HW4

Galen Byrd

4/26/2018

1

A

```
library(readxl)
track <- read_excel("track.xlsx")
```

```
## Warning in strptime(x, format, tz = tz): unknown timezone 'zone/tz/2018c.
## 1.0/zoneinfo/America/New_York'
```

```
n <- nrow(track)
p <- ncol(track)-1

(R <- cov(track[,-1]))
```

```
##                100m       200m       400m        800m      1500m
## 100m     0.12350249 0.20902182 0.43069956 0.016920438 0.03836684
## 200m     0.20902182 0.41557024 0.79905603 0.033115455 0.07788771
## 400m     0.43069956 0.79905603 2.12290020 0.080743131 0.18974209
## 800m     0.01692044 0.03311545 0.08074313 0.004055758 0.00911532
## 1500m    0.03836684 0.07788771 0.18974209 0.009115320 0.02430774
## 5000m    0.17441020 0.35913859 0.90887976 0.044062088 0.11592929
## 10,00m   0.40184545 0.81171145 2.07341549 0.100049327 0.26343721
## Marathon 1.68601222 3.54620963 9.47785704 0.473903333 1.24516296
##               5000m     10,00m   Marathon
## 100m     0.17441020  0.4018455  1.6860122
## 200m     0.35913859  0.8117114  3.5462096
## 400m     0.90887976  2.0734155  9.4778570
## 800m     0.04406209  0.1000493  0.4739033
## 1500m    0.11592929  0.2634372  1.2451630
## 5000m    0.64185811  1.4115480  6.8910485
## 10,00m   1.41154798  3.2678936 15.7321815
## Marathon 6.89104852 15.7321815 85.1381467
```

```
e.vec <- eigen(R)$vectors
(e.val <- eigen(R)$values)
```

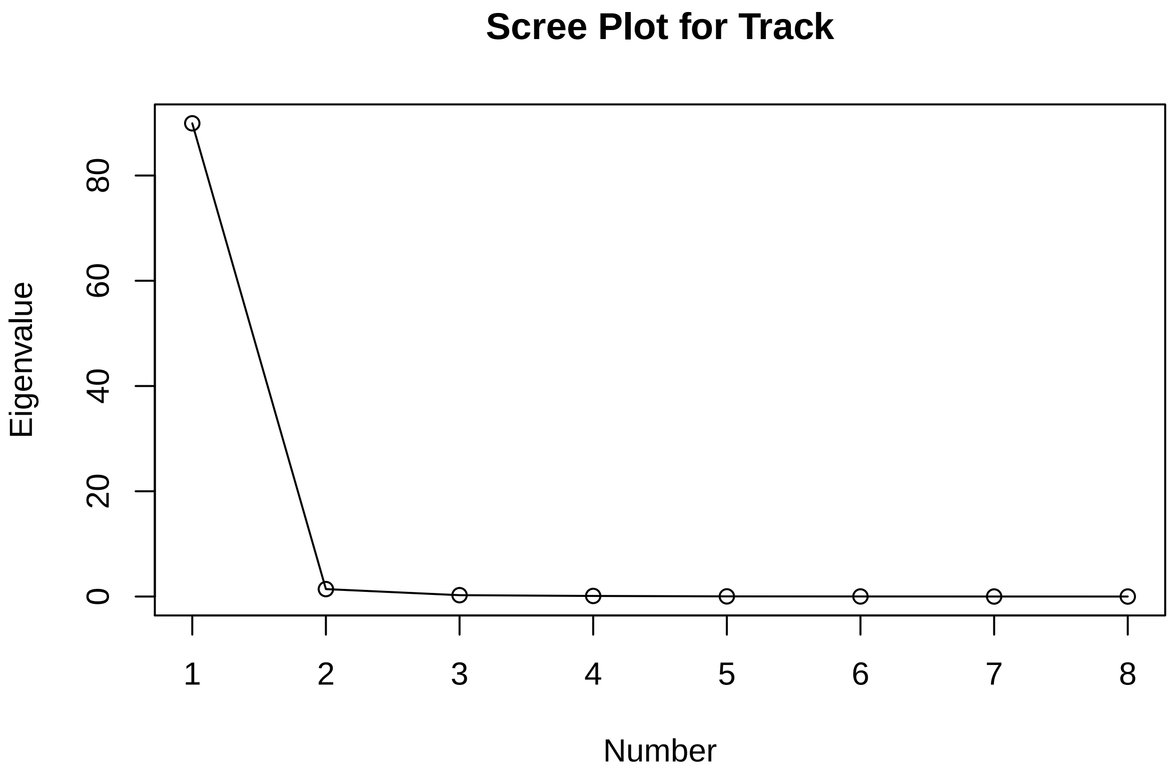
```
## [1] 8.991362e+01 1.412626e+00 2.598442e-01 1.094203e-01 2.730060e-02
## [6] 1.273280e-02 2.243554e-03 4.455645e-04
```

