Data 3 HW 7

Sean Duan

12/9/2020

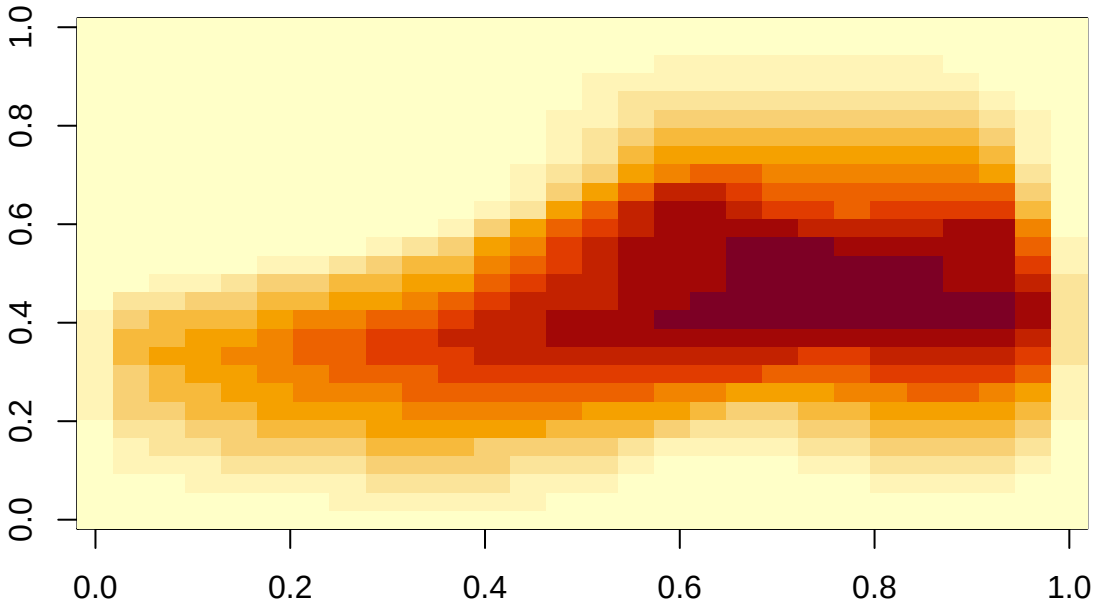
1.

Mean Image Plot

```
#average across all the rows for all the columns, then you have a vector of 784 elements, then put it i
s1<-colMeans(shoes)
s1<-as.matrix(s1)
s1<-t(s1)
class(s1)
```

```
## [1] "matrix" "array"
```

```
image(rotate(matrix(s1[1,], nrow=sqrt(ncol(s1)))))
```



The image of a 'mean' shoe looks very shoe-like, as expected! We can clearly see the tip of the shoe, the arch of the foot, and the hook in the heel. Additionally, the clear density of the image shows which relative

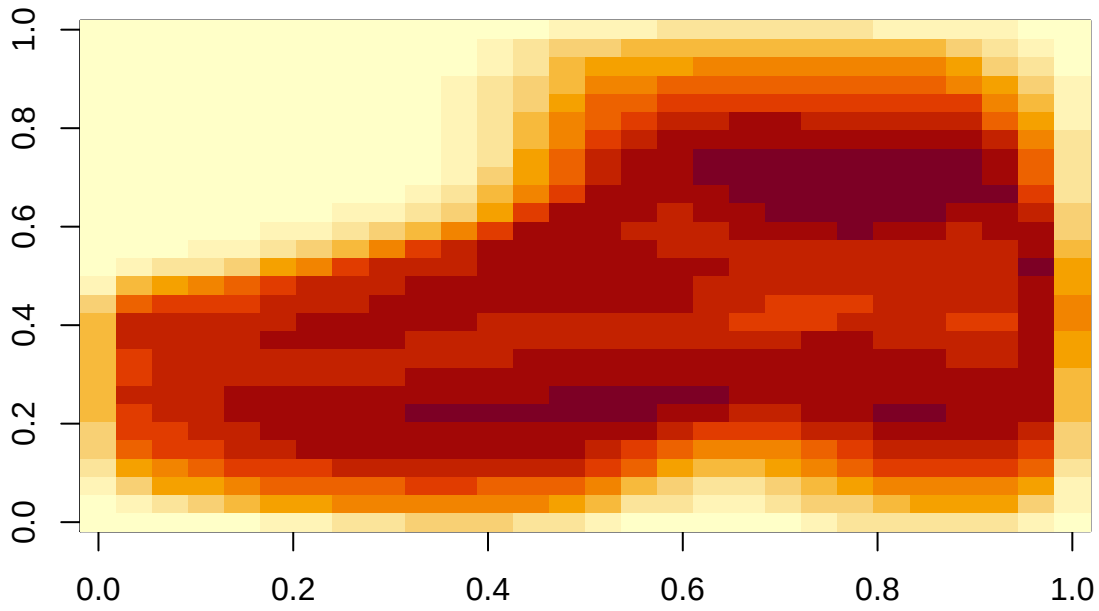
importance elements are integral to being a shoe.

Standard Deviation Image Plot

```
#do it for sdev as well
s2<-apply(shoes, 2, sd)
s2<-as.matrix(s2)
s2<-t(s2)
class(s2)
```

```
## [1] "matrix" "array"
```

```
image(rotate(matrix(s2[1,], nrow=sqrt(ncol(s2)))))
```



Conversely, in our image for the 'standard deviation' of a shoe, we can see a fair amount of differences. Unsurprisingly, areas in the upper left hand corner have very low deviation, as well as the edges of the image as a whole. Thus, none of our shoes likely have presence in that area. However, we see areas of increased density near the ankle, as well as the bottom of the arch. This indicates that when shoes do differ, they likely differ at those two places.

