Michael Leibert

Math 504

Homework 13

2 With this assignment you will find the file `users-shows.txt`. This file gives a 9985 by 563 matrix, call it A, corresponding to 563 television shows and 9985 television users.. The matrix is composed of 0's and 1's. A 1 means that the user likes the show, a 0 means they do not. The shows, if you are interested, are listed in the file `shows.txt`. The 500th user, let's call him Alex, has had his preferences for the first 100 shows removed from the matrix and replaced with all 0's. His actual preferences are in the file `Alex.txt`. Your goal is to use the svd to suggest 5 shows from the first 100 that you believe Alex would like. You can then check your answer against his actual preferences. You should use R's svd function to compute the SVD of A.

(a) As a warmup to this problem, show following. Given the SVD decomposition of $A = USV^T$, show that $A = \sum_{i=1}^{563} s_i u^{(i)} (v^{(i)})^T$ where $u^{(i)}$ and $v^{(i)}$ are the ith columns of U and V respectively.

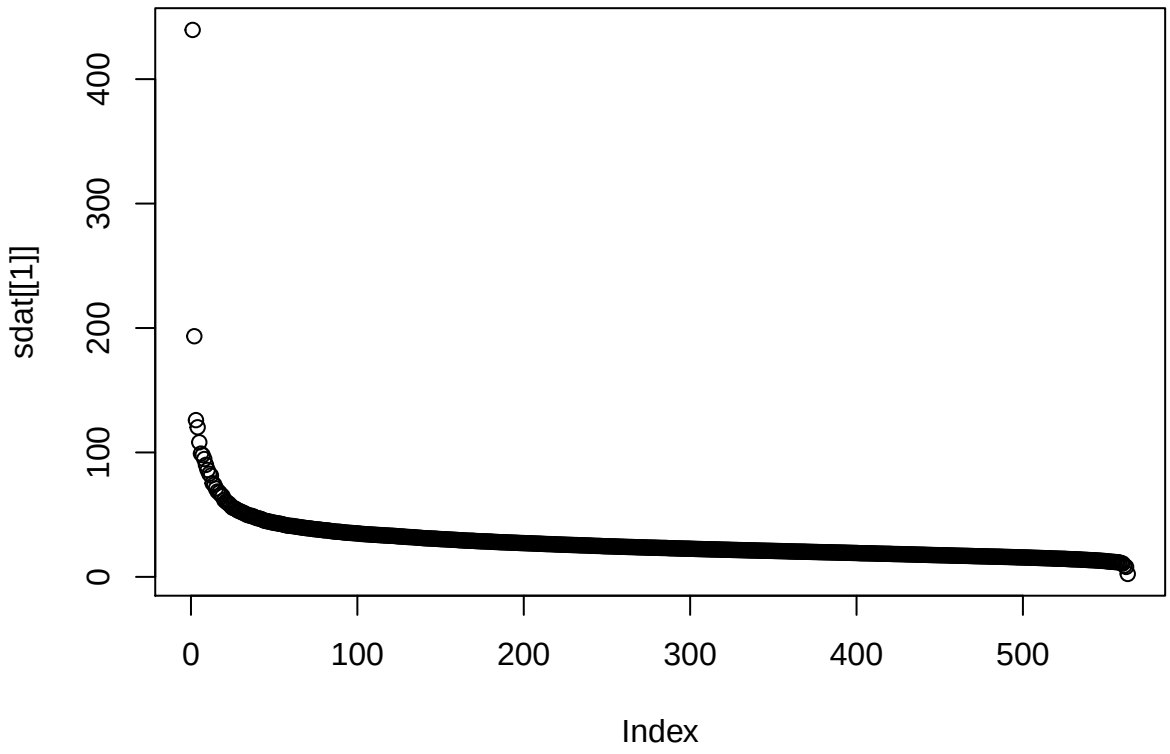
$$
\begin{aligned}
A &= USV^T \\
&= \begin{pmatrix} u_{11} & u_{12} & ... & u_{1m} \\ u_{21} & u_{22} & ... & u_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1} & u_{m2} & ... & u_{mm} \end{pmatrix} \begin{pmatrix} s_{11} & & & \\ & s_{22} & & \\ & & \ddots & \\ & & & s_{nn} \end{pmatrix} \begin{pmatrix} v_{11} & v_{21} & ... & v_{m1} \\ v_{12} & v_{22} & ... & v_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1m} & v_{2m} & ... & v_{mm} \end{pmatrix} \\
&= \begin{pmatrix} u_{11}s_{11} & u_{12}s_{22} & ... & u_{1n}s_{nn} \\ u_{21}s_{11} & u_{22}s_{22} & ... & u_{2n}s_{nn} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1}s_{11} & u_{m2}s_{22} & ... & u_{mn}s_{nn} \end{pmatrix} \begin{pmatrix} v_{11} & v_{21} & ... & v_{m1} \\ v_{12} & v_{22} & ... & v_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1m} & v_{2m} & ... & v_{mm} \end{pmatrix} \\
&= \begin{pmatrix} u_{11}s_{11}v_{11} + u_{12}s_{22}v_{12} + ... + u_{1n}s_{nn}v_{1n} & u_{11}s_{11}v_{21} + u_{12}s_{22}v_{22} + ... + u_{1n}s_{nn}v_{2n} & ... \\ u_{21}s_{11}v_{11} + u_{22}s_{22}v_{12} + ... + u_{2n}s_{nn}v_{1n} & u_{21}s_{11}v_{21} + u_{22}s_{22}v_{22} + ... + u_{2n}s_{nn}v_{2n} & ... \\ \vdots & \vdots & \ddots \\ u_{m1}s_{11}v_{11} + u_{m2}s_{22}v_{12} + ... + u_{mn}s_{nn}v_{1n} & u_{m1}s_{11}v_{21} + u_{m2}s_{22}v_{22} + ... + u_{mn}s_{nn}v_{2n} & ... \end{pmatrix} \\
&= \begin{pmatrix} u_{11}s_{11}v_{11} & u_{11}s_{11}v_{21} & ... & u_{11}s_{11}v_{n1} \\ u_{21}s_{11}v_{11} & u_{21}s_{11}v_{21} & ... & u_{21}s_{11}v_{n1} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1}s_{11}v_{11} & u_{m1}s_{11}v_{21} & ... & u_{m1}s_{11}v_{n1} \end{pmatrix} + \begin{pmatrix} u_{12}s_{22}v_{12} & u_{12}s_{22}v_{22} & ... & u_{12}s_{22}v_{n2} \\ u_{22}s_{22}v_{12} & u_{22}s_{22}v_{22} & ... & u_{22}s_{22}v_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m2}s_{22}v_{12} & u_{m2}s_{22}v_{22} & ... & u_{m2}s_{22}v_{n2} \end{pmatrix} + ...+ \\
&= s_{11} \begin{pmatrix} u_{11}v_{11} & u_{11}v_{21} & ... & u_{11}v_{n1} \\ u_{21}v_{11} & u_{21}v_{21} & ... & u_{21}v_{n1} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1}v_{11} & u_{m1}v_{21} & ... & u_{m1}v_{n1} \end{pmatrix} + s_{22} \begin{pmatrix} u_{12}v_{12} & u_{12}v_{22} & ... & u_{12}v_{n2} \\ u_{22}v_{12} & u_{22}v_{22} & ... & u_{22}v_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m2}v_{12} & u_{m2}v_{22} & ... & u_{m2}v_{n2} \end{pmatrix} + ...+ \\
&= s_{11} \begin{pmatrix} u_{11} \\ u_{21} \\ \vdots \\ u_{m1} \end{pmatrix} \begin{pmatrix} v_{11} & v_{21} & ...v_{n1} \end{pmatrix} + s_{22} \begin{pmatrix} u_{12} \\ u_{22} \\ \vdots \\ u_{m2} \end{pmatrix} \begin{pmatrix} v_{12} & v_{22} & ...v_{n2} \end{pmatrix} + ... + s_{nn} \begin{pmatrix} u_{1n} \\ u_{2n} \\ \vdots \\ u_{mn} \end{pmatrix} \begin{pmatrix} v_{1n} & v_{2n} & ...v_{nn} \end{pmatrix} \\
&= \sum_{i=1}^{n=563} s_i u^{(i)} \left(v^{(i)} \right)^T
\end{aligned}
$$

