Lab 3: Principal Components Analysis (PCA)

Prof. Gaston Sanchez

Stat 154, Fall 2017

Introduction

The goal of this lab is to go over the various options and steps required to perform a Principal Components Analysis (PCA). You will also learn about the functions `prcomp()` and `princomp()`, and how to use their outputs to answer questions like:

- How many principal components to retain.
- How to visualize the observations.
- How to visualize the relationships among variables.
- How to visualize supplementary variables.

Dataset Wholesale Customers

In this lab we are going to use a Wholesale Customers Data Set, kindly contributed to the UCI Machine Learning Repository by Margarida Cardoso. More info at:

https://archive.ics.uci.edu/ml/datasets/Wholesale+customers

```
dat <- read.csv('../../data/wholesale.csv')
```

You can find a csv data file `wholesale.csv` in the `data/` folder of the course github repository. Alternatively, you can also download the file directly from the UCI machine learning website:

```
# assemble the url so it fits on rendered pdf
uci_ml <- 'https://archive.ics.uci.edu/ml/machine-learning-databases'
csv_file <- '/00292/Wholesale%20customers%20data.csv'
url <- paste0(uci_ml, csv_file)

# download file to your working directory
# (note: the field separator is a colon ";")
download.file(url, destfile = 'wholesale.csv')
```

The first 5 rows of the data set look like:

```
head(dat, n = 5)
```

```
  Channel Region Fresh Milk Grocery Frozen Detergents_Paper Delicassen
1       2      3 12669 9656    7561    214              2674       1338
2       2      3  7057 9810    9568   1762              3293       1776
```

```
3       2       3  6353 8808    7684    2405              3516        7844
4       1       3 13265 1196    4221    6404               507        1788
5       2       3 22615 5410    7198    3915              1777        5185
```

As you can tell, there are eight variables: the first two are categorical, and the rest are quantitative. All quantitative variables represent annual spending in monetary units (m.u.) on diverse product categories. Below is a brief description of the variables, and their type of scale in parenthesis:

- `Channel`: 1 = Horeca (Hotel/Restaurant/Cafe), 2 = Retail (Nominal)
- `Region`: 1 = Lisbon, 2 = Oporto, 3 = Other (Nominal)
- `Fresh`: annual spending (m.u.) on fresh products (Continuous)
- `Milk`: annual spending (m.u.) on milk products (Continuous)
- `Grocery`: annual spending (m.u.) on grocery products (Continuous)
- `Frozen`: annual spending (m.u.) on frozen products (Continuous)
- `Detergents_Paper`: annual spending (m.u.) on detergents and paper products (Continuous)
- `Delicatessen`: annual spending (m.u.) on and delicatessen products (Continuous)

Your turn

Once you've imported the data in R:

- Check the structure of the data: `str()`
- Because `Channel` and `Region` are actually categorical variables, convert them to factors:
  - `Channel`: 1 = Horeca, 2 = Retail
  - `Region`: 1 = Lisbon, 2 = Oporto, 3 = Other
- After converting `Channel` and `Region` to factors your data (first rows) should look like:

```r
dat$Channel <- factor(
  dat$Channel,
  levels = 1:2,
  labels = c('Horeca', 'Retail'))

dat$Region <- factor(
  dat$Region,
  levels = 1:3,
  labels = c('Lisbon', 'Oporto', 'Other'))

head(dat, n = 5)
```

```
  Channel Region Fresh Milk Grocery Frozen Detergents_Paper Delicassen
1  Retail  Other 12669 9656    7561    214              2674        1338
2  Retail  Other  7057 9810    9568   1762              3293        1776
3  Retail  Other  6353 8808    7684   2405              3516        7844
4  Horeca  Other 13265 1196    4221   6404               507        1788
```

```
5  Retail  Other 22615 5410    7198   3915              1777       5185
```

- Confirm that `Channel` and `Region` have the following frequencies:

```
table(dat$Channel)
```

```
Horeca Retail
   298    142
```

```
table(dat$Region)
```

```
Lisbon Oporto  Other
    77     47    316
```

Perform an exploratory data analysis, for instance:

- get descriptive statistics with `summary()`.
- visually inspect quantitative variables: boxplots, histograms, density curves.
- compute the correlation matrix of quantitative variables.
- get a scatterplot matrix for the quantitative variables with `pairs()`, or even better with `ggpairs()` from the package `GGally`.
- What do you see in each scatterplot (e.g. pattern, trend, variability)?