2

A

```
#subtract mean value
shoes_1<-sweep(shoes, 2, s1)
```

```
#doing PCA here w/ scaling set to true
pca_shoes=prcomp(shoes_1)
#A
#the variance explained by each PC is explained by squaring these
pr.var<-((pca_shoes$sdev)^2)
pve<-pr.var/sum(pr.var)
head(pve)
```

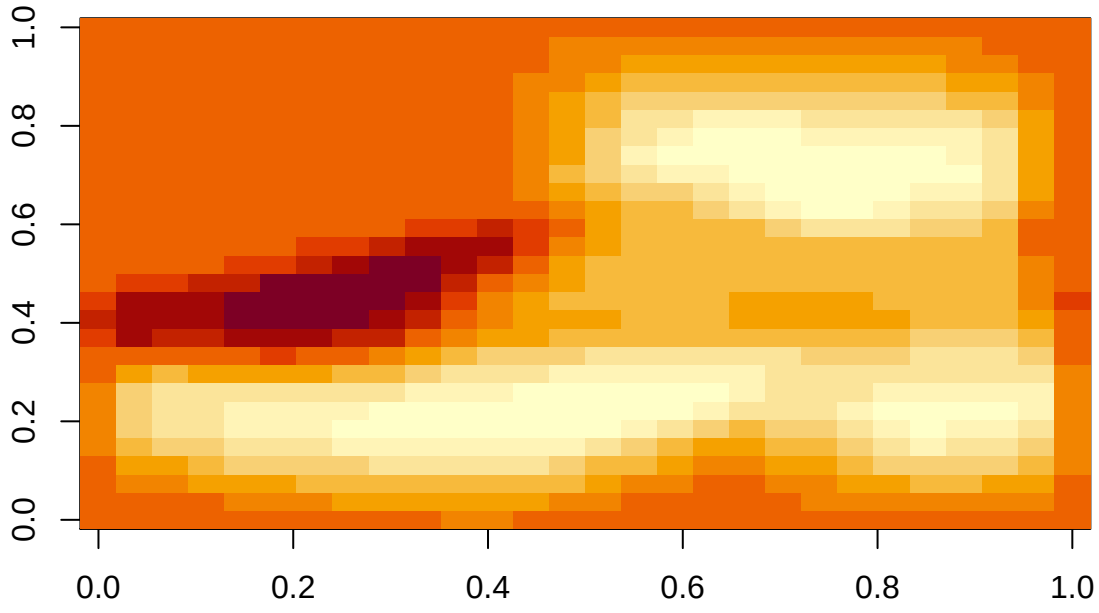
```
## [1] 0.32930935 0.12949081 0.07247230 0.03057633 0.02466056 0.02013794
```

```
#32%, 13%, 7%, 3%, and 2.5%
```

Our first five principle components account for 33%, 13%, 7%, 3%, and 2.5% of our variance, accordingly.

B - PC1

```
pc_loadings<-pca_shoes$rotation
pc_loadings<-as.data.frame(pc_loadings)
pc1<-pc_loadings$PC1
pc1<-as.matrix(pc1)
pc1<-t(pc1)
image(rotate(matrix(pc1[1,], nrow=sqrt(ncol(pc1)))))
```



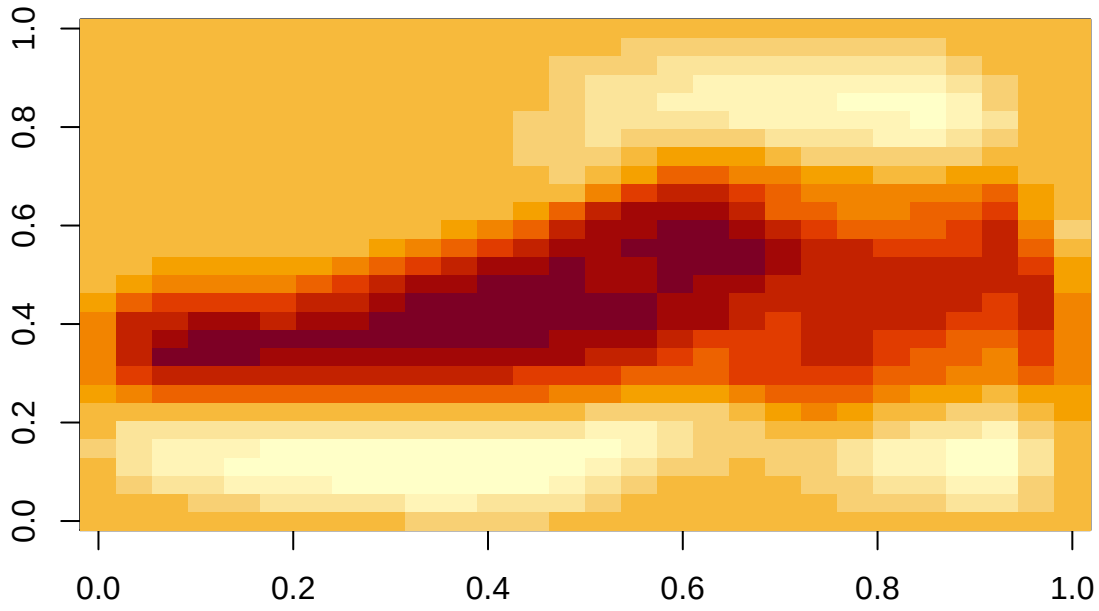
Our first pc seems to account for shoes that fall within a standard mens loafer design, wih a simple midsole arch, a firm heel extensor, and relatively little ankle support.

