STAT 32950: Homework 7

Robert Winter

Table of Contents

1 Exercise 1: Least Squares vs. Ridge Regression

Consider the linear regression model

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon.$$

Generate the data using the following R commands:

```
set.seed(41)
```

```r
x1 = rnorm(30)
x2 = x1 + rnorm(30, sd=0.01)
Y = rnorm(30, mean = 3+x1+x2)

q1data = cbind(Y, x1, x2) %>% as.data.frame()
```

1.1 Part (a)

Write the fitted model with estimated parameters by the least squares method (LS). The R command for fitting the LS model is `lm(Y~x1+x2)`.

First, we fit the model using ordinary least squares:

```r
lm(Y ~ x1 + x2, data = q1data) %>% summary()
```

```
Call:
lm(formula = Y ~ x1 + x2, data = q1data)

Residuals:
    Min      1Q  Median      3Q     Max
-2.3485 -0.5862 -0.2125  0.9731  1.9461

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.1189     0.2036  15.319 7.72e-15 ***
x1          -22.4981    23.2080  -0.969    0.341
x2           24.6947    23.2481   1.062    0.298
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.086 on 27 degrees of freedom
Multiple R-squared:  0.8436,    Adjusted R-squared:  0.832
F-statistic: 72.82 on 2 and 27 DF,  p-value: 1.325e-11
```

The fitted model is approximately

$$\begin{aligned}\hat{Y} &= \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 \\ &= 3.119 - 22.498 x_1 + 24.695 x_2.\end{aligned}$$

1.2 Part (b)

What is the true model with the true β_i's? Are the parameter estimates of the LS model in Part (a) good? Why so?

The true model is

$$\begin{aligned} Y &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon \\ &= 3 + x_1 + x_2 + \varepsilon. \end{aligned}$$

Our least squares estimate of the intercept term is pretty good: $\hat{\beta}_0 = 3.119 \approx 3 = \beta_0$. However, the least squares estimates of the coefficients on x_1 and x_2 are far from the truth: $\hat{\beta}_1 = -22.498$ is far from $\beta_1 = 1$, and $\hat{\beta}_2 = 24.695$ is far from $\beta_2 = 1$. The poor performance of the estimators $\hat{\beta}_1$ and $\hat{\beta}_2$ is not surprising. Since x_2 is constructed by jittering x_1 by a mean-zero normal distribution with very small standard deviation, the correlation between x_1 and x_2 is high. In fact, $\text{Corr}(x_1, x_2) \approx 0.99997 \approx 1$! Least squares estimation is unstable (hence not very good) when covariates are highly correlated, as they are here. This is why our estimates $\hat{\beta}_1$ and $\hat{\beta}_2$ are far from their true values and have very large standard errors, with neither estimate being statistically significantly different from 0.

```
cor(q1data$x1, q1data$x2)
```

```
[1] 0.9999702
```

1.3 Part (c)

Compute the residual sum of squares (RSS) of the fitted LS model and the RSS of the true model.

$$RSS = \sum_{j=1}^{n} \left[y_j - (\hat{\beta}_0 + \hat{\beta}_1 x_{1j} + \hat{\beta}_2 x_{2j}) \right]^2.$$

Are the two RSS comparable (or close in numerical values)? Give a reason (or an excuse) of performance of the LS parameter estimates in Part (a) (i.e., on whether bad parameter estimates could yield not so bad prediction values).

The RSS of the fitted LS model is approximately 31.870, while the RSS of the model with the true parameter values is approximately 34.929.

```
sum(q1data$residuals_LS^2) # Residuals from Part (a)
```

```
[1] 31.86998
```

```r
sum(q1data$residuals_true^2) # Residuals from fit with true
 ↪  coefficients
```

```
[1] 34.92911
```

Not only are these two values close to one another, but the RSS of the LS fitted model is actually smaller than that of the true model, despite the LS fitted model's coefficient estimates being far from their true values! The LS model is still able to predict the values of Y fairly well because the poor estimates of β_1 and β_2 essentially "counterbalance" each other. Since $\text{Corr}(x_1, x_2) \approx 1$, we have $x_1 \approx x_2$, and so the LS model's predicted values of Y can be written as

$$\begin{aligned}
\hat{Y}^{LS} &= \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 \\
&= 3.119 - 22.498 x_1 + 24.695 x_2 \\
&= (3 + 0.119) + (-23.498 + 1) x_1 + (23.695 + 1) x_2 \\
&= (3 + x_1 + x_2) + (0.119 - 23.498 x_1 + 23.695 x_2) \\
&\approx (\beta_0 + \beta_1 x_1 + \beta_2 x_2) + (0.119 - 23.498 x_1 + 23.695 x_1) \\
&= \hat{Y}^{true} + (0.119 + 0.197 x_1),
\end{aligned}$$

where the first term is the predicted value of Y from the model with true parameter values. In our dataset, $x_1 \in [-1.774, 2.274]$, and so the difference $\hat{Y}^{LS} - \hat{Y}^{true} \approx 0.119 + 0.197 x_1$ is never too far from 0. This is why the LS model's fitted values are fairly close to the true model's fitted values, and thus why their RSS values are close to each other.

1.4 Part (d)

Use the R function `lm.ridge` to fit a Ridge regression model with $\lambda = 1$. Write out the fitted Ridge model. Are the parameter estimates good?

First, we fit the model using ridge regression with $\lambda = 1$:

```r
lm.ridge(Y ~ x1 + x2, lambda = 1,
         data = q1data)
```

```
               x1       x2
3.152125 1.038297 1.082257
```

The fitted model is approximately

$$\begin{aligned}\hat{Y} &= \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 \\ &= 3.152 + 1.038 x_1 + 1.082 x_2.\end{aligned}$$

The parameter estimates are excellent now: $\hat{\beta}_0 = 3.152 \approx 3 = \beta_0$, $\hat{\beta}_1 = 1.038 \approx 1 = \beta_1$, and $\hat{\beta}_2 = 1.082 \approx 1 = \beta_2$.

1.5 Part (e): Comparison and Comments

What is the criterion of the LS method? That is, which function of the model parameters does the LS method try to optimize? What is the function of model parameters that the Ridge regression method tries to optimize? Compare the two methods using the results in Parts (a) and (d). What is the effect on parameter estimates by the Ridge Regression method?