(b) Compute the SVD of A and plot the singular values. How many singular values would accurately approximate this matrix? (What accurate means here is up to you.)

```
setwd("G:\\math\\504")
options(scipen=999)
 Norm <- function(w){  sqrt(sum(w^2))}
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
dat<-read.table("user-shows.txt")
shows<-read.table("shows.txt")
alex<-read.table("alex.txt");colnames(alex)<-as.vector(shows$V1)

sdat<-svd(as.matrix(dat))
par(mar=c(5.1,4.1,2.1,2.1))
plot(sdat[[1]])
```

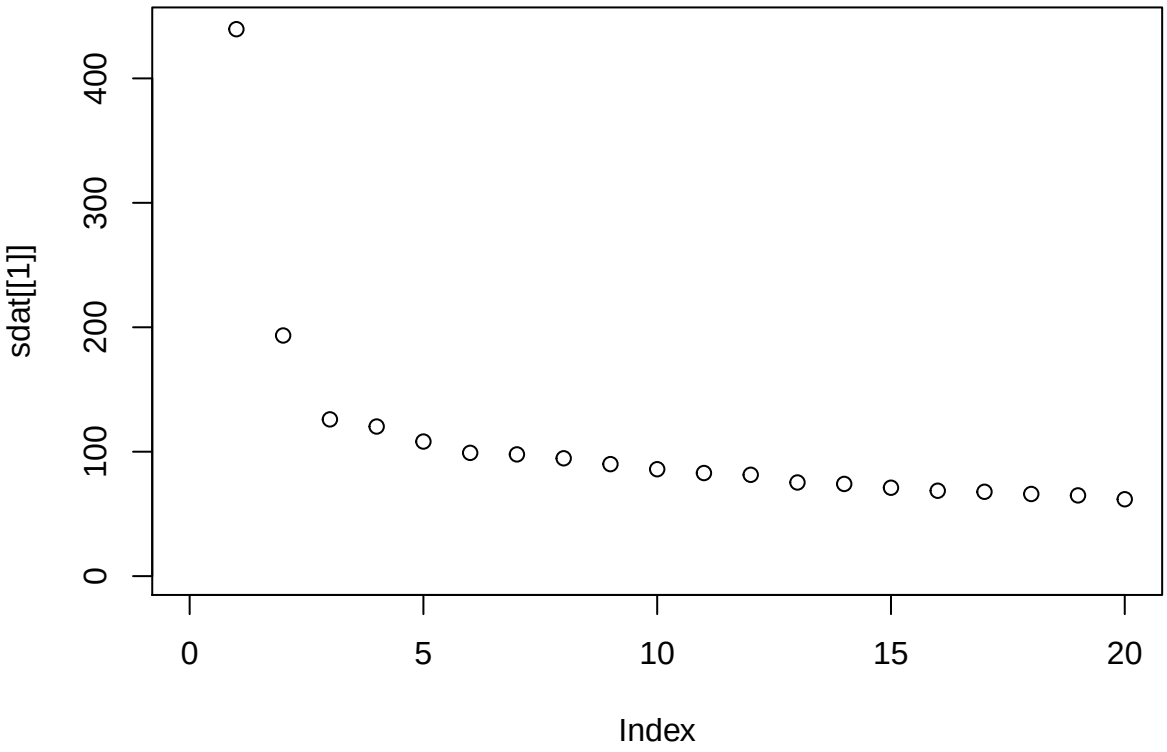



```
head( sum(sdat[[1]])-cumsum(sdat[[1]]) ); tail(sum(sdat[[1]])-cumsum(sdat[[1]]))
```

```
## [1] 15357.61 15164.23 15038.26 14918.02 14809.86 14710.79
```

```
## [1] 41.193799 30.080314 19.374069 10.347998  2.359547  0.000000
```

```
plot(sdat[[1]],xlim=c(0,20))
```

