

Käsitsi numbrid

Risto Hinno

Friday, June 26, 2015

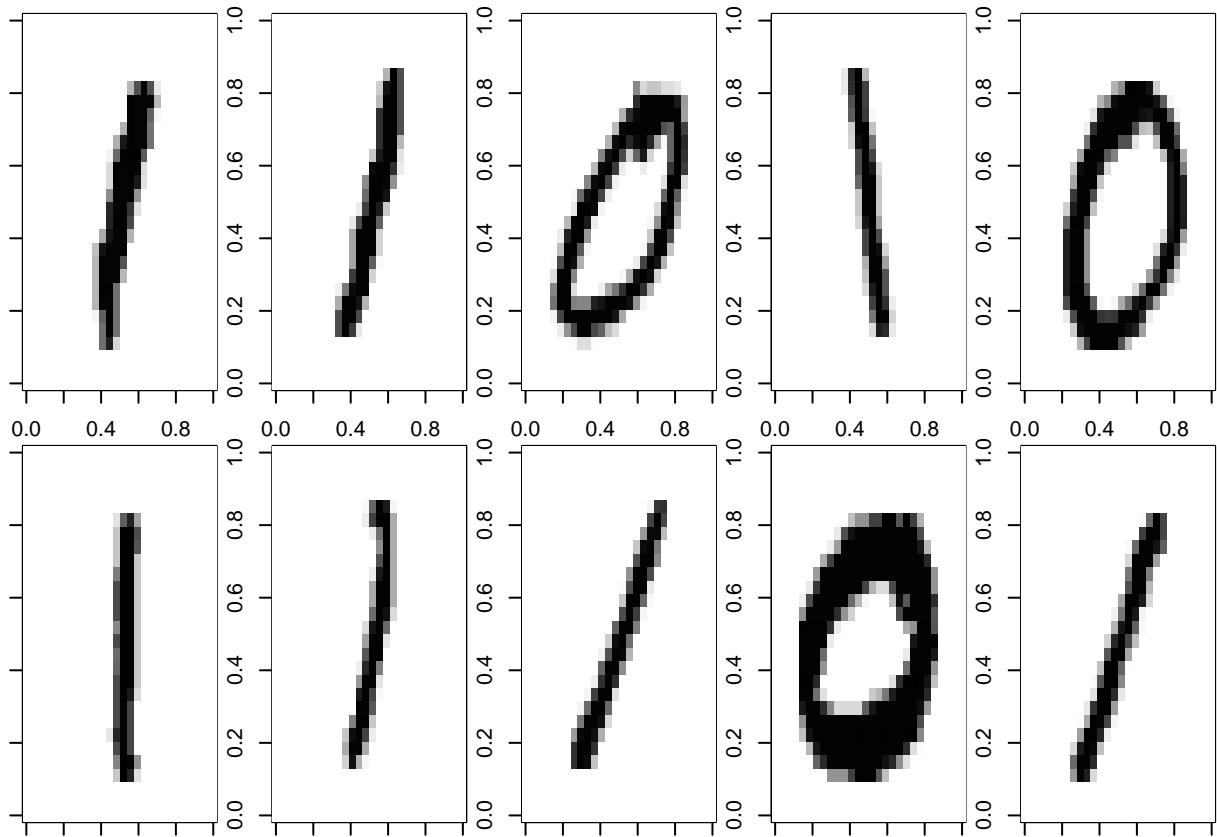
Ülesanne 1 (2 punkti) - andmestikuga tutvumine

Visualiseeri näiteid nii nullide kui ka ühtede seast.

Näpunäited:

- Abiks on ette antud funktsioon plot_digit, mille argumendiks sobib andmestiku üks rida (NB! ilma viimase veeruta)
- Alamjooniste tegemisel on kasuks käsk par(mfrow = c(mitu_riba, mitu_veergu))
- Ääriseid saad muuta par(mar = c(bottom, left, top, right))

```
#loeme andmed sisse
numbrid=read.csv("./data/numbrid.csv")
#funktsioon visualiseerimiseks
plot_digit = function(digit, ...){
  cols = grey(seq(1, 0, length = 256))
  image(t(matrix(as.numeric(digit), nrow=28, ncol=28)[28:1, ]), col = cols, ...)
}
#määrame mitu joonist ühele ekraanile tahame, ja margini
par(mfrow=c(2, 5), mar=c(1, 1, 1, 1) + 0.1)
#plotime 10 plotti
for (i in 1:10) {
  plot_digit(numbrid[i,])
}
```



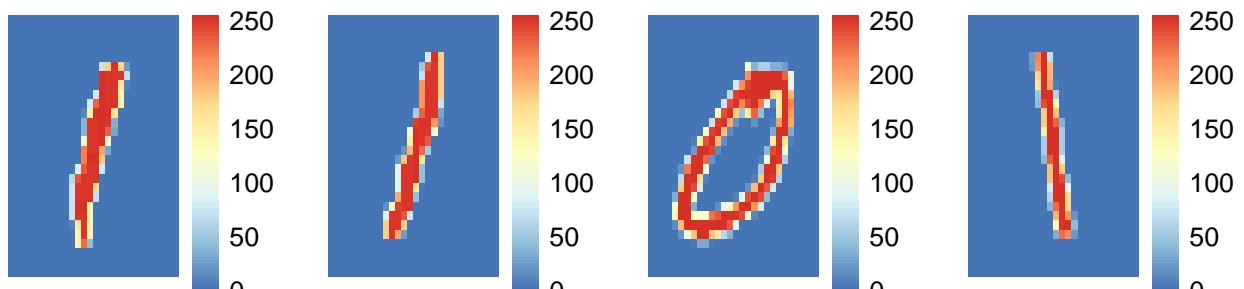
```
#taastame algseid parameetrid
par(mfrow=c(1, 1), mar=c(5, 4, 4, 2) + 0.1)
```

