Exercise #10

Principal Component Analysis

Brian Schweigler; 16-102-071

24/05/2022

Preliminaries

Set a seed for later:

```
set.seed(1786397)
```

Loading the Boston dataset, set CHAS as a factor and show an overview:

```
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))
boston_df = read.csv("Boston.txt", sep = " ", header = TRUE)
boston_df$chas <-
    factor(
        boston_df$chas,
        levels = c("0", "1"),
        labels = c("0", "1")
    )
summary(boston_df)
```

```
##       crim                zn             indus       chas        nox
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   0:471   Min.   :0.3850
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1: 35   1st Qu.:0.4490
##  Median : 0.25651   Median :  0.00   Median : 9.69           Median :0.5380
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14           Mean   :0.5547
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10           3rd Qu.:0.6240
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74           Max.   :0.8710
##        rm             age              dis              rad
##  Min.   :3.561   Min.   :  2.90   Min.   : 1.130   Min.   : 1.000
##  1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100   1st Qu.: 4.000
##  Median :6.208   Median : 77.50   Median : 3.207   Median : 5.000
##  Mean   :6.285   Mean   : 68.57   Mean   : 3.795   Mean   : 9.549
##  3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188   3rd Qu.:24.000
##  Max.   :8.780   Max.   :100.00   Max.   :12.127   Max.   :24.000
##       tax           ptratio          lstat            medv
##  Min.   :187.0   Min.   :12.60   Min.   : 1.73   Min.   : 5.00
##  1st Qu.:279.0   1st Qu.:17.40   1st Qu.: 6.95   1st Qu.:17.02
##  Median :330.0   Median :19.05   Median :11.36   Median :21.20
##  Mean   :408.2   Mean   :18.46   Mean   :12.65   Mean   :22.53
##  3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:16.95   3rd Qu.:25.00
##  Max.   :711.0   Max.   :22.00   Max.   :37.97   Max.   :50.00
```

No NAs; the range of the values can’t be gauged without further information.

R somehow already removed the IDs, thus we will not need to do this.

1 Normalize your dataset and consider all the variables except MEDV. Create a PCA model and plot it.

First we will normalize the dataset:

```
normalize <- function(x) {
    mean_x = mean(x)
  return ((x - mean_x) / sd(x))
}

boston_df_numeric = lapply(boston_df, as.numeric)
boston_normalized = as.data.frame(lapply(boston_df_numeric[1:13], normalize))
boston_normalized$medv = boston_df$medv
summary(boston_normalized)
```

```
##       crim                 zn               indus              chas
##  Min.   :-0.419367   Min.   :-0.48724   Min.   :-1.5563   Min.   :-0.2723
##  1st Qu.:-0.410563   1st Qu.:-0.48724   1st Qu.:-0.8668   1st Qu.:-0.2723
##  Median :-0.390280   Median :-0.48724   Median :-0.2109   Median :-0.2723
##  Mean   : 0.000000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.007389   3rd Qu.: 0.04872   3rd Qu.: 1.0150   3rd Qu.:-0.2723
##  Max.   : 9.924110   Max.   : 3.80047   Max.   : 2.4202   Max.   : 3.6648
##       nox               rm                age               dis
##  Min.   :-1.4644   Min.   :-3.8764   Min.   :-2.3331   Min.   :-1.2658
##  1st Qu.:-0.9121   1st Qu.:-0.5681   1st Qu.:-0.8366   1st Qu.:-0.8049
##  Median :-0.1441   Median :-0.1084   Median : 0.3171   Median :-0.2790
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.5981   3rd Qu.: 0.4823   3rd Qu.: 0.9059   3rd Qu.: 0.6617
##  Max.   : 2.7296   Max.   : 3.5515   Max.   : 1.1164   Max.   : 3.9566
##       rad               tax              ptratio            lstat
##  Min.   :-0.9819   Min.   :-1.3127   Min.   :-2.7047   Min.   :-1.5296
##  1st Qu.:-0.6373   1st Qu.:-0.7668   1st Qu.:-0.4876   1st Qu.:-0.7986
##  Median :-0.5225   Median :-0.4642   Median : 0.2746   Median :-0.1811
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 1.6596   3rd Qu.: 1.5294   3rd Qu.: 0.8058   3rd Qu.: 0.6024
##  Max.   : 1.6596   Max.   : 1.7964   Max.   : 1.6372   Max.   : 3.5453
##       medv
##  Min.   : 5.00
##  1st Qu.:17.02
##  Median :21.20
##  Mean   :22.53
##  3rd Qu.:25.00
##  Max.   :50.00
```