R Functions for PCA

R provides two built-in functions to perform Principal Components Analysis:

1. `prcomp()`
2. `princomp()`

Both functions are part of the `"stats"` package which comes in the default distribution of R. In addition to `prcomp()` and `princomp()`, there are various packages (e.g. `"FactoMineR"`, `"ade4"`, etc.) that provide other functions for PCA. In this lab, we will only discuss the base functions.

PCA with `prcomp()`

The main input of `prcomp()` is a data frame or a data matrix. In order to perform PCA on standardized data (mean = 0, variance = 1), we use the argument `scale. = TRUE`.

Let's apply `prcomp()` on the quantitative variables—these are the *active* variables. We'll use the categorical variables as *supplementary* variables. Although all the quantitative variables are supposedly measured on the same scale (monetary units), it is usually preferable to standardize the data so that no single variable dominates the output because of the magnitude of its variability.

```r
# PCA with prcomp()
pca_prcomp <- prcomp(dat[ ,3:8], scale. = TRUE)

# what's in a prcomp object?
names(pca_prcomp)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

The object `pca_prcomp` is an object of class `"prcomp"`, basically a list that contains the following results:

- `sdev` corresponds to the standard deviations of the principal components.
- `rotation` is the rotation matrix or loadings (i.e. eigenvectors of $\frac{1}{n-1}\mathbf{X}^\top\mathbf{X}$)
- `center` is the vector of means of the raw data.
- `scale` is the vector of standard deviations of the raw data.
- `x` is the matrix of principal components.

As we saw in lecture, the minimal output of a PCA procedure should consists of eigenvalues, loadings, and principal components:

The eigenvalues, $(\lambda_1, \lambda_2, \ldots)$, are the squared `sdev`:

```r
# eigenvalues
eigenvalues <- pca_prcomp$sdev^2
eigenvalues
```

```
## [1] 2.64497357 1.70258397 0.74006477 0.56373023 0.28567634 0.06297111
```

The reason why these are called `sdev` has to do with the variance of each principal component: $\lambda_k = var(\mathbf{z_k})$. The elements in `sdev` are simply the standard deviations of the PCs: $\sqrt{\lambda_k} = sd(\mathbf{z_k})$

The loadings or PC weights, $\mathbf{V}$, are in `rotation`:

```r
# loadings or weights
loadings <- pca_prcomp$rotation
round(loadings, 3)
```

```
##                     PC1    PC2    PC3    PC4    PC5    PC6
## Fresh            -0.043 -0.528 -0.812 -0.237  0.049  0.036
## Milk             -0.545 -0.083  0.060 -0.087 -0.827  0.038
## Grocery          -0.579  0.146 -0.108  0.106  0.315 -0.722
## Frozen           -0.051 -0.611  0.178  0.769  0.028  0.016
## Detergents_Paper -0.549  0.255 -0.136  0.172  0.340  0.686
## Delicassen       -0.249 -0.504  0.524 -0.552  0.315  0.075
```

And the principal components, aka scores, $\mathbf{Z} = \mathbf{XV}$, are in the object `x`:

```r
# scores or principal components
scores <- pca_prcomp$x
round(head(scores, 5), 3)
```

```
##         PC1    PC2    PC3    PC4    PC5    PC6
## [1,] -0.193  0.305 -0.141 -0.486 -0.495  0.007
## [2,] -0.434  0.328  0.319 -0.179 -0.365 -0.055
## [3,] -0.810 -0.814  1.522 -1.253  0.379  0.277
## [4,]  0.778 -0.652  0.163  0.380  0.276 -0.061
## [5,] -0.166 -1.270  0.066 -0.825  0.394  0.027
```

Before discussing what to do with the output of a PCA, let's quickly talk about the other function `princomp()`.

PCA with `princomp()`

Another function to perform PCA is `princomp()`. The main input is a data frame or a data matrix. In order to perform PCA on standardized data (mean = 0, variance = 1), we use the argument `cor = TRUE`, which means that the analysis is performed using the correlation matrix.

