COLUMBIA UNIVERSITY
School of Professional Studies

Applied Analytics: Frameworks and Methods 2

Dimension Reduction

Outline

- Motivation for Dimension Reduction
- Types and Relationship to other Techniques
- Exploratory Factor Analysis
- Principal Components Analysis

Why Reduce Dimensions?

- When faced with a large set of correlated variables, dimension reduction allows us to summarize this set with a smaller number of representative variables that collectively explain most of the variability in the original set.

- Parsimony is a desirable property for models
  - Predictions from simple (versus complex models) are more stable across samples
  - Simple models are easier to interpret and communicate to stakeholders.

- As number of predictors increases, the chance of finding correlations among a predictor or a set of predictors increases. Such correlations among predictors called *multicollinearity* inflates standard errors of coefficients, potentially leading to erroneous conclusions about the relevance of a predictor.

- In cases where $p > n$, traditional estimation techniques will not work.

Dimension Reduction vs. Clustering

- Like clustering, dimension reduction techniques are unsupervised learning approaches
- In contrast to clustering, dimension reduction seeks to group variables rather than observations.

Process

- Dimension reduction works by grouping variables together.
  - To better understand the process of dimension reduction, think of spinning a test tube containing muddy water in a centrifuge. Similar particles are grouped together in layers, each layer represents a factor or component of the original data.

- Dimension reduction involves loss of information
  - Representing a large number of variables using a small number of factors or components often results in loss of information. However, this is okay so long as the information loss is not substantial. To better understand this idea, think of .jpg or .gif image files used on websites. These are compact image files that look just like the original .bmp or .tiff image files they are derived from. It is only at high resolution that the pixilation of a .gif or .jpg is apparent.

Approaches based on Purpose

- Determine underlying themes
  - Identify underlying dimensions or factors that explain the correlations among a set of variables. This is frequently used in the social sciences to identify latent variables or constructs underlying measured variables
  - Technique: Factor Analysis
- Represent data with fewer components
  - To identify a set of uncorrelated variables to replace the original set of correlated variables. This may be done to use the variables in a supervised learning technique that is sensitive to multicollinearity
  - Technique: Principal Components Analysis
- Note: The distinction is not as cut and dry as indicated by the above classification. Factor analysis may also be used to reduce data while Principal Components Analysis may shed light on underlying meaning

Exploratory Factor Analysis

Purpose of Factor Analysis

- A fundamental assumption of factor analysis is that measured data represents deeper latent factors.
- It aims to find deeper latent factors underlying observed data
- Or, to confirm relationships between observed data and latent factors.
    - For e.g., if the questions included in the survey represent the constructs they were designed to measure. We will now pursue this idea using some survey data.

Expected Structure

- Unlike many supervised learning techniques which generate a unique solution, factor analysis generates many good solutions.
- Factor analysis assumes the existence of a theoretical mapping between the factor and variables.
- Thus, before conducting a factor analysis, it is important to formulate an expectation of the latent factors underlying the data and their relationship with the observed variables.

Toothpaste Survey

Consider a survey on the benefits people seek from toothpaste

Please indicate your level of agreement with the following statements.

	Strongly disagree	Disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Agree	Strongly agree
It is important to buy a toothpaste that prevents cavities	○	○	○	○	○	○	○
I like a toothpaste that gives shiny teeth	○	○	○	○	○	○	○
A toothpaste should strengthen your gums	○	○	○	○	○	○	○
I prefer a toothpaste that freshens breath	○	○	○	○	○	○	○
Prevention of tooth decay is not an important benefit offered by a toothpaste	○	○	○	○	○	○	○
The most important consideration in buying a toothpaste is attractive teeth	○	○	○	○	○	○	○

Toothpaste Survey

Can you identify any themes (latent factors) underlying these survey questions?

It is important to buy a toothpaste that prevents cavities

I like a toothpaste that gives shiny teeth

A toothpaste should strengthen your gums

I prefer a toothpaste that freshens breath

Prevention of tooth decay is not an important benefit offered by a toothpaste

The most important consideration in buying a toothpaste is attractive teeth

Toothpaste Survey

Can you identify any themes (latent factors) underlying these survey questions?

It is important to buy a toothpaste that prevents cavities

I like a toothpaste that gives shiny teeth

A toothpaste should strengthen your gums

I prefer a toothpaste that freshens breath

Prevention of tooth decay is not an important benefit offered by a toothpaste

The most important consideration in buying a toothpaste is attractive teeth

Toothpaste Survey

- The toothpaste survey was conducted on a small sample (n=30)
- Responses gathered are saved in a text file for use in factor analysis.

Steps in Factor Analysis

1. Suitability for Factor Analysis
   - Correlations
   - Bartlett's Test of Sphericity
   - KMO MSA
2. Determine Number of Factors
   - Scree Plot
   - Eigen-value
   - Parallel Analysis
   - Total variance explained
   - Extracted Communalities
3. Mapping Variables to Factors
4. Interpretation
5. Representing the Factor

Suitability for Factor Analysis

- Since factor analysis attempts to group similar variables, a basic requirement is that at least a few variables be related to each other.
- Correlation Matrix
  - A correlation matrix with some large and some small correlations is ideal for factor analysis.
  - If all correlations are small, there is no way to group variables. On the other hand, if all correlations are large, then all the variables will load onto the same factor.
- Bartlett's Test of Sphericity
  - Looks to see if there are at least some non-zero correlations by comparing correlation matrix to an identity matrix. A significant test indicates suitability for factor analysis.
- Kaiser-Meyer-Olkin's (KMO's) Measure of Sampling Adequacy (MSA)
  - Compares partial correlation matrix to pairwise correlation matrix. A partial correlation is a correlation after partialling out all other correlations. If the variables are strongly related, partial correlations should be small and MSA close to 1. If MSA > 0.5, data is suitable for factor analysis.

Determine Number of Factors

- It is critical to have an a priori expectation of the number of factors.
- Following data-driven methods must be used to corroborate an a priori solution or select a set of priori solutions
  - Scree Plot
    - Line graph of eigen values for each factor. Ideal number of factors is indicated by a sudden change in the line graph or what is known as the *elbow*.
  - Eigen-value
    - Select all factors with eigen value greater than 1
  - Parallel Analysis
    - Simulate a dataset with same variables and observations as original dataset. Compute correlation matrix and eigen values. Now, compare eigen values from simulated data to original data. Select factors with eigen values in the original data greater than eigen values in the simulated data.

Determine Number of Factors

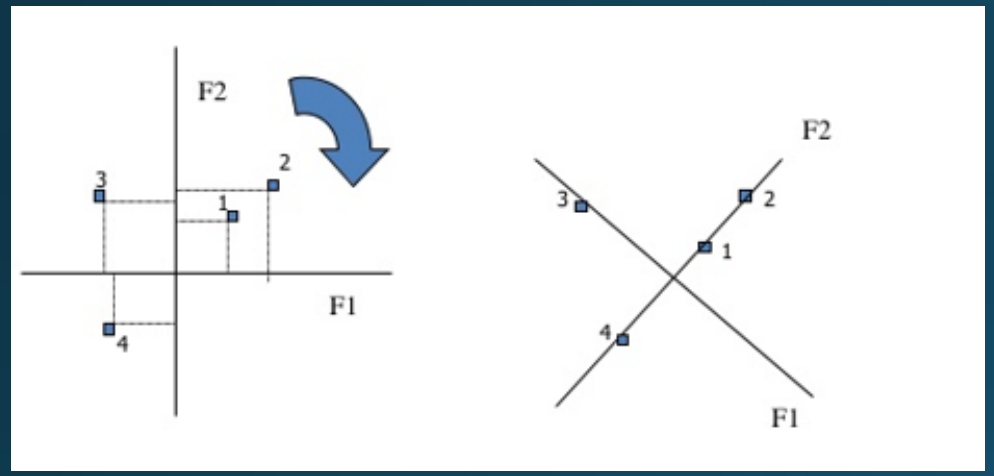
- Total Variance Explained
  - To ensure that the factors represents the original variables sufficiently well, the total variance explained by factors should be greater than 70%.

- Extracted Communalities
  - Communality reflects the amount of variance in a variable that can be explained by the factors.
  - Larger the communality, the more of the variable is captured by the factor solution.
  - On the other hand, a small communality implies most of the variance in the variable was not captured. Ideally, communality of each variable must be greater than 0.7, but a communality greater than 0.5 may be seen as acceptable.

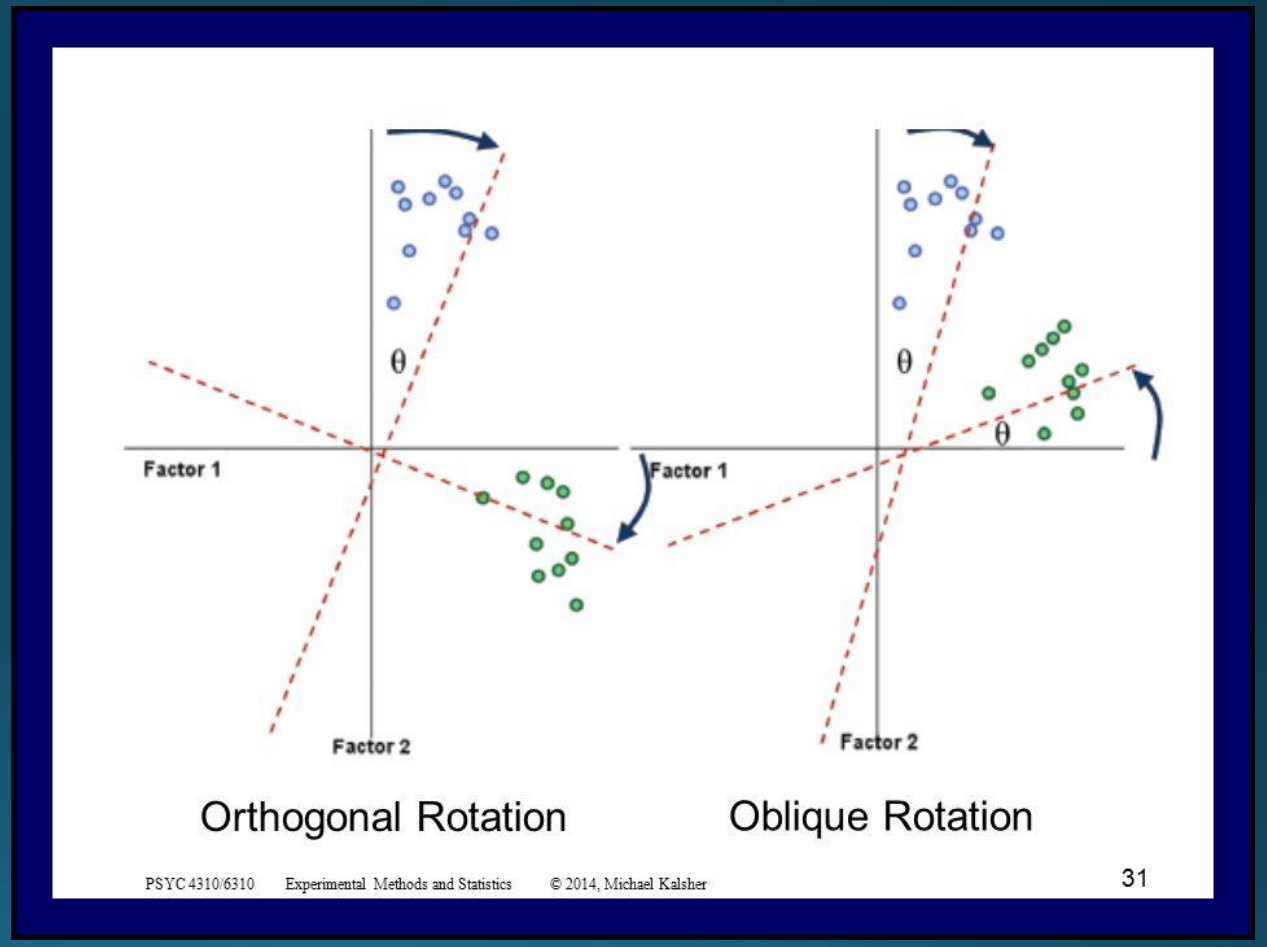
Mapping Variables to Factors

- Each variable is represented as a linear combination of factors
- An ideal factor solution is where each variable is expected to load on (i.e., related to) only one factor. Such a result is easy to interpret.
- In practice, each variable may load on many factors. This may still be acceptable so long as the loading on one factor is large and on all other factors is small.
- When the pattern of loadings does not show a clear preference of a variable for a factor, rotating the axes may help generate a clear mapping. There are two broad types of axes rotation
  - Orthogonal: Axes are rotated while constraining them to be at right angles. E.g., varimax, quartimax, equimax,
  - Oblique: Axes are allowed to have any angle between them. E.g., oblimin, promax

Axes Rotation



Source: Indian School of Mines



Interpretation

- Review pattern of factor loadings from the rotated matrix to interpret the factor analysis solution
- The meaning of each factor is derived from the variables loading on it.
- Review the variables to describe each factor.

