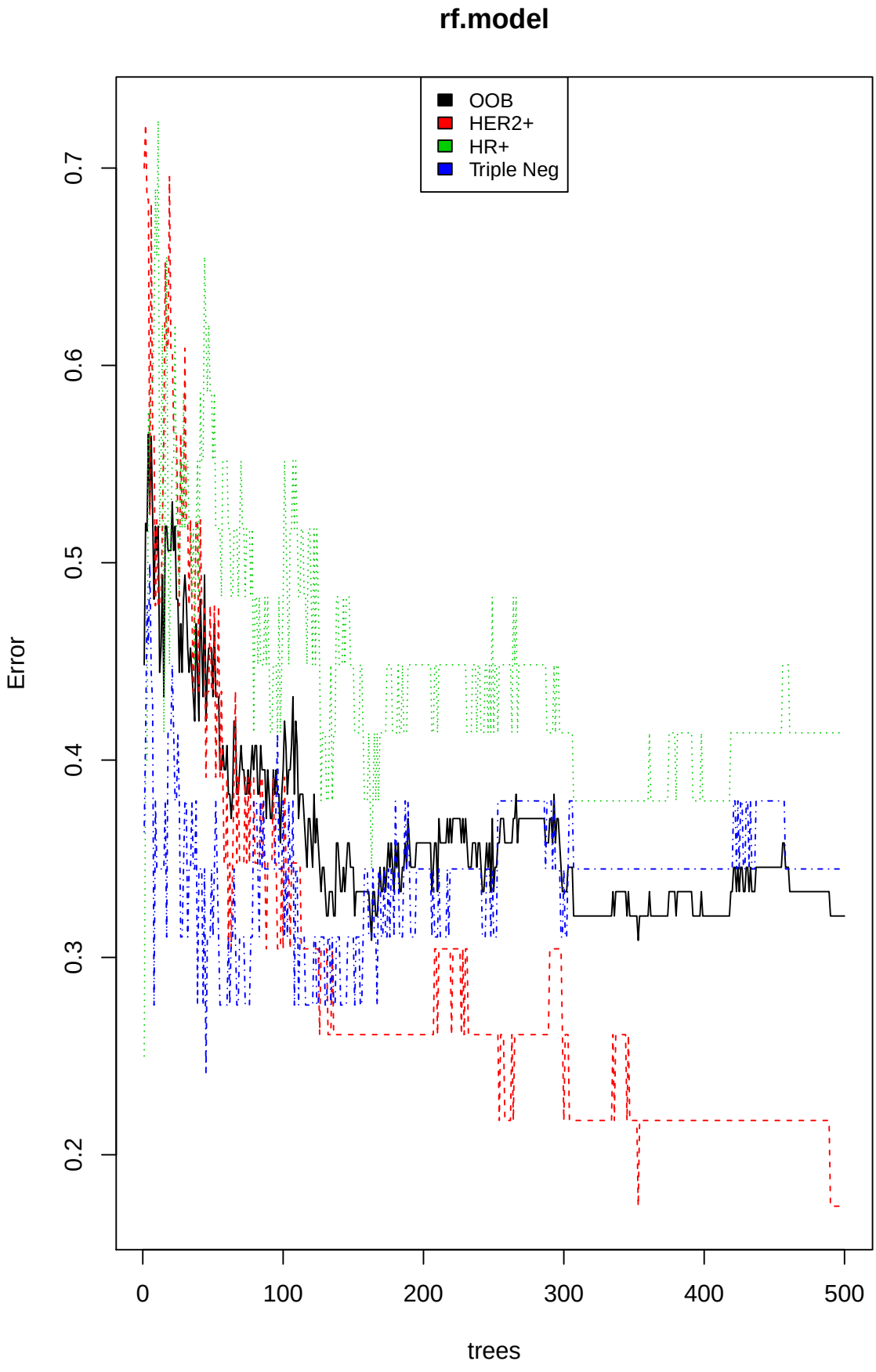
```
set.seed(130910)
is.train <- sample(c(TRUE,FALSE), nrow(train), replace = TRUE, prob = c(0.8,0.2))
test <- train[!is.train,] %>% as.data.frame()
train <- train[is.train,] %>% as.data.frame()

train$Sample <- NULL
test$Sample <- NULL
```

Fit a random forest, k-NN and support vector classifier.

