Ted Tinker's Midterm

Ted Tinker

November 10, 2017

```
setwd("/Users/Theodore/Desktop/R_Studio_Stuff/data")
drug_use <- read_csv('drug.csv',
col_names = c('ID','Age','Gender','Education','Country','Ethnicity',
'Nscore','Escore','Oscore','Ascore','Cscore','Impulsive',
'SS','Alcohol','Amphet','Amyl','Benzos','Caff','Cannabis',
'Choc','Coke','Crack','Ecstasy','Heroin','Ketamine',
'Legalh','LSD','Meth','Mushrooms','Nicotine','Semer','VSA'))
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   ID = col_integer(),
##   Age = col_double(),
##   Gender = col_double(),
##   Education = col_double(),
##   Country = col_double(),
##   Ethnicity = col_double(),
##   Nscore = col_double(),
##   Escore = col_double(),
##   Oscore = col_double(),
##   Ascore = col_double(),
##   Cscore = col_double(),
##   Impulsive = col_double(),
##   SS = col_double()
## )
```

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

Exercise One

```
drug_use <- drug_use %>% mutate_at(as.ordered, .vars=vars(Alcohol:VSA))
drug_use <- drug_use %>%
mutate(Gender = factor(Gender, labels=c("Male", "Female"))) %>%
mutate(Ethnicity = factor(Ethnicity, labels=c("Black", "Asian", "White",
"Mixed:White/Black", "Other",
"Mixed:White/Asian",
"Mixed:Black/Asian"))) %>%
mutate(Country = factor(Country, labels=c("Australia", "Canada", "New Zealand",
"Other", "Ireland", "UK", "USA")))
```

This code is given in the homework to factorise the irregularly normalized dataset.

A

```
drug_use <- cbind(mutate(drug_use,recent_cannabis_use= factor(
                                    ifelse(Cannabis %in% c("CL3","CL4","CL5","CL6"),"Yes","No"))))
```

Adds a column of boolean factors based on the column Cannabis.

B

```
seed=(100)  # For consistency
drug_use_subset <- select(drug_use,Age:SS, recent_cannabis_use)   # Choose columns
train.indices = sample(1:nrow(drug_use_subset), 1500)             # Randomize sample
drug_use_train=drug_use_subset[train.indices,]                    # Set training
drug_use_test=drug_use_subset[-train.indices,]                    # Set test

dim(drug_use_train)    # Print sizes
```

```
## [1] 1500   13
```

```
dim(drug_use_test)
```

```
## [1] 385  13
```

The resulting data-sets are of the expected sizes.

C

```
drug.log <- glm(recent_cannabis_use~.,family="binomial", data=drug_use_train) # Make GL Model
summary(drug.log)
```

```
##
## Call:
## glm(formula = recent_cannabis_use ~ ., family = "binomial", data = drug_use_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9113  -0.5980   0.1655   0.5384   2.5879
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   0.63381    0.77085   0.822 0.410953
## Age                          -0.82474    0.09162  -9.002  < 2e-16 ***
## GenderFemale                 -0.66402    0.15686  -4.233 2.30e-05 ***
## Education                    -0.30387    0.07872  -3.860 0.000113 ***
## CountryCanada                -0.61014    1.24089  -0.492 0.622932
## CountryNew Zealand           -0.92024    0.32281  -2.851 0.004362 **
## CountryOther                 -0.42073    0.47396  -0.888 0.374709
## CountryIreland               -0.16162    0.73500  -0.220 0.825958
## CountryUK                    -0.35753    0.37665  -0.949 0.342509
## CountryUSA                   -1.87725    0.19640  -9.558  < 2e-16 ***
## EthnicityAsian               -0.94519    1.04536  -0.904 0.365901
## EthnicityWhite                1.25219    0.76203   1.643 0.100334
## EthnicityMixed:White/Black    0.42290    1.22358   0.346 0.729623
```

```
## EthnicityOther               1.17693    0.87295   1.348 0.177589
## EthnicityMixed:White/Asian   1.47085    1.10247   1.334 0.182158
## EthnicityMixed:Black/Asian  11.66034  377.24769   0.031 0.975342
## Nscore                      -0.15909    0.09151  -1.738 0.082131 .
## Escore                      -0.18231    0.09591  -1.901 0.057319 .
## Oscore                       0.66394    0.09158   7.250 4.17e-13 ***
## Ascore                       0.04360    0.08083   0.539 0.589559
## Cscore                      -0.38680    0.09125  -4.239 2.24e-05 ***
## Impulsive                   -0.12389    0.10048  -1.233 0.217577
## SS                           0.58760    0.11261   5.218 1.81e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2073.5  on 1499  degrees of freedom
## Residual deviance: 1196.4  on 1477  degrees of freedom
## AIC: 1242.4
##
## Number of Fisher Scoring iterations: 12
```