Representing the Factor

- If the goal is to use the factors in further analysis, then they may be represented in one of three ways
  - Average scores of variables reflecting the factor
  - Weighted average of variables reflecting the factor, where weights are the factor loadings
  - Pick a variable as a representative of the factor

R Illustration

Data: Toothpaste

You must read the data before trying to run code on your own machine. To read data use the following code after setting your working directory. To set your working directory, modify the following to set the file path for the folder where the data file resides. `setwd('c:/thatawesomeclass/)`

```
data = read.csv('toothpaste.csv')
```

Explore

The survey consists of responses to six seven-point likert-scale items with anchors, strongly disagree and strongly agree.

- prevents_cavities: It is important to buy a toothpaste that prevents cavities
- shiny_teeth: I like a toothpaste that gives shiny teeth
- strengthens_gums: A toothpaste should strengthen your gums
- freshens_breath: I prefer a toothpaste that freshens breath
- prevent_decay_not_imp: Prevention of tooth decay is not an important benefit offered by a toothpaste
- attractive_teeth: The most important consideration in buying a toothpaste is attractive teeth

```
survey = data[,2:7]
head(survey)
```

	prevents_cavities <int>	shiny_teeth <int>	strengthens_gums <int>	freshens_breath <int>
1	7	3	6	4
2	1	3	2	4
3	6	2	7	4
4	4	5	4	6
5	1	2	2	3
6	6	3	6	4

6 rows | 1-5 of 7 columns

```
summary(survey)
```

```
##  prevents_cavities  shiny_teeth  strengthens_gums freshens_breath
##  Min.   :1.000     Min.   :2.0   Min.   :1.0      Min.   :2.0
##  1st Qu.:2.000     1st Qu.:3.0   1st Qu.:2.0      1st Qu.:3.0
##  Median :4.000     Median :4.0   Median :4.0      Median :4.0
##  Mean   :3.933     Mean   :3.9   Mean   :4.1      Mean   :4.1
##  3rd Qu.:6.000     3rd Qu.:5.0   3rd Qu.:6.0      3rd Qu.:5.0
##  Max.   :7.000     Max.   :7.0   Max.   :7.0      Max.   :7.0
##  prevent_decay_not_imp attractive_teeth
##  Min.   :1.0           Min.   :2.000
##  1st Qu.:2.0           1st Qu.:3.000
##  Median :3.5           Median :4.000
##  Mean   :3.5           Mean   :4.167
##  3rd Qu.:5.0           3rd Qu.:4.750
##  Max.   :7.0           Max.   :7.000
```

Suitability for Factor Analysis

Correlations

We examine bivariate correlations among responses to the six survey questions.

- A correlation matrix with some large and some small correlations is ideal for factor analysis.
- If all correlations are small, there is no way to group variables. On the other hand, if all correlations are large, then all the variables will load onto the same factor.

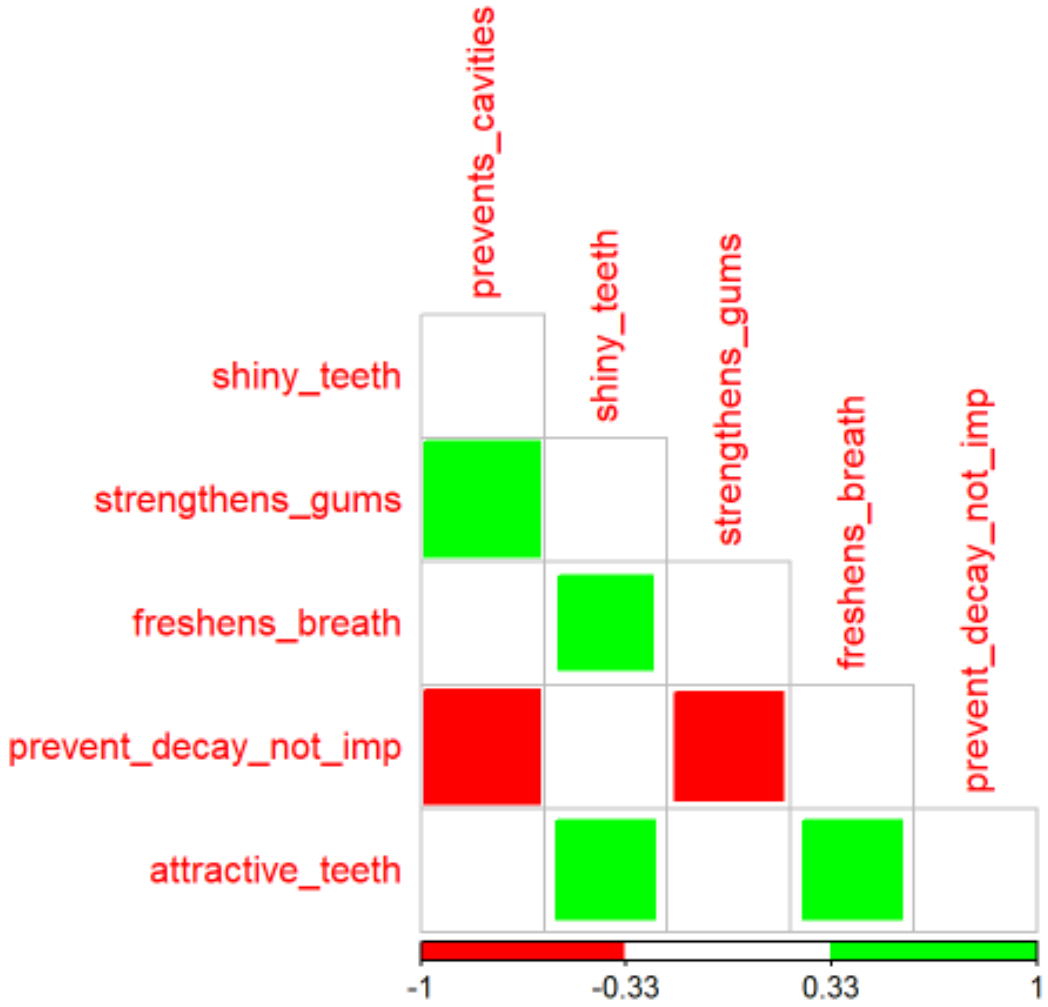
It is common to assume responses to itemized rating scales, the kind used for this survey, are on an interval scale. Therefore, we will compute pearson's product moment correlation. Note, if we treat the data as being ordinal (as some researchers would argue), we would examine polychoric correlations.

```
round(cor(survey), 3)
```

```
##                     prevents_cavities shiny_teeth strengthens_gums
## prevents_cavities                1.000      -0.053            0.873
## shiny_teeth                     -0.053       1.000           -0.155
## strengthens_gums                 0.873      -0.155            1.000
## freshens_breath                 -0.086       0.572           -0.248
## prevent_decay_not_imp           -0.858       0.020           -0.778
## attractive_teeth                 0.004       0.640           -0.018
##                     freshens_breath prevent_decay_not_imp
## prevents_cavities            -0.086                -0.858
## shiny_teeth                   0.572                 0.020
## strengthens_gums             -0.248                -0.778
## freshens_breath               1.000                -0.007
## prevent_decay_not_imp        -0.007                 1.000
## attractive_teeth              0.640                -0.136
##                     attractive_teeth
## prevents_cavities              0.004
## shiny_teeth                    0.640
## strengthens_gums              -0.018
## freshens_breath                0.640
## prevent_decay_not_imp         -0.136
## attractive_teeth               1.000
```

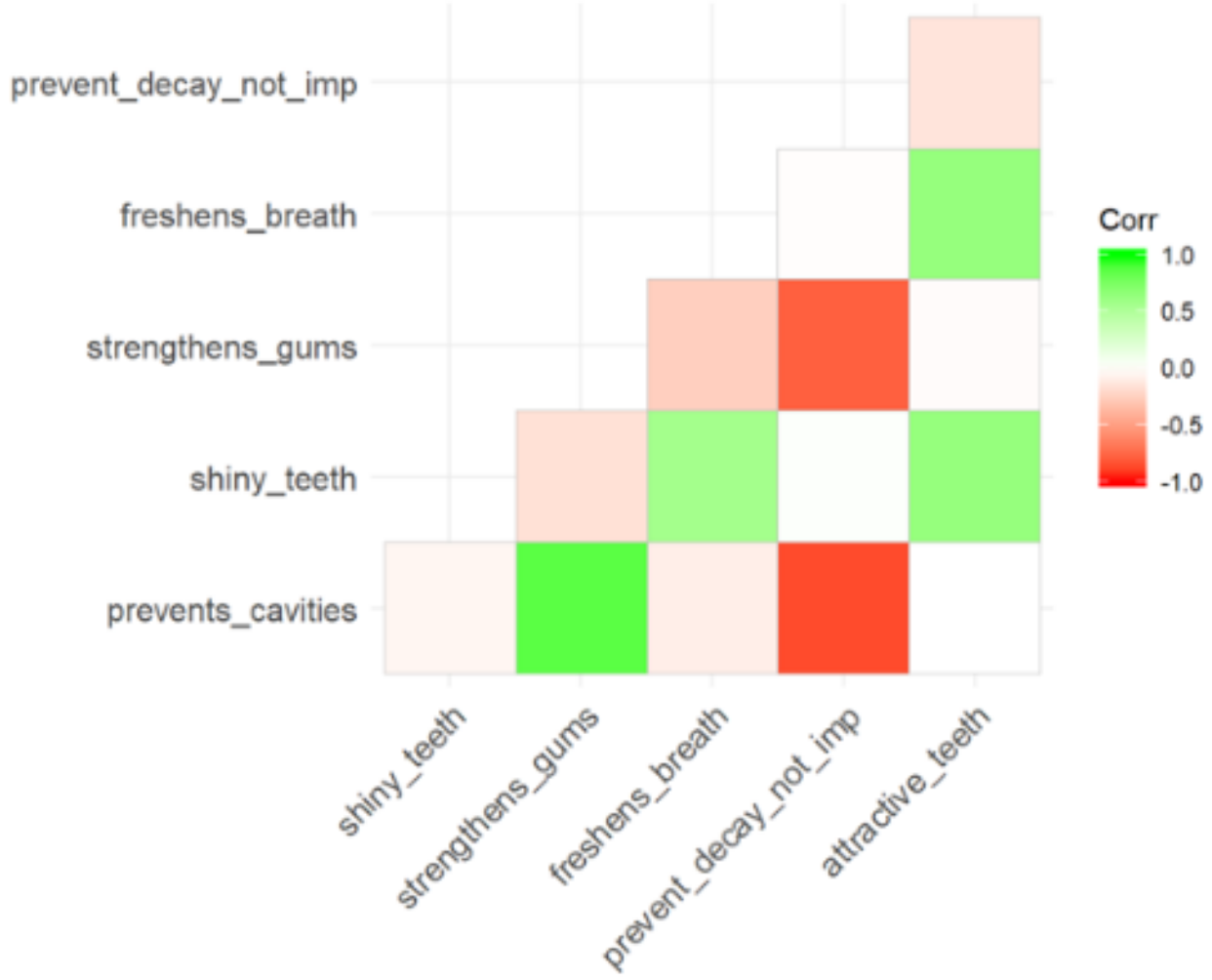
Since we are only interested in the pattern of correlations, we can look at a heatmap of correlations.

```
library(corrplot)
corrplot(cor(data[,2:7]),type = 'lower',col = c('red','white','green'),method = 'square',diag = F)
```



Here is a plot using ggcorrplot

```
library(ggcorrplot)
ggcorrplot(cor(data[,2:7]),colors = c('red','white','green'),type = 'lower')
```



