Class-5-Summary

Jenna G. Tichon

16/11/2019

7.2 What are the data? Matrices and their motivation

```
turtles = read.table(here("data", "PaintedTurtles.txt"), header = TRUE)
turtles[1:4, ]
```

```
##   sex length width height
## 1   f     98    81     38
## 2   f    103    84     38
## 3   f    103    86     42
## 4   f    105    86     40
```

```
load(here("data", "athletes.RData"))
athletes[1:3,]
```

```
##    m100 long weight highj  m400  m110  disc pole javel  m1500
## 1 11.25 7.43  15.48  2.27 48.90 15.13 49.28  4.7 61.32 268.95
## 2 10.87 7.45  14.97  1.97 47.71 14.46 44.36  5.1 61.76 273.02
## 3 11.18 7.44  14.20  1.97 48.29 14.81 43.66  5.2 64.16 263.20
```

```
load(here("data", "Msig3transp.RData"))
round(Msig3transp,2)[1:5, 1:6]
```

```
##              X3968 X14831 X13492 X5108 X16348  X585
## HEA26_EFFE_1 -2.61  -1.19  -0.06 -0.15   0.52 -0.02
## HEA26_MEM_1  -2.26  -0.47   0.28  0.54  -0.37  0.11
## HEA26_NAI_1  -0.27   0.82   0.81  0.72  -0.90  0.75
## MEL36_EFFE_1 -2.24  -1.08  -0.24 -0.18   0.64  0.01
## MEL36_MEM_1  -2.68  -0.15   0.25  0.95  -0.20  0.17
```

```
data("GlobalPatterns", package = "phyloseq")
GPOTUs = as.matrix(t(phyloseq::otu_table(GlobalPatterns)))
GPOTUs[1:4, 6:13]
```

```
## OTU Table:          [8 taxa and 4 samples]
##                      taxa are columns
##         246140 143239 244960 255340 144887 141782 215972 31759
## CL3          0      7      0    153      3      9      0     0
## CC1          0      1      0    194      5     35      3     1
## SV1          0      0      0      0      0      0      0     0
## M31Fcsw      0      0      0      0      0      0      0     0
```

```
library("SummarizedExperiment")
```

```
## Loading required package: GenomicRanges
```

```
## Loading required package: stats4
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which, which.max, which.min

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:dplyr':
##
##     first, rename

## The following object is masked from 'package:tidyr':
##
##     expand

## The following object is masked from 'package:base':
##
##     expand.grid

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following object is masked from 'package:phyloseq':
##
##     distance

## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice

## The following object is masked from 'package:purrr':
##
##     reduce

## Loading required package: GenomeInfoDb
```

```
## Loading required package: Biobase
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
##
## Attaching package: 'Biobase'
## The following object is masked from 'package:phyloseq':
##
##     sampleNames
## Loading required package: DelayedArray
## Loading required package: matrixStats
##
## Attaching package: 'matrixStats'
## The following objects are masked from 'package:Biobase':
##
##     anyMissing, rowMedians
## The following object is masked from 'package:dplyr':
##
##     count
## Loading required package: BiocParallel
##
## Attaching package: 'DelayedArray'
## The following objects are masked from 'package:matrixStats':
##
##     colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges
## The following object is masked from 'package:purrr':
##
##     simplify
## The following objects are masked from 'package:base':
##
##     aperm, apply, rowsum
data("airway", package = "airway")
assay(airway)[1:3, 1:4]
##                 SRR1039508 SRR1039509 SRR1039512 SRR1039513
## ENSG00000000003        679        448        873        408
## ENSG00000000005          0          0          0          0
## ENSG00000000419        467        515        621        365
metab = t(as.matrix(read.csv(here("data","metabolites.csv"), row.names = 1)))
metab[1:4, 1:4]
##          146.0985388 148.7053275 310.1505057 132.4512963
## KOGCHUM1    29932.36    17055.70     1132.82    785.5129
## KOGCHUM2    94067.61    74631.69    28240.85   5232.0499
## KOGCHUM3   146411.33   147788.71    64950.49  10283.0037
```

```
## WTGCHUM1   229912.57   384932.56   220730.39  26115.2007
```

Task

```
length(which(metab==0))
```

```
## [1] 604
```

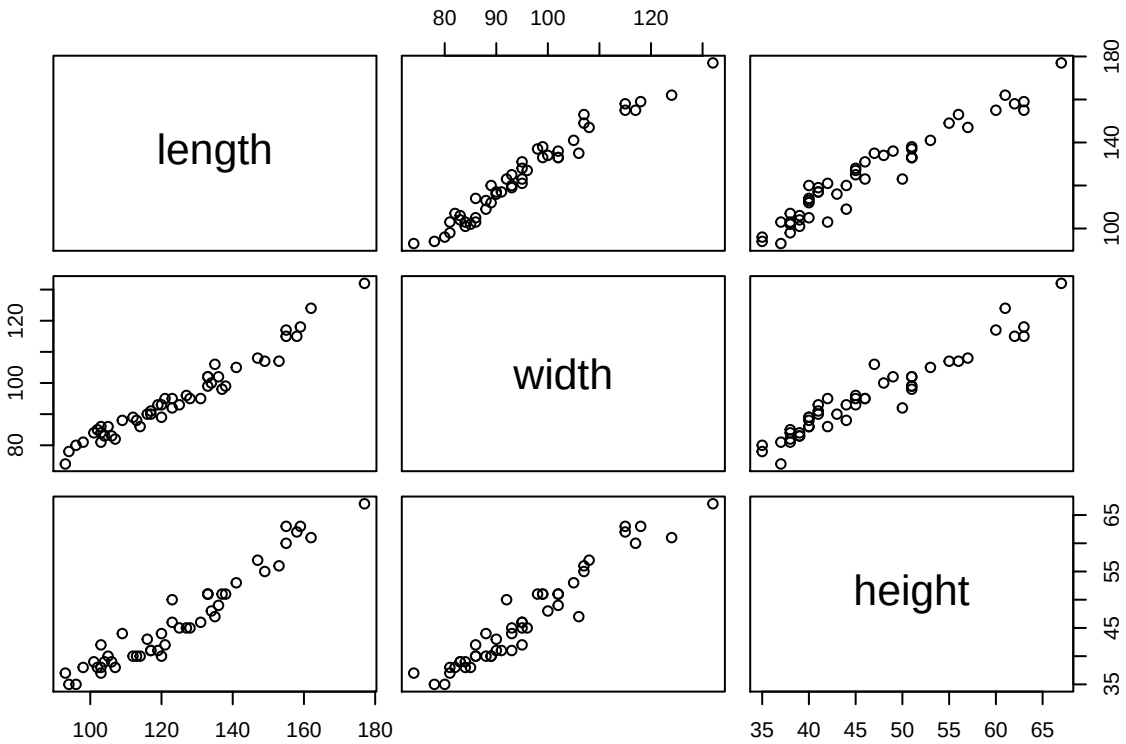
```
cor(turtles[,2:4])
```

```
##           length     width    height
## length 1.0000000 0.9783116 0.9646946
## width  0.9783116 1.0000000 0.9605705
## height 0.9646946 0.9605705 1.0000000
```

Q 7.3

(a)

```
plot(turtles[,2:4])
```




```
library("GGally")
```

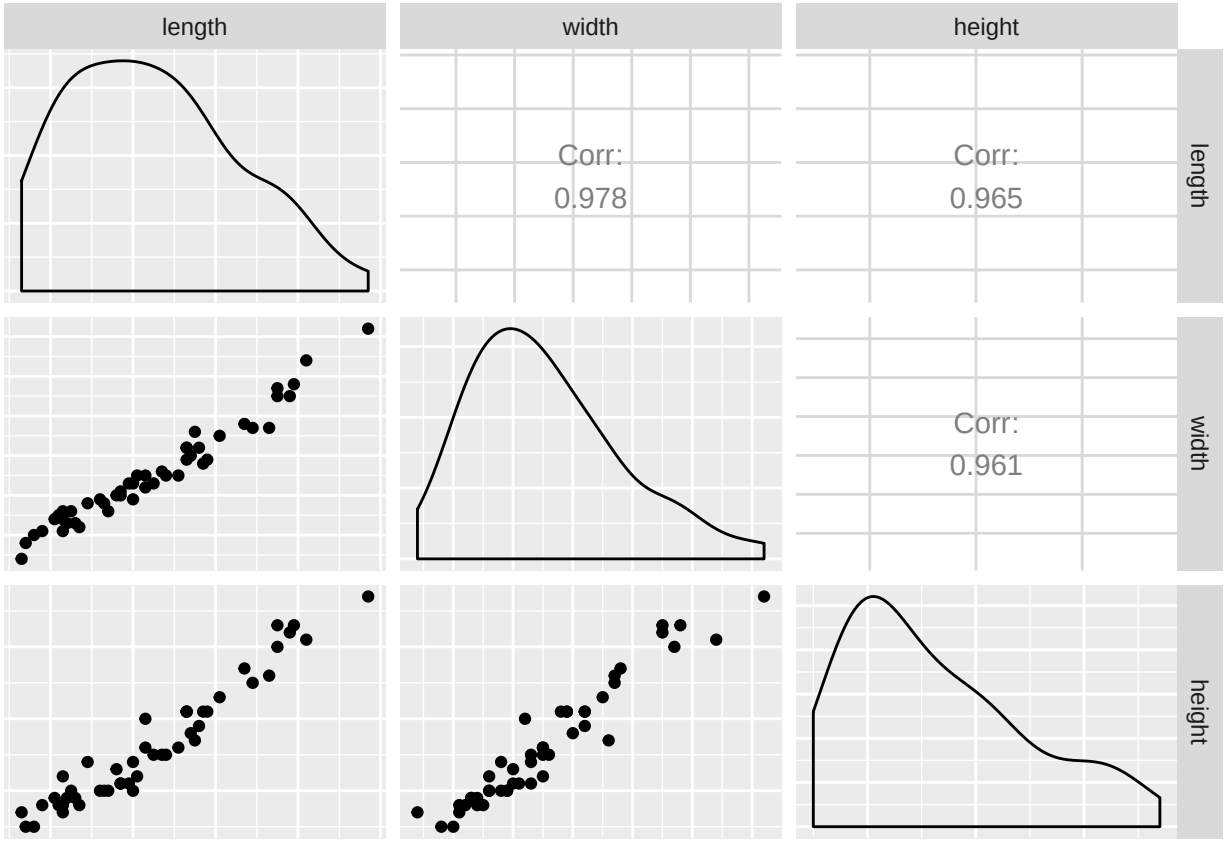
```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
##
## Attaching package: 'GGally'
```

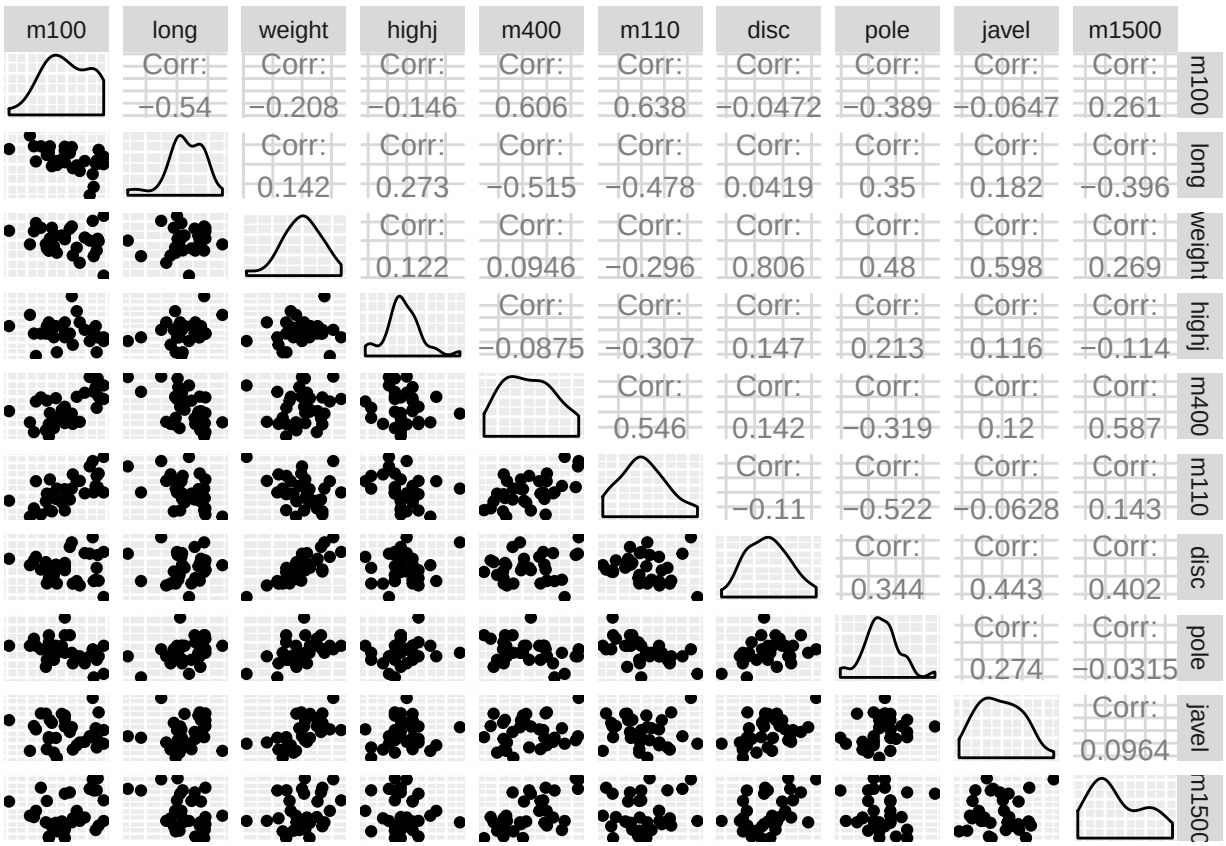
```
## The following object is masked from 'package:dplyr':
```

```
##
##     nasa
```

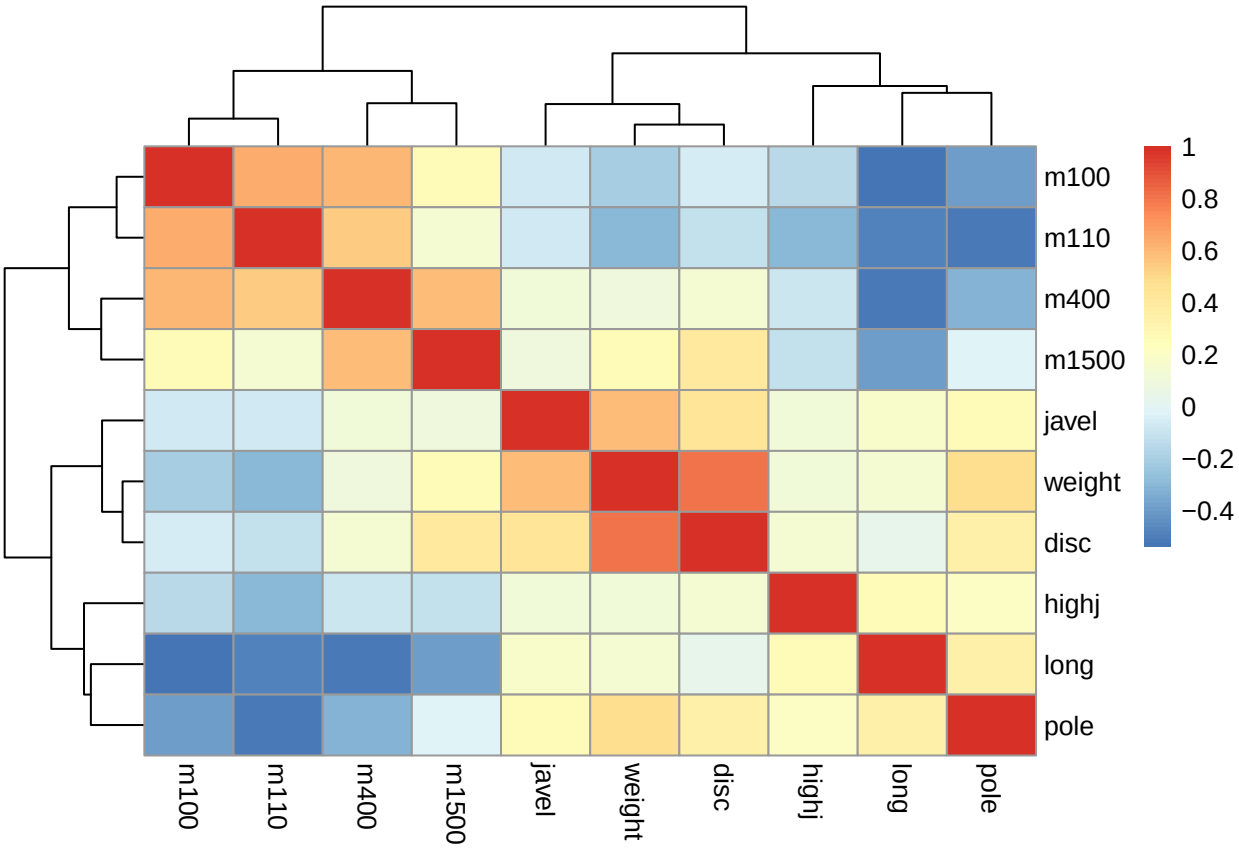
```
ggpairs(turtles[, -1], axisLabels = "none")
```




```
ggpairs(athletes, axisLabels = "none")
```




```
library("pheatmap")
pheatmap(cor(athletes), cell.width = 10, cell.height = 10)
```



7.2.2 Preprocessing the data

scale() in R makes everything in the matrix have mean 0 and standard deviation 1.

Q 7.5

Compute means and standard deviations of turtle them use scale function to create `scaledTurtles`.

