

Solution to Question 3.1

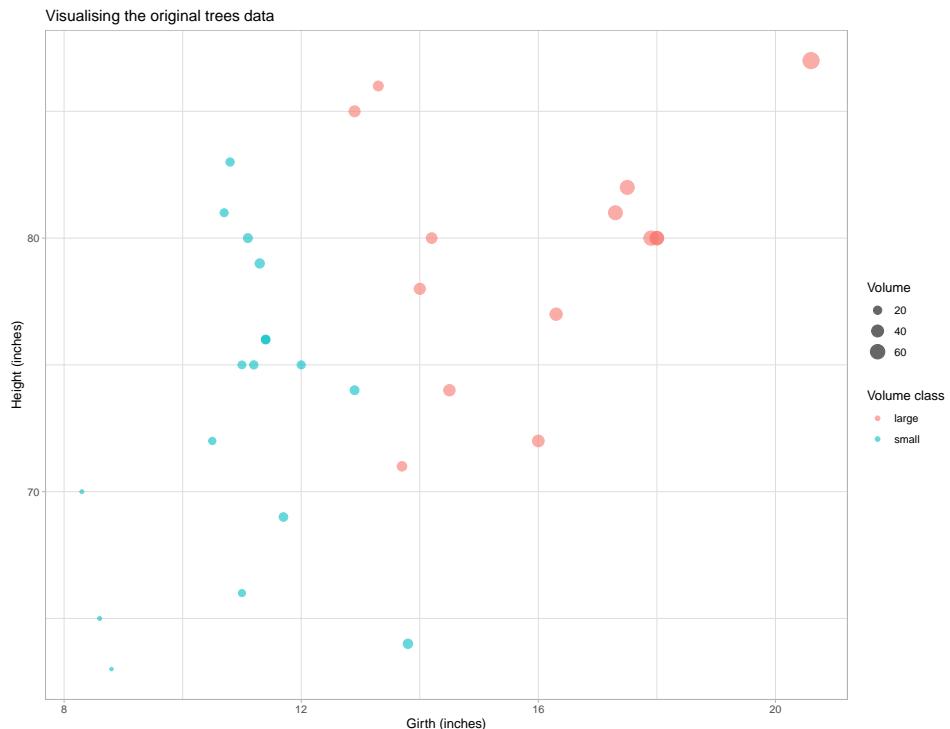
- (a) Use the following code to load the trees data, create a categorical variable (factor) from the volume variable and plot the data:

```
data(trees)

trees$vol_fac <- as.factor(ifelse(trees$Volume > 25, "large", "small"))

trees_plot <- ggplot(trees) +
  geom_point(aes(Girth, Height, size=Volume, col=vol_fac), alpha=0.6) +
  labs(x = "Girth (inches)", y = "Height (inches)",
       color = "Volume class")

trees_plot + ggtitle("Visualising the original trees data")
```



- (b) Use the following lines to scale the variables and apply the PCA:

```
tree_mx <- cbind("height" = trees$Height, "girth" = trees$Girth)
tree_mx <- scale(tree_mx, center=TRUE, scale=TRUE)
tree_pca <- prcomp(tree_mx, center=FALSE, scale.=FALSE)
tree_pca$rotation #loadings
```

Note that the R function `prcomp` provides the option to standardise the variables when apply PCA, but here no need to standardise again as the variables are already scaled. To interpret the PCA results, the summary of PCA result is as follows:

Importance of components:

PC1	PC2
Standard deviation	1.2326 0.6933
Proportion of Variance	0.7596 0.2404
Cumulative Proportion	0.7596 1.0000

It can be seen that the first PC captures about 75% of the total variation and the second PC explains the remaining 25% of the data variance. Also, from the PC coefficients printed below, it can be seen that both variables, height and girth, are equally important in the variability of the data.

PC1	PC2
height 0.7071068	-0.7071068
girth 0.7071068	0.7071068

(c) This can be done using the following lines:

```
pc_loadings <- t(tree_pca$rotation) * tree_pca$sdev

## Reuse plot from before and add PCs
library(tidyverse)
trees_plot2 <- tree_mx %>%
  ## Convert to data.frame and re-add Volume columns for plotting
  data.frame(Volume = trees$Volume, vol_fac = trees$vol_fac) %>%
  ggplot() +
  geom_point(aes(girth, height, size = Volume, col = vol_fac),
  alpha = 0.6) +
  labs(x = "Girth (inches), standardised", y = "Height (feet), standardised",
  color = "")

trees_plot2+
  geom_segment(
  data = data.frame(pc_loadings),
```

