Suhaas HW 3 ISLR

Suhaas Adiraju

2024-10-01

Homework 3

Question 8

library(ISLR) ### a

```
data(USArrests)
states = row.names(USArrests)
names(USArrests)
```

```
## [1] "Murder"   "Assault"  "UrbanPop" "Rape"
```

```
dim(USArrests)
```

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

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

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

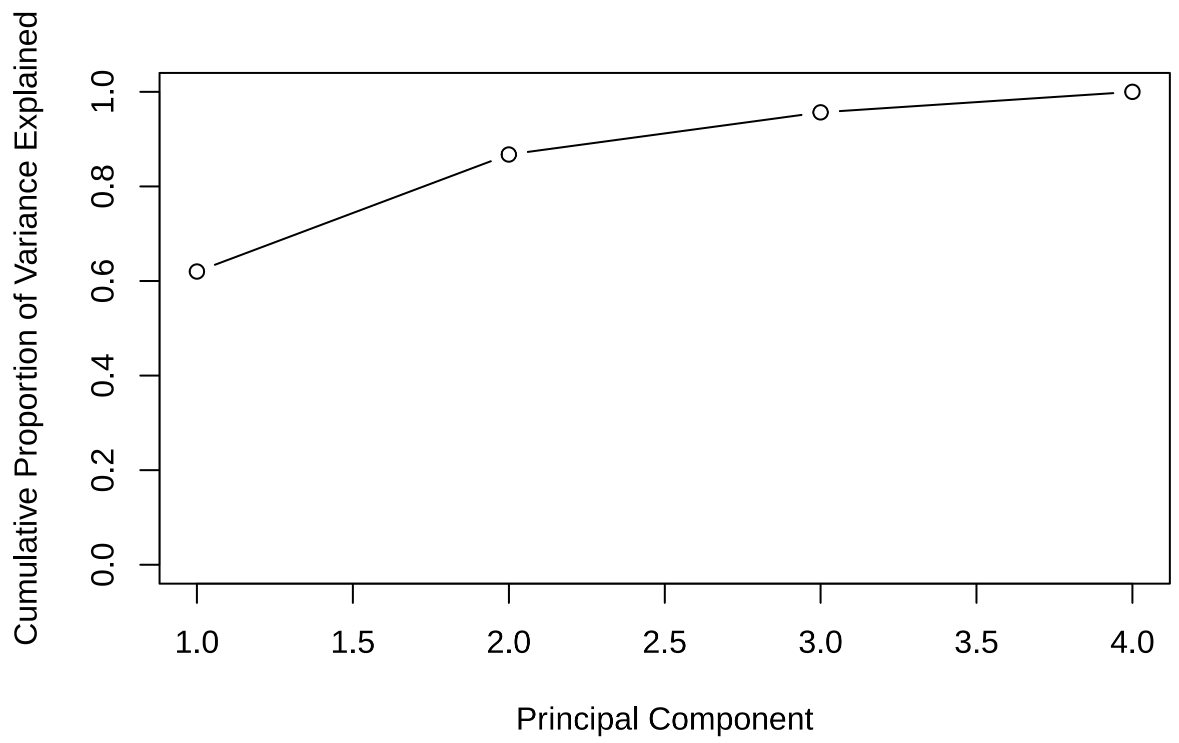
```
apply(USArrests,2,var)
```

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

```
pr.out = prcomp(USArrests, scale=TRUE)
pr.var = pr.out$sdev^2
PVE = pr.var/sum(pr.var)
PVE
```

```
## [1] 0.62006039 0.24744129 0.08914080 0.04335752
```

```
plot(cumsum (PVE ), xlab="Principal Component", ylab = "
Cumulative Proportion of Variance Explained", ylim=c(0,1) ,
type='b')
```



b

```
pr.loadings = pr.out$rotation
PC1var = var(pr.out$x[,1])
PC2var = var(pr.out$x[,2])
PC3var = var(pr.out$x[,3])
PC4var = var(pr.out$x[,4])

totalVar = (PC1var+PC2var+PC3var+PC4var)

PC1.PVEhand = PC1var/totalVar

PC2.PVEhand = PC2var/totalVar

PC3.PVEhand = PC3var/totalVar

PC4.PVEhand = PC4var/totalVar

PC1.PVEhand
```

```
## [1] 0.6200604
```

```
PC2.PVEhand
```

```
## [1] 0.2474413
```

```
PC3.PVEhand
```

```
## [1] 0.0891408
```

