PCA Analyses

Daniel J Carter

18 Feb 2018

Introduction

This document contains the code for reproducing the PCA found for the beers example in the PCA lecture for the Social Epidemiology course.

This code can run in RStudio which is freely available to download, or is on the school computers. Open a new R notebook or R script to copy the code into. To run a command in R, press ctrl-Enter with your cursor anywhere in the line you wish to run.

Load the appropriate libraries for this analysis and install the packages if necessary. To install the packages, you must uncomment the lines by removing the #. R packages are extensions to basic R that have been written by users to do common or useful tasks.

```
# install.packages("ggplot2")
# install.packages("dplyr")
# install.packages("devtools")
library(ggplot2)
library(dplyr)
library(devtools)

# install_github("vqv/ggbiplot")
library(ggbiplot)
```

Read in the dataset. Be sure to change the path to wherever you have stored the data on your computer.

This dataset contains 105 beers from Dieu du Ciel in Montreal. Data included in this small dataset include the name of the beer, its alcohol percentage, its average crowdsourced rating on RateBeer.com, its average crowdsourced rating for its style, the number of ratings it has received, and the style of the beer.

```
beers <- read.csv("D:/LSHTM/Teaching/beers.csv", header = F)
# read in the dataset as a data frame - data frames are like tables, they
# are the basic unit of analysis in R

colnames(beers) <- c("Name", "ABV", "Rating", "StyleRating", "N", "Style")

# The '<-' is the assignment operator in R, it is equivalent to an equals
# sign
```

We see from the summary that some of the beer styles have low numbers so we combine them with similar styles. Barley Wines, IPAs and Saisons all go under Ale, and Smoked beers go under Wheat.

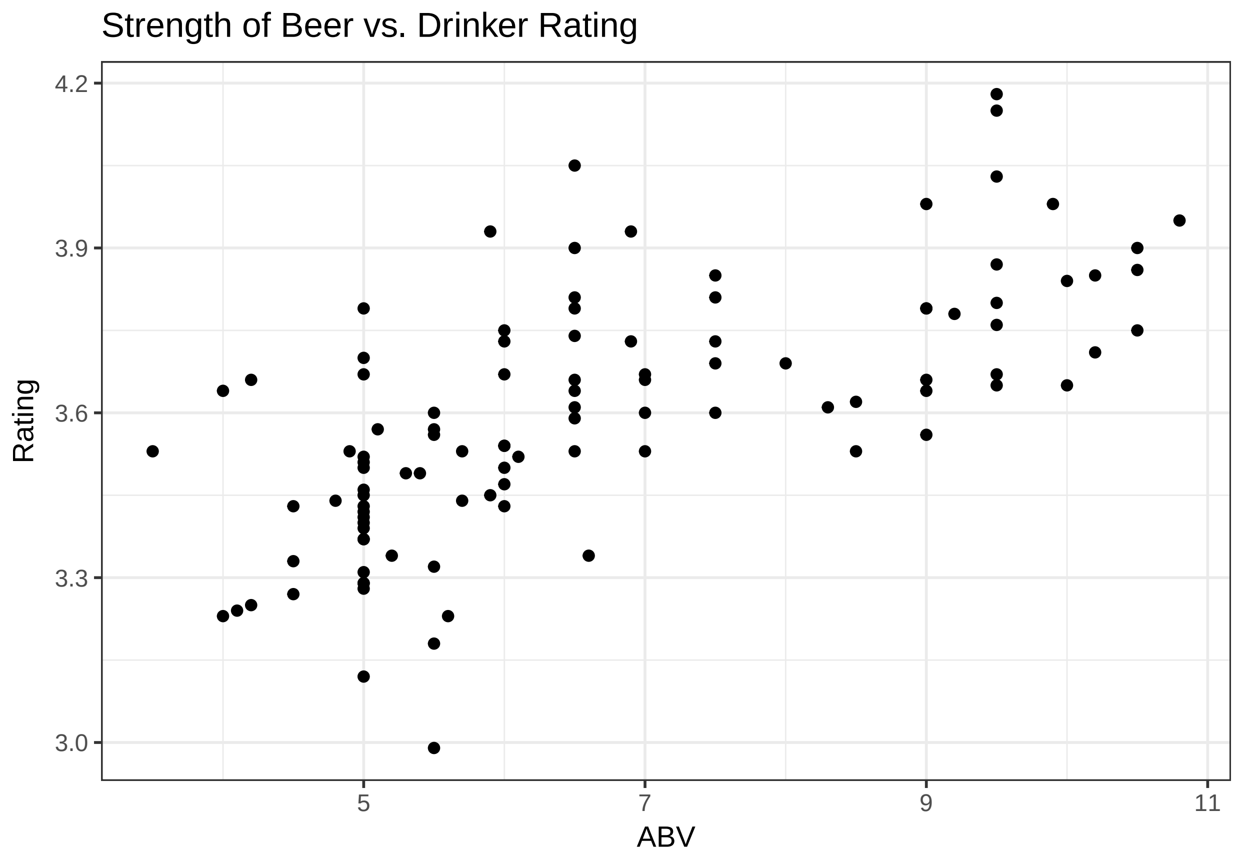
```
summary(beers$Style)
```

```
##          Ale Barley Wine     Belgian         IPA       Lager      Saison
##           32           2          10          11          10           8
##       Smoked       Stout       Wheat
##            5          15          12
```