(a)

```
#Find means and standard deviations
(turtles.mean<-turtles[,2:4] %>%
  map_dfr(mean) )
```

```
## # A tibble: 1 x 3
##   length width height
##    <dbl> <dbl>  <dbl>
## 1   125.  95.4   46.3
```

```
(turtles.sd<-turtles[,2:4] %>%
  map_dfr(sd) )
```

```
## # A tibble: 1 x 3
##   length width height
##    <dbl> <dbl>  <dbl>
## 1   20.5  12.7   8.39
```

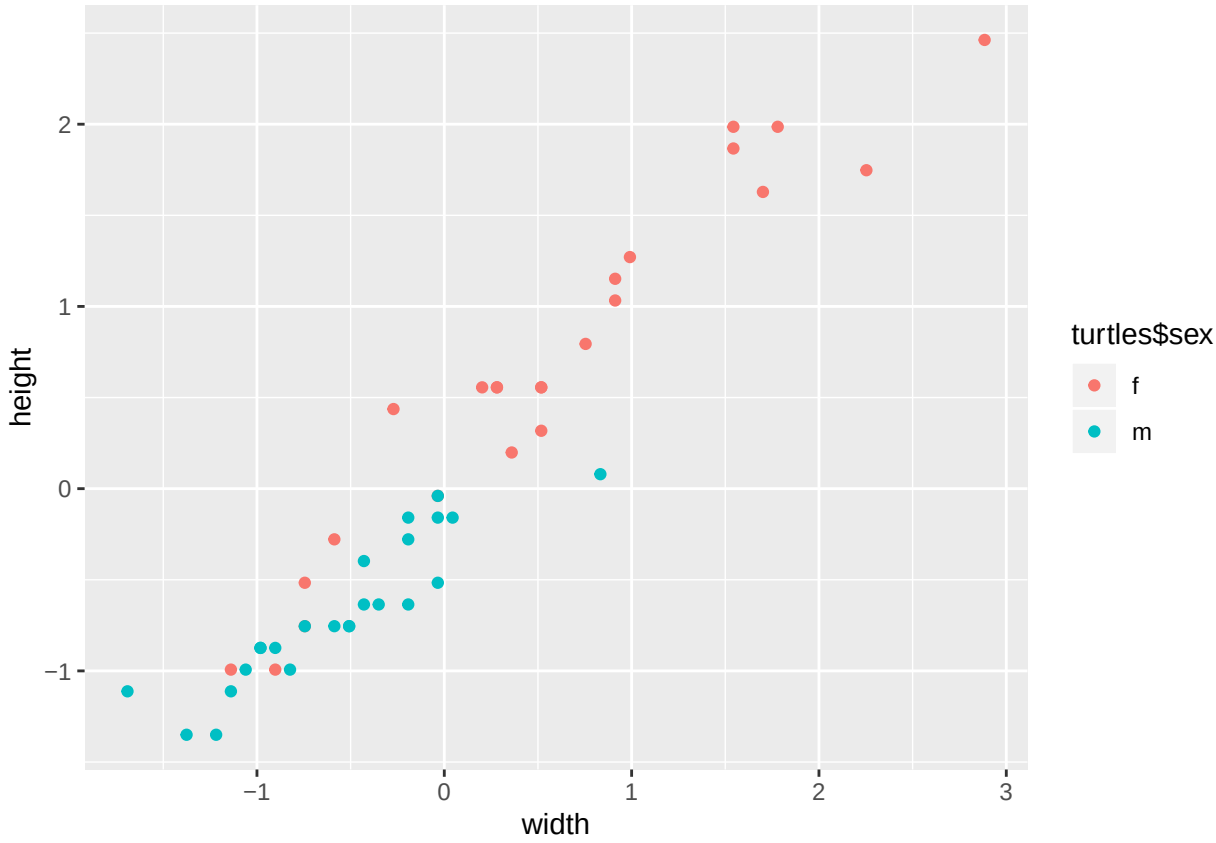
```
#Use scale functin
scaledTurtles<-scale(turtles[,2:4])
#Verify it now has mean 0 and standard deviation 1
apply(scaledTurtles, 2, sd)
```

```
## length  width height
##      1      1      1
```

```
apply(scaledTurtles, 2, mean)
```

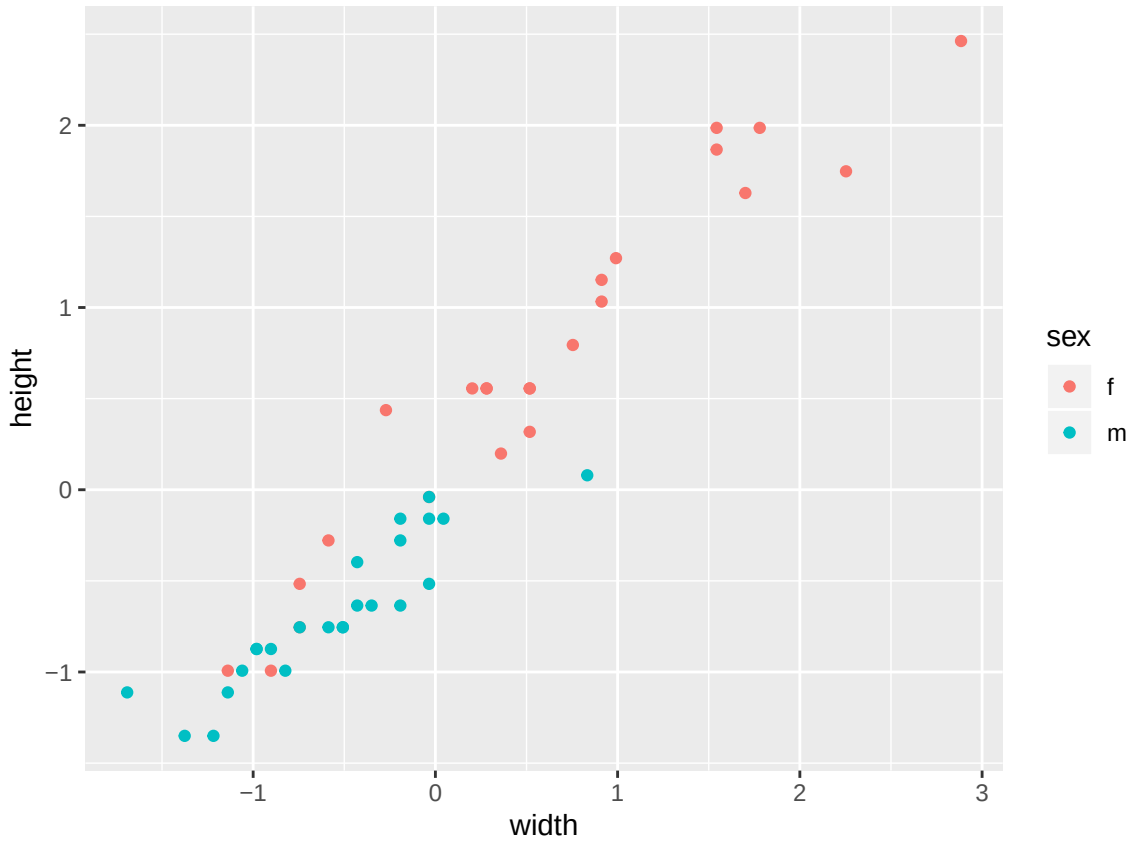
```
##        length         width        height
## -1.432050e-18  1.940383e-17 -2.870967e-16
```

```
ggplot(data = as.data.frame(scaledTurtles), mapping = aes(x = width, y = height, col = turtles$sex)) +
  geom_point()
```



Text code: note the simpler use of piping and combining scaledTurtles with the sex column before making the ggplot to make this simpler:

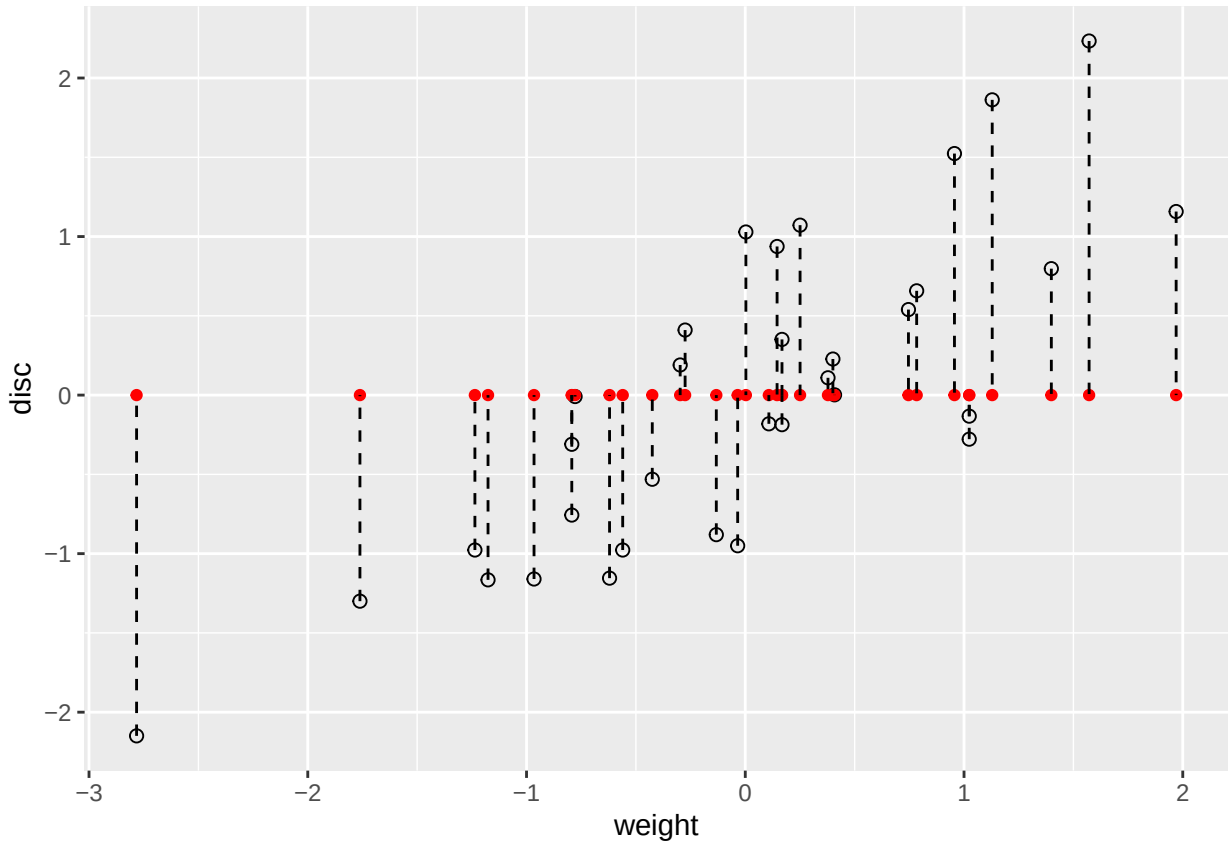
```
data.frame(scaledTurtles, sex = turtles[, 1]) %>%
  ggplot(aes(x = width, y = height, group = sex)) +
    geom_point(aes(color = sex)) + coord_fixed()
```



7.3 Dimension reduction

Orthogonal projection of the athletes data onto the vector at y=0

```
athletes = data.frame(scale(athletes))
#Produce a scatterplot of disc vs weight
ath_gg = ggplot(athletes, aes(x = weight, y = disc)) +
  geom_point(size = 2, shape = 21)
#Produce a straight line at y=0 in red then create dashed line from points to line. xend and yend say w
ath_gg + geom_point(aes(y = 0), colour = "red") +
  geom_segment(aes(xend = weight, yend = 0), linetype = "dashed")
```



(a) Calculate the variance of the red points

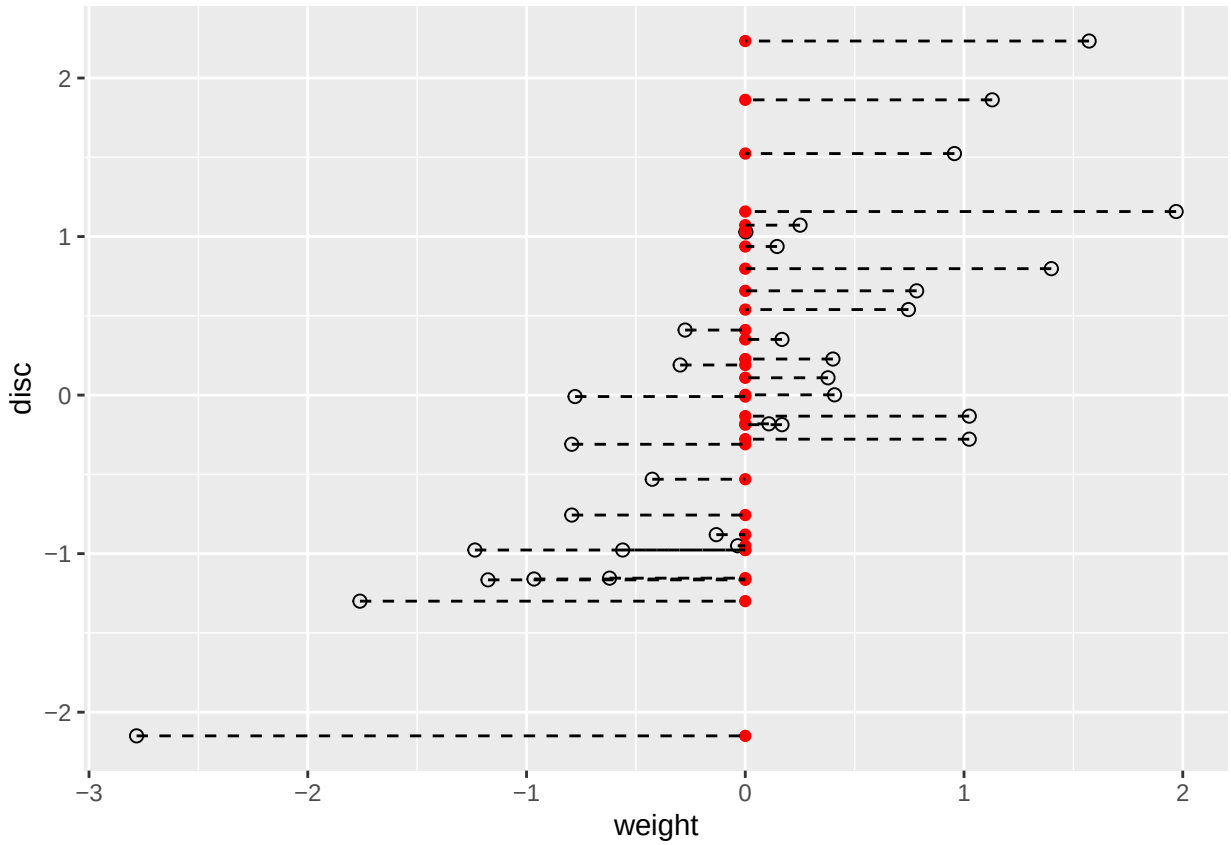
```
var(athletes$weight)
```

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

this has a variance of 1 because these are scaled points with sd 1.

(b) Make a plot showing projection lines onto the y axis and projected points

```
#Produce a scatterplot of disc vs weight
ath_gg = ggplot(athletes, aes(x = weight, y = disc)) +
  geom_point(size = 2, shape = 21)
#Produce a straight line at x=0 in red then create dashed line from points to line
ath_gg + geom_point(aes(x = 0), colour = "red") +
  geom_segment(aes(xend = 0, yend = disc), linetype = "dashed")
```



(c) Compute the variance of the points projected onto the vertical y axis

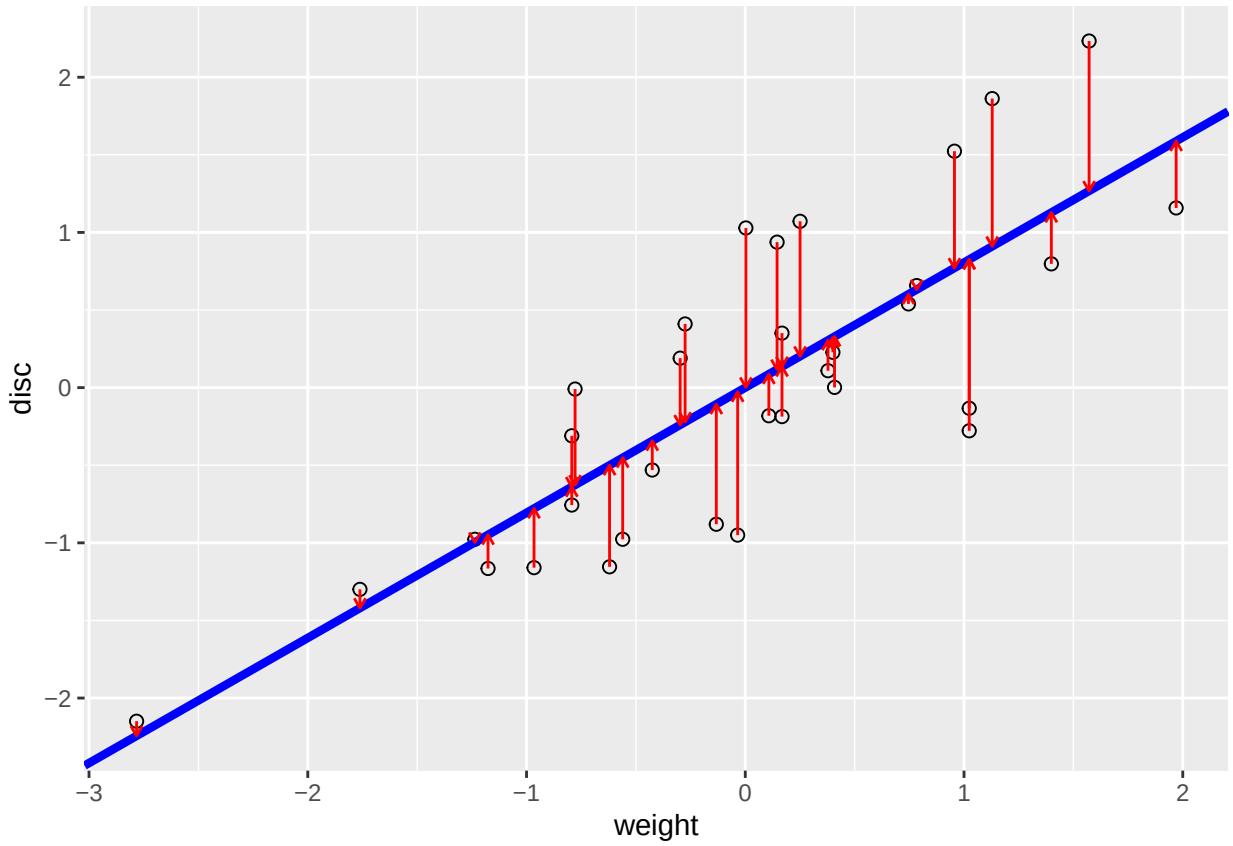
```
var(athletes$disc)
```

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

7.3.2 How do we summarize two-dimensional data by a line?

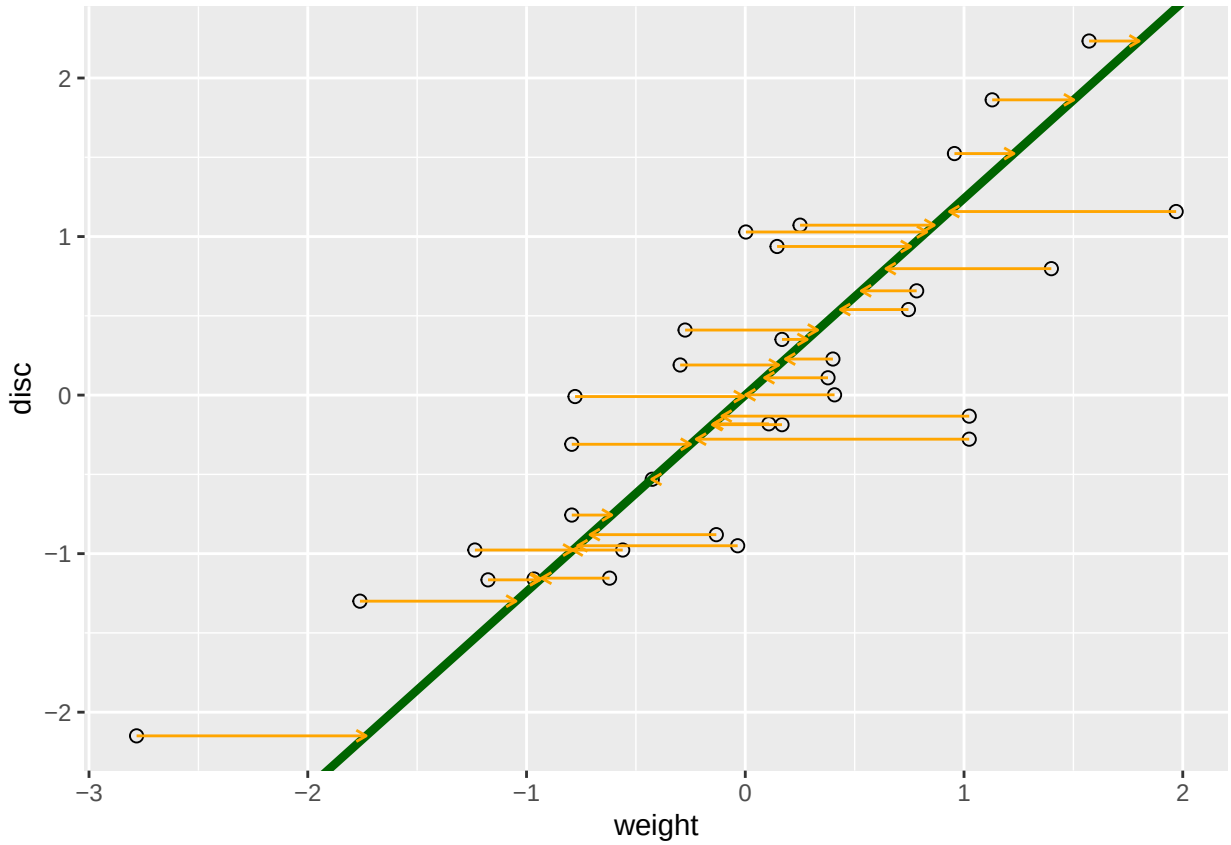
Regress disc on weight

```
#Regress disc on weight and extract the coefficients
reg1 = lm(disc ~ weight, data = athletes)
a1 = reg1$coefficients[1] # intercept
b1 = reg1$coefficients[2] # slope
#Fit the regression line to the scatterplot in blue
pline1 = ath_gg + geom_abline(intercept = a1, slope = b1,
    col = "blue", lwd = 1.5)
#Make arrows from points to fitted line
pline1 + geom_segment(aes(xend = weight, yend = reg1$fitted),
    colour = "red", arrow = arrow(length = unit(0.15, "cm")))
```



