Data Mining Final Project - Melbourne Real Estate

Group 4 - Cullen McNamee, Josep Nueno, and WanQi Tay

```r
library(knitr)
library(RColorBrewer)
library(scales)
library(knitr)
```

```r
housing.full = read.csv("Melbourne_housing_FULL.csv")
```

```r
#Remove rows without pricing information

housing = housing.full[complete.cases(housing.full$Price),]
dim(housing)
```

```
## [1] 27247    21
```

Data Preparation:

```r
sapply(housing, class)
```

```
##        Suburb       Address         Rooms          Type         Price
##      "factor"      "factor"     "integer"      "factor"     "integer"
##        Method       SellerG          Date      Distance      Postcode
##      "factor"      "factor"      "factor"      "factor"      "factor"
##      Bedroom2      Bathroom           Car      Landsize  BuildingArea
##     "integer"     "integer"     "integer"     "integer"     "numeric"
##     YearBuilt   CouncilArea     Lattitude    Longtitude    Regionname
##     "integer"      "factor"     "numeric"     "numeric"      "factor"
## Propertycount
##      "factor"
```

```r
# Taking numeric variables only

housing.pca = housing[,c(3,5,9,11,12,13,14,15,16,21)]
head(housing.pca)
```

```
##    Rooms   Price Distance Bedroom2 Bathroom Car Landsize BuildingArea
## 2      2 1480000      2.5        2        1   1      202           NA
## 3      2 1035000      2.5        2        1   0      156           79
## 5      3 1465000      2.5        3        2   0      134          150
## 6      3  850000      2.5        3        2   1       94           NA
## 7      4 1600000      2.5        3        1   2      120          142
## 11     2  941000      2.5        2        1   0      181           NA
##    YearBuilt Propertycount
## 2         NA          4019
## 3       1900          4019
## 5       1900          4019
## 6         NA          4019
## 7       2014          4019
## 11        NA          4019
```

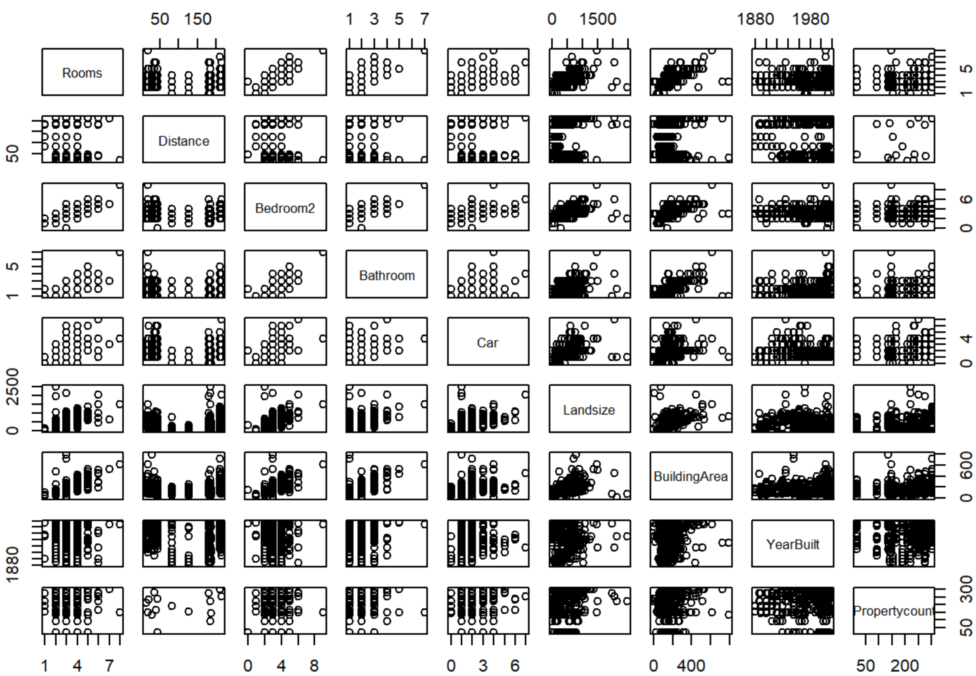
```r
# Convert Distance from Factor variables to Numeric Varibles

housing.pca[,3] = as.numeric(housing.pca[,3])
housing.pca[,10] = as.numeric(housing.pca[,10])
```

Date Cleaning - Row-wise

```
# Remove rows with at least one NA

housing.pca = na.omit(housing.pca)
dim(housing.pca)
```

```
## [1] 8895   10
```

```
plot(housing.pca[1:500,-c(2)])
```



The scatter plots show us there are some highly correlated variables such as Bedroom2 vs Landsize, Bedroom2 vs Building Area, and etc. Hence, we are going to use PCA to convert highly correlated variables into linear uncorrelated variables by grouping them together in the same principal.

Split datasets into training and test sets

```
set.seed(12345)

train.ind.pca = sample(nrow(housing.pca), size = floor(0.80 * nrow(housing.pca)))
```

```
train.pca = housing.pca[train.ind.pca,-c(2)]
test.pca = housing.pca[-train.ind.pca, -c(2)]
```

PCA for first dataset

```
pca = prcomp(train.pca, scale = TRUE)
pca
```

```
## Standard deviations (1, .., p=9):
## [1] 1.8220594 1.0979809 1.0085612 0.9985285 0.9126792 0.8511637 0.6869304
## [8] 0.6283496 0.1900610
##
## Rotation (n x k) = (9 x 9):
##                      PC1         PC2          PC3         PC4
## Rooms          0.504312196 -0.14608360 -0.094273476 -0.03789936
## Distance      -0.082649358 -0.63177654  0.183504753  0.16891552
## Bedroom2       0.502822915 -0.13343163 -0.093554297 -0.04029526
## Bathroom       0.432124187  0.01790749  0.049964062 -0.14598342
## Car            0.309333471  0.17800219  0.131885827  0.21298329
## Landsize       0.103694551  0.16224950  0.198994350  0.90503045
## BuildingArea   0.431746680 -0.10223070 -0.004712657 -0.06182021
## YearBuilt      0.079982827  0.67695985  0.294420857 -0.19295791
## Propertycount  0.005821612 -0.18616625  0.895818749 -0.20396035
##                      PC5         PC6         PC7         PC8
## Rooms          0.10277943  0.06737121  0.41449334  0.14216107
## Distance      -0.69895342 -0.15683981  0.13001753  0.04177083
## Bedroom2       0.10809489  0.06452482  0.44081751  0.12734438
## Bathroom      -0.23090350  0.22563128 -0.23302793 -0.79410677
## Car            0.06200563 -0.88622384 -0.10561166 -0.09734926
## Landsize       0.03988430  0.32001382  0.00301264 -0.01532511
## BuildingArea  -0.10520229  0.13433110 -0.70680352  0.51980211
## YearBuilt     -0.55078209  0.07083127  0.22755424  0.22630856
## Propertycount  0.34064012  0.07018559  0.01239295  0.01008639
##                        PC9
## Rooms         -0.7119606412
## Distance       0.0060884739
## Bedroom2       0.7018578055
## Bathroom      -0.0026388667
## Car           -0.0020443455
## Landsize       0.0003673786
## BuildingArea   0.0205408285
## YearBuilt     -0.0061023167
## Propertycount -0.0003936524
```