D

As a binomial general linear model, the predictions are made in log-odds form. Given a male's predictors, and finding his log-odds P, a female with the same predictors would have log-odds $P - 0.78153$. Therefore, the female's *odds* (as opposed to log-odds) would be lower than the man's. If the female were 2 age-units older than the male, they would have log-odds $P - 0.78153 - 2 \times -0.90433$.

Exercise Two

```
drugTree <- tree(recent_cannabis_use~.,data=drug_use_train,
                 control=tree.control(nobs=nrow(drug_use_train), minsize=10, mindev=1e-3))
```

This code is given in the homework

A

```
cv.tree(drugTree) # Default values should be right for 10-fold validation
```

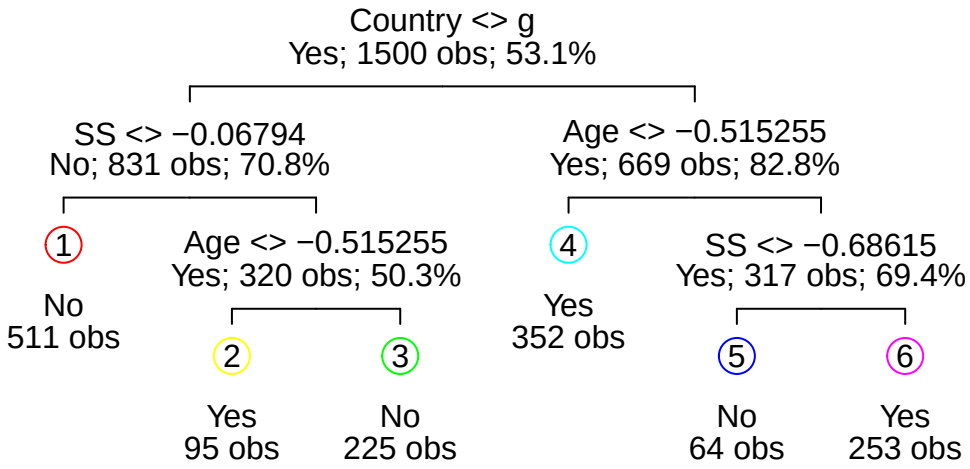
```
## $size
##  [1] 137 136 135 134 133 132 131 130 129 128 127 126 125 124 123 122 120
## [18] 119 117 116 114 111 109 108 107 104 103 101 100  99  98  96  94  91
## [35]  88  87  81  79  78  73  71  67  66  65  64  62  61  57  53  52  47
## [52]  46  43  41  40  38  37  35  33  32  31  25  23  22  21  19  18  16
## [69]  15  12  11  10   9   8   7   6   5   4   3   2   1
##
## $dev
##  [1] 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983
##  [8] 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983
## [15] 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983
```

```
## [22] 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983
## [29] 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983
## [36] 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983
## [43] 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983
## [50] 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983
## [57] 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983
## [64] 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983 1435.983
## [71] 1435.983 1435.983 1435.983 1438.946 1441.454 1426.672 1419.609
## [78] 1449.774 1513.619 1620.606 2074.548
##
## $k
##  [1]       -Inf   2.200898   2.283748   2.343877   2.405460   2.516009
##  [7]   2.551524   2.602390   2.639130   2.800711   2.804595   2.936585
## [13]   3.036627   3.147436   3.157113   3.255734   3.277932   3.309826
## [19]   3.402102   3.550147   3.653650   3.700948   3.727521   3.757334
## [25]   3.779441   3.826087   3.943441   3.947081   4.062496   4.159609
## [31]   4.337564   4.534491   4.544172   4.557574   4.573118   4.604623
## [37]   4.612110   5.018021   5.193052   5.242714   5.466229   5.470137
## [43]   5.487169   5.524078   5.531537   5.540350   5.725206   6.022324
## [49]   6.053354   6.054835   6.085054   6.307100   6.316554   6.477736
## [55]   6.584117   6.698128   6.833136   6.930662   7.250739   7.651019
## [61]   7.770406   7.778858   7.844541   7.859559   8.586615   8.703787
## [67]   9.113858   9.683343   9.788311  10.900600  13.947082  15.963340
## [73]  19.077240  19.333080  19.896506  22.236541  27.689589  39.388860
## [79]  81.431755 110.612543 455.208837
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

The best size for the tree seems to fluctuate when the code is run repeatedly, despite setting the seed. The best size hovers around 6 leaf nodes, where the deviation is generally minimal with minimal size.

B