Bartlett's Test of Sphericity

Looks to see if there are at least some non-zero correlations by comparing correlation matrix to an identity matrix. A significant test indicates suitability for factor analysis.

```
library(psych)
cortest.bartlett(cor(survey),n = 30)
```

```
## $chisq
## [1] 111.3138
##
## $p.value
## [1] 9.017094e-17
##
## $df
## [1] 15
```

KMO Measure of Sampling Adequacy (MSA)

Compares partial correlation matrix to pairwise correlation matrix. A partial correlation is a correlation after partialing out all other correlations. If the variables are strongly related, partial correlations should be small and MSA close to 1. If MSA > 0.5, data is suitable for factor analysis.

```
KMO(r = cor(survey))
```

```
## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = cor(survey))
## Overall MSA =  0.66
## MSA for each item =
##     prevents_cavities           shiny_teeth      strengthens_gums
##                  0.62                  0.70                  0.68
##       freshens_breath prevent_decay_not_imp      attractive_teeth
##                  0.64                  0.77                  0.56
```

Determine Number of Factors

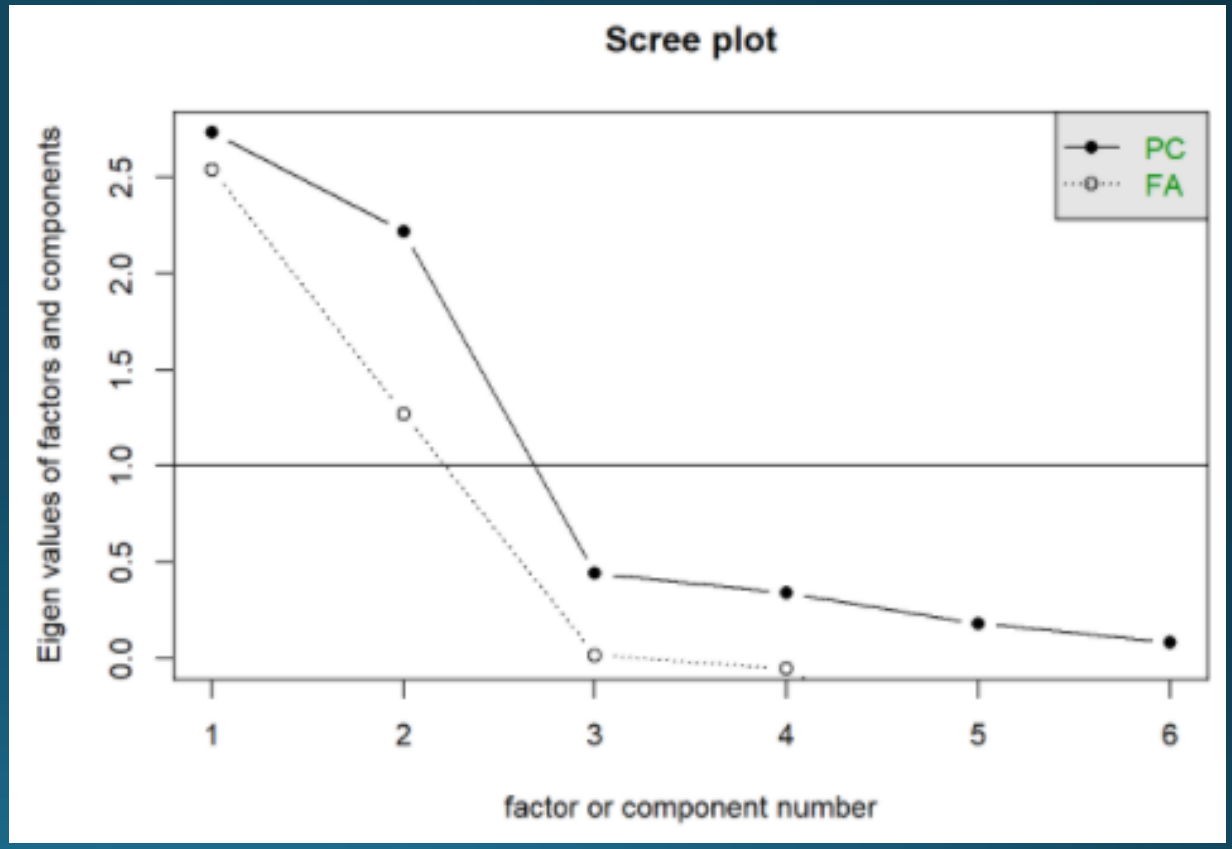
It is critical to have an a priori expectation of the number of factors.

Now, let us examine some data-driven methods used to corroborate an a priori solution or select from a set of a priori solutions.

Scree Plot

Line graph of eigen values for each factor. Ideal number of factors is indicated by a sudden change in the line graph or what is known as the elbow.

```
scree(cor(survey),factors = T, pc=T)
```



Eigen Value

According to the eigen-value criterion, all factors with eigen value greater than 1 are selected.

```
data.frame(factor = 1:ncol(survey), eigen = eigen(cor(survey))$values)
```

factor <int>	eigen <dbl>
1	2.73118833
2	2.21811927
3	0.44159791
4	0.34125765
5	0.18262823
6	0.08520861

6 rows

```
Console  Terminal  Render  Jobs
R 4.1.2 · ~/
> result = fa(r = survey,nfactors = 2,fm = 'pa',rotate = 'none')
>
> result$e.values[1:6]
[1] 2.73118833 2.21811927 0.44159791 0.34125765 0.18262823 0.08520861
>
> 100*result$e.values[1:6]/length(result$e.values)
[1] 45.519806 36.968654  7.359965  5.687627  3.043804  1.420144
>
```

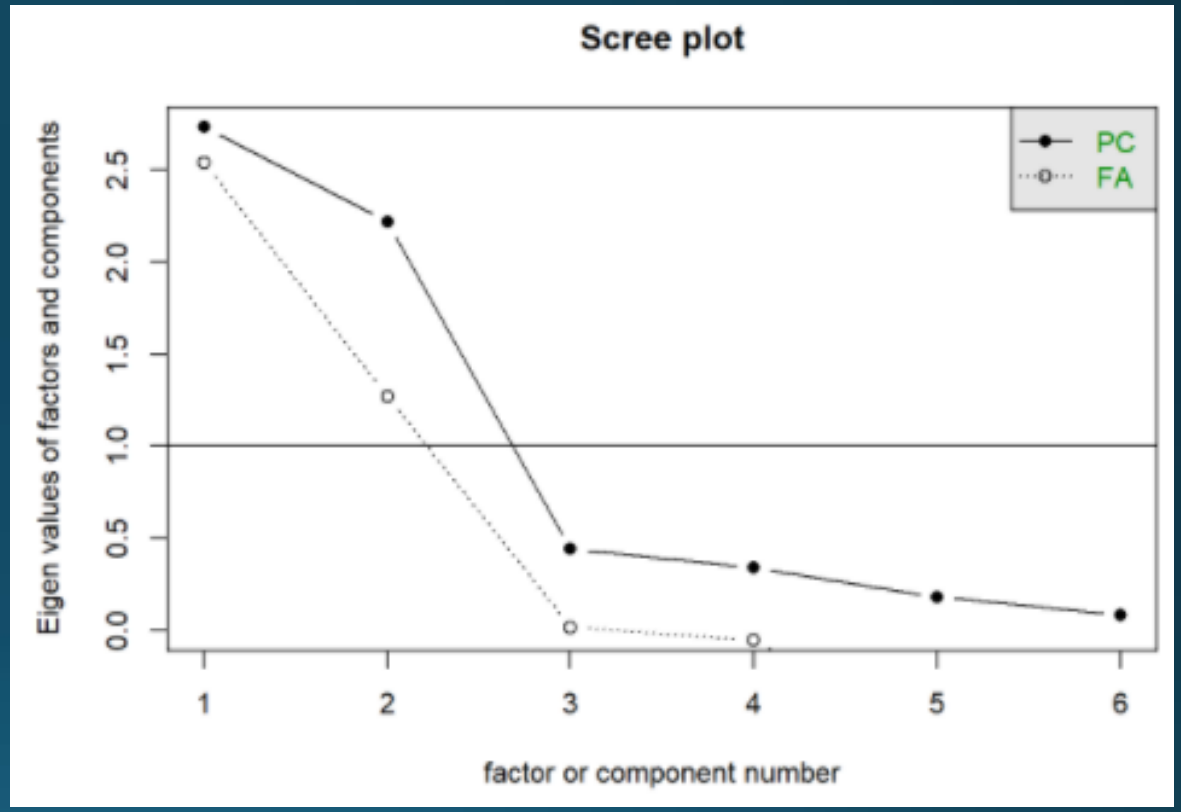
Eigen Value

According to the eigen-value criterion, all factors with eigen value greater than 1 are selected.

```
data.frame(factor = 1:ncol(survey), eigen = eigen(cor(survey))$values)
```

factor <int>	eigen <dbl>
1	2.73118833
2	2.21811927
3	0.44159791
4	0.34125765
5	0.18262823
6	0.08520861

6 rows

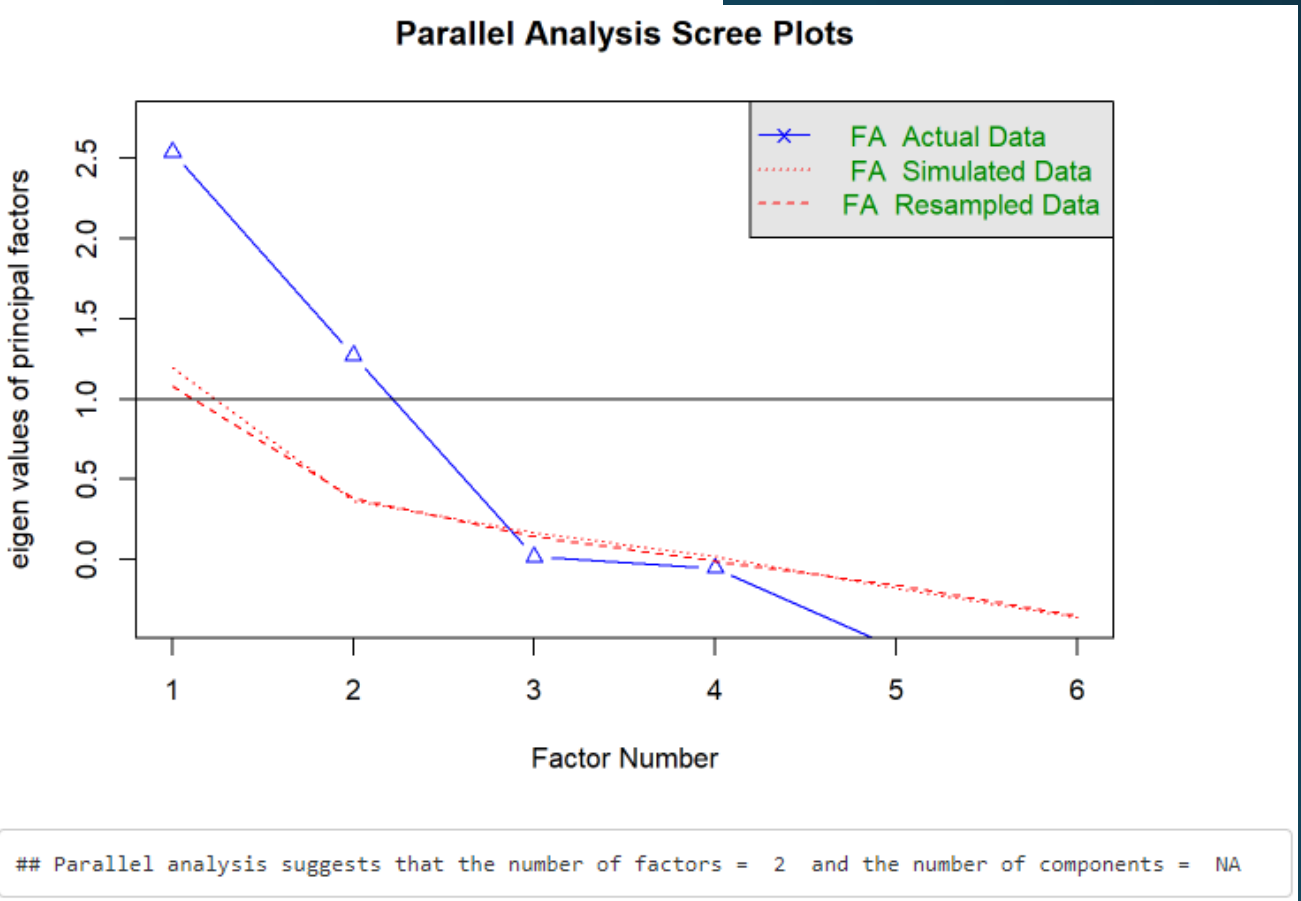



```
Console  Terminal  Jobs
R 4.1.2 · ~/
> result$values[1:6]
[1]  2.57007551  1.86673881  0.08860081  0.03904031 -0.04561196
[6] -0.08288154
> 100*result$values[1:6]/length(result$values)
[1] 42.8345918 31.1123134  1.4766801  0.6506719 -0.7601993 -1.3813589
```

Parallel Analysis

Simulate a dataset with same variables and observations as original dataset. Compute correlation matrix and eigen values. Now, compare eigen values from simulated data to original data. Select factors with eigen values in the original data greater than eigen values in the simulated data.

```
fa.parallel(survey,fa='fa',fm = 'pa')
```




```
## Parallel analysis suggests that the number of factors =  2  and the number of components =  NA
```

```
> ?fa.parallel
>
```

Files | Plots | Packages | Help | Viewer

R: Scree plots of data or correlation matrix compared to random... ▾ Find in Topic

fm	What factor method to use. (minres, ml, uls, wls, gls, pa) See fa for details.
fa	show the eigen values for a principal components (fa="pc") or a principal axis factor analysis (fa="fa") or both principal components and principal factors (fa="both")

Total Variance Explained

To ensure that the factors represents the original variables sufficiently well, the total variance explained by factors should be greater than 70%.