The LS method chooses $\beta_0, \beta_1,$ and β_2 to minimize the sum of squared residuals:

$$\hat{\beta}_{LS} = \arg\min_{\beta} \left\{ ||Y - \beta_0 - \beta_1 x_1 - \beta_2 x_2||_2^2 \right\}.$$

The ridge regression method chooses $\beta_0, \beta_1,$ and β_2 to minimize the sum of squared residuals *plus* a penalty proportional to the squared 2-norm of the coefficient estimates, $\beta = (\beta_0, \beta_1, \beta_2)$:

$$\begin{aligned}\hat{\beta}_{ridge} &= \arg\min_{\beta} \left\{ ||Y - \beta_0 - \beta_1 x_1 - \beta_2 x_2||_2^2 + \lambda ||\beta||_2^2 \right\} \\ &= \arg\min_{\beta} \left\{ ||Y - \beta_0 - \beta_1 x_1 - \beta_2 x_2||_2^2 + ||\beta||_2^2 \right\},\end{aligned}$$

since we used $\lambda = 1$ in Part (d).

The penalty term in the ridge method's minimization problem forces the model to choose smaller values of $\beta_0, \beta_1,$ and β_2 than the LS method. In other words, ridge regression pulls the β_i estimates closer to 0 than ordinary least squares estimation does.

With our toy dataset, this is clearly beneficial. Using just LS estimation, our estimates of β_1 and β_2 in Part (a) were far too large (in absolute value) relative to the parameters' true values. The ridge regression in Part (d) pulled the estimates of these parameters much closer to their true, more moderate values of 1.

2 Exercise 2: LASSO Regression Exercise

The dataset `Boston` of 506 observations and 14 variables is on housing values in the suburbs of Boston. The data and variable names can be obtained in R by the commands below.

```
data(Boston)
colnames(Boston)
```

```
[1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
[8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

The following describe the variables (variable names in capital letters).

```
1. CRIM: per capita crime rate by town
2. ZN: proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS: proportion of non-retail business acres per town
4. CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
5. NOX: nitric oxides concentration (parts per 10 million)
6. RM: average number of rooms per dwelling
7. AGE: proportion of owner-occupied units built prior to 1940
```

```
8. DIS: weighted distances to five Boston employment centres
9. RAD: index of accessibility to radial highways
10. TAX: full-value property-tax rate per $10,000
11. PTRATIO: pupil-teacher ratio by town
12. BLACK: 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
13. LSTAT: % lower status of the population
14. MEDV: Median value of owner-occupied homes in $1000's
```

More reference information about the data can be found at https://www.rdocumentation.org/packages/mlbench/versions/2.1-1/topics/BostonHousing.

2.1 Part (a)

Take the variable `medv` (median value of owner-occupied homes) as the response variable to fit LASSO regression models, using the first 300 observations as the training set and the rest (206) observations for validation (or calibration; this part of the data is not to be used in cross validation). Interpret your results.

First, we separate the data out into training and calibration sets:

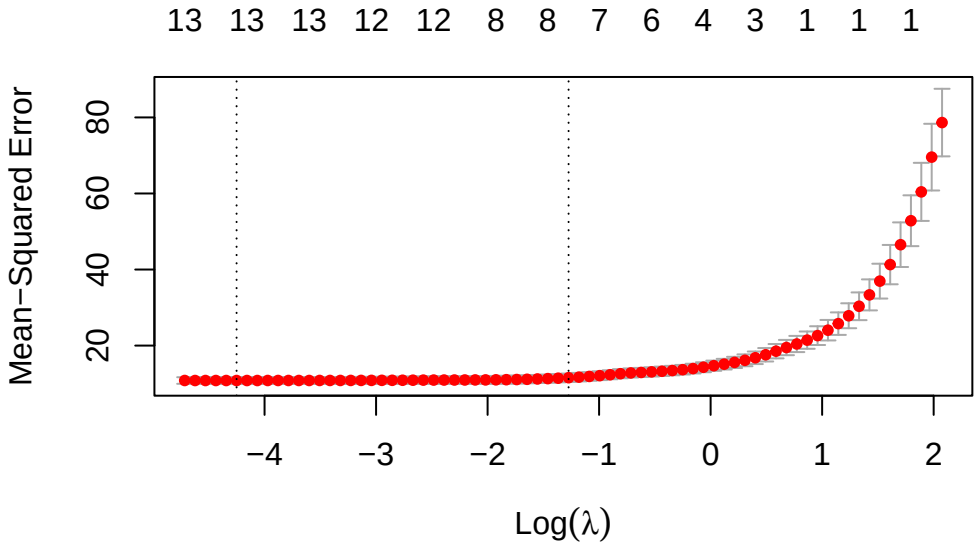
```
q2training = Boston[1:300,]
q2calibration = Boston[301:506,]
```

Now, we train a model of `mdev` on all other covariates using the training data.

```
q2X = as.matrix(q2training[,1:13])
q2Y = q2training[,14]

q2cvfit = cv.glmnet(q2X, q2Y)
```

To select the tuning parameter λ, we consider the MSE as a function of λ. We select $\lambda = \lambda_{1se}$, the largest value of λ such that the mean squared error is within one standard error of its minimum. In this case, $\lambda_{1se} \approx 0.307$ (so $\log \lambda_{1se} \approx -1.181$, the rightmost dotted vertical line in the figure).