B

```
(mean(e.val))
```

```
## [1] 11.46728
```

```r
plot(1:p,e.val, xlab="Number",ylab="Eigenvalue",main="Scree Plot for Track", type="l")
points(1:p,e.val)
```




```r
percentage <- rep(0,p)
for (i in 1:p){
  percentage[i] <- sum(e.val[1:i])/sum(e.val)
}
(percentage)
```

```
## [1] 0.9801107 0.9955091 0.9983416 0.9995343 0.9998319 0.9999707 0.9999951
## [8] 1.0000000
```

```r
# Use just first as it accounts for 98% of variability
```

C

```r
(e.vec[,1:2])
```

```
##              [,1]        [,2]
## [1,] -0.019865407 -0.21068958
## [2,] -0.041554499 -0.35892579
## [3,] -0.110631838 -0.82786251
## [4,] -0.005487699 -0.02317490
## [5,] -0.014386822 -0.04465255
## [6,] -0.079308444 -0.12996134
## [7,] -0.181098994 -0.29885393
## [8,] -0.972787446  0.18080736
```

```
# First one is -.97 marathon, -.18 10,000m and -.11 400m. We can call this long distance running
# Second one is not super interpretable, it is -.82 400m, -.35 200m, -.29 10,000m, -.21 100m
# and +.18 for marathon
```

D

```
z<-as.matrix(track[,-1])%*%e.vec[,1]
(order(z))
```

```
##  [1] 12 55 46 16 35 36 51 26 41  5 42 34  7 23 50  1 28 13 32  3 22  9 14
## [24] 10  6 24 45 27 18 25 20 52 40 47 33 43 11 49 29 15 17 48 54  8  4 19
## [47] 31 37 39 21 38 44 30  2 53
```

```
# Top 5: usa,australia,japan,portugal,netherla
# Bottom 5: cookis,wsamoa,singapor,domrep,malaysia
```

2

A

```
(R <- cor(track[,-1]))
```

```
##               100m      200m      400m      800m     1500m     5000m
## 100m     1.0000000 0.9226384 0.8411468 0.7560278 0.7002382 0.6194618
## 200m     0.9226384 1.0000000 0.8507270 0.8066265 0.7749513 0.6953770
## 400m     0.8411468 0.8507270 1.0000000 0.8701714 0.8352694 0.7786139
## 800m     0.7560278 0.8066265 0.8701714 1.0000000 0.9180442 0.8635939
## 1500m    0.7002382 0.7749513 0.8352694 0.9180442 1.0000000 0.9281140
## 5000m    0.6194618 0.6953770 0.7786139 0.8635939 0.9281140 1.0000000
## 10,00m   0.6325389 0.6965391 0.7872045 0.8690489 0.9346970 0.9746354
## Marathon 0.5199490 0.5961837 0.7049905 0.8064764 0.8655492 0.9321884
##             10,00m  Marathon
## 100m     0.6325389 0.5199490
## 200m     0.6965391 0.5961837
## 400m     0.7872045 0.7049905
## 800m     0.8690489 0.8064764
## 1500m    0.9346970 0.8655492
## 5000m    0.9746354 0.9321884
## 10,00m   1.0000000 0.9431763
## Marathon 0.9431763 1.0000000
```

```
e.vec <- eigen(R)$vectors
(e.val <- eigen(R)$values)
```