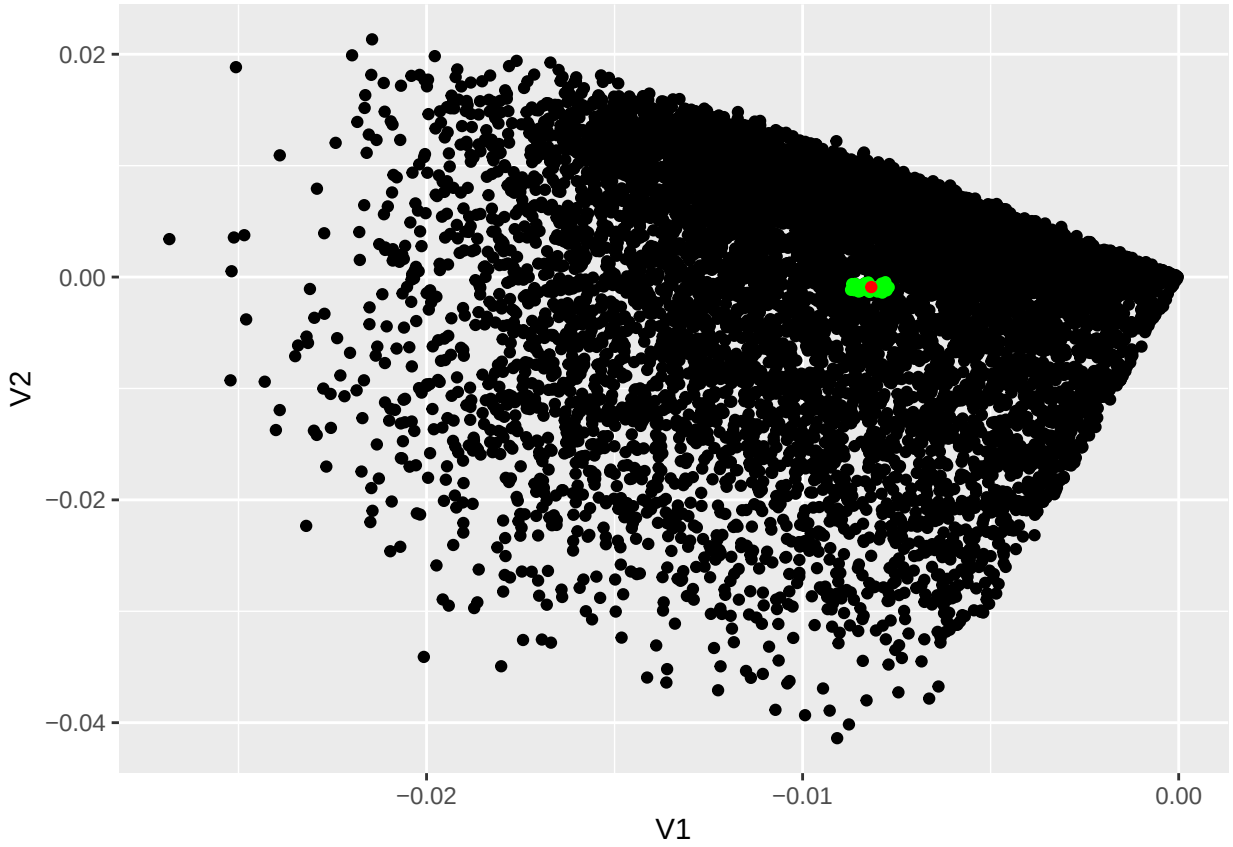


Five singular values may approximate this matrix accurately enough.

(c) Use the SVD to reduce the data to two dimensions as follows. Project the users onto the appropriate two dimenaional PCA space and plot; do the same for the shows. Using these projections, suggest five movies for Alex.

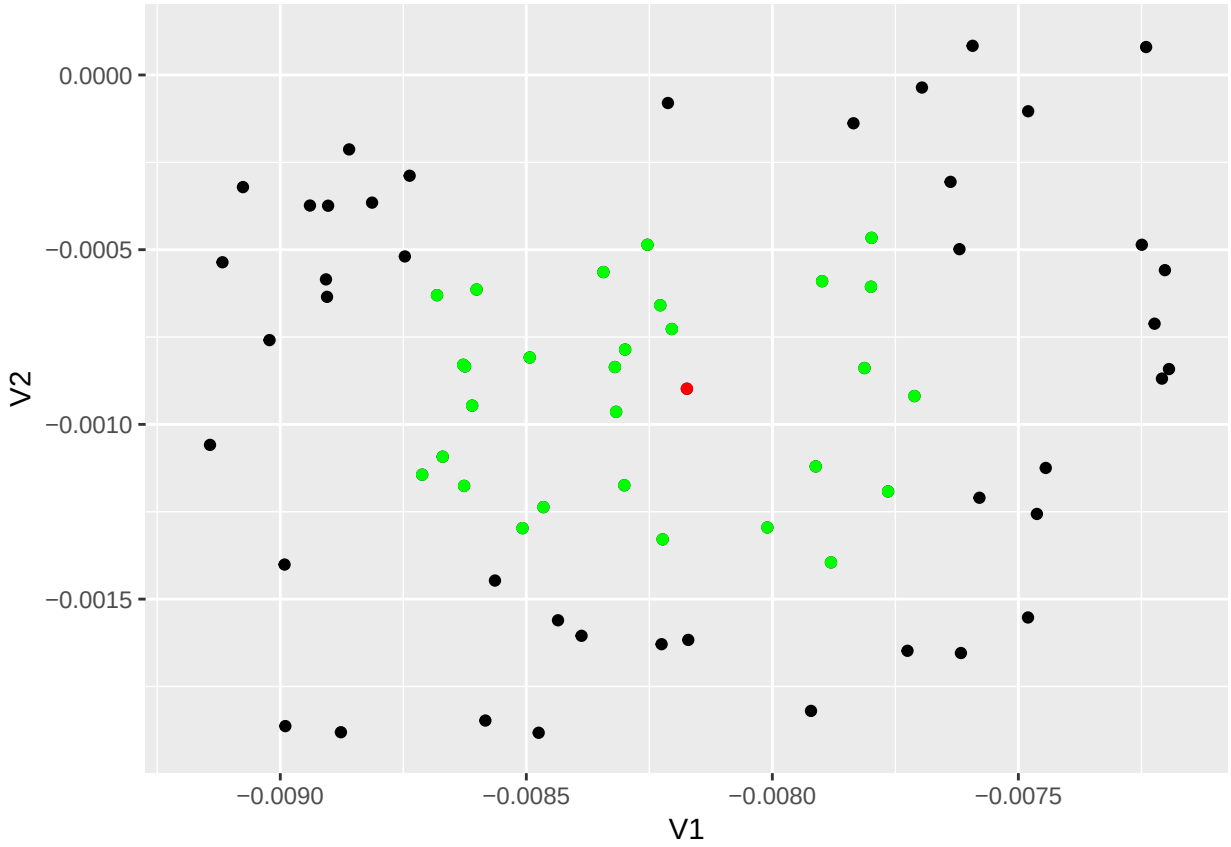
I did a euclidean distance measure for both people close to Alex and shows that are close to his.

```
U<-as.data.frame(sdat$u[,1:2]);V<-as.data.frame(sdat$v[,1:2] )
UU<-data.frame(apply(U , 1, function(z) Norm(z-as.numeric(U[ 500,]) )^2),   1:nrow(U))
buddies<-order(UU[,1])[ 2:30]

ggplot(U,aes(V1,V2)) + geom_point() +    geom_point(data=U[buddies,], colour="green") +
  geom_point(data=U[500,], colour="red")
```




```
ggplot(U,aes(V1,V2)) + geom_point() +    geom_point(data=U[500,], colour="red")+    xlim(-0.008173631-.001,
  ylim(-0.0008980471-.001, -0.0008980471+.001) + geom_point(data=U[buddies,], colour="green")
```