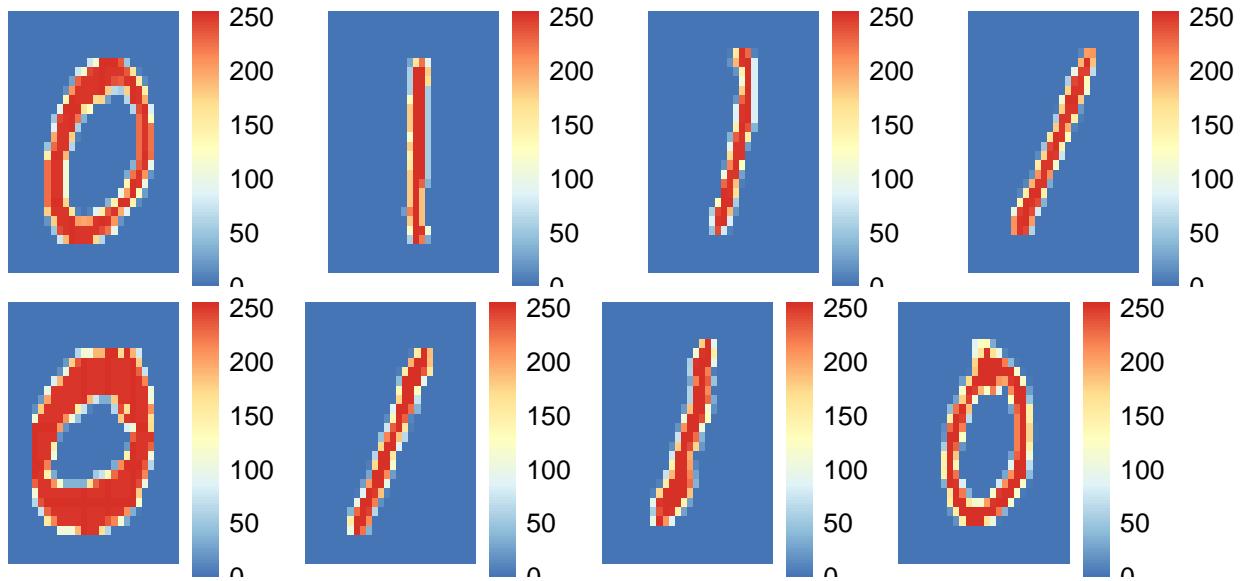
Teine võimalus visualiseerimiseks:

```
#kasutame pheatmapi põhjal olevat funktsioon
library(pheatmap)

plot_digit_pheatmap = function(digit){
  mat = matrix(as.numeric(digit), nrow=28, ncol=28)
  pheatmap(mat, cluster_cols=FALSE, cluster_rows=FALSE)
}

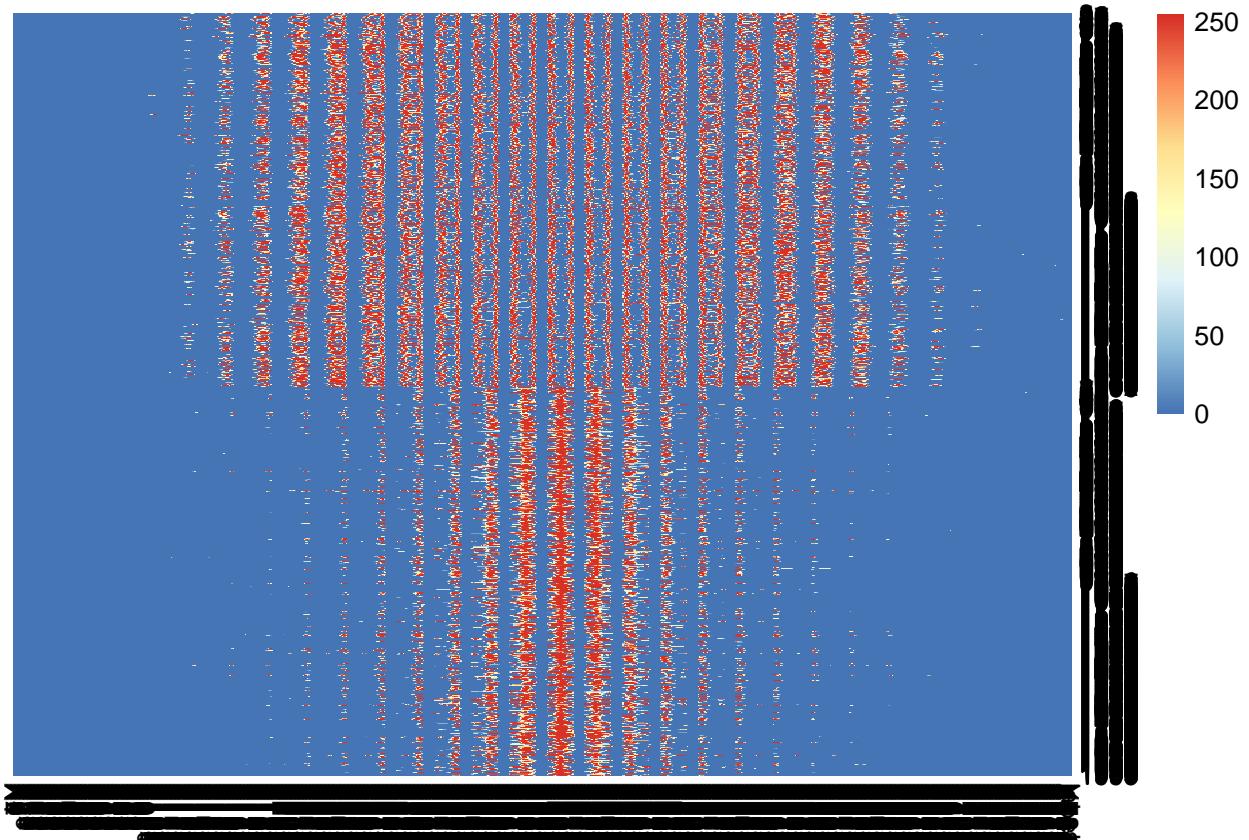
#plotime, siin ei saa mitut plotti ühele ekraanile
for (i in 1:12) {
  plot_digit_pheatmap(numbrid[i,])
}
```





Sorteeri andmestiku read selliselt, et üleval oleksid nullid ja all ühed. Visualiseeri kogu andmestikku kasutadaes pheatmap funktsooni. Kasuta argumente cluster_rows=FALSE, cluster_cols=FALSE.

```
numbrid_sort <- numbrid[order(numbrid$label),]
pheatmap(numbrid_sort, cluster_rows=FALSE, cluster_cols=FALSE)
```



- Küsimus: Miks on punased triibud vaheldumisi sinistega?

- Vastus: Kuna iga rida on ühe numbri pikslid (maatriks 28x28), siis nn pildi reaks jagamisel jäääb iga numbriga ühe tühi ala, mis on sinine. Kuna pilt on “viilutatud” reaks, siis hakkab mustei korduma igas viilus (iga 28 piksli järel).

