12 Unsupervised Learning

By: Udit (based on ISLR)

Setup

USArrests dataset is part of R.

```
dimnames(USArrests)
```

```
## [[1]]
##  [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"
##  [5] "California"     "Colorado"       "Connecticut"    "Delaware"
##  [9] "Florida"        "Georgia"        "Hawaii"         "Idaho"
## [13] "Illinois"       "Indiana"        "Iowa"           "Kansas"
## [17] "Kentucky"       "Louisiana"      "Maine"          "Maryland"
## [21] "Massachusetts"  "Michigan"       "Minnesota"      "Mississippi"
## [25] "Missouri"       "Montana"        "Nebraska"       "Nevada"
## [29] "New Hampshire"  "New Jersey"     "New Mexico"     "New York"
## [33] "North Carolina" "North Dakota"   "Ohio"           "Oklahoma"
## [37] "Oregon"         "Pennsylvania"   "Rhode Island"   "South Carolina"
## [41] "South Dakota"   "Tennessee"      "Texas"          "Utah"
## [45] "Vermont"        "Virginia"       "Washington"     "West Virginia"
## [49] "Wisconsin"      "Wyoming"
##
## [[2]]
## [1] "Murder"   "Assault"  "UrbanPop" "Rape"
```

```
dim(USArrests)
```

```
## [1] 50  4
```

```
summary(USArrests)
```

```
##      Murder          Assault         UrbanPop          Rape
##  Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30
##  1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07
##  Median : 7.250   Median :159.0   Median :66.00   Median :20.10
##  Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23
##  3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18
##  Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00
```

```
apply(USArrests,2,mean); apply(USArrests,2,var)
```

```
##   Murder  Assault UrbanPop     Rape
##    7.788  170.760   65.540   21.232
```

```
##     Murder    Assault   UrbanPop       Rape
##   18.97047 6945.16571  209.51878   87.72916
```

PCA

Variances for individual variables matter in PCA analysis. In this context, it's best to standardize. **Rotations** are same as **loadings**. PCA is invariant to sign-flip.

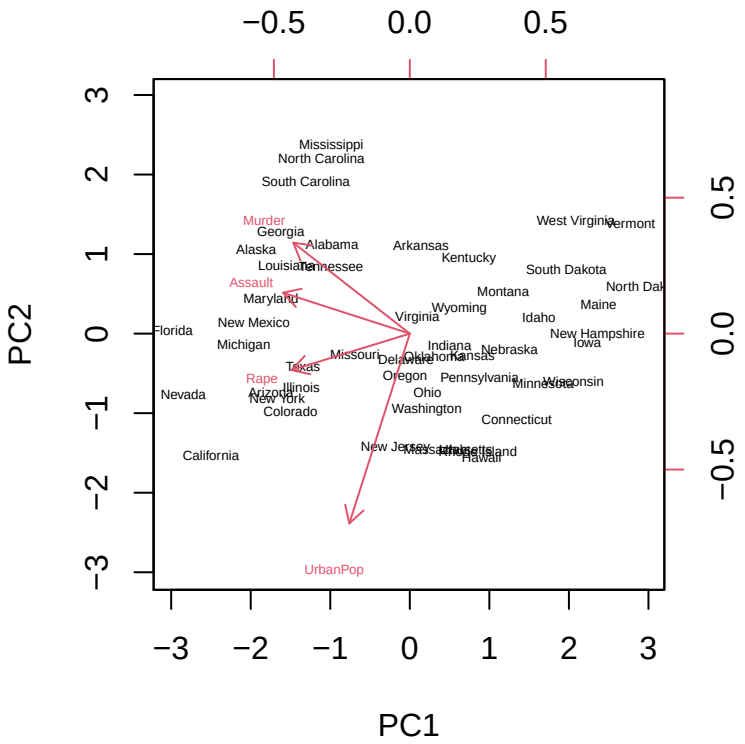
```
pca.out = prcomp(USArrests, scale=TRUE)
pca.out
```

```
## Standard deviations (1, .., p=4):
## [1] 1.5748783 0.9948694 0.5971291 0.4164494
##
## Rotation (n x k) = (4 x 4):
##                 PC1        PC2        PC3         PC4
## Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
## Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
## UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
## Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

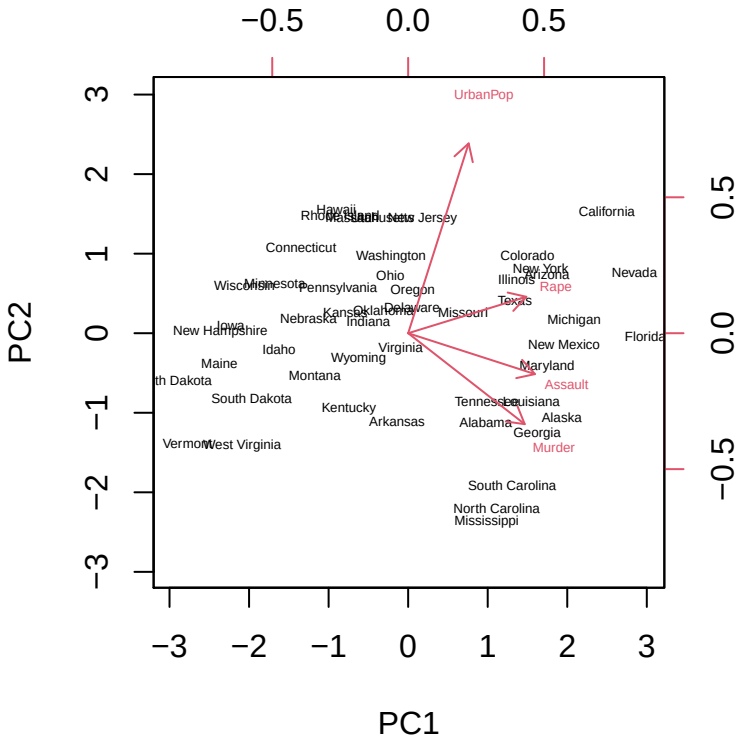
```
names(pca.out)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

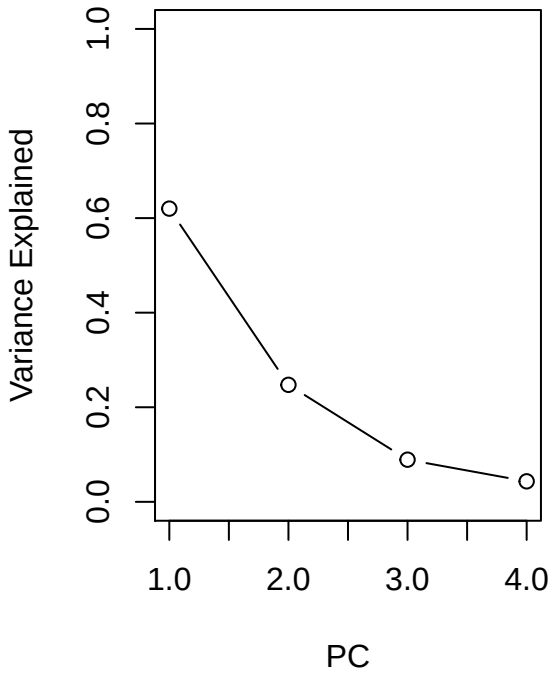
```
biplot(pca.out, scale=0, cex=0.4, pch=19)
```