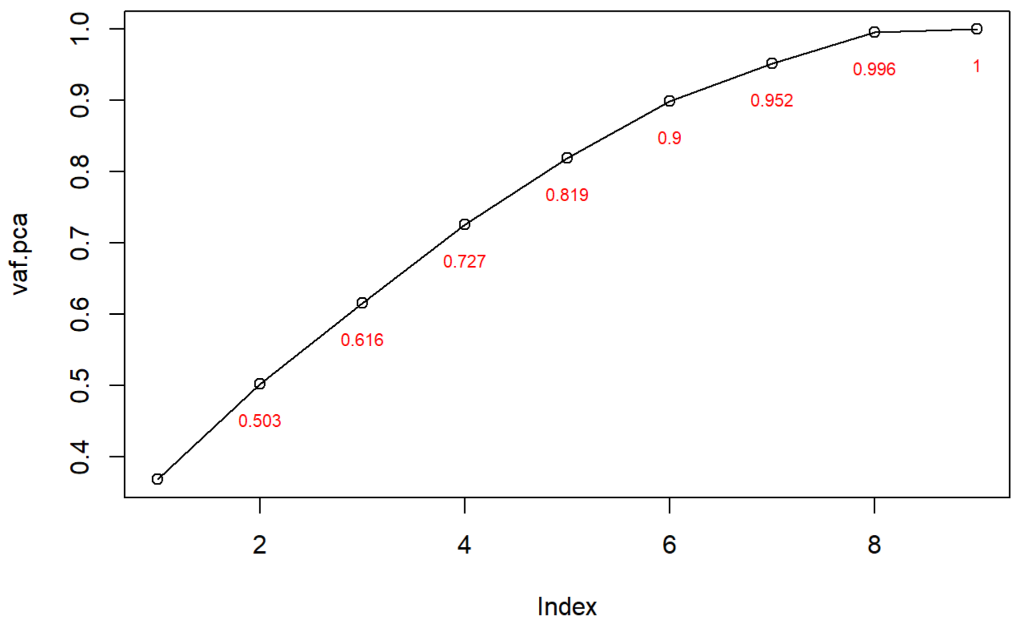
rotation = loadings x = scores

```
head(pca$x)
```

```
##              PC1        PC2        PC3        PC4        PC5         PC6
## 21098 -0.9184882  0.2618054 -0.8804219  0.5498480 -1.6127701  0.23038747
## 29796  1.7585831  1.2486231 -0.9122948 -0.3462028 -0.4146767  0.17295845
## 23188  0.5789407 -0.7997369  0.9896169  0.1027154  0.6485411 -0.33347783
## 30328 -0.8725577  0.9837246 -1.3570570  0.1679262 -0.3169323  0.36839165
## 12380 -0.3026767 -1.1139956 -1.4054470  0.8170596 -0.2821688 -0.74546163
## 4391  -2.3726799  0.2307613  1.4950607 -0.5920937 -1.0162914 -0.06753059
##              PC7       PC8         PC9
## 21098  0.9537657 0.7428042 0.010948284
## 29796  0.2787673 0.4444826 0.007437551
## 23188  1.1419472 0.8606237 0.007591075
## 30328  0.7136261 0.5572149 0.002063791
## 12380 -0.4824245 0.7218891 0.044182914
## 4391   0.7329208 0.1169233 0.002080329
```

Scree Plot

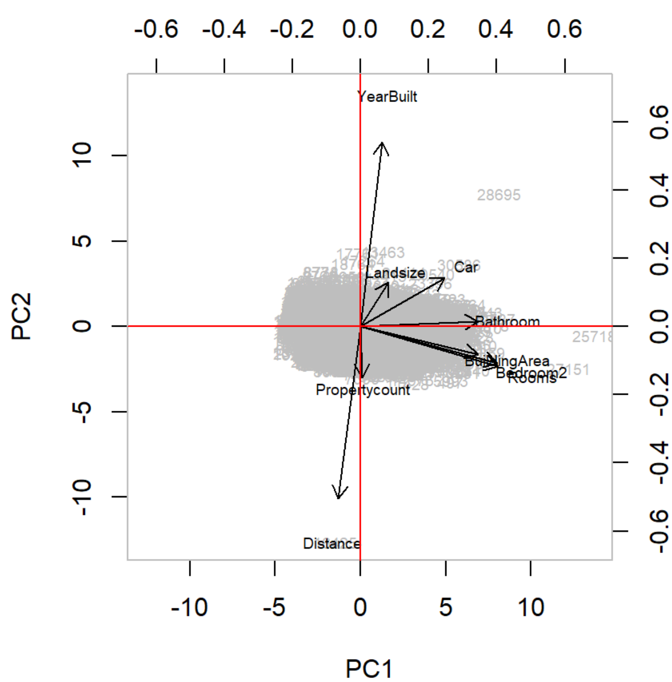
```
vaf.pca = cumsum(pca$sdev^2)/sum(pca$sdev^2)
plot(vaf.pca, type = "o")
text(c(1:9), vaf.pca-0.05, labels = round(vaf.pca,3), cex = 0.7, col = "red")
```



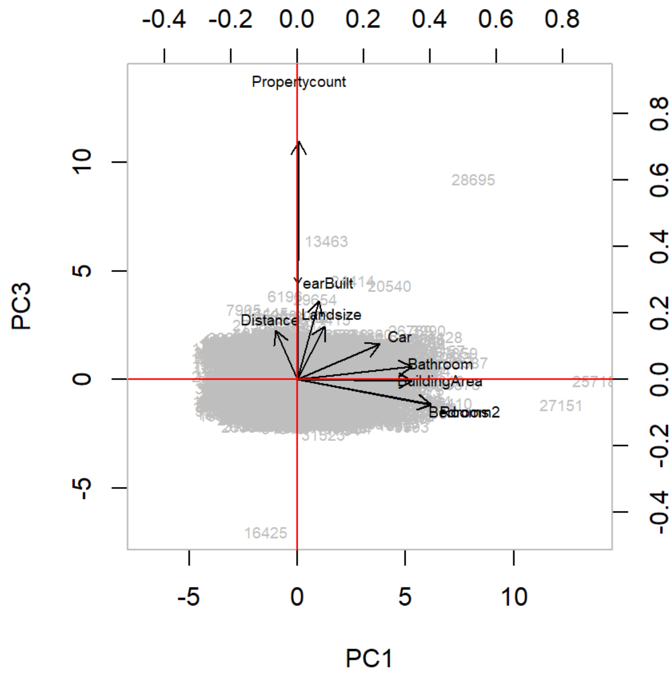
Choose 5 principal components as that accounts for >80% of the data's variance.