```
plot(q2cvfit, label = T)
```




```
q2cvfit$lambda.1se; log(q2cvfit$lambda.1se)
```

```
[1] 0.2797748
```

```
[1] -1.27377
```

Using $\lambda \approx 0.307$, our fitted model is approximately

$$\hat{medv} = -19.634 + 9.164 rm - 0.020 age - 0.273 dis - 0.009 tax - 0.571 ptratio + 0.007 black - 0.123 lstat.$$

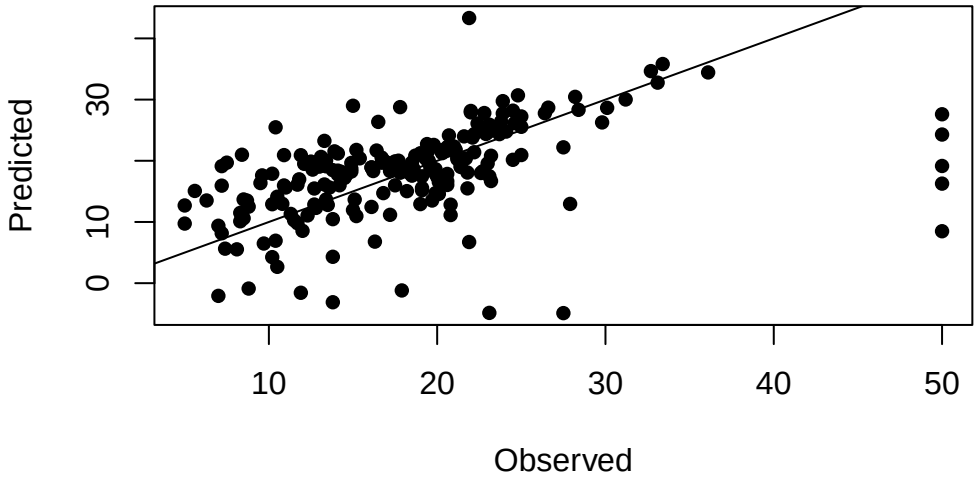
The coefficient estimates on $crim, zn, indus, chas, nox,$ and rad have all been forced to 0 by the LASSO method.

```
coef(q2cvfit, s = "lambda.1se")
```

```
14 x 1 sparse Matrix of class "dgCMatrix"
                       s1
(Intercept) -19.415821167
crim          .
zn            .
indus         .
chas          0.064409103
nox           .
rm            9.182120591
age          -0.022443118
dis          -0.318059931
rad           .
tax          -0.008863193
ptratio      -0.579094638
black         0.007808078
lstat        -0.120815115
```

To check the quality of our model, we compare the actual values of `medv` among observations in the calibration data with the fitted values of `medv` from our LASSO regression, plotted below. For the most part, the (observed, fitted) pairs hug the 45^o line, which shows that even eliminating six covariates, our LASSO model still predicts the `medv` values for novel covariate values fairly well.

```
plot(q2calibration[,14], predict(q2cvfit,
 ↪  newx=as.matrix(q2calibration[,1:13]), s="lambda.1se"), xlab =
 ↪  "Observed", ylab = "Predicted", pch = 16)
abline(0,1)
```



2.2 Part (b)

Compare your fitted LASSO model with the linear model fitted by the ordinary least squares method. Comment.

Below, we fit a linear model to the data using the OLS method. For a fair comparison, we also fit this model on just the 300 training observations.

```
lm(medv ~ ., data = q2training) %>% summary()
```

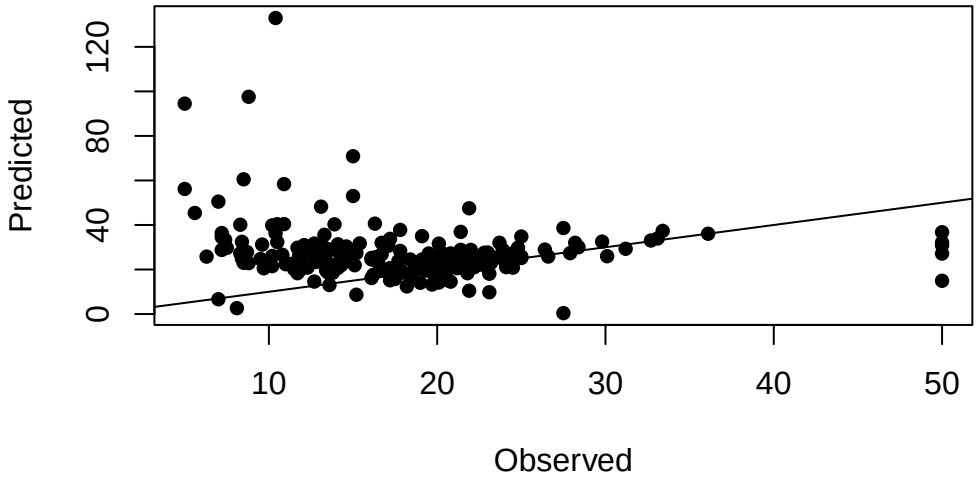
```
Call:
lm(formula = medv ~ ., data = q2training)

Residuals:
    Min      1Q  Median      3Q     Max
-7.7999 -1.8527 -0.3129  1.7408 11.8129

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -12.399469   5.174291  -2.396 0.017201 *
crim          1.201254   0.490473   2.449 0.014919 *
zn            0.014842   0.011114   1.335 0.182807
indus         0.023552   0.045781   0.514 0.607335
chas          0.602065   0.666396   0.903 0.367040
```

```
nox          -8.827642   4.099852  -2.153 0.032142 *
rm            9.130622   0.413824  22.064  < 2e-16 ***
age          -0.047359   0.010709  -4.423 1.39e-05 ***
dis          -1.013286   0.176982  -5.725 2.61e-08 ***
rad           0.167866   0.134521   1.248 0.213097
tax          -0.014567   0.003514  -4.145 4.47e-05 ***
ptratio      -0.641834   0.105411  -6.089 3.65e-09 ***
black         0.016779   0.005014   3.347 0.000928 ***
lstat        -0.109764   0.052516  -2.090 0.037492 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.179 on 286 degrees of freedom
Multiple R-squared:  0.8776,    Adjusted R-squared:  0.8721
F-statistic: 157.8 on 13 and 286 DF,  p-value: < 2.2e-16
```

With respect to some parameters, the OLS method produces very similar estimates as the LASSO method did. For example, our LASSO regression estimated a coefficient of approximately 9.182 on the `rm` variable, and our OLS regression estimated a statistically significant (at the $\alpha = 0.001$ level) coefficient of approximately 9.131 on the same variable. Additionally, for some parameters, the OLS method agrees with the LASSO method in terms of dropped covariates. For example, the LASSO regression imposed coefficients of 0 on the `zn`, `indus`, and `rad` variables, and the OLS regression estimated that the coefficients on each of these variables were not statistically significantly different from 0 at the $\alpha = 0.05$ level. On the other hand, there are some parameters for which the two methods produced very different results. For instance, while the LASSO regression imposed a coefficient of 0 on the `nox` variable, the OLS regression estimated a statistically significant (at the $\alpha = 0.05$ level) coefficient of approximately -8.828 on this variable.

We also plot the OLS regression's fitted values of `mdev` among observations in the calibration data against these observations' true `mdev` values, for comparison with the plot in Part (a). The (observed, fitted) pairs still hug the 45° line for the most part, though predicted values seem systematically too high for small observed values. In general, though, this scatterplot does not look significantly better or worse than the one in Part (a), indicating that our LASSO model was able to reduce the number of model covariates without losing too much predictive power.