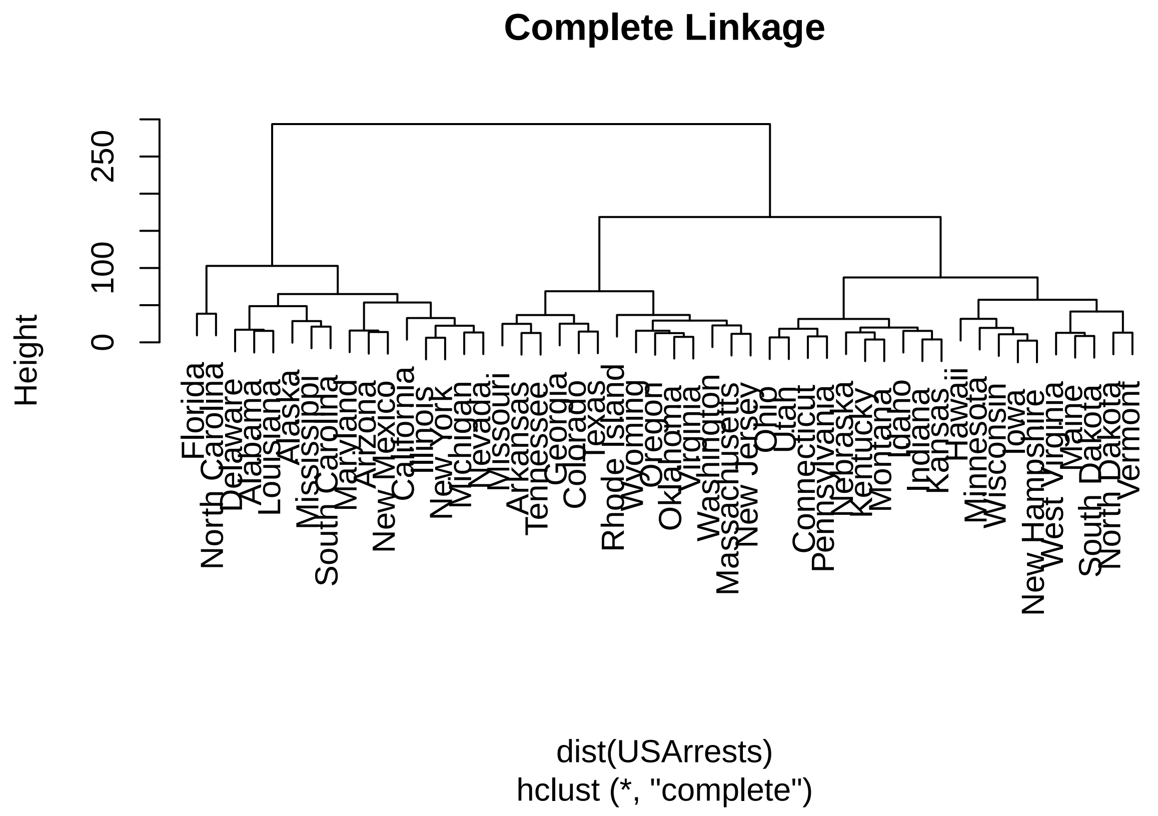
```
PC4.PVEhand
```

```
## [1] 0.04335752
```

Question 9

a

```
Arr.complete = hclust(dist(USArrests),method='complete')
plot(Arr.complete,main="Complete Linkage")
```



b

```r
print(sort(cutree(Arr.complete,3),decreasing=TRUE))
```

```
##    Connecticut         Hawaii          Idaho        Indiana           Iowa 
##              3              3              3              3              3 
##         Kansas       Kentucky          Maine      Minnesota        Montana 
##              3              3              3              3              3 
##       Nebraska  New Hampshire   North Dakota           Ohio   Pennsylvania 
##              3              3              3              3              3 
##   South Dakota           Utah        Vermont  West Virginia      Wisconsin 
##              3              3              3              3              3 
##       Arkansas       Colorado        Georgia  Massachusetts       Missouri 
##              2              2              2              2              2 
##     New Jersey       Oklahoma         Oregon   Rhode Island      Tennessee 
##              2              2              2              2              2 
##          Texas       Virginia     Washington        Wyoming        Alabama 
##              2              2              2              2              1 
##         Alaska        Arizona     California       Delaware        Florida 
##              1              1              1              1              1 
##       Illinois      Louisiana       Maryland       Michigan    Mississippi 
##              1              1              1              1              1 
##         Nevada     New Mexico       New York North Carolina South Carolina 
##              1              1              1              1              1
```

c

```r
Arrest.scaled = scale(USArrests)
sd(Arrest.scaled[,1])
```

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

```r
sd(Arrest.scaled[,2])
```