Biplot

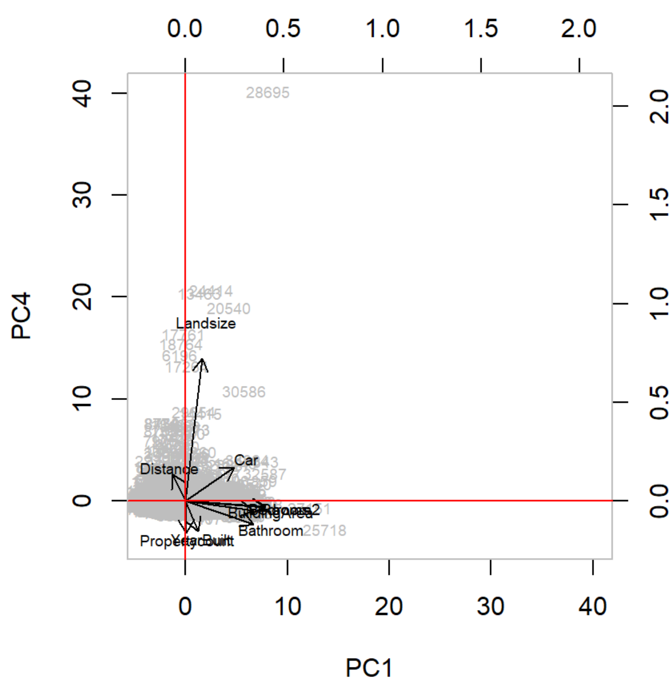
```
biplot(pca, choices = c(1,2), cex = 0.6, scale = 0, col = c(8,1))
abline(h = 0, v = 0, col = "red")
```



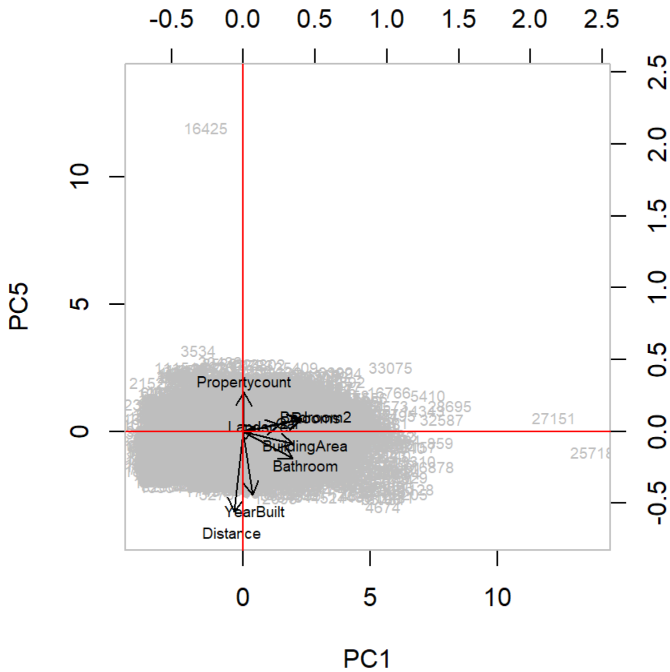

```
biplot(pca, choices = c(1,3), cex = 0.6, scale = 0, col = c(8,1))
abline(h = 0, v = 0, col = "red")
```



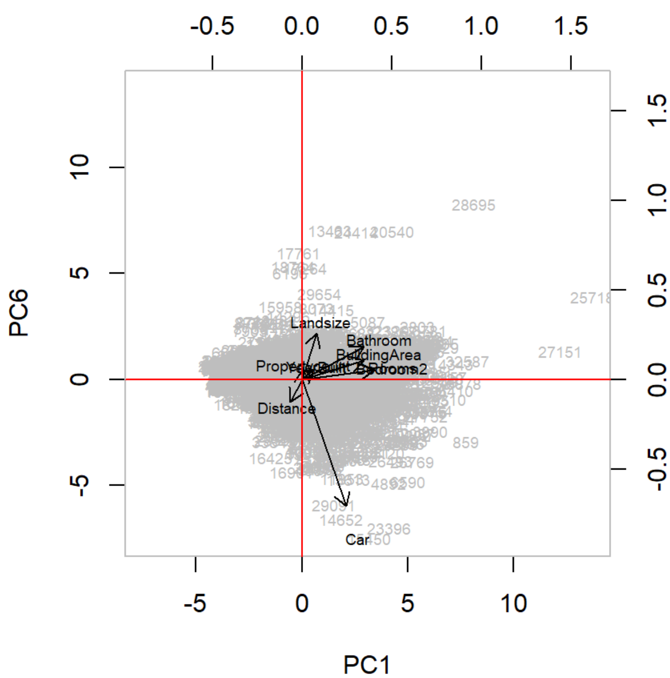

```
biplot(pca, choices = c(1,4), cex = 0.6, scale = 0, col = c(8,1))
abline(h = 0, v = 0, col = "red")
```