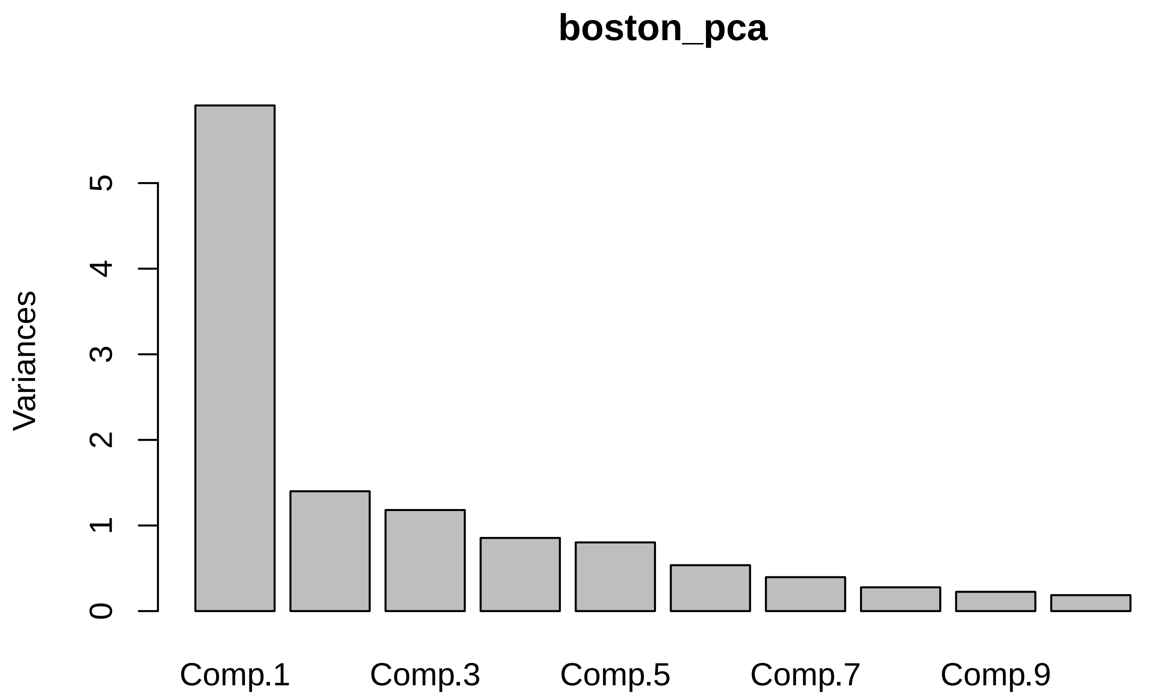
Now we can calculate the PCA:

```
boston_pca <- princomp(boston_normalized[, -ncol(boston_normalized)], cor = TRUE)
summary(boston_pca, loadings = T)
```

```
## Importance of components:
##                           Comp.1    Comp.2     Comp.3     Comp.4    Comp.5
## Standard deviation     2.4304497 1.1832423 1.08659863 0.92436016 0.8957392
## Proportion of Variance 0.4922571 0.1166719 0.09839138 0.07120348 0.0668624
## Cumulative Proportion  0.4922571 0.6089290 0.70732039 0.77852386 0.8453863
##                           Comp.6    Comp.7     Comp.8     Comp.9    Comp.10
## Standard deviation     0.7322458 0.6293692 0.52723411 0.47451400 0.43151563
```

```
## Proportion of Variance 0.0446820 0.0330088 0.02316465 0.01876363 0.01551715
## Cumulative Proportion  0.8900683 0.9230771 0.94624170 0.96500533 0.98052248
##                           Comp.11     Comp.12
## Standard deviation     0.41256242 0.252036760
## Proportion of Variance 0.01418398 0.005293544
## Cumulative Proportion  0.99470646 1.000000000
##
## Loadings:
##         Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9 Comp.10
## crim     0.251  0.274  0.351         0.192  0.760  0.155  0.272         0.109
## zn      -0.266  0.250  0.359  0.194  0.402 -0.295 -0.401  0.374  0.259 -0.268
## indus    0.355                              -0.345  0.174  0.633 -0.374  0.313
## chas           -0.503  0.201  0.811 -0.197
## nox      0.350 -0.233                0.215 -0.208                0.203 -0.136
## rm      -0.196 -0.273  0.561 -0.402 -0.285        -0.327        -0.441
## age      0.323 -0.293        -0.163         0.109 -0.600         0.392  0.456
## dis     -0.331  0.343         0.234                -0.122 -0.162 -0.166  0.690
## rad      0.322  0.231  0.408        -0.119 -0.149        -0.462
## tax      0.342  0.213  0.331               -0.329        -0.168         0.115
## ptratio  0.211  0.393 -0.184  0.109 -0.702        -0.318  0.255  0.126 -0.186
## lstat    0.315  0.128 -0.264  0.185  0.346        -0.425 -0.220 -0.598 -0.255
##         Comp.11 Comp.12
## crim
## zn
## indus   -0.116  -0.251
## chas
## nox      0.807
## rm       0.135
## age     -0.188
## dis      0.402
## rad     -0.115  -0.633
## tax     -0.221   0.721
## ptratio  0.214
## lstat
```

```
plot(boston_pca)
```