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

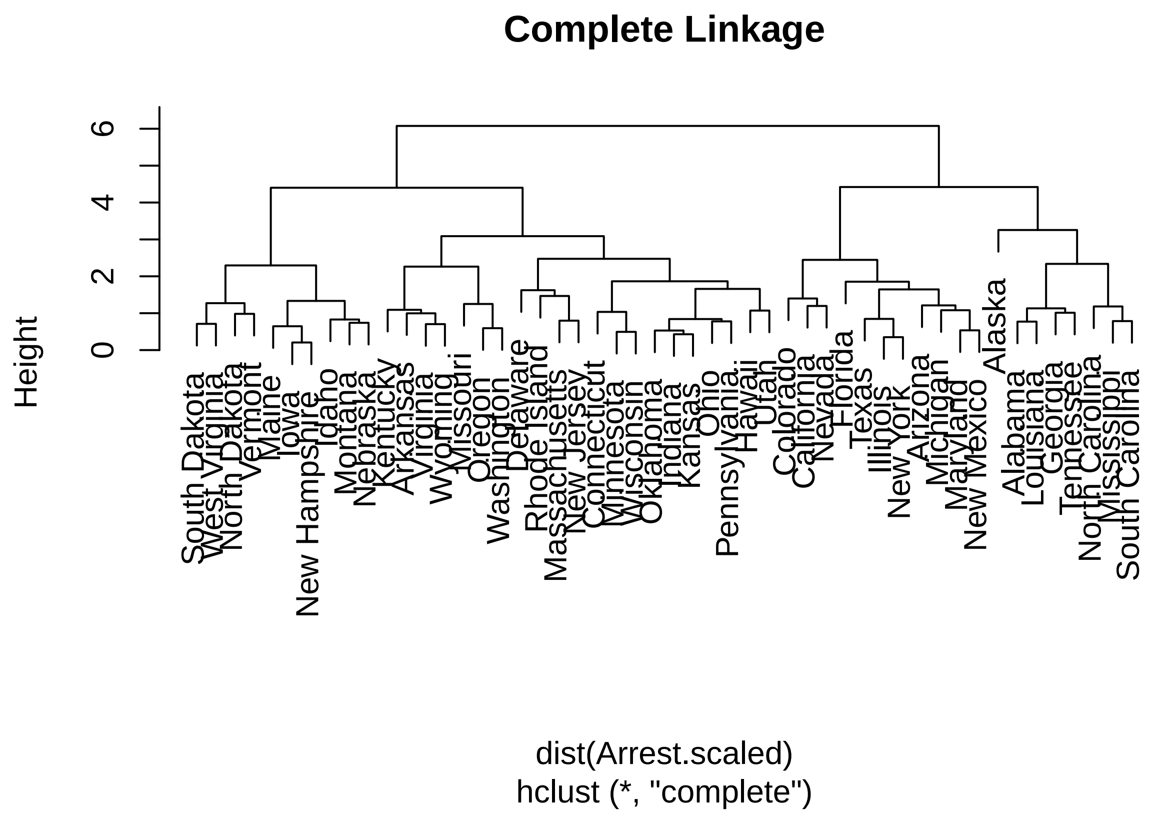
```r
sd(Arrest.scaled[,3])
```

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

```r
sd(Arrest.scaled[,4])
```

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

```r
Arr.complete.scaled= hclust(dist(Arrest.scaled),method='complete')
plot(Arr.complete.scaled,main="Complete Linkage")
```



```r
print(sort(cutree(Arr.complete.scaled,3),decreasing=TRUE))
```

```
##       Arkansas    Connecticut       Delaware         Hawaii          Idaho
##              3              3              3              3              3
##        Indiana           Iowa         Kansas       Kentucky          Maine
##              3              3              3              3              3
##  Massachusetts      Minnesota       Missouri        Montana       Nebraska
##              3              3              3              3              3
##  New Hampshire     New Jersey   North Dakota           Ohio       Oklahoma
##              3              3              3              3              3
##         Oregon   Pennsylvania   Rhode Island   South Dakota           Utah
##              3              3              3              3              3
##        Vermont       Virginia     Washington  West Virginia      Wisconsin
##              3              3              3              3              3
##        Wyoming        Arizona     California       Colorado        Florida
##              3              2              2              2              2
##       Illinois       Maryland       Michigan         Nevada     New Mexico
##              2              2              2              2              2
##       New York          Texas        Alabama         Alaska        Georgia
##              2              2              1              1              1
##      Louisiana    Mississippi North Carolina South Carolina      Tennessee
##              1              1              1              1              1
```

d

```r
threeclust = sum(cutree(Arr.complete,3)==3)
threeclustScale = sum(cutree(Arr.complete.scaled,3)==3)
twoclust = sum(cutree(Arr.complete,3)==2)
twoclustScale = sum(cutree(Arr.complete.scaled,3)==2)
oneclust = sum(cutree(Arr.complete,3)==1)
oneclustScale = sum(cutree(Arr.complete.scaled,3)==1)

print('3 groups raw vs scaled:')
```

```
## [1] "3 groups raw vs scaled:"
```

```r
threeclust
```

```
## [1] 20
```

```r
threeclustScale
```

```
## [1] 31
```

```r
print('2 groups raw vs scaled:')
```

```
## [1] "2 groups raw vs scaled:"
```

```r
twoclust
```

```
## [1] 14
```

```r
twoclustScale
```

```
## [1] 11
```

```r
print('1 group raw vs scaled:')
```

```
## [1] "1 group raw vs scaled:"
```

```r
oneclust
```

```
## [1] 16
```

```r
oneclustScale
```

```
## [1] 8
```

We can see using the cutree function, that prior to scaling, the clustering includes less states in larger clusters, and more states in smaller clusters. This likely is a function of the disproportionate variability in the unscaled dataset thus more states are more unique, and thereby more singleton clusters are created. Disproportionate variability in unscaled data can be driven by high values in certain categories, for example urban population in california will be extremely high, and given we are using euclidean distance will be far away from other states if unscaled, this has a disproportionate effect on the state's position wrt arrests, but is not what we are interested in. So yes, I would scale data, IN THIS CASE. in general, the problem is very contextual. For example, if we simply had number of arrests per state, with no other variables included, I perhaps would not need to scale.