Since the three above tests corroborated the a priori two-factor solution, we will now run an exploratory factor analysis using principal axis factoring with two factors. Next, we examine the Cumulative Variance explained by the two factors.

```
result = fa(r = survey,nfactors = 2,fm = 'pa',rotate = 'none')
result$Vaccounted
```

```
##                            PA1       PA2
## SS loadings           2.5700755 1.8667388
## Proportion Var        0.4283459 0.3111231
## Cumulative Var        0.4283459 0.7394691
## Proportion Explained  0.5792615 0.4207385
## Cumulative Proportion 0.5792615 1.0000000
```

Extracted Communalities

- Communality reflects the amount of variance in a variable that can be explained by the factors.
- Larger the communality, the more of the variable is captured by the factor solution.
- On the other hand, a small communality implies most of the variance in the variable was not captured. Ideally, communality of each variable must be greater than 0.7, but a communality greater than 0.5 may be seen as acceptable.

```
data.frame(communality = result$communality)
```

	communality <dbl>
prevents_cavities	0.9266887
shiny_teeth	0.5625047
strengthens_gums	0.8370333
freshens_breath	0.6016147
prevent_decay_not_imp	0.7898264
attractive_teeth	0.7191465

6 rows

Mapping Variables to Factors

Each variable is represented as a linear combination of factors. An ideal factor solution is where each variable is expected to load on (i.e., related to) only one factor. Such a result is easy to interpret. In practice, each variable may load on many factors. This may still be acceptable so long as the loading on one factor is large and on all other factors is small.

When the pattern of loadings does not show a clear preference of a variable for a factor, rotating the axes may help generate a clear mapping. There are two broad types of axes rotation

- Orthogonal: Axes are rotated while constraining them to be at right angles. E.g., varimax, quartimax, equimax
- Oblique: Axes are allowed to have any angle between them. E.g., oblimin, promax

Here is the pattern of loadings for the unrotated solution (i.e., rotation='none)

```
print(result$loadings, cut=0)
```

```
##
## Loadings:
##                       PA1    PA2
## prevents_cavities      0.948  0.168
## shiny_teeth           -0.207  0.721
## strengthens_gums       0.914  0.039
## freshens_breath       -0.247  0.735
## prevent_decay_not_imp -0.850 -0.260
## attractive_teeth      -0.101  0.842
##
##                  PA1   PA2
## SS loadings    2.570 1.867
## Proportion Var 0.428 0.311
## Cumulative Var 0.428 0.739
```

To make the matrix of loadings easier to interpret, let us exclude small loadings, say below 0.15. Note, four of the six variables load on both factors.

```
print(result$loadings, cut=0.15)
```

```
##
## Loadings:
##                       PA1    PA2
## prevents_cavities       0.948  0.168
## shiny_teeth            -0.207  0.721
## strengthens_gums        0.914
## freshens_breath        -0.247  0.735
## prevent_decay_not_imp -0.850 -0.260
## attractive_teeth                0.842
##
##                  PA1   PA2
## SS loadings    2.570 1.867
## Proportion Var 0.428 0.311
## Cumulative Var 0.428 0.739
```

```
> print(result$loadings, cut=0.25)

Loadings:
                       PA1    PA2
prevents_cavities       0.948
shiny_teeth                    0.721
strengthens_gums        0.914
freshens_breath                0.735
prevent_decay_not_imp -0.850 -0.260
attractive_teeth               0.842

                 PA1   PA2
SS loadings    2.570 1.867
Proportion Var 0.428 0.311
Cumulative Var 0.428 0.739
```

Now, let's examine the matrix after an orthogonal rotation using varimax. Is the mapping of variables to factors more clear?

```
fa_varimax = fa(r = survey,nfactors = 2,fm = 'pa',rotate = 'varimax')
print(fa_varimax$loadings,cut=0.15)
```

```
##
## Loadings:
##                       PA1    PA2
## prevents_cavities      0.962
## shiny_teeth                  0.748
## strengthens_gums      0.902 -0.155
## freshens_breath              0.771
## prevent_decay_not_imp -0.886
## attractive_teeth             0.844
##
##                  PA1   PA2
## SS loadings    2.539 1.898
## Proportion Var 0.423 0.316
## Cumulative Var 0.423 0.739
```

Now, let's try an oblique rotation using oblimin. What do you think of the mapping of variables to factors?

```
fa_oblimin = fa(r = survey,nfactors = 2,fm = 'pa',rotate = 'oblimin')
print(fa_oblimin$loadings,cut=0.15)
```

```
##
## Loadings:
##                       PA1    PA2
## prevents_cavities      0.963
## shiny_teeth                  0.747
## strengthens_gums      0.901
## freshens_breath              0.767
## prevent_decay_not_imp -0.887
## attractive_teeth             0.848
##
##                  PA1   PA2
## SS loadings    2.543 1.892
## Proportion Var 0.424 0.315
## Cumulative Var 0.424 0.739
```

Interpretation

Review pattern of factor loadings from the rotated matrix to interpret the factor analysis solution. The meaning of each factor is derived from the variables loading on it. So, let's review the variables to describe each factor.

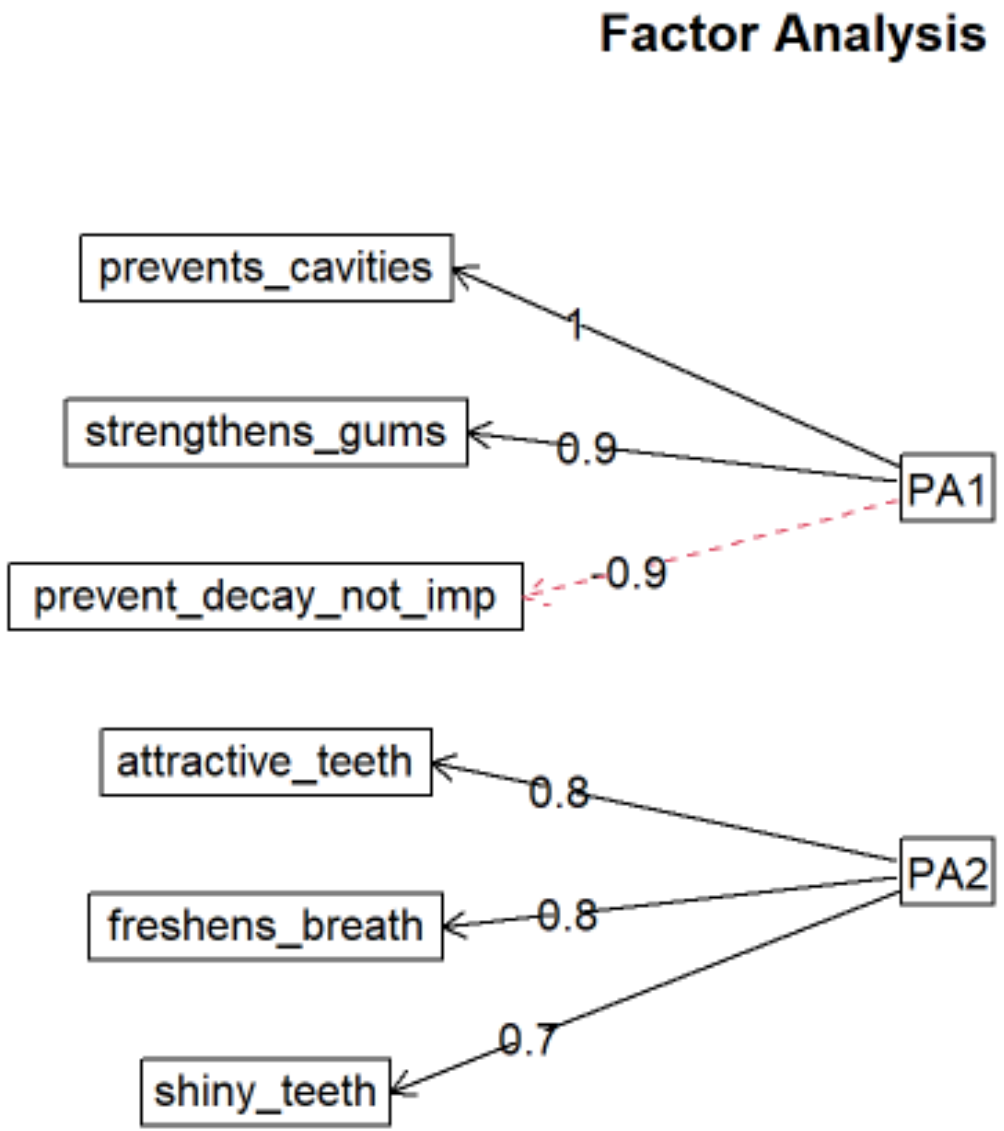
Since oblimin offered the clearest mapping of variables to factors, that matrix is used. To make interpretation easier, matrix is sorted.

What need is reflected in the first factor, PA1? What need is reflected by the second factor, PA2? Are these consistent with our a priori expectation?

```
print(fa_oblimin$loadings,cut=0.15, sort=T)
```

```
##
## Loadings:
##                       PA1    PA2
## prevents_cavities      0.963
## strengthens_gums       0.901
## prevent_decay_not_imp -0.887
## shiny_teeth                   0.747
## freshens_breath               0.767
## attractive_teeth              0.848
##
##                  PA1   PA2
## SS loadings    2.543 1.892
## Proportion Var 0.424 0.315
## Cumulative Var 0.424 0.739
```

```
fa.diagram(fa_oblimin,sort = T)
```



Representing the Factor

If the goal is to use the factors in further analysis, then they may be represented in one of three ways

Average scores of variables reflecting the factor

```
factor1_avg = rowMeans(survey[,c('prevents_cavities','strengthens_gums','prevent_decay_not_imp')])
factor2_avg = rowMeans(survey[,c('attractive_teeth','freshens_breath','shiny_teeth')])
cbind(factor1_avg, factor2_avg)
```

```
##       factor1_avg factor2_avg
##  [1,]    5.000000    3.666667
##  [2,]    2.666667    3.666667
##  [3,]    4.666667    3.000000
##  [4,]    3.333333    5.333333
##  [5,]    3.000000    2.333333
##  [6,]    4.666667    3.666667
##  [7,]    5.000000    3.000000
##  [8,]    4.666667    4.000000
##  [9,]    3.666667    3.333333
## [10,]    3.666667    6.000000
## [11,]    5.000000    3.333333
## [12,]    2.666667    3.666667
## [13,]    4.666667    3.000000
## [14,]    3.666667    5.666667
## [15,]    3.000000    3.000000
## [16,]    5.000000    3.666667
## [17,]    4.666667    3.333333
## [18,]    5.000000    3.666667
## [19,]    3.666667    3.333333
## [20,]    3.333333    5.666667
## [21,]    2.666667    3.000000
## [22,]    4.000000    4.000000
## [23,]    2.333333    3.666667
## [24,]    4.000000    6.333333
## [25,]    3.666667    3.666667
## [26,]    3.666667    6.000000
## [27,]    4.333333    3.666667
## [28,]    3.000000    5.333333
## [29,]    3.000000    6.666667
## [30,]    3.666667    3.000000
```