B - PC2

```
pc2<-pc_loadings$PC2
pc2<-as.matrix(pc2)
pc2<-t(pc2)
class(pc2)
```

```
## [1] "matrix" "array"
```

```
image(rotate(matrix(pc2[1,], nrow=sqrt(ncol(pc2)))))
```

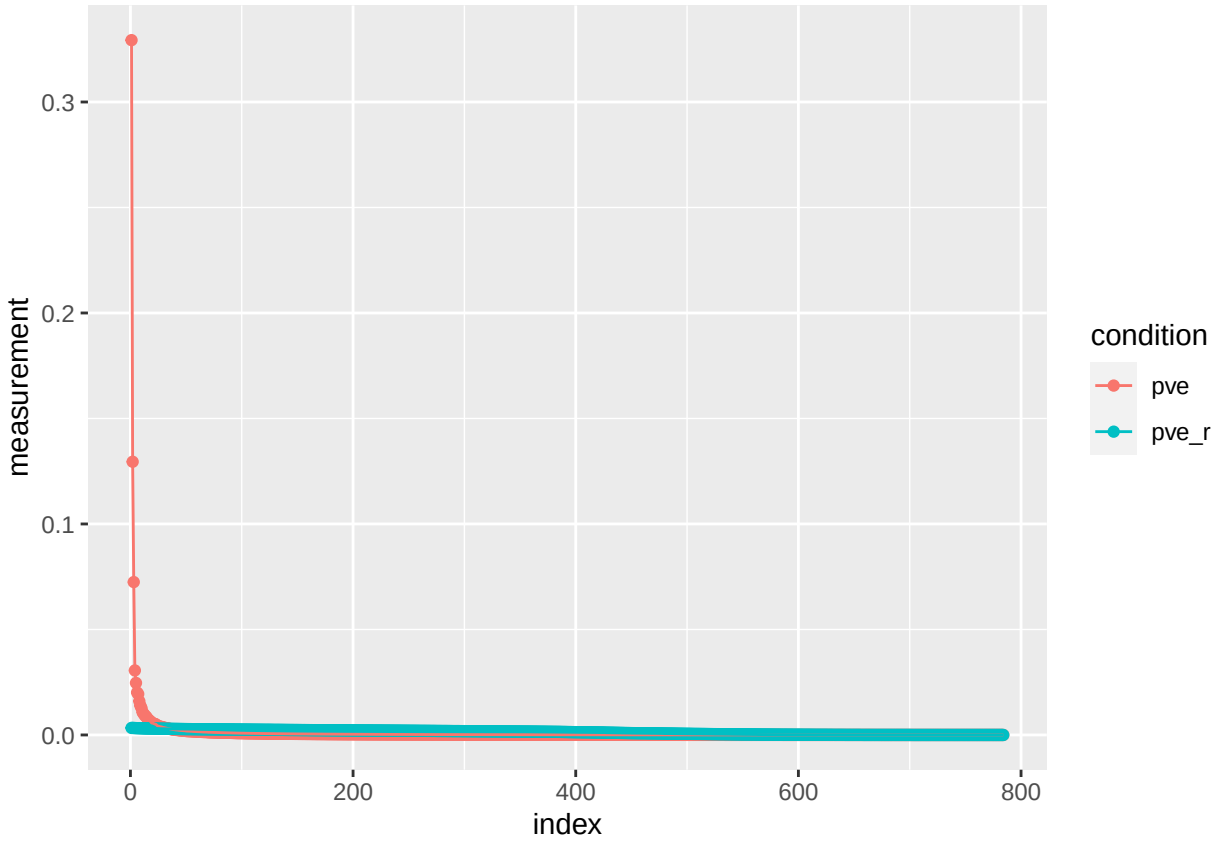


Our second PC seems to account for shoes that are more athletic in origin, with a slimmer profile, more ankle support, deeper midsole arch, and a larger heel extension.

C

```
shoes_2<-as.data.frame(shoes_1)

z<-as.data.frame(lapply(shoes_2, sample))
#this works and samples w/o replacement <3
#trying to generate our PCA on our randomized col dataset and calc the PCA effect
pca_shoes_random=prcomp(z)

pr.var_r<-((pca_shoes_random$sdev)^2)
pve_r<-pr.var_r/sum(pr.var_r)

pca_plot_dat<-cbind(pve,pve_r)
```

```
pca_plot_dat<-as.data.frame(pca_plot_dat)
pca_plot_dat$index<-seq(1:nrow(pca_plot_dat))

pca_plot_dat_wide<-gather(pca_plot_dat, condition, measurement, pve:pve_r, factor_key = TRUE)

pca_plot<-ggplot(pca_plot_dat_wide, aes(x=index, y=measurement, group = condition, color = condition))+
pca_plot
```