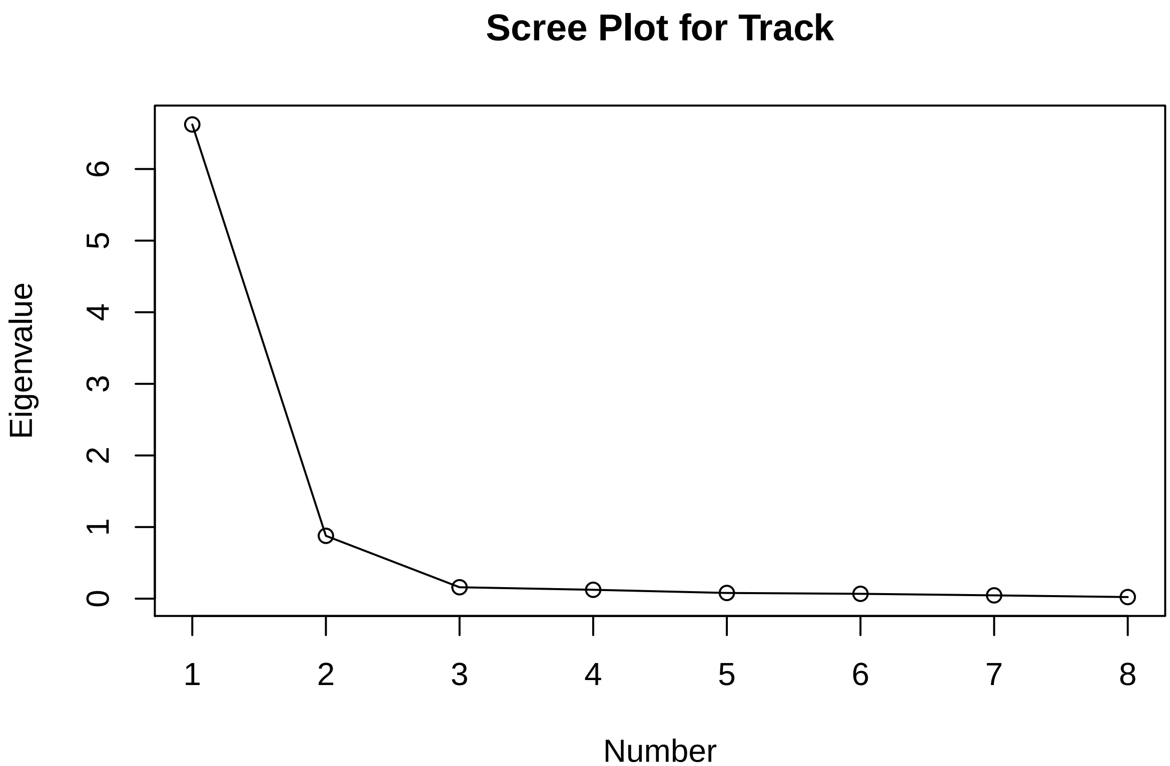
```
## [1] 6.62214613 0.87761829 0.15932114 0.12404939 0.07988027 0.06796515
## [7] 0.04641953 0.02260010
```

B

```r
(mean(e.val))
```

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

```r
plot(1:p,e.val, xlab="Number",ylab="Eigenvalue",main="Scree Plot for Track", type="l")
points(1:p,e.val)
```




```r
percentage <- rep(0,p)
for (i in 1:p){
  percentage[i] <- sum(e.val[1:i])/sum(e.val)
}
(percentage)
```

```
## [1] 0.8277683 0.9374706 0.9573857 0.9728919 0.9828769 0.9913725 0.9971750
## [8] 1.0000000
```

```r
# Use just first two as they account for 93% of variability and the second eigenvalue
# adds a significant amout of variability and the scree plot levels off there.
```