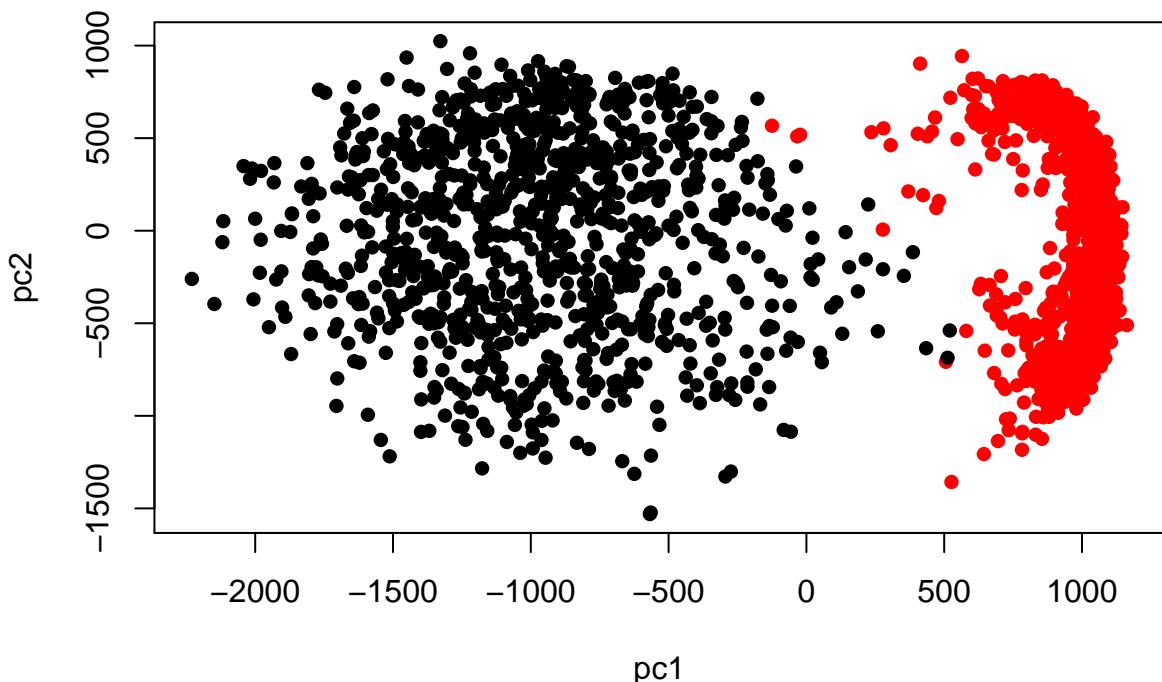
Ülesanne 2 (4 punkti)

Tee andmestikul PCA (kontrolli, et oled eelnevalt andmestikust eemaldanud tunnuse label). PCA tegemiseks kasuta funktsiooni prcomp. Milline on sinu arvates andmestiku “efektiivne dimensionaalsus” praegusel juhul?

```
pca = prcomp(numbrid[, -785])
#koodi ei joooksuta, kuid sellega saab vaadata faktorite olulisust
#summary(pca)
```

Tee hajuvusdiagramm PC1 vs PC2. Seejärel märgi joonisele, millised punktid kujutavad numbrit 0 ja millised numbrit 1 (võid kasutada värvit või argumenti pch).

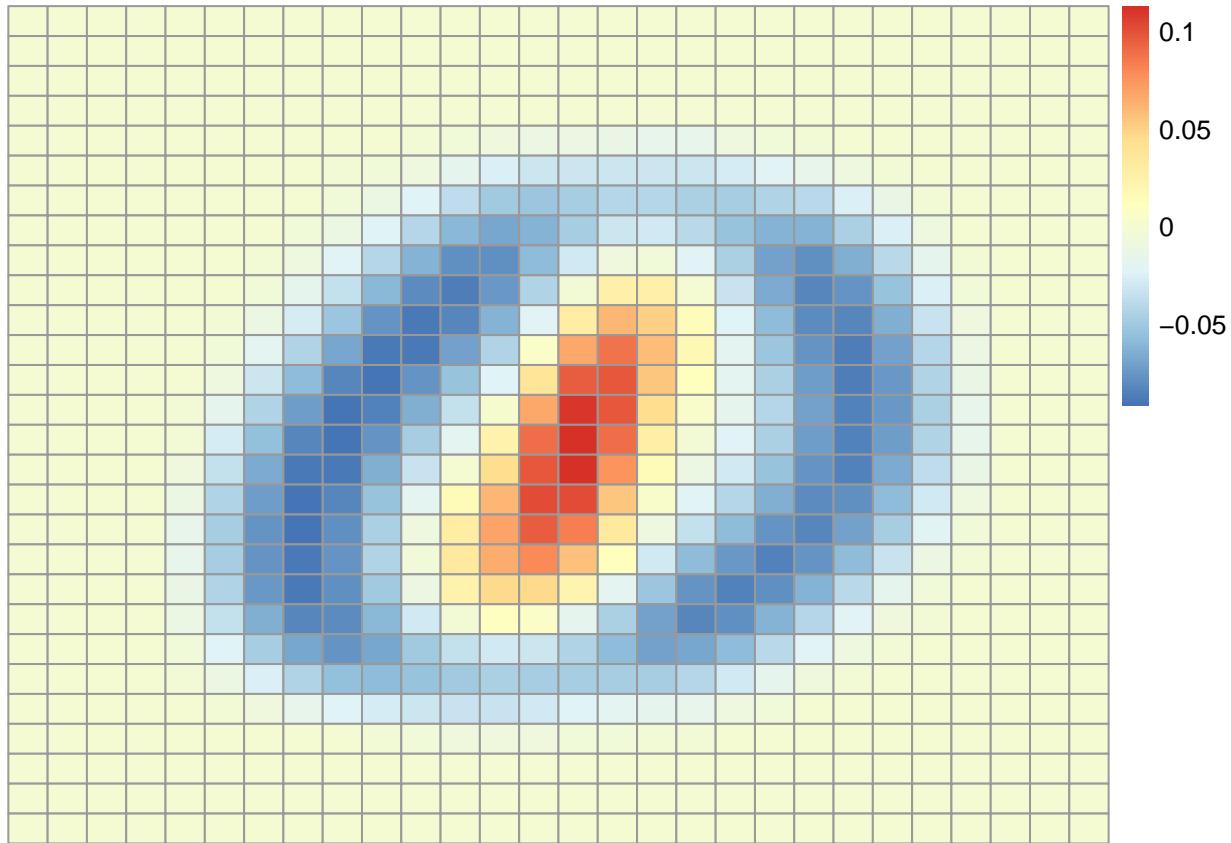
```
pc1 = pca$x[, 1]
pc2 = pca$x[, 2]
plot(pc1, pc2, pch=16, col=as.factor(numbrid$label))
```



- Küsimus: Mida võiks selle joonise põhjal tähistada PC1?
- Vastus: näidata, kas tegu on numbriga 1 või 0

Visualiseeri PCA kaalusid.

```
pca_rotate=pca$rotation  
plot_digit_pheatmap(pca_rotate[,1])
```