```
drugTree <- prune.tree(drugTree,best=6)  # Prune tree
draw.tree(drugTree,nodeinfo=TRUE)        # Draw tree
```



The first variable to be split is Country.

C

```
table(drug_use_test$recent_cannabis_use, predict(drugTree,drug_use_test[,-13],type="class")) # Conf
```

```
##
##         No Yes
##    No  158  25
##    Yes  50 152
```

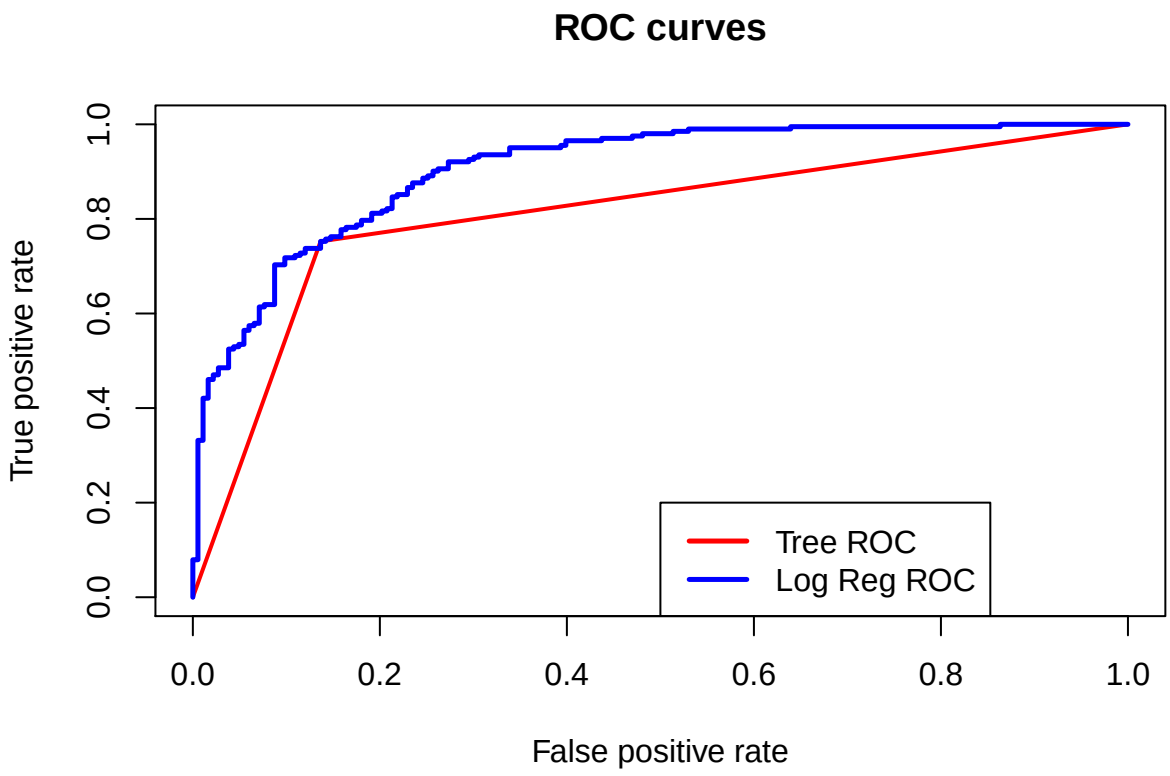
The True Positive Rate is equal to the number of True Positives divided by the number of True Positives plus False Negatives. In this confusion matrix, True Positives are represented by "Yes," "Yes." False Negatives are represented by "Yes," "No." So (using a particular iteration of the code because repeated iterations seem to fluctuate), $TPR = \frac{147}{147+64} \approx .69668$.

Similarly, The False Positive Rate is equal to the number of False Positives divided by the number of False Positives plus True Negatives. In this confusion matrix, False positives are represented by "No," "Yes." True negatives are represented by "No," "No." So, $FPR = \frac{25}{149+25} \approx .14369$.

Exercise Three

A