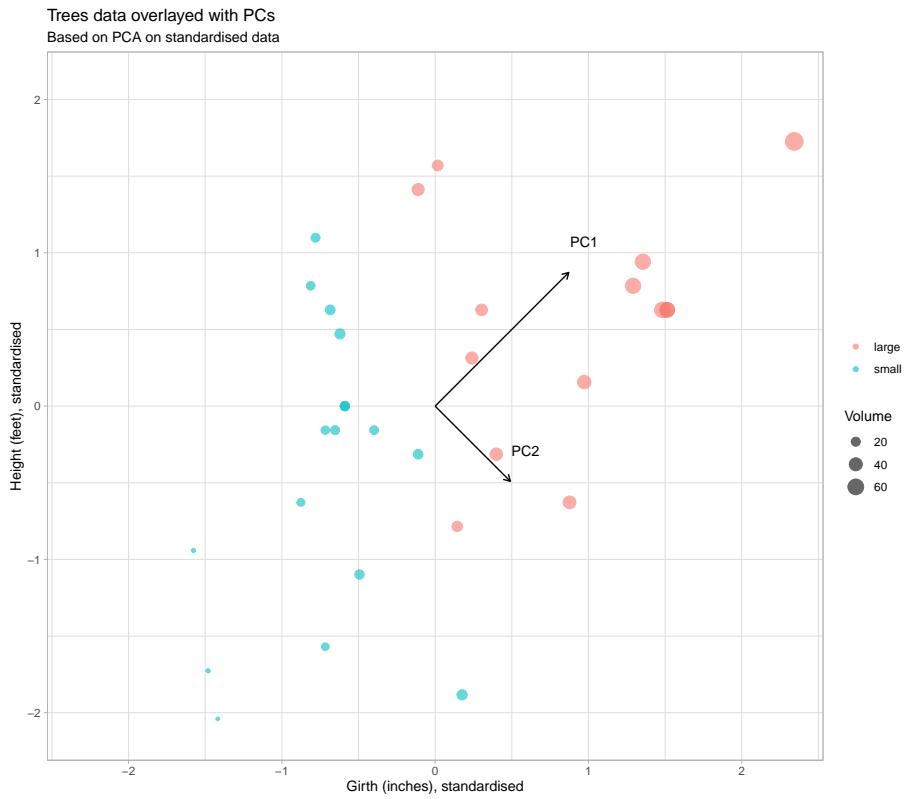
```

aes(x = 0, xend = girth, y = 0, yend = height),
arrow = arrow(length=unit(0.2,"cm"))
) +
geom_text(
data = data.frame(pc_loadings),
aes(x = girth, y = height, label = rownames(pc_loadings)),
nudge_x = 0.1, nudge_y = 0.2
) +
coord_equal(xlim = c(-2.3, 2.3), ylim = c(-2.1, 2.1)) +
ggtitle("Trees data overlayed with PCs",
subtitle = "Based on PCA on standardised data")

ggbiplot(tree_pca, groups = trees$vol_fac, alpha = 0) +
## Add points layer to color and size points
geom_point(aes(col = trees$vol_fac, size = trees$Volume), alpha = 0.6) +
labs(size = "Volume", col = "") +
theme(aspect.ratio = 0.6) +
ggtitle("Biplot for the PCA on non-standardised data")

```

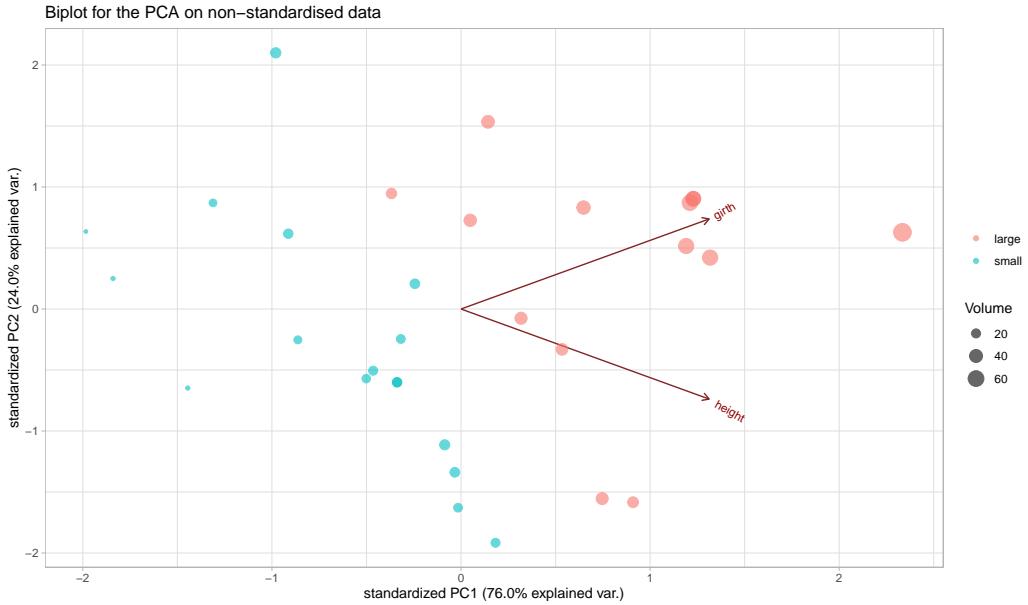
The resulting plot is shown below, where PC1 points to the direction of greatest variability, with PC2 orthogonal and pointing to the direction of second greatest variability. Note also from the lengths of the PC vectors that the contributions of the height and girth variables are equal to both PCs, as seen before.