We can see in unscaled data california is in a singleton cluster, after scaling it is in a 2 state cluster. If we look at the crime values for california and compare to Nevada, whome it was most similarly clustered with, the crime values are highly similar

```
USArrests['California',]
```

```
##            Murder Assault UrbanPop Rape
## California      9     276       91 40.6
```

```
USArrests['Nevada',]
```

```
##        Murder Assault UrbanPop Rape
## Nevada   12.2     252       81   46
```

```
USArrests['Arizona',]
```

```
##         Murder Assault UrbanPop Rape
## Arizona    8.1     294       80   31
```

versus in the unscaled dendrogram, height wise it is similarly singleton to missouri or georgia. Which across all values seem much less similar to values seen in California

```
USArrests['California',]
```

```
##            Murder Assault UrbanPop Rape
## California      9     276       91 40.6
```

```
USArrests['Georgia',]
```

```
##         Murder Assault UrbanPop Rape
## Georgia   17.4     211       60 25.8
```

```
USArrests['Missouri',]
```

```
##          Murder Assault UrbanPop Rape
## Missouri      9     178       70 28.2
```

Question 11

a

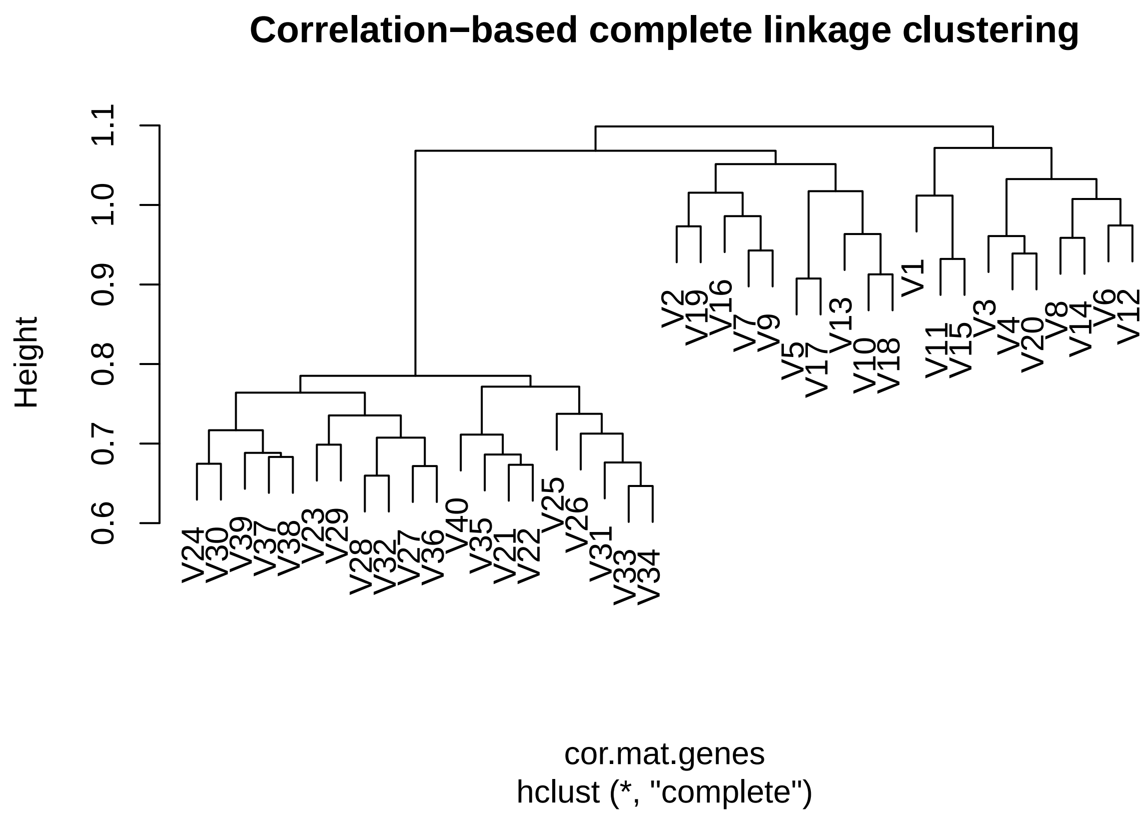
```
getwd()
```

```
## [1] "C:/Users/suhaas.adiraju/Desktop/Statistical ML/StatisticalMLCourse"
```