```
pred.tree <- predict(drugTree,drug_use_test[,-13],type="class") # Predict test values with tree
pred.log <- predict(drug.log,drug_use_test[,-13], type="response")    # Predict test values with logist
pred1<-prediction(as.numeric(pred.tree),as.numeric(drug_use_test$recent_cannabis_use))
pred2<-prediction(as.numeric(pred.log),as.numeric(drug_use_test$recent_cannabis_use))
perftree= performance(pred1, measure="tpr", x.measure="fpr")     # Measaure performance of tree
perflog=performance(pred2,measure = "tpr",x.measure = "fpr")     # Measure performance of logreg
plot(perftree, col="red", lwd=2, main="ROC curves")
plot(perflog,col="blue",lwd=2.5,add=TRUE)                        # Plot FPR vs TPR
legend(.5,.2, c("Tree ROC","Log Reg ROC"), lty=c(1,1),lwd=c(2.5,2.5),col=c("red","blue"))
```



The prediction tree's ROC curve is simple and geometric. The Logistic Regression's ROC curve is more typically curved.

B

```
auctree = performance(pred1, "auc")@y.values  # Measure performances and print
auclog=performance(pred2,"auc")@y.values
print(auctree)
```

```
## [[1]]
## [1] 0.8079316
```

```
print(auclog)
```

```
## [[1]]
## [1] 0.9052102
```

The prediction tree has a much lower AUC than the logistic regression. We might say this means the logistic regression is "better" regarding TPR and FPR—though the tree might be preferable in a situation where interpretability was most important.

Exercise Four

We do not include the reading-in of data as it is far too lengthy.

A

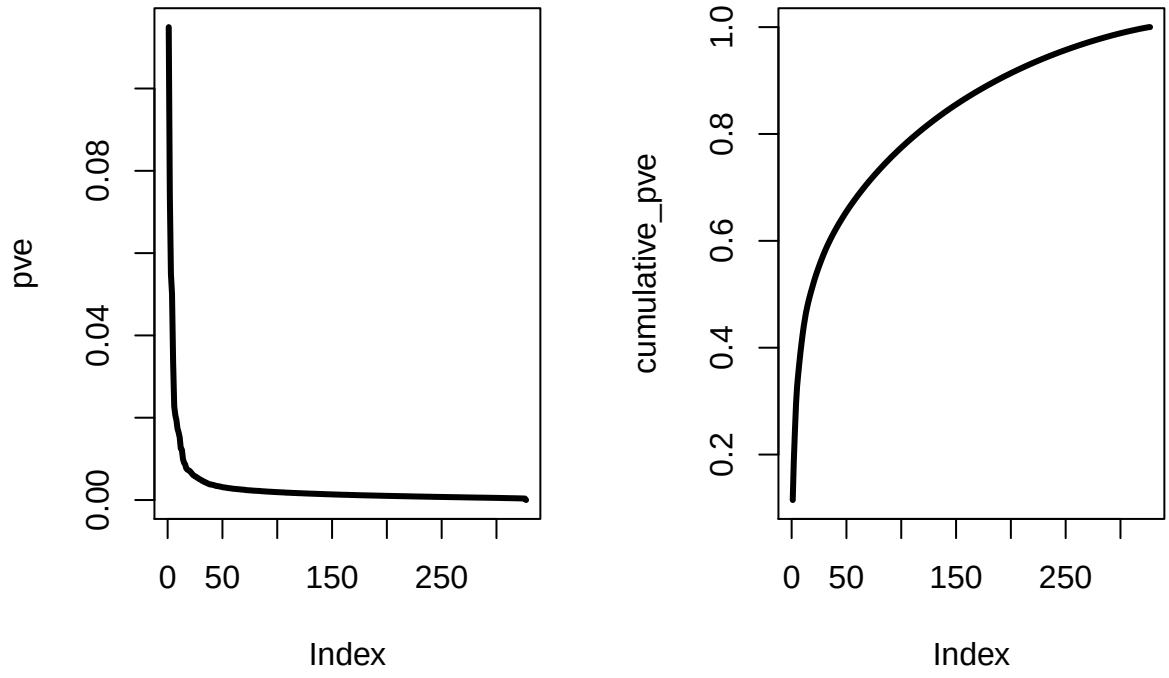
```
table(leukemia_data$Type)
```