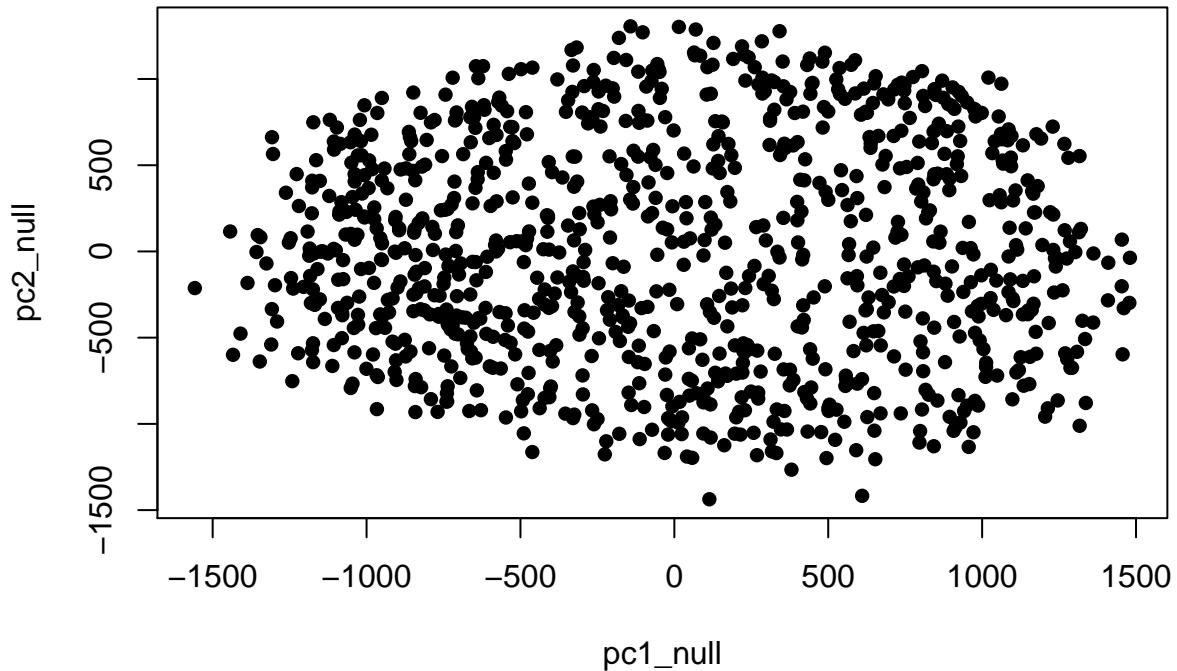


- Küsimus: Millistel pikslitel on absoluutväärtuselt suured kaalud? Interpretteeri selle abil PC1 tähendust (milliste pikslite intensiivsus peab olema suur ja milliste pikslite intensiivsus madal, et PC1 väärtus oleks võimalikult suur).
- Vastus: keskmistel pikslitel. Keskmiste oma suur (need kus on nr 1) ja madal keskohta ümbritsevate pikslite juures (kus on nr 0).

Ülesanne 3 (4 punkti)

Tee nüüd PCA andmestikul, mis koosneb ainult nullidest. Lisaks tee hajuvusdiagramm PC1 vs PC2.

```
numbrid_null=subset(numbrid, label==0)  
pca_null = prcomp(numbrid_null[, -785])  
pc1_null = pca_null$x[, 1]  
pc2_null = pca_null$x[, 2]  
plot(pc1_null, pc2_null, pch=16, col=as.factor(numbrid_null$label))
```



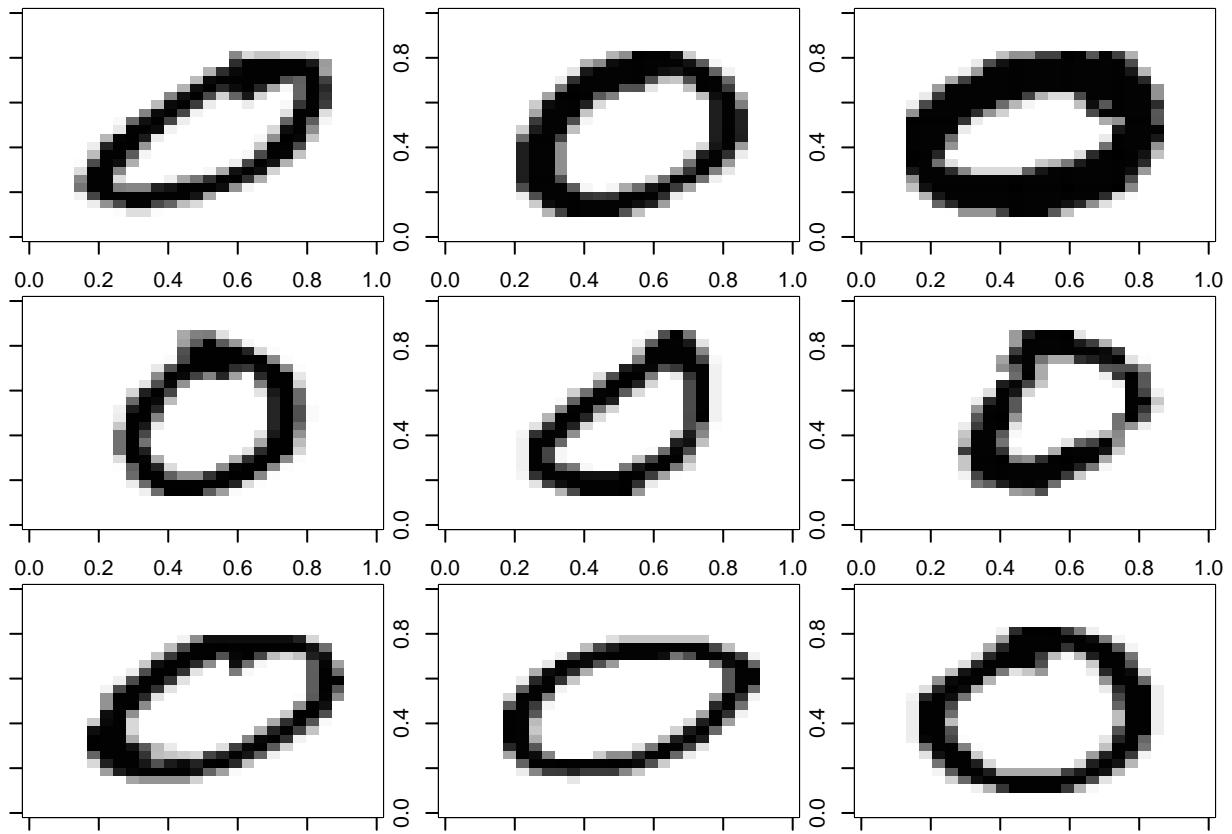
- Küsimus: Kas oskad selle joonise põhjal tõlgendada peakomponente PC1 ja PC2?
- vastus: ei

Vali üheksa numbrit tasandi erinevatest nurkadest ja visualiseeri neid funktsiooni plot_digit abil.

```
#numbrite leidmiseks kasutasin eelmise graafiku peal seda funktsiooni ja klikkisin
#punktidele
#identify(pc1_null, pc2_null, n=9)
#numbrid mis märkisin:
arvud=c(353, 452, 522, 588, 653, 782, 806, 866, 931)

#plotime
par(mfrow=c(3, 3), mar=c(1, 1, 1, 1) + 0.1)

for (i in 1:length(arvud)) {
  plot_digit(numbrid_null[i,])
}
```



```
# Pärast taasta esialgsed graafilised parameetrid
par(mfrow=c(1, 1), mar=c(5, 4, 4, 2) + 0.1)
```

- Küsimus: Kuidas tölgendad selle joonise põhjal peakomponente PC1 ja PC2?
- Vastus: pärius hästi veel öelda ei oska, kuid tundub, et see jagab kuidagi nulle kuju ja paksuse järgi klastritesse.