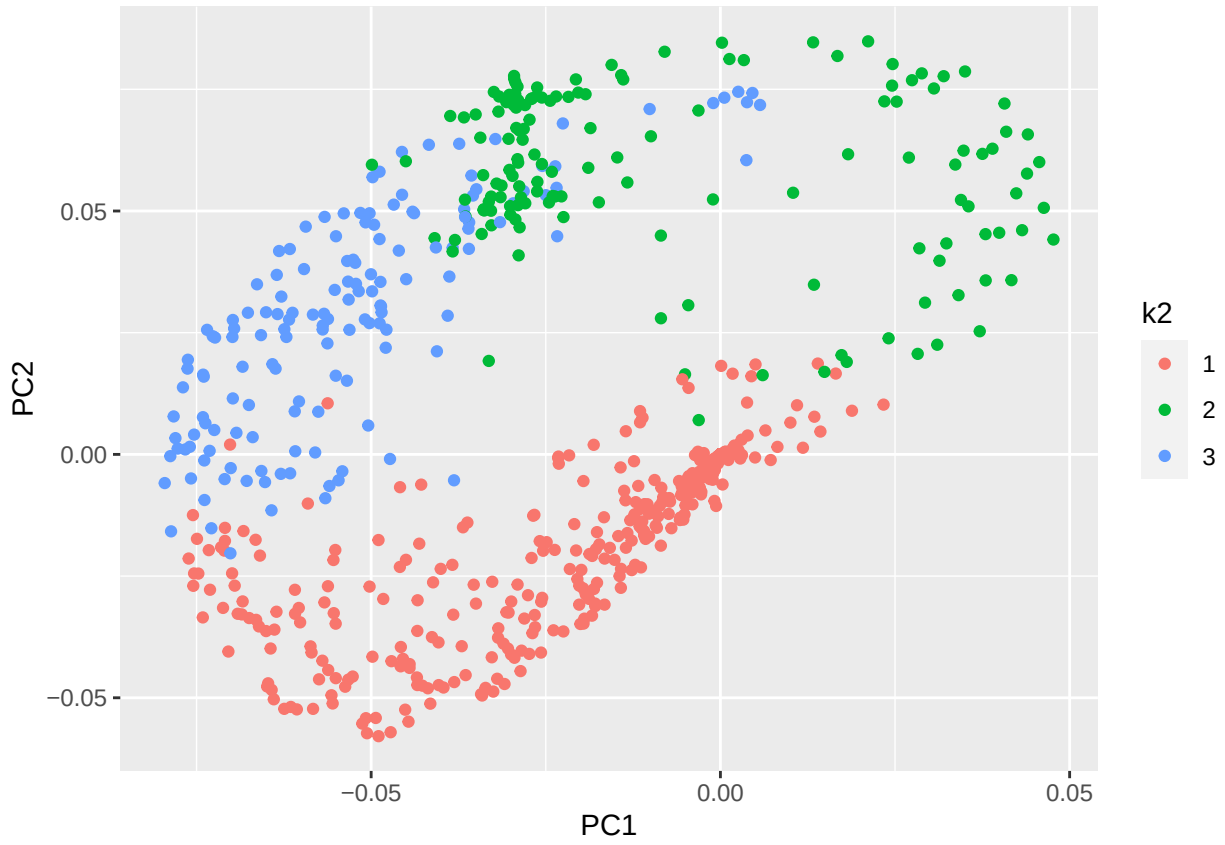


We can clearly see when plotting randomized PC's against the PC's that actually originate from our data that the 'crossover' point is roughly at 35 principle components. Thus, our first 34 principle components are superior to random noise.

3

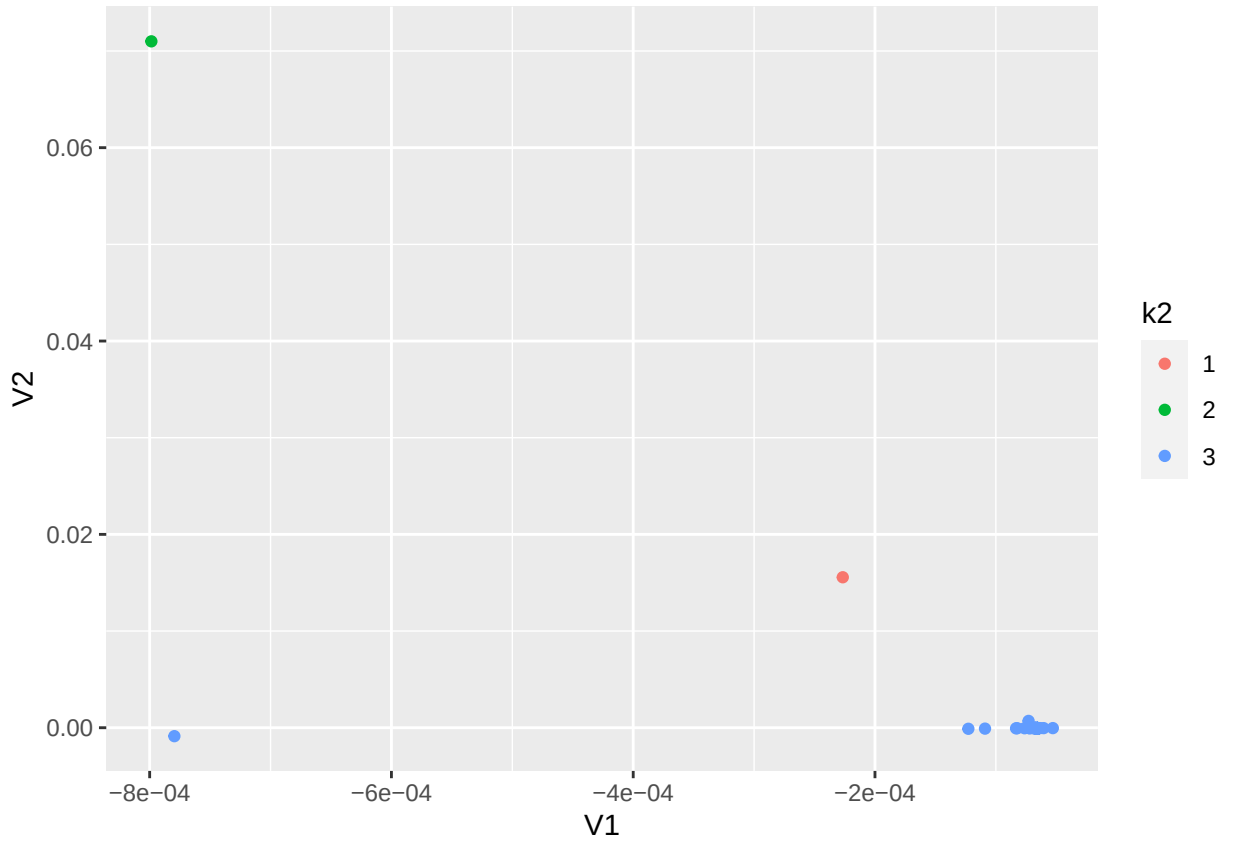
A

```
kmean_loading<-pc_loadings[,1:4]
km.out=kmeans(kmean_loading , 3, nstart =20)
k2<-as.factor(km.out$cluster)
km_plot<-ggplot(kmean_loading, aes(x=PC1, y=PC2, color = k2))+geom_point()
km_plot
```