As should be the case, component 1 has the highest impact. It is interesting how not all variables impact all the different components.

2. Which predictor variable contributes the most to component 1? And which contributes the least?

This is quite straightforward:

```
comp1_max_contribution = max(abs(boston_pca$loadings[,1]))
comp1_max_contribution
```

```
## [1] 0.3548731
```

```
max_row_name = which(abs(boston_pca$loadings[,1]) == comp1_max_contribution)
max_row_name
```

```
## indus
##     3
```

So indus has the highest impact on PCA's component 1.

```
comp1_min_contribution = min(abs(boston_pca$loadings[,1]))
comp1_min_contribution
```

```
## [1] 0.007588523
```

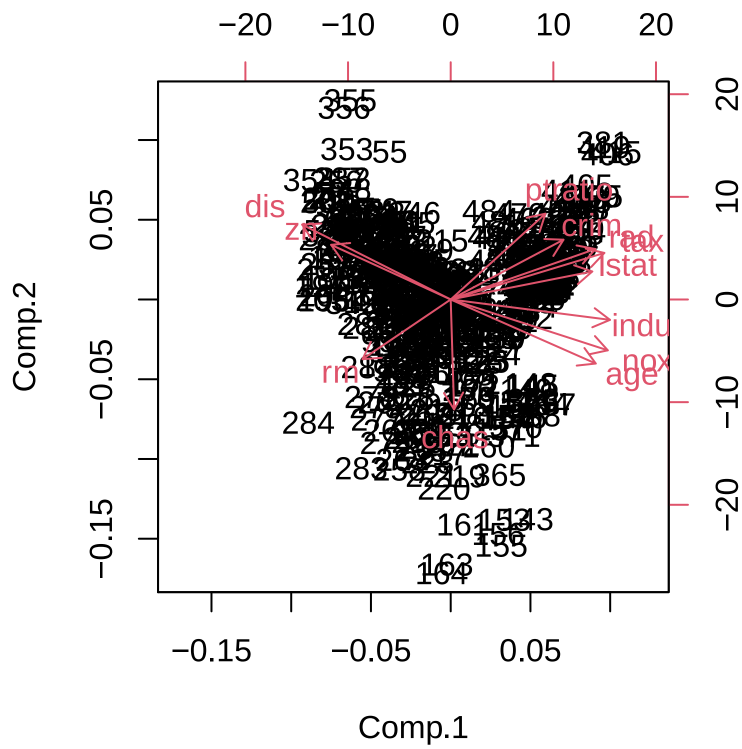
```
min.row.num = which(abs(boston_pca$loadings[,1]) == comp1_min_contribution)
min.row.num
```

```
## chas
##    4
```