Weighted average of variables reflecting the factor, where weights are the factor loadings

```
factor1_score = fa_oblimin$scores[,'PA1']
factor2_score = fa_oblimin$scores[,'PA2']
cbind(factor1_score, factor2_score)
```

```
##       factor1_score factor2_score
##  [1,]  1.3095117650   -0.21876389
##  [2,] -1.2587732787   -0.25164387
##  [3,]  1.1775515843   -0.85550393
##  [4,]  0.1089419341    0.93755942
##  [5,] -1.3900297067   -1.37616380
##  [6,]  0.9997394889   -0.26391694
##  [7,]  0.4922287403   -0.92447082
##  [8,]  1.1758466287   -0.12970053
##  [9,] -0.7721901102   -0.56005431
## [10,] -1.1521863431    1.39886378
## [11,]  1.0662185508   -0.71449026
## [12,] -1.0603310535   -0.12402292
## [13,]  1.3759938095   -0.72788298
## [14,]  0.0659038623    1.24615689
## [15,] -1.2999957279   -0.68329822
## [16,]  0.9117397612   -0.30897646
## [17,]  0.6253007909   -0.54020271
## [18,]  1.5089522105   -0.27062069
## [19,] -0.9706323355   -0.68767526
## [20,] -0.4434195263    1.26730906
## [21,] -1.2802909619   -0.80582249
## [22,]  0.5553038563   -0.04052887
## [23,] -0.9723313258   -0.07896339
## [24,] -0.0006888493    1.76972440
## [25,]  0.8654111161   -0.09726684
## [26,] -0.2871278196    1.53849815
## [27,]  0.5600412833   -0.29800397
## [28,] -0.7363011563    0.66095237
## [29,]  0.0407579166    2.11393615
## [30,] -1.2151451039   -0.97502705
```

Pick a variable as a representative of the factor. Here, we are selecting the variable with the largest factor loading.

```
factor1_surrogate = survey[,'prevents_cavities']
factor2_surrogate = survey[,'attractive_teeth']
cbind(factor1_surrogate, factor2_surrogate)
```

```
##       factor1_surrogate factor2_surrogate
##  [1,]                 7                 4
##  [2,]                 1                 4
##  [3,]                 6                 3
##  [4,]                 4                 5
##  [5,]                 1                 2
##  [6,]                 6                 4
##  [7,]                 5                 3
##  [8,]                 6                 4
##  [9,]                 3                 3
## [10,]                 2                 6
## [11,]                 6                 3
## [12,]                 2                 4
## [13,]                 7                 3
## [14,]                 4                 6
## [15,]                 1                 4
## [16,]                 6                 4
## [17,]                 5                 4
## [18,]                 7                 4
## [19,]                 2                 3
## [20,]                 3                 6
## [21,]                 1                 3
## [22,]                 5                 4
## [23,]                 2                 4
## [24,]                 4                 7
## [25,]                 6                 4
## [26,]                 3                 7
## [27,]                 4                 5
## [28,]                 3                 3
## [29,]                 4                 7
## [30,]                 2                 2
```

Principal Components Analysis

Principal Components Analysis

- Used to reduce the dimensionality of the data by representing a large number of variables with a fewer number of components. Similar variables get grouped into the same component while dissimilar variables are placed in different components.
- Like factor analysis, it reduces the data based on similarity (typically inferred from correlation).
- It is used when the primary concern is to determine the minimum number of factors that will account for maximum variance in the data for use in subsequent multivariate analysis. The factors are called principal components.
- This reduced number of components can be used for further analysis instead of the original set of variables.

Principal Component Analysis

- Curse of dimensionality

Principal Components Analysis vs. Factor Analysis

- Factor analysis derives a mathematical model from which factors are estimated, whereas principal components analysis merely decomposes the original data into a set of linear variates.
- Factor analysis decomposes shared variance while principal components analysis decomposes total variance.
- In theory, only Factor Analysis can estimate the underlying factors
- In practice, the solutions generated from principal components analysis and factor analysis differ little.

Steps in Principal Components Analysis

1. Prepare Data
   - Impute missing values
   - Standardize Variables
   - Split into Train and Test
   - Exclude variables not relevant to analysis
2. Suitability for Principal Components Analysis
   - Correlations
   - Bartlett's Test of Sphericity
   - KMO MSA
3. Determine Number of Components
   - Scree Plot
   - Eigen-value
   - Parallel Analysis
   - Total variance explained
4. Describing Components
5. Apply Component Structure to Test Set

Prepare Data

- Impute missing values
  - Principal Components Analysis uses data on all variables. A missing observation on one variable will cause the entire row of data to be ignored for analysis. Therefore, it is important to impute missing values.
- Standardize Variables
  - To ensure all variables to receive the same weight in analysis.
- Split into Train and Test
  - Determine best component structure using train set
  - Apply component structure from Train set to generate components in Test
- Exclude variables not relevant to analysis

Suitability for Principal Components Analysis

- Since Principal Components Analysis analysis attempts to group similar variables, a basic requirement is that at least a few variables be related to each other.
- Correlation Matrix
  - A correlation matrix with some large and some small correlations is ideal for analysis.
- Bartlett's Test of Sphericity
  - Looks to see if there are at least some non-zero correlations by comparing correlation matrix to an identity matrix. A significant test indicates suitability for analysis.
- KMO's Measure of Sampling Adequacy (MSA)
  - Compares partial correlation matrix to pairwise correlation matrix. A partial correlation is a correlation after partialling out all other correlations. If the variables are strongly related, partial correlations should be small and MSA close to 1. If MSA > 0.5, data is suitable for analysis.

Determine number of components

A dataset with p variables will generate p components. The goal is to pick out the top few components that capture most of the variance in the original data. This is done based on the following criteria.

- Eigen-value
  - Select all components with eigen value greater than 1
- Scree Plot
  - Line graph of eigen values for each component. Ideal number of factors is indicated by a sudden change in the line graph or what is known as the *elbow*.
- Parallel Analysis
  - Simulate a dataset with same variables and observations as original dataset. Compute correlation matrix and eigen values. Now, compare eigen values from simulated data to original data. Select factors with eigen values in the original data greater than eigen values in the simulated data.
- Total Variance Explained
  - To ensure that the components represents the original variables sufficiently well, the total variance explained by components should be greater than 70%.

Describe Components

COLUMBIA UNIVERSITY
School of Professional Studies

- Examining elements comprising each component.
- Examine relationships amongst variables and between variables and components

Apply Component Structure

- In order to use the components, for downstream analysis,
  - apply the component structure to the test set.
  - extract the components
  - combine components with other variables in the original dataset

R Illustration

Principal Components Analysis

Data: Wine

You must read the data before trying to run code on your own machine. To read data use the following code after setting your working directory. To set your working directory, modify the following to set the file path for the folder where the data file resides.

`setwd('c:/thatawesomeclass/)`

```
wine = read.table('wine_m4.csv',header=TRUE,sep=';')
```

Explore Data

This dataset contains information on chemical properties of a set of wines (e.g., fixed acidity, alcohol) and a rating of quality.

```
str(wine)
```

```
## 'data.frame':    4898 obs. of  12 variables:
##  $ fixed.acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
##  $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
##  $ citric.acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
##  $ residual.sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
##  $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 ...
##  $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
##  $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
##  $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
##  $ pH                  : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
##  $ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
##  $ alcohol             : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
##  $ quality             : int  6 6 6 6 6 6 6 6 6 6 ...
```

Prepare Data

Missing Values

Principal Components Analysis uses data on all variables. A missing observation on one variable will cause the entire row of data to be ignored for analysis. Therefore, it is important to impute missing values. These can be imputed using any of a variety of imputation functions available in packages such as mice, caret, and missMDA

Standardize variables

It is important to standardize variables to ensure they receive the same weight in analysis. We do not need to standardize at this stage because the principal components analysis functions we will be using include an argument for standardizing variables.

Split Data

```
library(caret)
set.seed(1706)
split = createDataPartition(y=wine$quality,p = 0.7,list = F,groups = 100)
train_wine = wine[split,]
test_wine = wine[-split,]
```

Drop Outcome Variable

Drop the outcome variable, ** quality **, from train and test sets. Only keeping variables to be reduced to fewer components

```
train = train_wine[,1:11]
test = test_wine[,1:11]
```

Suitability for Principal Components Analysis

Principal Components Analysis is used to reduce the dimensionality of the data by representing a large number of variables with a fewer number of components. Similar variables get grouped into the same component while dissimilar variables are placed in different components.

Correlations

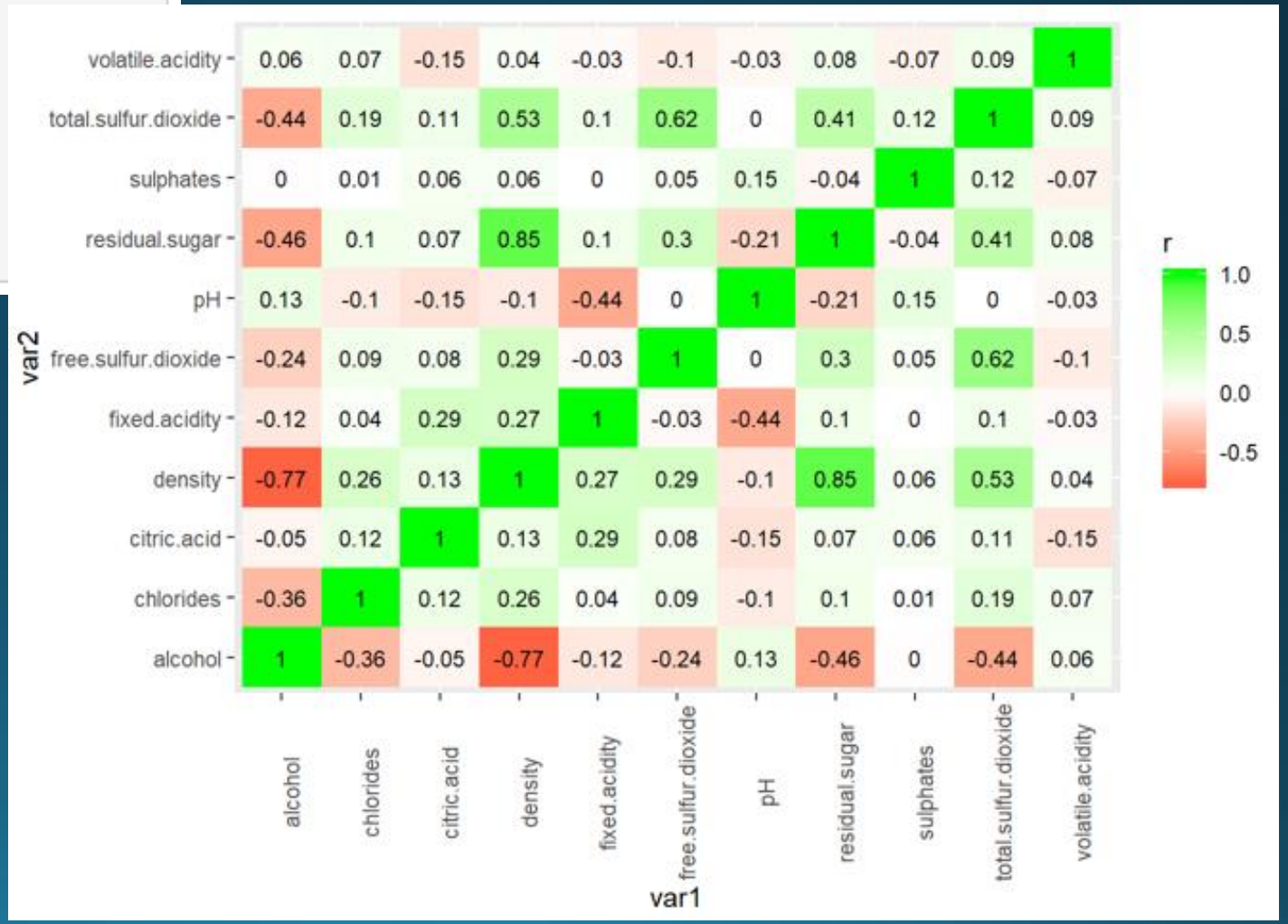
Similarity is typically judged by the correlation. Presence of large correlations in the data indicates similarity in the data. From the perspective of supervised learning techniques, correlated variables indicate redundancy.

```
round(cor(train,use='complete.obs'), 3)
```

```
##                       fixed.acidity volatile.acidity citric.acid residual.sugar
## fixed.acidity                 1.000           -0.030       0.291          0.100
## volatile.acidity             -0.030            1.000      -0.149          0.082
## citric.acid                   0.291           -0.149       1.000          0.075
## residual.sugar                0.100            0.082       0.075          1.000
## chlorides                     0.035            0.069       0.125          0.101
## free.sulfur.dioxide          -0.031           -0.096       0.077          0.297
## total.sulfur.dioxide          0.096            0.088       0.107          0.410
## density                       0.266            0.042       0.130          0.849
## pH                           -0.439           -0.033      -0.151         -0.205
## sulphates                    -0.002           -0.066       0.059         -0.044
## alcohol                      -0.121            0.064      -0.053         -0.458
##                       chlorides free.sulfur.dioxide total.sulfur.dioxide density
## fixed.acidity             0.035              -0.031                0.096   0.266
## volatile.acidity          0.069              -0.096                0.088   0.042
## citric.acid               0.125               0.077                0.107   0.130
## residual.sugar            0.101               0.297                0.410   0.849
## chlorides                 1.000               0.088                0.193   0.262
## free.sulfur.dioxide       0.088               1.000                0.618   0.289
## total.sulfur.dioxide      0.193               0.618                1.000   0.525
## density                   0.262               0.289                0.525   1.000
## pH                       -0.098               0.000                0.001  -0.105
## sulphates                 0.007               0.053                0.119   0.057
## alcohol                  -0.359              -0.240               -0.442  -0.770
##                           pH sulphates alcohol
## fixed.acidity         -0.439    -0.002  -0.121
## volatile.acidity      -0.033    -0.066   0.064
## citric.acid           -0.151     0.059  -0.053
## residual.sugar        -0.205    -0.044  -0.458
## chlorides             -0.098     0.007  -0.359
## free.sulfur.dioxide    0.000     0.053  -0.240
## total.sulfur.dioxide   0.001     0.119  -0.442
## density               -0.105     0.057  -0.770
## pH                     1.000     0.151   0.133
## sulphates              0.151     1.000  -0.005
## alcohol                0.133    -0.005   1.000
```