Redo flipping x and y

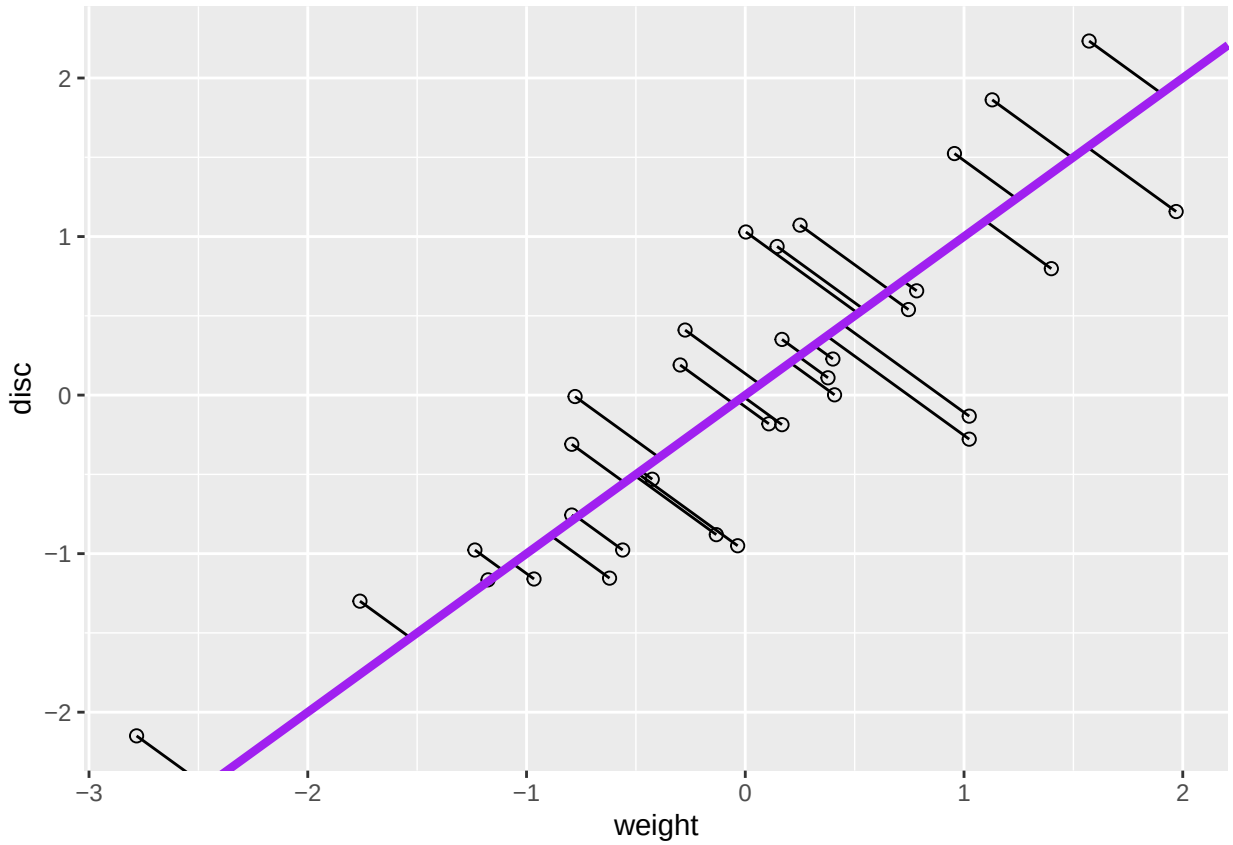
```
reg2 = lm(weight ~ disc, data = athletes)
a2 = reg2$coefficients[1] # intercept
b2 = reg2$coefficients[2] # slope
pline2 = ath_gg + geom_abline(intercept = -a2/b2, slope = 1/b2,
    col = "darkgreen", lwd = 1.5)
pline2 + geom_segment(aes(xend=reg2$fitted, yend=disc),
    colour = "orange", arrow = arrow(length = unit(0.15, "cm")))
```



ALERT!!: Look up single value decomposition

Make the line that minimizes the orthogonal distances called the **principal component** line. Here are three ways of fitting it:

```
xy = cbind(athletes$disc, athletes$weight)
#Calculate singular value decomposition (svd) of matrix with just x,y
svda = svd(xy)
#Calculate principal components
pc = xy %*% svda$v[, 1] %*% t(svda$v[, 1])
#slope of line
bp = svda$v[2, 1] / svda$v[1, 1]
#intercept of line
ap = mean(pc[, 2]) - bp * mean(pc[, 1])
#ath_gg already has scatterplot. geom_segment says to take each point and end at the xy principal compo
ath_gg + geom_segment(xend = pc[, 1], yend = pc[, 2]) +
  geom_abline(intercept = ap, slope = bp, col = "purple", lwd = 1.5)
```



Note this passes through the origin as we centered all data and is in the middle of both regression lines

Q 7.8

```
var(athletes$disc*bp+ap)
```

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

Principal components is minimizing the orthogonal projections onto the line. It also maximized the variance of projections along the line. We want to think of liking maximizing variance as being able to "see" as much of the data as possible.

7.4.1 Optimal Lines

7.6 The inner workings of PCA: rank reduction

7.6.1 Rank-one matrices

Looking for u and v such that $u * v^t$=matrix. This is not unique so pick vectors where sums of squares sum to 1. Then there is a scaling factor to get the "non-decimal" values back.

```
#Square Matrix
X = matrix(c(780,  75, 540,
             936,  90, 648,
            1300, 125, 900,
             728,  70, 504), nrow = 3)
```

```r
#u and v decomposition
u = c(0.8196, 0.0788, 0.5674)
v = c(0.4053, 0.4863, 0.6754, 0.3782)
#scaling factor
s1 = 2348.2
#Show sums of squares equal 1
sum(u^2)
```

```
## [1] 0.9998964
```

```r
sum(v^2)
```

```
## [1] 0.9999562
```

```r
#Get original matrix back from decomposition
s1 * u %*% t(v)
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] 780.03419 935.92555 1299.8645 727.87794
## [2,]  74.99597  89.98406  124.9748  69.98143
## [3,] 540.00903 647.93089  899.8818 503.90183
```

```r
X - s1 * u %*% t(v)
```

```
##              [,1]       [,2]       [,3]       [,4]
## [1,] -0.034187016 0.07445066 0.13548011 0.12205890
## [2,]  0.004033752 0.01594279 0.02522674 0.01856789
## [3,] -0.009026004 0.06911092 0.11819353 0.09816522
```

```r
#The scaling factor is the first observation in the d, which matches the first columns of u and v which
svd(X)
```

```
## $d
## [1] 2.348244e+03 2.141733e-13 6.912584e-15
##
## $u
##            [,1]         [,2]        [,3]
## [1,] 0.81963482  0.569413084  0.06298807
## [2,] 0.07881104 -0.003168944 -0.99688454
## [3,] 0.56743949 -0.822045435  0.04747341
##
## $v
##           [,1]        [,2]       [,3]
## [1,] 0.4052574  0.88432390 -0.1978361
## [2,] 0.4863089 -0.13032307  0.7123009
## [3,] 0.6754290 -0.44787634 -0.5829493
## [4,] 0.3782403  0.01984752  0.3371327
```

7.6.2 How do we find such a decomposition in a unique way?

```r
(Xtwo = matrix(c(12.5, 35.0, 25.0, 25, 9, 14, 26, 18, 16, 21, 49, 32,
       18, 28, 52, 36, 18, 10.5, 64.5, 36), ncol = 4, byrow = TRUE))
```

```
##      [,1] [,2] [,3] [,4]
## [1,] 12.5 35.0 25.0   25
## [2,]  9.0 14.0 26.0   18
```

```
## [3,] 16.0 21.0 49.0   32
## [4,] 18.0 28.0 52.0   36
## [5,] 18.0 10.5 64.5   36
```

```
(USV = svd(Xtwo))
```

```
## $d
## [1] 1.350624e+02 2.805191e+01 3.103005e-15 1.849559e-15
##
## $u
##            [,1]        [,2]        [,3]       [,4]
## [1,] -0.3443135  0.77172177  0.51932548 -0.1139032
## [2,] -0.2641680  0.07127562 -0.30861530 -0.5038052
## [3,] -0.4752787 -0.04146976 -0.03855246  0.8027556
## [4,] -0.5283361  0.14255124 -0.64225606 -0.1029211
## [5,] -0.5537460 -0.61426761  0.47018986 -0.2796374
##
## $v
##            [,1]        [,2]       [,3]        [,4]
## [1,] -0.2499835  0.04041051 -0.9670994  0.02437097
## [2,] -0.3430854  0.87975766  0.1330304  0.30103454
## [3,] -0.7548732 -0.46675628  0.1862426  0.42144835
## [4,] -0.4999671  0.08082102  0.1110645 -0.85508219
```

```
#How far is approximation with first singular vectors from Xtwo
Xtwo - USV$d[1] * USV$u[, 1] %*% t(USV$v[, 1])
```

```
##             [,1]       [,2]        [,3]        [,4]
## [1,]  0.87481760  19.045230 -10.1044650  1.74963521
## [2,]  0.08079747   1.759002  -0.9332405  0.16159494
## [3,] -0.04700978  -1.023427   0.5429803 -0.09401956
## [4,]  0.16159494   3.518005  -1.8664809  0.32318987
## [5,] -0.69632883 -15.159437   8.0428540 -1.39265765
```

```
#How far off is second approximation from the first approximation
Xtwo - USV$d[1] * USV$u[, 1] %*% t(USV$v[, 1]) -
       USV$d[2] * USV$u[, 2] %*% t(USV$v[, 2])
```

```
##              [,1]          [,2]         [,3]         [,4]
## [1,] 7.216450e-15 -1.065814e-14 8.881784e-15 4.884981e-15
## [2,] 2.040035e-15 -5.995204e-15 1.054712e-14 3.219647e-15
## [3,] 2.865763e-15 -9.547918e-15 1.554312e-15 6.231127e-15
## [4,] 4.385381e-15 -5.773160e-15 1.776357e-14 7.049916e-15
## [5,] 5.107026e-15 -1.776357e-15 1.776357e-14 1.776357e-14
```

```
#Little improvement from third
Xtwo - USV$d[1] * USV$u[, 1] %*% t(USV$v[, 1]) -
       USV$d[2] * USV$u[, 2] %*% t(USV$v[, 2]) -
       USV$d[3] * USV$u[, 3] %*% t(USV$v[, 3])
```

```
##              [,1]          [,2]         [,3]         [,4]
## [1,] 8.774901e-15 -1.087252e-14 8.581660e-15 4.706004e-15
## [2,] 1.113907e-15 -5.867810e-15 1.072547e-14 3.326006e-15
## [3,] 2.750071e-15 -9.532004e-15 1.576592e-15 6.244413e-15
## [4,] 2.458026e-15 -5.508040e-15 1.813474e-14 7.271259e-15
## [5,] 6.518025e-15 -1.970448e-15 1.749184e-14 1.760153e-14
```

```
#Little from fourth
Xtwo - USV$d[1] * USV$u[, 1] %*% t(USV$v[, 1]) -
       USV$d[2] * USV$u[, 2] %*% t(USV$v[, 2]) -
       USV$d[3] * USV$u[, 3] %*% t(USV$v[, 3]) -
       USV$d[4] * USV$u[, 4] %*% t(USV$v[, 4])
```

```
##              [,1]          [,2]         [,3]         [,4]
## [1,] 8.780035e-15 -1.080910e-14 8.670447e-15 4.525864e-15
## [2,] 1.136616e-15 -5.587301e-15 1.111818e-14 2.529225e-15
## [3,] 2.713886e-15 -9.978963e-15 9.508493e-16 7.513991e-15
## [4,] 2.462665e-15 -5.450736e-15 1.821496e-14 7.108487e-15
## [5,] 6.530630e-15 -1.814752e-15 1.770982e-14 1.715927e-14
```

Only second approximation improves so it is of rank 2. We'll also notice that second d from svd decomp was last "non-zero" term.

SVD for the turtles data:

```
turtles.svd = svd(scaledTurtles)
turtles.svd$d
```

```
## [1] 11.746475  1.419035  1.003329
```

```
turtles.svd$v
```

```
##           [,1]       [,2]        [,3]
## [1,] 0.5787981 -0.3250273 -0.74789704
## [2,] 0.5779840 -0.4834699  0.65741263
## [3,] 0.5752628  0.8127817  0.09197088
```

```
dim(turtles.svd$u)
```

```
## [1] 48  3
```

Note: co-efficients are mostly equal for all three axes.

```
#Sum of squares for v is 1
sum(turtles.svd$v[,1]^2)
```

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

```
# sum of the d's squares divided by n-1 (48-1) equals 3, number of columns (p)
sum(turtles.svd$d^2) / 47
```

```
## [1] 3
```

```
#if coeff^2/(n-1)=p this mean (n-1)p = sum of soefficients squared
```

7.6.3 Singular value decomposition

TAke all of the u's and add them to a matrix U and all of the v's and add them a matrix V.

$$\mathbf{X} = USV^t, V^tV = \mathbb{I}, U^TU = \mathbb{I}$$

S is the diagonal matrix of the singular values.

$$X_{ij} = u_{i1}s_1v_{1j} + u_{i2}s_2v_{2j} + \ldots + u_{ir}s_rv_{rj}$$

Principal Components

We can create more informative variables by taking the singular value decomposiont coefficients in front of the original variables:

$$Z_1 = c_1\mathbf{X}_{.1} + c_2\mathbf{X}_{.2} + \cdots + c_p\mathbf{X}_{.p}$$

These c's are the $(c_1, c_2, \dots)$ from the usv$v of svd(X). They have decreasing variance

Q 7.18

WRite principal components of turtles data two ways:

```
turtles.svd$d[1] %*% turtles.svd$u[,1]
```

```
##           [,1]      [,2]      [,3]      [,4]       [,5]       [,6]        [,7]
## [1,] -1.983668 -1.705579 -1.340216 -1.420782 -0.9423811 0.04689258 -0.09048395
##          [,8]      [,9]     [,10]     [,11]     [,12]     [,13]    [,14]
## [1,] 0.71721 0.8540019 0.8540019 0.5854404 0.8016959 0.7846503 0.858507
##          [,15]    [,16]    [,17]    [,18]    [,19]    [,20]    [,21]    [,22]
## [1,] 1.353953 1.93447 1.808307 1.989887 2.890979 2.776548 2.907215 3.140809
##          [,23]    [,24]     [,25]     [,26]     [,27]     [,28]     [,29]
## [1,] 3.362087 4.562009 -2.512689 -2.439124 -2.291411 -1.693556 -1.688242
##           [,30]     [,31]     [,32]     [,33]     [,34]     [,35]     [,36]
## [1,] -1.910913 -1.654375 -1.597856 -1.683736 -1.086174 -1.103512 -1.166447
##           [,37]      [,38]      [,39]      [,40]      [,41]     [,42]
## [1,] -0.7219127 -0.8307375 -0.7851402 -0.6374268 -0.8600987 -0.403541
##           [,43]      [,44]         [,45]       [,46]     [,47]     [,48]
## [1,] -0.4211712 -0.1937018 -0.0003910952 -0.01772898 0.1355914 0.8187415
```

```
scaledTurtles %*% turtles.svd$v[,1]
```

```
##                [,1]
##  [1,] -1.9836684602
##  [2,] -1.7055794726
##  [3,] -1.3402164350
##  [4,] -1.4207818191
##  [5,] -0.9423811150
##  [6,]  0.0468925757
##  [7,] -0.0904839545
##  [8,]  0.7172099610
##  [9,]  0.8540018654
## [10,]  0.8540018654
## [11,]  0.5854403531
## [12,]  0.8016958980
## [13,]  0.7846503260
## [14,]  0.8585070441
## [15,]  1.3539533202
## [16,]  1.9344701590
## [17,]  1.8083074735
## [18,]  1.9898872486
## [19,]  2.8909792543
## [20,]  2.7765475313
## [21,]  2.9072153955
```

```
## [22,]  3.1408088253
## [23,]  3.3620866667
## [24,]  4.5620089799
## [25,] -2.5126887623
## [26,] -2.4391243571
## [27,] -2.2914109209
## [28,] -1.6935561972
## [29,] -1.6882415877
## [30,] -1.9109134857
## [31,] -1.6543752488
## [32,] -1.5978564155
## [33,] -1.6837364090
## [34,] -1.0861739982
## [35,] -1.1035118831
## [36,] -1.1664470694
## [37,] -0.7219127043
## [38,] -0.8307375049
## [39,] -0.7851402035
## [40,] -0.6374267672
## [41,] -0.8600986652
## [42,] -0.4035410247
## [43,] -0.4211712224
## [44,] -0.1937018329
## [45,] -0.0003910952
## [46,] -0.0177289800
## [47,]  0.1355913785
## [48,]  0.8187414700
```

7.7 Plotting the observation in the principal plane

```
svda
```

```
## $d
## [1] 7.602846 2.489323
##
## $u
##               [,1]         [,2]
##  [1,] -0.278192883  0.208330831
##  [2,] -0.119546348 -0.058682802
##  [3,] -0.048276377  0.052060965
##  [4,] -0.134040796 -0.035739782
##  [5,]  0.102616972  0.137142217
##  [6,]  0.010019192  0.138453928
##  [7,]  0.006824091 -0.082104836
##  [8,] -0.069359125 -0.369773648
##  [9,] -0.038110681 -0.115471180
## [10,] -0.230624021  0.161144922
## [11,] -0.082863082 -0.328530103
## [12,] -0.045320633 -0.076390140
## [13,]  0.144129137  0.010356503
## [14,] -0.012581552  0.194685316
## [15,]  0.001738280 -0.100692907
```

```
## [16,] -0.058418591 -0.049181987
## [17,] -0.353899433  0.187909578
## [18,] -0.290877089 -0.230624191
## [19,]  0.088908991 -0.029982630
## [20,] -0.204300672 -0.170894230
## [21,]  0.091611698 -0.260024109
## [22,]  0.094186974 -0.212442741
## [23,] -0.122976708  0.233292922
## [24,]  0.142989560 -0.118382146
## [25,]  0.165080354 -0.151730392
## [26,]  0.284717506  0.131176173
## [27,]  0.197699757 -0.055159871
## [28,] -0.100697618  0.224960385
## [29,]  0.217750755  0.003024453
## [30,]  0.073212275  0.218419252
## [31,] -0.095933460  0.291447307
## [32,]  0.205831583  0.073548840
## [33,]  0.458701942  0.179854102
##
## $v
##            [,1]       [,2]
## [1,] -0.7071068  0.7071068
## [2,] -0.7071068 -0.7071068
```

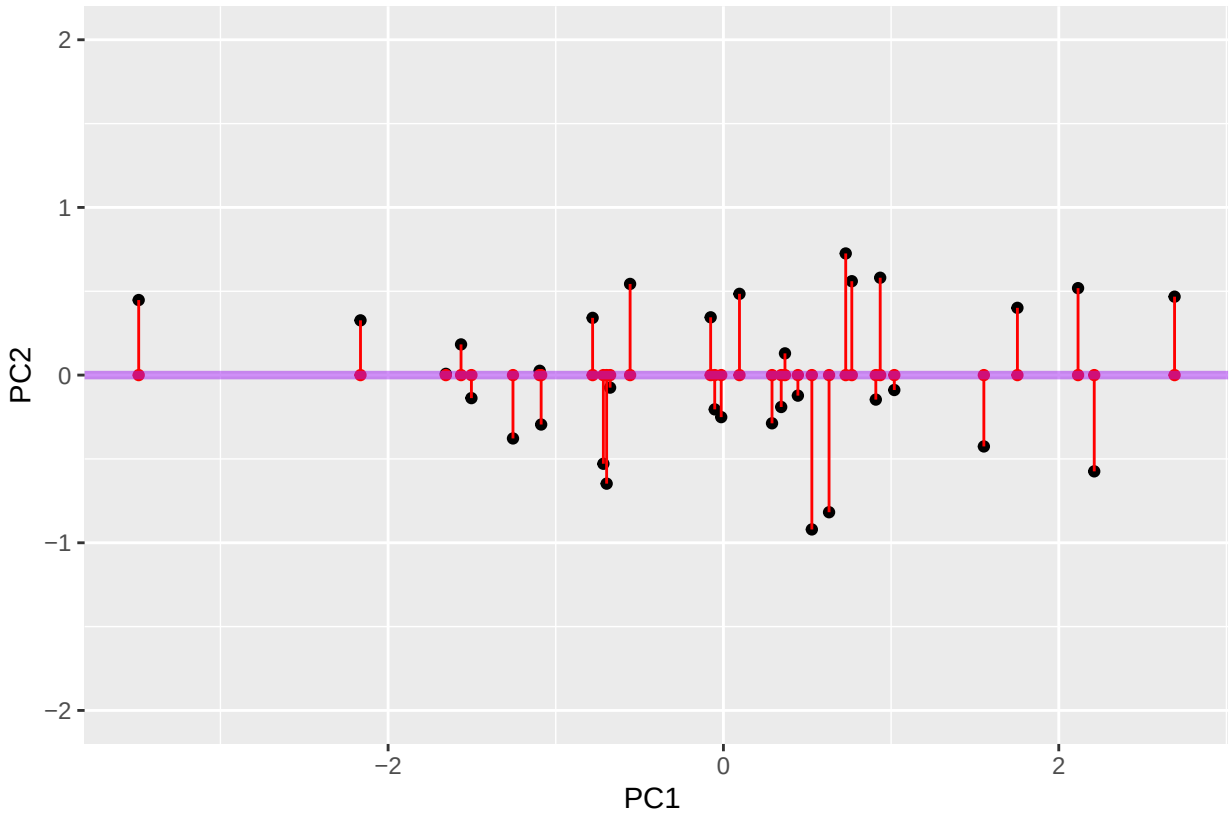
Looking at the v entry from the athletes SVD, we have the first principal component as:

$$Z_1 = -0.707 * \text{athletes}disc - 0.707 * atheletes\text{weight}$$

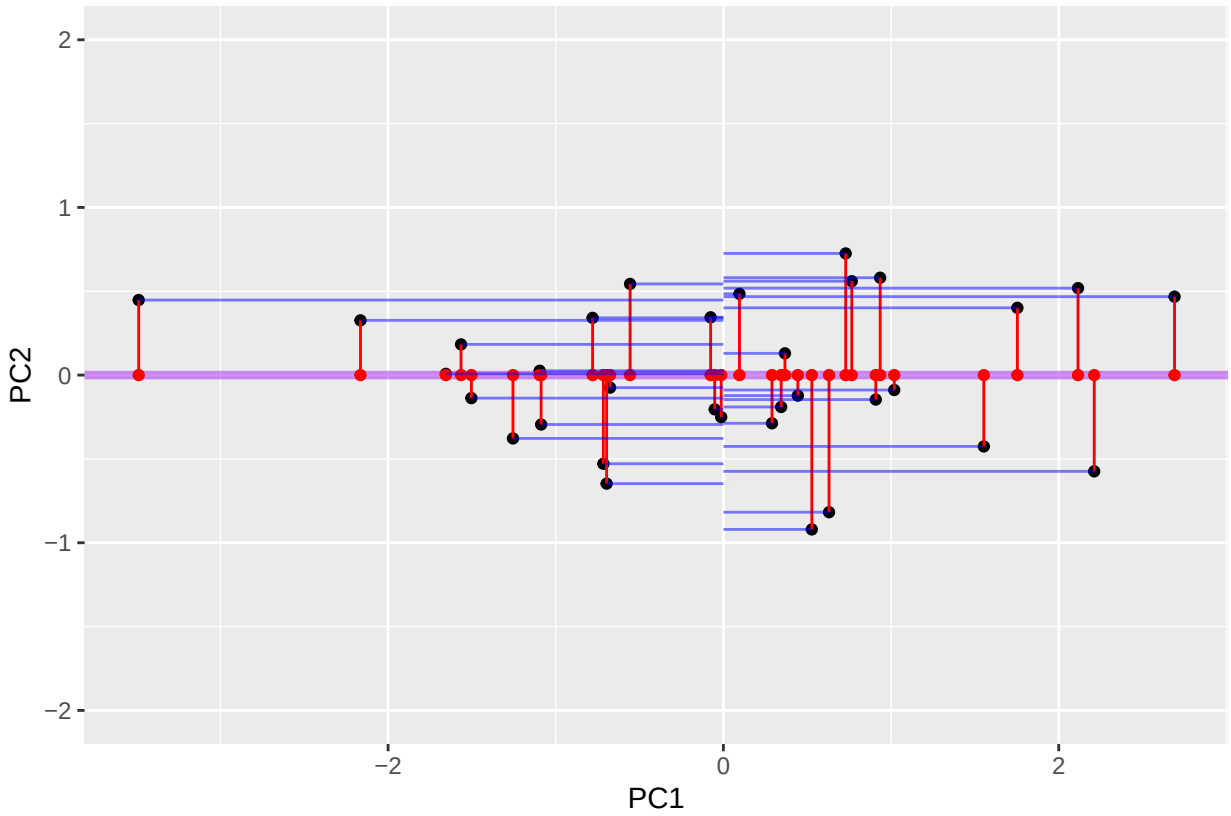
Q 7.19

Rotate our graph so that the purple pca line is horizontal to x-axis we get the first **principal plane**

```
#Find the two Z lines from rescaling using the svd
ppdf = tibble(PC1n = -svda$u[, 1] * svda$d[1],
              PC2n = svda$u[, 2] * svda$d[2])
#Plot those values, add points for x of pc with y=o, add points for y of pc with x=0
ggplot(ppdf, aes(x = PC1n, y = PC2n)) + geom_point() + xlab("PC1 ")+
    ylab("PC2") + geom_point(aes(x=PC1n,y=0),color="red") +
#Add segments of points to both rescaled 0 axes
    geom_segment(aes(xend = PC1n, yend = 0), color = "red") +
    geom_hline(yintercept = 0, color = "purple", lwd=1.5, alpha=0.5) +
    xlim(-3.5, 2.7) + ylim(-2,2) + coord_fixed()
```




```
segm = tibble(xmin = pmin(ppdf$PC1n, 0), xmax = pmax(ppdf$PC1n, 0), yp = seq(-1, -2, length = nrow(ppdf)
ggplot(ppdf, aes(x = PC1n, y = PC2n)) + geom_point() + ylab("PC2") + xlab("PC1") +
    geom_hline(yintercept=0,color="purple",lwd=1.5,alpha=0.5) +
    geom_point(aes(x=PC1n,y=0),color="red")+
    xlim(-3.5, 2.7)+ylim(-2,2)+coord_fixed() +
    geom_segment(aes(xend=PC1n,yend=0), color="red")+
    geom_segment(data=segm,aes(x=xmin,xend=xmax,y=yo,yend=yo), color="blue",alpha=0.5)
```



Note: sum of squares of red segments is square of second singular value. variance of red lines is larger than blue as its the first component. ratio of standard deviation of red to blue is ratio of the singular values

```
sd(ppdf$PC1n)/sd(ppdf$PC2n)
```

```
## [1] 3.054182
```

```
svda$d[1]/svda$d[2]
```

```
## [1] 3.054182
```

Using prcomp:

```
prcomp(athletes[,1:2])
```

```
## Standard deviations (1, .., p=2):
## [1] 1.2407931 0.6785517
##
## Rotation (n x k) = (2 x 2):
##             PC1        PC2
## m100 -0.7071068 -0.7071068
## long  0.7071068 -0.7071068
```

```
svda$v
```

```
##            [,1]       [,2]
## [1,] -0.7071068  0.7071068
## [2,] -0.7071068 -0.7071068
```

7.7.1 PCA of the turtles data

```
cor(scaledTurtles)
```

```
##           length     width    height
## length 1.0000000 0.9783116 0.9646946
## width  0.9783116 1.0000000 0.9605705
## height 0.9646946 0.9605705 1.0000000
```

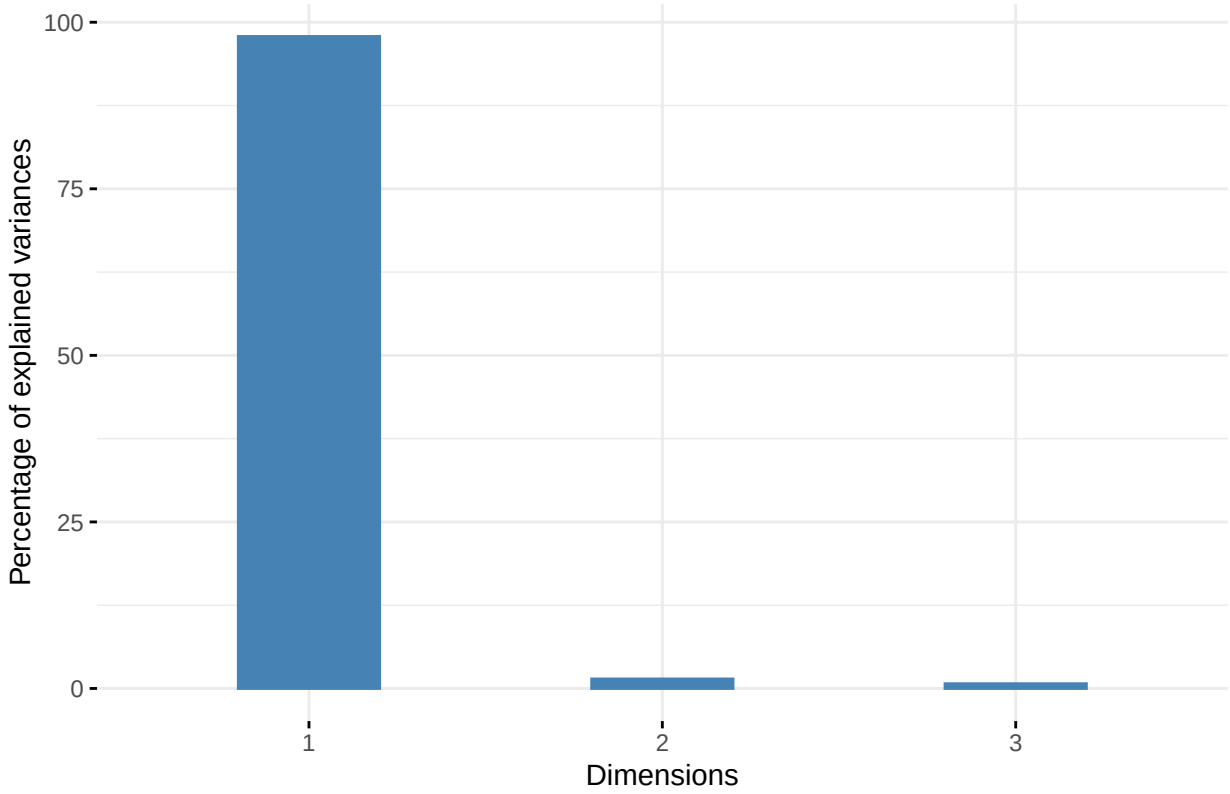
```
pcaturtles = princomp(scaledTurtles)
pcaturtles
```

```
## Call:
## princomp(x = scaledTurtles)
##
## Standard deviations:
##    Comp.1    Comp.2    Comp.3
## 1.6954576 0.2048201 0.1448180
##
##  3  variables and  48 observations.
```

```
library("factoextra")
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```
fviz_eig(pcaturtles, geom = "bar", bar_width = 0.4) + ggtitle("")
```



This shows that one principal components captures most of the variability of the data.

Q 7.21

Three different pca functions developed at various points in time:

```
library(ade4)
```

```
##
## Attaching package: 'ade4'
```

```
## The following object is masked from 'package:GenomicRanges':
##
##     score
```

```
## The following object is masked from 'package:BiocGenerics':
##
##     score
```

```
svd(scaledTurtles)$v[, 1]
```

```
## [1] 0.5787981 0.5779840 0.5752628
```

```
prcomp(turtles[, -1])$rotation[, 1]
```

```
##    length     width    height
## 0.8068646 0.4947448 0.3227958
```

```
princomp(scaledTurtles)$loadings[, 1]
```

```
##    length     width    height
## 0.5787981 0.5779840 0.5752628
```

```
dudi.pca(turtles[, -1], nf = 2, scannf = FALSE)$c1[, 1]
```

```
## [1] -0.5787981 -0.5779840 -0.5752628
```

Without scaling in princomp:

```
princomp(turtles[,-1])$loadings[, 1]
```

```
##    length     width    height
## 0.8068646 0.4947448 0.3227958
```

We see that without scaling first, all of the values in the principle components are different but equal with scaling.

Q 7.22

```
(res = princomp(scaledTurtles))
```

```
## Call:
## princomp(x = scaledTurtles)
##
## Standard deviations:
##    Comp.1    Comp.2    Comp.3
## 1.6954576 0.2048201 0.1448180
##
##  3  variables and  48 observations.
```

```
(PC1 = scaledTurtles %*% res$loadings[,1])
```

```
##                [,1]
##  [1,] -1.9836684602
##  [2,] -1.7055794726
##  [3,] -1.3402164350
##  [4,] -1.4207818191
##  [5,] -0.9423811150
##  [6,]  0.0468925757
##  [7,] -0.0904839545
##  [8,]  0.7172099610
##  [9,]  0.8540018654
## [10,]  0.8540018654
## [11,]  0.5854403531
## [12,]  0.8016958980
## [13,]  0.7846503260
## [14,]  0.8585070441
## [15,]  1.3539533202
## [16,]  1.9344701590
## [17,]  1.8083074735
## [18,]  1.9898872486
## [19,]  2.8909792543
## [20,]  2.7765475313
## [21,]  2.9072153955
## [22,]  3.1408088253
## [23,]  3.3620866667
## [24,]  4.5620089799
## [25,] -2.5126887623
## [26,] -2.4391243571
## [27,] -2.2914109209
## [28,] -1.6935561972
## [29,] -1.6882415877
## [30,] -1.9109134857
## [31,] -1.6543752488
## [32,] -1.5978564155
## [33,] -1.6837364090
## [34,] -1.0861739982
## [35,] -1.1035118831
## [36,] -1.1664470694
## [37,] -0.7219127043
## [38,] -0.8307375049
## [39,] -0.7851402035
## [40,] -0.6374267672
## [41,] -0.8600986652
## [42,] -0.4035410247
## [43,] -0.4211712224
## [44,] -0.1937018329
## [45,] -0.0003910952
## [46,] -0.0177289800
## [47,]  0.1355913785
## [48,]  0.8187414700
```

```r
(sd1 = sqrt(mean(res$scores[, 1]^2)))
```

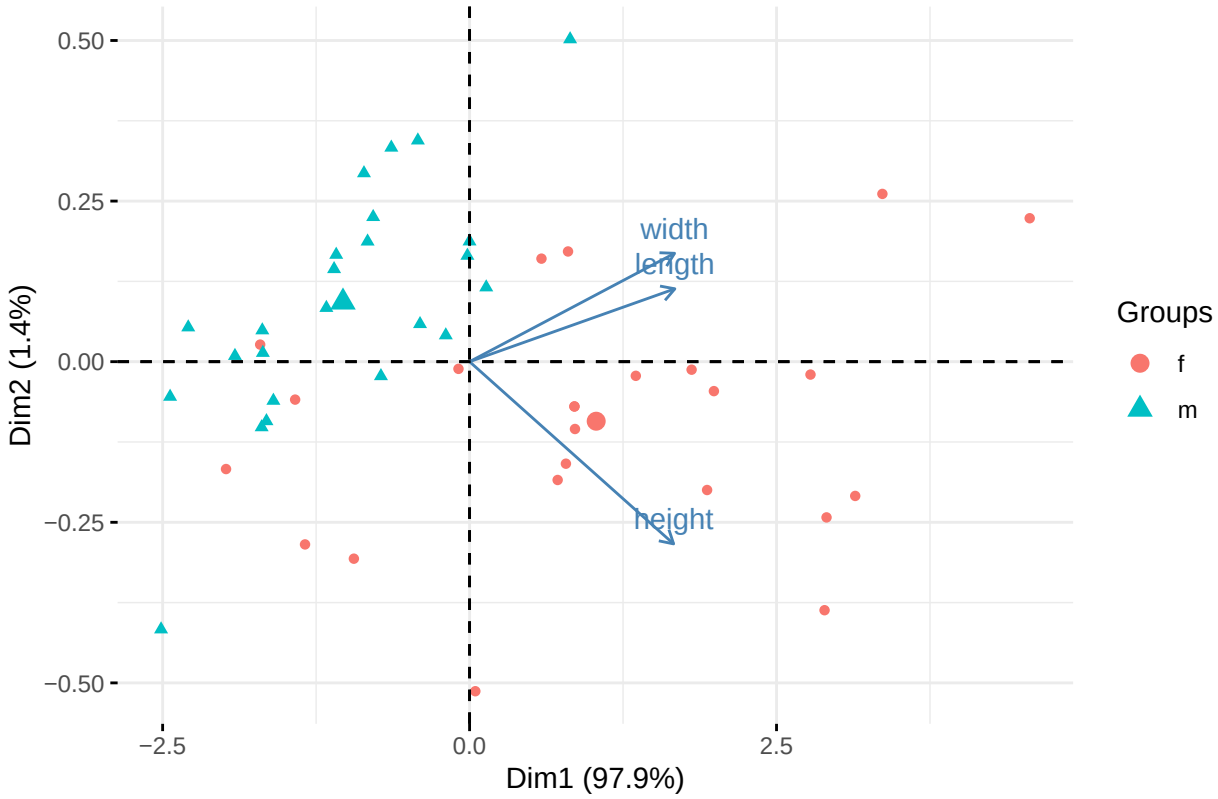
```
## [1] 1.695458
```

```
(res$scores)
```

```
##              Comp.1       Comp.2       Comp.3
##  [1,] -1.9836684602 -0.167152488 -0.134411527
##  [2,] -1.7055794726  0.026616857 -0.107424095
##  [3,] -1.3402164350 -0.284470249 -0.254984022
##  [4,] -1.4207818191 -0.059047170 -0.160036396
##  [5,] -0.9423811150 -0.306657345 -0.161534105
##  [6,]  0.0468925757 -0.512977683  0.076480320
##  [7,] -0.0904839545 -0.011185276 -0.035276987
##  [8,]  0.7172099610 -0.184140239  0.067633478
##  [9,]  0.8540018654 -0.069717059 -0.087956862
## [10,]  0.8540018654 -0.069717059 -0.087956862
## [11,]  0.5854403531  0.160396974  0.085160361
## [12,]  0.8016958980  0.171575254  0.043506318
## [13,]  0.7846503260 -0.158804367  0.265559143
## [14,]  0.8585070441 -0.104794074  0.250211250
## [15,]  1.3539533202 -0.022024628  0.026660717
## [16,]  1.9344701590 -0.199755276  0.046330670
## [17,]  1.8083074735 -0.012473257  0.193141742
## [18,]  1.9898872486 -0.045838632  0.328245701
## [19,]  2.8909792543 -0.386867832 -0.090338572
## [20,]  2.7765475313 -0.020058792 -0.161190690
## [21,]  2.9072153955 -0.242417826  0.030166350
## [22,]  3.1408088253 -0.208967720 -0.099866694
## [23,]  3.3620866667  0.261170953 -0.279584195
## [24,]  4.5620089799  0.223284088 -0.212508003
## [25,] -2.5126887623 -0.416643767  0.057013087
## [26,] -2.4391243571 -0.054525680 -0.092008629
## [27,] -2.2914109209  0.053494906 -0.122704414
## [28,] -1.6935561972 -0.101963915 -0.191413462
## [29,] -1.6882415877  0.048888684 -0.195803096
## [30,] -1.9109134857  0.009035984  0.059124503
## [31,] -1.6543752488 -0.092497277 -0.030003352
## [32,] -1.5978564155 -0.060758811  0.043027757
## [33,] -1.6837364090  0.013811669  0.142365016
## [34,] -1.0861739982  0.166460641 -0.060017856
## [35,] -1.1035118831  0.144188814  0.028361145
## [36,] -1.1664470694  0.083775927  0.168603593
## [37,] -0.7219127043 -0.022448287  0.001306140
## [38,] -0.8307375049  0.187105560  0.059738211
## [39,] -0.7851402035  0.225246620  0.007874765
## [40,] -0.6374267672  0.333267206 -0.022821020
## [41,] -0.8600986652  0.293414505  0.232106579
## [42,] -0.4035410247  0.058609519 -0.019180241
## [43,] -0.4211712224  0.344445485 -0.064475063
## [44,] -0.1937018329  0.041113377  0.152439273
## [45,] -0.0003910952  0.187275023  0.069880042
## [46,] -0.0177289800  0.165003196  0.158259043
## [47,]  0.1355913785  0.115768588  0.256847448
## [48,]  0.8187414700  0.501954874 -0.178546507
```

Q 7.23

```
fviz_pca_biplot(pcaturtles, label = "var", habillage = turtles[, 1]) +
  ggtitle("")
```



Q 7.24

First component (“x”) always has largest variance so the pca plot is always wider than it is tall (“y”)

Q 7.25

Females tend to be taller. Width and length are highly correlatd with one another as their axes are very similar

Q 7.26

COmpare the variance of each new coordinate to the eigenvalues returned by the PCA dudi.pca function

```
pcadudit = dudi.pca(scaledTurtles, nf = 2, scannf = FALSE)
apply(pcadudit$li, 2, function(x) sum(x^2)/48)
```

```
##      Axis1      Axis2
## 2.93573765 0.04284387
```

```
pcadudit$eig
```

```
## [1] 2.93573765 0.04284387 0.02141848
```

The sums of squares of each new coordiate are similar to the eigenvalues.