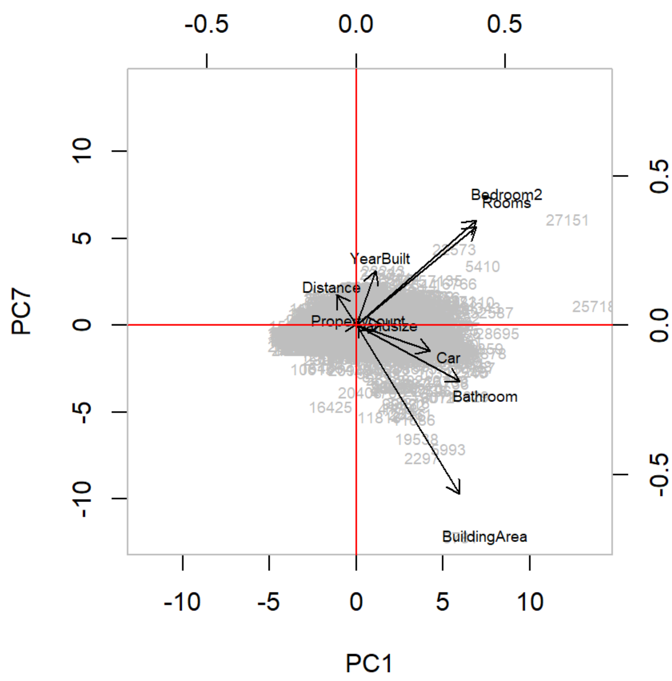

```
biplot(pca, choices = c(1,5), cex = 0.6, scale = 0, col = c(8,1))
abline(h = 0, v = 0, col = "red")
```



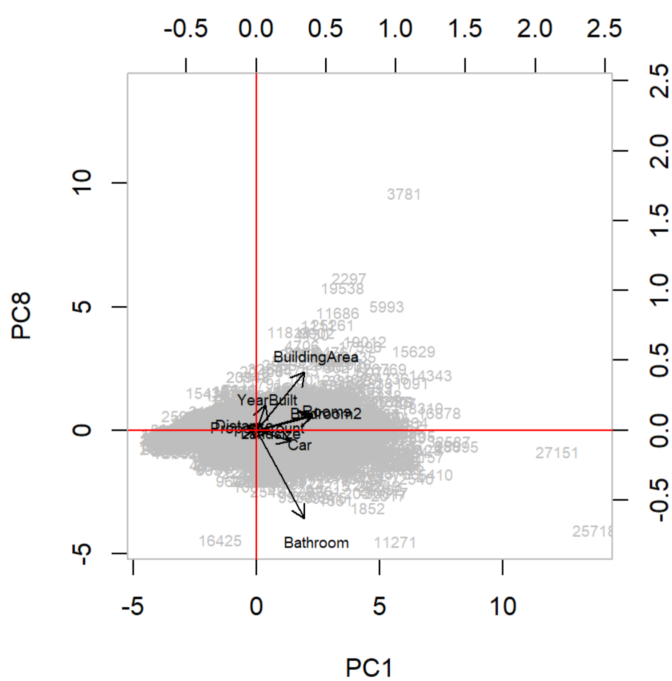

```
biplot(pca, choices = c(1,6), cex = 0.6, scale = 0, col = c(8,1))
abline(h = 0, v = 0, col = "red")
```



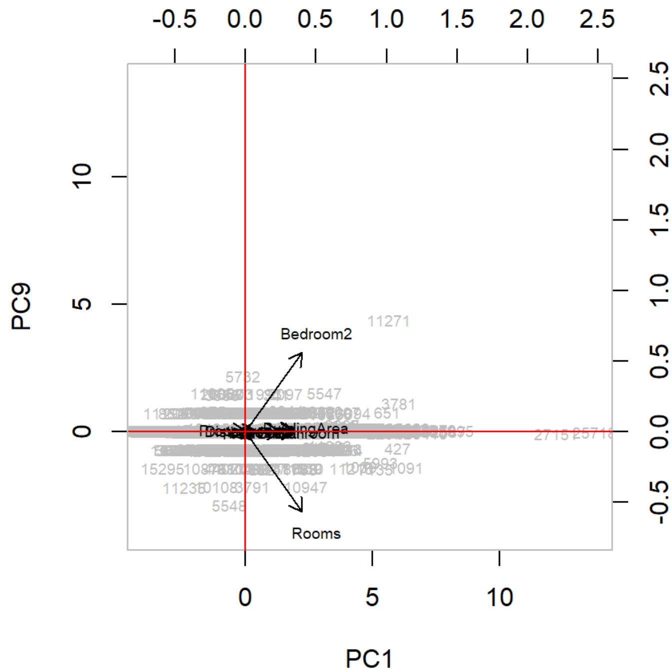

```
biplot(pca, choices = c(1,7), cex = 0.6, scale = 0, col = c(8,1))
abline(h = 0, v = 0, col = "red")
```




```
biplot(pca, choices = c(1,8), cex = 0.6, scale = 0, col = c(8,1))
abline(h = 0, v = 0, col = "red")
```




```
biplot(pca, choices = c(1,9), cex = 0.6, scale = 0, col = c(8,1))
abline(h = 0, v = 0, col = "red")
```



PCA Holdout validation

```
## Measuring consistency

predict.pca = predict(pca, newdata = test.pca) # predicting using all principal components

cor.train.pca = round( cor(as.vector(scale(train.pca)), as.vector(pca$x[,1:5] %*% t(pca$rotation)[1:5,])),2)
cor.test.pca = round( cor(as.vector(scale(test.pca)), as.vector(predict.pca[,1:5] %*% t(pca$rotation)[1:5,])),2)

kable(rbind(cor.train.pca, cor.test.pca), col.names = c("Correlation"))
```

	Correlation
cor.train.pca	0.91
cor.test.pca	0.88

The R2 (correlation) values from train and test are very high and consistent.

```
# Measuring accuracy

predictions.pca.scaled = predict.pca[,1:5] %*% t(pca$rotation)[1:5,]

predictions.pca = predictions.pca.scaled

# Reversed the normalized predictions for accuracy test
for(i in 1:9) {

  predictions.pca[, i] = (predictions.pca.scaled[, i] * sd(train.pca[,i])) + mean(train.pca[,i])
}

head(predictions.pca)
```

```
##       Rooms Distance Bedroom2  Bathroom       Car   Landsize BuildingArea
## 3  2.162805 96.75197 2.138238 0.6374912 0.7081317 -115.78858     55.45211
## 33 3.280071 93.54640 3.248836 1.2359533 1.2363651  -33.42604    132.39057
## 38 1.966178 82.85962 1.956781 1.0488160 1.1330815  110.41234     68.63709
## 57 2.811835 81.26658 2.792546 1.2829470 1.3318279   85.60014    115.14099
## 62 2.170105 97.08760 2.145466 0.6414602 0.7191197  -93.57057     56.01155
## 72 3.033574 40.80140 3.030088 1.8367497 1.9096588  256.43872    150.60828
##    YearBuilt Propertycount
## 3   1903.421      184.9756
## 33  1891.383      189.3510
## 38  1983.353      189.7752
## 57  1942.588      187.8084
## 62  1903.446      184.8846
## 72  2011.612      164.8681
```

```
# Accuracy Test: Test Set vs Denormalized predictions
cor(as.numeric(unlist(test.pca)), as.numeric(unlist(predictions.pca)))^2
```

```
## [1] 0.9837182
```

The test set has an R2 value of 0.98 which indicate that those 5 principal components are good enough to explain most of the data’s variance.