C

```r
(e.vec[,1:2])
```

```
##            [,1]        [,2]
## [1,] -0.3175565 -0.56687750
## [2,] -0.3369792 -0.46162589
## [3,] -0.3556454 -0.24827331
## [4,] -0.3686841 -0.01242993
## [5,] -0.3728099  0.13979665
```

```
## [6,] -0.3643741  0.31203045
## [7,] -0.3667726  0.30685985
## [8,] -0.3419261  0.43896267
```

```r
# First one is -.3+ for all variables. We can call this general running ability
# Second one is negative for the short distances and positive for long distances. As
# the distances get longer/shorter the absolute values get larger. We can call
# this long vs short ability. A positive number means the runner is a stronger long
# distance runner and a negative number means they are a stronger short distance runner.
```

D

```r
track.sc <- scale(track[,-1],center=T,scale=apply(track[,-1],2,sd))
z<-as.matrix(track.sc)%*%e.vec[,1:2]
(order(z[,1]))
```

```
##  [1] 12 55 36 41 46 51 26 23 13 42  7 16 35 33 50  5 28 10 52  1 34 32 25
## [24] 22  9 11 37  3 40 27 44 15 45 24 30 14 47 38  6 39 48 49 17  8 43  4
## [47] 31 18  2 20 19 54 29 21 53
```

```r
# Top 5: usa,gbni,italy,ussr,gdr
# Bottom 5: cookis,wsamoa,maritiu,png,singapor
```

E

```r
# The two results are different. They both agree that usa/cookis and wsamoa are the
# best/worst respectively, but after that they give different answers.
```

3

A

```r
track3<-cbind(track[,1],100/track[,2],200/track[,3],400/track[,4],800/(track[,5]*60),
              1500/(track[,6]*60),5000/(track[,7]*60),10000/(track[,8]*60),42195/(track[,9]*60))
```

B

```r
(R <- cov(track3[,-1]))
```

```
##                100m       200m       400m       800m      1500m      5000m
## 100m     0.09044592 0.07985342 0.06436012 0.05645566 0.05575291 0.05787011
## 200m     0.07985342 0.08272329 0.06308333 0.05785661 0.05966131 0.06364612
## 400m     0.06436012 0.06308333 0.06698808 0.05650067 0.05789093 0.06427206
## 800m     0.05645566 0.05785661 0.05650067 0.06272468 0.06116187 0.06898807
## 1500m    0.05575291 0.05966131 0.05789093 0.06116187 0.07216120 0.08024026
## 5000m    0.05787011 0.06364612 0.06427206 0.06898807 0.08024026 0.10417999
## 10,00m   0.06050769 0.06514185 0.06648693 0.07075964 0.08239365 0.10338860
## Marathon 0.04822916 0.05397306 0.05815799 0.06376408 0.07389448 0.09565101
```

```
##              10,00m   Marathon
## 100m     0.06050769 0.04822916
## 200m     0.06514185 0.05397306
## 400m     0.06648693 0.05815799
## 800m     0.07075964 0.06376408
## 1500m    0.08239365 0.07389448
## 5000m    0.10338860 0.09565101
## 10,00m   0.10862125 0.09919269
## Marathon 0.09919269 0.10214340
```

```r
e.vec <- eigen(R)$vectors
(e.val <- eigen(R)$values)
```