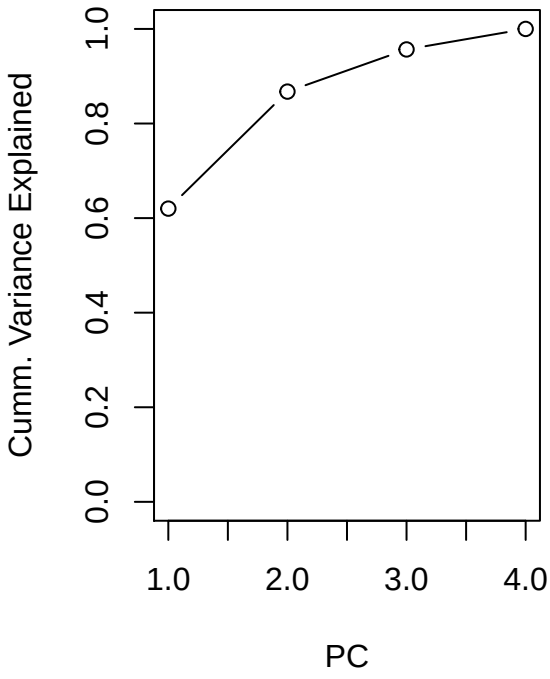



```
# Flipping Sign
pca.out$rotation = -pca.out$rotation
pca.out$x = -pca.out$x
biplot(pca.out, scale=0, cex=0.4, pch=19)
```




```
pve = pca.out$sdev^2/sum(pca.out$sdev^2)
par(mfrow=c(1,2))
plot(pve, xlab="PC", ylab="Variance Explained", type="b", ylim=c(0,1))
plot(cumsum(pve), xlab="PC", ylab="Cumm. Variance Explained", type="b", ylim=c(0,1))
```





PCA - Matrix Completion

```
# X = data.matrix(USArrests)
# X.std = scale(X)
# Z.pca = prcomp(X.std)$x
# Phi.inv = solve(prcomp(X.std)$rotation)
# X.recov = Z.pca %*% Phi.inv
# X.std[1:5,]; X.recov[1:5,]  # gives same result

# We scale the data to begin with - stylized example to avoid de-scaling later
X = data.matrix(scale(USArrests))
pca.X = prcomp(X)   # by default "scale = FALSE"
summary(pca.X)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.5749 0.9949 0.59713 0.41645
## Proportion of Variance 0.6201 0.2474 0.08914 0.04336
## Cumulative Proportion  0.6201 0.8675 0.95664 1.00000
```

```
# pca.X$rotation
# pca.X$x[1:5,]

# Omitting data at random
nomit = 20
set.seed(15)
in.row = sample(1:50,nomit)
in.col = sample(1:4, nomit, replace=TRUE)
```

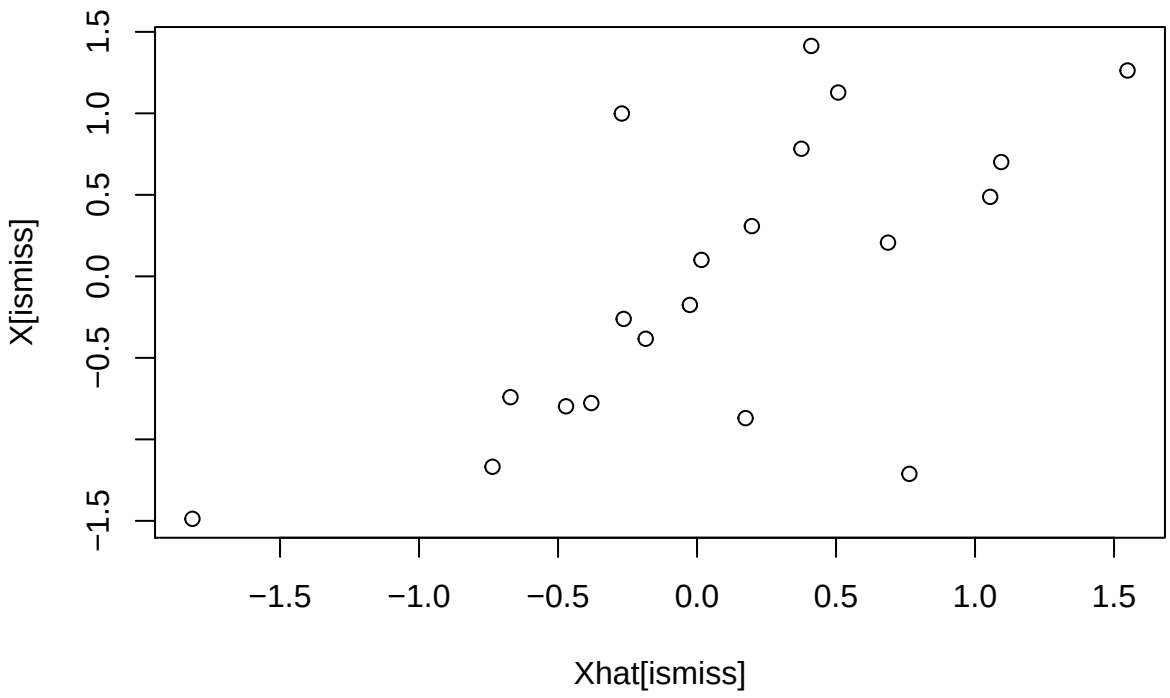
```r
index.na = cbind(in.row, in.col)
X.omit = X
X.omit[index.na] = NA

# Create Xhat where NA is replaced with average values
Xbar = colMeans(X.omit, na.rm=TRUE)
Xhat = X.omit
for(i in 1:ncol(Xhat)){
  Xhat[,i][is.na(Xhat[,i])] = Xbar[i]
}

# Function to reconstruct X from M PCs
Reconstruct.X <- function(X, M=1){
  # fit PCA -> get scores -> inverse of Loadings -> reconstructed X
  res = prcomp(X)
  as.matrix(res$x[,1:M]) %*% solve(res$rotation)[1:M,]
}

# Evaluating missing values using PCA
thresh = 1e-7
rel.error = 1
iter = 0
ismiss = is.na(X.omit)
mssold = mean((scale(X.omit, Xbar, FALSE)[!ismiss])^2)
mss0 = mean(X.omit[!ismiss]^2)

while(rel.error > thresh){
  iter = iter+1
  Xnew = Reconstruct.X(Xhat, M=1)
  #Xnew = fit.svd(Xhat,M=1)
  Xhat[ismiss] = Xnew[ismiss]

  #mean squared error of the non-missing elements
  mss = mean(((X.omit-Xnew)[!ismiss])^2)
  rel.error = (mssold-mss)/mss0
  mssold = mss

  cat("Iter:", iter, "MSS: ", mss, "Rel Error", rel.error, "\n")
}
```

```
## Iter: 1 MSS:  0.3822319 Rel Error 0.6193383
## Iter: 2 MSS:  0.3706956 Rel Error 0.0114846
## Iter: 3 MSS:  0.369492 Rel Error 0.001198144
## Iter: 4 MSS:  0.36934 Rel Error 0.0001513878
## Iter: 5 MSS:  0.3693182 Rel Error 2.167396e-05
## Iter: 6 MSS:  0.3693148 Rel Error 3.35145e-06
## Iter: 7 MSS:  0.3693143 Rel Error 5.475705e-07
## Iter: 8 MSS:  0.3693142 Rel Error 9.367298e-08
```

```r
cor(Xhat[ismiss], X[ismiss])
```

```
## [1] 0.6540486
```

```r
plot(Xhat[ismiss], X[ismiss])
```