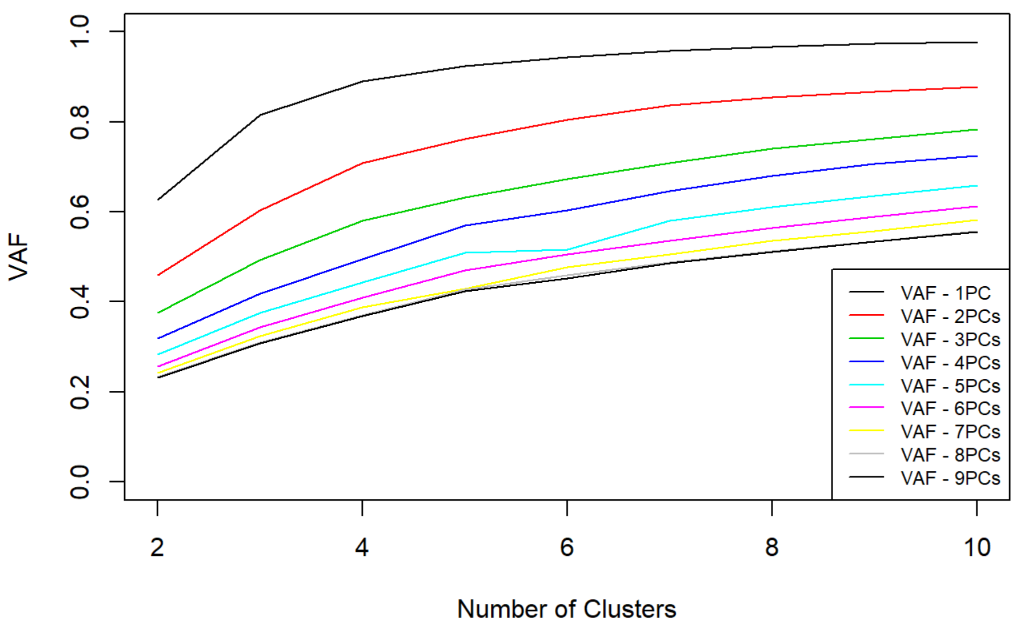
How Dimension Reduction Improve Clustering Result?

```
vaf.kmeans.fw = matrix(nrow = 9, ncol = 10)
colnames(vaf.kmeans.fw) = c("No. of Clusters", "VAF - 1PC", "VAF - 2PCs", "VAF - 3PCs", "VAF - 4PCs","VAF - 5PCs","VAF - 6PC
s", "VAF - 7PCs", "VAF - 8PCs", "VAF - 9PCs")

for (j in 1:9) {
   z = data.frame(pca$x[, 1:j])

    for (i in 2:10) {

    cluster.object = kmeans(z, centers = i, nstart = 20, iter = 100)
    vaf.kmeans.fw[i-1, 1] = i
    vaf.kmeans.fw[i-1, j+1] = cluster.object$betweenss/cluster.object$totss
  }
}

vaf.kmeans.fw
```

```
##       No. of Clusters VAF - 1PC VAF - 2PCs VAF - 3PCs VAF - 4PCs
##  [1,]               2 0.6271336  0.4602905  0.3758540  0.3185815
##  [2,]               3 0.8151465  0.6044504  0.4935936  0.4192101
##  [3,]               4 0.8905141  0.7090518  0.5804368  0.4950284
##  [4,]               5 0.9247848  0.7616132  0.6320832  0.5696073
##  [5,]               6 0.9430392  0.8057756  0.6736458  0.6032647
##  [6,]               7 0.9581658  0.8367610  0.7096325  0.6469617
##  [7,]               8 0.9662570  0.8543550  0.7405440  0.6796790
##  [8,]               9 0.9731856  0.8668341  0.7626671  0.7062211
##  [9,]              10 0.9775269  0.8772854  0.7842085  0.7256193
##       VAF - 5PCs VAF - 6PCs VAF - 7PCs VAF - 8PCs VAF - 9PCs
##  [1,]  0.2826695  0.2573893  0.2432192  0.2326606  0.2317269
##  [2,]  0.3766538  0.3432510  0.3243522  0.3100941  0.3088501
##  [3,]  0.4440896  0.4104999  0.3878983  0.3708961  0.3694084
##  [4,]  0.5093080  0.4701873  0.4294815  0.4269379  0.4247270
##  [5,]  0.5163900  0.5056846  0.4775571  0.4596071  0.4515920
##  [6,]  0.5810147  0.5356458  0.5057628  0.4874249  0.4863044
##  [7,]  0.6108605  0.5650237  0.5358890  0.5138007  0.5111138
##  [8,]  0.6358299  0.5896163  0.5575037  0.5343742  0.5339445
##  [9,]  0.6590641  0.6118870  0.5824431  0.5576098  0.5553760
```

Scree Plot for K-Means Clustering

```
plot(vaf.kmeans.fw[,1], vaf.kmeans.fw[,2], type = "l", xlab = "Number of Clusters", ylab = "VAF", xlim = c(2,10), ylim = c(0
,1), col = 1)
lines(vaf.kmeans.fw[,1], vaf.kmeans.fw[,3], type = "l", col = 2)
lines(vaf.kmeans.fw[,1], vaf.kmeans.fw[,4], type = "l", col = 3)
lines(vaf.kmeans.fw[,1], vaf.kmeans.fw[,5], type = "l", col = 4)
lines(vaf.kmeans.fw[,1], vaf.kmeans.fw[,6], type = "l", col = 5)
lines(vaf.kmeans.fw[,1], vaf.kmeans.fw[,7], type = "l", col = 6)
lines(vaf.kmeans.fw[,1], vaf.kmeans.fw[,8], type = "l", col = 7)
lines(vaf.kmeans.fw[,1], vaf.kmeans.fw[,9], type = "l", col = 8)
lines(vaf.kmeans.fw[,1], vaf.kmeans.fw[,10], type = "l", col = 9)
legend("bottomright", legend=c("VAF - 1PC", "VAF - 2PCs", "VAF - 3PCs", "VAF - 4PCs","VAF - 5PCs","VAF - 6PCs", "VAF - 7PCs"
, "VAF - 8PCs", "VAF - 9PCs"), col = 1:9, lty=1, cex = 0.75)
```



As dimensions reduced, the clustering results improved.

