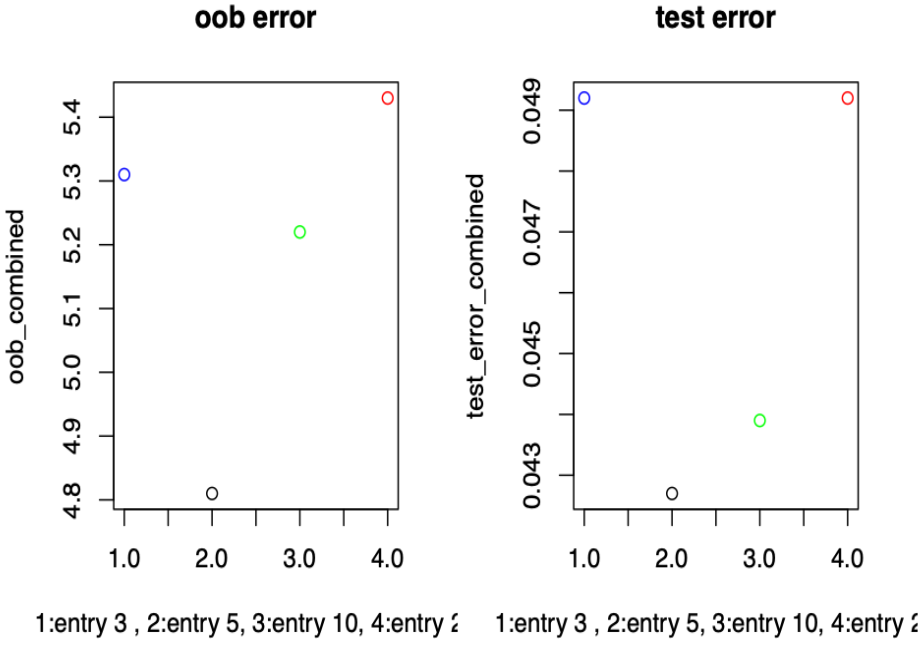
write up for hw5

Hyungkyu Lim

1)

There are plots for oob error and test error in terms of the number of randomly selected inputs for each tree. Total entries that I randomly chosen are 4; entry 3, entry 5, entry 10, entry 20. The maximum of the oob error is 5.43(entry 20) and the minimum of the oob error is 4.81(entry 5). The maximum of tbe test error is 0.0492(entry 3 and entry 20) and the minimum of the test error is 0.0427(entry 5). Based on two plots, entry 5 has minimum values for both plots.



2) Based on the test error of neural network method, I think that newural network method is better than the additive model.

```
## [1] 0.0652
```

3)