```
##
##    BCR-ABL   E2A-PBX1 Hyperdip50        MLL     OTHERS      T-ALL
##         15         27         64         20         79         43
##   TEL-AML1
##         79
```

The most common types of leukemia are TEL-AML1 and "Others." The least common type is BCR-ABL, with only 15 observations.

B

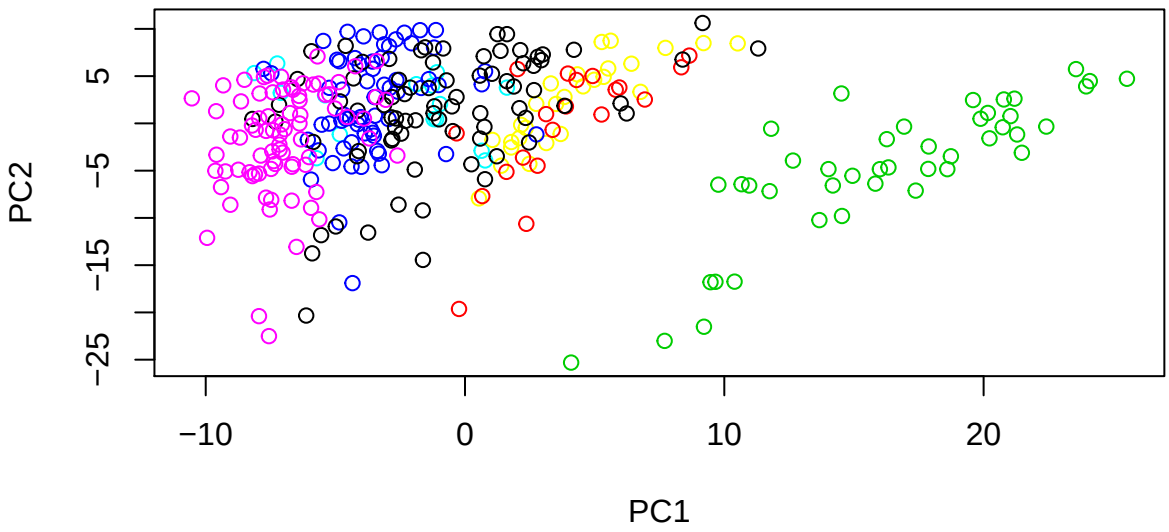
```
pve <- (leuk.PCA$sdev^2) / sum(leuk.PCA$sdev^2)
cumulative_pve <- cumsum(leuk.PCA$sdev^2) / sum(leuk.PCA$sdev^2)
## This will put the next two plots side by side
par(mfrow=c(1, 2))
## Plot proportion of variance explained
plot(pve, type="l", lwd=3)
plot(cumulative_pve, type="l", lwd=3)
```



As expected, the first few components represent the most variance. It takes less than fifty columns to account for more than half the information.

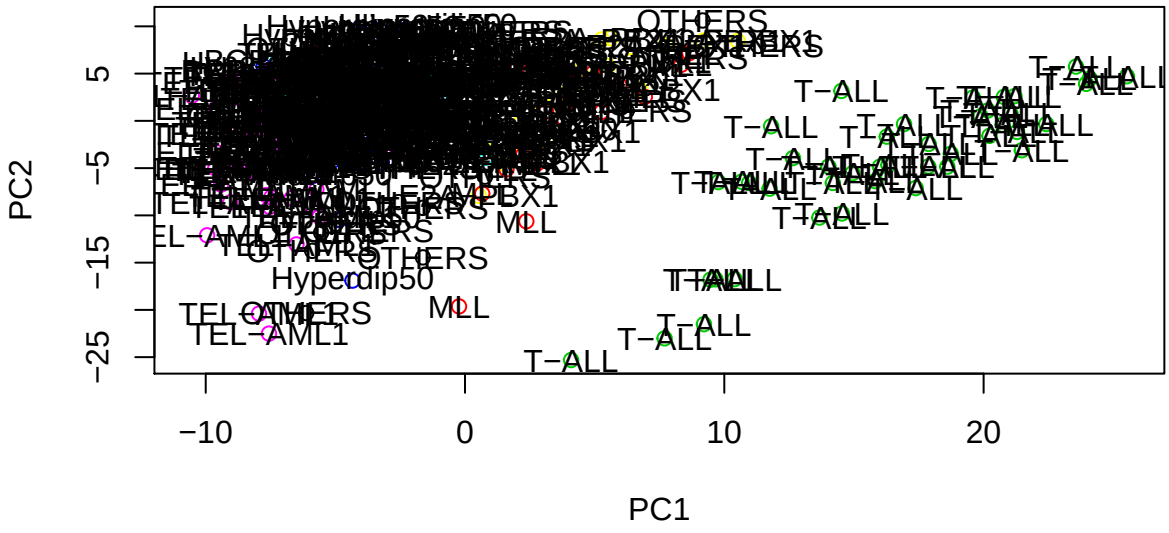
C