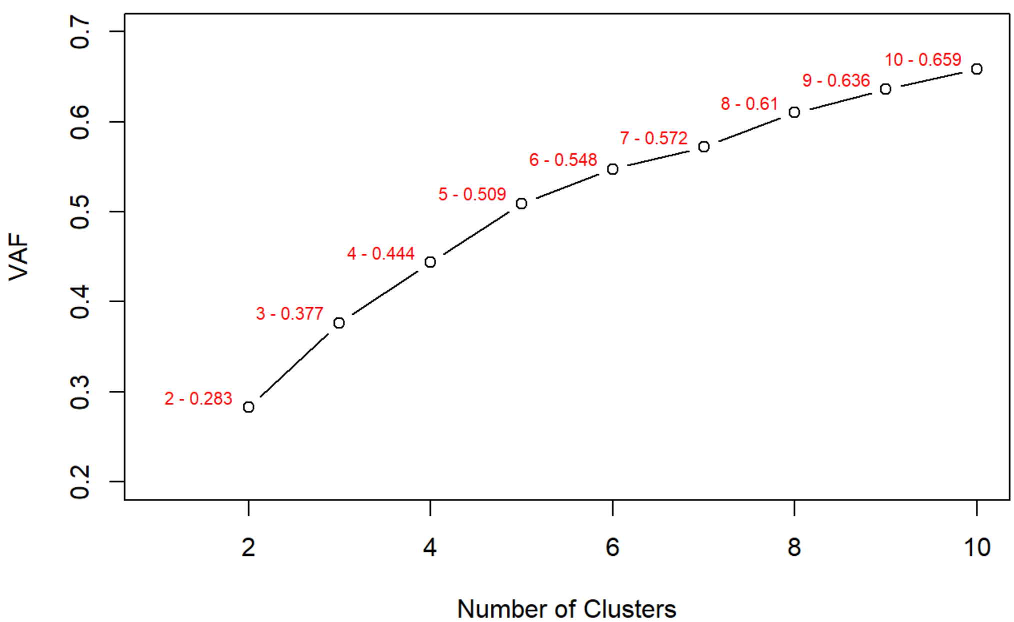
K-Means Clustering Using 5 Principal Components

```
# K-means clustering
# Determine the correct number of clusters via VAF

x = data.frame(pca$x[, 1:5])
vaf.kmeans = matrix(nrow = 9, ncol = 2)
colnames(vaf.kmeans) = c("No. of Clusters", "VAF")


for (i in 2:10) {
  cluster.object = kmeans(x, centers = i, nstart = 20, iter = 100)
  vaf.kmeans[i-1, 1] = i
  vaf.kmeans[i-1, 2] = cluster.object$betweenss/cluster.object$totss
}

vaf.kmeans
```

```
##       No. of Clusters       VAF
##  [1,]                2 0.2826695
##  [2,]                3 0.3766538
##  [3,]                4 0.4436889
##  [4,]                5 0.5091252
##  [5,]                6 0.5476015
##  [6,]                7 0.5720033
##  [7,]                8 0.6101226
##  [8,]                9 0.6360656
##  [9,]               10 0.6587696
```

Scree Plot for K-Means Clustering

```
plot(vaf.kmeans[,1], vaf.kmeans[,2], type = "b", xlab = "Number of Clusters", ylab = "VAF", xlim = c(1,10), ylim = c(0.2, 0.
7))
text(vaf.kmeans[,1], vaf.kmeans[,2]+0.01, labels = paste(vaf.kmeans[,1],"-", round(vaf.kmeans[,2],3)), pos = 2, cex = 0.7, c
ol = "red")
```



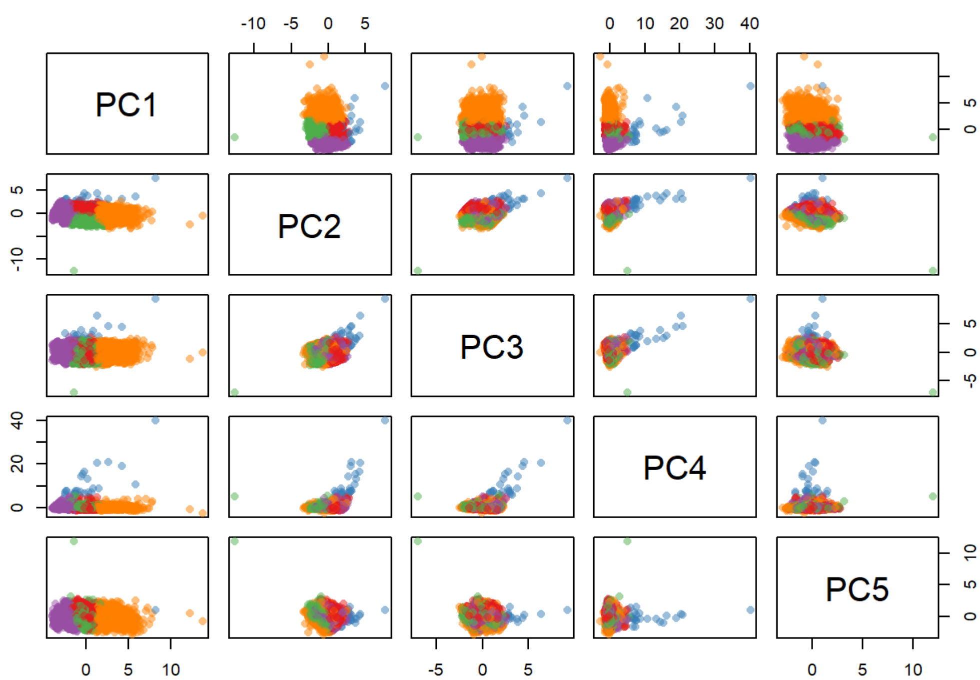

```
# Calculate size of clusters

size.kmeans = matrix(nrow = 9, ncol = 11)
colnames(size.kmeans) = c("No. of Clusters", "Size.1", "Size.2", "Size.3", "Size.4", "Size.5", "Size.6", "Size.
7", "Size.8", "Size.9", "Size.10")

for (i in 2:10){
  size.kmeans[(i-1), 1] = i
  size.kmeans[(i-1), 2:(i+1)] = round((kmeans(x, centers = i, nstart = 20, iter.max = 100)$size)/nrow(train.pca), 4)
}

size.kmeans
```

```
##       No. of Clusters Size.1 Size.2 Size.3 Size.4 Size.5 Size.6 Size.\n7
##  [1,]               2 0.3536 0.6464     NA     NA     NA     NA       NA
##  [2,]               3 0.3065 0.3380 0.3555     NA     NA     NA       NA
##  [3,]               4 0.3524 0.3375 0.0032 0.3068     NA     NA       NA
##  [4,]               5 0.2195 0.2399 0.2315 0.3058 0.0034     NA       NA
##  [5,]               6 0.1956 0.0034 0.1892 0.2105 0.2112 0.1901       NA
##  [6,]               7 0.1889 0.0032 0.1648 0.1020 0.2064 0.1619   0.1727
##  [7,]               8 0.0967 0.1148 0.1113 0.1769 0.0032 0.1563   0.1646
##  [8,]               9 0.1644 0.1502 0.0032 0.1041 0.1568 0.1484   0.0859
##  [9,]              10 0.0791 0.0488 0.1182 0.1017 0.0965 0.0942   0.1515
##       Size.8 Size.9 Size.10
##  [1,]     NA     NA      NA
##  [2,]     NA     NA      NA
##  [3,]     NA     NA      NA
##  [4,]     NA     NA      NA
##  [5,]     NA     NA      NA
##  [6,]     NA     NA      NA
##  [7,] 0.1762     NA      NA
##  [8,] 0.1386 0.0483      NA
##  [9,] 0.1776 0.0032  0.1291
```