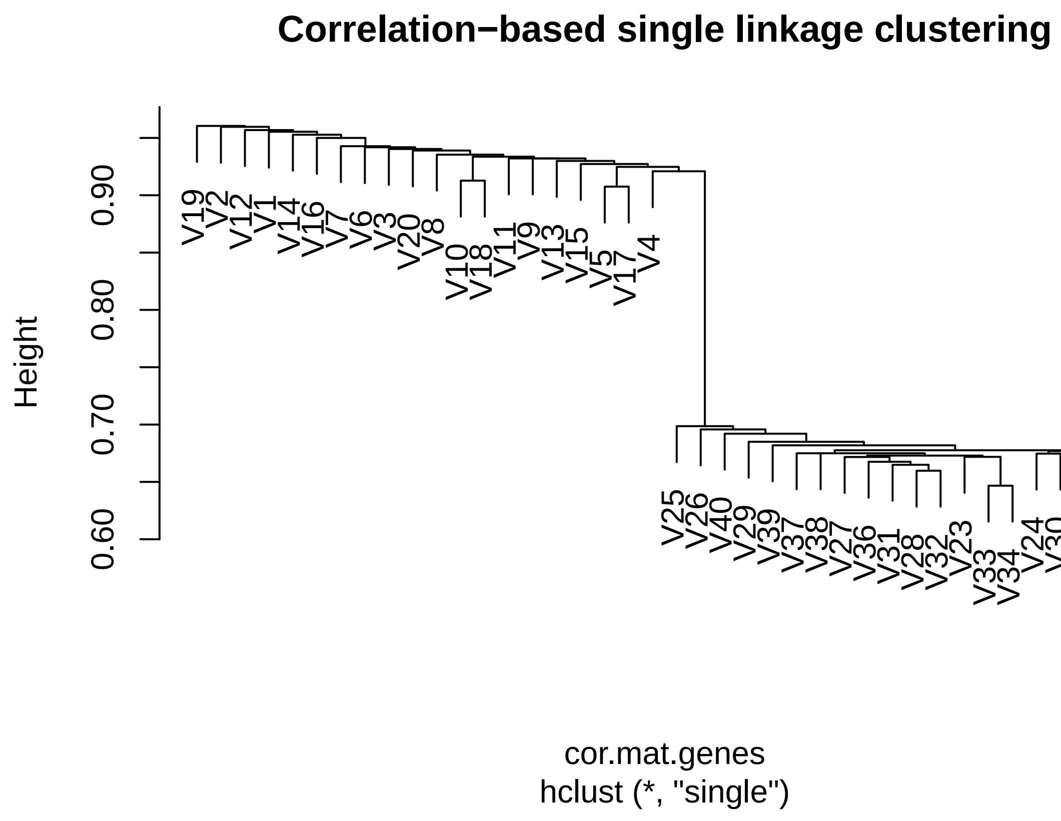
```
Ch.Data= read.csv('C:\\Users\\suhaas.adiraju\\Desktop\\Statistical ML\\StatisticalMLCourse\\Ch10Ex11.cs
```

b

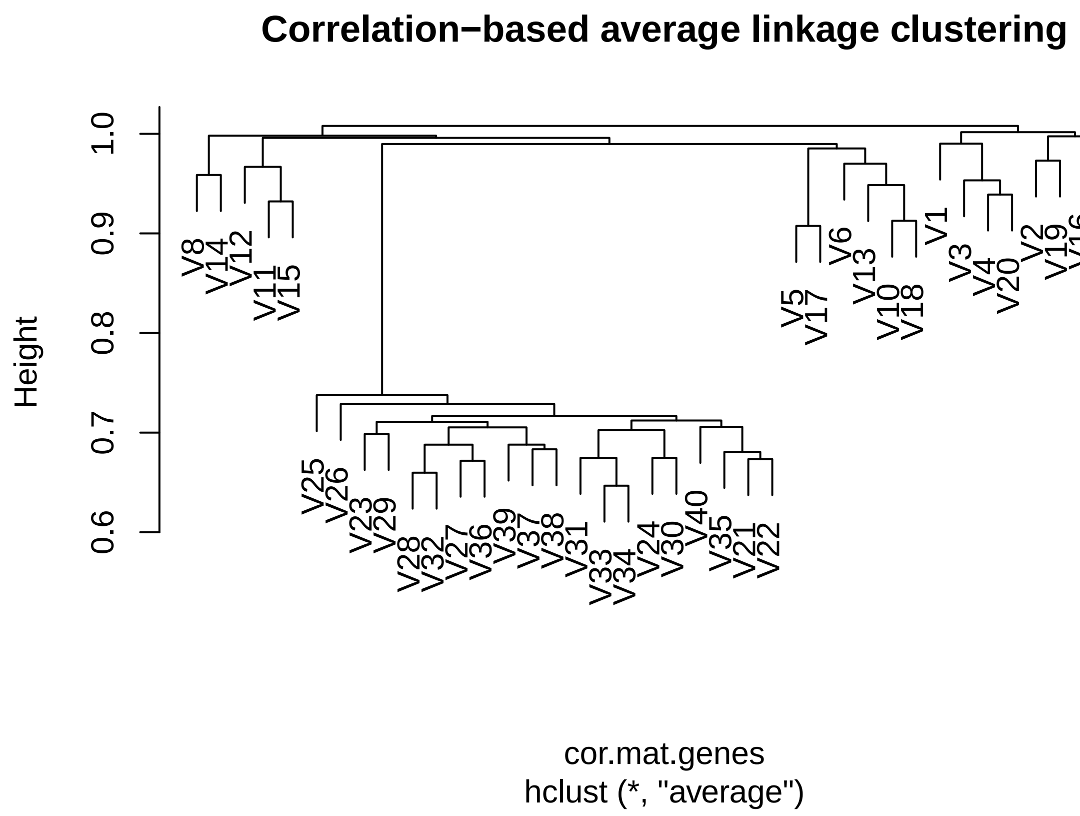
```
sample = names(Ch.Data)
cor.mat.genes= as.dist(1-cor((Ch.Data)))
plot(hclust(cor.mat.genes,method='complete'),main='Correlation-based complete linkage clustering')
```



```
cor.mat.genes= as.dist(1-cor((Ch.Data)))
plot(hclust(cor.mat.genes,method='single'),main='Correlation-based single linkage clustering')
```



```
cor.mat.genes= as.dist(1-cor((Ch.Data)))
plot(hclust(cor.mat.genes,method='average'),main='Correlation-based average linkage clustering')
```