```
rainbow_colors <- rainbow(7)
plot_colors <- # The provided code did not work for me. I use a different method.
   with(leukemia_data,
        data.frame(Type = c("BCR-ABL","E2A-PBX1","Hyperdip50","MLL","OTHERS","T-ALL","TEL-AML1"),
                   color = rainbow_colors))
plot(PC2~PC1,leuk.PCA$x[,1:2],col=plot_colors$color[match(leukemia_data$Type, plot_colors$Type)])
```



Bands of color are generally oriented vertically, but slant toward the upper-right corner as PC1 increases. The green band is visibly shifted rightward compared to the other colors.

```
plot(PC2~PC1,leuk.PCA$x[,1:2],col=plot_colors$color[match(leukemia_data$Type, plot_colors$Type)])
with(leuk.PCA,text(x=leuk.PCA$x[,1],y=leuk.PCA$x[,2],labels=leukemia_data$Type)) # Add labels
```



Judging from the labels, the green band represents T-ALL.

```
absoluteLoadings <- leuk.PCA$rotation[sort.list(abs(as.double(
  leuk.PCA$rotation[,1])),decreasing=TRUE),1] # Get loadings, make positive, and sort.
head(absoluteLoadings,6)
```

```
##        CTGF    HLA-DPB1       TERF2     HLA-DMA     POU2AF1       CD79A
## -0.10868454 -0.10673306 -0.10490604 -0.09934950 -0.09221272 -0.09120526
```

(Professor, thank you for your advice after class—I misunderstood the way order() returns the ordering of the set.) The six most influential and highest-weighted variables represented by PC1 are listed above, with their weights. On a lark, I've elected to find the most influential loadings in PC2 as well:

```
absoluteLoadings2 <- leuk.PCA$rotation[sort.list(abs(as.double(
  leuk.PCA$rotation[,2])),decreasing=TRUE),2]
head(absoluteLoadings2,6)
```