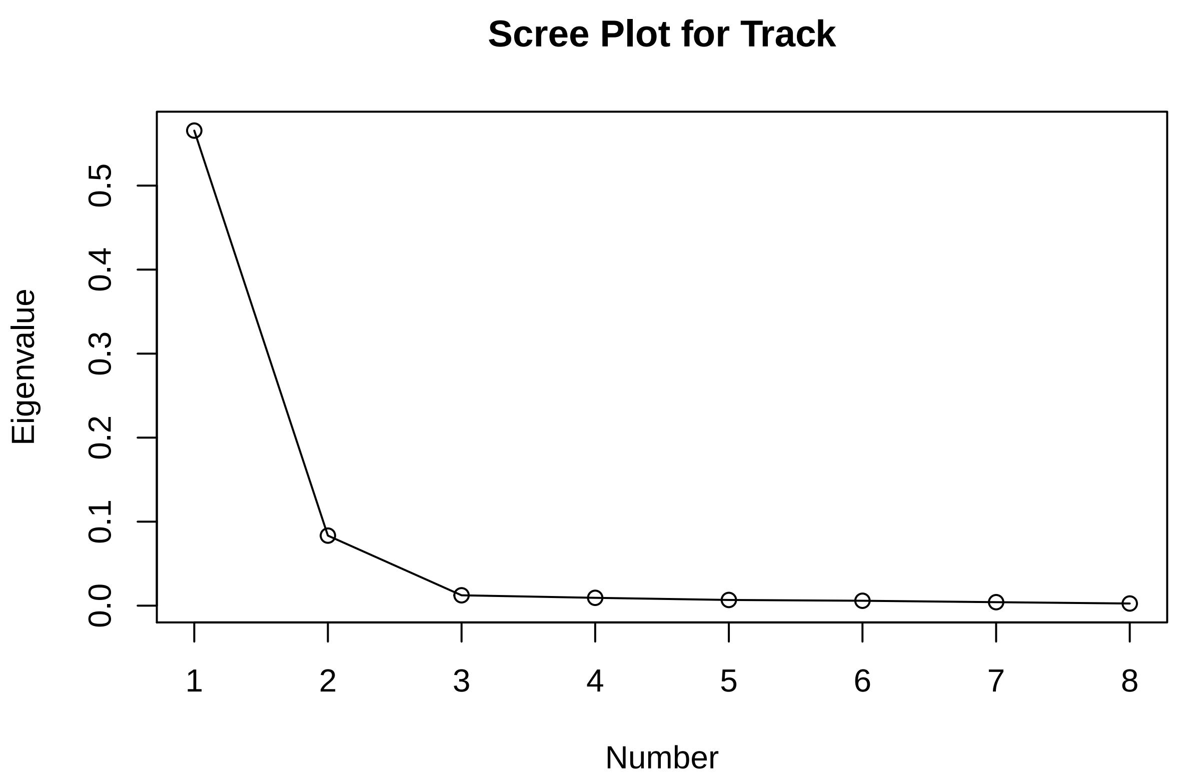
```
## [1] 0.565471242 0.083396555 0.012371382 0.009348032 0.006793014 0.005854749
## [7] 0.004160540 0.002592306
```

C

```r
(mean(e.val))
```

```
## [1] 0.08624848
```

```r
plot(1:p,e.val, xlab="Number",ylab="Eigenvalue",main="Scree Plot for Track", type="l")
points(1:p,e.val)
```




```r
percentage <- rep(0,p)
for (i in 1:p){
  percentage[i] <- sum(e.val[1:i])/sum(e.val)
}
(percentage)
```

```
## [1] 0.8195380 0.9404047 0.9583346 0.9718827 0.9817278 0.9902131 0.9962430
## [8] 1.0000000
```

```
# Use first two as they account for 94% of variability
```

D

```
(e.vec[,1:2])
```

```
##            [,1]        [,2]
## [1,] -0.3152136 -0.60269352
## [2,] -0.3248404 -0.47002261
## [3,] -0.3094965 -0.23076768
## [4,] -0.3120833 -0.05598404
## [5,] -0.3427922  0.07902824
## [6,] -0.4063244  0.29554455
## [7,] -0.4178300  0.29648919
## [8,] -0.3804563  0.42184529
```

```
# First one is -.3+ for all variables. We can call this general running
# Second one is similar to the second from 2.negative for the short distances
# and positive for long distances. As the distances get longer/shorter the
# absolute values get larger. We can call this long vs short ability. A positive
# number means the runner is a stronger long distance runner and a negative
# number means they are a stronger short distance runner.
```