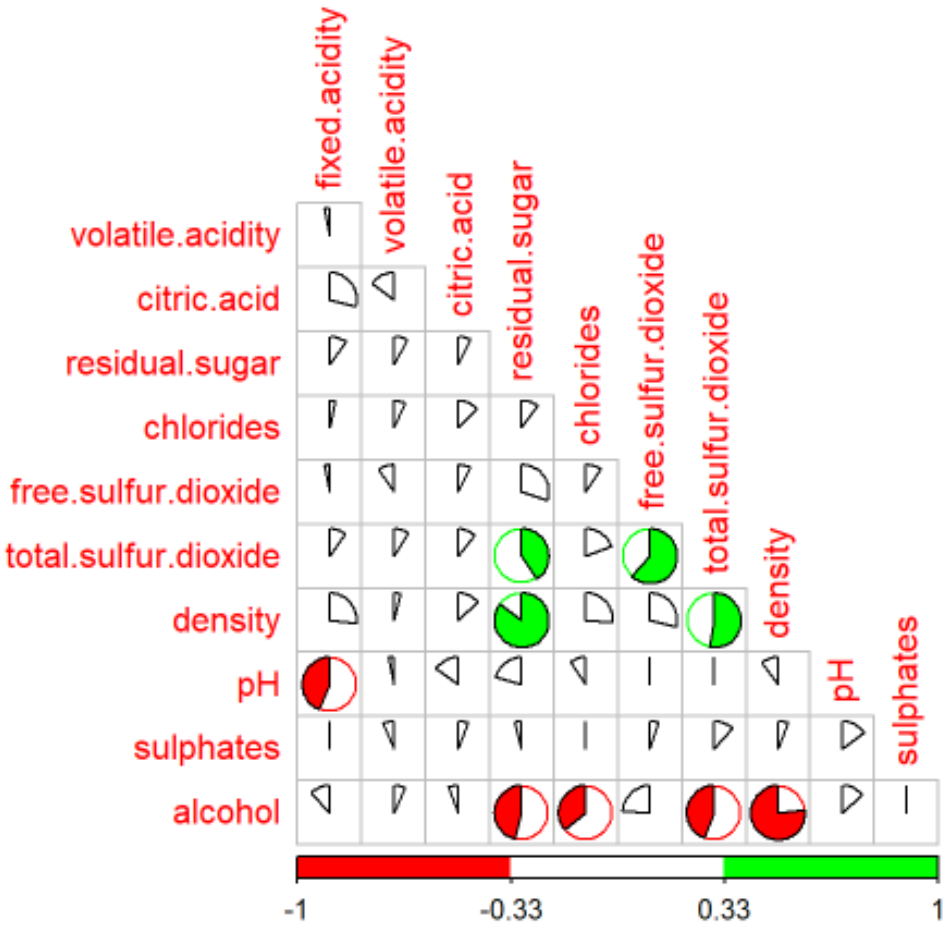
Visualizing the matrix to spot high correlations may be easier. Here, we use ggplot2 to construct a correlation matrix. High correlations indicates pairwise similarity.

```
library(tidyr); library(dplyr); library(ggplot2)
corMatrix = as.data.frame(cor(train))
corMatrix$var1 = rownames(corMatrix)
corMatrix %>%
  gather(key=var2,value=r,1:11)%>%
  ggplot(aes(x=var1,y=var2,fill=r))+
  geom_tile()+
  geom_text(aes(label=round(r,2)),size=3)+
  scale_fill_gradient2(low = 'red',high='green',mid = 'white')+
  theme(axis.text.x=element_text(angle=90))
```



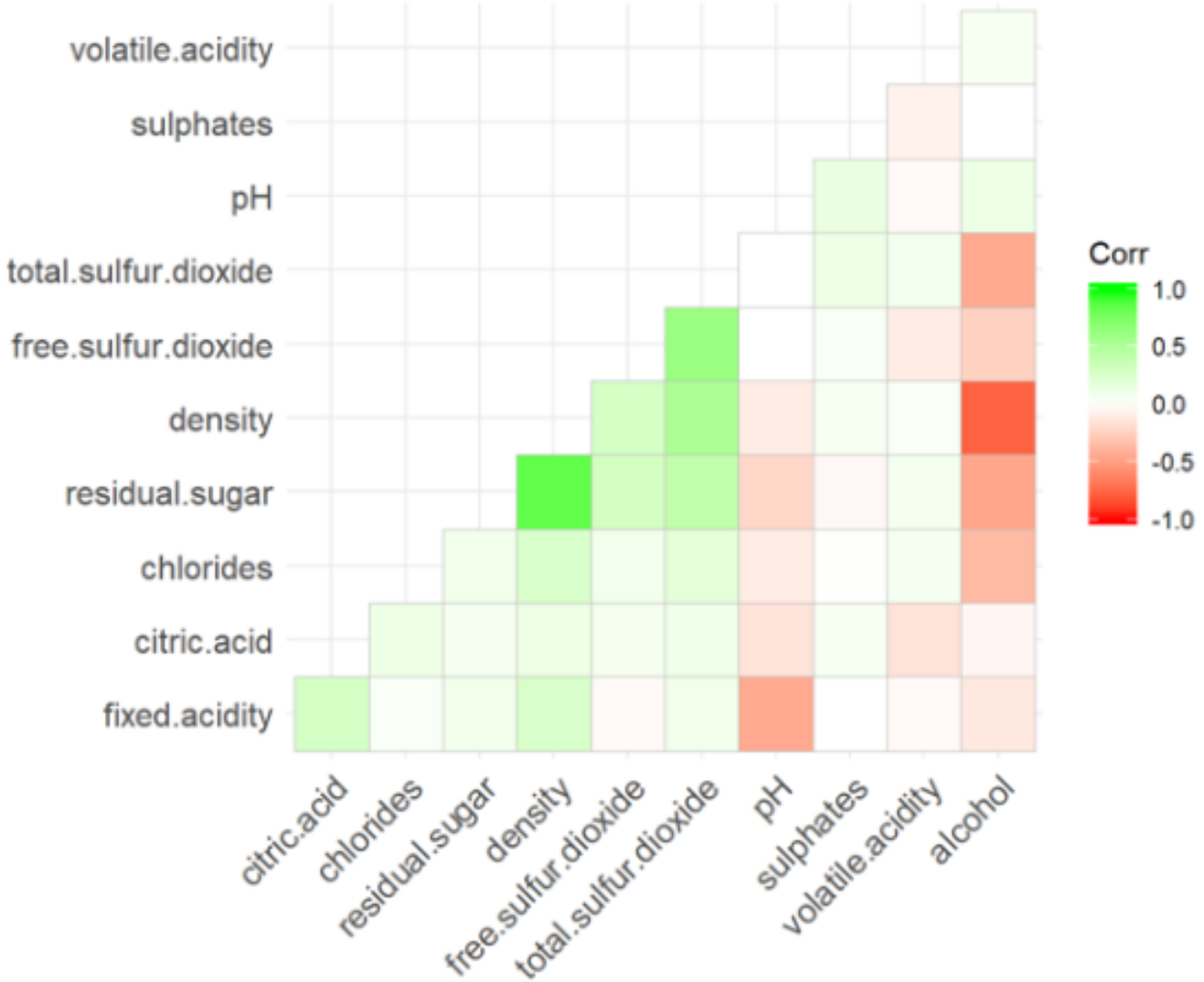
Of course, one could also employ corrplot or ggcorrplot to construct a similar plot. Both these plots have the added benefit of only representing the lower half and grouping variables based on similarity. In the following correlation heat maps, similar variables are placed side-by-side. Do you see any groupings?

```
library(corrplot)
corrplot(cor(train),type = 'lower',col = c('red','white','green'),method = 'pie',diag = F,order='original')
```



Here is a plot using ggcorrplot. This

```
library(ggcorrplot)
ggcorrplot(cor(train),colors = c('red','white','green'),hc.order = T,type = 'lower')
```



Bartlett's Test of Sphericity

Looks to see if there are at least some non-zero correlations by comparing correlation matrix to an identity matrix. A significant test indicates suitability for principal component analysis.

```
library(psych)
cortest.bartlett(cor(train),n = nrow(train))
```

```
## $chisq
## [1] 17203.22
##
## $p.value
## [1] 0
##
## $df
## [1] 55
```

KMO Measure of Sampling Adequacy (MSA)

Compares partial correlation matrix to pairwise correlation matrix. A partial correlation is a correlation after partialing out all other correlations. If the variables are strongly related, partial correlations should be small and MSA close to 1. If MSA > 0.5, data is suitable for principal component analysis.

```
KMO(cor(train))
```

```
## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = cor(train))
## Overall MSA =  0.38
## MSA for each item =
##        fixed.acidity     volatile.acidity          citric.acid
##                 0.18                 0.23                 0.68
##       residual.sugar            chlorides  free.sulfur.dioxide
##                 0.34                 0.66                 0.59
## total.sulfur.dioxide              density                   pH
##                 0.72                 0.41                 0.17
##            sulphates              alcohol
##                 0.19                 0.37
```

Determine Number of Components

A dataset with p variables will generate p components. Our goal is to pick out the top few components that capture most of the variance in the original data.