REmember:

- Each principal component has variance measured by the eigenvalue which is the square of the singular value
- New variable are always orthogonal, centered, uncorrelated, hence independent
- When scaled, sum of the variances equals number of variables. (sum of variances is sum of diagonal of cross product matrix, i.e. eigenvalues are the diagonal of the cross product matrix)
- principal components are ordered by variances (i.e. eignenvalues)

7.7.2 A complete analysis: the decathlon athletes:

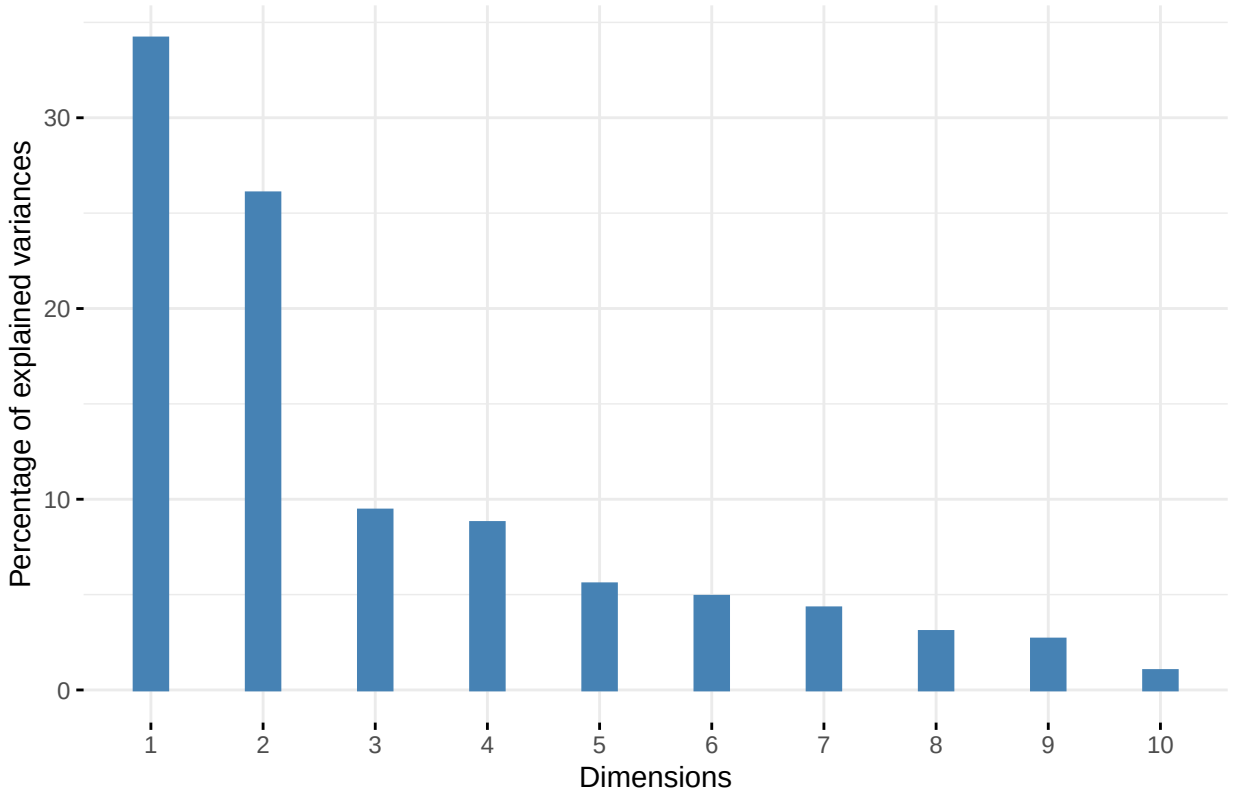
```
cor(athletes) %>% round(1)
```

```
##        m100 long weight highj m400 m110 disc pole javel m1500
## m100    1.0 -0.5   -0.2  -0.1  0.6  0.6  0.0 -0.4  -0.1   0.3
## long   -0.5  1.0    0.1   0.3 -0.5 -0.5  0.0  0.3   0.2  -0.4
## weight -0.2  0.1    1.0   0.1  0.1 -0.3  0.8  0.5   0.6   0.3
## highj  -0.1  0.3    0.1   1.0 -0.1 -0.3  0.1  0.2   0.1  -0.1
## m400    0.6 -0.5    0.1  -0.1  1.0  0.5  0.1 -0.3   0.1   0.6
## m110    0.6 -0.5   -0.3  -0.3  0.5  1.0 -0.1 -0.5  -0.1   0.1
## disc    0.0  0.0    0.8   0.1  0.1 -0.1  1.0  0.3   0.4   0.4
## pole   -0.4  0.3    0.5   0.2 -0.3 -0.5  0.3  1.0   0.3   0.0
## javel  -0.1  0.2    0.6   0.1  0.1 -0.1  0.4  0.3   1.0   0.1
## m1500   0.3 -0.4    0.3  -0.1  0.6  0.1  0.4  0.0   0.1   1.0
```

```
#Scree plot to choose k
pca.ath = dudi.pca(athletes, scannf = FALSE)
pca.ath$eig
```

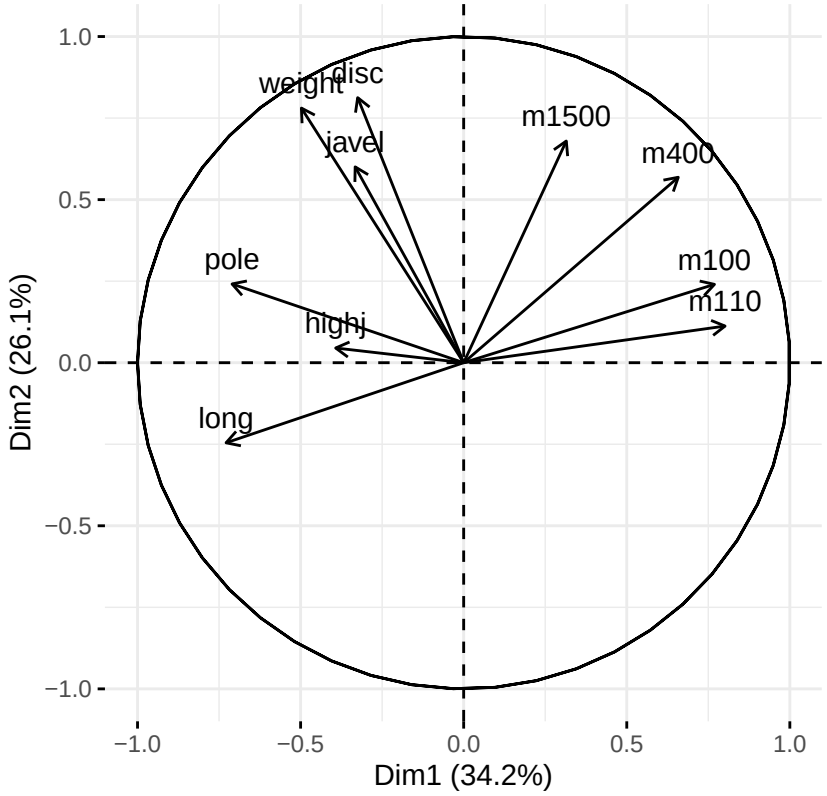
```
##  [1] 3.4182381 2.6063931 0.9432964 0.8780212 0.5566267 0.4912275 0.4305952
##  [8] 0.3067981 0.2669494 0.1018542
```

```
fviz_eig(pca.ath, geom = "bar", bar_width = 0.3) + ggtitle("")
```



Note this drops off around 2 so that's likely a good number of components

```
fviz_pca_var(pca.ath, col.circle = "black") + ggtitle("")
```



Variables projected on two new axes. Space between denotes correlation. The opposite correlation is because some are running and some are

throwing/jumping. The best athletes throw/jump the best (which is high) but run the best (which is low times)

Q 7.28

Change the signs on the running events to get a positive correlation:

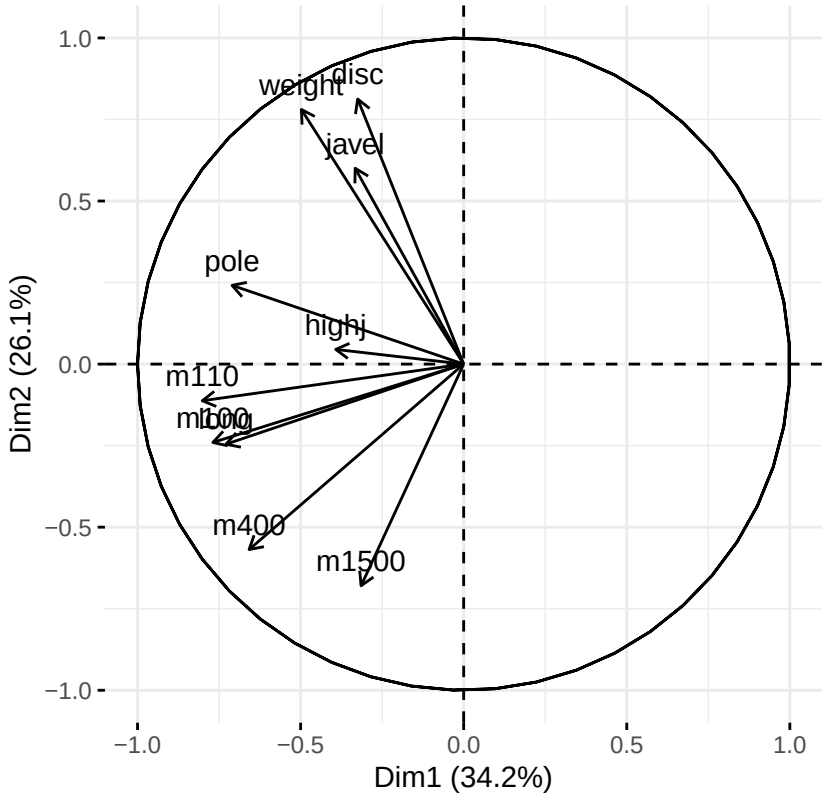
```
athletes[, c(1, 5, 6, 10)] = -athletes[, c(1, 5, 6, 10)]
cor(athletes) %>% round(1)
```

```
##        m100 long weight highj m400 m110 disc pole javel m1500
## m100    1.0  0.5    0.2   0.1  0.6  0.6  0.0  0.4   0.1   0.3
## long    0.5  1.0    0.1   0.3  0.5  0.5  0.0  0.3   0.2   0.4
## weight  0.2  0.1    1.0   0.1 -0.1  0.3  0.8  0.5   0.6  -0.3
## highj   0.1  0.3    0.1   1.0  0.1  0.3  0.1  0.2   0.1   0.1
## m400    0.6  0.5   -0.1   0.1  1.0  0.5 -0.1  0.3  -0.1   0.6
## m110    0.6  0.5    0.3   0.3  0.5  1.0  0.1  0.5   0.1   0.1
## disc    0.0  0.0    0.8   0.1 -0.1  0.1  1.0  0.3   0.4  -0.4
## pole    0.4  0.3    0.5   0.2  0.3  0.5  0.3  1.0   0.3   0.0
## javel   0.1  0.2    0.6   0.1 -0.1  0.1  0.4  0.3   1.0  -0.1
## m1500   0.3  0.4   -0.3   0.1  0.6  0.1 -0.4  0.0  -0.1   1.0
```

```
pcan.ath = dudi.pca(athletes, nf = 2, scannf = FALSE)
pcan.ath$eig
```

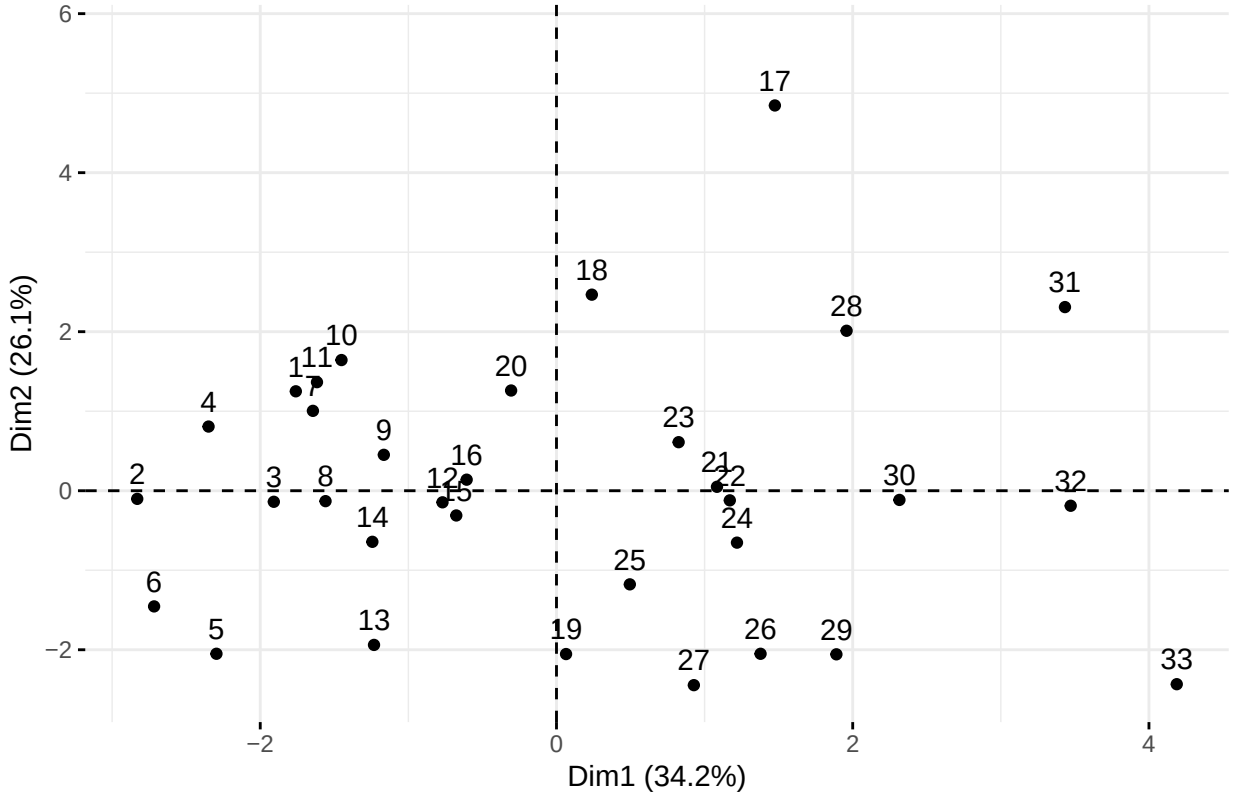
```
##  [1] 3.4182381 2.6063931 0.9432964 0.8780212 0.5566267 0.4912275 0.4305952
##  [8] 0.3067981 0.2669494 0.1018542
```

```
fviz_pca_var(pcan.ath, col.circle="black") + ggtitle("")
```



Plot athletes projected onto first principal plane:

```
fviz_pca_ind(pcan.ath) + ggtitle("") + ylim(c(-2.5,5.7))
```



The best athletes are on the far right.

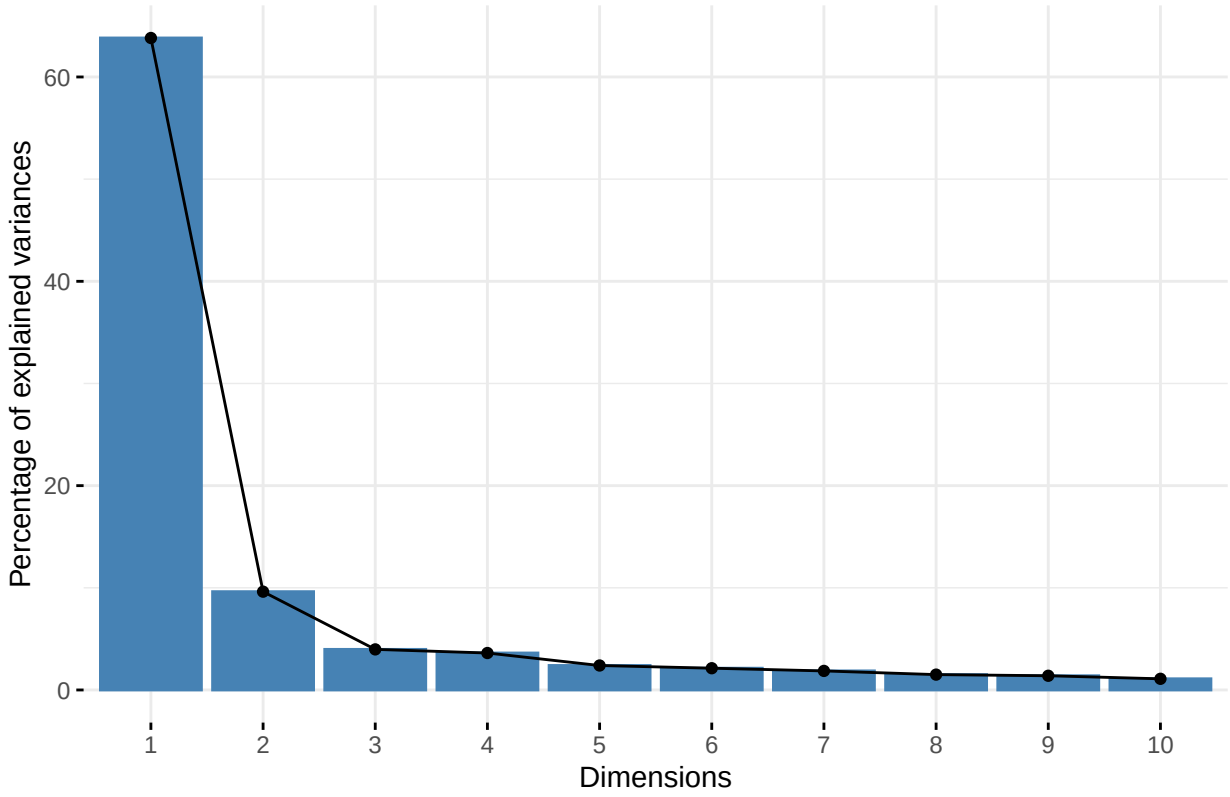
Project the olympic decatholon score on to the principal component:

```
data("olympic", package = "ade4")
olympic$score
```

```
##  [1] 8488 8399 8328 8306 8286 8272 8216 8189 8180 8167 8143 8114 8093 8083 8036
## [16] 8021 7869 7860 7859 7781 7753 7745 7743 7623 7579 7517 7505 7422 7310 7237
## [31] 7231 7016 6907
```

7.8 PCA as an exploratory tool: using extra information

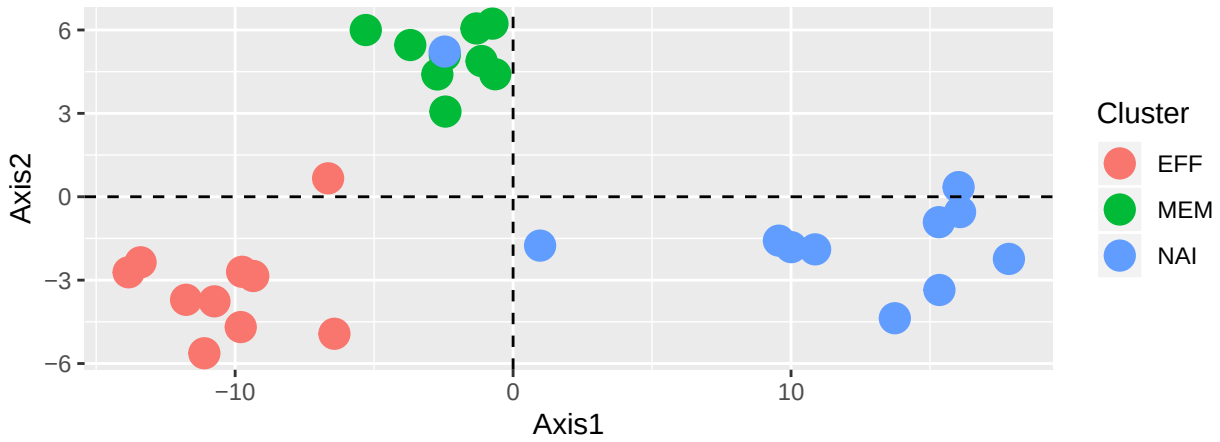
```
pcaMsig3 = dudi.pca(Msig3transp, center = TRUE, scale = TRUE,
                    scannf = FALSE, nf = 4)
#One principal component seems sufficient
fviz_screeplot(pcaMsig3) + ggtitle("")
```




```
ids = rownames(Msig3transp)
celltypes = factor(substr(ids, 7, 9))
status = factor(substr(ids, 1, 3))
table(celltypes)
```

```
## celltypes
## EFF MEM NAI
##  10   9  11
```

```
cbind(pcaMsig3$li, tibble(Cluster = celltypes, sample = ids)) %>%
ggplot(aes(x = Axis1, y = Axis2)) +
  geom_point(aes(color = Cluster), size = 5) +
  geom_hline(yintercept = 0, linetype = 2) +
  geom_vline(xintercept = 0, linetype = 2) +
  scale_color_discrete(name = "Cluster") + coord_fixed()
```