I selected the 0.01, 1, 5 and 10 for cost. With this, I generated two plots for train error and test error in terms of linear kernel. As the cost incrases, train error decreases however, there is no specific pattren of test error. Optimal cost of linear kernel is 0.01. In radial kernel, there is no specific pattern for two plots. However interesting thing is that most errors for both are about 0. With this, optimal cost of radial kernel is same with linear kernel. For polynomial plots, the train error is about 0.19 and the test error is about 0.11. Three kinds of kernels all have a good performance but I guess radial kernel is just a little bit better than linear and polynomial kernel.

```
##
## Call:
## svm(formula = Purchase ~ ., data = oj.train, kernel = "linear",
##     cost = 0.01)
```

```
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##       gamma:  0.05555555556
##
## Number of Support Vectors:  533
##
##  ( 266 267 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
##     train.pred
##       CH  MM
##   CH 539  75
##   MM  98 288
```

```
## [1] 0.173
```

```
##     test.pred
##      CH MM
##   CH 36  3
##   MM  2 29
```

```
## [1] 0.071
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = oj.train, kernel = "linear",
##     cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##       gamma:  0.05555555556
##
## Number of Support Vectors:  425
##
##  ( 212 213 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
##     train.pred_1
##       CH  MM
##   CH 537  77
```

```
##   MM  93 293
```

```
## [1] 0.17
```

```
##     test.pred_1
##      CH MM
##   CH 35  4
##   MM  2 29
```

```
## [1] 0.08571428571
```

```
## [1] 0.09
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = oj.train, kernel = "linear",
##     cost = 5)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  5
##       gamma:  0.05555555556
##
## Number of Support Vectors:  423
##
##  ( 211 212 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
##     train.pred_2
##       CH  MM
##   CH 541  73
##   MM  94 292
```

```
## [1] 0.167
```

```
##     test.pred_2
##      CH MM
##   CH 37  2
##   MM  3 28
```

```
## [1] 0.07142857143
```

```
## [1] 0.071
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = oj.train, kernel = "linear",
##     cost = 10)
##
##
## Parameters:
##    SVM-Type:  C-classification
```

```
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.05555555556
##
## Number of Support Vectors:  423
##
##  ( 211 212 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

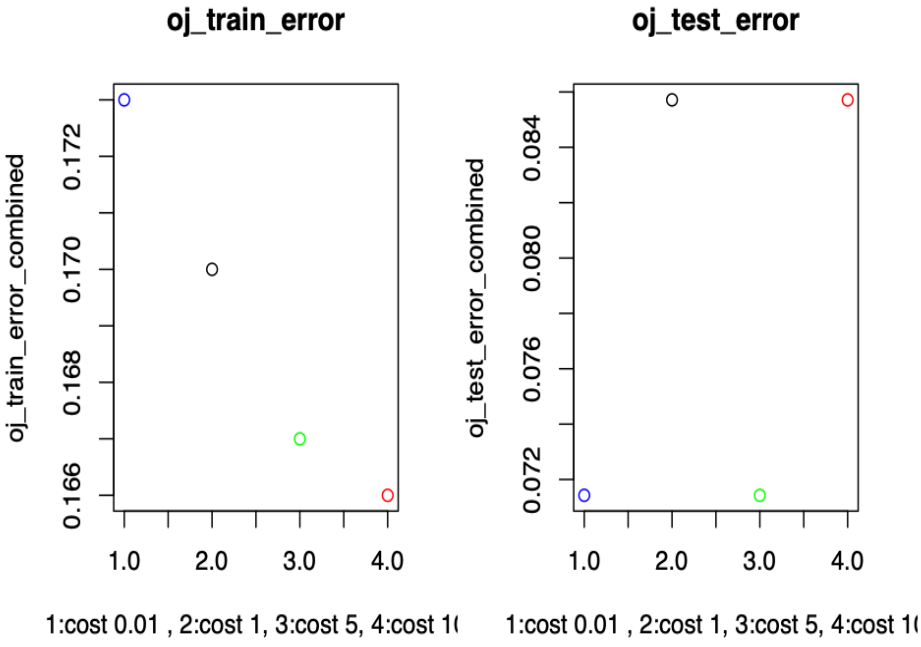
```
##     train.pred_3
##       CH  MM
##   CH 538  76
##   MM  90 296
```

```
## [1] 0.166
```

```
##     test.pred_3
##      CH MM
##   CH 37  2
##   MM  4 27
```

```
## [1] 0.08571428571
```

```
## [1] 0.086
```




```
##
## Parameter tuning of 'svm':
```

```
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.171
##
## - Detailed performance results:
##             cost error    dispersion
## 1   0.01000000000 0.178 0.05050852513
## 2   0.01778279410 0.178 0.05493430420
## 3   0.03162277660 0.173 0.05056349144
## 4   0.05623413252 0.176 0.05037636130
## 5   0.10000000000 0.177 0.04854551129
## 6   0.17782794100 0.173 0.05121848625
## 7   0.31622776602 0.174 0.05232377832
## 8   0.56234132519 0.173 0.05229191567
## 9   1.00000000000 0.174 0.04812021982
## 10  1.77827941004 0.174 0.05015531433
## 11  3.16227766017 0.173 0.04967673276
## 12  5.62341325190 0.172 0.04802776974
## 13 10.00000000000 0.171 0.05087020521
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = oj.train, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.05555555556
##
## Number of Support Vectors:  465
##
##  ( 231 234 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
##     train.pred
##       CH  MM
##   CH 539  75
##   MM  98 288
```