Punktikeste asemel visualiseeri numbreid.

```
#kasutan etteantud funktsiooni
plot_digit = function(digit, x=NA, y=NA, scale=1, add=FALSE, transparency=FALSE, ...){
  if(is.na(x)){
    x = 0
  }
  if(is.na(y)){
    y = 0
  }

  x_id = seq(x, x + scale, length=28)
  y_id = seq(y, y + scale, length=28)

  if(transparency==TRUE){
    tmp = as.character(round(seq(0, 0.99, length=256)*100))
    tmp[nchar(tmp) == 1] = paste("0", tmp[nchar(tmp) == 1], sep="")
  }
}
```

```

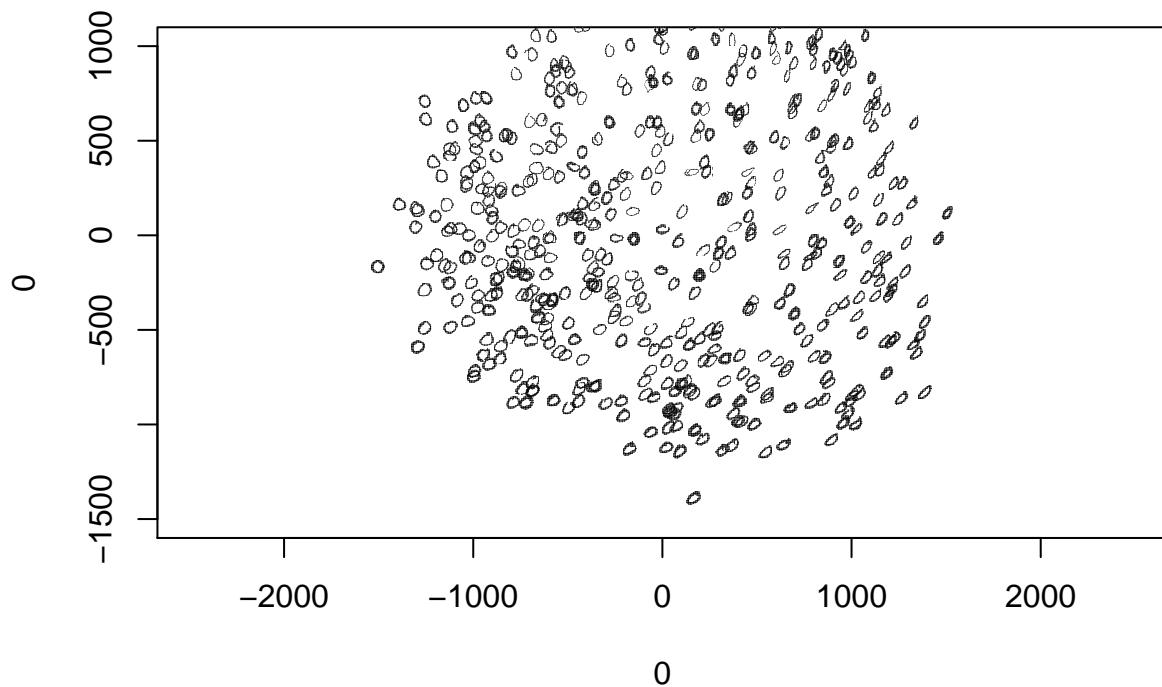
        cols = colorRampPalette(c("#FFFFFF", "#000000"))(256)
        cols = paste(cols, tmp, sep="")
    }
else{
    cols = grey(seq(1, 0, length = 256))
}

image(x_id, y_id, t(matrix(as.numeric(digit), nrow=28, ncol=28)[28:1, ]),
      col = cols, axes=F, asp=1, add=add, ...)
}

#plotime nullid
plot(0, 0, type = "n", xlim=c(-1500, 1500), ylim=c(-1500, 1000), asp=1)
#plotin 500, pole rohkem vaja
for(i in 1:500){
    x = pc1_null[i]
    y = pc2_null[i]

    plot_digit(numbrid_null[i, -785], x, y, scale=100, add=TRUE, transparency = T)
}

```



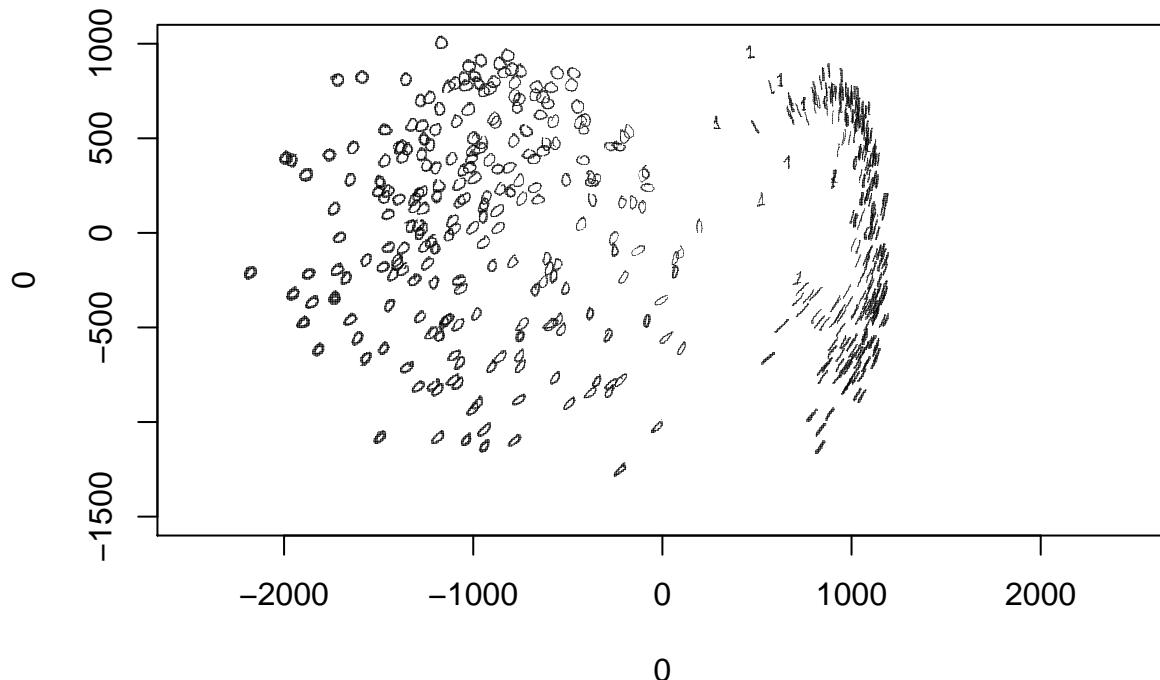
- Küsimus: Kuidas tölgendad selle joonise põhjal peakomponente PC1 ja PC2?
- Vastus: PC1 näitab, kas 0-d on ovaalsed või ringikujulised, PC2 näitab vist kui püstised 0-d on (või kas on diagonaalsed)