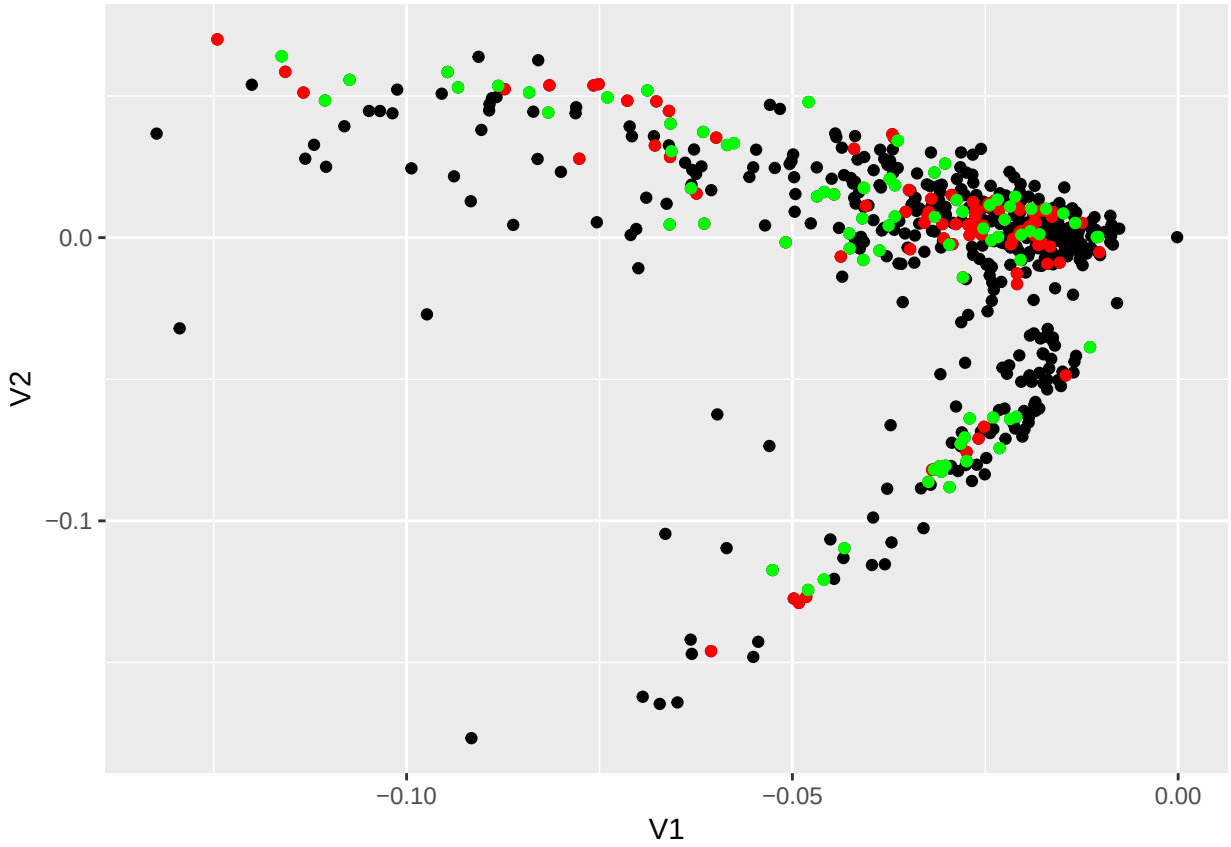
```
## Warning: Removed 9913 rows containing missing values (geom_point).
```




```
colnames(dat)<-as.vector(shows$V1)
recommend<-dat[buddies,]
recommend<-rbind(recommend,colSums(recommend))
```

```
for(i in 1:length(Mr) ){
    dattt<- (data.frame(apply(V[-NO,] , 1, function(z) Norm(z-
        V[which(dat[500,] == 1),][i,]  )^2),1:(nrow(V)-length(NO))) )
    Mr[i]<-order(dattt[  ,1])[1] }
```

```
ggplot(V,aes(V1,V2)) + geom_point() +
  geom_point(data=V[ which(dat[500,] == 1),], colour="red")  +
     geom_point(data=V[ Mr,], colour="green")
```




```
data.frame(colnames(recommend[ ,which( recommend[30, ] > 14 )])[-c(1,3)]  )
```

```
##   colnames.recommend...which.recommend.30......14.....c.1..3..
## 1                                2009 NCAA Basketball Tournament
## 2                                                    Family Guy
## 3                                              FOX 28 News at 10pm
## 4                                             10TV News HD at 11pm
## 5                                                2009 NBA Playoffs
```

3. Let A be an $m \times n$ matrix

(a) Go through the three cases for A - thin, square, and fat - and derive the formula for the psuedoinverse in terms of the SVD of A. (We did this in class, showing that $\tilde{A}^{-1} = V\tilde{S}^{-1}U^T$.)

Recall, $S_{m\times n} = \begin{pmatrix} s_{11} & & & \\ & s_{22} & & \\ & & \ddots & \\ & & & s_{nn} \\ & & & \end{pmatrix}$ and $S_{n\times m} = \begin{pmatrix} s_{11} & & & \\ & s_{22} & & \\ & & \ddots & \\ & & & s_{mm} \end{pmatrix}$

For the case $m > n$, a thin matrix

$$
\begin{aligned}
A_{m\times n}\ \boldsymbol{x}_{n\times 1} &= \boldsymbol{b}_{m\times 1} \\
U_{m\times m}\ S_{m\times n}\ V^T_{n\times n}\ \boldsymbol{x}_{n\times 1} &= \boldsymbol{b}_{m\times 1} \\
S_{m\times n}\ V^T_{n\times n}\ \boldsymbol{x}_{n\times 1} &= U^T_{m\times m}\ \boldsymbol{b}_{m\times 1} \\
S_{m\times n}\ \boldsymbol{y}_{n\times 1} &= \boldsymbol{z}_{m\times 1}
\end{aligned}
$$

We now have a diagonal matrix $S_{m\times n}$. If we want to get as close as possible to z, we can write:

$$
y_1 = \frac{z_1}{s_{11}},\ y_2 = \frac{z_2}{s_{22}}\ ,\ ...,\ y_n = \frac{z_n}{s_{nn}}
$$