(d) This can be done using the following code:

```
ggbiplot(tree_pca, groups = trees$vol_fac, alpha = 0) +
  ## Add points layer to color and size points
  geom_point(aes(col = trees$vol_fac, size = trees$Volume), alpha = 0.6) +
  labs(size = "Volume", col = "") +
  theme(aspect.ratio = 0.6) +
  ggtitle("Biplot for the PCA on non-standardised data")
```

The resulting plot is shown below, where it is clearly seen that the height and girth variables have equal contributions to the PCs. PC1 can be interpreted as separating small and large volume trees. A potential explanation of PC2 would be the separation between trees that have a similar volume but differ in their height-to-girth ratios, i.e. short wide trees and long thin trees.



- (e) To repeat the above codes for PCA without scaling the variables and by changing the unit of height from inches to millimeters, try the following lines in R:

```

trees$vol_fac <- as.factor(ifelse(trees$Volume > 25, "large", "small"))
tree_mx <- cbind("height" = trees$Height, "girth" = trees$Girth)
trees$height <- trees$Height *25.4#change the unit from inches to mm
tree_mx <- cbind("height" = trees$height, "girth" = trees$Girth)

tree_pca <- prcomp(tree_mx, center=FALSE, scale.=FALSE)
summary(tree_pca)
tree_pca$rotation
pc_loadings <- t(tree_pca$rotation) * tree_pca$sdev

ggbiplot(tree_pca, groups = trees$vol_fac, alpha = 0) +
## Add points layer to color and size points
geom_point(aes(col = trees$vol_fac, size = trees$Volume), alpha = 0.6) +
labs(size = "Volume", col = "") +
theme(aspect.ratio = 0.6) +
ggttitle("Biplot for the PCA on non-standardised data")

```

The PCA result in this case would be as follows, where it can be seen that the first PC is completely dominated by the height variable, while the second component is basically girth variable. Since the PCs are ordered by the amount of variance they retain from the original data, we would conclude that most of the variance in the data comes from the height variable but we know that this may not be the case because of the high scale of height measures in mm.

Importance of components:

PC1 PC2

Standard deviation 1969 2.731

Proportion of Variance 1 0.000

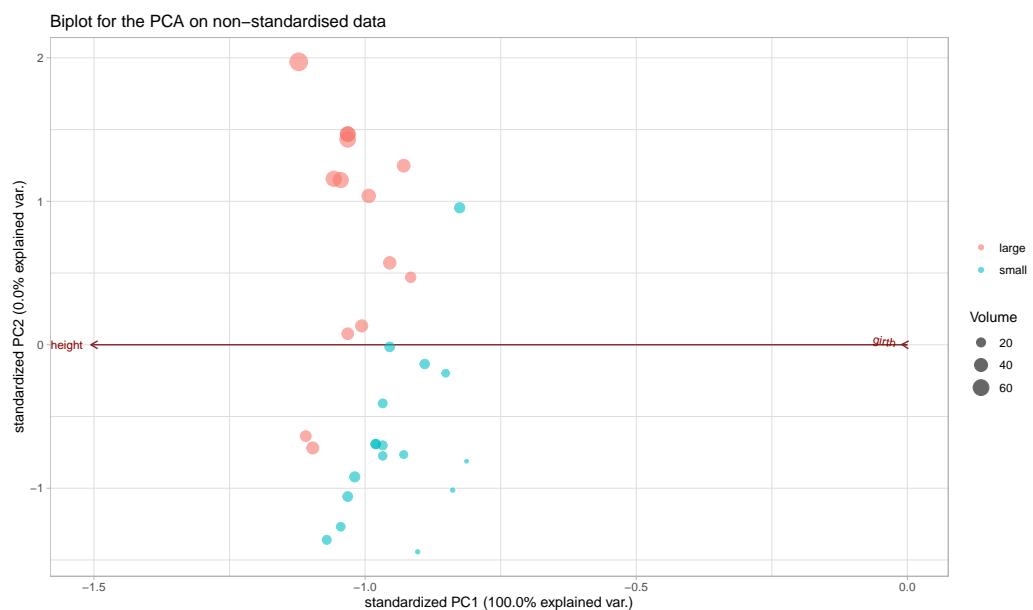
Cumulative Proportion 1 1.000

PC1 PC2

height -0.999976301 -0.006884536

girth -0.006884536 0.999976301

Also, from the resulting plot shown below, the PCs are not in the directions we expect. PC1 should point to the direction of greatest variability. This is again because of the very different scales of the two variables. So, it is important to ensure the scales of variables are not much different when applying the PCA.



Solution to Question 3.2

- (a) Use the following code in R to load the data into R and create the variables Y and X (scaled):

```
load(file="tumor_tissue.rda")
Y <- as.factor(tumor_tissue[, 1])
levels(Y) <- c("Normal", "Tissue")
X <- tumor_tissue[, -1]
X <- scale(X, center=TRUE, scale=TRUE)
```

Note that, as usual, you may need to use the correct directory of the data file in your computer.

- (b) To apply the SVD to matrix X , use the following lines:

```
svd_X <- svd(X)
V <- svd_X$v #eigenvectors of sample covariance X^T*X/n
svd_X$d
```

From the singular values, printed below, it can be seen that only 62 singular values are non-zero, which is equal to n . This is line with the theory as the rank of X is $n = 62$ (check its rank in R as we did before).

```
[1] 2.341920e+02 1.096012e+02 9.084794e+01 8.306221e+01 6.329142e+01
[6] 6.178608e+01 5.333760e+01 5.201533e+01 4.509846e+01 4.352087e+01
[11] 4.108040e+01 3.837002e+01 3.564075e+01 3.445785e+01 3.212196e+01
[16] 3.097257e+01 3.073367e+01 2.851180e+01 2.775367e+01 2.678623e+01
[21] 2.595366e+01 2.530222e+01 2.463830e+01 2.422768e+01 2.359226e+01
[26] 2.247301e+01 2.229530e+01 2.162848e+01 2.066229e+01 2.014215e+01
[31] 1.969091e+01 1.919493e+01 1.888744e+01 1.877634e+01 1.815804e+01
[36] 1.757171e+01 1.729534e+01 1.688516e+01 1.627497e+01 1.607163e+01
[41] 1.556565e+01 1.536045e+01 1.498127e+01 1.477190e+01 1.459136e+01
[46] 1.426410e+01 1.394016e+01 1.369916e+01 1.300973e+01 1.281477e+01
[51] 1.252073e+01 1.224269e+01 1.160828e+01 1.139784e+01 1.091438e+01
[56] 1.059572e+01 1.002578e+01 9.869309e+00 8.867742e+00 8.446363e+00
[61] 7.195360e+00 1.040021e-13
```

To store the scores of the PCs in a matrix Z , use the following line:

```
Z <- svd_X$u %*% diag(svd_X$d) #calculate and store the scores
```

(c) To apply the PCA on the scaled matrix X, use the following code:

```
pca_x <- prcomp(X, center=FALSE, scale.=FALSE) #do not standardise again  
princomp(X) #this function is only applicable to low dimensional data n>p  
summary(pca_x)  
pca_x$x #scores - same as Z in the above obtained directly using SVD  
pca_x$rotation #loadings  
View(pca_x$rotation-V) #check the loadings using prcomp and SVD are the same
```

The summary of PCA result, printed below, shows that the first 20 PCs explain about 90% of the data variance. Although one prefers to ideally have just few PCs for this percentage, but we note that there are 2000 variables in the data so having 20 PCs capturing 90% of the variance is still very good.

Importance of components:

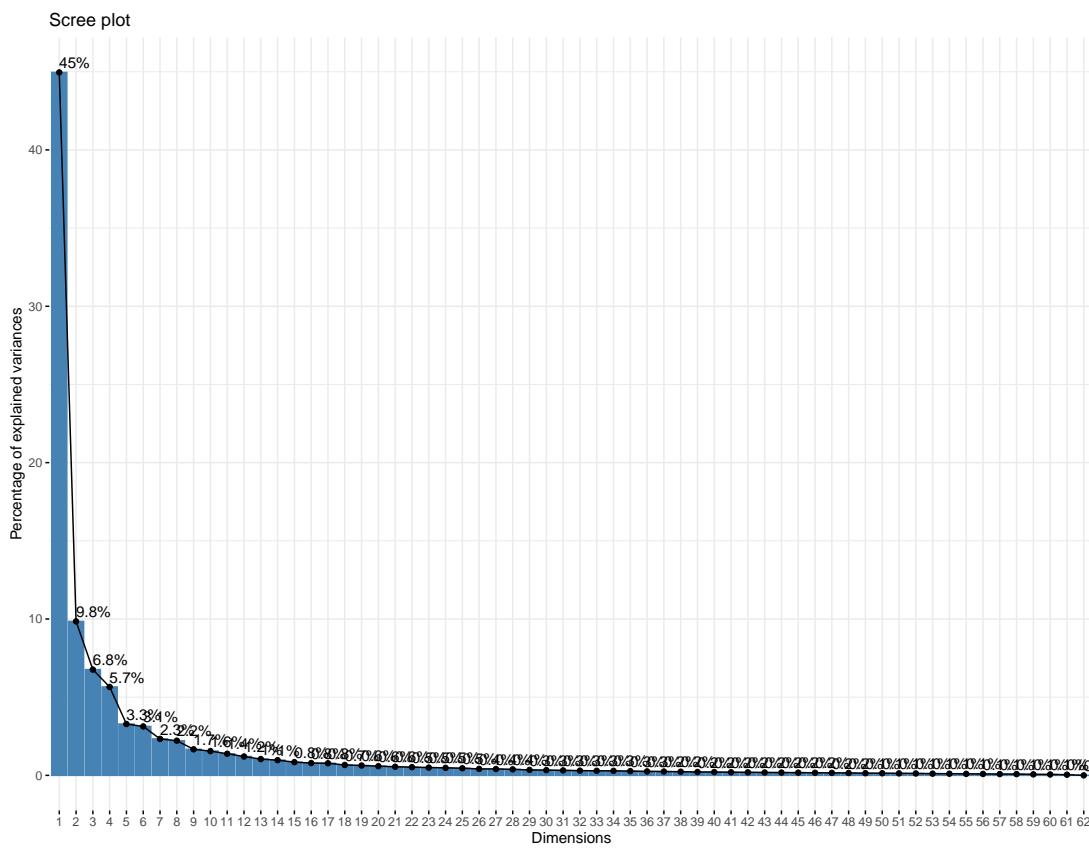
PC1	PC2	PC3	PC4	PC5							
Standard deviation		29.9852	14.03300	11.63189	10.63503	8.10364					
Proportion of Variance		0.4496	0.09846	0.06765	0.05655	0.03283					
Cumulative Proportion		0.4496	0.54802	0.61567	0.67222	0.70506					
PC6	PC7	PC8	PC9	PC10	PC11						
Standard deviation		7.91090	6.82918	6.65988	5.77427	5.57228	5.25981				
Proportion of Variance		0.03129	0.02332	0.02218	0.01667	0.01553	0.01383				
Cumulative Proportion		0.73635	0.75967	0.78184	0.79851	0.81404	0.82787				
PC12	PC13	PC14	PC15	PC16	PC17						
Standard deviation		4.91278	4.56333	4.41188	4.11280	3.96563	3.93504				
Proportion of Variance		0.01207	0.01041	0.00973	0.00846	0.00786	0.00774				
Cumulative Proportion		0.83994	0.85035	0.86008	0.86854	0.87640	0.88415				
PC18	PC19	PC20	PC21	PC22	PC23						
Standard deviation		3.65056	3.55349	3.42963	3.32303	3.23962	3.15461				
Proportion of Variance		0.00666	0.00631	0.00588	0.00552	0.00525	0.00498				
Cumulative Proportion		0.89081	0.89712	0.90300	0.90853	0.91377	0.91875				
PC24	PC25	PC26	PC27	PC28	PC29						
Standard deviation		3.10204	3.02068	2.87737	2.85462	2.76924	2.6455				
Proportion of Variance		0.00481	0.00456	0.00414	0.00407	0.00383	0.0035				
Cumulative Proportion		0.92356	0.92812	0.93226	0.93634	0.94017	0.9437				
PC30	PC31	PC32	PC33	PC34	PC35						
Standard deviation		2.57894	2.52116	2.45766	2.41829	2.40406	2.3249				