The smallest contribution to component 1 is from chas, but I am unsure if this is due to the factorizing or if it worked properly.

With a biplot, we also see that indus has barely the largest contribution to comp 1 and chas the smallest. But of note is that CHAS has the largest contribution to component 2.

```
biplot(boston_pca)
```



−20
−10
0
10
20
Comp.2
0.05
−0.05
−0.15
−0.15
−0.05
0.05
Comp.1
dis
zn
ptratio
crim
rad
tax
lstat
indus
nox
age
rm
chas

3. Estimate the proportion of variance explained by all the components. If we want to explain only 80% of the original data, how many components should we use?

```
summary(boston_pca)
```

```
## Importance of components:
##                           Comp.1    Comp.2     Comp.3     Comp.4    Comp.5
## Standard deviation     2.4304497 1.1832423 1.08659863 0.92436016 0.8957392
## Proportion of Variance 0.4922571 0.1166719 0.09839138 0.07120348 0.0668624
## Cumulative Proportion  0.4922571 0.6089290 0.70732039 0.77852386 0.8453863
##                           Comp.6    Comp.7     Comp.8     Comp.9    Comp.10
## Standard deviation     0.7322458 0.6293692 0.52723411 0.47451400 0.43151563
## Proportion of Variance 0.0446820 0.0330088 0.02316465 0.01876363 0.01551715
## Cumulative Proportion  0.8900683 0.9230771 0.94624170 0.96500533 0.98052248
##                           Comp.11     Comp.12
## Standard deviation     0.41256242 0.252036760
## Proportion of Variance 0.01418398 0.005293544
## Cumulative Proportion  0.99470646 1.000000000
```

We need 5 of the components to boserve at least 80% of the variance in the data.

In general, the standard deviation and proportional variation can be also seen as follows:

```
boston_pca$sdev
```

```
##    Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6    Comp.7    Comp.8
## 2.4304497 1.1832423 1.0865986 0.9243602 0.8957392 0.7322458 0.6293692 0.5272341
##    Comp.9   Comp.10   Comp.11   Comp.12
## 0.4745140 0.4315156 0.4125624 0.2520368
```

```
var.proportion = (boston_pca$sdev)^2/sum((boston_pca$sdev)^2)
var.proportion
```

```
##      Comp.1      Comp.2      Comp.3      Comp.4      Comp.5      Comp.6
## 0.492257148 0.116671856 0.098391382 0.071203475 0.066862395 0.044681998
##      Comp.7      Comp.8      Comp.9     Comp.10     Comp.11     Comp.12
## 0.033008799 0.023164650 0.018763628 0.015517145 0.014183979 0.005293544
```

4. Generate a new dataset using only the components selected in Problem 3. Create a multiple regression model using these components as predictors for the target variable MEDV

Now we divide this into test (30%) and train (70%) data set:

```
# Generate the training and test sets
boston_pca_data = data.frame(boston_pca$scores, medv = boston_df$medv)
train_size = round(nrow(boston_pca_data)*0.7)
train_index = sample( c(1: nrow(boston_pca_data)), train_size)
boston_train = boston_pca_data[train_index,]
boston_test = boston_pca_data[-train_index,]
```