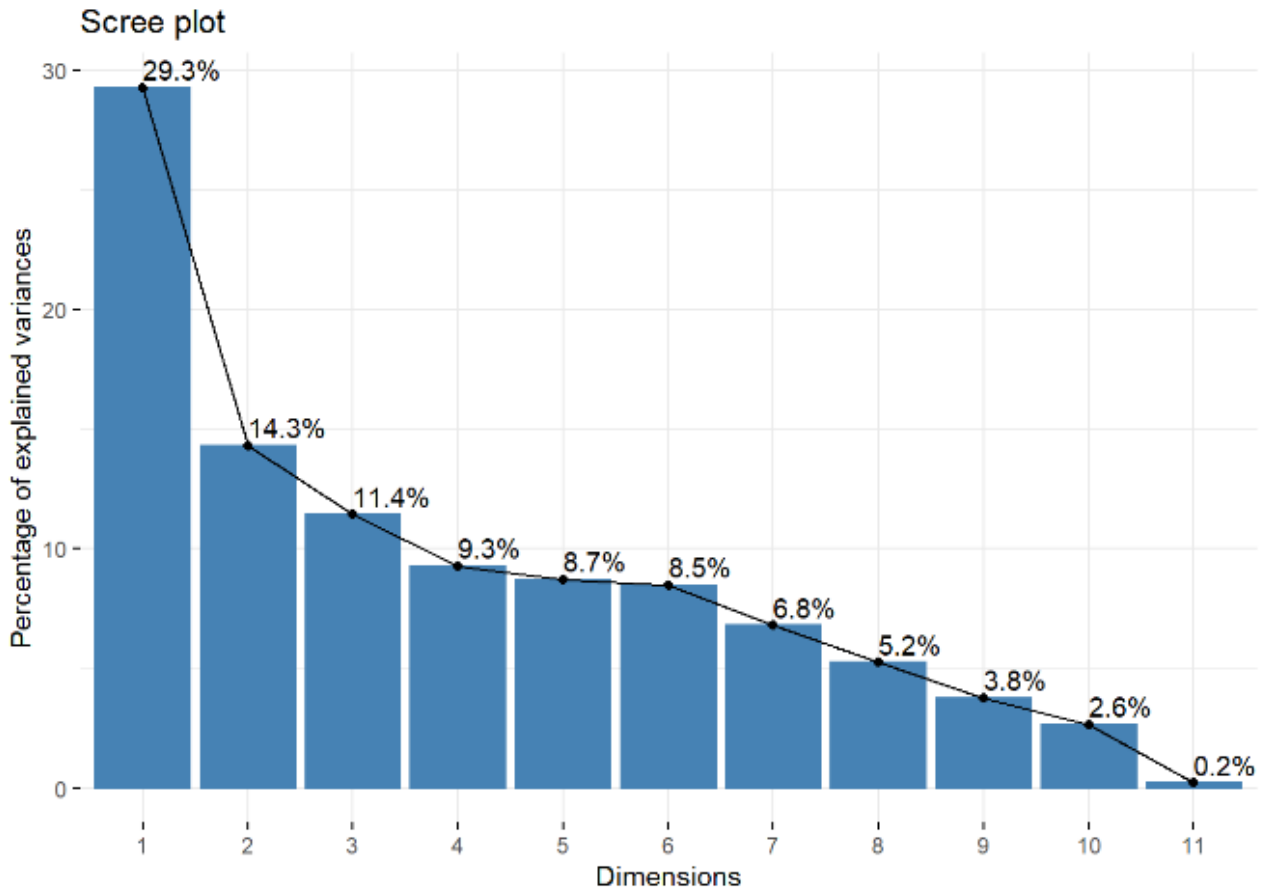
There are a number of functions for conducing principal components analysis including prcomp (from stats package), principal (from psych), caret, PCA (from FactoMineR), and dudi.pca (from ade4). The final results from principal components analysis are the same for all functions. The differences are in the optional arguments, helper functions and presentation of results. Here, we illustrate the use of two different functions: stats::prcomp and FactoMineR::PCA. We also use library(factoextra) to visualize the results.

Scree Plot

Line graph of eigen values for each component. Ideal number of components is indicated by a sudden change in the line graph or what is known as the elbow.

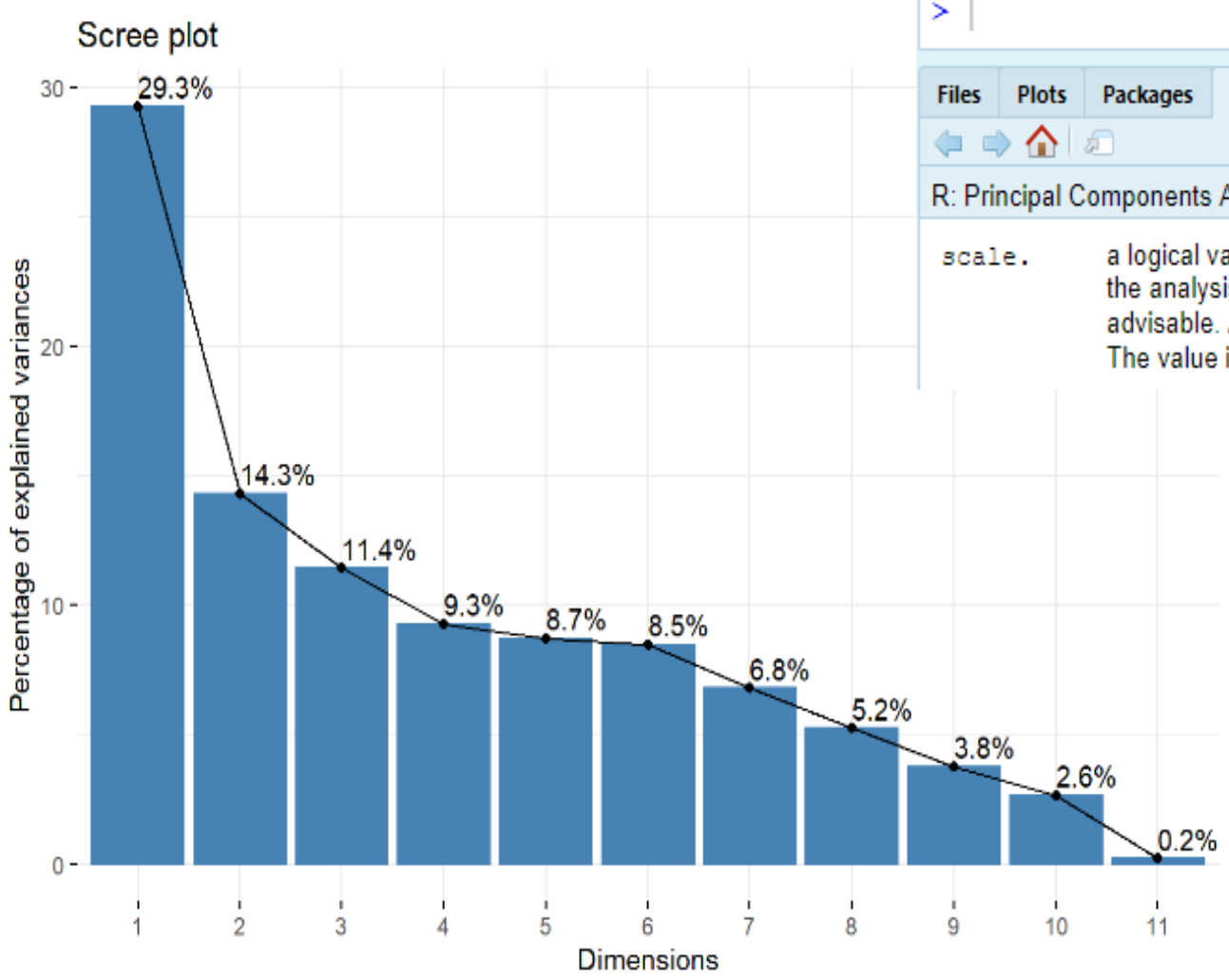
Using FactoMineR

```
library(FactoMineR)
pca_facto = PCA(train,graph = F)
library(factoextra)
fviz_eig(pca_facto,ncp=11,addlabels = T)
```



Using prcomp

```
pca = prcomp(train,scale. = T)
fviz_eig(pca,ncp = 11,addlabels = T)
```




```
> ?prcomp
>
```

Files | Plots | Packages | Help | Viewer

R: Principal Components Analysis — Find in Topic

`scale.` a logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place. The default is `FALSE` for consistency with S, but in general scaling is advisable. Alternatively, a vector of length equal the number of columns of `x` can be supplied. The value is passed to `scale`.

Eigen Value

According to the eigen-value criterion, all components with eigen value greater than 1 are selected.

Using library(FactoMineR)

```
pca_facto$eig
```

```
##         eigenvalue percentage of variance cumulative percentage of variance
## comp 1  3.21772952             29.252087                          29.25209
## comp 2  1.57098677             14.281698                          43.53378
## comp 3  1.25618853             11.419896                          54.95368
## comp 4  1.01829095              9.257190                          64.21087
## comp 5  0.95825442              8.711404                          72.92227
## comp 6  0.93142123              8.467466                          81.38974
## comp 7  0.74610042              6.782731                          88.17247
## comp 8  0.57619368              5.238124                          93.41060
## comp 9  0.41366920              3.760629                          97.17122
## comp 10 0.28894120              2.626738                          99.79796
## comp 11 0.02222408              0.202037                         100.00000
```

```
pca_facto$eig[pca_facto$eig[,'eigenvalue']>1,]
```

```
##         eigenvalue percentage of variance cumulative percentage of variance
## comp 1   3.217730              29.25209                          29.25209
## comp 2   1.570987              14.28170                          43.53378
## comp 3   1.256189              11.41990                          54.95368
## comp 4   1.018291               9.25719                          64.21087
```

Using prcomp

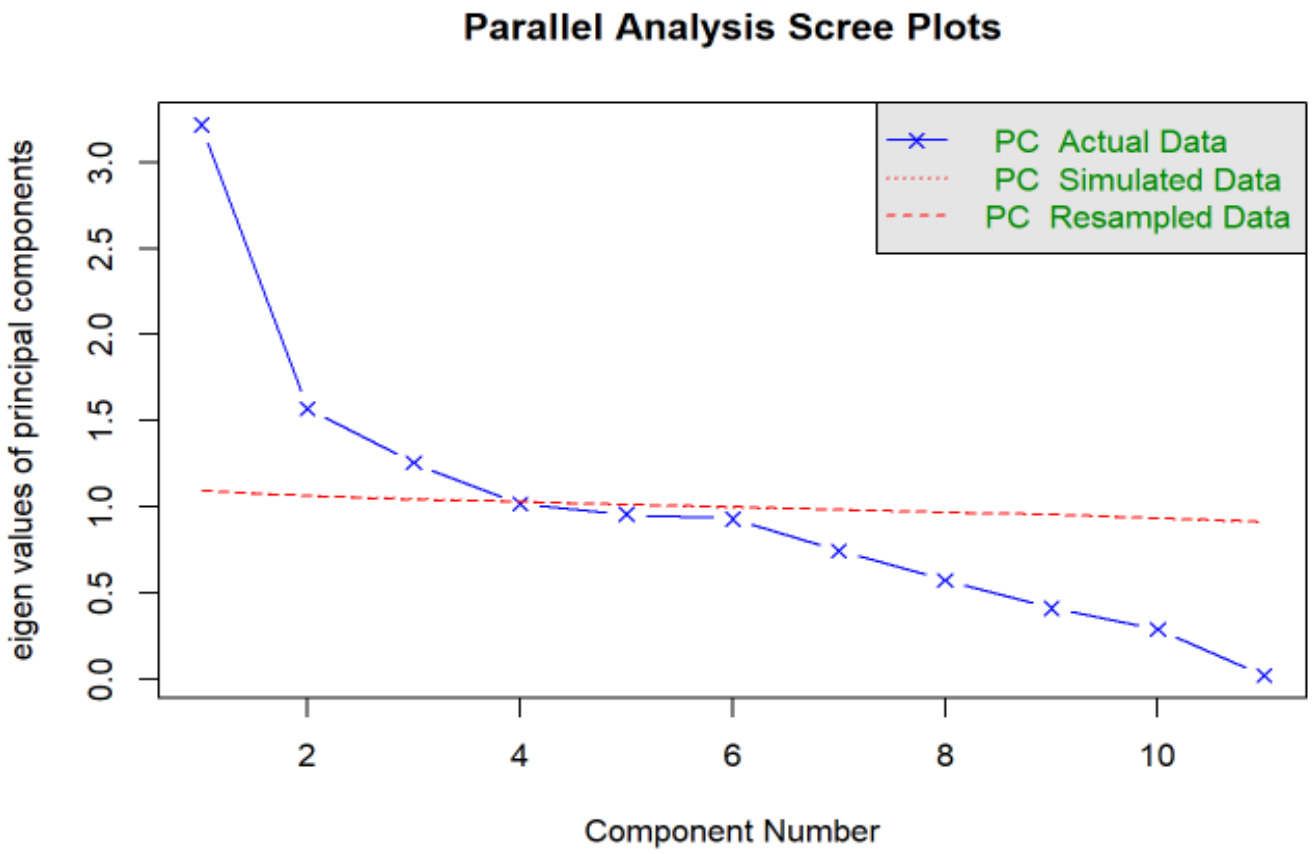
```
data.frame(component = 1:length(pca$sdev), eigen_value = (pca$sdev)^2)
```

component <int>	eigen_value <dbl>
1	3.21772952
2	1.57098677
3	1.25618853
4	1.01829095
5	0.95825442
6	0.93142123
7	0.74610042
8	0.57619368
9	0.41366920
10	0.28894120
11	0.02222408

Parallel Analysis

Simulate a dataset with same variables and observations as original dataset. Compute correlation matrix and eigen values. Now, compare eigen values from simulated data to original data. Select components with eigen values in the original data greater than eigen values in the simulated data.

```
library(psych)
fa.parallel(train,fa='pc')
```




```
## Parallel analysis suggests that the number of factors =  NA  and the number of components =  3
```

Total Variance Explained

To ensure that the principal components represent the original variables sufficiently well, the total variance explained by principal components should be greater than 70%.

```
pca_facto$eig
```

```
##         eigenvalue percentage of variance cumulative percentage of variance
## comp 1  3.21772952             29.252087                          29.25209
## comp 2  1.57098677             14.281698                          43.53378
## comp 3  1.25618853             11.419896                          54.95368
## comp 4  1.01829095              9.257190                          64.21087
## comp 5  0.95825442              8.711404                          72.92227
## comp 6  0.93142123              8.467466                          81.38974
## comp 7  0.74610042              6.782731                          88.17247
## comp 8  0.57619368              5.238124                          93.41060
## comp 9  0.41366920              3.760629                          97.17122
## comp 10 0.28894120              2.626738                          99.79796
## comp 11 0.02222408              0.202037                         100.00000
```

The results from each method differ widely:

- Scree Plot: 2, 3, or 4 components
- Eigen Value: 4 components
- Parallel Analysis: 3 components

However, the use of any fewer than 5 components would explain less than 70% of the original data. So, we go with a five-component structure suggested by the Scree plot.