The separation of genes in to simply two groups is not trivial/automatic, but depends on the clustering used. none of the dendrograms only form two clusters, but using average and complete distances certainly provide a better 2 cluster estimation compared to single. Another way to show this result

```
sort(cutree(hclust(cor.mat.genes,method='average'),2))
```

```
##  V1  V2  V3  V4  V7  V9 V16 V19 V20  V5  V6  V8 V10 V11 V12 V13 V14 V15 V17 V18
##   1   1   1   1   1   1   1   1   1   2   2   2   2   2   2   2   2   2   2   2
## V21 V22 V23 V24 V25 V26 V27 V28 V29 V30 V31 V32 V33 V34 V35 V36 V37 V38 V39 V40
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
```

```
sort(cutree(hclust(cor.mat.genes,method='complete'),2))
```

```
##  V1  V3  V4  V6  V8 V11 V12 V14 V15 V20  V2  V5  V7  V9 V10 V13 V16 V17 V18 V19
##   1   1   1   1   1   1   1   1   1   1   2   2   2   2   2   2   2   2   2   2
## V21 V22 V23 V24 V25 V26 V27 V28 V29 V30 V31 V32 V33 V34 V35 V36 V37 V38 V39 V40
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
```

```
sort(cutree(hclust(cor.mat.genes,method='single'),2))
```

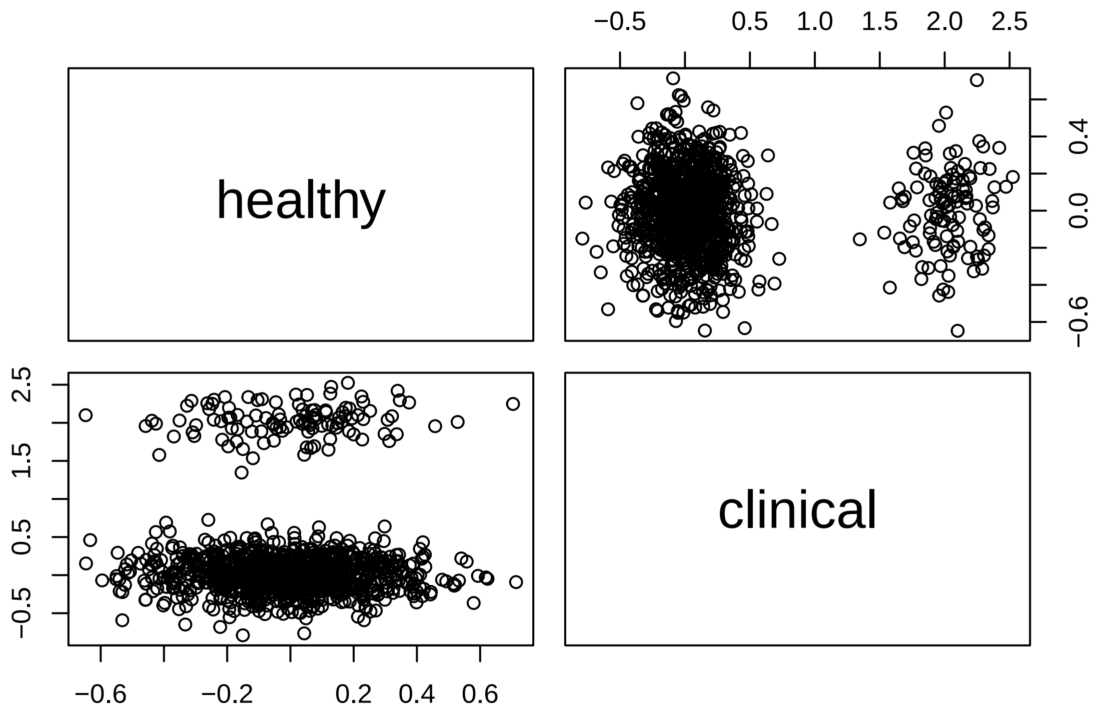
```
##  V1  V2  V3  V4  V5  V6  V7  V8  V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V20 V21
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## V22 V23 V24 V25 V26 V27 V28 V29 V30 V31 V32 V33 V34 V35 V36 V37 V38 V39 V40 V19
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   2
```