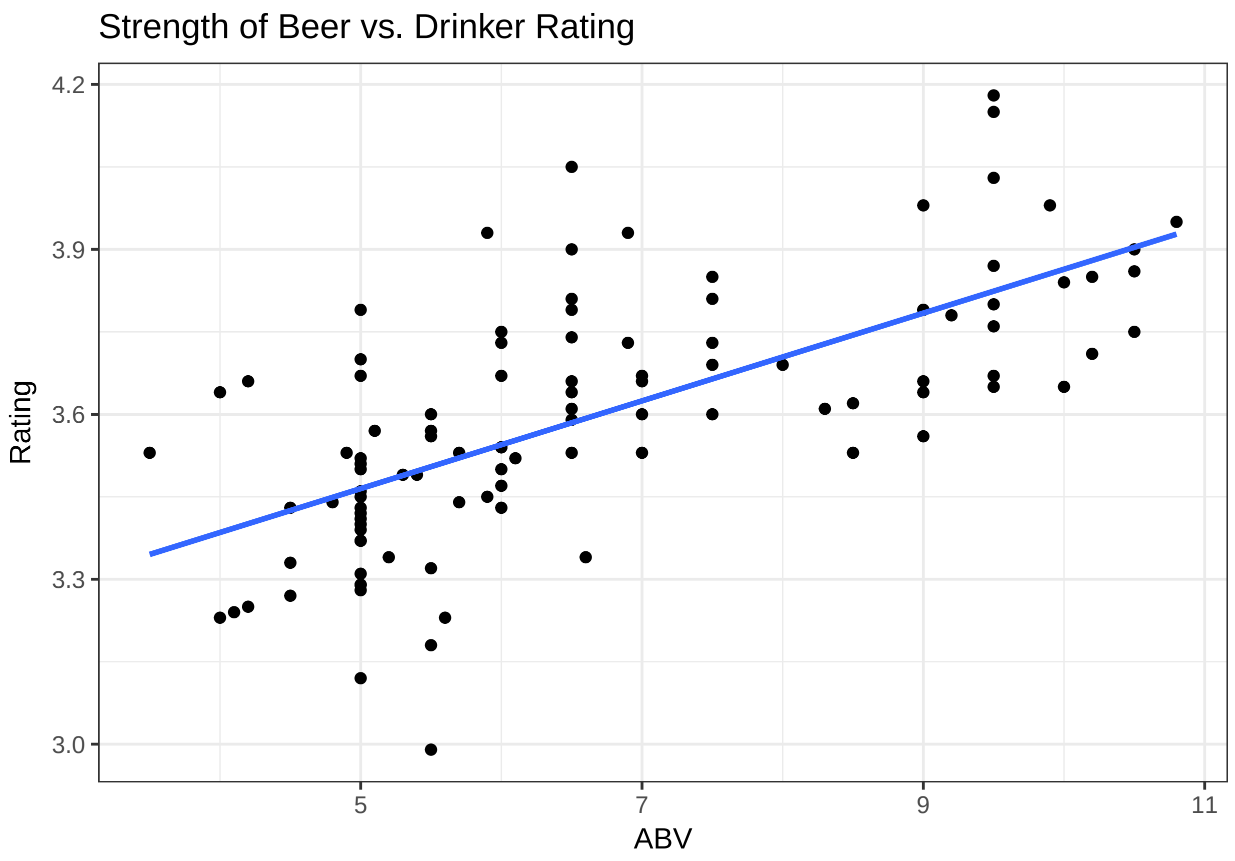
```r
beers$Style[beers$Style == "Barley Wine"] <- "Ale"
beers$Style[beers$Style == "IPA"] <- "Ale"
beers$Style[beers$Style == "Saison"] <- "Ale"

beers$Style[beers$Style == "Smoked"] <- "Wheat"


# To refer to a particular column of a particular dataset we use the dollar
# sign like so: data$column

# Square brackets indicate a subset of the data or column:
# data$column[subset] e.g. the command beers$Style[beers$Style == 'Barley
# Wine'] takes the subset of rows where the column 'Style' in dataset beers
# is 'Barley Wine' and replaces all the values in that column with 'Ale' by
# using the assignment operator '<-'
```