```r
# PCA with princomp()
pca_princomp <- princomp(dat[ ,3:8], cor = TRUE)

# what's in a princomp object?
names(pca_princomp)
```

```
## [1] "sdev"     "loadings" "center"   "scale"    "n.obs"    "scores"
## [7] "call"
```

The object `pca_princomp` is an object of class `"princomp"`, basically a list that contains various results:

- `sdev` corresponds to the standard deviations of the principal components.
- `loadings`: object of class `"loadings"`
- `center`: vector of variable means of the raw data.
- `scale`: vector of standard deviations of the raw data.
- `n.obs`: number of observations in the data.
- `scores`: matrix of principal components.
- `call`: function call.

The eigenvales, PCs and loadings are:

```r
# eigenvalues
pca_princomp$sdev^2
```

```
##     Comp.1     Comp.2     Comp.3     Comp.4     Comp.5     Comp.6
## 2.64497357 1.70258397 0.74006477 0.56373023 0.28567634 0.06297111
```

```
# scores or principal components
round(head(pca_princomp$scores, 5), 3)
```

```
##      Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
## [1,] -0.193  0.305  0.141  0.486 -0.495 -0.007
## [2,] -0.434  0.328 -0.319  0.179 -0.366  0.055
## [3,] -0.811 -0.815 -1.523  1.254  0.379 -0.278
## [4,]  0.779 -0.653 -0.163 -0.380  0.276  0.061
## [5,] -0.166 -1.271 -0.066  0.826  0.394 -0.027
```

```
# loadings or weights
pca_princomp$loadings
```

```
##
## Loadings:
##                  Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
## Fresh                   -0.528  0.812  0.237
## Milk             -0.545                      -0.827
## Grocery          -0.579  0.146  0.108 -0.106  0.315  0.722
## Frozen                  -0.611 -0.178 -0.769
## Detergents_Paper -0.549  0.255  0.136 -0.172  0.340 -0.686
## Delicassen       -0.249 -0.504 -0.524  0.552  0.315
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.167  0.167  0.167  0.167  0.167  0.167
## Cumulative Var  0.167  0.333  0.500  0.667  0.833  1.000
```

If you carefully look at the loadings, you should notice that some values are left in blank. Why is this? Check the documentation `?princomp`

Challenge: Notice that `pca_princomp$loadings` is an object of class `"loadings"`. How would you retrieve just the matrix of loadings—like the one below?

```
round(unclass(pca_princomp$loadings), 4)
```

```
                  Comp.1  Comp.2  Comp.3  Comp.4  Comp.5  Comp.6
Fresh            -0.0429 -0.5279  0.8123  0.2367  0.0487 -0.0360
Milk             -0.5451 -0.0832 -0.0604  0.0872 -0.8266 -0.0380
Grocery          -0.5793  0.1461  0.1084 -0.1060  0.3150  0.7217
Frozen           -0.0512 -0.6113 -0.1784 -0.7687  0.0279 -0.0156
Detergents_Paper -0.5486  0.2552  0.1362 -0.1717  0.3396 -0.6859
Delicassen       -0.2487 -0.5042 -0.5239  0.5521  0.3147 -0.0751
```

Important Note

The signs of the columns of the loadings and scores are arbitrary, and so may differ between different programs for PCA, and even between different builds of R.

Look around at the output of your neighbors to see who has similar results to yours, and who has different outputs.

Your turn

What are the differences between `prcomp()` and `princomp()`? Spend some time reading the help documentation of both functions to find out the main differences between them. Are there any cases when it would be better to use one function or the other?

Stages of a Principal Components Analysis

The primary goal in PCA is to summarize the systematic patterns of variation in a data set, which is done via the principal components (PCs). Derived from this idea, the most common uses of the PCA results is to visualize multivariate data and/or to perform a dimension reduction (for other analytical purposes).

Eigenvalues and Proportion of Variance Explained

The first step when examining the results of a PCA is to look at how much variability is captured by each PC. This is done by examining the eigenvalues.

Your turn

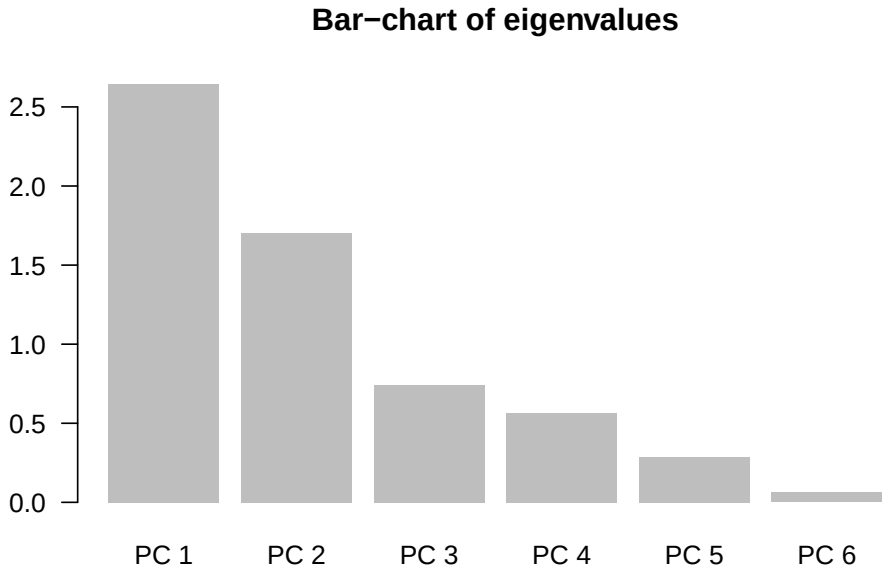
Compute a table containing the eigenvalues, the variance in terms of percentages, and the cumulative percentages, like the table below. Analysts typically look at a bar-chart of the eigenvalues (see figure below). Plot your own bar-chart.