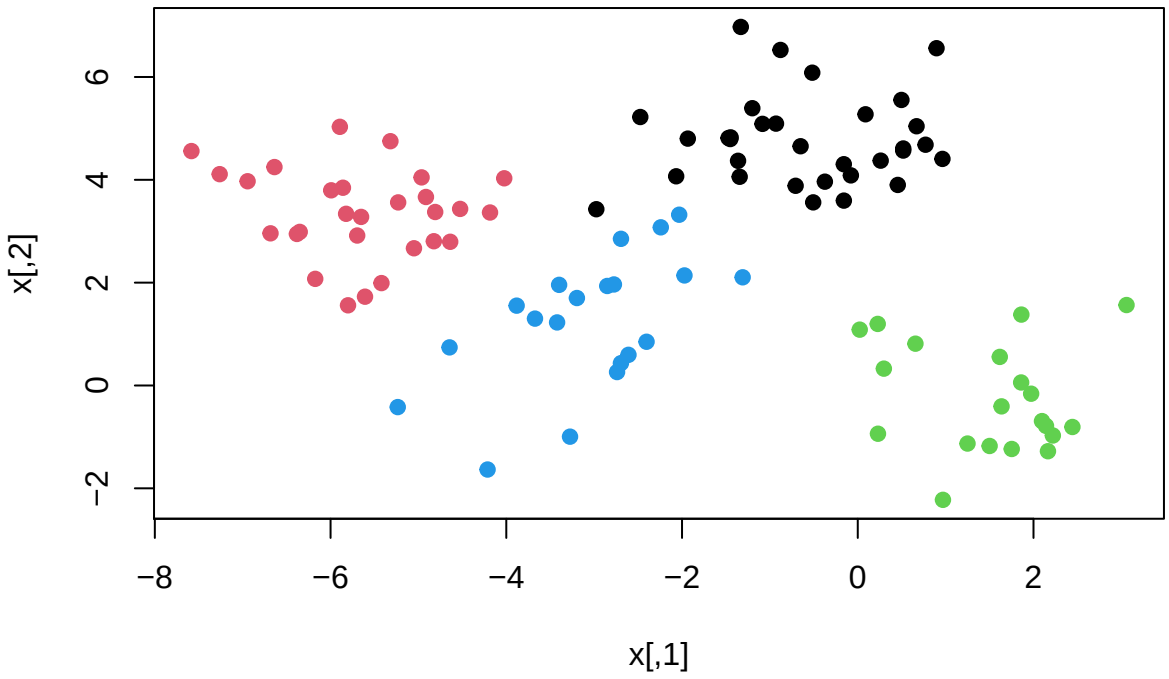

```
# Singular Value Decomposition - v:loadings, u:std. scores, d:std. deviations
svd(X)$v
```

```
##            [,1]       [,2]       [,3]        [,4]
## [1,] -0.5358995  0.4181809 -0.3412327  0.64922780
## [2,] -0.5831836  0.1879856 -0.2681484 -0.74340748
## [3,] -0.2781909 -0.8728062 -0.3780158  0.13387773
## [4,] -0.5434321 -0.1673186  0.8177779  0.08902432
```

```
# Reconstructing X using SVD
fit.svd = function(X, M=1){
  res = svd(X)
  res$u[,1:M] %*% (res$d[1:M]*t(res$v[,1:M])) # transpose = inverse in this case
}
```

K-Means Clustering

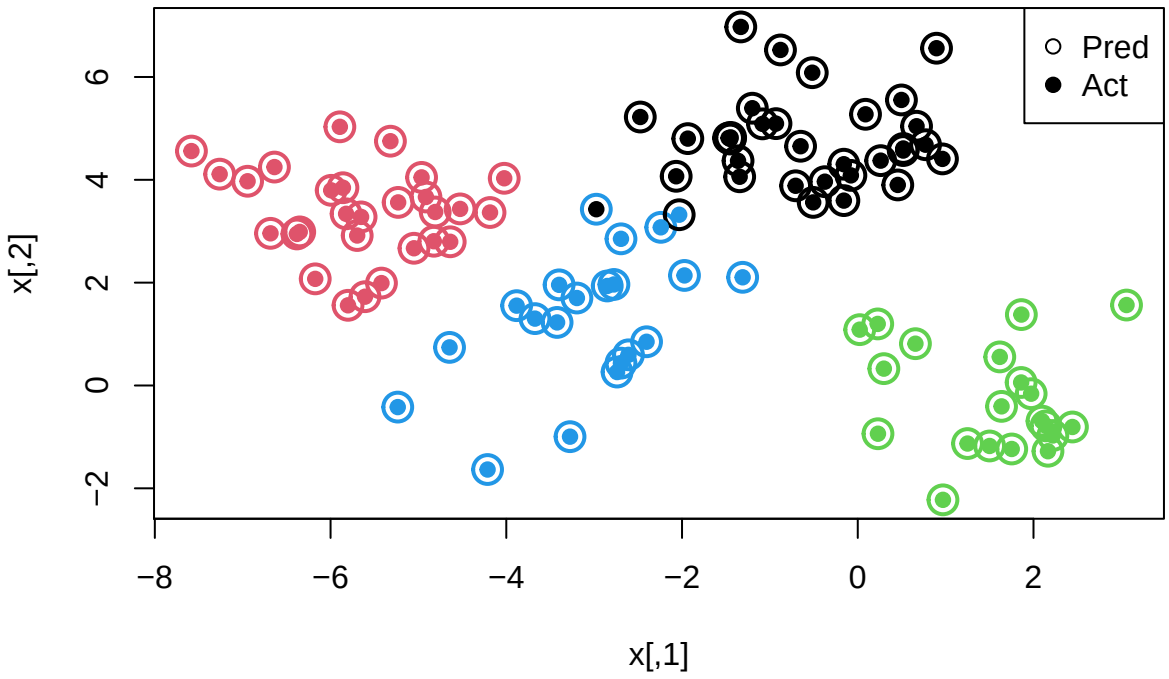
```
# Create Stylized data
set.seed(101)
x=matrix(rnorm(100*2), ncol=2)
x.mean = matrix(rnorm(8,sd=4),4,2)
cluster.assign = sample(1:4, 100, replace=TRUE)
x = x + x.mean[cluster.assign,]
plot(x, col=cluster.assign, pch=19)
```




```
# Running K-Means
km.out = kmeans(x, 4, nstart=15)  #15 random starts
km.out
```

```
## K-means clustering with 4 clusters of sizes 32, 28, 20, 20
##
## Cluster means:
##         [,1]       [,2]
## 1 -0.5787702  4.7639233
## 2 -5.6518323  3.3513316
## 3  1.4989983 -0.2412154
## 4 -3.1104142  1.2535711
##
## Clustering vector:
##   [1] 2 4 1 2 4 1 2 4 1 1 3 1 1 3 4 3 2 3 2 2 2 2 2 3 1 1 4 2 4 1 2 3 2 4 4 3 3
##  [38] 4 3 3 2 4 4 2 2 3 2 1 2 4 2 1 1 3 3 4 3 1 1 1 4 2 2 2 4 4 1 1 3 2 2 1 1 3
##  [75] 1 3 2 1 1 1 4 1 4 1 2 3 1 2 2 1 1 4 2 4 1 1 3 3 1 1
##
## Within cluster sum of squares by cluster:
## [1] 53.04203 42.40322 34.95921 48.52107
##  (between_SS / total_SS =  85.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

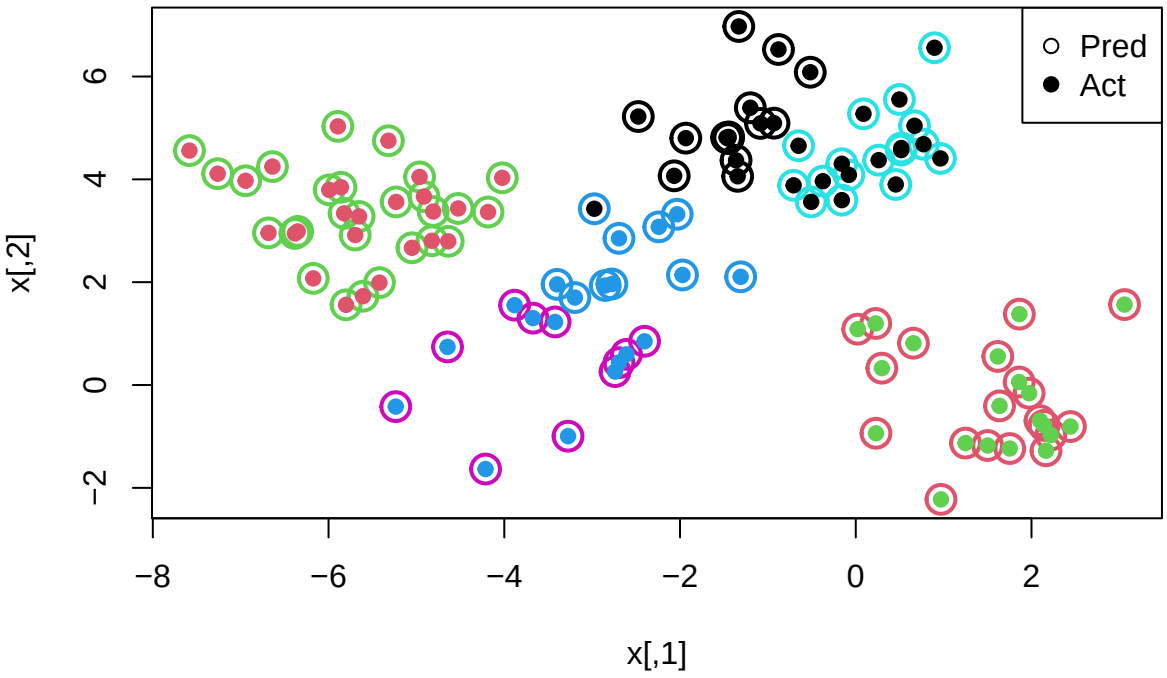
```
plot(x, col=km.out$cluster, cex=2, pch=1, lwd=2) #predicted clusters
points(x, col=cluster.assign, pch=19)            #true cluster
legend("topright", c("Pred", "Act"), pch=c(1,19))
```