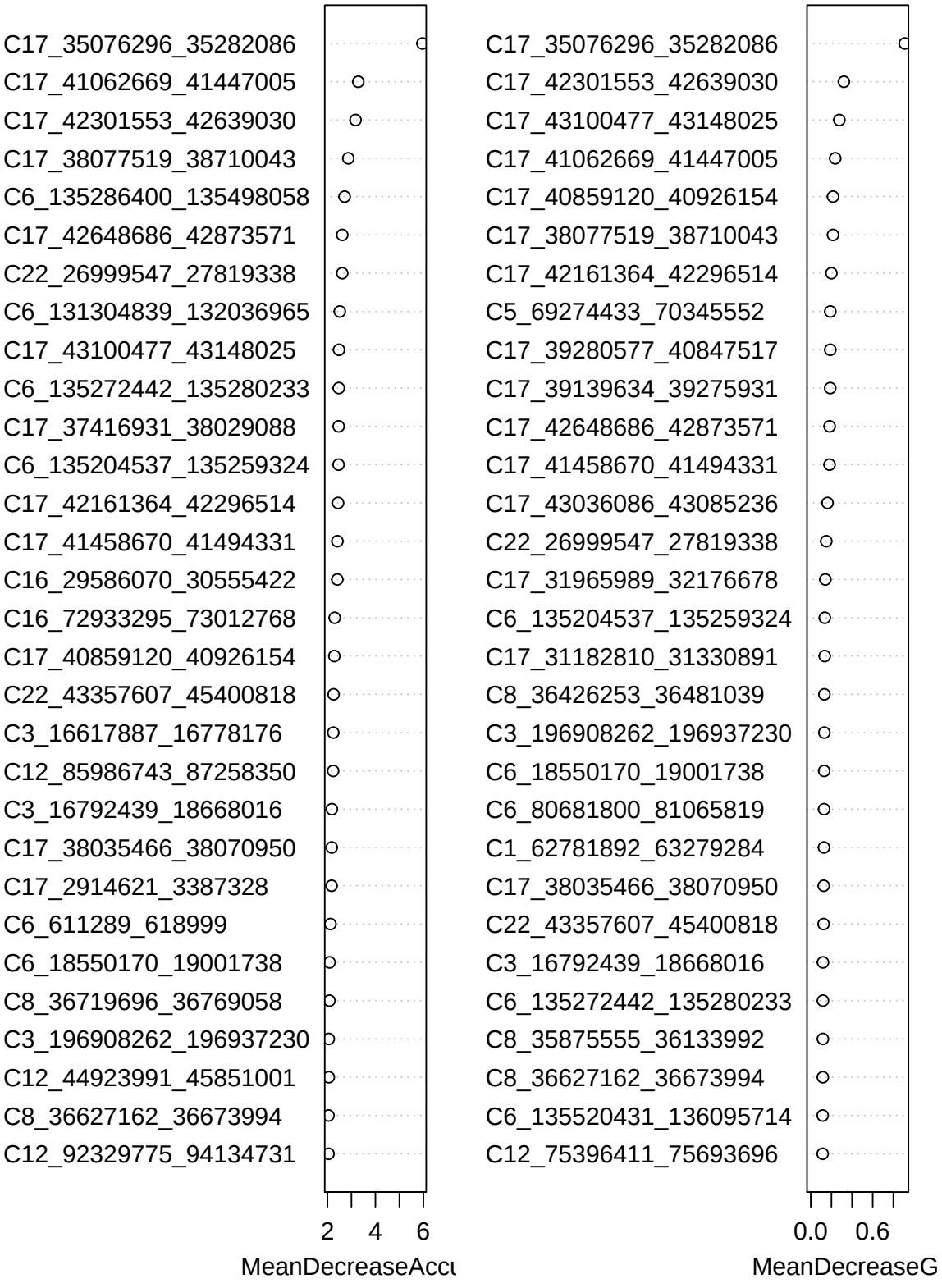
Random forest may be interesting as it's not been used by the authors of paper Iwan found. k-NN is a very naive but sometimes powerful feature. It suffers from high variance, however. The support vector classifier isn't tried in that same paper with a non-linear kernel. May be a good idea to try this too.