```
##    32815_at       HHLA1         NF2      TPTEP1      ERVH-1     1937_at
## -0.08265765 -0.06973381 -0.06939831 -0.06555561 -0.06362221 -0.06125024
```

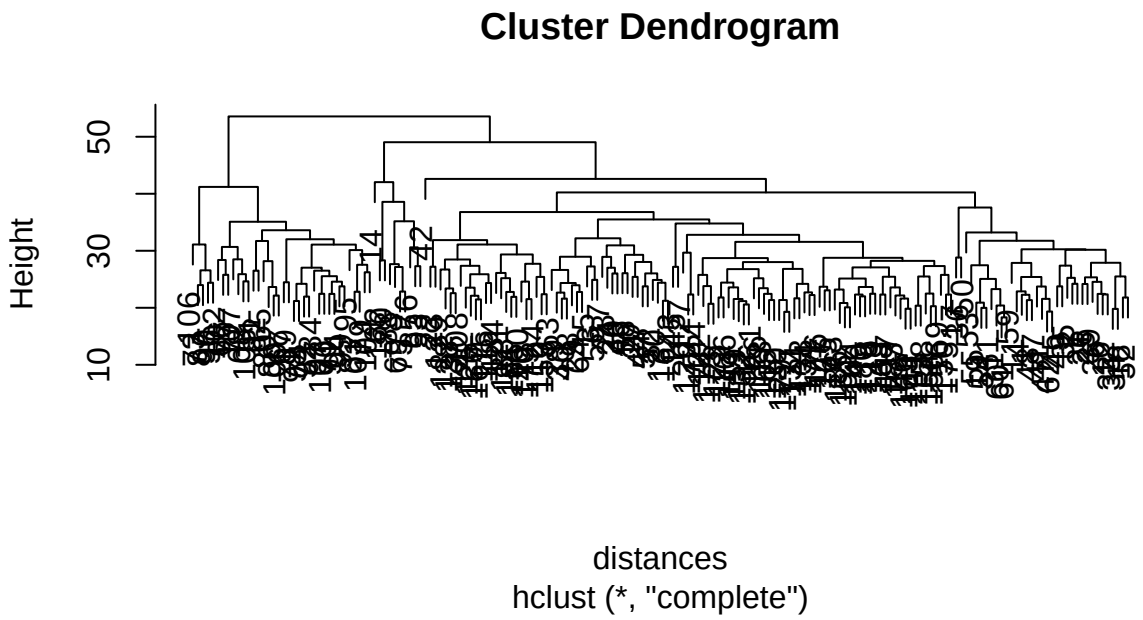
None of the six most heavily weighted variables are shared. I suppose this makes sense, because PC1 and PC2 are meant to be uncorrelated.

F