```
# Running K-Means
km.out = kmeans(x, 6, nstart=15)  #15 random starts
km.out
```

```
## K-means clustering with 6 clusters of sizes 14, 20, 28, 10, 17, 11
##
## Cluster means:
##         [,1]       [,2]
## 1 -1.3923436  5.1505243
## 2  1.4989983 -0.2412154
## 3 -5.6518323  3.3513316
## 4 -2.5447017  2.4479477
## 5  0.1767241  4.5304027
## 6 -3.5266739  0.3557550
##
## Clustering vector:
##   [1] 3 6 1 3 4 5 3 6 5 1 2 1 5 2 4 2 3 2 3 3 3 3 3 2 5 1 6 3 4 5 3 2 3 6 4 2 2
##  [38] 4 2 2 3 6 4 3 3 2 3 1 3 6 3 1 5 2 2 6 2 4 5 1 6 3 3 3 6 4 1 5 2 3 3 5 5 2
##  [75] 1 2 3 1 1 5 4 5 6 1 3 2 5 3 3 5 5 4 3 6 5 1 2 2 5 1
##
## Within cluster sum of squares by cluster:
## [1] 12.908378 34.959205 42.403216  7.504918 13.951138 18.216096
##  (between_SS / total_SS =  89.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
plot(x, col=km.out$cluster, cex=2, pch=1, lwd=2) #predicted clusters
points(x, col=cluster.assign, pch=19)            #true cluster
legend("topright", c("Pred", "Act"), pch=c(1,19))
```



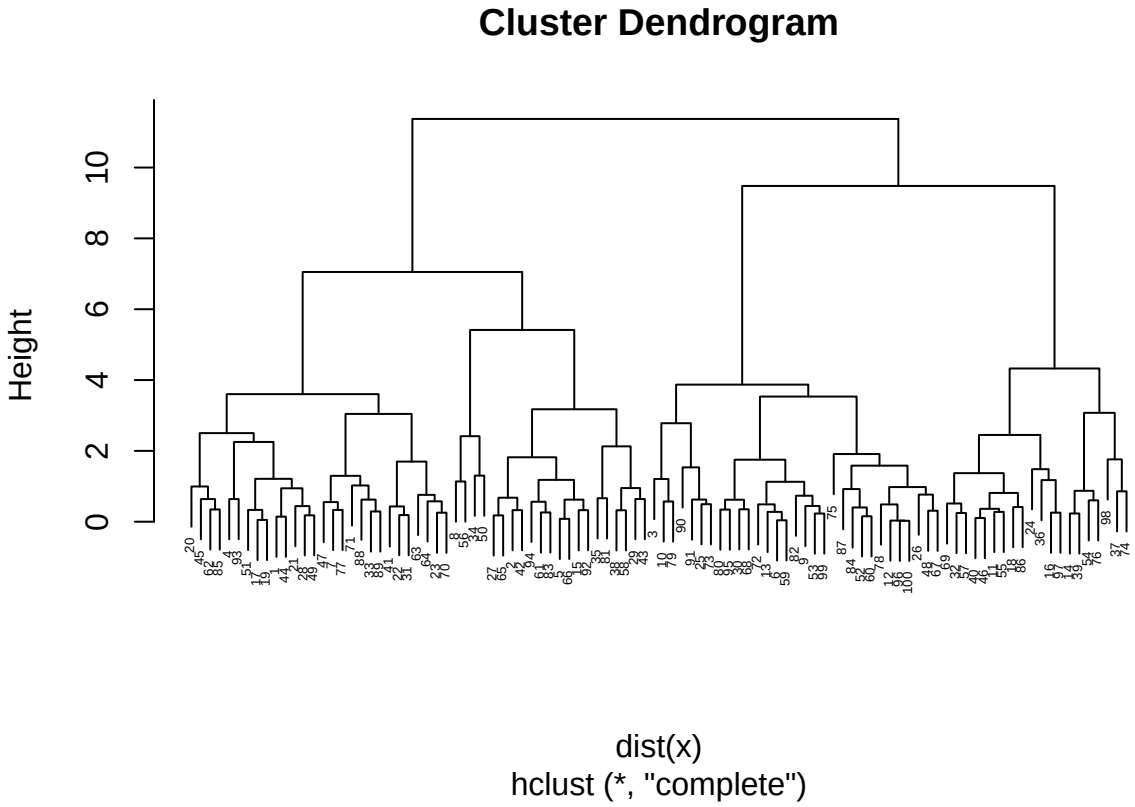
Hierarchical Clustering

Types of linkage approaches:

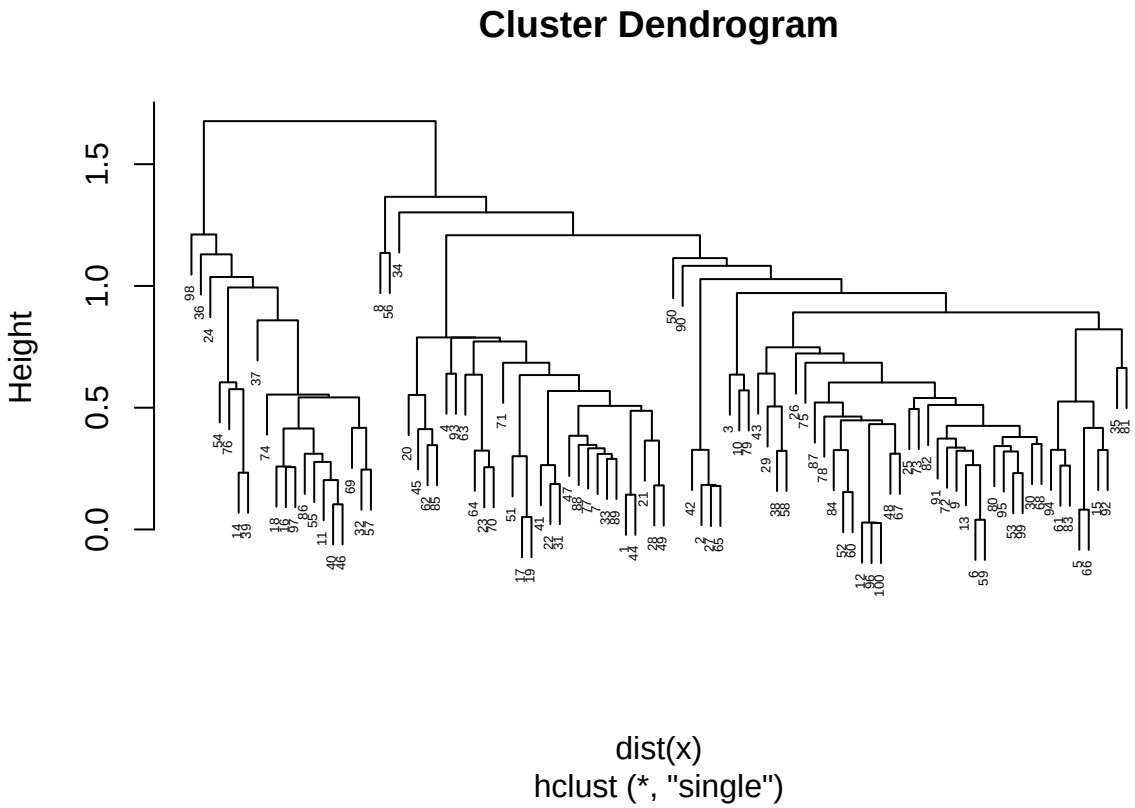
- **Complete**: Largest value among pair-wise distances
- **Single**: Smallest value among pair-wise distances
- **Average**: Average value among pair-wise distances
- **Centroid**: Distance between means

Distance() uses **Euclidean** distance by default.

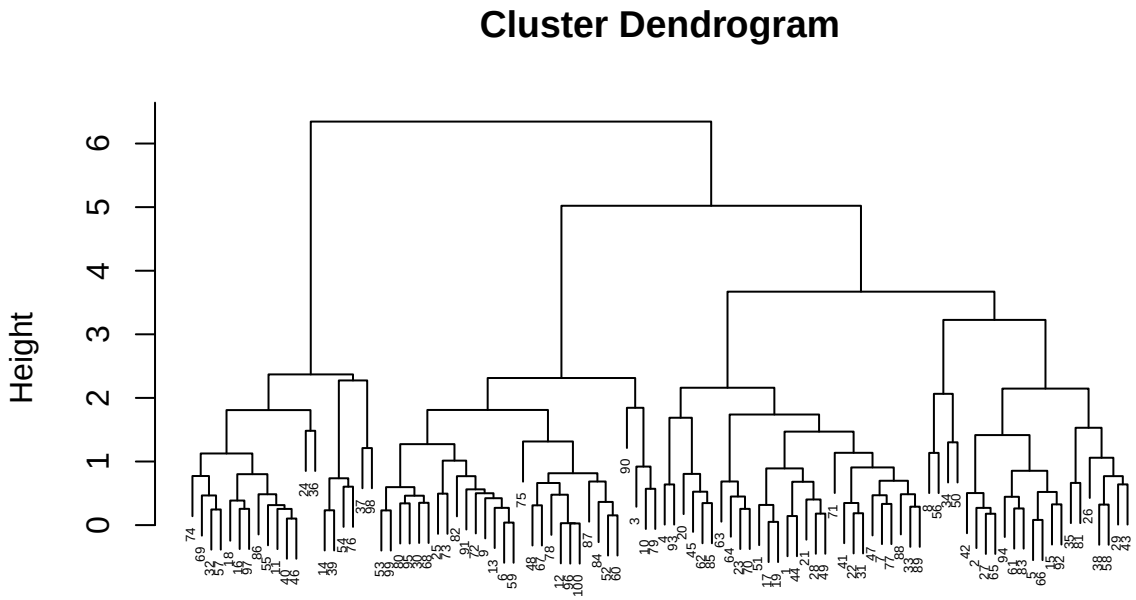
```r
hc.complete = hclust(dist(x), method="complete")
plot(hc.complete, cex=0.4)
```