```
eigs <- eigenvalues
eigs_perc <- 100 * eigs / sum(eigs)
eigs_cum <- cumsum(eigs_perc)

eigs_df <- data.frame(
  eigenvalue = eigs,
  percentage = eigs_perc,
  'cumulative percentage' = eigs_cum
```

```
)

print(round(eigs_df, 4), print.gap = 2)
```

```
   eigenvalue  percentage  cumulative.percentage
1      2.6450     44.0829                44.0829
2      1.7026     28.3764                72.4593
3      0.7401     12.3344                84.7937
4      0.5637      9.3955                94.1892
5      0.2857      4.7613                98.9505
6      0.0630      1.0495               100.0000
```

```
barplot(eigs, border = NA, las = 1, names.arg = paste('PC', 1:6),
        main = 'Bar-chart of eigenvalues')
```



- How much of the variation in the data is captured by the first PC?
- How much of the variation in the data is captured by the second PC?
- How much of the variation in the data is captured by the first two PCs?

Choosing the number of components

A related concern is to be able to determine how many PCs should be retained. There are various strategies:

- Retain just enough components to explain some specified, large percentage of the total variation of the original variables. For example, how many PCs would you retain to capture 70% of the total variation?

- Exclude those PCs whose eigenvalues are less than the average: $\sum_{h=1}^{p} \lambda_h / p$. Using this criterion, how many PCs would you retain?
- When the PCs are extracted from the correlation matrix, like in this case, the average eigenvalue is one; components with eigenvalues less than one are therefore excluded. This rule is known as *Kaiser's rule*. Using this criterion, how many PCs would you retain?
- Jollife (1972) proposed a variation of the Kaiser's rule: exclude PCs whose eigenvalues are less than 0.7. Under this criterion: how many PCs would you retain?
- Cattel (1965) suggests plotting the eigenvalues with the so-called *scree-plot* or *scree diagram*—like the bar-chart of eigenvalues. Cattel suggests looking for an "elbow" in the diagram, this would correspond to a point where "large" eigenvalues cease and "small" eigenvalues begin. With this rule, how many PCs would you retain?

Variable Loadings and Correlations with PCs

The next stage consists of examining how PCs are formed. Recall that PCs are linear combinations of the input variables, in which the coefficients of such linear combinations are given by the loadings. A starting point is to check the matrix of loadings. The larger the loading of a variable in a given PC, the more associated the variable is with that PC.

Another way to examine how variables are associated with the PCs is to look at their correlations.

Your turn

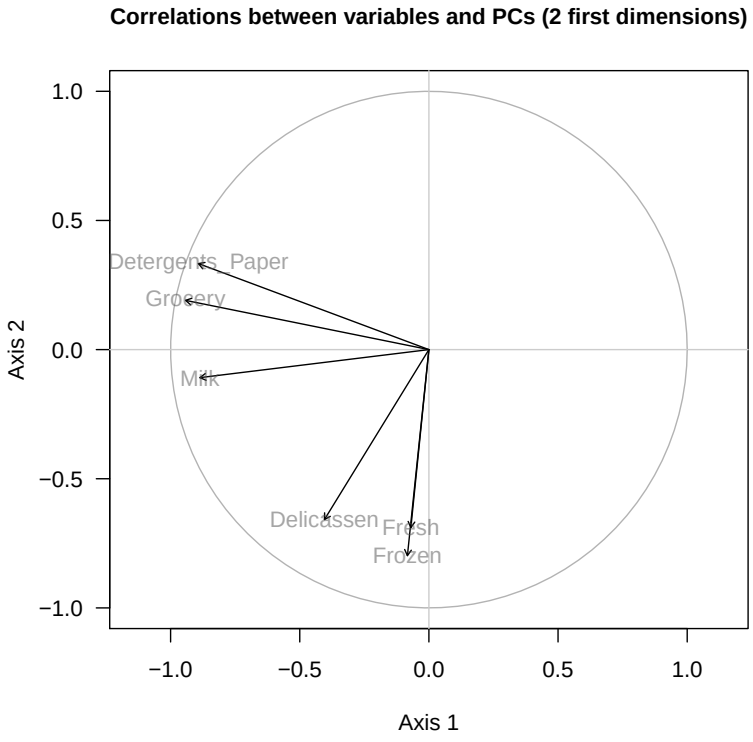
Calculate the correlations between the active variables and the PCs. You should get a matrix like the one below:

```
pc_cors <- cor(dat[ ,3:8], scores)
round(pc_cors, 4)
```

```
                    PC1     PC2     PC3     PC4     PC5     PC6
Fresh           -0.0697 -0.6889 -0.6988 -0.1777  0.0260  0.0090
Milk            -0.8865 -0.1085  0.0519 -0.0655 -0.4418  0.0095
Grocery         -0.9421  0.1906 -0.0932  0.0796  0.1684 -0.1811
Frozen          -0.0832 -0.7976  0.1535  0.5771  0.0149  0.0039
Detergents_Paper -0.8923  0.3330 -0.1172  0.1289  0.1815  0.1721
Delicassen      -0.4044 -0.6579  0.4507 -0.4145  0.1682  0.0189
```

Less common, but extremely helpful, is to use the columns of the correlation matrix to visualize how variables are associated with the PCs.

```r
# function to create a circle
circle <- function(center = c(0, 0), npoints = 100) {
  r = 1
  tt = seq(0, 2 * pi, length = npoints)
  xx = center[1] + r * cos(tt)
  yy = center[1] + r * sin(tt)
  return(data.frame(x = xx, y = yy))
}
corcir <- circle(c(0, 0), npoints = 100)

plot(pc_cors[ ,1:2], type = 'n', las = 1, xlim = c(-1, 1), ylim = c(-1, 1),
     xlab = "Axis 1", ylab = "Axis 2", asp = 1)
title("Correlations between variables and PCs (2 first dimensions)",
      cex.main = 1)
lines(corcir, col = "gray70")
abline(h = 0, v = 0, col = "gray80")
arrows(rep(0, nrow(pc_cors)), rep(0, nrow(pc_cors)), pc_cors[,1], pc_cors[,2],
       length = 0.05)
text(pc_cors[ ,1], pc_cors[ ,2], rownames(pc_cors), col = "#55555588")
```



This graph is known as the *circle of correlations*. The variables are plotted as arrows, with the length of the arrow equal to the correlation coefficient. The closer the arrowhead to the circumference (of radious 1), the better its representation with the associated PCs. Try to plot a graph like the one above.

Your turn

With either the loadings, or the correlations between variables and PCs, analysts typically try to give labels to the PCs. Looking at how variables are associated with the PCs:

- What label—and interpretation—would you give to PC1?
- What label—and interpretation—would you give to PC2?
- What label—and interpretation—would you give to PC3?
- What does it mean that all the correlations with PC1 have the same sign?
- What does it mean that in PC2 some loadings are positive and the others negative?

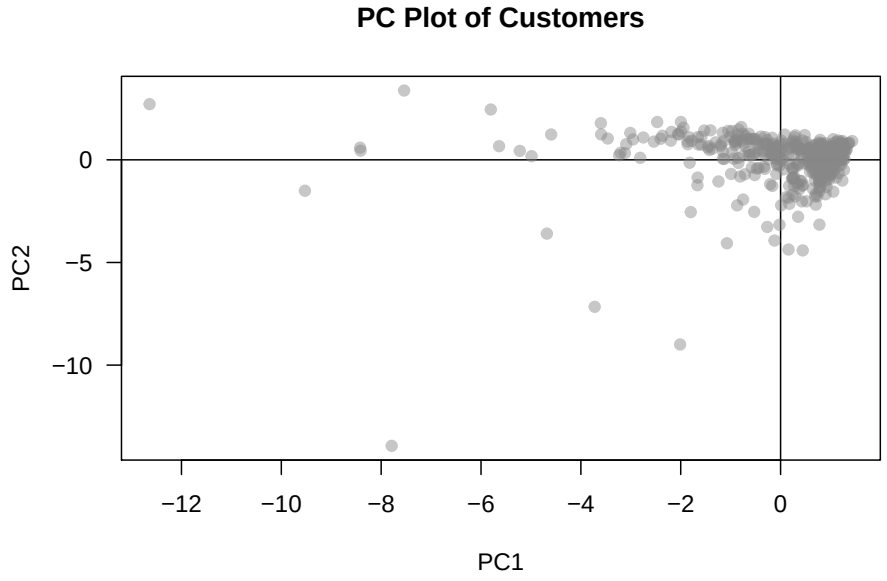
Visualizing observations

The next stage involves visualizing the observations in a low dimensional space. This is typically achieved with scatterplots of the components.

Your turn

Begin with a scatterplot of the first two PCs (see figure below).