7.8.1 Mass Spectroscopy Data Analysis

```
library(xcms)
```

```
## Loading required package: MSnbase
```

```
## Loading required package: mzR
```

```
## Warning in fun(libname, pkgname): mzR has been built against a different Rcpp version (1.0.2)
## than is installed on your system (1.0.3). This might lead to errors
## when loading mzR. If you encounter such issues, please send a report,
## including the output of sessionInfo() to the Bioc support forum at
## https://support.bioconductor.org/. For details see also
## https://github.com/sneumann/mzR/wiki/mzR-Rcpp-compiler-linker-issue.
```

```
## Loading required package: ProtGenerics
```

```
##
## Attaching package: 'ProtGenerics'
```

```
## The following object is masked from 'package:stats':
##
##     smooth
```

```
##
## This is MSnbase version 2.12.0
##   Visit https://lgatto.github.io/MSnbase/ to get started.
```

```
##
## Attaching package: 'MSnbase'
```

```
## The following object is masked from 'package:base':
##
##     trimws
```

```
##
## This is xcms version 3.8.1
```

```
##
## Attaching package: 'xcms'
```

```
## The following object is masked from 'package:SummarizedExperiment':
##
##     distance
```

```
## The following object is masked from 'package:GenomicRanges':
##
##     distance
```

```
## The following object is masked from 'package:IRanges':
##
##     distance
```

```
## The following object is masked from 'package:phyloseq':
##
##     distance
```

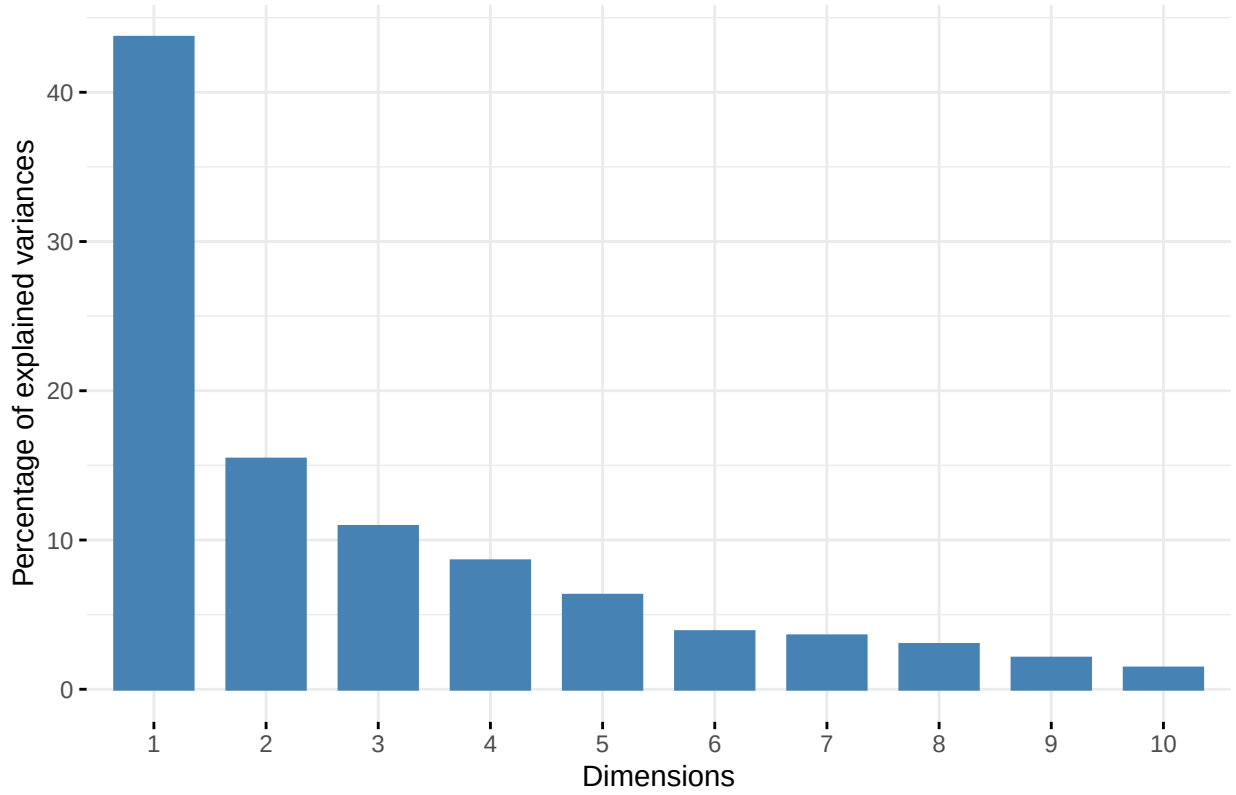
```
## The following objects are masked from 'package:dplyr':
##
##     collect, groups
```

```
## The following object is masked from 'package:stats':
##
##     sigma
```

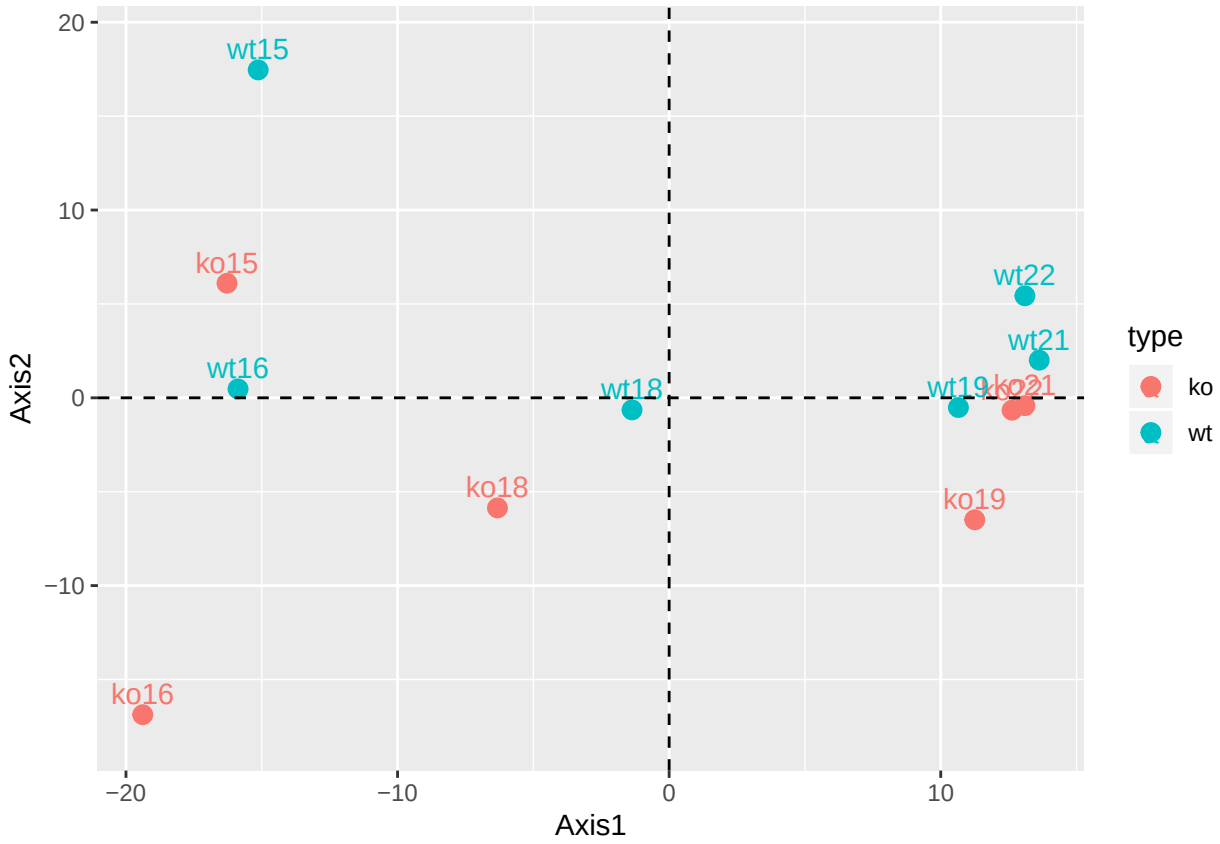
```
load(here("data","mat1xcms.RData"))
dim(mat1)
```

```
## [1] 399  12
```

```
pcamat1 = dudi.pca(t(mat1), scannf = FALSE, nf = 3)
fviz_eig(pcamat1, geom = "bar", bar_width = 0.7) + ggtitle("")
```

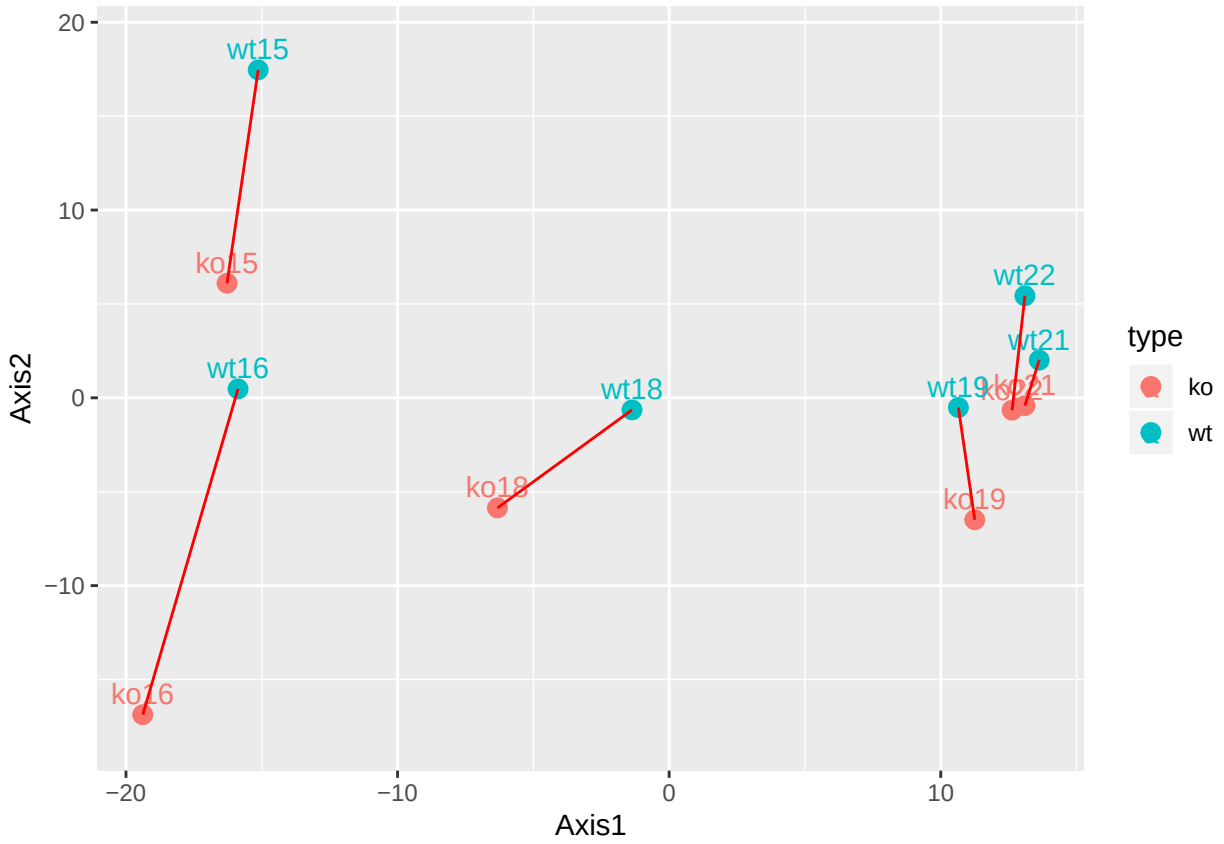



```
#use one principal component
dfmat1 = cbind(pcamat1$li, tibble(
    label = rownames(pcamat1$li),
    number = substr(label, 3, 4),
    type = factor(substr(label, 1, 2))))
pcsplot = ggplot(dfmat1,
  aes(x=Axis1, y=Axis2, label=label, group=number, colour=type)) +
 geom_text(size = 4, vjust = -0.5)+ geom_point(size = 3)+ylim(c(-18,19))
pcsplot + geom_hline(yintercept = 0, linetype = 2) +
  geom_vline(xintercept = 0, linetype = 2)
```



Knockouts are below their wildlife type, not random scattering.

```
pcsplot + geom_line(colour = "red")
```

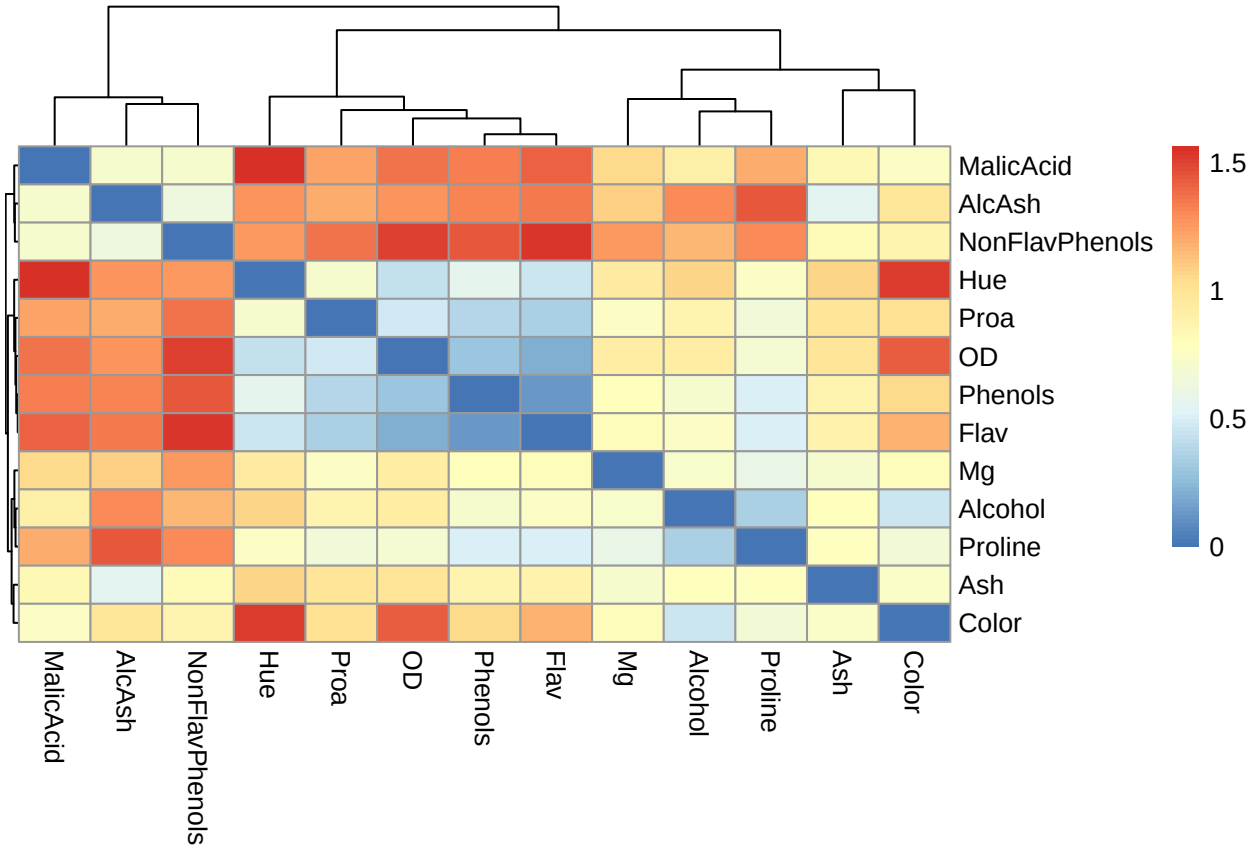


7.8.2 Biplots and scaling

```
library("pheatmap")
load(here("data", "wine.RData"))
load(here("data", "wineClass.RData"))
wine[1:2, 1:7]
```

```
##   Alcohol MalicAcid  Ash AlcAsh  Mg Phenols Flav
## 1   14.23      1.71 2.43   15.6 127    2.80 3.06
## 2   13.20      1.78 2.14   11.2 100    2.65 2.76
```

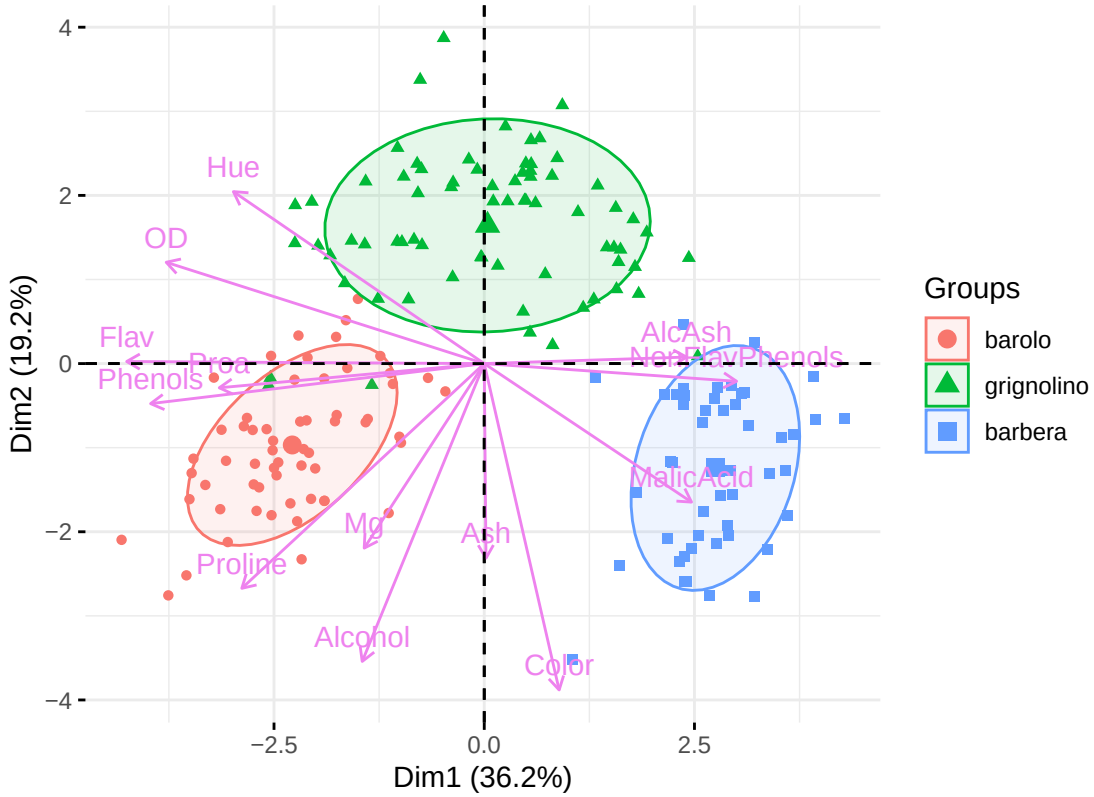
```
pheatmap(1 - cor(wine), treeheight_row = 0.2)
```




```
winePCAd = dudi.pca(wine, scannf=FALSE)
table(wine.class)
```

```
## wine.class
##     barolo grignolino    barbera
##         59         71         48
```

```
fviz_pca_biplot(winePCAd, geom = "point", habillage = wine.class,
   col.var = "violet", addEllipses = TRUE, ellipse.level = 0.69) +
   ggtitle("") + coord_fixed()
```



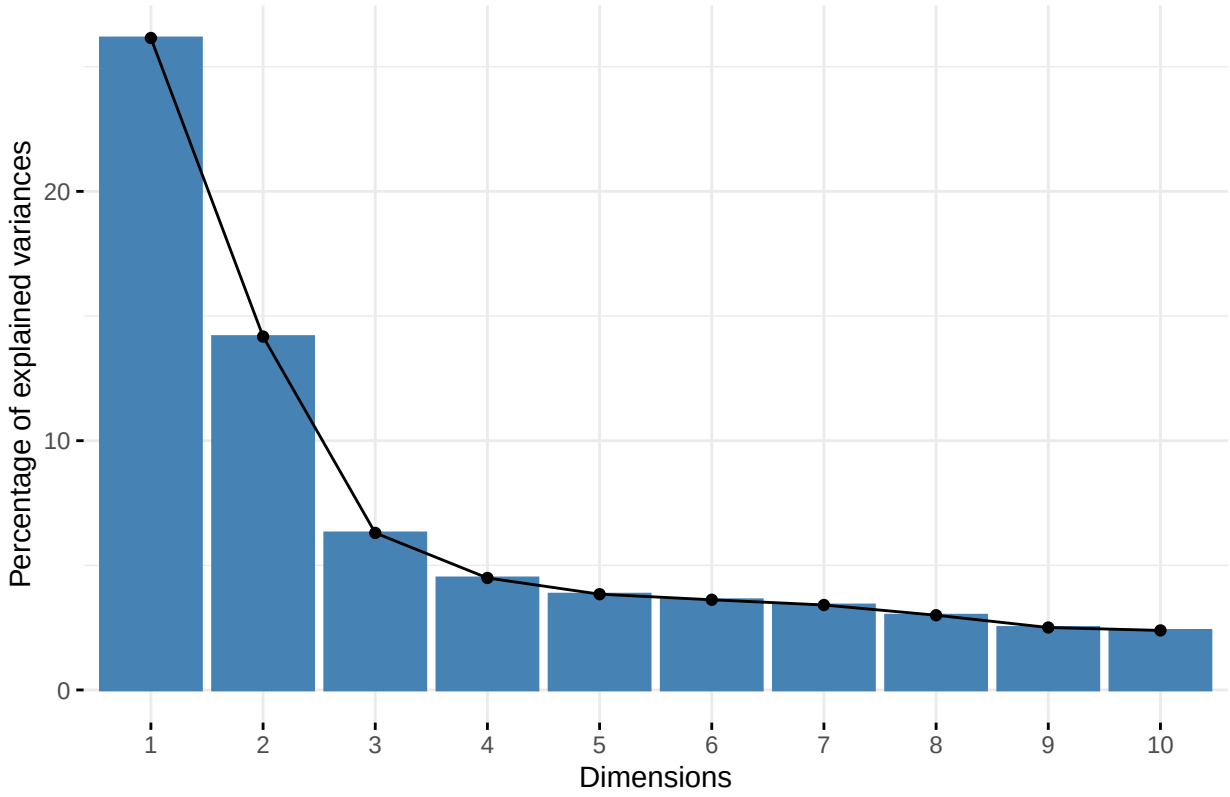
ALERT!!: Hue and alcohol uncorrelated. Does this mean that we are looking at 90 degrees equals uncorrelated? What does 180 degrees mean?