We produce a scatter plot of strength of beer (measures in alcohol by volume, ABV %) vs. the overall rating.

```r
ggplot(beers, aes(x = ABV, y = Rating)) + # 'aes'(thestics) sets the axes
  geom_point() + # geom_point says it will be a scatter plot
  labs(title = "Strength of Beer vs. Drinker Rating") + # add a title label
  theme_bw() # plot on a white background for visibility
```



```r
ggplot(beers, aes(x = ABV, y = Rating)) +
  geom_point() + #
  labs(title = "Strength of Beer vs. Drinker Rating") +
  geom_smooth(method = lm, se = F) +
  theme_bw()
```



```
## add a line using the lm (linear model) method using the x and y variables from 'aes'
```

We conclude that as ABV increases, so does the average rating of the beer.

Underlying Process

We now go through the underlying steps of actually computing a PCA - although you should take the time to understand what the analyses you apply actually do. . . if you just want to know how to do it, you can skip this section.

Step 1: Standardise

The variables of interest now have mean 0 and variance 1. This is to prevent the scale of measurement of the variables obscuring the variability in the dataset.

```r
beers$ABV.standard <- scale(beers$ABV, scale = T)
beers$Rating.standard <- scale(beers$Rating, scale = T)
# the scale function in R gives mean 0, variance 1 to the new variable
```

```
round(mean(beers$ABV.standard), 2)  # mean (rounded to 2 decimal places)
```

```
## [1] 0
```

```
round(sd(beers$ABV.standard), 2)  # SD = sqrt(variance)
```

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

Step 2: Find the Variance-Covariance Matrix

We obtain the covariance of Rating and ABV (i.e. how much do the two variables change together?) and put this information into a matrix form. The matrix below displays how the variables of interest vary with one another.