Finally, we will try a boosted decision tree classifier from the gbm package. Such boosting can be very powerful in reducing the bias of the classifier (but not necessarily the variance). It is prone to overfit but still it may be good to use it here as boosting allows for heterogeneous instances which is clearly something we are dealing with in the case of cancer cells; that was also the motivation for using the knowledge-based network as we saw in the lecture.

```r
set.seed(12)
rf.model <- randomForest(Subgroup ~ ., data=train, ntrees = 1000, importance=TRUE)
plot(rf.model)
legend("top", colnames(rf.model$err.rate),col=1:4,cex=0.8,fill=1:4)
```


rf.model
OOB
HER2+
HR+
Triple Neg
Error
0.2
0.3
0.4
0.5
0.6
0.7
0
100
200
300
400
500
trees

```r
varImpPlot(rf.model)
```


rf.model
C17_35076296_35282086
C17_41062669_41447005
C17_42301553_42639030
C17_38077519_38710043
C6_135286400_135498058
C17_42648686_42873571
C22_26999547_27819338
C6_131304839_132036965
C17_43100477_43148025
C6_135272442_135280233
C17_37416931_38029088
C6_135204537_135259324
C17_42161364_42296514
C17_41458670_41494331
C16_29586070_30555422
C16_72933295_73012768
C17_40859120_40926154
C22_43357607_45400818
C3_16617887_16778176
C12_85986743_87258350
C3_16792439_18668016
C17_38035466_38070950
C17_2914621_3387328
C6_611289_618999
C6_18550170_19001738
C8_36719696_36769058
C3_196908262_196937230
C12_44923991_45851001
C8_36627162_36673994
C12_92329775_94134731
2
4
6
MeanDecreaseAccu
C17_35076296_35282086
C17_42301553_42639030
C17_43100477_43148025
C17_41062669_41447005
C17_40859120_40926154
C17_38077519_38710043
C17_42161364_42296514
C5_69274433_70345552
C17_39280577_40847517
C17_39139634_39275931
C17_42648686_42873571
C17_41458670_41494331
C17_43036086_43085236
C22_26999547_27819338
C17_31965989_32176678
C6_135204537_135259324
C17_31182810_31330891
C8_36426253_36481039
C3_196908262_196937230
C6_18550170_19001738
C6_80681800_81065819
C1_62781892_63279284
C17_38035466_38070950
C22_43357607_45400818
C3_16792439_18668016
C6_135272442_135280233
C8_35875555_36133992
C8_36627162_36673994
C6_135520431_136095714
C12_75396411_75693696
0.0
0.6
MeanDecreaseG