We can't do anything about z_{n+1} to z_m. So our pseudoinverse is an $n \times m$ matrix,

$$
\widetilde{S}^{-1} = \begin{pmatrix}
\frac{1}{s_{11}} & & & \\
& \frac{1}{s_{22}} & & \\
& & \ddots & \\
& & & \frac{1}{s_{nn}} \\
& & &
\end{pmatrix}
$$

$$
\begin{aligned}
\boldsymbol{y}_{n\times 1} &= \widetilde{S}^{-1}_{n\times m}\ U^T_{m\times m}\ \boldsymbol{b}_{m\times 1} \\
\boldsymbol{x}_{n\times 1} &= V_{n\times n}\ \widetilde{S}^{-1}_{n\times m}\ U^T_{m\times m}\ \boldsymbol{b}_{m\times 1} \\
\boldsymbol{x}_{n\times 1} &= \widetilde{A}^{-1}_{n\times m}\ \boldsymbol{b}_{m\times 1}
\end{aligned}
$$

For the case $m = n$, a square matrix

$$
\begin{aligned}
A_{n\times n}\ \boldsymbol{x}_{n\times 1} &= \boldsymbol{b}_{n\times 1} \\
U_{n\times n}\ S_{n\times n}\ V^T_{n\times n}\ \boldsymbol{x}_{n\times 1} &= \boldsymbol{b}_{n\times n} \\
S_{n\times n}\ V^T_{n\times n}\ \boldsymbol{x}_{n\times 1} &= U^T_{n\times n}\ \boldsymbol{b}_{n\times 1} \\
S_{n\times n}\ \boldsymbol{y}_{n\times 1} &= \boldsymbol{z}_{n\times 1}
\end{aligned}
$$

We now have a diagonal matrix $S_{n\times n}$ with only one solution. In this case the pseudoinverse is actually the inverse:

$$
y_1 = \frac{z_1}{s_{11}},\ y_2 = \frac{z_2}{s_{22}}\ ,\ ...,\ y_n = \frac{z_n}{s_{nn}}
$$

$$\widetilde{S}^{-1} = \begin{pmatrix} \frac{1}{s_{11}} & & & \\ & \frac{1}{s_{22}} & & \\ & & \ddots & \\ & & & \frac{1}{s_{nn}} \end{pmatrix}$$

$$\begin{aligned} \boldsymbol{y}_{n\times 1} &= \widetilde{S}^{-1}_{n\times n}\ U^T_{n\times n}\ \boldsymbol{b}_{n\times 1} \\ \boldsymbol{x}_{n\times 1} &= V_{n\times n}\ \widetilde{S}^{-1}_{n\times n}\ U^T_{n\times n}\ \boldsymbol{b}_{n\times 1} \\ \boldsymbol{x}_{n\times 1} &= \widetilde{A}^{-1}_{n\times n}\ \boldsymbol{b}_{n\times 1} \\ \boldsymbol{x}_{n\times 1} &= A^{-1}_{n\times n}\ \boldsymbol{b}_{n\times 1} \end{aligned}$$

For the case $m < n$, a fat matrix

$$\begin{aligned} A_{m\times n}\ \boldsymbol{x}_{n\times 1} &= \boldsymbol{b}_{m\times 1} \\ U_{m\times m}\ S_{m\times n}\ V^T_{n\times n}\ \boldsymbol{x}_{n\times 1} &= \boldsymbol{b}_{m\times 1} \\ S_{m\times n}\ V^T_{n\times n}\ \boldsymbol{x}_{n\times 1} &= U^T_{m\times m}\ \boldsymbol{b}_{m\times 1} \\ S_{m\times n}\ \boldsymbol{y}_{n\times 1} &= \boldsymbol{z}_{m\times 1} \end{aligned}$$

We now have a diagonal matrix $S_{m\times n}$. We also have many solutions, so we will choose the Penrose pseudoinverse.

$$y_1 = \frac{z_1}{s_{11}},\ y_2 = \frac{z_2}{s_{22}}\ ,\ ...,\ y_n = \frac{z_m}{s_{nn}} \text{ and } y_{m+1} = y_n = 0$$

$$\widetilde{S}^{-1} = \begin{pmatrix} \frac{1}{s_{11}} & & & \\ & \frac{1}{s_{22}} & & \\ & & \ddots & \\ & & & \frac{1}{s_{nn}} \\ & & & \end{pmatrix}$$

$$\begin{aligned} \boldsymbol{y}_{n\times 1} &= \widetilde{S}^{-1}_{n\times m}\ U^T_{m\times m}\ \boldsymbol{b}_{m\times 1} \\ \boldsymbol{x}_{n\times 1} &= V_{n\times n}\ \widetilde{S}^{-1}_{n\times m}\ U^T_{m\times m}\ \boldsymbol{b}_{m\times 1} \\ \boldsymbol{x}_{n\times 1} &= \widetilde{A}^{-1}_{n\times m}\ \boldsymbol{b}_{m\times 1} \end{aligned}$$

(b) Suppose A is thin and we want to find the closest x such that $Ax = b$. There are two methods that we have used to do this: 1) using the normal equations and 2) using the pseudoinverse. Show that the two methods give the same result.

$$
\begin{aligned}
x &= \left(A^T A\right)^{-1} A^T b \\
&= \left(V S^T U^T U S V^T\right)^{-1} V S^T U^T b \\
&= \left(V S^T I S V^T\right)^{-1} V S^T U^T b \\
&= V \left(S^T S\right)^{-1} V^T V S^T U^T b \\
&= V \widetilde{S}^{-1} \widetilde{S}^T I S^T U^T b \\
&= V \widetilde{S}^{-1} \widetilde{S}^T S^T U^T b \\
x &= V \widetilde{S}^{-1} U^T b
\end{aligned}
$$

(c) Consider the dataset $y = \sin(x)$ for $x = 1, 2, \ldots, 5$. Let $\mathcal{F}$ be the set of all polynomials of degree 7 or less. Suppose we would like to fit the (x, y) data using $f(x) \in \mathcal{F}$. Explain how this reduces to solving $B\alpha = y$ where B is a model matrix. Is B thin, fat, or square? What does that imply about the number of solutions for $B\alpha = y$. Now choose $f(x)$ in two ways. For each, graph the datapoints and the resultant $f(x)$. Then, contrast the two methods.