```
hc.single = hclust(dist(x), method="single")
plot(hc.single, cex=0.4)
```



```r
hc.avg = hclust(dist(x), method="average")
plot(hc.avg, cex=0.4)
```



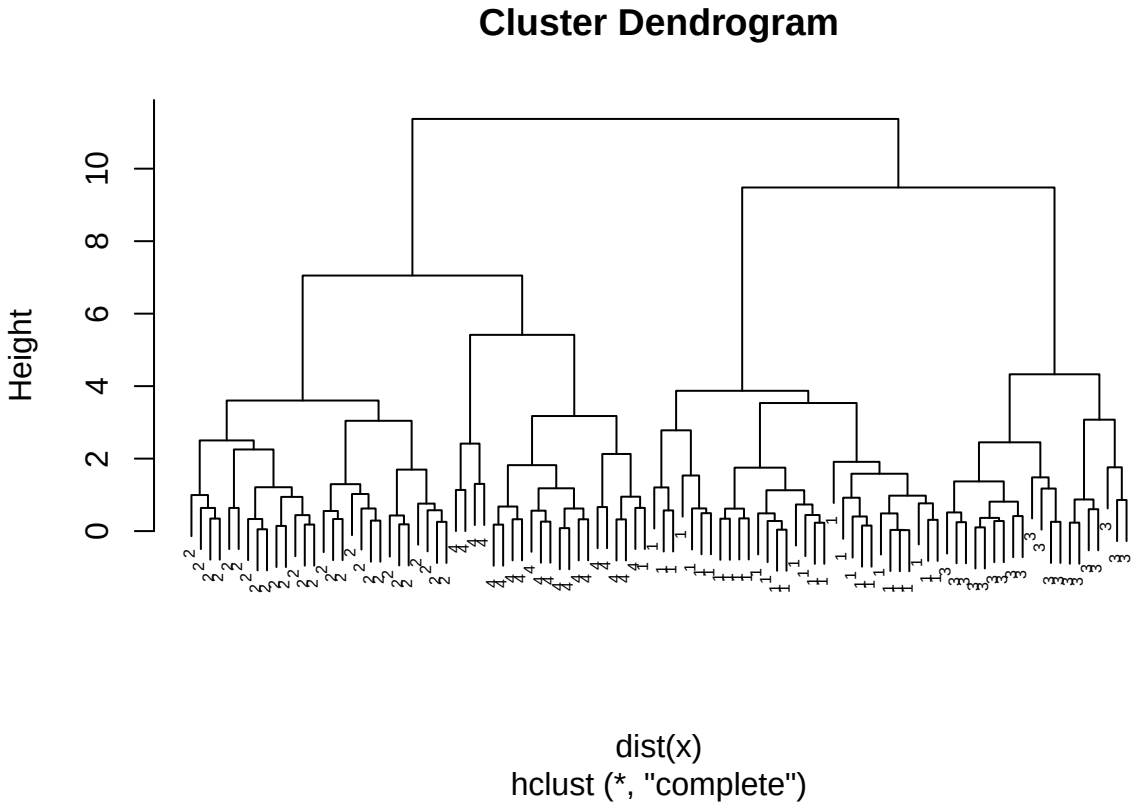

```r
# Cutting Tree
hc.cut = cutree(hc.complete, 4)
table(pred=hc.cut, true=cluster.assign)
```

```
##     true
## pred  1  2  3  4
##    1  0 28  0  0
##    2  1  0  0 20
##    3 31  0  0  0
##    4  0  0 20  0
```

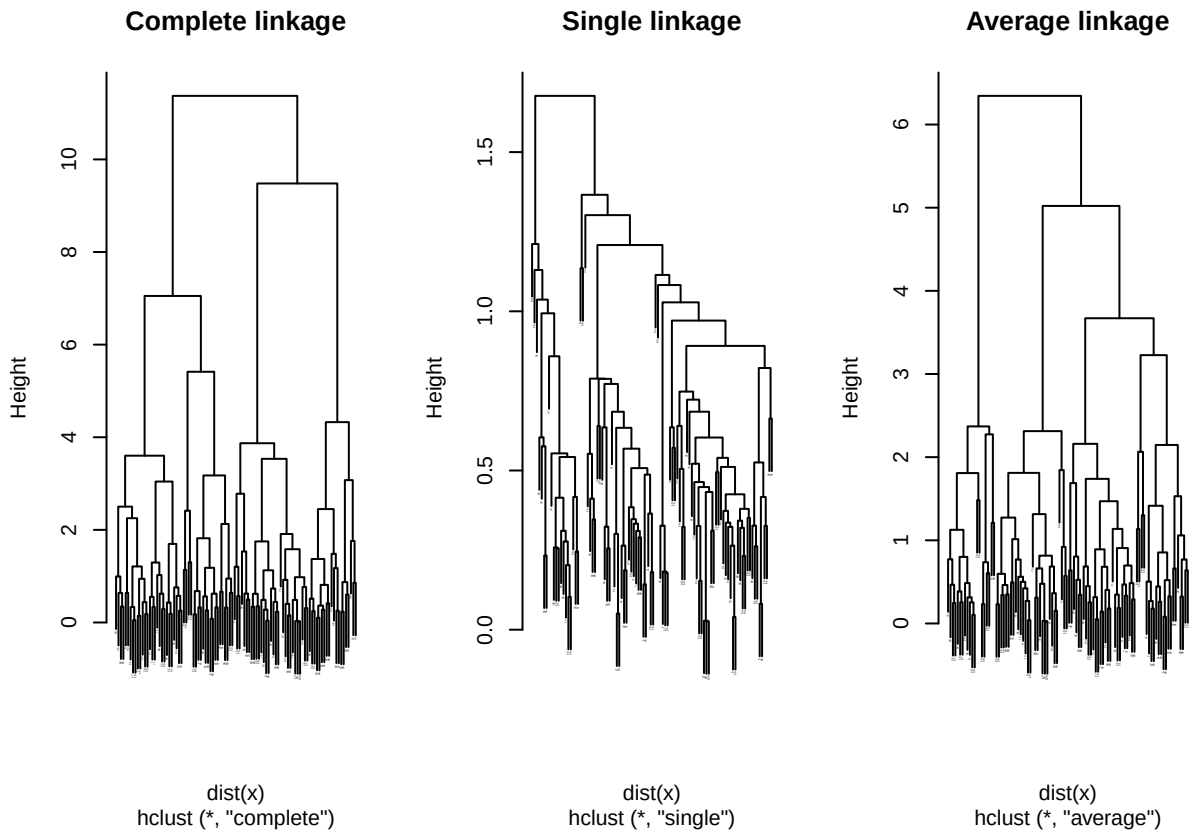
```r
table(pred=km.out$cluster, true=cluster.assign)
```