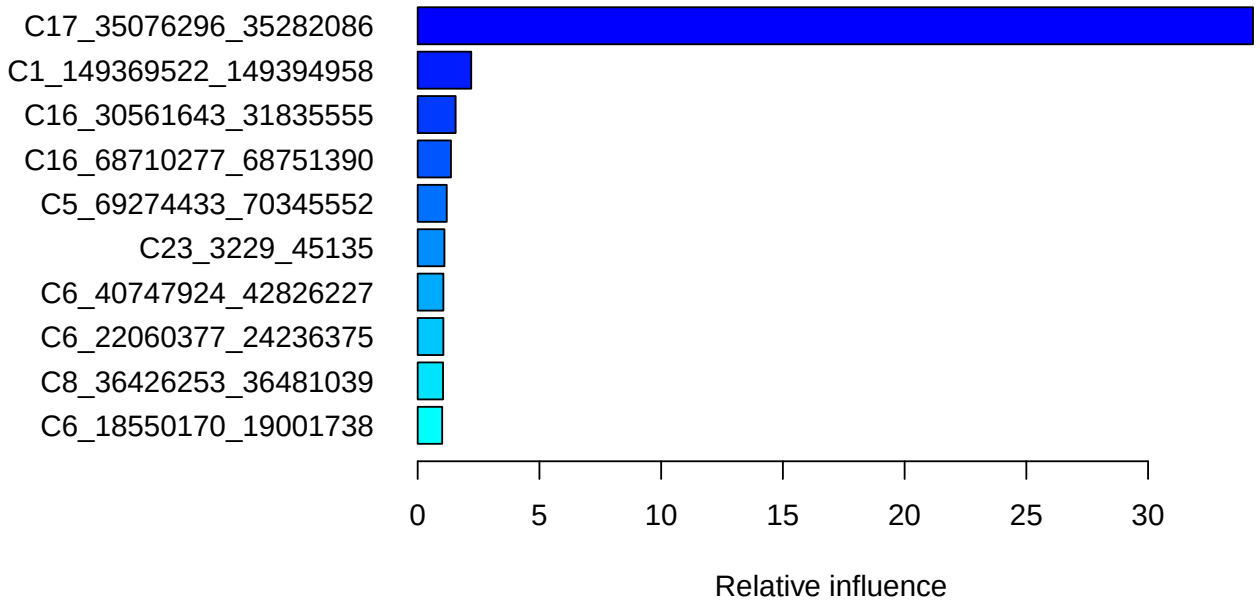
```
rf.predict <- predict(rf.model, test)
mean(test$Subgroup == rf.predict)
```

```
## [1] 0.6315789
```

Boosted decision trees. I took this implementation from some Rbloggers blog. Maybe we should look at the parameters too (but it works already quite well.) Summary shows one mutation that is really important for this cell (there is a one-to-one correspondence