B is the 5×7 model matrix,

$$
\begin{pmatrix}
h_1(x_1) & h_2(x_1) & \ldots & h_8(x_1) \\
h_1(x_2) & h_2(x_2) & \ldots & h_8(x_2) \\
\vdots & \vdots & \ddots & \vdots \\
h_1(x_5) & h_2(x_5) & \ldots & h_8(x_5)
\end{pmatrix}
$$

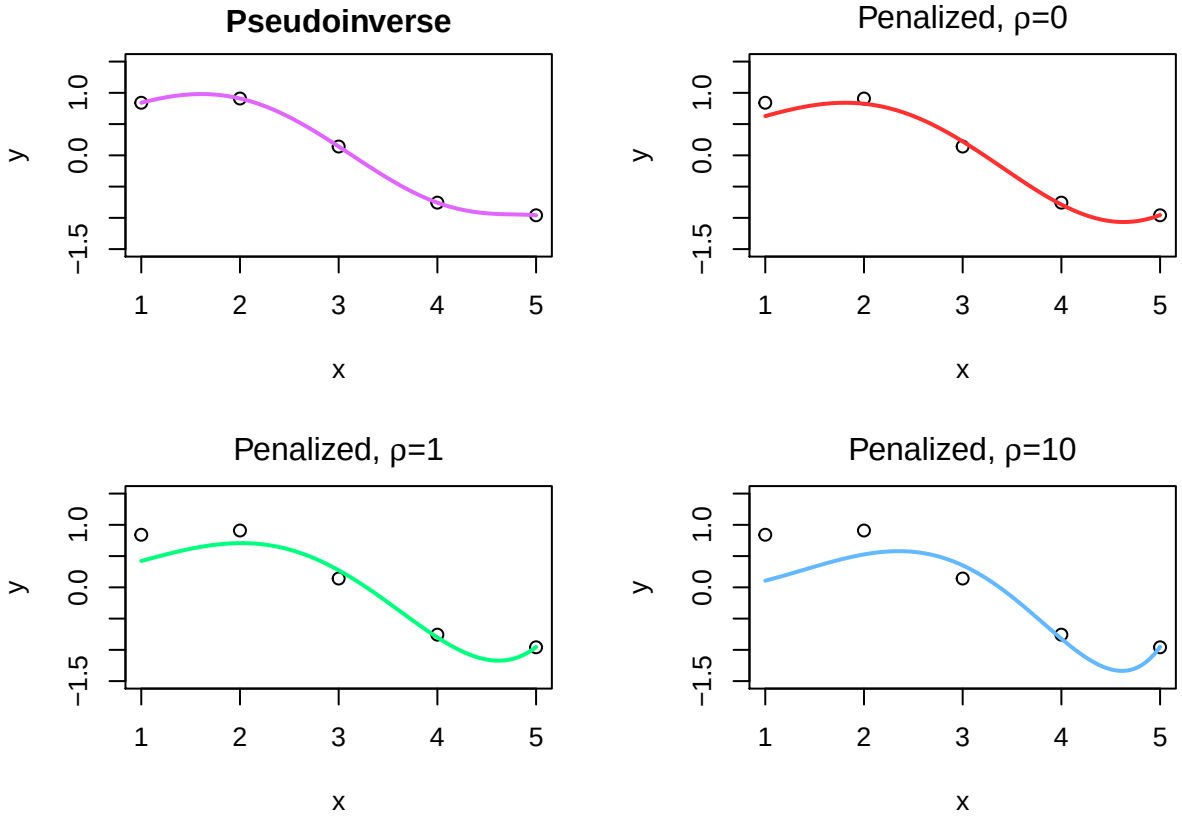
where $h_1(x) = 1$, $h_2(x) = x$, $h_3(x) = x^2$, $h_4(x) = x^3$, $h_5(x) = x^4$, $h_6(x) = x^5$, $h_7(x) = x^6$, and $h_8(x) = x^7$. We wish to solve for α by finding an inverse of B that will meet our needs. B is a fat matrix which has infinite many solutions.

1. Use the pseudoinverse to solve $B\alpha = y$.

2. Choose $f(x)$ by minimizing the penalized loss function: $\|y - B\alpha\|^2 + \rho\|\alpha\|^2$. Try $\rho = 0, 1, 10$.

```
x<-1:5;y<-as.matrix( sin(x) )

MM <- function(x) {
    nx <- length(x)
    m <- cbind(rep(1, nx), x, x^2, x^3, x^4, x^5, x^6 )
    colnames(m )<-NULL
    return(m)   }

B<-MM(x)

Alpha0<-solve(t(B) %*% B + 0 + diag(max(dim(B))) ) %*% t(B) %*% y
Alpha1<-solve(t(B) %*% B + 1 + diag(max(dim(B))) ) %*% t(B) %*% y
Alpha10<-solve(t(B) %*% B + 10 + diag(max(dim(B))) ) %*% t(B) %*% y

V1<-eigen(t(B)%*%B)$vectors
U1<-diag(dim(B)[1])
S1<-matrix(0,dim(B)[1],dim(B)[2])
```

```r
for( i in 1:min(c(dim(V1),dim(U1))) ){  a<-B%*%V1[,i]; U1[,i]<-a/Norm(a); S1[i,i]<-1/Norm(a)} ;rm(a)
PS<- V1 %*% t(S1) %*% t(U1) %*% y

x_grid <- seq(min(x), max(x), .001)
B_grid <- MM(x_grid)
y_gridps <-B_grid  %*% PS
y_grid0 <-B_grid  %*% Alpha0
y_grid1 <-B_grid  %*% Alpha1
y_grid10 <-B_grid  %*% Alpha10

par(mfrow=c(2,2),mar=c(5.1,4.1,2.1,2.1))
plot(x,y , ylim = c(-1.5,1.5) , main="Pseudoinverse" )
lines(x_grid, y_gridps , col="mediumorchid1", lwd=2)
plot(x,y, ylim = c(-1.5,1.5),main=expression(paste("Penalized, ", rho, "=0")))
lines(x_grid, y_grid0, col="firebrick1", lwd=2)
plot(x,y,ylim = c(-1.5,1.5),main=expression(paste("Penalized, ", rho, "=1")))
lines(x_grid, y_grid1, col="springgreen", lwd=2)
plot(x,y,ylim = c(-1.5,1.5),main=expression(paste("Penalized, ", rho, "=10")))
lines(x_grid, y_grid10, col="steelblue1", lwd=2)
```



For the penalized methods we can see that the ρ is giving a better fit for the smaller ρ's. We do note that all the penalized methods are just approximations rather than solutions. The pseudoinverse is actually a solution for $B\alpha = y$ and we see it going precisely through each point.