```
leukemia_subset <- dplyr::filter(leukemia_data, Type %in% c("T-ALL","TEL-AML1","Hyperdip50"))
# Be sure to use dplyr::filter not default filter
distances <- dist(leukemia_subset,diag=TRUE,upper=TRUE) # Defaults to Euclidean
```

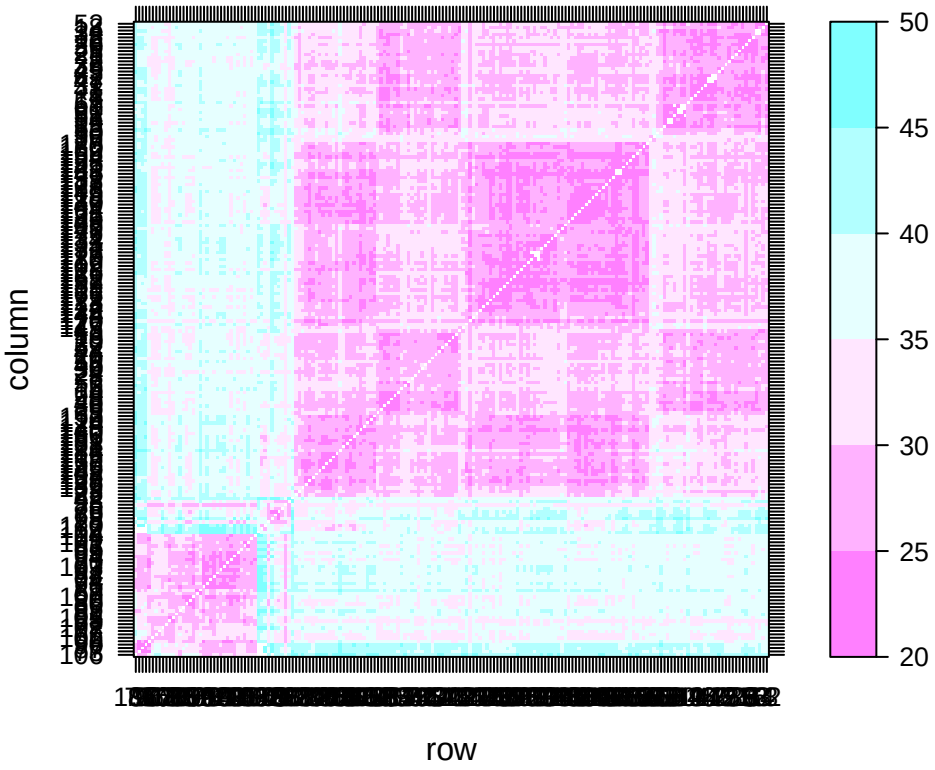
```
## Warning in dist(leukemia_subset, diag = TRUE, upper = TRUE): NAs introduced
## by coercion
```

```
leuk.clust <- hclust(distances)    # Defaults to complete linkages
plot(leuk.clust) # Plot Dendogram
```



G

```
distances <- as.matrix(distances) # Transform from vector
levelplot(distances[leuk.clust$order,leuk.clust$order], at=pretty(c(20,50),n=10)) # Make plot
```




```
leukemia_subset$Type[leuk.clust$order]
```

```
##   [1] "T-ALL"      "T-ALL"      "T-ALL"      "T-ALL"      "T-ALL"
##   [6] "T-ALL"      "T-ALL"      "T-ALL"      "T-ALL"      "T-ALL"
##  [11] "T-ALL"      "T-ALL"      "T-ALL"      "T-ALL"      "T-ALL"
##  [16] "T-ALL"      "T-ALL"      "T-ALL"      "T-ALL"      "T-ALL"
##  [21] "T-ALL"      "T-ALL"      "T-ALL"      "T-ALL"      "T-ALL"
##  [26] "T-ALL"      "T-ALL"      "T-ALL"      "T-ALL"      "T-ALL"
##  [31] "T-ALL"      "T-ALL"      "T-ALL"      "T-ALL"      "T-ALL"
##  [36] "T-ALL"      "Hyperdip50" "TEL-AML1"   "TEL-AML1"   "T-ALL"
##  [41] "T-ALL"      "T-ALL"      "T-ALL"      "T-ALL"      "T-ALL"
##  [46] "T-ALL"      "Hyperdip50" "Hyperdip50" "Hyperdip50" "TEL-AML1"
##  [51] "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"
##  [56] "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"
##  [61] "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"
##  [66] "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"
##  [71] "TEL-AML1"   "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50"
##  [76] "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50"
##  [81] "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50"
##  [86] "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50"
##  [91] "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50"
```

```
##  [96] "Hyperdip50" "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"
## [101] "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"
## [106] "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"
## [111] "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"
## [116] "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"
## [121] "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"
## [126] "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"
## [131] "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"
## [136] "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"
## [141] "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"
## [146] "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"   "TEL-AML1"
## [151] "TEL-AML1"   "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50"
## [156] "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50"
## [161] "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50"
## [166] "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50"
## [171] "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50"
## [176] "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50"
## [181] "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50" "Hyperdip50"
## [186] "Hyperdip50"
```

The "T-All" observations are almost contained in one continuous chunk, but "TEL-AML1" and "Hyperdip50" are somewhat mixed. The latter two are probably more similar in terms of their genetic distribution than either of them are to "T-All."