We are choosing cluster size of 5 because it is where the ‘elbow’ is located. Besides, too many clusters are not practical and it is also hard to interpret.

```
kmeans =  kmeans(x, 5, nstart = 50, iter.max=100)
```

```
palette(alpha(brewer.pal(9,'Set1'), 0.5))
plot(x, col = kmeans$cluster, pch=16)
```




```
# Cluster sizes
sort(table(kmeans$cluster))
```

```
##
##    2    3    4    5    1
##   24 1562 1647 1707 2176
```

K-Means Clustering Holdout validation

```
y = kmeans(predict.pca[,1:5], centers = kmeans$centers)
(y.vaf = y$betweenss/y$totss)
```

```
## [1] 0.581164
```

```
kmeans$betweenss/kmeans$totss
```

```
## [1] 0.509308
```

VAF of the holdout is 0.58 which is slightly higher than the training set of 0.51. I think the results are pretty consistent and the difference could be due to different sample size used.

```
sort(table(y$cluster))
```

```
##
##   2   3   5   4   1
##   2 379 382 445 571
```

Results

```
# Training Set
# First cluster
clust1 = rownames(train.pca[kmeans$cluster == 1,])

# Second Cluster
clust2 = rownames(train.pca[kmeans$cluster == 2,])

# Third Cluster
clust3 = rownames(train.pca[kmeans$cluster == 3,])

# Fourth Cluster
clust4 = rownames(train.pca[kmeans$cluster == 4,])

# Fifth Cluster
clust5 = rownames(train.pca[kmeans$cluster == 5,])
```

```
# Test Set
# First cluster
clust1.test = rownames(test.pca[y$cluster == 1,])

# Second Cluster
clust2.test = rownames(test.pca[y$cluster == 2,])

# Third Cluster
clust3.test = rownames(test.pca[y$cluster == 3,])

# Fourth Cluster
clust4.test = rownames(test.pca[y$cluster == 4,])

# Fifth Cluster
clust5.test = rownames(test.pca[y$cluster == 5,])
```

Means of Each Cluster [Training Set]

```
train.result = as.data.frame(rbind(colMeans(housing.pca[clust1,]), colMeans(housing.pca[clust2,]), colMeans(housing.pca[clus
t3,]), colMeans(housing.pca[clust4,]), colMeans(housing.pca[clust5,])))
cluster.size.train = c(table(kmeans$cluster)[[1]],table(kmeans$cluster)[[2]],table(kmeans$cluster)[[3]],table(kmeans$cluste
r)[[4]],table(kmeans$cluster)[[5]])
cbind(cluster.size.train, train.result)
```

```
##   cluster.size.train    Rooms     Price  Distance Bedroom2 Bathroom
## 1               2176 3.004596  880240.9  53.79825 2.991268 1.517463
## 2                 24 2.291667  742833.3 121.37500 2.291667 1.541667
## 3               1562 3.246479 1346258.8 180.09859 3.207426 1.569142
## 4               1647 1.880389  681137.5 136.89617 1.867031 1.073467
## 5               1707 4.244288 1509187.8  94.67428 4.217926 2.417692
##        Car   Landsize BuildingArea YearBuilt Propertycount
## 1 1.813879   528.2812     130.3306  1976.476      169.2532
## 2 1.875000 12364.5833     109.2083  1997.000      167.3333
## 3 1.518566   427.1908     145.2467  1936.691      188.6460
## 4 0.996357   285.8689      80.0425  1965.904      180.5665
## 5 2.356180   652.1412     241.3221  1977.435      177.0123
```