between this chromosome having a value 2 and it being a HER-2 cell if we inspect the validation and training data).

Train the boosted decision tree classifier (gbm).

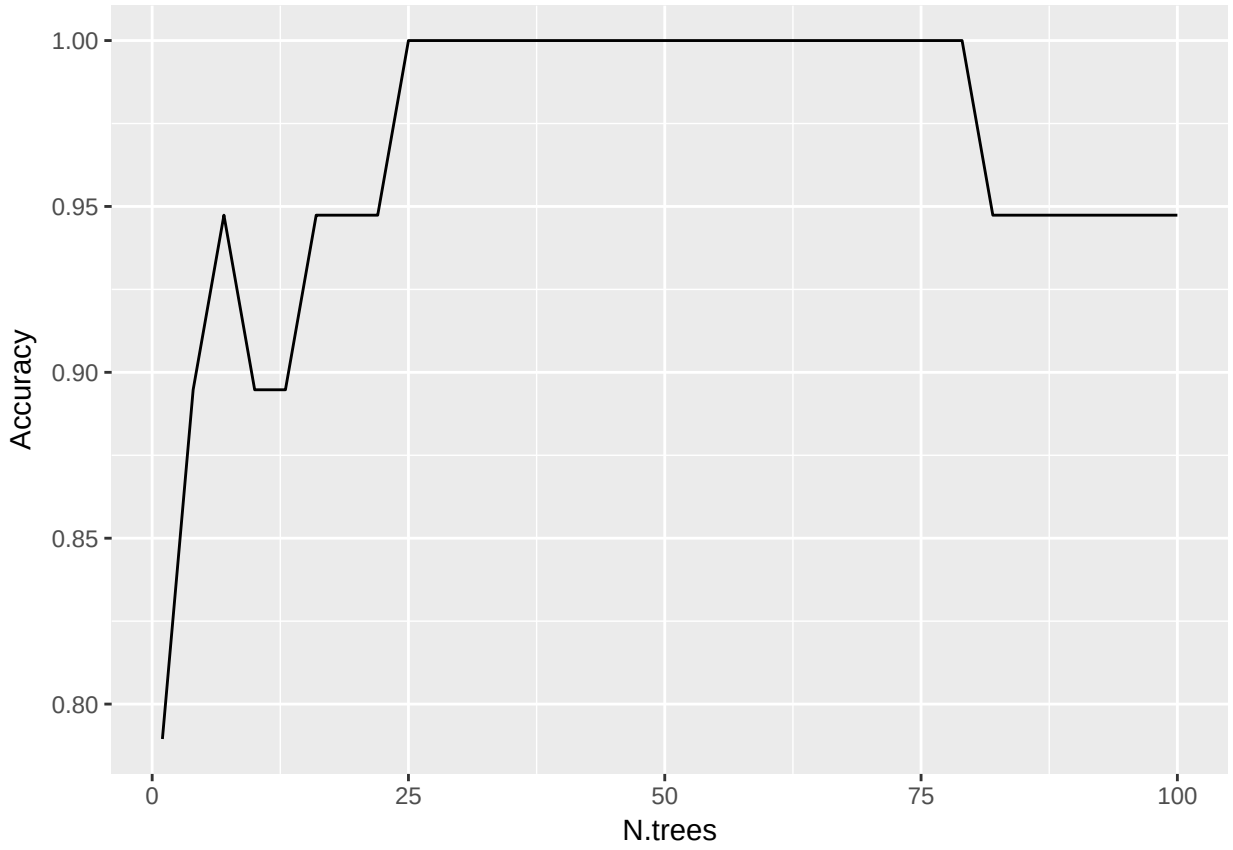
```
rf.boost = gbm(Subgroup ~ . ,data = train, distribution = "multinomial", n.trees = 10000, shrinkage = 0
par(oma=c(0,10,0,0))
output <-summary(rf.boost, cBars=10, order=TRUE, las=1)
```