7.8.3 An example of weighted PCA

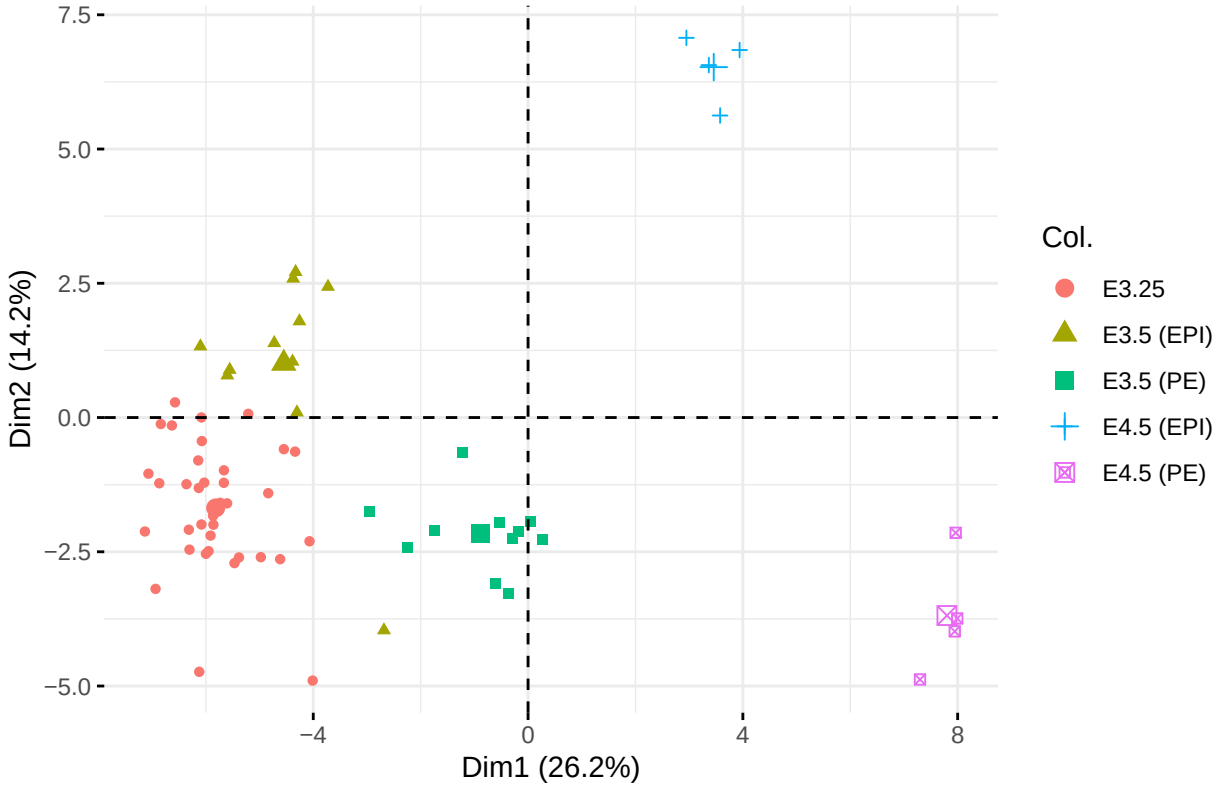
```
data("x", package = "Hiiragi2013")
xwt = x[, x$genotype == "WT"]
sel = order(rowVars(Biobase::exprs(xwt)), decreasing = TRUE)[1:100]
xwt = xwt[sel, ]
tab = table(xwt$sampleGroup)
tab
```

```
##
##      E3.25 E3.5 (EPI)  E3.5 (PE) E4.5 (EPI)  E4.5 (PE)
##         36         11         11          4          4
```

```
xwt$weight = 1 / as.numeric(tab[xwt$sampleGroup])
pcaMouse = dudi.pca(as.data.frame(t(Biobase::exprs(xwt))),
  row.w = xwt$weight,
  center = TRUE, scale = TRUE, nf = 2, scannf = FALSE)
fviz_eig(pcaMouse) + ggtitle("")
```




```
fviz_pca_ind(pcaMouse, geom = "point", col.ind = xwt$sampleGroup) +
  ggtitle("") + coord_fixed()
```



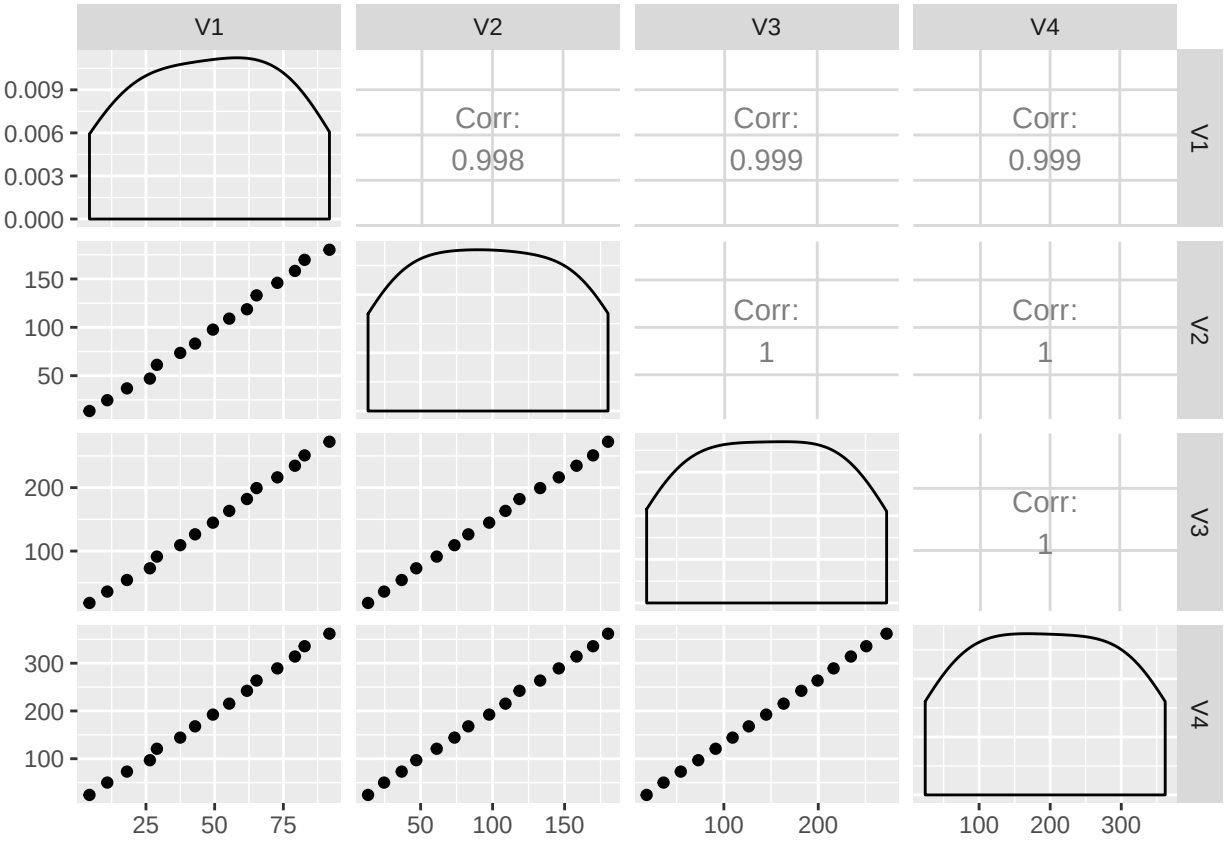
7.11 Exercises

exercise 7.1

```
u=seq(2, 30 , by = 2)
v=seq(3, 12, by = 3)
X1 = u %*% t(v)
Materr = matrix(rnorm(60,1), nrow = 15, ncol = 4)
X = X1 + Materr
```

ALERT!!: How am I supposed to visualize this?

```
ggpairs(as.data.frame(X))
```



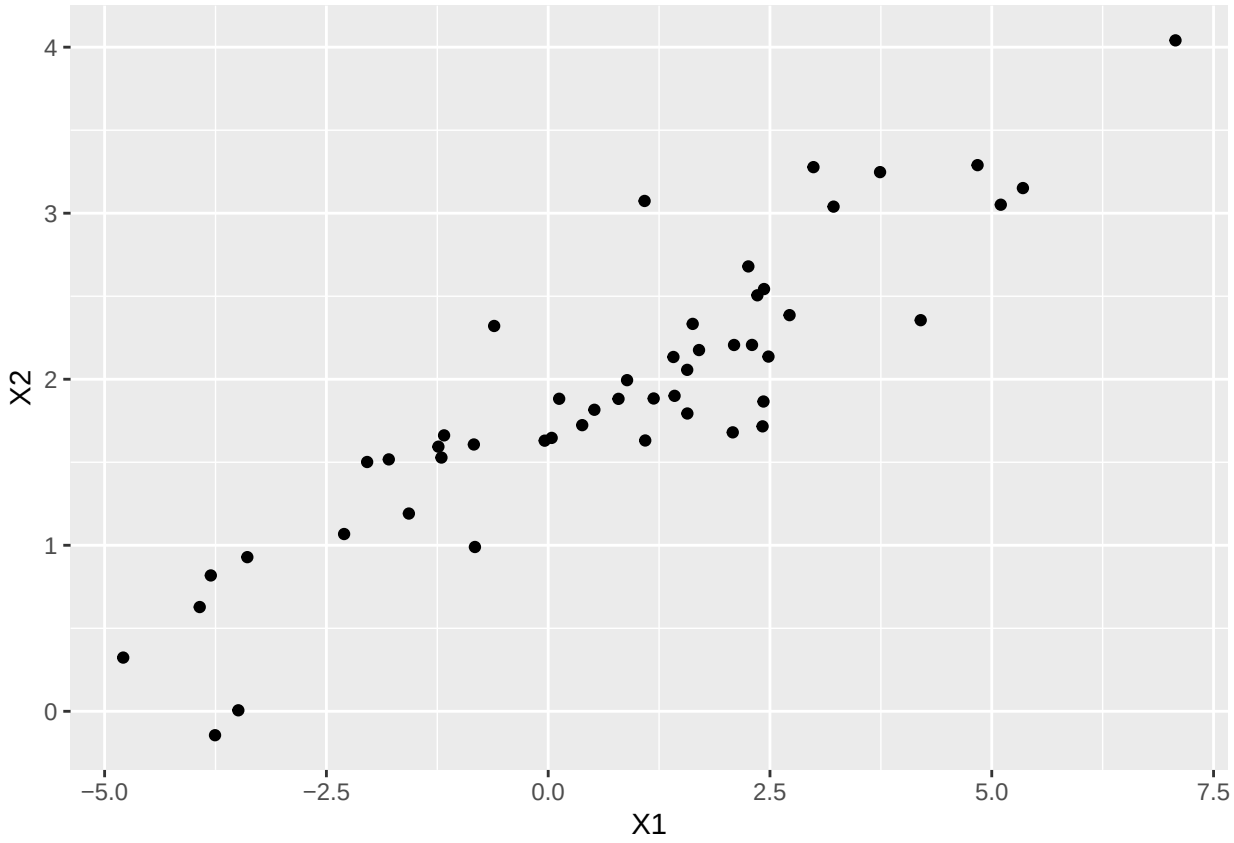
exercise 7.2

```
library(rockchalk)
```

```
##
## Attaching package: 'rockchalk'
```

```
## The following object is masked from 'package:dplyr':
##
##     summarize
```

```
mu1 = 1; mu2 = 2; s1=2.5; s2=0.8; rho=0.9;
sigma = matrix(c(s1^2, s1*s2*rho, s1*s2*rho, s2^2),2)
sim2d = data.frame(mvrnorm(50, mu = c(mu1,mu2), Sigma = sigma))
ggplot(data.frame(sim2d),aes(x=X1,y=X2)) +
  geom_point()
```




```
svdn<-svd(scale(sim2d))
svdn$d
```

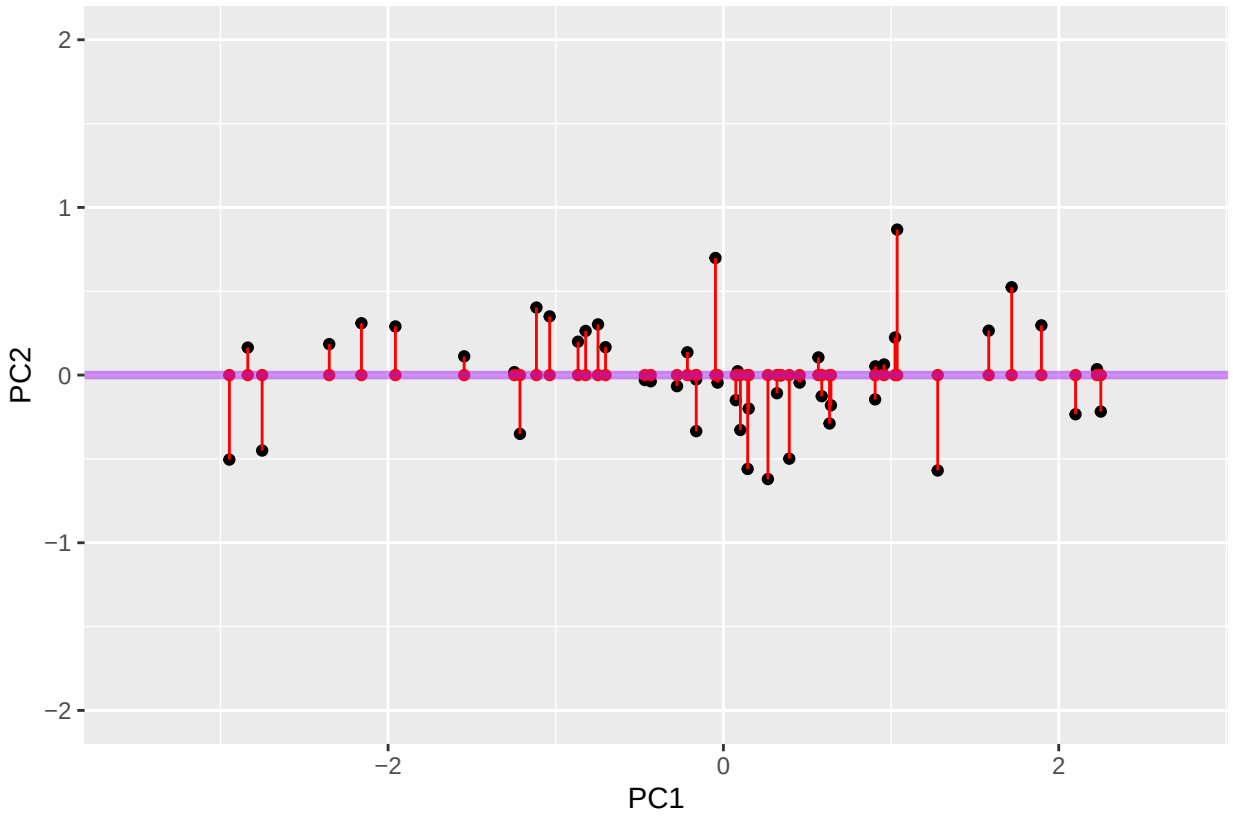
```
## [1] 9.646650 2.223092
```

```
ppdfn =tibble(PC1n =-svdn$u[, 1]*svdn$d[1],PC2n = svdn$u[, 2]*svdn$d[2])
#Plot those values, add points for x of pc with y=o, add points for y of pc with x=0
ggplot(ppdfn,aes(x = PC1n, y = PC2n))+ geom_point()+ xlab("PC1")+ ylab("PC2")+ geom_point(aes(x=PC1n,y=0
#Add segments of points to both rescaled 0 axes
geom_segment(aes(xend = PC1n, yend = 0), color = "red")+geom_hline(yintercept = 0, color = "purple", lwd
```

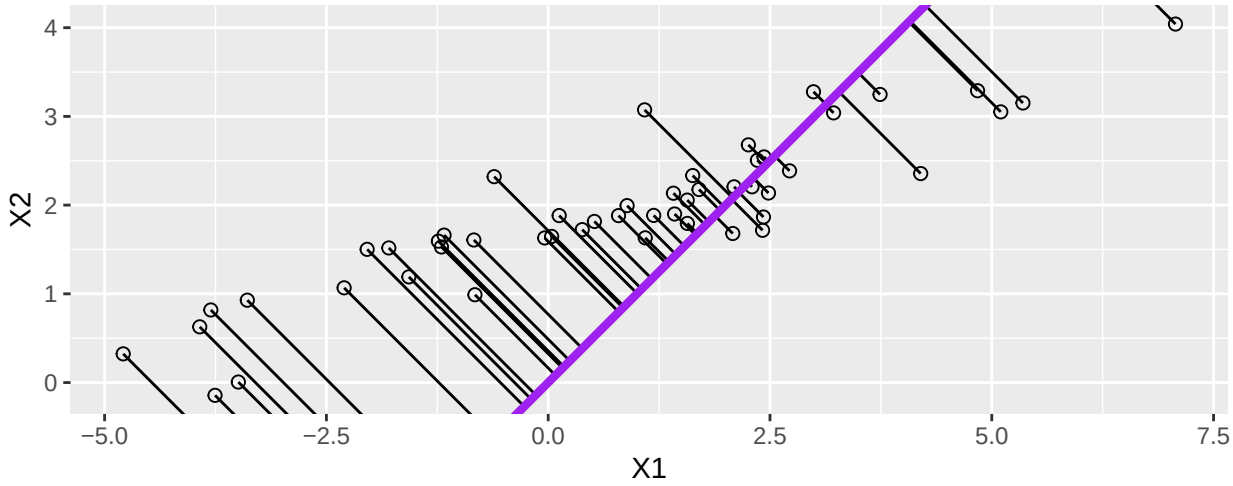
```
## Warning: Removed 1 rows containing missing values (geom_point).
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

```
## Warning: Removed 1 rows containing missing values (geom_segment).
```




```
#Not rotated to 0
pc = as.matrix(sim2d) %*% svdn$v[, 1] %*% t(svdn$v[, 1])
bp = svdn$v[2, 1] / svdn$v[1, 1]
ap = mean(pc[, 2]) - bp * mean(pc[, 1])
ggplot(data.frame(sim2d),aes(x=X1,y=X2)) +
  geom_point(size = 2, shape = 21) + geom_segment(xend = pc[, 1], yend = pc[, 2]) +
  geom_abline(intercept = ap, slope = bp, col = "purple", lwd = 1.5) + coord_fixed()
```



exercise 7.3