```r
plot(q2calibration[,14], predict(lm(medv~., data=q2training),
 ↪  newdata=q2calibration[,1:13]), xlab = "Observed", ylab =
 ↪  "Predicted", pch = 16)
abline(0,1)
```



3 Exercise 3: PCA vs. Sparse PCA

The dataset `hearlossData.csv` can be input into R by the following commands.

```
hearloss =
 ↪  read.csv("C:/Users/rewin/OneDrive/Documents/STAT_32950/Homework/HW7/hearlossData.csv",
 ↪  header = F)
colnames(hearloss)=c("Left5c", "Left1k", "Left2k", "Left4k", "Right5c",
 ↪  "Right1k", "Right2k", "Right4k")
```

The data consists of 100 observations from males, aged 39. The measurements are decibel loss (in comparison to a reference standard) at frequencies 500Hz, 1000Hz, 2000Hz, and 4000Hz for the left and the right ear, respectively. More detailed information can be found in Chapter 5 in the library e-book *A User's Guide to Principal Components* by Jackson.

3.1 Part (a)

Conduct a principal component analysis.

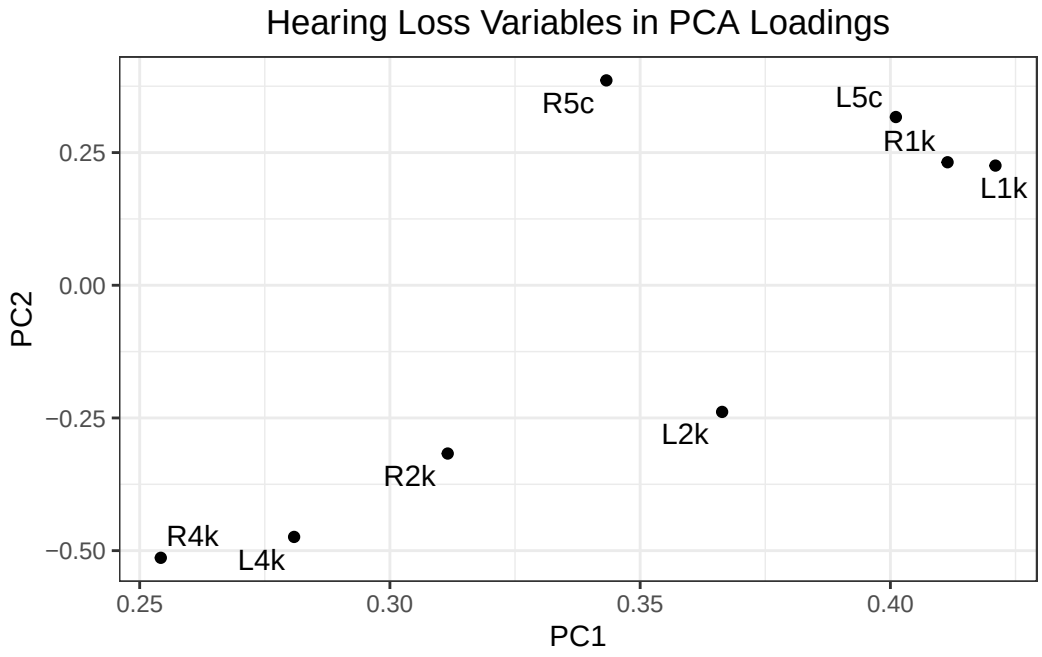
We perform a principal component analysis on the *scaled* (i.e., so that each variable's variance has been scaled to unity) decibel loss data below. Notice that the first two principal

components have nonzero loadings on all eight covariates, that the first principal component captures roughly 49.1% of the total variation in the scaled data, and that the second principal component captures roughly 20.2% of the total variation in the scaled data.

```
q3PCA = princomp(hearloss, cor = T)
summary(q3PCA, loadings = T, digits = 3)
```

```
Importance of components:
                          Comp.1    Comp.2    Comp.3     Comp.4     Comp.5
Standard deviation     1.9821719 1.2721328 0.9875853 0.68321459 0.58317235
Proportion of Variance 0.4911257 0.2022902 0.1219156 0.05834777 0.04251125
Cumulative Proportion  0.4911257 0.6934159 0.8153315 0.87367925 0.91619050
                          Comp.6     Comp.7     Comp.8
Standard deviation     0.5620420 0.44733783 0.39303135
Proportion of Variance 0.0394864 0.02501389 0.01930921
Cumulative Proportion  0.9556769 0.98069079 1.00000000