	Rating	ABV
Rating	0.674	1.000
ABV	1.000	0.674

```
cov <- cov(beers$ABV.standard, beers$Rating.standard)
# finds the covariance between the two variables = 0.67357

vcov <- matrix(c(cov, 1, 1, cov), nrow = 2, ncol = 2)
# puts the values into a matrix
```

Step 3: Find the eigenvectors and rotate

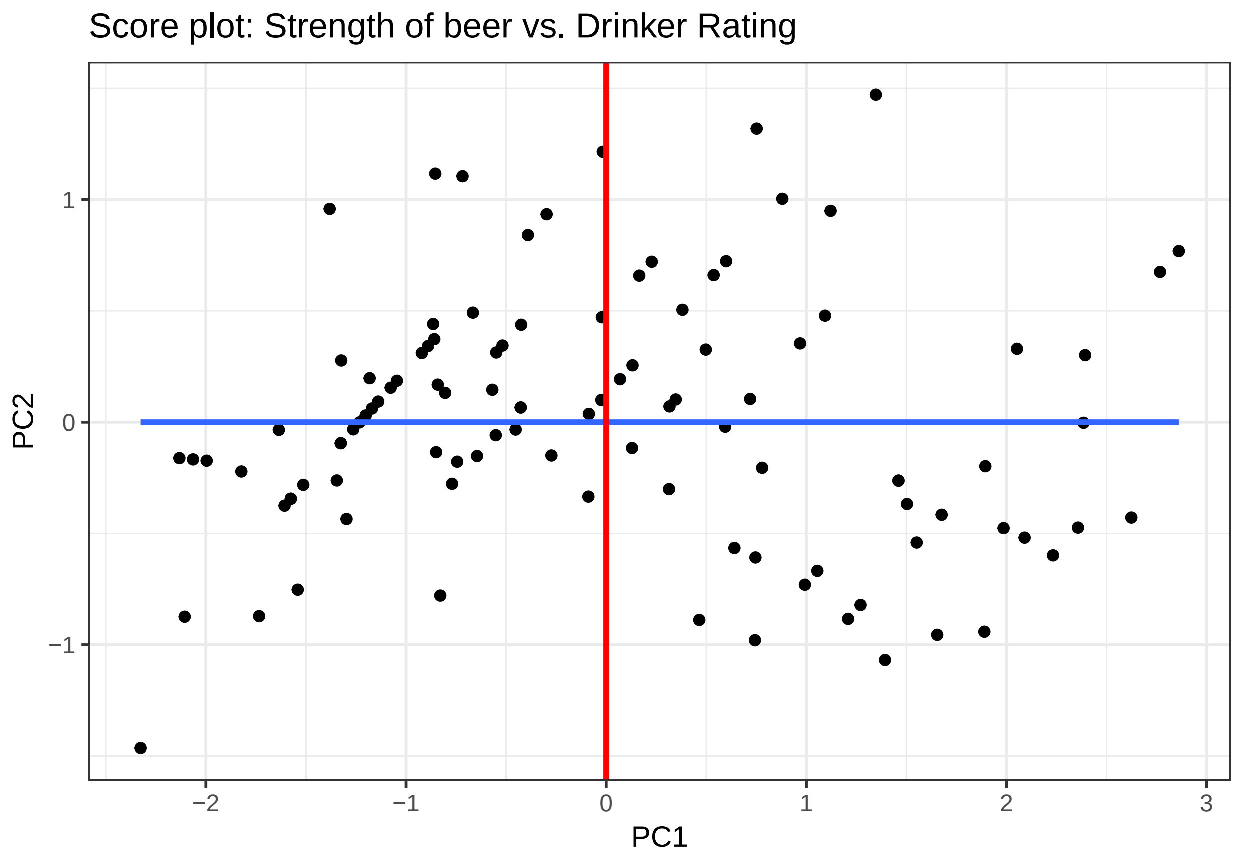
Recall that the eigenvectors determine by how much we rotate the original data to change the axes to new perpendicular axes that minimise error and maximise variance, i.e. by how much do we rotate the data to make the new axes the princpal components.