Tee samasugune joonis nagu eelmises punktis esialgse nullide ja ühtede andmestiku peal.

```
plot(0, 0, type = "n", xlim=c(-1500, 1500), ylim=c(-1500, 1000), asp=1)

for(i in 1:500){
  x = pc1[i]
  y = pc2[i]

  plot_digit(numbrid[i, -785], x, y, scale=100, add=TRUE, transparency = T)
}
```

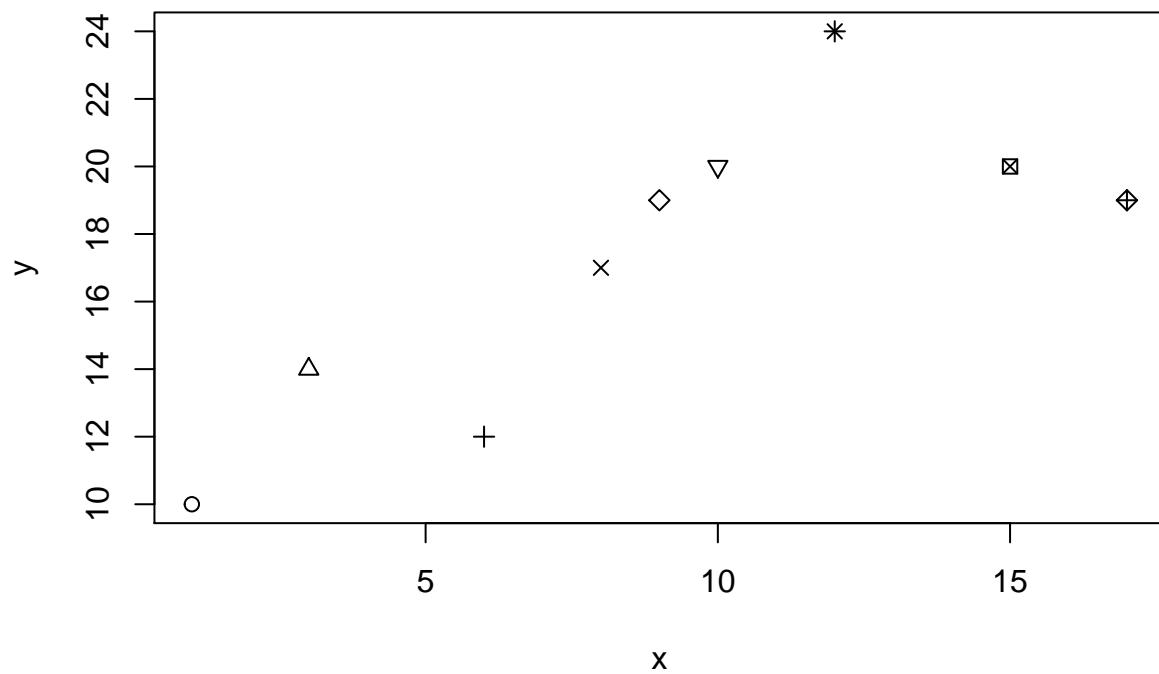


- Küsimus: Kuidas tölgendad selle joonise põhjal peakomponente PC1 ja PC2?
- Vastus: PC1 näitab, kas tegu on 0 või 1-ga, PC2, kas kriipsud on püsti või “viltu”

Boonusülesanne 1 (2 punkti) - implementeer PCA

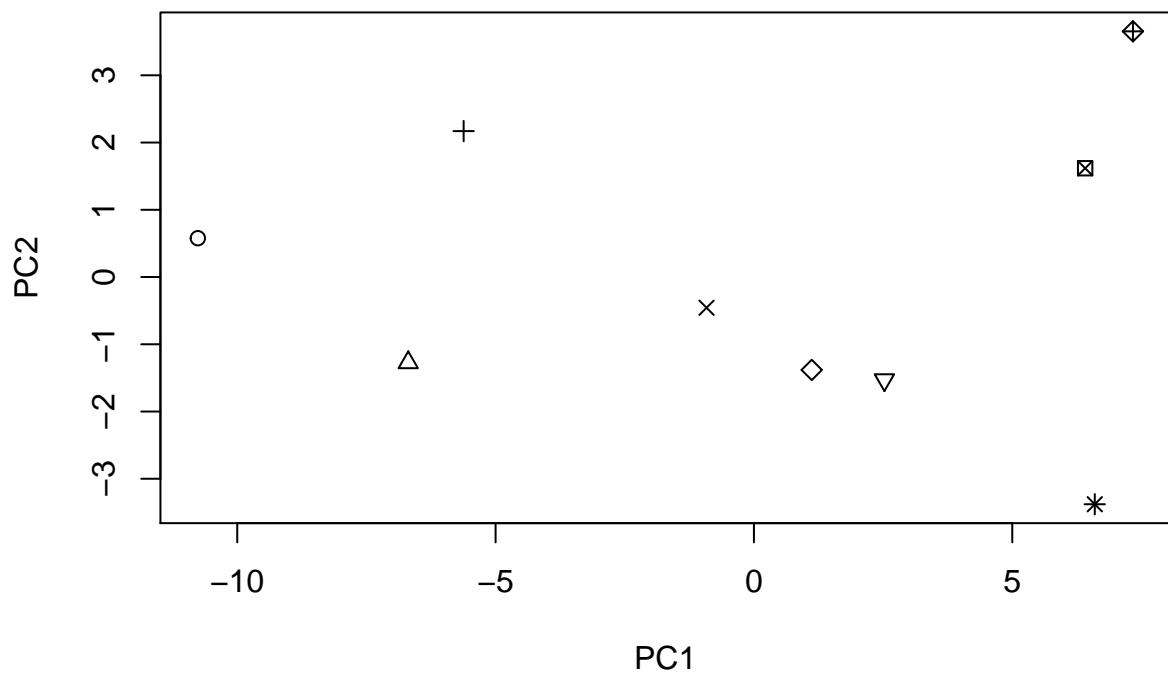
Selle ülesande eesmärk on implementeerida omaenda PCA.

```
#teeme pseudoandmed
x <- c(1,3,6,8,9,10,15,12,17)
y <- c(10,14,12,17,19,20,20,24,19)
m <- cbind(x, y)
plot(m, pch=seq(9))
```