Now we will only use the first 5 components:

```
boston_small = boston_train[,c(1:5,13)]
pca_lm_small = lm(medv ~ ., data = boston_small)
summary(pca_lm_small)
```

```
##
## Call:
## lm(formula = medv ~ ., data = boston_small)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.581  -2.979  -0.806   1.883  33.401
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  22.5063     0.2941  76.527  < 2e-16 ***
## Comp.1       -2.2487     0.1237 -18.184  < 2e-16 ***
## Comp.2       -2.6336     0.2486 -10.593  < 2e-16 ***
## Comp.3        3.0910     0.2743  11.267  < 2e-16 ***
## Comp.4       -1.7467     0.3297  -5.298 2.08e-07 ***
## Comp.5       -1.3761     0.3333  -4.129 4.58e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.519 on 348 degrees of freedom
## Multiple R-squared:  0.6412, Adjusted R-squared:  0.6361
## F-statistic: 124.4 on 5 and 348 DF,  p-value: < 2.2e-16
```

```
MSE_train_lm_small = mean(pca_lm_small$residuals^2)
print(sprintf("MSE Train Small = %f", MSE_train_lm_small))
```

```
## [1] "MSE Train Small = 29.945893"
```

```
boston_small_prediction = predict(pca_lm_small, boston_test)
MSE_test_lm_small = mean(boston_small_prediction^2)
print(sprintf("MSE Test Small = %f", MSE_test_lm_small))
```

```
## [1] "MSE Test Small = 569.174042"
```

Test MSE is worse than Train MSE, so this is not a useful model.

5. Compare the model created in Problem 4 with a multiple regression model using all the components

Let's get to it:

```
pca_lm_full = lm(medv ~ ., data = boston_train)
summary(pca_lm_full)
```

```
##
## Call:
## lm(formula = medv ~ ., data = boston_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.3370  -3.2142  -0.6896   2.2473  24.9651
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 22.44952    0.27171  82.622  < 2e-16 ***
## Comp.1      -2.29097    0.11485 -19.947  < 2e-16 ***
## Comp.2      -2.75957    0.23111 -11.941  < 2e-16 ***
## Comp.3       3.09703    0.25430  12.179  < 2e-16 ***
## Comp.4      -1.77364    0.30557  -5.804 1.48e-08 ***
## Comp.5      -1.53765    0.30836  -4.986 9.81e-07 ***
## Comp.6      -0.04797    0.39284  -0.122  0.90289
## Comp.7       1.10234    0.42830   2.574  0.01048 *
## Comp.8       0.06862    0.52522   0.131  0.89614
## Comp.9       1.76789    0.55495   3.186  0.00158 **
## Comp.10     -1.43454    0.63560  -2.257  0.02464 *
## Comp.11     -3.94167    0.66052  -5.968 6.04e-09 ***
## Comp.12     -3.54407    1.08787  -3.258  0.00124 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.084 on 341 degrees of freedom
## Multiple R-squared:  0.7017, Adjusted R-squared:  0.6912
## F-statistic: 66.84 on 12 and 341 DF,  p-value: < 2.2e-16
```

```
MSE_train_lm_full = mean(pca_lm_full$residuals^2)
print(sprintf("MSE Train Full = %f", MSE_train_lm_full))
```

```
## [1] "MSE Train Full = 24.900369"
```

```
boston_full_prediction = predict(pca_lm_full, boston_test)
MSE_test_lm_full = mean(boston_full_prediction^2)
print(sprintf("MSE Test Full = %f", MSE_test_lm_full))
```

```
## [1] "MSE Test Full = 574.226813"
```

Welp, this model is also not all that useable. Slightly better perfromance than the small one, but still not all that useable in the real world.

Nonetheless, the full model does not perform all that much better than the small one, which should be noted.