We can see single linkage clustered every sample except for V19 in cluster 1, this is clearly not useful.

c

There are several ways we can explore this question, I would start with very basic exploratory analysis and plotting. We know our two groups already, this is an advantage, we can average each gene(row) across samples, within each respective group of healthy versus diseased. Then use the pairs function to see if there are major differences plotting healthy versus diseased gene average values:

```
healthy = apply(Ch.Data[,1:20],1,mean)
clinical = apply(Ch.Data[,21:40],1,mean)
gene.frame = data.frame(healthy,clinical)
pairs(gene.frame)
```



Look at that! it seems that indeed there is a subgroup of genes that express much higher mean values in the clinical group. We could pull out the identity of these genes by setting a threshold and indexing the data frame

```
clinical.hit.genes= which((gene.frame$clinical>1.0),useNames = TRUE)
clinical.hit.genes
```

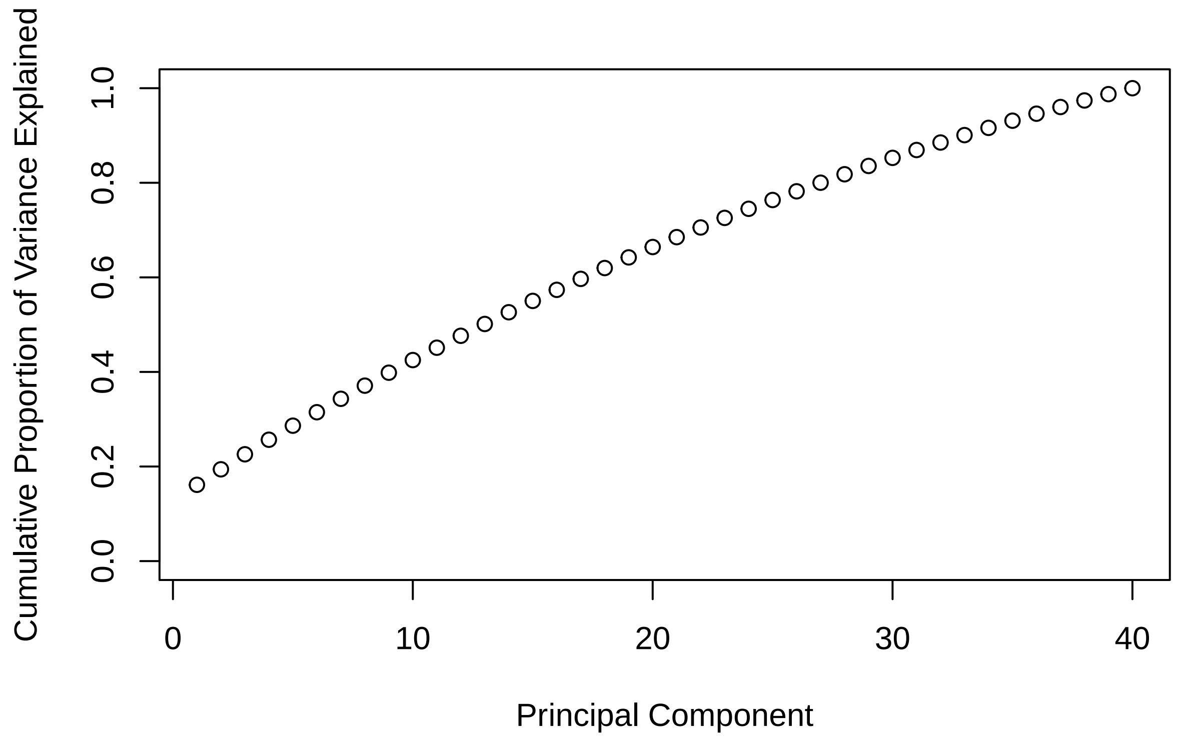
```
##   [1]  11  12  13  14  15  16  17  18  19  20 501 502 503 504 505 506 507 508
##  [19] 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526
##  [37] 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544
##  [55] 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562
##  [73] 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580
##  [91] 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598
## [109] 599 600
```

although this is useful information, when we average across samples in each group, we are assuming a lot of stability across samples or rather overlooking the within group variability, as well as other nuances. To take a more fine-grained approach of preserving individual sample contributions lets use PCA

```
pr.out.gene = prcomp((Ch.Data), scale=TRUE)
pr.var.gene = pr.out.gene$sdev^2
PVE.gene = pr.var.gene/sum(pr.var.gene)
PVE.gene
```