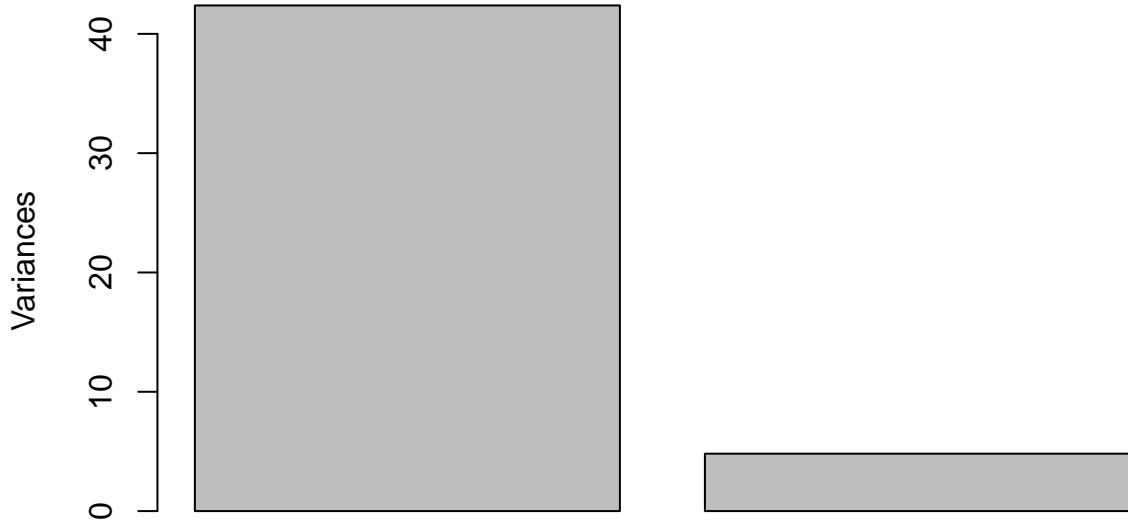
Alguses kasutame prcompi

```
pca_proc <- prcomp(m)
plot(pca_proc$x, pch=seq(9)) #the pca plot
```



```
plot(pca_proc) #the proportion of variance capture by each PC
```

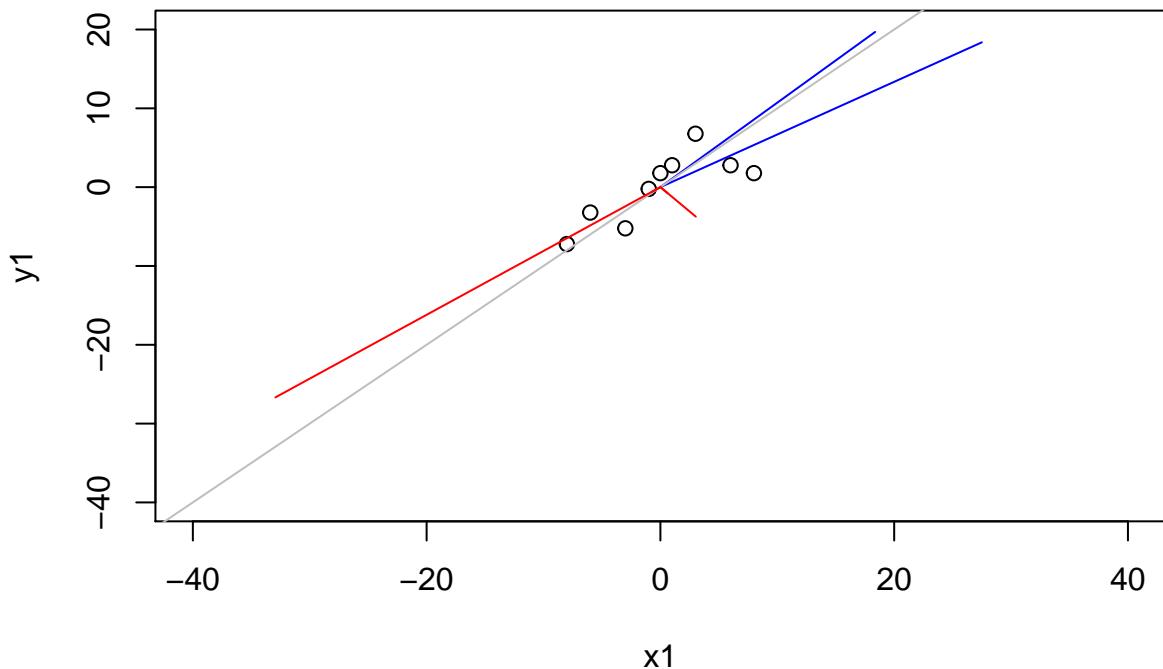
pca_proc



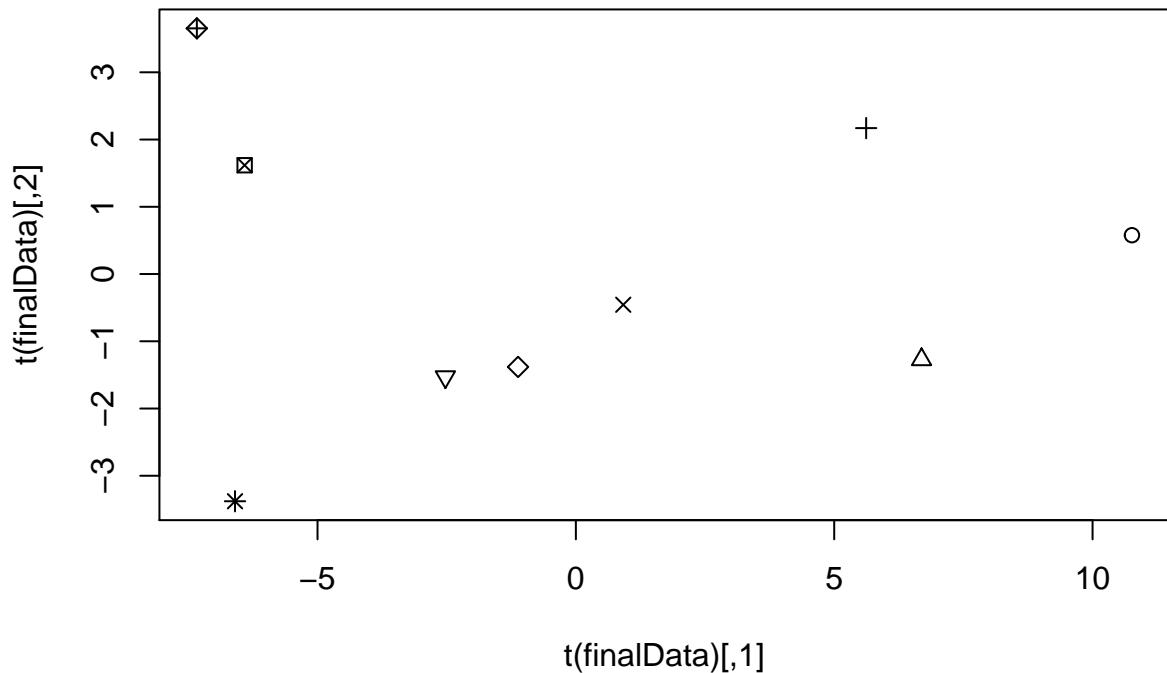
Et käsitsi teha, siis kige pealt arvutame iga näitaja keskmise ja lahutame igast vaatlusest näitaja keskmise maha (tsentreerime andmed 0 ümber). Seejärel arvutame kovariatsiooni maatrikis vektoritele x_1 ja y_1 ja plotime need. Need vektorid näitavad variatsiooni suunda andmetes (kui sarnane on $\text{cov}(x, x)$ $\text{cov}(x, y)$ -le ja kui sarnane on $\text{cov}(y, y)$ $\text{cov}(y, x)$ -le). X -i kovariatsioon iseendaga on võrdne variatsiooniga ($\text{var}(x) == \text{cov}(x, x)$). Kovariatsionimaatriksi kohta on infot [siin](#).

```
#tsentreerime
x1 <- x - mean(x)
y1 <- y - mean(y)
plot(x1, y1, xlim=c(-40, 40), ylim=c(-40, 20))
m1 <- cbind(x1, y1)
#kovariatsioon
cM <- cov(m1)
lines(x=c(0, cM[1,1]), y=c(0, cM[2,1]), col="blue")
lines(x=c(0, cM[1,2]), y=c(0, cM[2,2]), col="blue")
#cov(x, y) ei saa kunagi olla suurem kui cov(x, x), üks joontest graafikul on
#alati üleval pool diagonaali ning teine allpool. Plotime ka diagonaal (slope=1).
abline(0, 1, col="grey")
#kuigi plotitud vektorid näitavad variatsiooni suunda, ei tee nad seda väga kasulikult.
#Kovariatsionimaatriksi eigenvektorite leidmisenega saame me kirjeldada
#variatsiooni kahe ristuva ortogoonilise vektoriga, mis näitavad variatsiooni
#suunda (nende kahe joone asemel, mida algsest kasutame).
eigenVecs <- eigen(cM)$vectors
lines(x=c(0, eigenVecs[1,1]), y=c(0, eigenVecs[2,1]), col="red")
lines(x=c(0, eigenVecs[1,2]), y=c(0, eigenVecs[2,2]), col="red")
#kuna eigenvektorid on ühikvektorid (ehk pikkusega 1), on lihtsam, kui neid
```