```
plot(pca_prcomp$x, type = "n", las = 1)
abline(h = 0, v = 0)
points(pca_prcomp$x[ ,1], pca_prcomp$x[ ,2], pch = 19,
       col = "#88888877")
title(main = "PC Plot of Customers")
```



- Also plot PC1 - PC3, and then plot PC2 - PC3. If you want, continue visualizing other scatterplots.
- What patterns do you see?

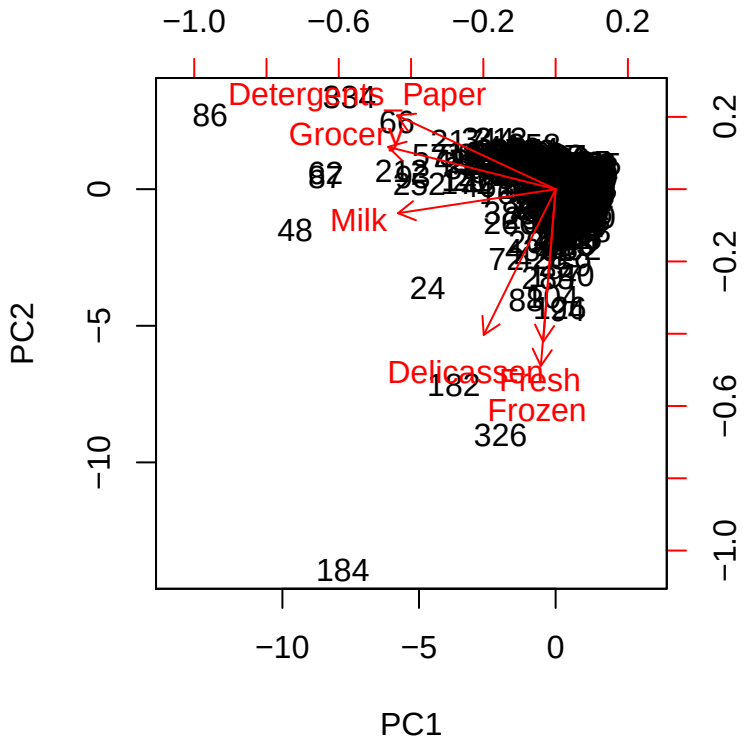
- Try adding numeric labels to the points to see which observations seem to be potential outliers.

Biplot: Another visual display

A fourth (optional) stage consists of obtaining a simultaneous visual display of both the observations and the variables in a low dimensional space. This is done with the so-called **biplot**, originally proposed by Ruben Gabriel in 1971, and available in R with the homonym function `biplot()`.

Some authors are opposed to this type of visualization arguing that it is a fictitious display. Personally, I'm not opposed to this type of display, as long as you know how to read it. The important thing to keep in mind is that, in a biplot, you are superimposing two different low-dimensional spaces: one corresponds to the observations, and the other corresponds to the variables. Also, because you are superimposing two clouds of objects, typically the scale of one—or both—of them will be distorted.