(If you want to know how eigenvectors are calculated, click for a useful video from Khan Academy)

```
eigens <- eigen(vcov)
# this finds the eigenvalues and eigenvectors and stores them in objects
# called 'values' and 'vectors' respectively we can access these objects
# like we access columns, using a dollar sign $

eigenvectors <- eigens$vectors

data_values <- as.matrix(dplyr::select(beers, c(ABV.standard, Rating.standard)))
# we select (invoking the dplyr package we loaded) from 'beers' only the
# columns of interest and convert to a matrix

rotated <- as.data.frame(data_values %*% eigenvectors)
colnames(rotated) <- c("PC1", "PC2")
# we multiply the data by the eigenvectors (and rename)
```

The resulting rotated data are called scores and the scores are displayed on the plot.

```
ggplot(rotated, aes(PC1, PC2)) +
  geom_point() + #
  labs(title = "Score plot: Strength of beer vs. Drinker Rating") +
  geom_smooth(method = lm, se = F) +
  geom_vline(xintercept = 0, colour = "red", size = 1) +
  # adds a vertical line through x = 0 in red +
  theme_bw()
```



Step 4: Find the eigenvectors

We can extract the eigenvalues from the variance covariance matrix. PC1 explains 1.67/2.00 % of the total variance in the dataset. The negative sign has no bearing on the interpretation (indeed - there is no such thing as negative variance).

```
eigenvalues <- eigens$values
round(eigenvalues, 2)
```

```
## [1]  1.67 -0.33
```

PCA Computation & Exploration

Now, all that is very complicated. Luckily, there is a built-in function in R to do all this in the background, for as many dimensions of data as we might like!

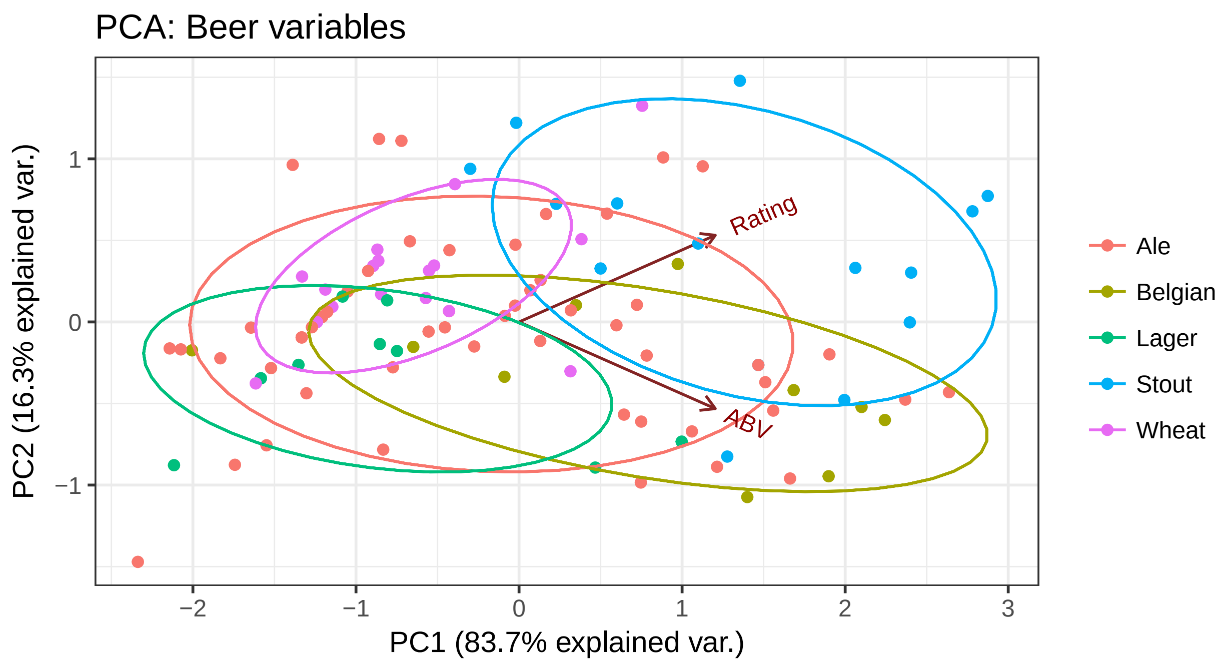
The princomp() command conducts a PCA. Note that cor = T(rue) indicates that the variables must be centered (on 0) and scaled (to have variance 1) to be interpretable. The summary gives us the proportion of variance (the eigenvalues) explained by each component.

```
beercols <- dplyr::select(beers, c(ABV, Rating))  ## select the revelant columns

beers_pca <- princomp(beercols, cor = T)
summary(beers_pca)
```

```
## Importance of components:
##                           Comp.1    Comp.2
## Standard deviation     1.2936662 0.5713386
## Proportion of Variance 0.8367861 0.1632139
## Cumulative Proportion  0.8367861 1.0000000
```