(3) In this problem you will implement K-means on two datasets

(a) Here is a fact I mentioned in class that is essential to the kmeans algorithm. Suppose you are given N points $x^{(i)}$ for $i = 1, 2, \ldots, N$, with each point in $\mathbb{R}^n$. Compute the point $m \in \mathbb{R}^n$ that minimizes the sum of squared distances from each $x^{(i)}$ to m:

$$\sum_{i=1}^{N} \|x^{(i)} - m\|^2 \tag{1}$$

Hint: You can take the gradient of this expression, set it to zero, and solve for m. You should find that m is the mean of the $x^{(i)}$.

$$\mathcal{L} = \sum_{i=1}^{N} \left|\left|x^{(i)} - m\right|\right|^2 = \left(x_1^{(1)} - m_1\right)^2 + \left(x_2^{(1)} - m_2\right)^2 + ... + \left(x_n^{(N)} - m_n\right)^2$$

$$\nabla\mathcal{L} = \begin{pmatrix} -2\left(x_1^{(1)} - m_1\right) - 2\left(x_1^{(2)} - m_1\right) - ... - 2\left(x_1^{(N)} - m_1\right) \\ -2\left(x_2^{(1)} - m_2\right) - 2\left(x_2^{(2)} - m_2\right) - ... - 2\left(x_2^{(N)} - m_2\right) \\ \vdots \\ -2\left(x_n^{(1)} - m_n\right) - 2\left(x_n^{(2)} - m_n\right) - ... - 2\left(x_n^{(N)} - m_n\right) \end{pmatrix}$$

We set $\nabla\mathcal{L} = \mathbf{0}$, and we solve for the jth partial derivative, which will be the same as all the partials 1 through n.

$$\begin{aligned} 0 &= -2\left(x_j^{(1)} - m_j\right) - 2\left(x_j^{(2)} - m_j\right) - ... - 2\left(x_j^{(N)} - m_j\right) \\ 0 &= -2\sum_{i=1}^{N}\left(x_j^{(i)} - m_j\right) \\ 0 &= \sum_{i=1}^{N} x_j^{(i)} - \sum_{i=1}^{N} m_j \\ 0 &= \sum_{i=1}^{N} x_j^{(i)} - Nm_j \\ Nm_j &= \sum_{i=1}^{N} x_j^{(i)} \\ m_j &= \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)} \end{aligned}$$

(b) Write a function **MyKmeans(x, K)** that accepts a data matrix X and the number of kmeans K and returns the solution to the kmeans problem as well as the number of iterations needed to reach the solution through the kmeans algorithm discussed in class. (You can check your answer against R's kmeans function and, if you like, you can also include a parameter in **MyKmeans** that chooses a starting value for the assignments or means.)

```
MyKmeans<-function(x,K){
    Norm <- function(w){  sqrt(sum(w^2))}
    N<-nrow(x); p<-ncol(x)
    x<-cbind(x,matrix(NA,N,1));names(x)[ncol(x)]<-"Assignment"
    meanz<-matrix(NA,N,p)
    x[,ncol(x)]<-sample(1:K,N,replace=T)
    mus<-matrix(NA,K,p);RAND<-sample(1:N,K);iter<-0
    for(i in 1:K){ mus[i,]<- as.numeric( x[RAND[i],1:p] ) }

    repeat{

    STOP<-x[,3];iter<-iter+1

    for( i in 1:K) { meanz[,i]<-apply(x[,1:p], 1,
        function(z) Norm(z-mus[i,]))^2 }

    x[,3]<-apply( meanz , 1, which.min)

    for(i in 1:K){ mus[i,]<-colSums( x[which(x[,3] == i),1:p] ) /
        nrow( x[which(x[,3] == i),1:p]) }

    if( all(x[,3] == STOP) == T ) {break} }
    ASSIGNMENT<-x[,3];names(ASSIGNMENT)<-1:length(ASSIGNMENT)

    newList <- list("Cluster Means" = mus, "Cluster Vector" = ASSIGNMENT,
        "Iterations" = iter)
    return(newList)
    }

MyKmeans(faithful,2);kmeans(faithful,2)
```

```
## $`Cluster Means`
##         [,1]     [,2]
## [1,] 2.09433 54.75000
## [2,] 4.29793 80.28488
##
## $`Cluster Vector`
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26
##   2   1   2   1   2   1   2   2   1   2   1   2   2   1   2   1   1   2   1   2   1   1   2   2   2   2
##
## $Iterations
## [1] 3
```

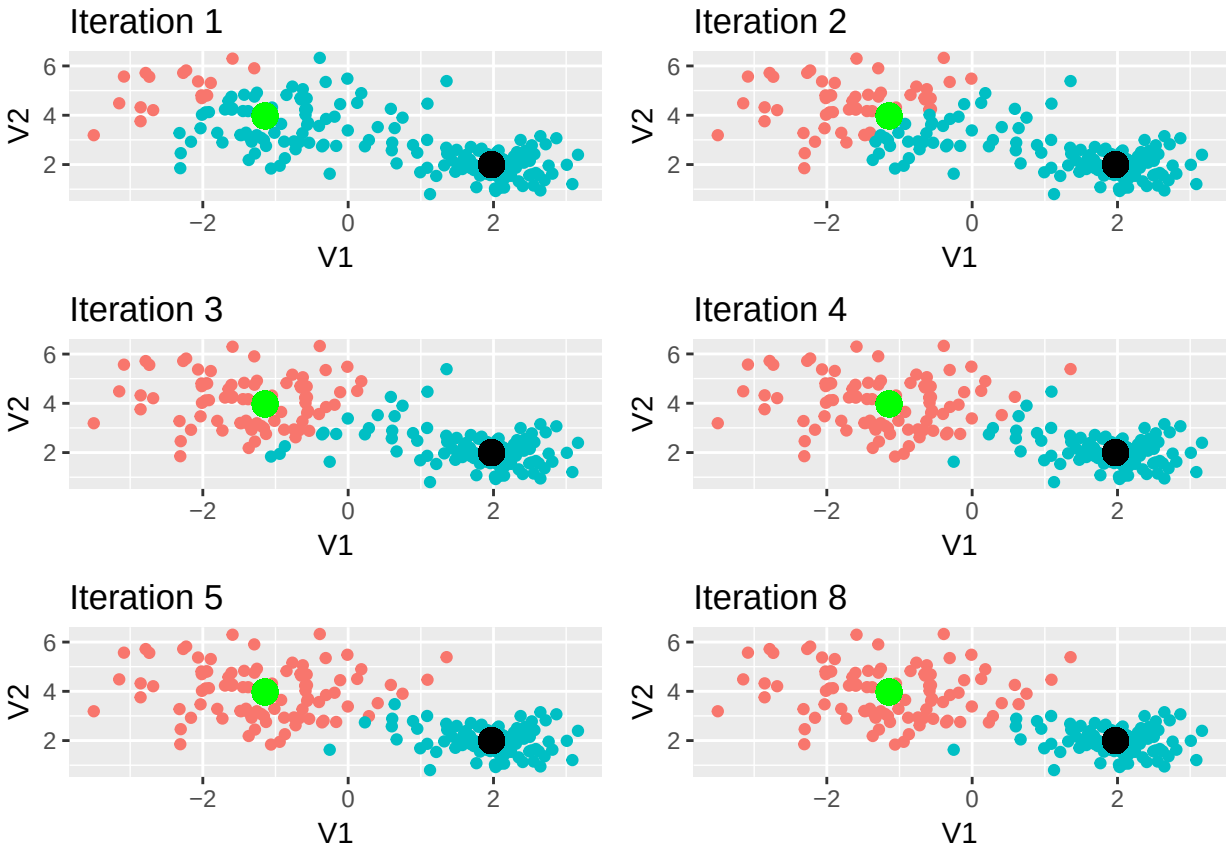
```
## K-means clustering with 2 clusters of sizes 172, 100
##
## Cluster means:
##   eruptions  waiting
## 1   4.29793 80.28488
## 2   2.09433 54.75000
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26
##   1   2   1   2   1   2   1   1   2   1   2   1   1   2   1   2   2   1   2   1   2   2   1   1   1   1
##
## Within cluster sum of squares by cluster:
## [1] 5445.591 3456.178
##  (between_SS / total_SS =  82.4 %)
##
```

```
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
```