Loadings:
        Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
Left5c   0.401  0.317  0.158  0.328         0.446  0.329  0.546
Left1k   0.421  0.225         0.482 -0.379                -0.623
Left2k   0.366 -0.239 -0.470  0.282  0.439        -0.526  0.186
Left4k   0.281 -0.474  0.430  0.161  0.350 -0.417  0.427
Right5c  0.343  0.386  0.259 -0.488  0.498  0.195 -0.159 -0.343
Right1k  0.411  0.232        -0.372 -0.351 -0.614         0.361
Right2k  0.312 -0.317 -0.563 -0.391 -0.111  0.265  0.478 -0.147
Right4k  0.254 -0.514  0.426 -0.159 -0.396  0.366 -0.414
```

We also plot the loadings of the eight original hearing loss variables in the plane of the first two principal components. Notice that the variables corresponding to hearing loss (in both ears) at the 500Hz and 1000Hz frequencies are generally plotted in the upper-right of the figure (with relatively large PC1 and PC2 loadings), while the variables corresponding to hearing loss (in both ears) at the 2000Hz and 4000Hz frequencies are generally plotted in the lower-left of the figure (with relatively small PC1 and PC2 loadings).



3.2 Part (b)

Conduct a sparse principal component analysis to highlight important frequency relationship in hearing and hearing loss.

Now, we perform a sparse principal component analysis on the *scaled* (i.e., unit variance) decibel loss data. In particular, we form $K = 2$ principal components, each with 4 nonzero variable loadings. The first principal component captures roughly 37.7% of the total variation in the scaled data, and has nonzero loadings on the variables corresponding to hearing loss at lower frequencies (500Hz and 1000Hz) in both ears. The second principal component captures an additional 25.1% of the total variation in the scaled data, and has nonzero loadings on the variables corresponding to hearing loss at higher frequencies (2000Hz and 4000Hz) in both ears. Thus, the first PC seems to capture a man's hearing loss at lower frequencies (such as 500Hz and 1000Hz), while the second PC seems to capture a man's hearing loss at higher frequencies (such as 2000Hz and 4000Hz).

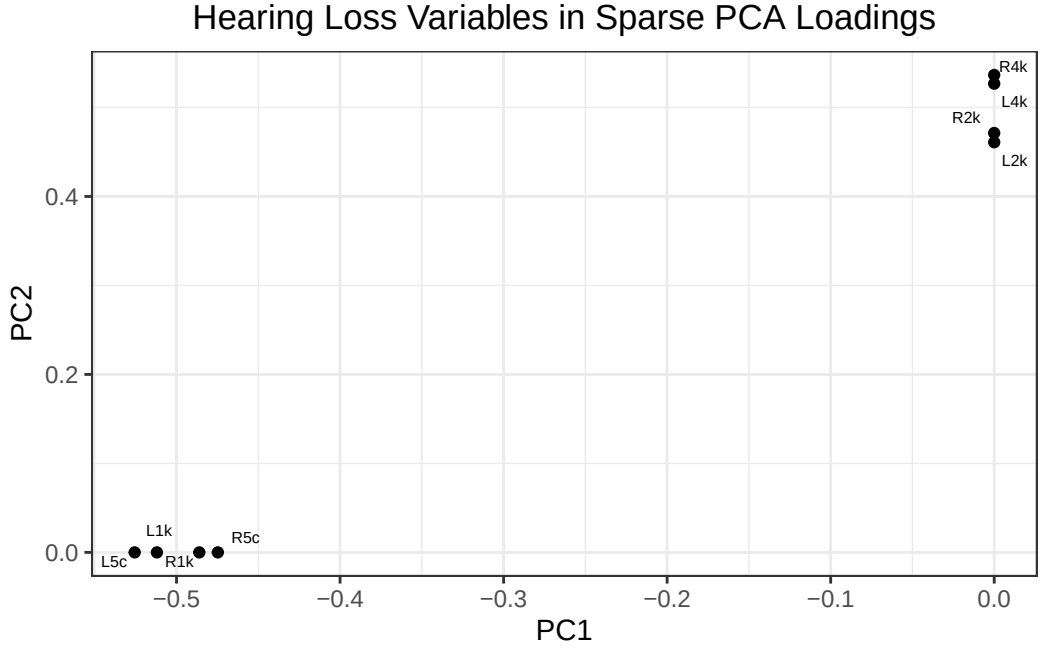
```
q3spca = spca(hearloss, K = 2, para = c(4, 4),
              type = "predictor", sparse = "varnum",
              use.corr = T)

q3spca$pev
```

```
[1] 0.3774926 0.2508558
```

```
q3spca$loadings # Notice that PC1 just has coefficients on 5c and 1k
 ↪  (L&R) while PC2 just has coefficients on 2k and 4k (L&R)
```

```
              PC1       PC2
Left5c  -0.5255778 0.0000000
Left1k  -0.5119926 0.0000000
Left2k   0.0000000 0.4609540
Left4k   0.0000000 0.5267774
Right5c -0.4747405 0.0000000
Right1k -0.4860586 0.0000000
Right2k  0.0000000 0.4713131
Right4k  0.0000000 0.5365547
```

The below plot of the loadings of the eight hearing loss variables in the sparse PC plane further illustrates this point: PC1 captures hearing loss at lower frequencies in both ears, and PC2 captures hearing loss at higher frequencies in both ears. The main takeaway is that variation in 39-year-old men's hearing loss can be pretty well summarized in two dimensions, with one dimension corresponding to hearing loss at lower frequencies and the other corresponding to hearing loss at higher frequencies.



4 Exercise 4: PCA vs. ICA

The data `tableICA` can be read in R by the command below.

```r
tableICA =
 ↪  read.table("C:/Users/rewin/OneDrive/Documents/STAT_32950/Homework/HW7/tableICA")
```

4.1 Part (a)

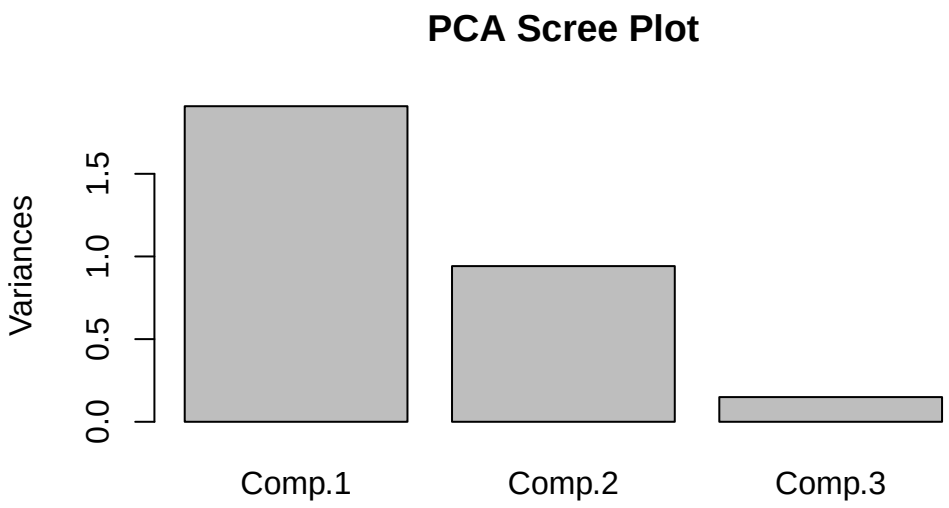
Conduct a Principal Component Analysis. Plot the observations in the space of the first two principal components. Provide the scree plot. Comment on the PCA results.

We perform a principal component analysis on the *scaled* (i.e., so that each variable's variance has been scaled to unity) data below. The first principal component captures roughly 63.6% of the total variation in the scaled data, and the second principal component captures an additional 31.4% of the total variation, so that the first two PCs together capture just over 95.0% of the total variation in the data. That the first two PCs capture the vast majority of the variation in the data is also illustrated in the scree plot below.