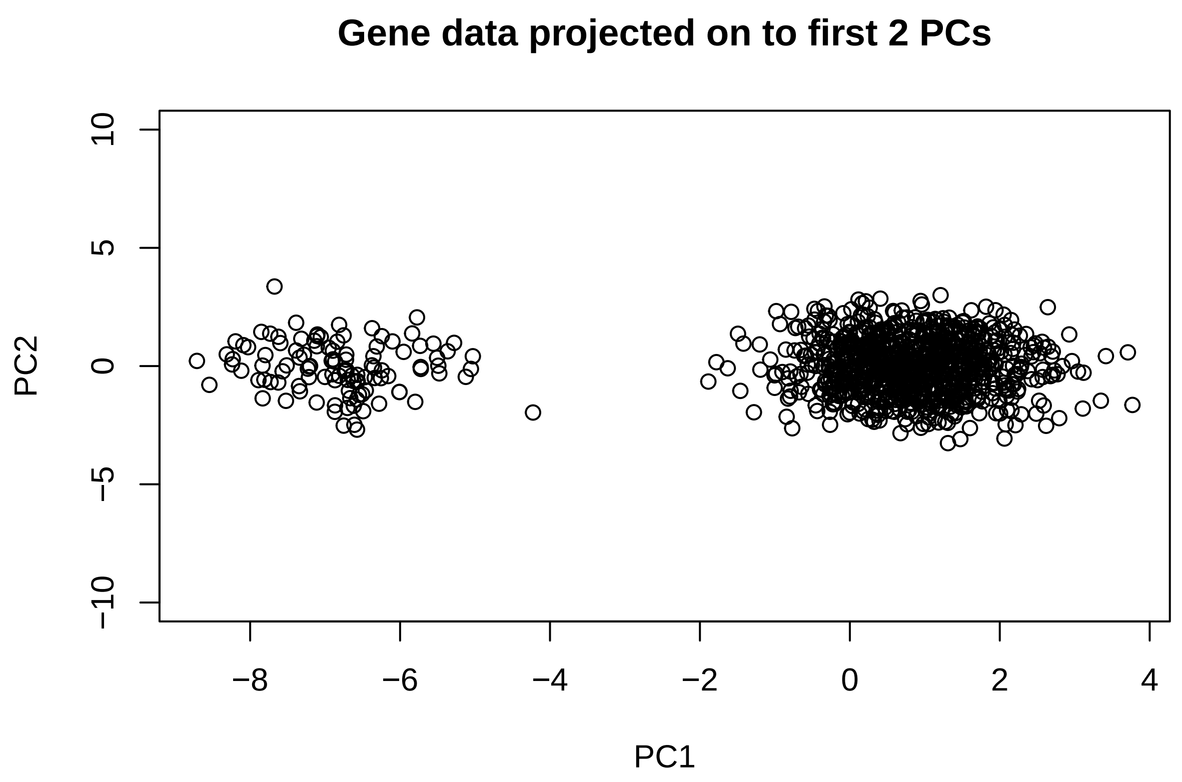
```
##  [1] 0.16130137 0.03280179 0.03176830 0.03075558 0.02972109 0.02847757
##  [7] 0.02833663 0.02783324 0.02736489 0.02674424 0.02610345 0.02519941
## [13] 0.02499759 0.02484029 0.02396715 0.02339585 0.02313948 0.02299186
## [19] 0.02250569 0.02193219 0.02091515 0.02041673 0.02023852 0.01921437
## [25] 0.01863679 0.01836530 0.01819977 0.01786939 0.01757227 0.01708818
## [31] 0.01658588 0.01587270 0.01569743 0.01524706 0.01516708 0.01485582
## [37] 0.01409545 0.01388385 0.01331786 0.01258275
```

```
plot(cumsum (PVE.gene ), xlab="Principal Component", ylab = "
Cumulative Proportion of Variance Explained", ylim=c(0,1) ,
)
```



We see from our cumulative plot and the proportion of variance explained values that our first component captures the highest proportion of variance, and all following components add about the same small amount of information. lets plot our data mapped in the first 2 components.

```r
plot(pr.out.gene$x[,'PC1'],pr.out.gene$x[,'PC2'],ylim = c(-10,10),main='Gene data projected on to first
```



We can see in this plot the data varies a lot on PC1, and a little on PC2, and via the PC1 variance, there is a distinct data cloud with large negative score values, these could be positive too, as PCA weights are identical up to sign changes. Basically, these genes differ alot among the samples. We should index them, and sort them, to see which genes they are and which are the most variable along PC1

```r
gene.hits.pca.idx = which(pr.out.gene$x[,'PC1'] <= -4,useNames = TRUE)
top.genes = data.frame(gene.hits.pca.idx, -(pr.out.gene$x[gene.hits.pca.idx
,'PC1']))
sorted.top.genes = order(top.genes$X..pr.out.gene.x.gene.hits.pca.idx...PC1...,decreasing=TRUE)
top.genes[sorted.top.genes,1]
```

```
##   [1] 551 589 508 548 509 511 566 565 540 534 561 502 584 586  12 558 593 568
##  [19] 582 578 503  13 549 572 538 505 554 587 545 550 521  11 522 526 560 555
##  [37] 570 597 559 541 516 590 571 576 517  14 546 600 520 529 515 514 506 535
##  [55] 537 596 542 523 595 594 512 599 562 591 536 556 544 527 575  20 533 564
##  [73] 530 539 579  18 513 543 567 524 501 569 583 563 588 531  16 585 592 581
##  [91]  19  15 574 553 580 598  17 519 507 547 552 504 557 510 577 573 528 525
## [109] 532 518
```