Evaluate accuracy of thsi classifier on the test set. We see that it has extremely high accuracy. May be reflecting overfitting.

```
n.trees = seq(from=1 ,to=100, by=3) #no of trees-a vector of 100 values

#Generating a dataframe of predictions for the number of trees.
prob_df <- predict(rf.boost,test,n.trees = n.trees, type="response") %>% melt()
test_labels <- test$Subgroup %>% as.data.frame() %>% mutate(Sample = row_number())
names(test_labels) <- c("True_type","Sample")
names(prob_df) <- c("Sample","Type", "N.trees", "p_hat")
predictions <- prob_df %>%
  group_by(Sample, N.trees) %>%
  filter(p_hat == max(p_hat)) %>%
  arrange(N.trees, Sample, Type) %>%
  merge(test_labels, by=c("Sample")) %>%
  group_by(N.trees) %>%
  summarize(Accuracy = sum(Type == True_type)/n())

#Calculating The Mean squared Test Error
predictions %>% ggplot(aes(x=N.trees, y=Accuracy)) +
```

```
  geom_line()
```



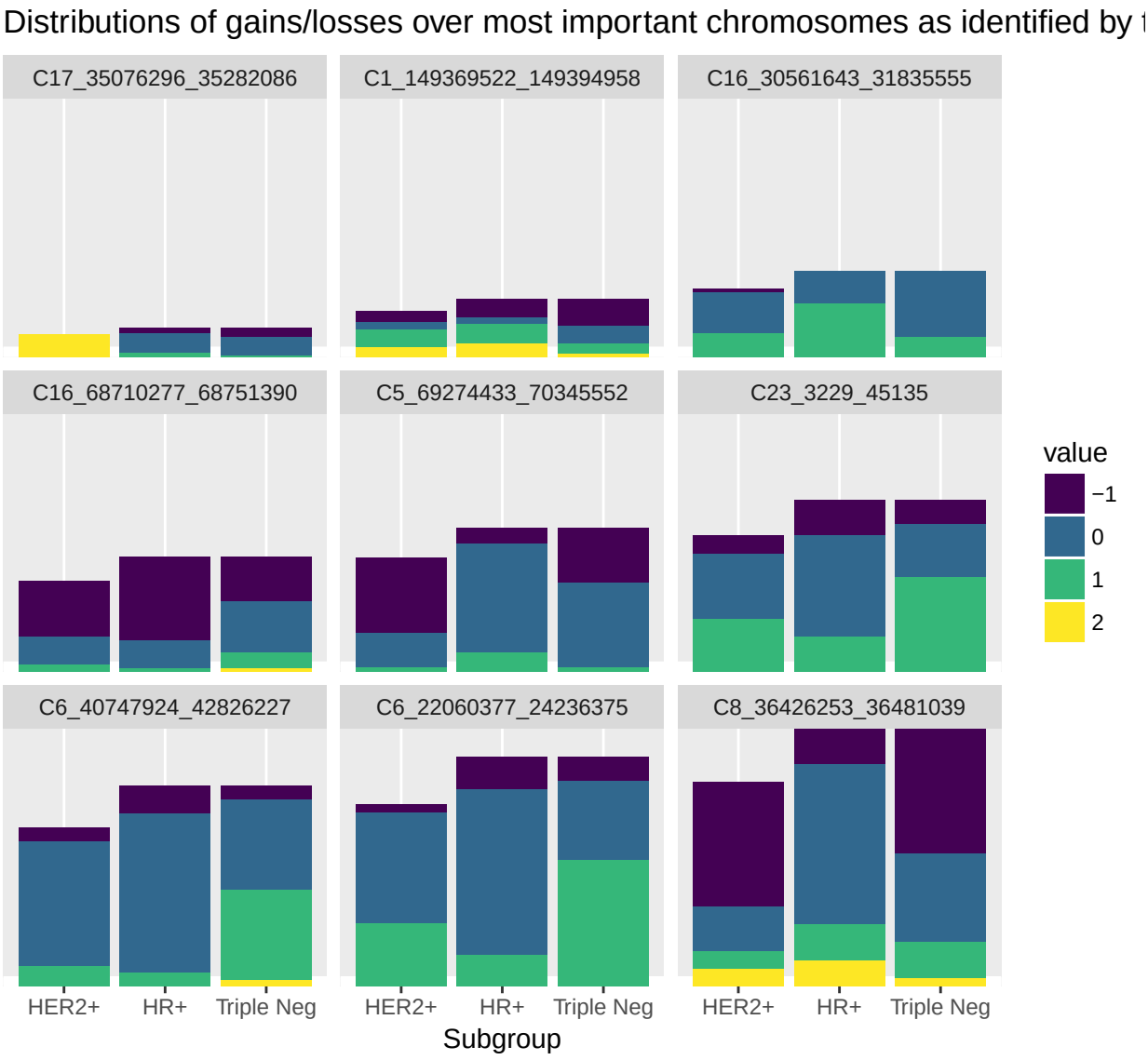
gbm() also reports on the importances of different features (i.e. chromosome regions). Let us plot the associations between the k most important features for test and training data. Note that there does not seem to be another magic marker that corresponds one-to-one to a cell-type.

```
k = 9
numRows = floor(k/3)
selected_cols <- output$var[1:k] %>% as.character()
test[c(selected_cols, "Subgroup")] %>% melt() %>%
  mutate(value = as.factor(value)) %>%
  ggplot(aes(x=Subgroup, y=variable, fill = value)) +
  geom_col() + facet_wrap( ~ variable, nrow = numRows) +
  theme(axis.ticks.y = element_blank(), axis.text.y=element_blank(), axis.title.y = element_blank()) +
  ggtitle("Distributions of gains/losses over most important chromosomes as identified by the GBM classi
  scale_fill_viridis(discrete=T)
```

```
## Using Subgroup as id variables
```


```
train[c(selected_cols, "Subgroup")] %>% melt() %>%
  mutate(value = as.factor(value)) %>%
  ggplot(aes(x=Subgroup, y=variable, fill = value)) +
  geom_col() + facet_wrap( ~ variable, nrow = numRows) +
  theme(axis.ticks.y = element_blank(), axis.text.y=element_blank(), axis.title.y = element_blank()) +
  ggtitle("Distributions of gains/losses over most important chromosomes as identified by the GBM classi
  scale_fill_viridis(discrete=T)
```

```
## Using Subgroup as id variables
```