Proportion of Variance	0.00333	0.00318	0.00302	0.00292	0.00289	0.0027
Cumulative Proportion	0.94700	0.95017	0.95319	0.95612	0.95901	0.9617
PC36	PC37	PC38	PC39	PC40	PC41	
Standard deviation	2.24983	2.21444	2.16192	2.08380	2.05776	1.99298
Proportion of Variance	0.00253	0.00245	0.00234	0.00217	0.00212	0.00199
Cumulative Proportion	0.96424	0.96669	0.96903	0.97120	0.97332	0.97530
PC42	PC43	PC44	PC45	PC46	PC47	
Standard deviation	1.96670	1.91815	1.89135	1.86823	1.82633	1.78485
Proportion of Variance	0.00193	0.00184	0.00179	0.00175	0.00167	0.00159
Cumulative Proportion	0.97724	0.97908	0.98087	0.98261	0.98428	0.98587
PC48	PC49	PC50	PC51	PC52	PC53	
Standard deviation	1.75400	1.66572	1.64076	1.60312	1.56752	1.4863
Proportion of Variance	0.00154	0.00139	0.00135	0.00128	0.00123	0.0011
Cumulative Proportion	0.98741	0.98880	0.99014	0.99143	0.99266	0.9938
PC54	PC55	PC56	PC57	PC58	PC59	
Standard deviation	1.45934	1.39744	1.35664	1.28367	1.2636	1.13540
Proportion of Variance	0.00106	0.00098	0.00092	0.00082	0.0008	0.00064
Cumulative Proportion	0.99483	0.99580	0.99672	0.99755	0.9983	0.99899
PC60	PC61	PC62				
Standard deviation	1.08145	0.92127	1.332e-14			
Proportion of Variance	0.00058	0.00042	0.000e+00			
Cumulative Proportion	0.99958	1.00000	1.000e+00			

To get a scree plot using the R package `factoextra`, try the following code:

```
library(factoextra)
fviz_eig(pca_x, ncp=62, addlabels=TRUE)
```

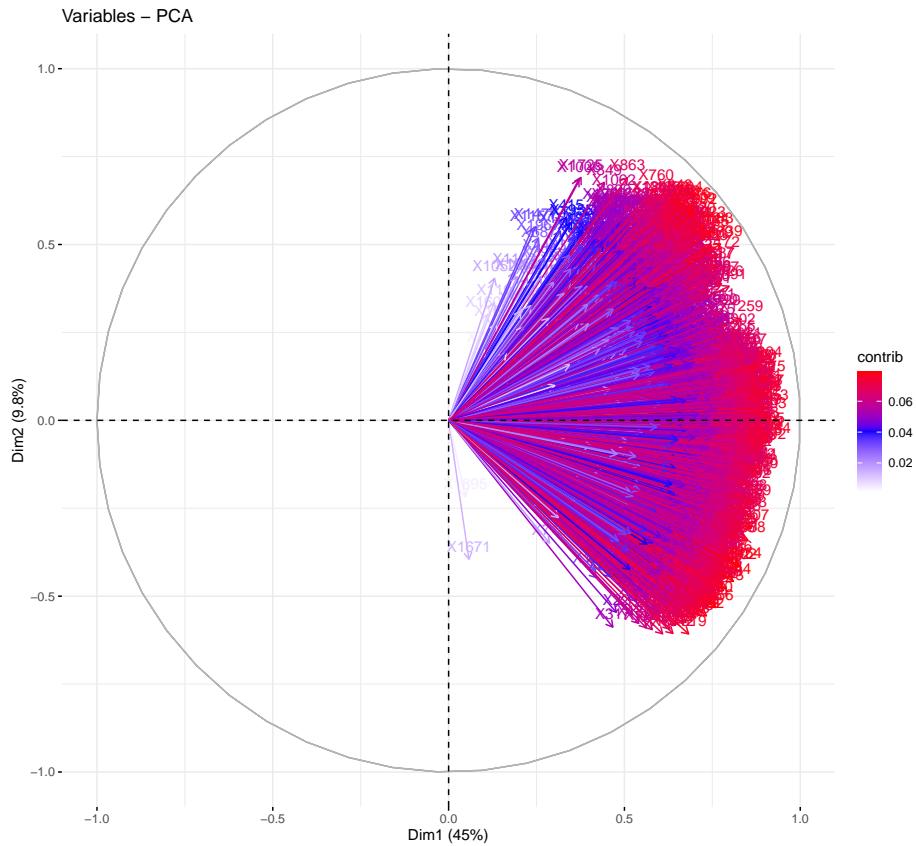
The scree plot is shown below. If one wants to use the “elbow” method, perhaps the first 7 PCs would not be a bad choice as they capture about 75% of the data variation. In practice, it depends on the importance of decision by the user whether a high percentage of data variance (here 90% with 20 PCs) is preferred or a really smaller number of PCs but still reasonably high variance (here 75% with 7 PCs) is preferred. Any of these results sounds remarkable since there are 2000 variables in the data.



- (d) Use the following line to get a biplot visualising the similarities and dissimilarities between the samples:

```
fviz_pca_var(pca_x, col.var="contrib", gradient.cols=c("white", "blue", "red"))
```

The resulting plot is shown below. Since there are 2000 variables, the plot looks messy and it is not easy to clearly see the result for each variable, however a main conclusion here is that all the variables positively contribute to the first PC (the leading PC).