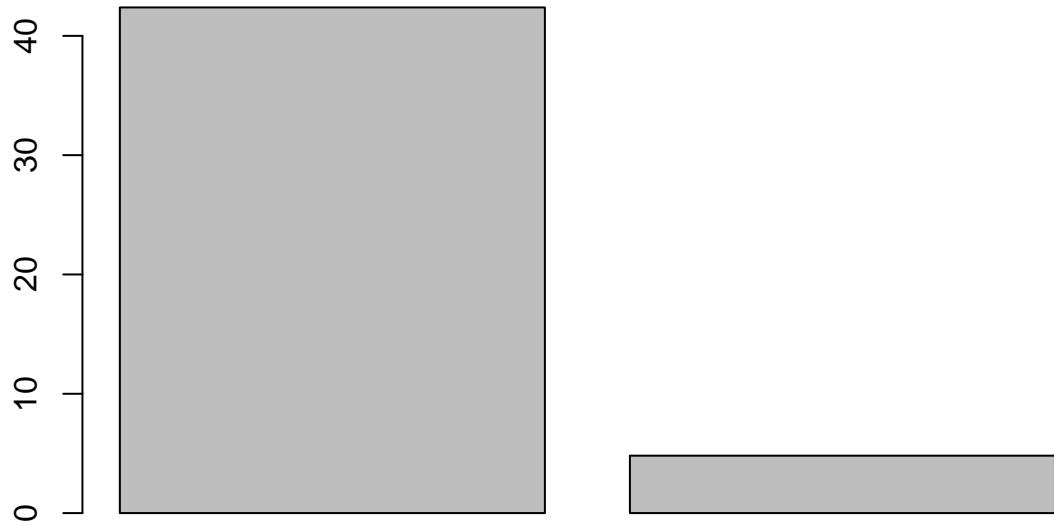
```
#visualiseerida, kui palju igaiüks neist mõjutab andmeid (korrutame nad vastava
#eigenväärtusega, mis näitab, kui palju variatsioonist selgitab iga eigenvektor).
eVal1 <- eigen(cM)$values[1]
eVal2 <- eigen(cM)$values[2]
lines(x=c(0, eVal1*eigenVecs[1,1]), y=c(0, eVal1*eigenVecs[2,1]), col="red")
lines(x=c(0, eVal2*eigenVecs[1,2]), y=c(0, eVal2*eigenVecs[2,2]), col="red")
```



```
#See maatriks sisaldab eigenvektoreid ning me tahame kuvada nende
#eigenvektoriteena. Praegusel juhul me kasutame mõlemat eigenvektorit, kuid
#kõrgedimensiooniliste andmete puhul kasutame ainult osa eigenvektoreid
#(see ongi PCA point).
rowFeatVec <- t(eigenVecs)
rowDataAdj <- t(m1)
#Viimaks kasutame maatriksi korrutamist, et saada punkt iga eigenvektoori ja iga
#originaalse tsentreeeritud andmete (m1) vahel. See operatsioon kirjeldab, mil
#määral igat punkti mõjutab iga eigenvektor. Plotime tulemused.
#NB! maatriksi korrutamise operaator R-s on %*%
transFData <- rowFeatVec %*% rowDataAdj
finalData <- rowFeatVec %*% rowDataAdj
plot(t(finalData), pch=seq(9))
```



```
#lõpuks plotime scree ploti ekvivalendi, mille me tegime üleval
#(plotime eiganväärtsused)
barplot(eigen(cM)$values)
```



Boonusülesanne 2 (2 punkti) - klassifitseeri numbreid

Paku välja moodus, kuidas pikslite põhjal eristada numbreid 0 ja 1. Leia mitmel juhul sinu meetod prognoosib õigesti, mitmel juhul valesti ja raporteeri täpsus (õigete klassifitseerimiste arv koguarvust). Võiksid täpsuseks saada vähemalt 90%.

```
#paneme pc1-d ja labelid kokku
ennustus=data.frame(pc1, numbrid$label)
nullid=subset(ennustus, pc1<=300)
sum(nullid$numbrid.label)

## [1] 6

#tundub, et 6 1-te on nullide tabelis, väga täpne. viga on:
sum(nullid$numbrid.label)/nrow(nullid)*100

## [1] 0.6103764

#ühtede puhul on viga
uhed=subset(ennustus, pc1>300)
sum(uhed$numbrid.label)/nrow(uhed)*100

## [1] 99.50836
```

```
#vigaade arv mõlemas
viga_uhed=nrow(uhed)-sum(uhed$numbrid.label)
viga_nullid=sum(nullid$numbrid.label)
#kogu viga
(viga_uhed+viga_nullid)/nrow(numbrid)*100
```

```
## [1] 0.55
```