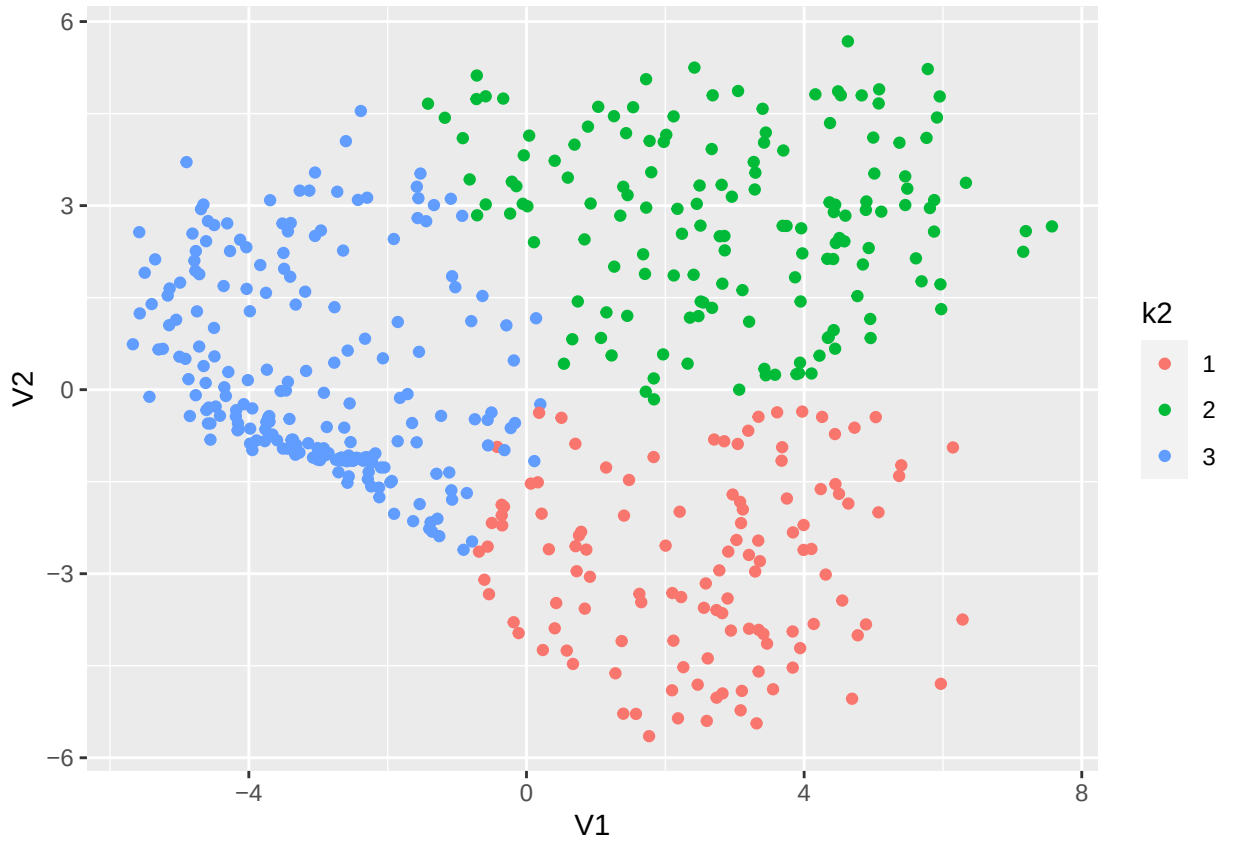
4

```
#do data preprocessing (split train/test, and feature scaling)
shoes_4<-scale(shoes)
shoes_4[is.na(shoes_4)] <- 0

#really reduce the # size so we have a smaller set!


min=sample(1:nrow(shoes_4), 1000)
shoes_5<-shoes_4[min,]

shoes_5<-as.data.frame(shoes_5)
train=sample(1:nrow(shoes_5), 500)
shoes_k = kpca(~., data = shoes_5[-train,], kernel = 'rbfdot', features = 4)

training_set_pca = as.data.frame(predict(shoes_k, shoes_5[train,]))
km.out=kmeans(training_set_pca , 3, nstart =20)
k2<-as.factor(km.out$cluster)
km_plot<-ggplot(training_set_pca, aes(x=V1, y=V2, color = k2))+geom_point()
km_plot
```



Using the rbfdot kernal, we see that there is very strong separation between our three categories on our first two kernal principle components.

```
shoes_k = kpca(~., data = shoes_5[-train,], kernel = 'laplacedot', features = 4)
training_set_pca = as.data.frame(predict(shoes_k, shoes_5[train,]))
#scatter plots
km.out=kmeans(training_set_pca , 3, nstart =20)
k2<-as.factor(km.out$cluster)
km_plot<-ggplot(training_set_pca, aes(x=V1, y=V2, color = k2))+geom_point()
km_plot
```