```
## [1] 0.158
```

```
##     test.pred_radial
##      CH MM
##   CH 36  3
```

```
##   MM  4 27
```

```
## [1] 0.1
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = oj.train, kernel = "radial",
##     cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.05555555556
##
## Number of Support Vectors:  465
##
##  ( 231 234 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
##     train.pred_1_radial
##       CH  MM
##   CH 558  56
##   MM 102 284
```

```
## [1] 0.158
```

```
##     test.pred_1_radial
##      CH MM
##   CH 36  3
##   MM  4 27
```

```
## [1] 0.1
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = oj.train, kernel = "radial",
##     cost = 5)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  5
##       gamma:  0.05555555556
##
## Number of Support Vectors:  427
##
##  ( 208 219 )
##
##
```

```
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
##     train.pred_2_radial
##       CH  MM
##   CH 563  51
##   MM 102 284
```

```
## [1] 0.153
```

```
##     test.pred_2_radial
##      CH MM
##   CH 36  3
##   MM  4 27
```

```
## [1] 0.1
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = oj.train, kernel = "radial",
##     cost = 10)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.05555555556
##
## Number of Support Vectors:  415
##
##  ( 203 212 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

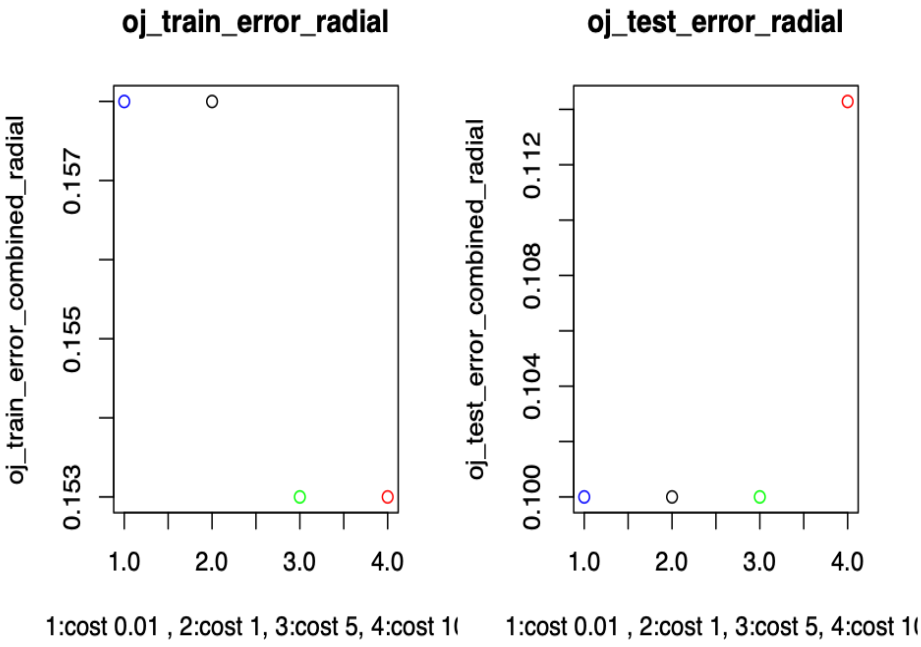
```
##     train.pred_3_radial
##       CH  MM
##   CH 561  53
##   MM 100 286
```

```
## [1] 0.153
```

```
##     test.pred_3_radial
##      CH MM
##   CH 35  4
##   MM  4 27
```

```
## [1] 0.1142857143
```

```
## [1] 0.114
```




```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##         cost
##  0.316227766
##
## - best performance: 0.179
##
## - Detailed performance results:
##              cost error    dispersion
## 1   0.01000000000 0.386 0.03238655414
## 2   0.01778279410 0.386 0.03238655414
## 3   0.03162277660 0.273 0.03917198545
## 4   0.05623413252 0.198 0.04589843861
## 5   0.10000000000 0.189 0.04254409477
## 6   0.17782794100 0.186 0.03777124126
## 7   0.31622776602 0.179 0.04254409477
## 8   0.56234132519 0.179 0.04383048153
## 9   1.00000000000 0.188 0.03392802840
## 10  1.77827941004 0.180 0.03887301263
## 11  3.16227766017 0.183 0.03772709018
## 12  5.62341325190 0.182 0.04049691346
## 13 10.00000000000 0.188 0.03823901440

##
## Call:
## svm(formula = Purchase ~ ., data = oj.train, kernel = "polynomial",
```

```
##     degree = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  2
##       gamma:  0.05555555556
##      coef.0:  0
##
## Number of Support Vectors:  546
##
##  ( 268 278 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
##     train.pred_poly
##       CH  MM
##   CH 569  45
##   MM 140 246
```