E

```
track.sc <- scale(track3[,-1],center=T,scale=apply(track3[,-1],2,sd))
z<-as.matrix(track.sc)%*%e.vec
order(z[,1])
```

```
##  [1] 53 21 29 54 19 20  2 31 18  4 43  8 17 49 39 48 38 47  6 14 30 45 24
## [24] 15 44 27 40  3 37 11  9 22 25 52 34 32  1 10 28  5 50 33 16 35  7 13
## [47] 42 23 26 51 46 41 36 55 12
```

```
# Top 5: usa,gbni,italy,ussr,gdr
# Bottom 5: cookis,wsamoa,mauritiu,png,singapor
```

F

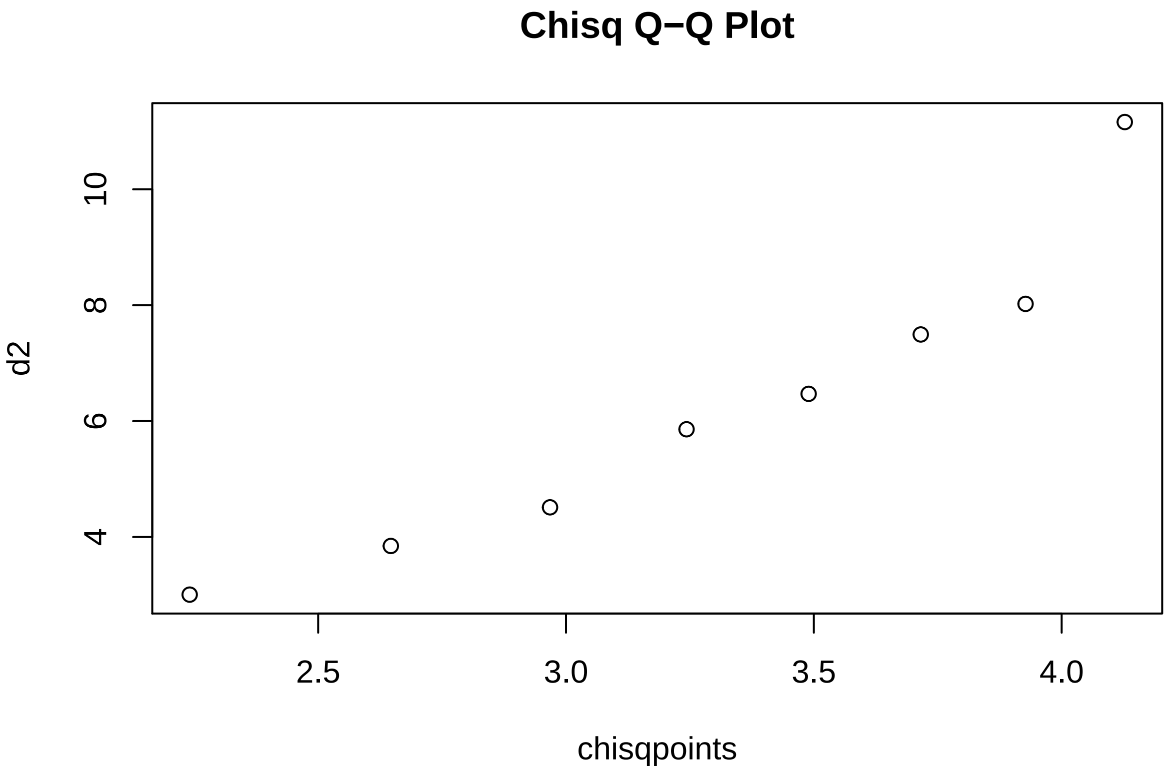
```
# I prefer the method from question 3 because we started by standardizing the scale of the variables
```

4

A

```
R <- cov(track[,-1])
meanvec<-colMeans(track[,-1])
```

```r
d2 <- NULL
for (i in 2:9){
  d2 <- c(d2,(as.matrix(track[i,-1])-meanvec)%*%solve(R)%*%(t(as.matrix(track[i,-1])-meanvec)))
}
j=2:9
chisqpoints <- qchisq((j-1/2)/n,df=p)
qqplot(chisqpoints,d2, main="Chisq Q-Q Plot")
```