The other classifiers

Very simple k-nearest neighbor classifier. Doesn't seem to work very well. May suffer from noise. I hypothesized a correlation-based measure would maybe be more appropriate. However, problem is that the matrix of observations isn't invertible so it does not work.... We could try some dimension reduction e.g. through PCA and then do it but not a fan.

```
nn3 <- knn(train[,-1], test[,-1], unlist(train$Subgroup),  k=5, prob=TRUE)
mean(nn3 == test$Subgroup)
```

```
## [1] 0.4736842
```

Fit an SVM classifier We will try both the linear and RBF kernel.

```
tuned_svm<-tune(svm, train.x=train[,-1], train.y = unlist(train$Subgroup),kernel="linear", range=list(c
print(tuned_svm)
```

```
##
```

```
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##  0.01
##
## - best performance: 0.3083333
```

```
svm_good_model<-svm(Subgroup~., data=train, kernel="linear",cost=tuned_svm$best.parameters$cost)
svm.predict <- predict(svm_good_model, test)
mean(svm.predict == test$Subgroup)
```

```
## [1] 0.7368421
```

Tuned SVM seems to be overfitting; performance on test set declines.

```
library(e1071)
tuned_svm<-tune(svm, train.x=train[,-1], train.y = unlist(train$Subgroup),kernel="radial", range=list(c
print(tuned_svm)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost        gamma
##     8 3.051758e-05
##
## - best performance: 0.2819444
```

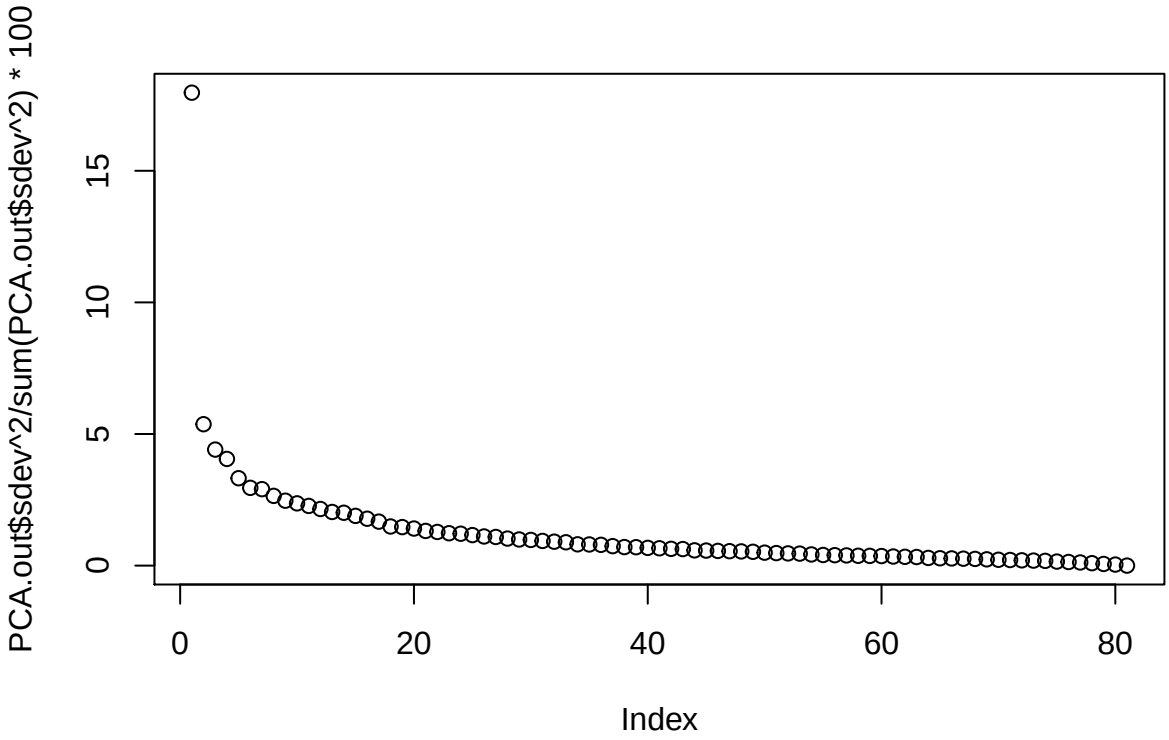
```
svm_good_model<-svm(Subgroup~., data=train, kernel="radial",cost=tuned_svm$best.parameters$cost)
svm.predict <- predict(svm_good_model, test)
mean(svm.predict == test$Subgroup)
```

```
## [1] 0.6842105
```