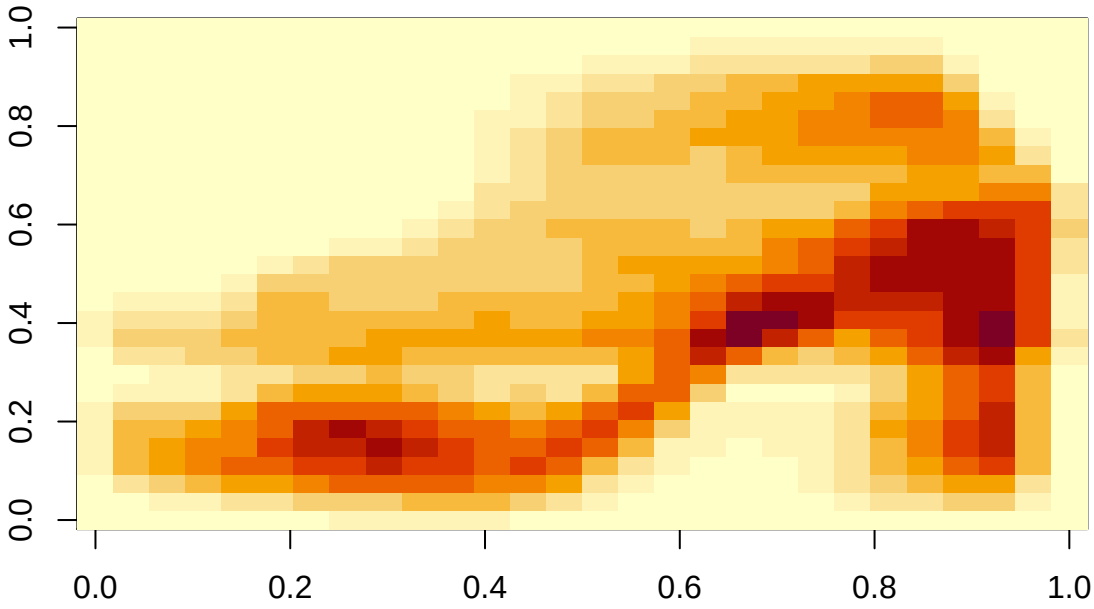


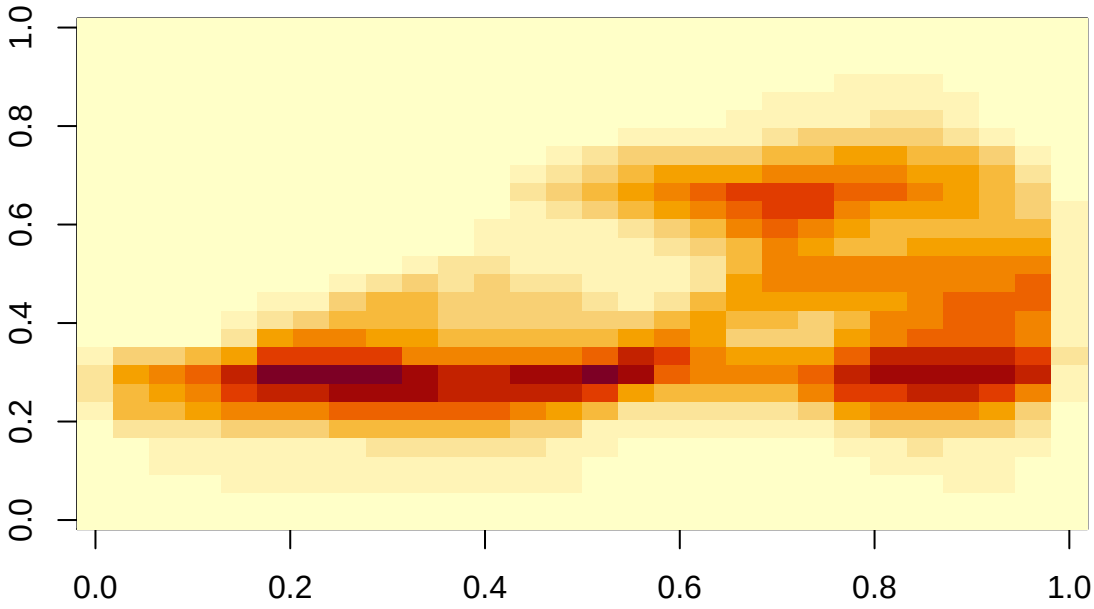
Using the laplacedot kernal, we see good, but less perfect, separation between our three categories on our first two kernal principle components.

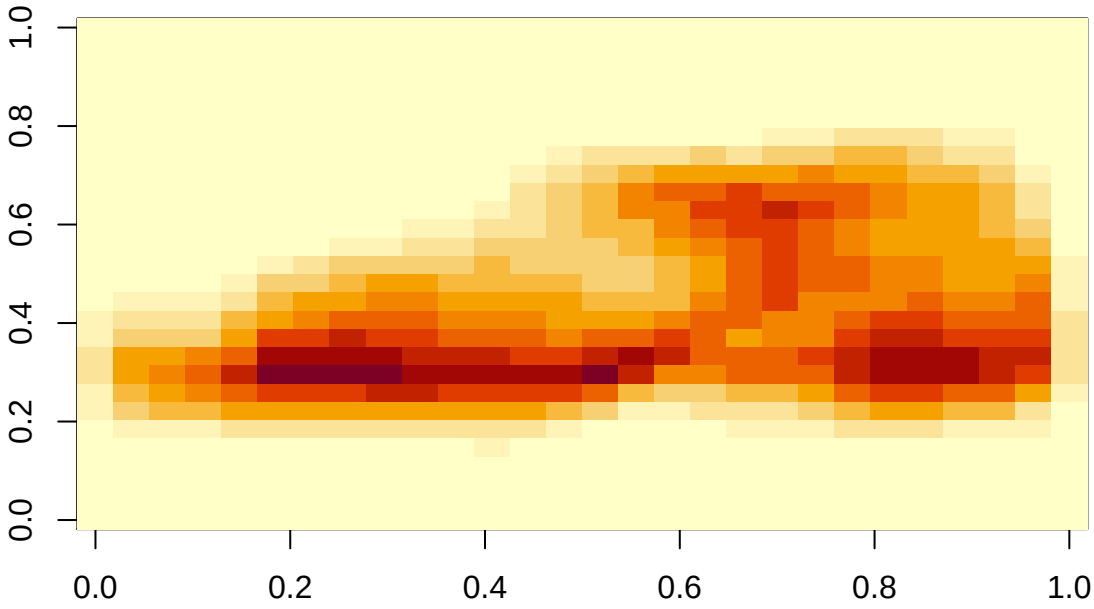
I was not able to get any of the other kernals to work using the kpca package. It seemed like there were not specified parameters that were missing, that I was not able to specify.