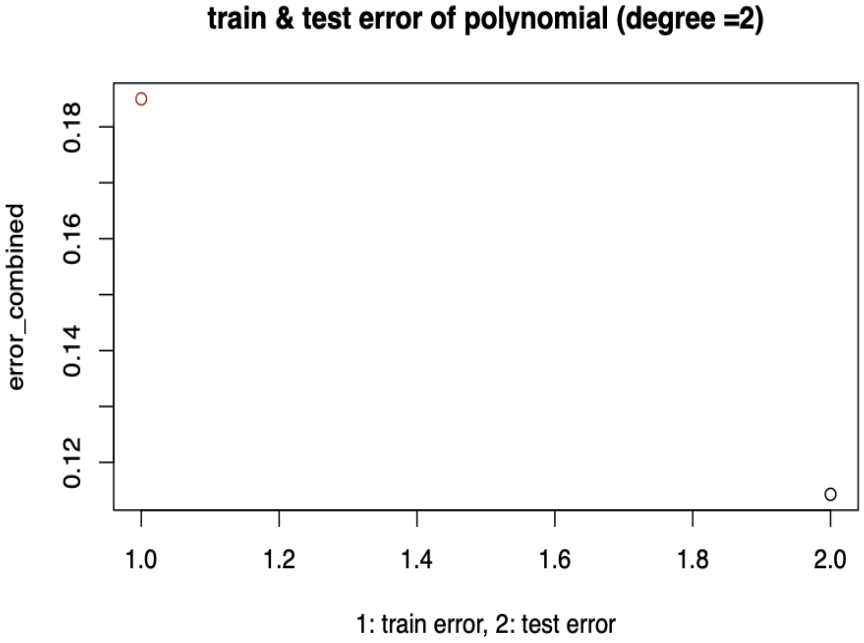
```
## [1] 0.185
```

```
##     test.pred_poly
##      CH MM
##   CH 37  2
##   MM  6 25
```

```
## [1] 0.114
```