Arrows for each variable point in the direction of increasing values of that variable. Thus, higher ratings are associated with Stout (and not Lager) and higher alcohol is associated with Belgian beers (and not Wheat beers). PC1 is associated with higher ratings and higher alcohol content (both arrows point in the direction of increasing PC1 values) and PC2 is associated with higher rating and lower alcohol content. This could also have been determined from the loadings.

```
g <- ggbiplot(beers_pca, obs.scale = 1, group = beers$Style, ellipse = T) +
    scale_color_discrete(name = "") + labs(title = "PCA: Beer variables") +
    theme_bw()

print(g)
```



```
beers_pca$loadings
```

```
##
```

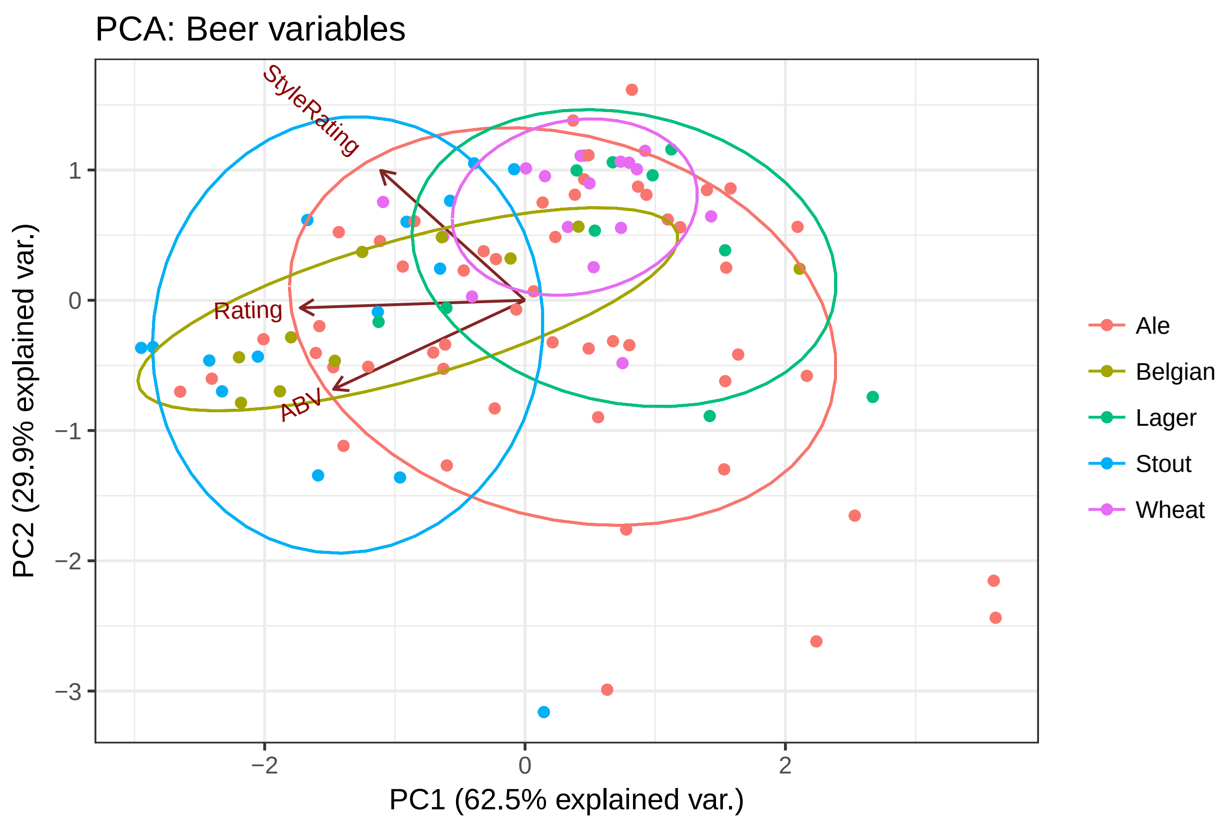
```
## Loadings:
##        Comp.1 Comp.2
## ABV     0.707 -0.707
## Rating  0.707  0.707
##
##                Comp.1 Comp.2
## SS loadings       1.0    1.0
## Proportion Var    0.5    0.5
## Cumulative Var    0.5    1.0
```

```
# obs.scale expands the graph group says which variable to overlay on the
# graph and ellipse adds a rough cluster region scale_color_discrete adds
# color to match the group variable
```