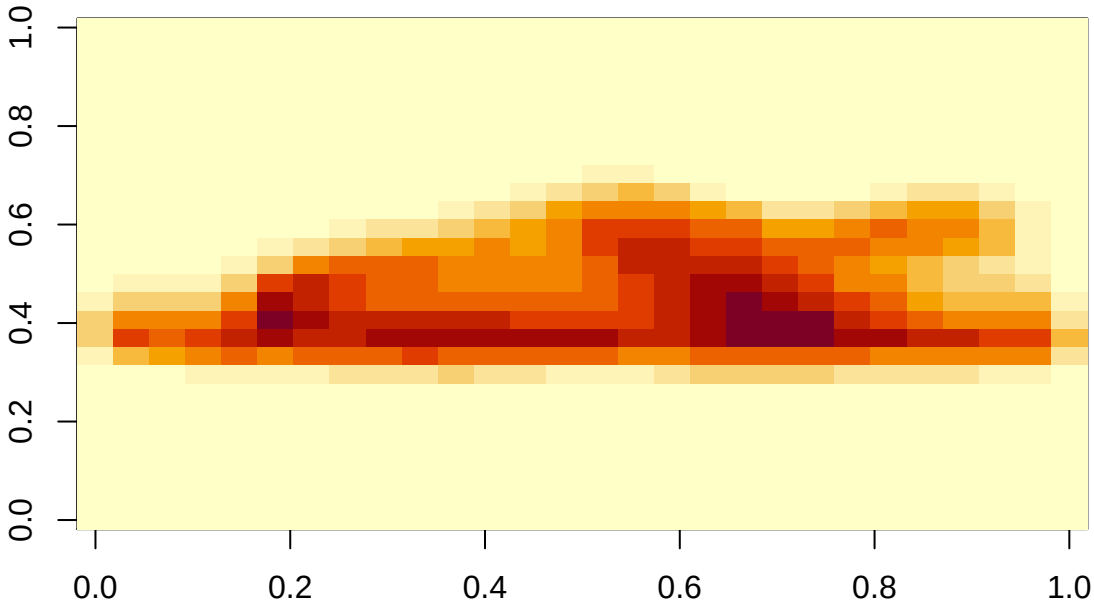
5

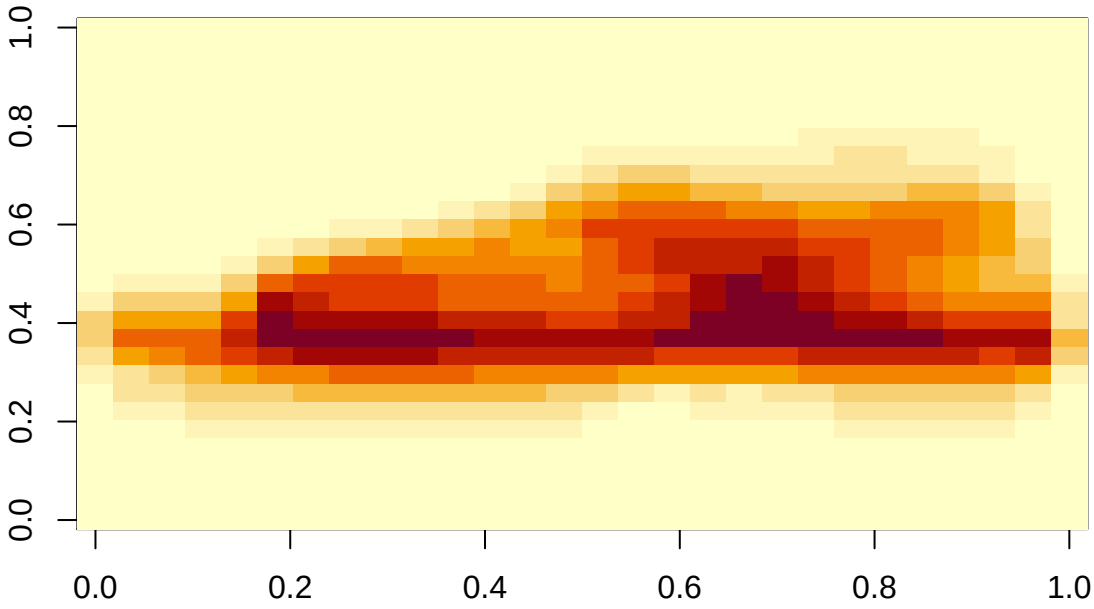
```
#performing a 'single run' of the NMF algo
res <- nmf(shoes[train,], 3, method = 'lee')

#fitted(res) pulls out or fitted matrice, the command below autoplots our first 6 images
V.hat <- fitted(res)
for (i in 1:6) {
  image(rotate(matrix(V.hat[i,], nrow=sqrt(ncol(V.hat)))))
}
```