Compute Price per Land Size/Building Area [Training Set]

```
# Training Set

price.per.m2 = 1:nrow(train.pca)

for(i in 1:nrow(train.pca)){
  if(housing.pca[train.ind.pca[i],7] == 0) {
    price.per.m2[i] = housing.pca[train.ind.pca[i],2]/housing.pca[train.ind.pca[i],8]
  } else {
      price.per.m2[i] = housing.pca[train.ind.pca[i],2]/housing.pca[train.ind.pca[i],7]
  }
}
mean(price.per.m2)
```

```
## [1] 4070.962
```

```
# First Training Cluster

price.per.m2.1 = 1:length(clust1)

for(i in 1:length(clust1)){
  if(housing.pca[clust1[i],7] == 0) {
    price.per.m2.1[i] = housing.pca[clust1[i],2]/housing.pca[clust1[i],8]
  } else {
      price.per.m2.1[i] = housing.pca[clust1[i],2]/housing.pca[clust1[i],7]
  }
}
mean(price.per.m2.1)
```

```
## [1] 3014.381
```

```
# Second Training Cluster

price.per.m2.2 = 1:length(clust2)

for(i in 1:length(clust2)){
  if(housing.pca[clust2[i],7] == 0) {
    price.per.m2.2[i] = housing.pca[clust2[i],2]/housing.pca[clust2[i],8]
  } else {
      price.per.m2.2[i] = housing.pca[clust2[i],2]/housing.pca[clust2[i],7]
  }
}
mean(price.per.m2.2)
```

```
## [1] 72.74635
```

```r
# Third Training Cluster

price.per.m2.3 = 1:length(clust3)

for(i in 1:length(clust3)){
  if(housing.pca[clust3[i],7] == 0) {
    price.per.m2.3[i] = housing.pca[clust3[i],2]/housing.pca[clust3[i],8]
  } else {
      price.per.m2.3[i] = housing.pca[clust3[i],2]/housing.pca[clust3[i],7]
  }
}
mean(price.per.m2.3)
```

```
## [1] 4813.767
```

```r
# Fourth Training Cluster

price.per.m2.4 = 1:length(clust4)

for(i in 1:length(clust4)){
  if(housing.pca[clust4[i],7] == 0) {
    price.per.m2.4[i] = housing.pca[clust4[i],2]/housing.pca[clust4[i],8]
  } else {
      price.per.m2.4[i] = housing.pca[clust4[i],2]/housing.pca[clust4[i],7]
  }
}
mean(price.per.m2.4)
```

```
## [1] 6217.934
```

```r
# Fifth Training Cluster

price.per.m2.5 = 1:length(clust5)

for(i in 1:length(clust5)){
  if(housing.pca[clust5[i],7] == 0) {
    price.per.m2.5[i] = housing.pca[clust5[i],2]/housing.pca[clust5[i],8]
  } else {
      price.per.m2.5[i] = housing.pca[clust5[i],2]/housing.pca[clust5[i],7]
  }
}
mean(price.per.m2.5)
```

```
## [1] 2722.837
```

Means of Each Cluster [Holdout Set]

```r
test.result = as.data.frame(rbind(colMeans(housing.pca[clust1.test,]), colMeans(housing.pca[clust2.test,]), colMeans(housin
g.pca[clust3.test,]), colMeans(housing.pca[clust4.test,]), colMeans(housing.pca[clust5.test,])))
cluster.size.test = c(table(y$cluster)[[1]],table(y$cluster)[[2]],table(y$cluster)[[3]],table(y$cluster)[[4]],table(y$cluste
r)[[5]])
cbind(cluster.size.test, test.result)
```

```
##   cluster.size.test    Rooms     Price  Distance Bedroom2 Bathroom
## 1               571 3.171629  957856.3  51.96322 3.152364 1.556918
## 2                 2 3.000000  725000.0 163.50000 3.000000 2.000000
## 3               379 3.248021 1376801.8 182.68074 3.229551 1.583113
## 4               445 1.898876  684380.5 132.27416 1.883146 1.096629
## 5               382 4.358639 1594644.1 102.49738 4.353403 2.568063
##        Car   Landsize BuildingArea YearBuilt Propertycount
## 1 1.896673   580.0088    137.75142  1973.438      176.6655
## 2 1.000000 39900.0000    135.00000  2002.000      168.0000
## 3 1.474934   422.6332    149.73026  1937.588      185.3562
## 4 1.056180   282.9034     80.50447  1968.807      176.2045
## 5 2.395288   683.1204    260.79681  1981.969      184.2644
```

Compute Price per Land Size/Building Area [Training Set]

```r
# Holdout Set

price.per.m2.ho = 1:nrow(test.pca)

for(i in 1:nrow(test.pca)){
  if(housing.pca[(-train.ind.pca),7][i] == 0) {
    price.per.m2.ho[i] = housing.pca[-train.ind.pca,2][i]/housing.pca[-train.ind.pca,8][i]
  } else {
      price.per.m2.ho[i] = housing.pca[-train.ind.pca,2][i]/housing.pca[-train.ind.pca,7][i]
  }
}
mean(price.per.m2)
```

```
## [1] 4070.962
```

```r
# First Holdout Cluster

price.per.m2.ho.1 = 1:length(clust1.test)

for(i in 1:length(clust1.test)){
  if(housing.pca[clust1.test[i],7] == 0) {
    price.per.m2.ho.1[i] = housing.pca[clust1.test[i],2]/housing.pca[clust1.test[i],8]
  } else {
      price.per.m2.ho.1[i] = housing.pca[clust1.test[i],2]/housing.pca[clust1.test[i],7]
  }
}
mean(price.per.m2.ho.1)
```

```
## [1] 2440.486
```

```r
# Second Holdout Cluster

price.per.m2.ho.2 = 1:length(clust2.test)

for(i in 1:length(clust2.test)){
  if(housing.pca[clust2.test[i],7] == 0) {
    price.per.m2.ho.2[i] = housing.pca[clust2.test[i],2]/housing.pca[clust2.test[i],8]
  } else {
      price.per.m2.ho.2[i] = housing.pca[clust2.test[i],2]/housing.pca[clust2.test[i],7]
  }
}
mean(price.per.m2.ho.2)
```

```
## [1] 18.01055
```

```r
# Third Holdout Cluster

price.per.m2.ho.3 = 1:length(clust3.test)

for(i in 1:length(clust3.test)){
  if(housing.pca[clust3.test[i],7] == 0) {
    price.per.m2.ho.3[i] = housing.pca[clust3.test[i],2]/housing.pca[clust3.test[i],8]
  } else {
      price.per.m2.ho.3[i] = housing.pca[clust3.test[i],2]/housing.pca[clust3.test[i],7]
  }
}
mean(price.per.m2.ho.3)
```

```
## [1] 4486.163
```

```
# Fourth Holdout Cluster

price.per.m2.ho.4 = 1:length(clust4.test)

for(i in 1:length(clust4.test)){
  if(housing.pca[clust4.test[i],7] == 0) {
    price.per.m2.ho.4[i] = housing.pca[clust4.test[i],2]/housing.pca[clust4.test[i],8]
  } else {
      price.per.m2.ho.4[i] = housing.pca[clust4.test[i],2]/housing.pca[clust4.test[i],7]
  }
}
mean(price.per.m2.ho.4)
```

```
## [1] 6196.722
```

```
# Fifth Holdout Cluster

price.per.m2.ho.5 = 1:length(clust5.test)

for(i in 1:length(clust5.test)){
  if(housing.pca[clust5.test[i],7] == 0) {
    price.per.m2.ho.5[i] = housing.pca[clust5.test[i],2]/housing.pca[clust5.test[i],8]
  } else {
      price.per.m2.ho.5[i] = housing.pca[clust5.test[i],2]/housing.pca[clust5.test[i],7]
  }
}
mean(price.per.m2.ho.5)
```

```
## [1] 2661.863
```