```
##     true
## pred  1  2  3  4
##    1 14  0  0  0
##    2  0  0 20  0
##    3  0 28  0  0
##    4  1  0  0  9
##    5 17  0  0  0
##    6  0  0  0 11
```

```r
# Plotting True Clusters
plot(hc.complete, labels=cluster.assign, cex=0.5)
```




```
# Comparing all three linkages
par(mfrow=c(1,3))
plot(hc.complete, cex=0.1, main="Complete linkage")
plot(hc.single, cex=0.1, main="Single linkage")
plot(hc.avg, cex=0.1, main="Average linkage")
```




```
par(mfrow=c(1,1))
```

Example - NC160 data

Unsupervised techniques are often used in the analysis of genomic data. **NCI60** cancer cell line microarray data, consists of 6,830 gene expression measurements on 64 cancer cell lines

```
library(ISLR2)
nci.labs = NCI60$labs
nci.data = NCI60$data
dim(nci.data)
```

```
## [1]   64 6830
```

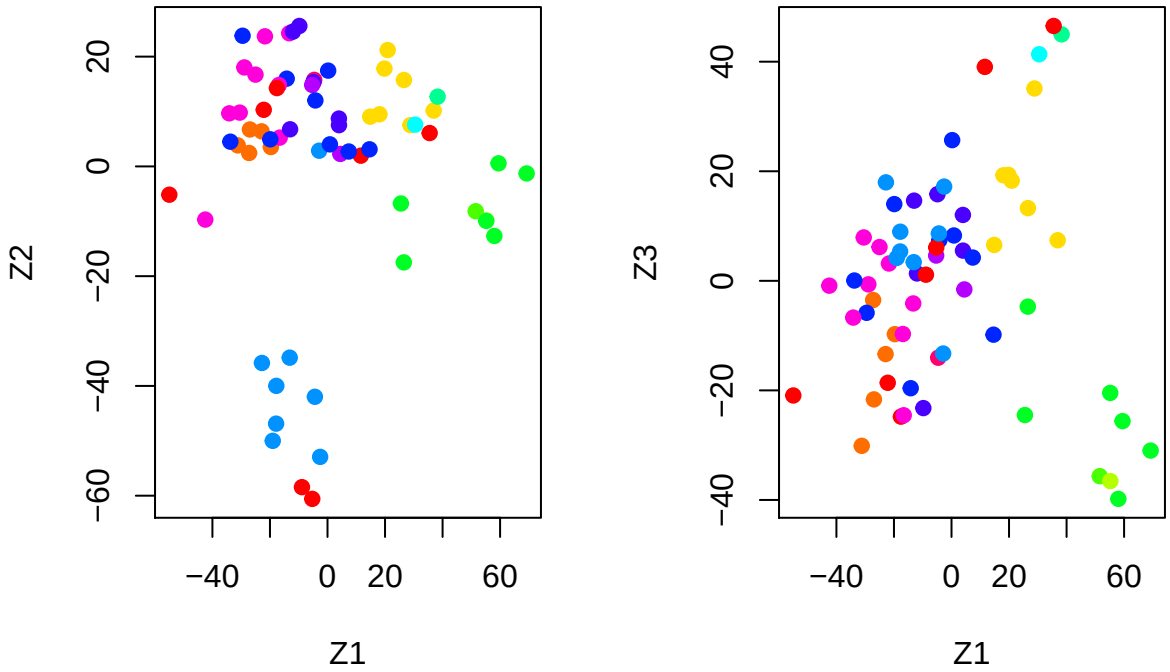
```
table(nci.labs)
```

```
## nci.labs
##      BREAST         CNS       COLON K562A-repro K562B-repro    LEUKEMIA
##           7           5           7           1           1           6
## MCF7A-repro MCF7D-repro    MELANOMA       NSCLC     OVARIAN    PROSTATE
##           1           1           8           9           6           2
##       RENAL     UNKNOWN
##           9           1
```

```
# Plotting First 3 PCs
pr.out = prcomp(nci.data, scale=TRUE)
summary(pr.out)$importance[,1:5]
```

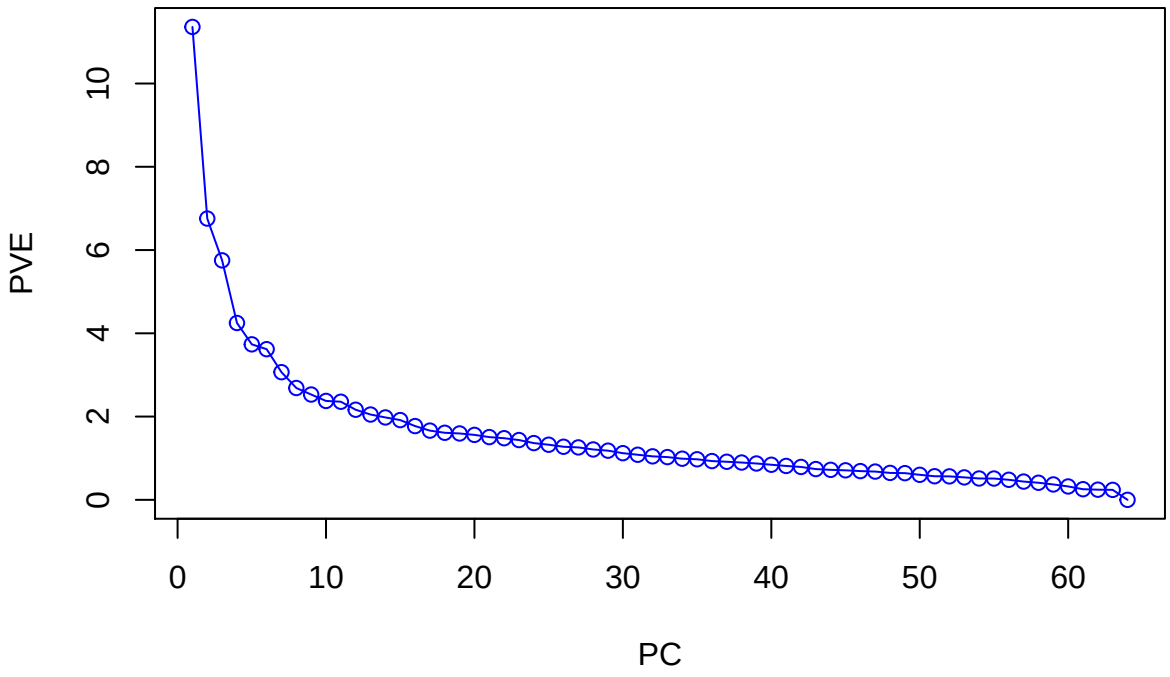
```
##                             PC1      PC2      PC3      PC4      PC5
## Standard deviation     27.85347 21.48136 19.82046 17.03256 15.97181
## Proportion of Variance  0.11359  0.06756  0.05752  0.04248  0.03735
## Cumulative Proportion   0.11359  0.18115  0.23867  0.28115  0.31850
```