(e) Similar to Question 3.1, this can be done using the following code:

```
ggbiplot(pca_x, groups = Y, alpha = 0) +  
## Add points layer to color points  
geom_point(aes(col = Y), alpha = 0.6) +  
theme(aspect.ratio = 0.6) +  
ggtitle("Biplot for the first two PCs on tumor data")
```

(f) To get the histograms for the loadings of the first and second PCs, use the following lines:

```

# First PC, first column of V
hist(V[, 1], breaks = 50, xlab = "PC 1 loadings", main = "")
# Add vertical line at 95% quantile
abline(v = quantile(V[, 1], 0.95), col = "red", lwd = 2)

# Second PC, second column of V
hist(V[, 2], breaks = 50, xlab = "PC 2 loadings", main = "")
abline(v = c(
quantile(V[, 2], 0.05),

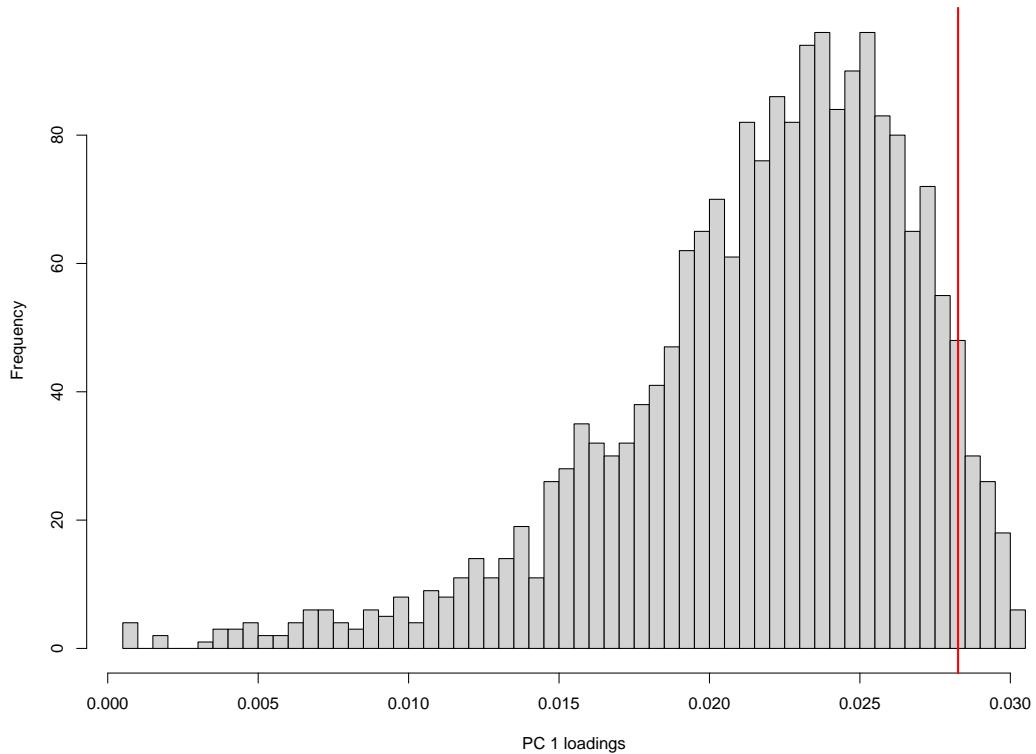
```