```r
q4pca = princomp(tableICA, cor = T)
summary(q4pca, loadings = T, digits = 3)
```

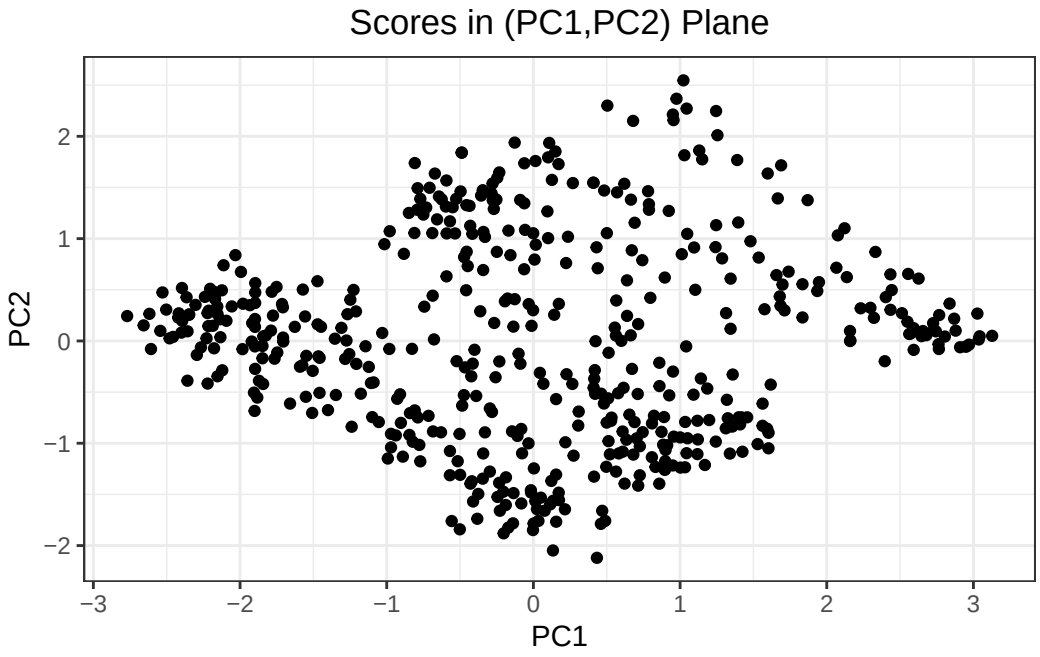
```
Importance of components:
                          Comp.1    Comp.2     Comp.3
Standard deviation     1.3816211 0.9704111 0.38655570
Proportion of Variance 0.6362923 0.3138992 0.04980844
Cumulative Proportion  0.6362923 0.9501916 1.00000000

Loadings:
   Comp.1 Comp.2 Comp.3
V1  0.318  0.922  0.223
V2 -0.694         0.717
V3 -0.646  0.382 -0.660
```

```r
screeplot(q4pca, main = "PCA Scree Plot")
```



Now, we plot the scores of the data in the plane of the first two principal components. Notice that the scores seem to fill in a parallelogram-like shape. So, while our analysis produced two PCs that capture the vast majority of the variation in the data, these two PCs are clearly not independent.