```
# biplot
biplot(pca_prcomp, scale = 0)
```



The `scale = 0` argument to `biplot()` ensures that the arrows are scaled to represent the loadings, while the PCs are scaled to unit variance; also, when specifying `scale = 0` the distances between the observations correspond to Mahalanobis distances. Other values for `scale` give slightly different biplots with different scaling distortions.

Your turn: Graph various `biplot()`'s with different values of `scale` (e.g. 0, 0.3, 0.5, 1).

How do the relative positions of the arrows change with respect to the points? Under which scale value you find it easier to read the biplot?

Plotting Supplementary Variables

In this example, we are treating the categorical variables `Channel` and `Region` as supplmentary variables, meaning that they do NOT intervene in the construction of the PCs. However, when you have supplementary variables, it is interesting to plot them.

With categorical supplementary variables, we represent them on the plot of observations through their categories. In other words, calculate the vector of PCs means for each categorical variable.

Your turn

Obtain the vectors with the means of PCs for each category of the supplementary variables `Channel` and `Region`. Confirm your results with the following values:

```
scores_channel <- as.data.frame(scores)
scores_channel$Channel <- dat$Channel

sup_channel <- scores_channel %>%
  group_by(Channel) %>%
  summarise_each(funs(mean))
```

```
`summarise_each()` is deprecated.
Use `summarise_all()`, `summarise_at()` or `summarise_if()` instead.
To map `funs` over all variables, use `summarise_all()`
```

```
sup_channel[,1:3]
```

```
# A tibble: 2 x 3
  Channel       PC1        PC2
   <fctr>     <dbl>      <dbl>
1  Horeca  0.654399 -0.2741028
2  Retail -1.373316  0.5752299
```

```
scores_region <- as.data.frame(scores)
scores_region$Region <- dat$Region

sup_region <- scores_region %>%
  group_by(Region) %>%
  summarise_each(funs(mean))
```

```
`summarise_each()` is deprecated.
Use `summarise_all()`, `summarise_at()` or `summarise_if()` instead.
```

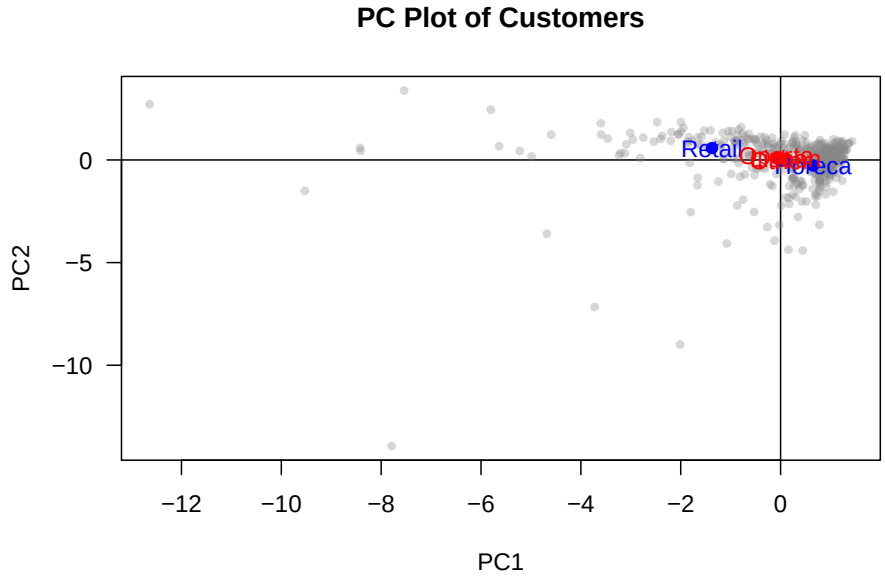
```
To map `funs` over all variables, use `summarise_all()`
```

```
sup_region[,1:3]
```

```
# A tibble: 3 x 3
  Region         PC1         PC2
  <fctr>       <dbl>       <dbl>
1 Lisbon  0.10160060  0.05964504
2 Oporto -0.08859145  0.10151168
3  Other -0.01158053 -0.02963202
```

Now, use the obtained PC means of each category to plot them on the chart of observations, something like this:

```
plot(pca_prcomp$x, type = "n", las = 1)
abline(h = 0, v = 0)
points(pca_prcomp$x[ ,1], pca_prcomp$x[ ,2], pch = 20,
       col = "#88888855")
points(sup_channel$PC1, sup_channel$PC2, pch = 19,
       col = "blue")
points(sup_region$PC1, sup_region$PC2, pch = 19,
       col = "red")
text(sup_channel$PC1, sup_channel$PC2, labels = sup_channel$Channel, col = "blue")
text(sup_region$PC1, sup_region$PC2, labels = sup_region$Region, col = "red")
title(main = "PC Plot of Customers")
```



Contributions of Individuals

Looking at the PC plot of customers, you should be able to tell that most of the points are concentrated aroung the origin, with a handful of customers further away. These seem to be influential observations that are driving the results. It is important to assess the contributions of all observations, and determine whether it would be better to remove outliers.

Let $\mathbf{Z}$ denote the matrix of principal components or scores. You can calculate the contribution for the i-th individual on the k-th PC as:

$$Ctr_k(i) = \frac{z_{ik}^2}{\sum_{i=1}^{n} z_{ik}^2} = \frac{z_{ik}^2}{\lambda_k}$$

where:

- z_{ik} is the value of individual i on the k-th PC
- λ_k is the eigenvalue associated to the k-th PC

Your turn

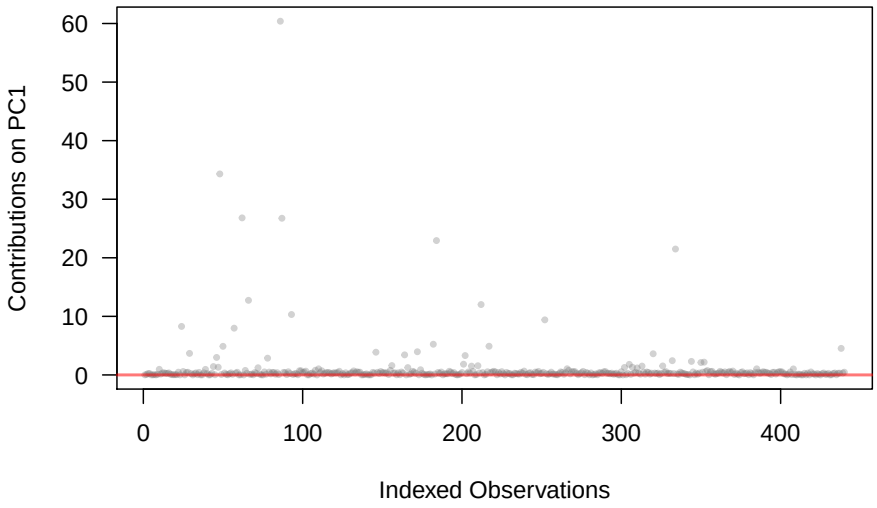
Compute the contributions of the observations on all PCs. You can either create a matrix of contributions or a data.frame. This is what the first five rows of `contributions` on all 6 PCs should look like:

```
contribs <- sweep(scores^2, 2, eigenvalues, FUN = "/")
head(contribs, n = 5)
```

```
            PC1         PC2          PC3         PC4       PC5           PC6
[1,] 0.01409327 0.05454910 0.026756620 0.41877812 0.8567256 0.0008709474
[2,] 0.07118851 0.06320377 0.137196013 0.05660086 0.4667677 0.0471854145
[3,] 0.24819076 0.38933237 3.128808032 2.78350544 0.5018092 1.2204421228
[4,] 0.22870342 0.24969051 0.035824712 0.25564956 0.2663107 0.0584115578
[5,] 0.01043058 0.94730701 0.005922419 1.20820043 0.5427431 0.0114008784
```

The main question regarding contributions is: *What is the proportion that each individual contribute to a principal component?* If all observations had the same influence, then each observation should contribute around $1/439 = 0.0023$ or 0.23%. So, how do contributions for PC1 look like compare to the reference value 0.23%? To answer this question, you can get a plot of contributions for PC1

```
plot(contribs[,1], type = 'n', las = 1,
     xlab = "Indexed Observations", ylab = "Contributions on PC1")
points(1:nrow(contribs), contribs[,1], pch = 19, cex = 0.5, col = "#77777755")
abline(h = 1/439, col = '#FF000088', lwd = 2)
```



Notice that most observations have a contribution close to 0.0023 (indicated with the red line) but there are some that have considerably large contributions. You may ask: what is considered an elevated contribution value? Unfortunately, the answer is not simple. We could say that an observation has a large contribution when its value is unusually large compared to the others. One way to roughly determine a threshold—and be able to draw the line—is to look at the 80-th or 90-th quantile of contributions. For instance, confirm that the 90-th percentile of the contributions is:

```
apply(contribs, 2, quantile, probs = 0.9)
```

```
##       PC1       PC2       PC3       PC4       PC5       PC6
## 1.040497 1.173247 1.808378 1.408796 1.434097 2.089999
```

Your turn

- Remove those observations that have a contribution of 10 or more on PC1.
- Rerun a PCA and go through all the stages that we've discussed so far.