B

```r
# Data is MVN because Chi-Square Q-Q plot is linear so use Maximum likelihood method
```

C

```r
library(psych)
R <- cor(track[,-1])
FA.ML1 <- factanal(covmat=R, factors=1, rotation="none")
Psi.ml1 <- diag(diag(R-FA.ML1$loadings%*%t(FA.ML1$loadings)))
(FA.ML1.res <- round(R-(FA.ML1$loadings%*%t(FA.ML1$loadings)+Psi.ml1),2))
```

```
##          100m  200m  400m  800m 1500m 5000m 10,00m Marathon
## 100m     0.00  0.43  0.30  0.16  0.07 -0.03  -0.03    -0.11
## 200m     0.43  0.00  0.25  0.15  0.08 -0.03  -0.03    -0.09
## 400m     0.30  0.25  0.00  0.14  0.06 -0.02  -0.02    -0.06
## 800m     0.16  0.15  0.14  0.00  0.06 -0.02  -0.02    -0.04
## 1500m    0.07  0.08  0.06  0.06  0.00 -0.01  -0.01    -0.03
```

```
## 5000m    -0.03 -0.03 -0.02 -0.02 -0.01  0.00   0.01     0.01
## 10,00m   -0.03 -0.03 -0.02 -0.02 -0.01  0.01   0.00     0.02
## Marathon -0.11 -0.09 -0.06 -0.04 -0.03  0.01   0.02     0.00
```

```
FA.ML2 <- factanal(covmat=R, factors=2, rotation="none")
Psi.ml2 <- diag(diag(R-FA.ML2$loadings%*%t(FA.ML2$loadings)))
(FA.ML2.res <-round(R-(FA.ML2$loadings%*%t(FA.ML2$loadings)+Psi.ml2),2))
```

```
##           100m  200m  400m  800m 1500m 5000m 10,00m Marathon
## 100m      0.00  0.01  0.00 -0.01 -0.02     0   0.00     0.00
## 200m      0.01  0.00 -0.01 -0.01  0.00     0   0.00     0.00
## 400m      0.00 -0.01  0.00  0.03  0.01     0   0.00     0.00
## 800m     -0.01 -0.01  0.03  0.00  0.04     0  -0.01     0.00
## 1500m    -0.02  0.00  0.01  0.04  0.00     0   0.00    -0.01
## 5000m     0.00  0.00  0.00  0.00  0.00     0   0.00     0.00
## 10,00m    0.00  0.00  0.00 -0.01  0.00     0   0.00     0.00
## Marathon  0.00  0.00  0.00  0.00 -0.01     0   0.00     0.00
```

```
# We should use m=2
```

D

```
FA.ML2 <- factanal(covmat=R, factors=2, rotation="varimax")
Psi.ml2 <- diag(diag(R-FA.ML2$loadings%*%t(FA.ML2$loadings)))
FA.ML2.res <-round(R-(FA.ML2$loadings%*%t(FA.ML2$loadings)+Psi.ml2),2)
(L <-FA.ML2$loadings)
```

```
##
## Loadings:
##          Factor1 Factor2
## 100m     0.291   0.914
## 200m     0.382   0.882
## 400m     0.543   0.744
## 800m     0.691   0.622
## 1500m    0.799   0.530
## 5000m    0.901   0.394
## 10,00m   0.907   0.399
## Marathon 0.915   0.278
##
##                Factor1 Factor2
## SS loadings      4.112   3.225
## Proportion Var   0.514   0.403
## Cumulative Var   0.514   0.917
```

```
# Factor 1 is long distance running ability
# Factor 2 is short distance running ability
```

E

```
v<-track[53,-1]
(f <- t(as.matrix(L))%*%solve(R)%*%t(as.matrix(v-meanvec)))
```

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

```
## Factor1 -2.3362597
## Factor2  0.4137613
```