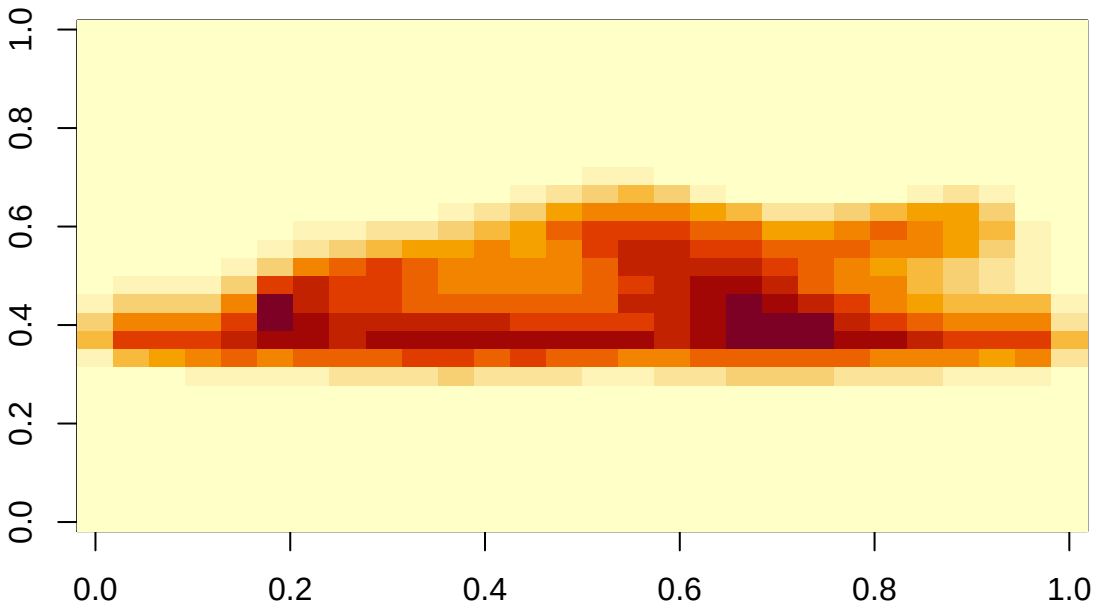

0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0



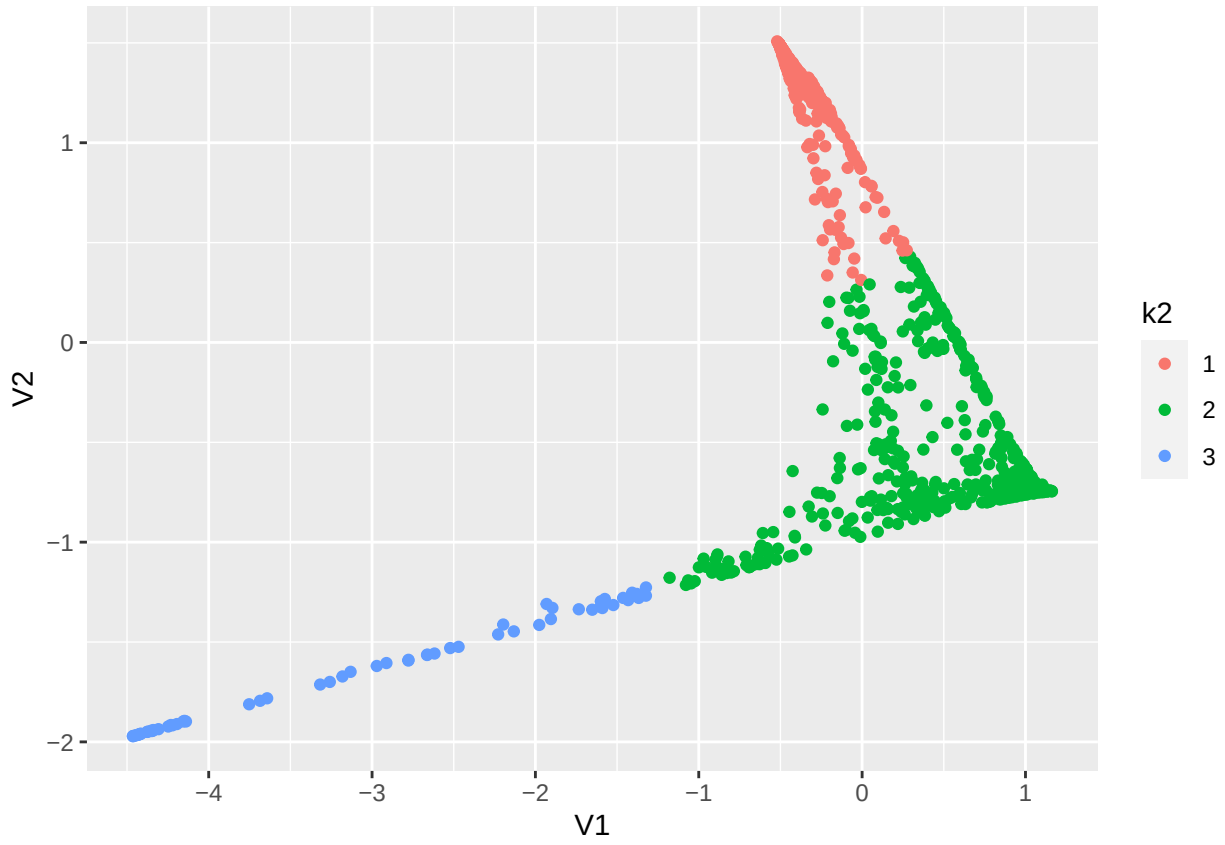
Compared to the matrices created for PCA, we see that our first two matrices seemingly account for female shoes, which our PCA matrices did not account for (at least the first two of them).

6

```
k40 <- lle(X=shoes_5, m=2, k=5, reg=2, ss=FALSE, id=TRUE, v=0.9 )
```

```
## finding neighbours
## calculating weights
## intrinsic dim: mean=3.744, mode=4
## computing coordinates
```

```
lle_pca<-k40$Y
lle_pca<-as.data.frame(lle_pca)
#scatter plots
km.out=kmeans(lle_pca , 3, nstart =20)
k2<-as.factor(km.out$cluster)
km_plot<-ggplot(lle_pca, aes(x=V1, y=V2, color = k2))+geom_point()
km_plot
```


V2
V1
1
0
−1
−2
−4
−3
−2
−1
0
1
k2
1
2
3