Describe Components

Based on the analysis above we run a principal components analysis with five components.

```
pca_facto = PCA(train,scale.unit = T,ncp = 5,graph = F)
```

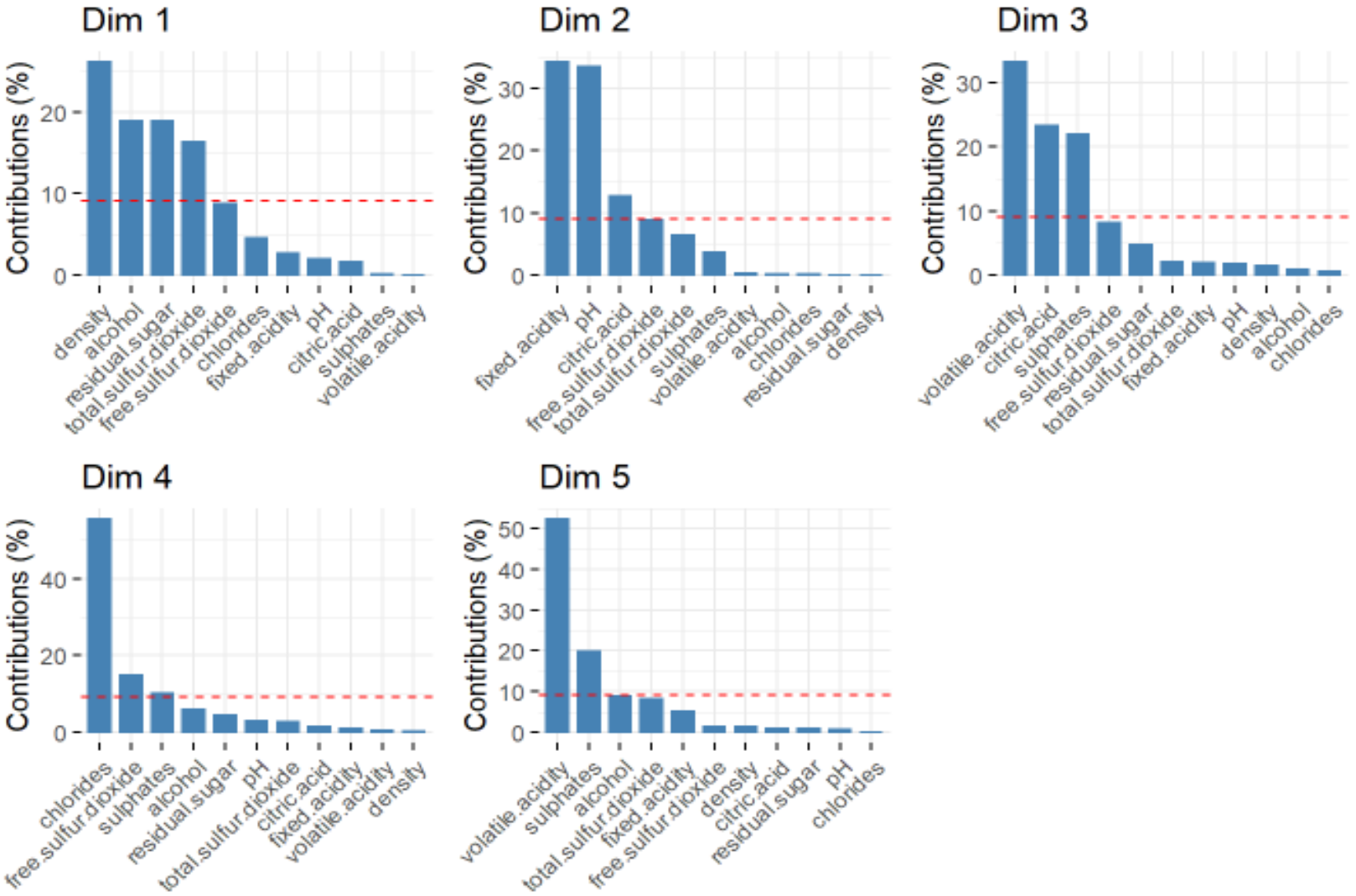
Examining elements comprising each component. For any component, size of loadings indicate the importance of the variable in describing the component.

```
pca_facto$var$contrib %>%
  round(2)
```

```
##                       Dim.1 Dim.2 Dim.3 Dim.4 Dim.5
## fixed.acidity          2.74 34.26  1.87  0.91  5.19
## volatile.acidity       0.02  0.28 33.22  0.59 52.40
## citric.acid            1.64 12.65 23.26  1.38  0.97
## residual.sugar        18.87  0.04  4.72  4.44  0.93
## chlorides              4.58  0.14  0.57 55.58  0.09
## free.sulfur.dioxide    8.72  8.84  8.13 14.92  1.50
## total.sulfur.dioxide 16.34  6.45  2.08  2.75  8.08
## density              26.18  0.02  1.47  0.16  1.30
## pH                     1.95 33.46  1.81  2.94  0.74
## sulphates              0.08  3.70 21.98 10.27 19.94
## alcohol               18.88  0.16  0.89  6.06  8.87
```

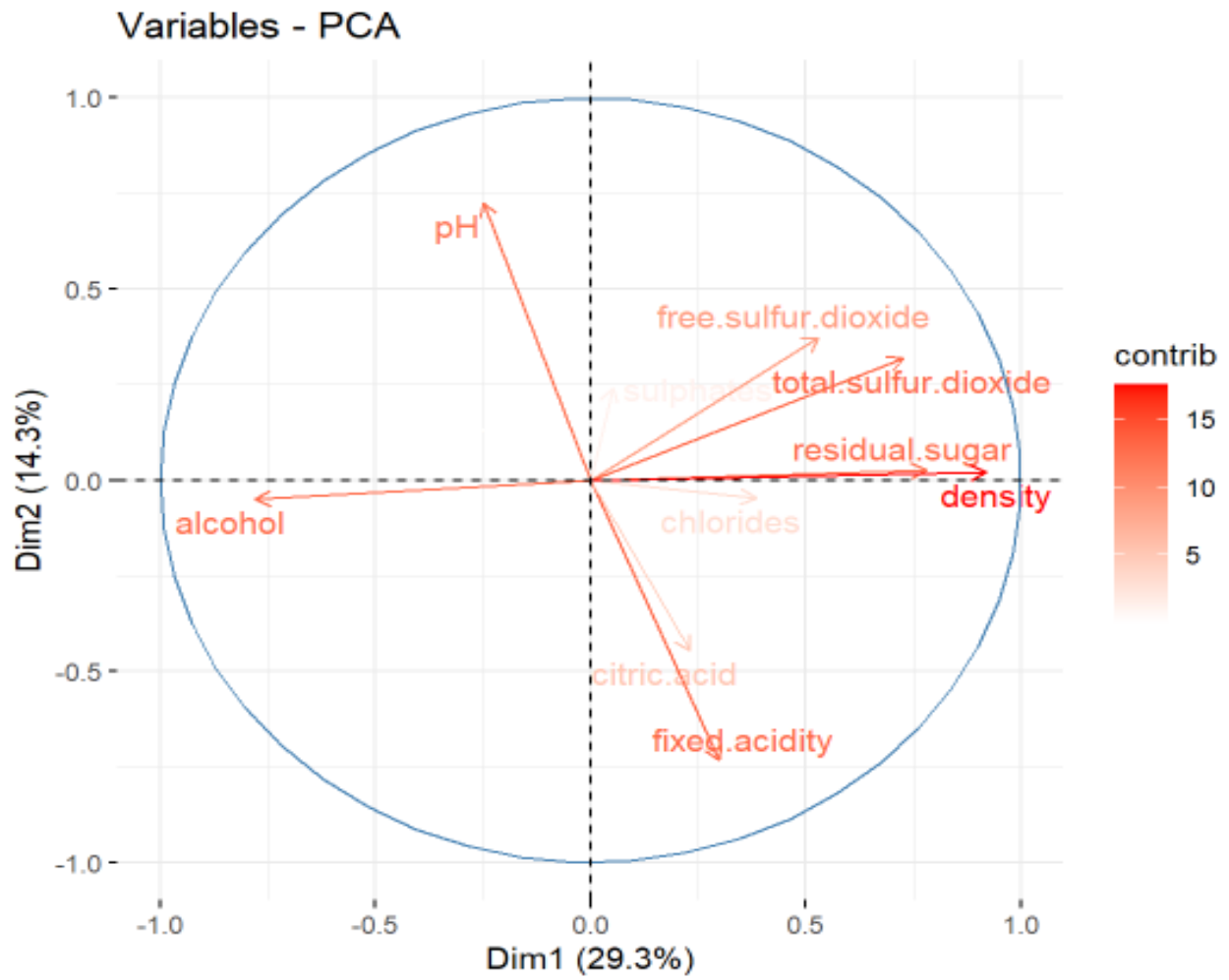
Contributions of each variable to each component are charted out below.

```
library(factoextra);library(gridExtra)
charts = lapply(1:5,FUN = function(x) fviz_contrib(pca_facto,choice = 'var',axes = x,title=paste('Dim',x)))
grid.arrange(grobs = charts,ncol=3,nrow=2)
```



Next, let us visually examine the relationships between variables. The following plot charts the first two most important components, Dim1 and Dim2 which explains 44% of the total variance. Angle between a variable and component reflects strength of relationship (smaller the angle, stronger the relationship) and the color indicates the contribution of the variable to the first two components. The picture is helpful but one must bear in mind that it only represents the first two components (44% of variance). All five components represent about 72% of variance.

```
fviz_pca_var(X = pca_facto,col.var = 'contrib',gradient.cols = c('red'),col.circle = 'steelblue',repel = T)
```



Apply Component Structure

In order to use the components, for downstream analysis, we first apply the component structure to the test set. Next, extract the components. Finally, combine components with other variables in the original dataset.

First, we illustrate this with the FactoMineR object

```
trainComponents = pca_facto$ind$coord
testComponents = predict(pca_facto,newdata=test)$coord

trainComponents = cbind(trainComponents,quality = train_wine$quality)
testComponents = cbind(testComponents,quality = test_wine$quality)
```

Next, we illustrate the same using prcomp() object. Note: The sign of the scores for prcomp() are the opposite of FactoMineR but the numbers are identical.

```
trainComponents2 = pca$x[,1:5]
trainComponents2 = cbind(trainComponents2,quality = train_wine$quality)

testComponents2 = predict(pca,newdata = test)[,1:5]
testComponents2 = cbind(testComponents2,quality = test_wine$quality)
```

Conclusion

- In this session, we
  - discussed the motivation for conducting dimension reduction
  - reviewed dimension reduction techniques
  - examined exploratory factor analysis
  - examined principal components analysis