```
color <- function(vec){
  cols <- rainbow(length(unique(vec)))
  return(cols[as.numeric(as.factor(vec))])
}

par(mfrow=c(1,2))
plot(pr.out$x[,1:2], col=color(nci.labs), pch=19, xlab="Z1", ylab="Z2")
plot(pr.out$x[,c(1,3)], col=color(nci.labs), pch=19, xlab="Z1", ylab="Z3")
```

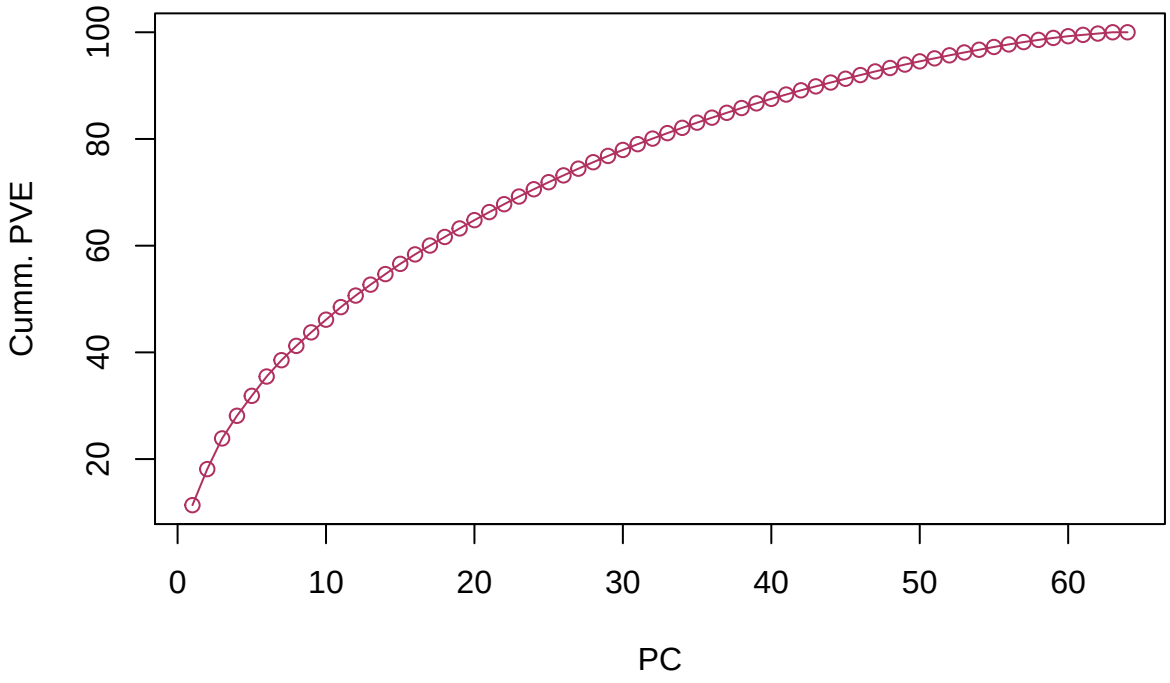


On the whole, cell lines corresponding to a single cancer type do tend to have similar values on the first few principal component score vectors. This indicates that cell lines from the same cancer type tend to have pretty similar gene expression levels.

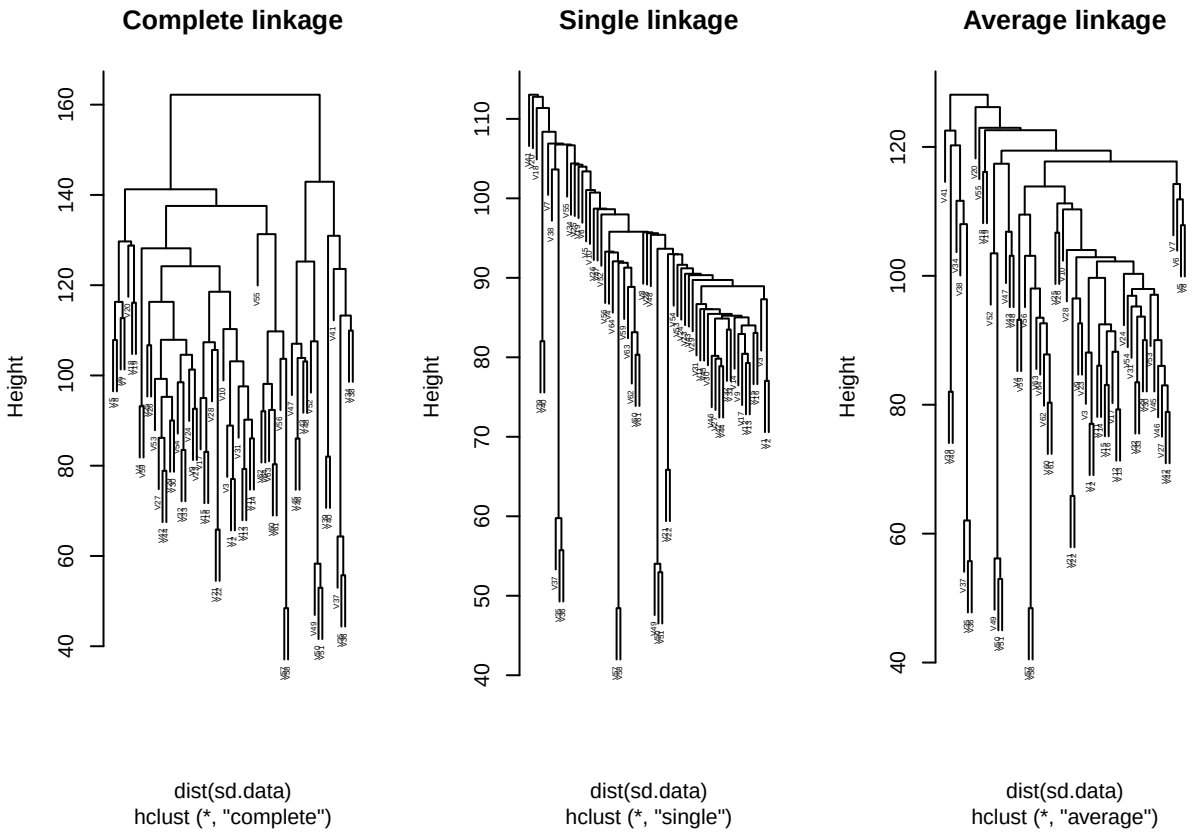
```
# Evaluating PVE
pve = 100*pr.out$sdev^2/sum(pr.out$sdev^2)
plot(pve, type="o", ylab="PVE", xlab="PC", col="blue")
```




```
plot(cumsum(pve), type="o", ylab="Cumm. PVE", xlab="PC", col="maroon")
```



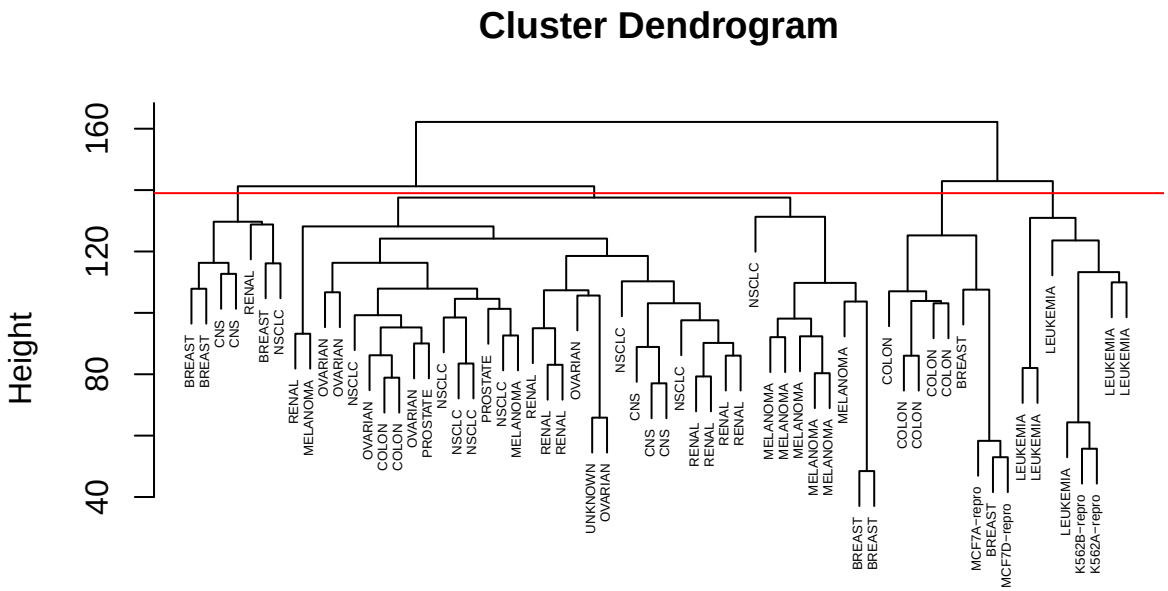
```
sd.data = scale(nci.data)