```
ggplot(as.data.frame(q4pca$scores), aes(x=Comp.1, y=Comp.2)) +
  theme_bw() +
  geom_point() +
  xlab("PC1") +
  ylab("PC2") +
  ggtitle("Scores in (PC1,PC2) Plane") +
  theme(plot.title = element_text(hjust = 0.5))
```



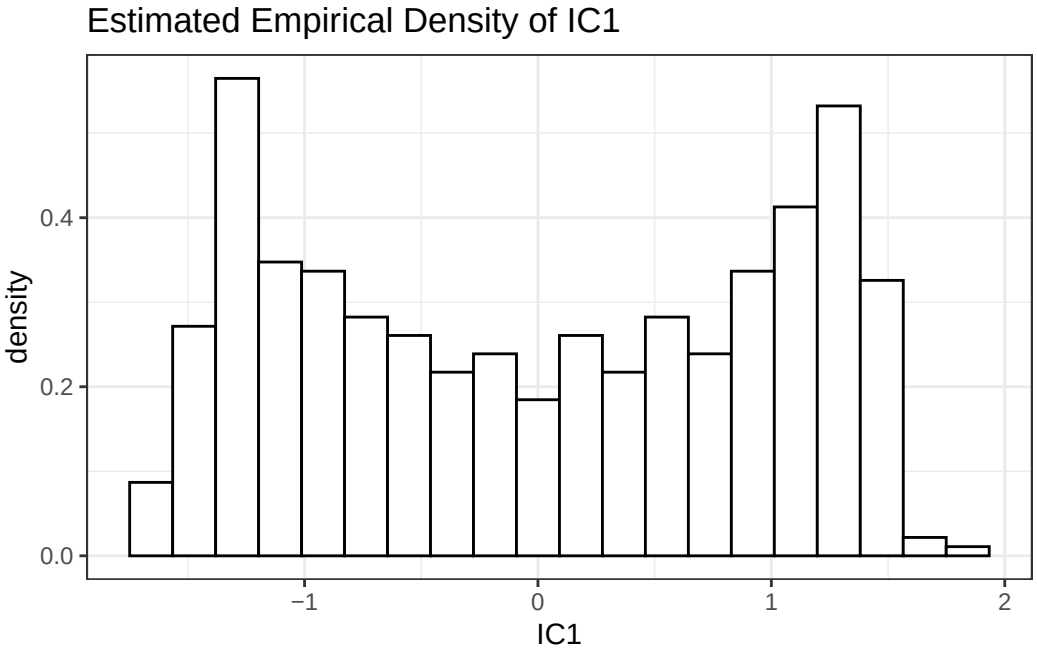
4.2 Part (b)

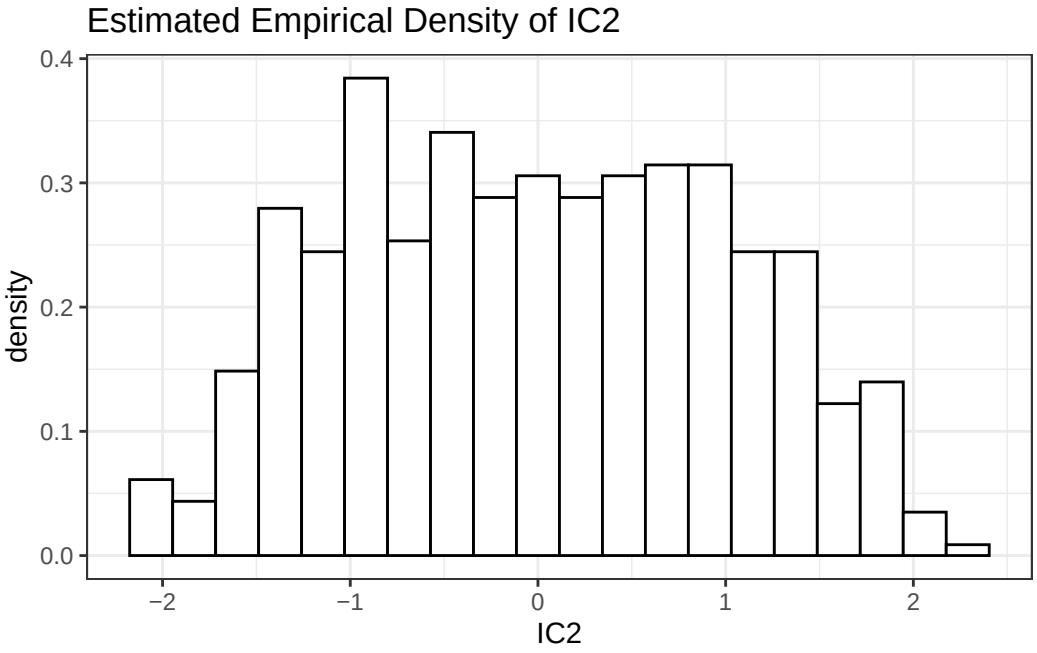
Conduct an Independent Component Analysis. Interpret your results. Plot the three independent components recovered.

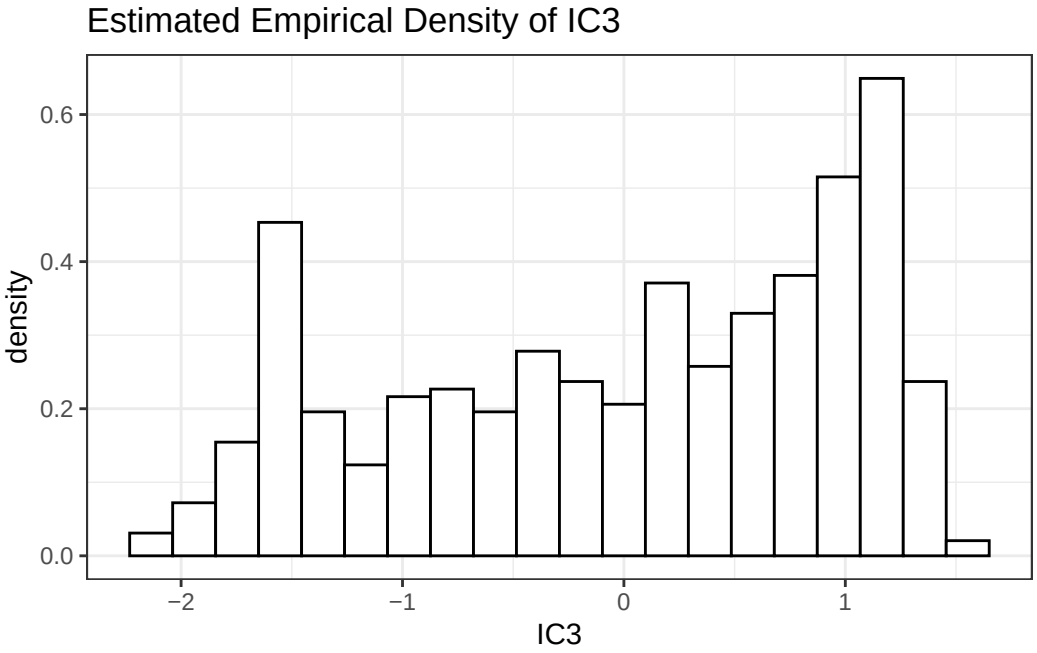
Now, we conduct an independent component analysis assuming that there are three independent components.