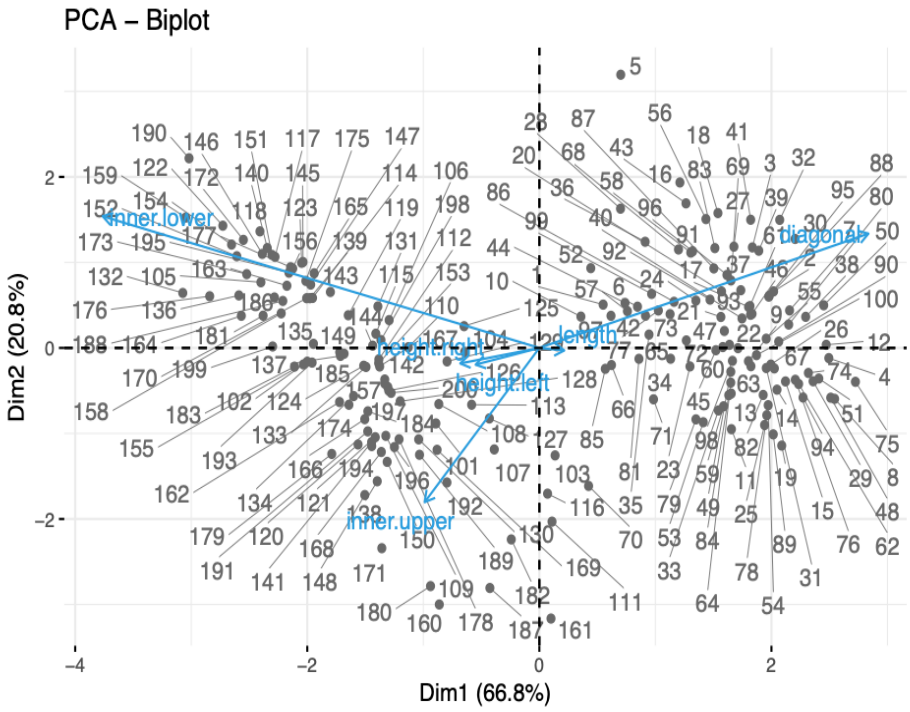
```
##
## Call:
## svm(formula = Purchase ~ ., data = oj.train, kernel = "polynomial",
##     degree = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  2
##       gamma:  0.05555555556
##      coef.0:  0
##
## Number of Support Vectors:  546
##
##  ( 268 278 )
##
##
## Number of Classes:  2
##
```

```
## Levels:
##  CH MM
```

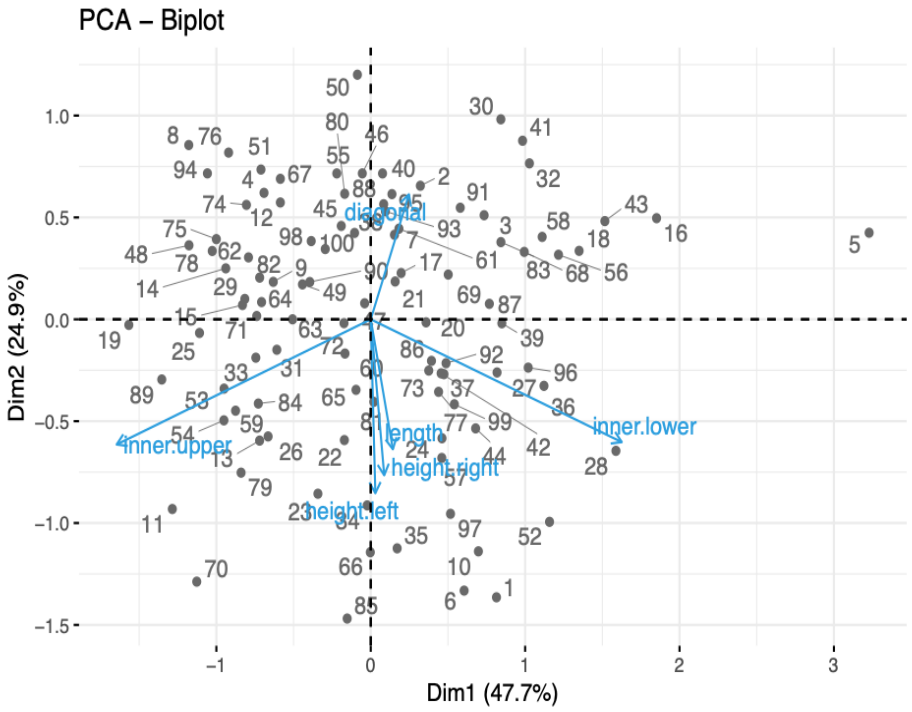


5) Based on biplot in terms of 200 bank notes combined, inner.lower, height.right, height.left and inner upper variables are highly correlated. With this diagonal and length are highly variables correlated. If we divide into two groups; group 1: inner.lower,height.right,height.left and inner upper variables and group 2: diagonal and length, they are negatively correlated against group1. In biplot of genuine, it looks all variables are correlated each other except diagonal and all observations are evenly distributed. In biplot of counterfeit, inner.lower, diagonal and height.left are highly correlated and obsevations are scattered around inner.lower and inner.upper variables. As a result, ther is a little bit difference between PCA of the 100 genuine bank notes and PCA of the 100 counterfeit bank notes.