```
svdn<-svd(scale(sim2d))
svdn$d
```

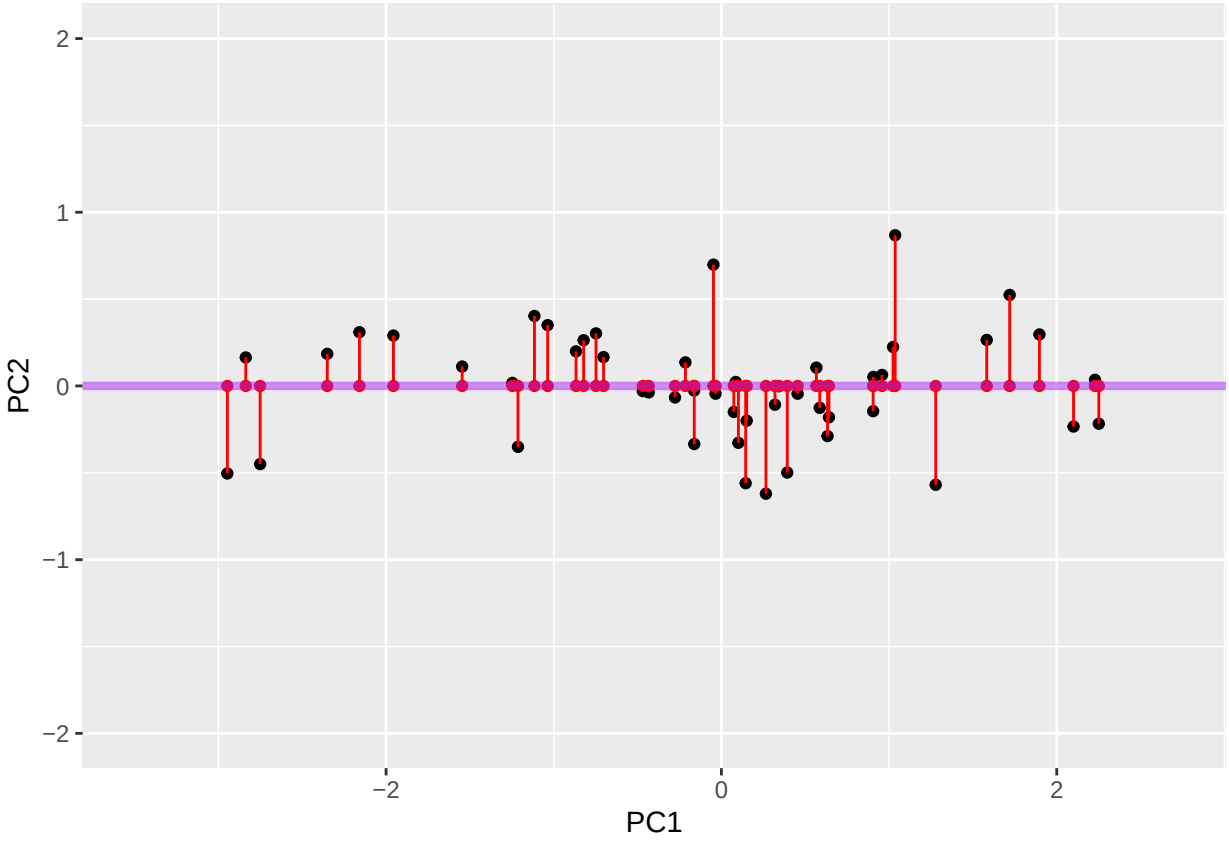
```
## [1] 9.646650 2.223092
```

```
ppdfn =tibble(PC1n =-svdn$u[, 1]*svdn$d[1],PC2n = svdn$u[, 2]*svdn$d[2])
#Plot those values, add points for x of pc with y=o, add points for y of pc with x=0
ggplot(ppdfn,aes(x = PC1n, y = PC2n))+ geom_point()+ xlab("PC1")+ ylab("PC2")+ geom_point(aes(x=PC1n,y=0
#Add segments of points to both rescaled 0 axes
geom_segment(aes(xend = PC1n, yend = 0), color = "red")+geom_hline(yintercept = 0, color = "purple", lwd
```

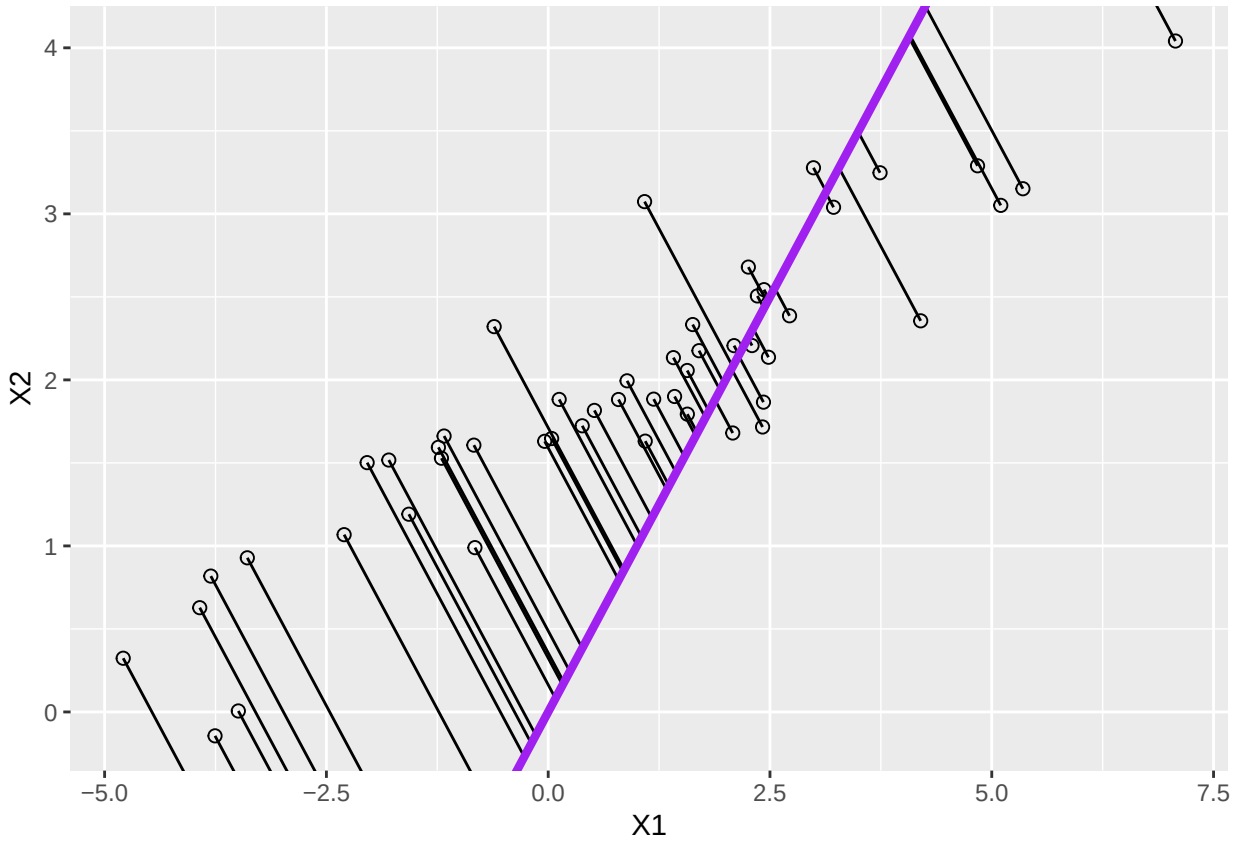
```
## Warning: Removed 1 rows containing missing values (geom_point).
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

```
## Warning: Removed 1 rows containing missing values (geom_segment).
```

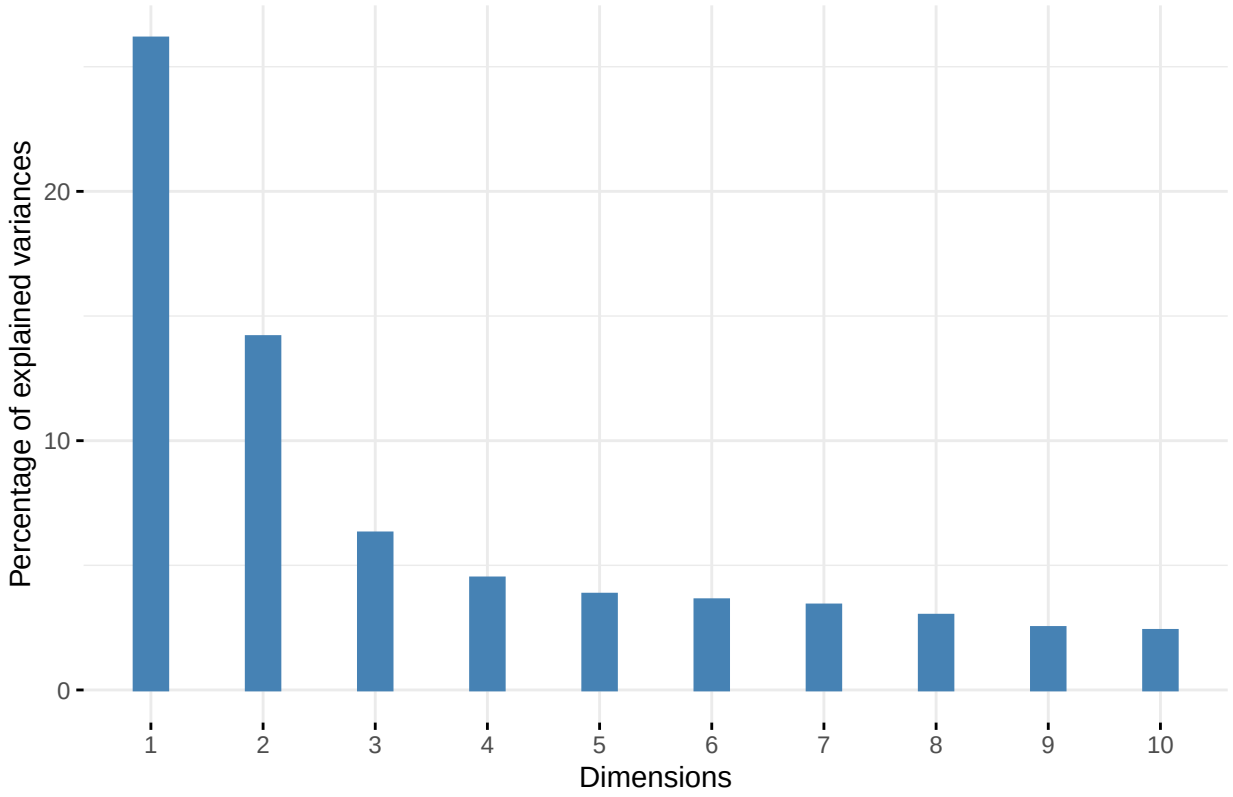



```
#Not rotated to 0
pc = as.matrix(sim2d) %*% svdn$v[, 1] %*% t(svdn$v[, 1])
bp = svdn$v[2, 1] / svdn$v[1, 1]
ap = mean(pc[, 2]) - bp * mean(pc[, 1])
ggplot(data.frame(sim2d),aes(x=X1,y=X2)) +
  geom_point(size = 2, shape = 21) + geom_segment(xend = pc[, 1], yend = pc[, 2]) +
  geom_abline(intercept = ap, slope = bp, col = "purple", lwd = 1.5)
```

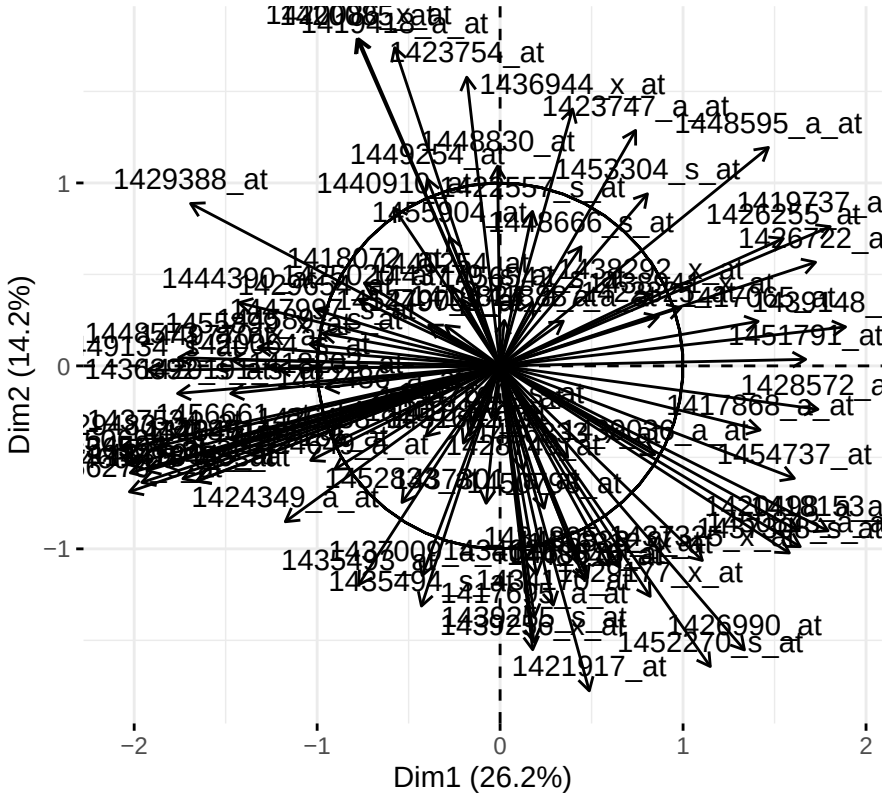


exercise 7.4

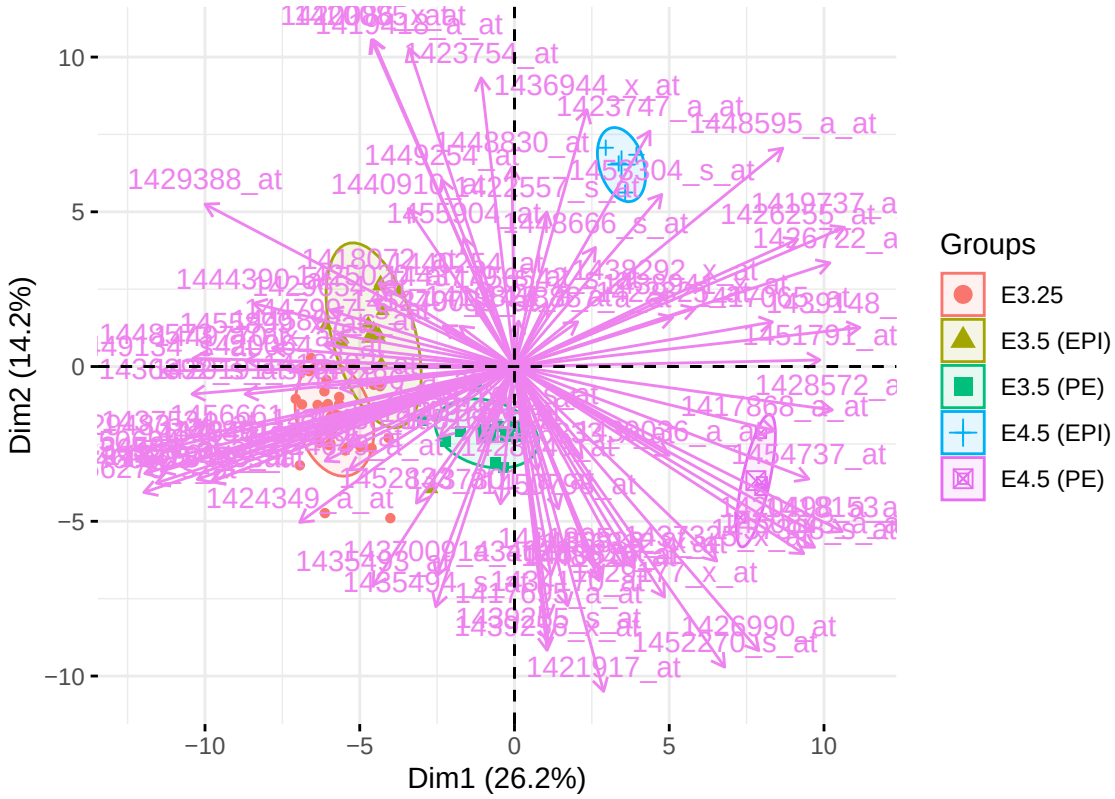
```
fviz_eig(pcaMouse, geom = "bar", bar_width = 0.3) + ggtitle("")
```




```
fviz_pca_var(pcaMouse, col.circle = "black") + ggtitle("")
```

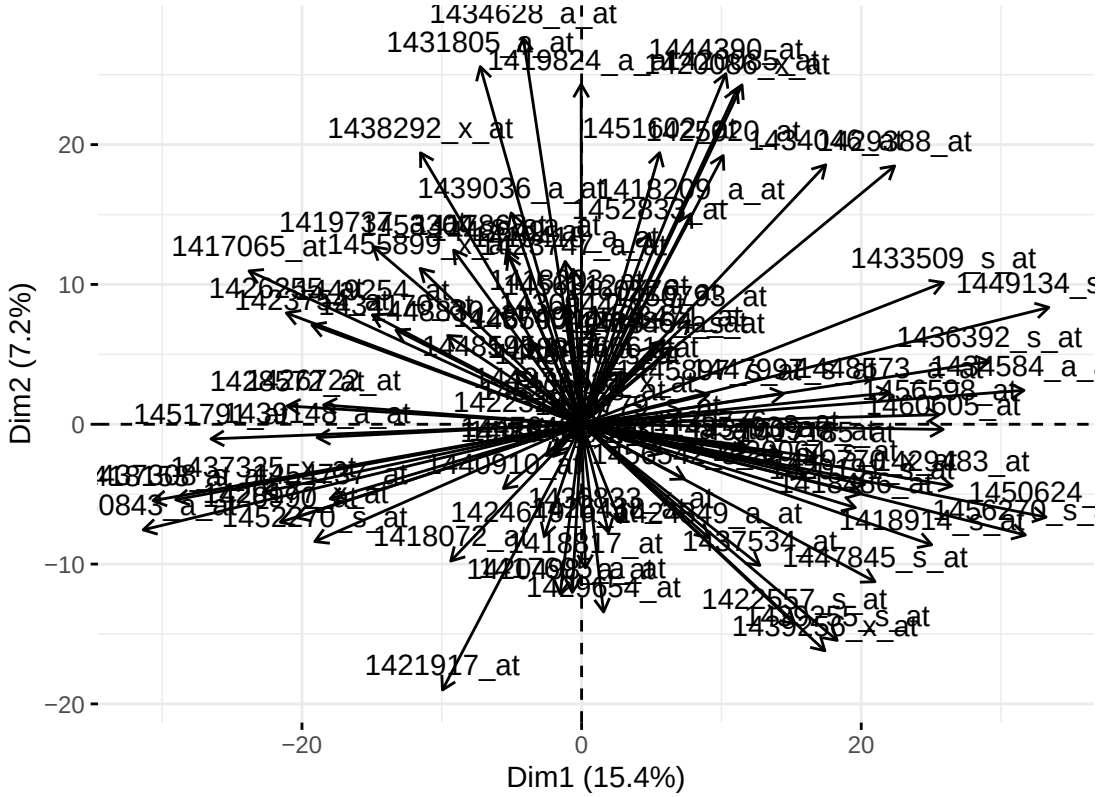


```
fviz_pca_biplot(pcaMouse, geom = "point", habillage = xwt$sampleGroup, col.var = "violet", addEllipses =
```




```
#From Jill's
pca_unweighted = dudi.pca(as.data.frame(t(Biobase::exprs(xwt))),
  row.w = as.numeric(tab[xwt$sampleGroup]),
  center = TRUE, scale = TRUE, nf = 2, scannf = FALSE)

fviz_pca_var(pca_unweighted, col.circle = "black") + ggtitle("") + coord_fixed()
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```




```
largePC1 <-pca_unweighted[,(pca_unweighted$co$Comp1 >= 30 | pca_unweighted$co$Comp1 <= -30)]

fviz_pca_biplot(largePC1, geom = "point", label = "var",
                habillage = xwt$sampleGroup) +
  ggtitle("") +
  coord_fixed()
```

```
## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
```

```
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
```

```
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
```

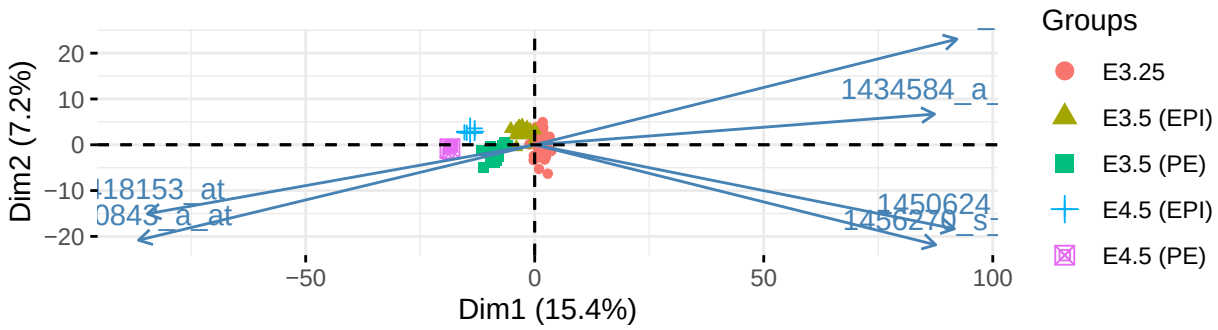
```
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
```

```
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
```

```
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
```

```
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
```

```
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length

## Warning in x * res.pca$norm: longer object length is not a multiple of shorter
## object length
```


Groups
E3.25
E3.5 (EPI)
E3.5 (PE)
E4.5 (EPI)
E4.5 (PE)
Dim2 (7.2%)
Dim1 (15.4%)
20
10
0
−10
−20
−50
0
50
100
1434584_a_
1450624_
1456270_s_
+18153_at
0843_a_at