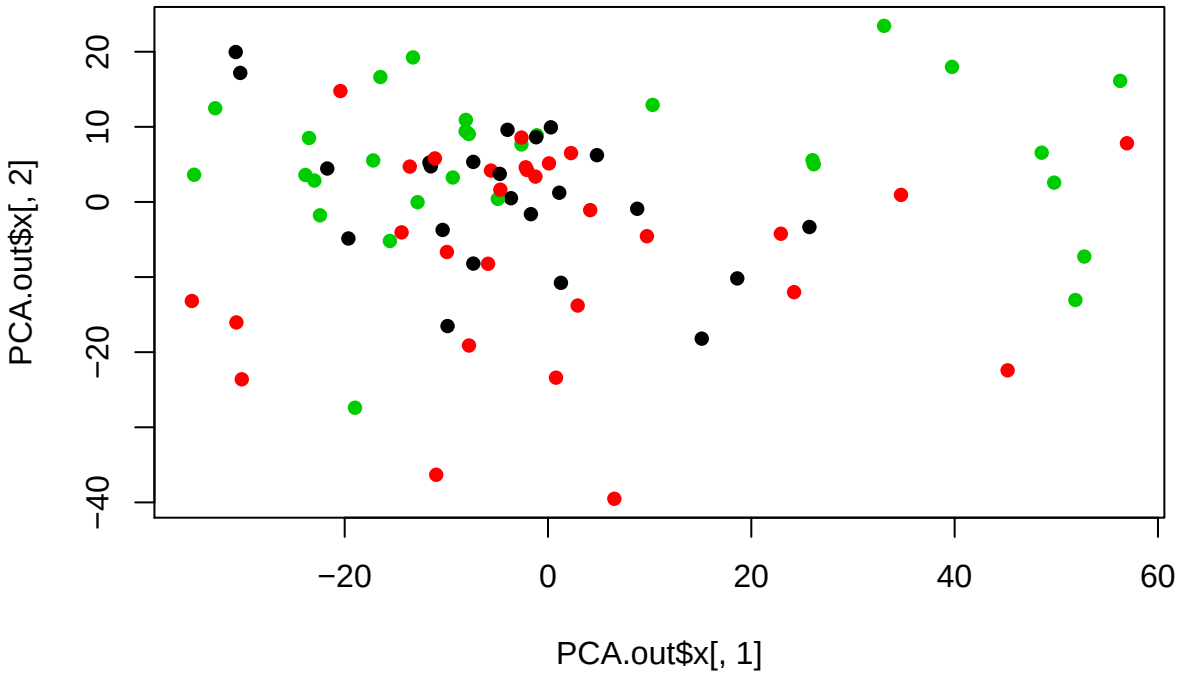
Scores on the first two principal components.

PCA doesn't yield clear patterns in the scores. Also, it's discussed in literature that it's only really appropriate when you think some underlying unobservable process generates the labels. Here, that seems a bit far-fetched.

```
PCA.out <- scale(train[,-1], center = TRUE, scale = TRUE) %>%  prcomp()
plot(PCA.out$sdev^2/sum(PCA.out$sdev^2)*100)
```




```
plot(PCA.out$x[,1], PCA.out$x[,2], col = train$Subgroup, pch=16)
```



Other ideas:

- Investigate if perfect association stems from clinical test. If so, discuss that it may be cheating.
- Try binary classification between the triple negative and HR+ cells to select features that distinguish the two.

•