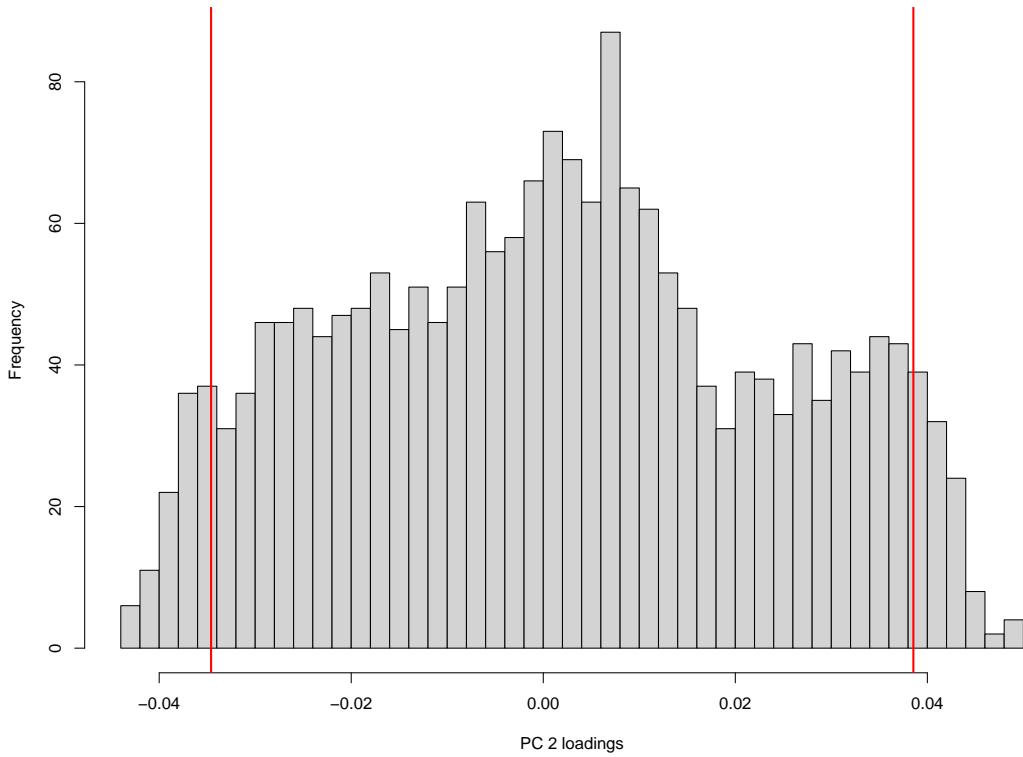
```

quantile(V[, 2], 0.95)
), col = "red", lwd = 2)

```

The resulting histogram are shown below. Note that the vertical lines are added at the 95th percentile for PC1 and at the 5th and 95th percentiles for PC2 to reflect where the highest loadings (in absolute value) are situated. Since there are no negative loadings for PC1 as seen earlier, we only show the 95th percentile. To interpret the histograms, first note that the PC loadings reflect the contributions of each variable/feature (in this case: gene) to the PC. From these histograms, it seems that only a minor fraction of the genes are really driving these first 2 PCs, especially for PC2 where the bulk of genes have loadings very close to 0. This is exactly where sparse PCA can be useful to shrink loading and set those very small ones to 0 and provide sparse PCs. This will be investigated in the next part of the question.





(g) To apply the sparse PCA for the first PC, use the following code:

```
library(glmnet)
# For PC1
set.seed(1)
fit_loadings1 <- cv.glmnet(X, Z[, 1],
alpha = 0.5, nfolds = 5
)
fit_loadings1
```

The sparse PCA result for the first PC is shown below, where it can be seen that for PC1 around 90 to 100 genes are most important, based on the range of lambda values between `lambda.min` and `lambda.1se`. With this information, we can choose one of these lambda values and construct PC that will have non-zero loadings for only a smaller number of genes.

```
Call: cv.glmnet(x = X, y = Z[, 1], nfolds = 5, alpha = 0.5)
Measure: Mean-Squared Error
```

Lambda	Index	Measure	SE	Nonzero
--------	-------	---------	----	---------

```

min  1.195     83   9.944 1.659      101
1se  1.655     76  11.312 2.079      96

```

Similarly, to apply the sparse PCA for the second PC, use the following code:

```

# For PC2
set.seed(1)
fit_loadings2 <- cv.glmnet(X, Z[, 2], alpha = 0.5, nfolds = 5)
fit_loadings2

```

The sparse PCA result for the second PC is shown below, where it can be seen that for PC2 around 65 to 80 genes are most important, based on the range of lambda values between `lambda.min` and `lambda.1se`. Again, we can choose one of these lambda values and construct PC that will have non-zero loadings for only a smaller number of genes.

```

Call: cv.glmnet(x = X, y = Z[, 2], nfolds = 5, alpha = 0.5)
Measure: Mean-Squared Error

```

Lambda	Index	Measure	SE	Nonzero	
min	0.3062	90	9.365	4.023	79
1se	1.0263	64	13.239	5.403	69

Also, the following code produces some plots for the sparse PCA for the first and second PCs, including the coefficient profile plots showing how many features are important for each fit.

```

par(mfrow = c(2, 1))
plot(fit_loadings1, main = "PC1")
plot(fit_loadings2, main = "PC2")

par(mfrow = c(2, 1))
plot(fit_loadings1$glmnet.fit, main = "PC1", xvar = "lambda")
abline(v = log(fit_loadings1$lambda.min), lty = 3)
abline(v = log(fit_loadings1$lambda.1se), lty = 3)
plot(fit_loadings2$glmnet.fit, main = "PC2", xvar = "lambda")
abline(v = log(fit_loadings2$lambda.min), lty = 3)
abline(v = log(fit_loadings2$lambda.1se), lty = 3)

```