# Plotting Hierarchical Tree
par(mfrow=c(1,3))
plot(hclust(dist(sd.data), method="complete"), cex=0.4, main="Complete linkage")
plot(hclust(dist(sd.data), method="single"), cex=0.4, main="Single linkage")
plot(hclust(dist(sd.data), method="average"), cex=0.4, main="Average linkage")
```




```
# Cutting Tree
hc.comp = hclust(dist(sd.data), method="complete")
hc.cut = cutree(hc.comp, 4)
table(pred=hc.cut, true=nci.labs)
```

```
##      true
## pred BREAST CNS COLON K562A-repro K562B-repro LEUKEMIA MCF7A-repro MCF7D-repro
##    1      2   3     2           0           0        0           0           0
##    2      3   2     0           0           0        0           0           0
##    3      0   0     0           1           1        6           0           0
##    4      2   0     5           0           0        0           1           1
##      true
## pred MELANOMA NSCLC OVARIAN PROSTATE RENAL UNKNOWN
##    1        8     8       6        2     8       1
##    2        0     1       0        0     1       0
##    3        0     0       0        0     0       0
##    4        0     0       0        0     0       0
```

```
par(mfrow=c(1,1))
plot(hc.comp, labels=nci.labs, cex=0.4)
abline(h=139,col="red")
```

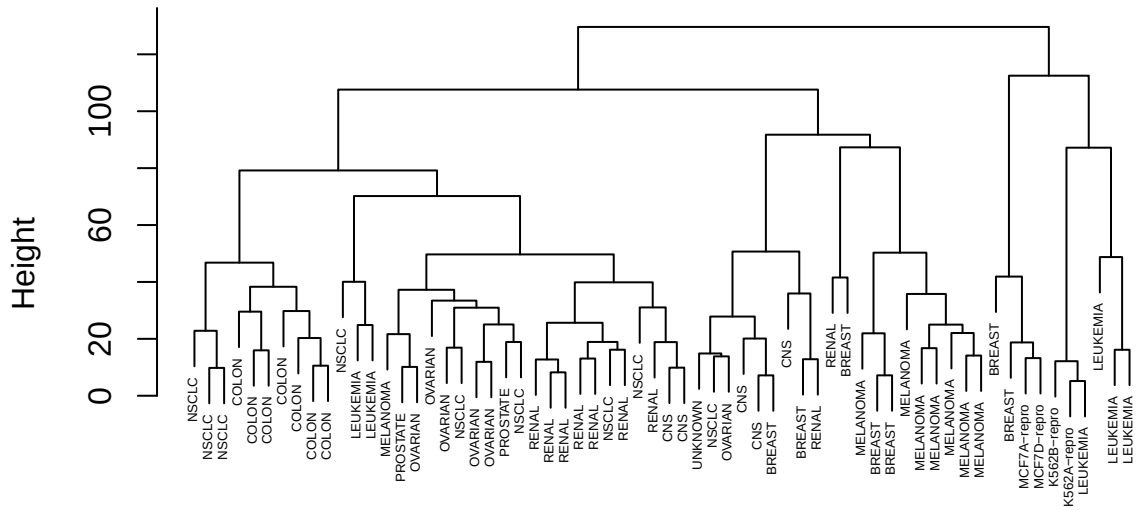


dist(sd.data)
hclust (*, "complete")

```
# Compare with K-mean clustering
set.seed(2)
km.out = kmeans(sd.data, 4, nstart=20)
km.clusters = km.out$cluster
table(k.mean=km.clusters, hc=hc.cut)
```

```
##       hc
## k.mean  1  2  3  4
##      1 11  0  0  9
##      2 20  7  0  0
##      3  9  0  0  0
##      4  0  0  8  0
```

```
# Clustering on first few PCAs instead
hc.out = hclust(dist(pr.out$x[,1:5]), method = "complete")
plot(hc.out, labels=nci.labs, main="HC using first 5 PCAs", cex=0.4)
```



dist(pr.out$x[, 1:5])
hclust (*, "complete")

```
table(PCA = cutree(hc.out,4), nci.labs)
```

```
##    nci.labs
## PCA BREAST CNS COLON K562A-repro K562B-repro LEUKEMIA MCF7A-repro MCF7D-repro
##   1      0   2     7           0           0        2           0           0
##   2      5   3     0           0           0        0           0           0
##   3      0   0     0           1           1        4           0           0
##   4      2   0     0           0           0        0           1           1
##    nci.labs
## PCA MELANOMA NSCLC OVARIAN PROSTATE RENAL UNKNOWN
##   1        1     8       5        2     7       0
##   2        7     1       1        0     2       1
##   3        0     0       0        0     0       0
##   4        0     0       0        0     0       0
```

Quiz

```
load("10.R.Rdata")

x.full = rbind(x, x.test)
pca.x  = prcomp(x.full, scale=TRUE, retx=TRUE)

# Partial Variance Explained
sum(pca.x$sdev[1:5]^2)/sum(pca.x$sdev^2)
```

```
## [1] 0.3498565
```

```r
# Regression using PCAs
dat = data.frame(y = c(y, y.test), z = pca.x$x[,1:5])
train = seq(1,300)
pca.mod = lm(y~., data=dat, subset=train)
summary(pca.mod)
```

```
##
## Call:
## lm(formula = y ~ ., data = dat, subset = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3289 -0.6992  0.0319  0.8075  2.5240
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.09541    0.06107   1.562 0.119314
## z.PC1        0.07608    0.01159   6.564 2.36e-10 ***
## z.PC2       -0.02276    0.01314  -1.732 0.084309 .
## z.PC3       -0.04023    0.01538  -2.616 0.009352 **
## z.PC4       -0.06368    0.02237  -2.847 0.004722 **
## z.PC5       -0.16069    0.04299  -3.738 0.000223 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.056 on 294 degrees of freedom
## Multiple R-squared:  0.1906, Adjusted R-squared:  0.1769
## F-statistic: 13.85 on 5 and 294 DF,  p-value: 3.704e-12
```

```r
# Predict & MSE
y.pred = predict(pca.mod, newdata=dat[-train,])
sqrt(mean((y.test-y.pred)^2))
```

```
## [1] 0.9961315
```

```r
# Compare with Simple Linear Regression
dat.train = data.frame(y, x)
dat.test  = data.frame(y.test, x.test)
slr.mod = lm(y~., data=dat.train)

y.pred.slr = predict(slr.mod, newdata=dat.test)
sqrt(mean((y.test-y.pred.slr)^2))
```

```
## [1] 1.91238
```

```r
# Scores - Manually or through PRComp
temp1 = as.matrix(scale(x.full)) %*% pca.x$rotation
temp2 = pca.x$x
temp1[1,1:5]
```

```
##        PC1        PC2        PC3        PC4        PC5
## -9.1247391  2.1221414  0.6550655 -1.1670852  0.1121413
```

```r
temp2[1,1:5]
```

```
##        PC1        PC2        PC3        PC4        PC5
## -9.1247391  2.1221414  0.6550655 -1.1670852  0.1121413
```