```
set.seed(41) # just so that components are in same order every time
q4ica = fastICA(tableICA, n.comp = 3)
```

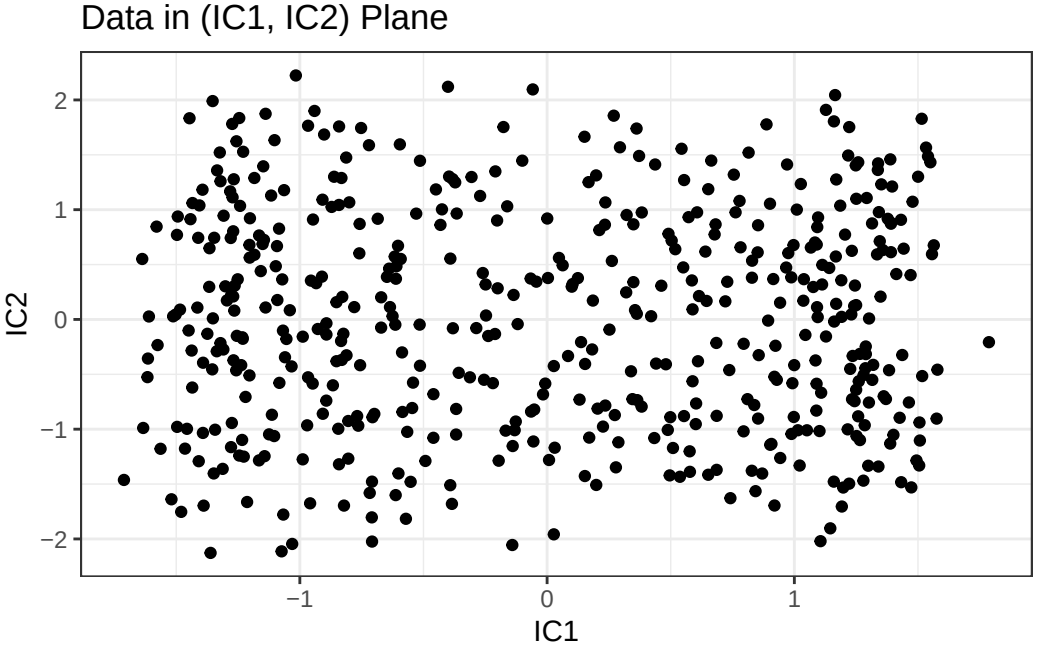
Below are histograms depicting the estimated empirical distributions of the three independent source components. Notice that none of the components appear to have normal distributions: one is roughly symmetric and bimodal, one has a semicircular shape (almost like the Wigner semicircle distribution), and one generally resembles a triangular distribution.

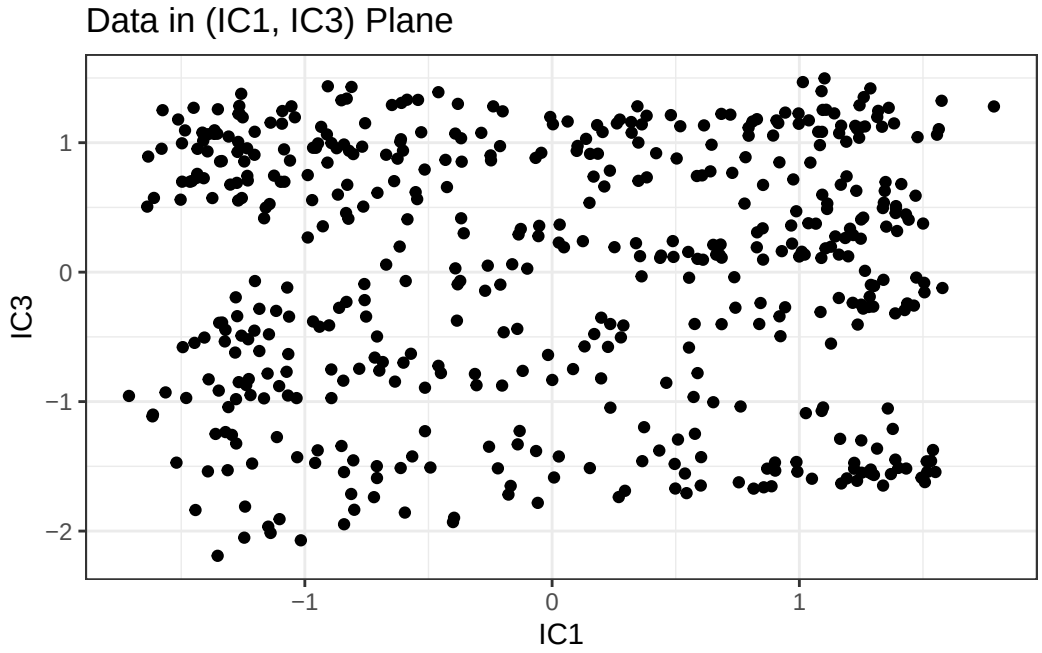

Estimated Empirical Density of IC1
density
0.0
0.2
0.4
−1
0
1
2
IC1

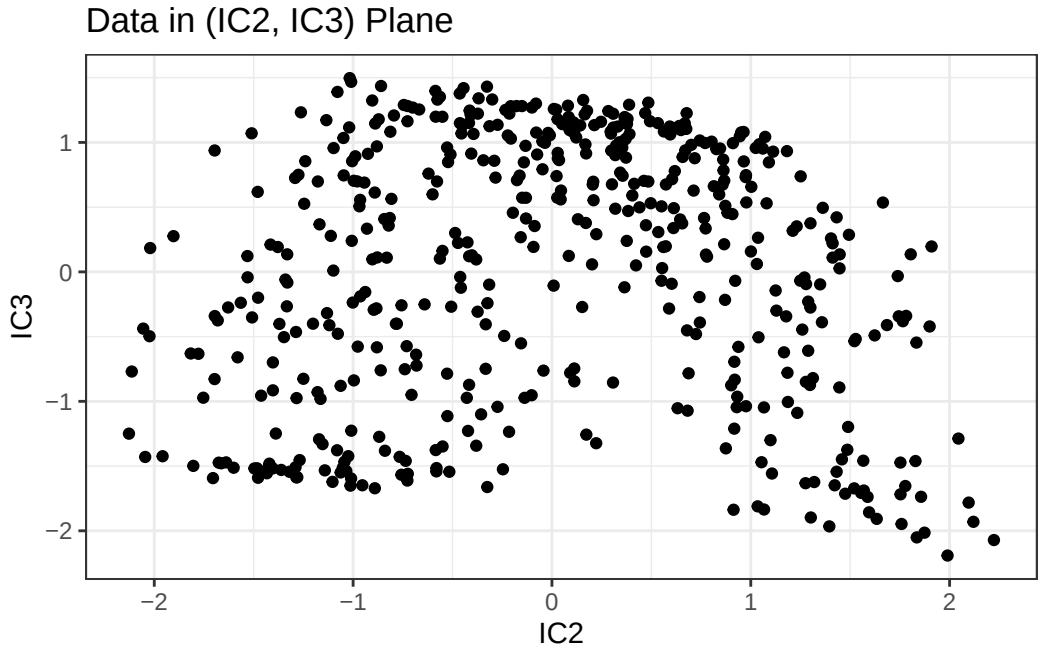

Estimated Empirical Density of IC2
density
0.0
0.1
0.2
0.3
0.4
−2
−1
0
1
2
IC2



We also plot the "unmixed" data in the three pairwise independent component planes: (IC1, IC2), (IC1, IC3), and (IC2, IC3). While the data in the (IC1, IC2) and (IC1, IC3) planes have vaguely box-like shapes and the data in the (IC2, IC3) plane has a vaguely rainbow-like shape, in none of these planes is the shape of the data as clear as it was in the case of the first two principal components.







4.3 Part (c)

Compare Parts (a) and (b) and comment.

In Part (a), we found that the first two principal components captured roughly 95% of the variation in the data, but when the data were plotted in the coordinate plane of these two

PCs, they formed a distinctly parallelogram-shaped cloud. That is, the first two principal components of the data are not independent of each other.

In Part (b), we produced three independent components and found that none of their estimated empirical densities resembled a normal distribution — as is expected in ICA. When we plotted the data in the three pairwise coordinate planes for these three independent components, we found that none of the data clouds had very distinct shapes — meaning that we found three independent components that really do seem independent of one another.

Both methods yield valuable results for analyzing these data. While PCA allowed us to view the three-dimensional data in a two-dimensional plane that still captured the vast majority of the variation in the data, ICA allowed us to discern three independent, non-Gaussian signals that underlie the data.