Technical note: there is also the prcomp() command which uses the mathematical algorithm Singular Value Decomposition to conduct the PCA - this is technically more correct than the eigenvalue method but in practice the two often result in very similar results.

What happens when we also want to see how people rated the beer against others of its style?

```
beercols2 <- dplyr::select(beers, c(ABV, Rating, StyleRating))
## select the revelant columns

beers_pca2 <- princomp(beercols2, cor = T)


g2 <- ggbiplot(beers_pca2, obs.scale = 1, var.scale = 2, group = beers$Style, ellipse = T) +
  scale_color_discrete(name = '') + #
  labs(title = "PCA: Beer variables") +
  theme_bw()

print(g2)
```



```
beers_pca2$loadings
```

```
##
## Loadings:
##             Comp.1 Comp.2 Comp.3
## ABV         -0.582 -0.565  0.585
## Rating      -0.684        -0.728
## StyleRating -0.439  0.824  0.358
##
##                Comp.1 Comp.2 Comp.3
## SS loadings     1.000  1.000  1.000
## Proportion Var  0.333  0.333  0.333
## Cumulative Var  0.333  0.667  1.000
```

```
summary(beers_pca2)
```

```
## Importance of components:
##                           Comp.1    Comp.2     Comp.3
## Standard deviation     1.3688184 0.9473749 0.47834821
## Proportion of Variance 0.6245546 0.2991731 0.07627234
## Cumulative Proportion  0.6245546 0.9237277 1.00000000
```

In this case, we see that overall Rating and ABV are actually more closely correlated than overall Rating and Style Rating, and that a higher rating and higher ABV beer is probably suggestive of a Belgian beer (and maybe a Stout, but definitely not a Lager!) PC1 is associated with higher ratings overall and higher alcohol content.

If we examine the loadings, we see that Rating does not have any bearing on PC2 (this can be see in the biplot as well - increasing values of Rating are associated with a PC2 value of 0). PC2 is associated with lower alcohol content and good style ratings. We might conclude PC1 is measuring a construct of ‘is the beer good overall’ and PC2 is measuring ‘is the beer consistent with its style’ and proceed to further analyses accordingly.

Note also that the relationships between variables and data has changed quite dramatically - it is generally useful to think a priori which variables you actually want to explore. This has to do with what you think the underlying construct might be.

Social Epidemiology

So what does this all have to do with social epidemiology?

PCA has been used in social epi in a number of different ways and it gets much more useful when there are multiple dimensions of data. PCA is used to explore relationships between variables and relationships of variables to particular clusters in the dataset (like WHO regions). It is good for elucidating what particular **constructs** might be underlying the data.

Sometimes, we can use the first principal component (or first and second principal components - they are uncorrelated) as a proxy measure of some underlying construct for the variables. If you had data on several measures of socioeconomic position, for example, it would be possible to use PC1 as a proxy variable for the underlying construct of SEP and to thus adjust for it in an analysis.