(c) Apply **MyKmeans** to the attached dataset `synthetic kmeans data.csv` with $K = 2$. This is an artifical dataset for which the sample points are in $\mathbb{R}^2$. Plot the points of the dataset and the location of your 2 means at various iterations to see how the means move to their optimal location.

```
skd<-read.csv("synthetic kmeans data.csv")
MyKmeans(skd,2)
```

```
## $`Cluster Means`
##           [,1]     [,2]
## [1,] -1.143436 3.972099
## [2,]  1.973343 1.995245
##
## $`Cluster Vector`
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
##
## $Iterations
## [1] 6
```



(d) The attached dataset `tumor microarray data.csv` comes from the Elements of Statistical Learning book. Each row represents a cancer cell. The first column, labeled **cancer**, gives the type of the cancer cell (e.g.

RENAL, LEUKEMIA). The rest of the columns are numeric and give measurement of different proteins in the cell. The point here is to attempt to distinguish cancer cells by the level of proteins found in the cell. Perform K-means using R's kmeans function (or Python's). The cluster associated with a given mean are the sample points assigned to it. Try different K, and determine if the clusters formed separate the cancers (e.g. certain cancers are found within certain clusters). (See Elements of Statistical Learning Table 14.2, which does this for $K = 2$.)

```
tmd<-read.csv("tumor microarray data.csv")
#not running it directly - lots of output. will run in loop below to get information for k = 2,3,4,5
# MyKmeans(tmd[-1],2)
tmdKM<-tmd
for( i in 2:5){tmdKM[,ncol(tmdKM)+1]<-MyKmeans(tmd[-1],i)[[2]] }
colnames(tmdKM)[(ncol(tmdKM)-3):(ncol(tmdKM) )]
```

```
## [1] "V6832" "V6833" "V6834" "V6835"
```

```
table( tmdKM[,c(1,ncol(tmdKM)-3)] )
```

```
##              V6832
## cancer        1 2
##   BREAST      5 2
##   CNS         5 0
##   COLON       0 7
##   K562A-repro 0 1
##   K562B-repro 0 1
##   LEUKEMIA    0 6
##   MCF7A-repro 0 1
##   MCF7D-repro 0 1
##   MELANOMA    8 0
##   NSCLC       6 3
##   OVARIAN     5 1
##   PROSTATE    2 0
##   RENAL       9 0
##   UNKNOWN     1 0
```

```
table( tmdKM[,c(1,ncol(tmdKM)-2)] )
```

```
##              V6833
## cancer        1 2 3
##   BREAST      2 2 3
##   CNS         0 0 5
##   COLON       7 0 0
##   K562A-repro 1 0 0
##   K562B-repro 1 0 0
##   LEUKEMIA    6 0 0
##   MCF7A-repro 1 0 0
##   MCF7D-repro 1 0 0
##   MELANOMA    0 7 1
##   NSCLC       3 0 6
##   OVARIAN     0 0 6
##   PROSTATE    0 0 2
##   RENAL       0 0 9
##   UNKNOWN     0 0 1
```

```
table( tmdKM[,c(1,ncol(tmdKM)-1)] )
```

```
##              V6834
## cancer        1 2 3 4
##   BREAST      1 2 4 0
##   CNS         0 0 5 0
```

```
##   COLON       4 3 0 0
##   K562A-repro 0 1 0 0
##   K562B-repro 0 1 0 0
##   LEUKEMIA    0 5 0 1
##   MCF7A-repro 0 1 0 0
##   MCF7D-repro 0 1 0 0
##   MELANOMA    1 0 7 0
##   NSCLC       9 0 0 0
##   OVARIAN     6 0 0 0
##   PROSTATE    2 0 0 0
##   RENAL       8 0 1 0
##   UNKNOWN     1 0 0 0
```

```
table( tmdKM[,c(1,ncol(tmdKM))] )
```

```
##              V6835
## cancer        1 2 3 4 5
##   BREAST      4 0 0 1 2
##   CNS         0 0 0 0 5
##   COLON       1 3 3 0 0
##   K562A-repro 1 0 0 0 0
##   K562B-repro 1 0 0 0 0
##   LEUKEMIA    6 0 0 0 0
##   MCF7A-repro 1 0 0 0 0
##   MCF7D-repro 1 0 0 0 0
##   MELANOMA    1 0 0 0 7
##   NSCLC       2 0 1 0 6
##   OVARIAN     0 0 0 0 6
##   PROSTATE    0 0 0 0 2
##   RENAL       0 0 0 0 9
##   UNKNOWN     0 0 0 0 1
```

For $K = 2$, it looks like a lot of cancers can be put into a certain cluster. For $K = 3, 4, 5$, we see Renal, colon, and NSCLC very surely in one cluster, and some of the other cancers spread across 2-3 clusters.