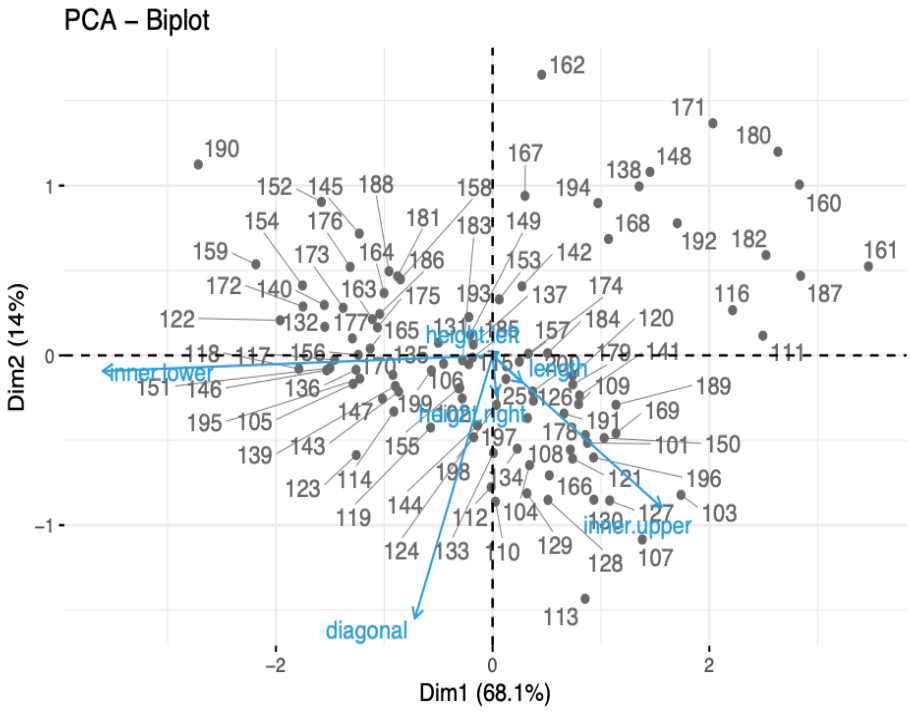
```
## Importance of components:
##                           PC1       PC2       PC3       PC4       PC5
## Standard deviation     1.732139 0.9672748 0.4933697 0.4412015 0.2919107
## Proportion of Variance 0.667520 0.2081600 0.0541600 0.0433100 0.0189600
## Cumulative Proportion  0.667520 0.8756800 0.9298300 0.9731400 0.9921000
##                           PC6
## Standard deviation     0.1884534
## Proportion of Variance 0.0079000
## Cumulative Proportion  1.0000000
```




```
## Importance of components:
##                           PC1       PC2       PC3       PC4       PC5
## Standard deviation     0.8300896 0.5993959 0.4308224 0.295356 0.2831667
## Proportion of Variance 0.4774200 0.2489300 0.1286000 0.060440 0.0555600
## Cumulative Proportion  0.4774200 0.7263500 0.8549500 0.915390 0.9709500
##                           PC6
## Standard deviation     0.2047786
## Proportion of Variance 0.0290500
## Cumulative Proportion  1.0000000
```




```
## Importance of components:
##                           PC1       PC2       PC3       PC4       PC5
## Standard deviation     1.244718 0.5649727 0.4401211 0.3222669 0.2904762
## Proportion of Variance 0.681030 0.1403100 0.0851500 0.0456500 0.0370900
## Cumulative Proportion  0.681030 0.8213300 0.9064800 0.9521300 0.9892200
##                           PC6
## Standard deviation     0.1566098
## Proportion of Variance 0.0107800
## Cumulative Proportion  1.0000000
```


PCA – Biplot
Dim2 (14%)
Dim1 (68.1%)
inner.lower
diagonal
inner.upper
1
0
−1
−2
2