Multivariate Statistics Cookbook

Brendon Mizener

2019-12-09

Contents

Chapter 1

Introduction

This is a Rmarkdown/bookdown document created using the bookdown package for RMarkdown. It was created as the final project for Research Methods 3: Topics in Multivariate Analysis, taken in the fall semester of 2019 from Dr. Herve Abdi. The TA was Ju-Chi Yu. Much of the code comes from class lectures or templates provided by them. Each of the chapters in the book is a different multivariate analysis technique. There are three fundamental types of analyses presented. Single table, two-table, and multi- (three or more) table.

For more information on the format of this document, check out the bookdown website.

The majority of the references are works by Dr. Abdi, and can be found here. The references for each chapter are provided and will be listed in full in the bibliography.

1.1 How to Use this Book

Each of the chapters contains a single multivariate technique, explanations, analyses, plots, and a general guideline for how to use each, and what kind of data are best suited to each type of analysis. Where possible, links or citations are provided to various articles, book chapters, and other resources to offer more in-depth guidance for each of the techniqes. The data analyzed come from a few sources. The introduction has an in-depth description of each of the datasets, and each of the recipe pages will have a more brief description of the data.

Chapter 2

What's in this book?

2.1 Quick lookup for techniques by data type

Part 1: The Backbone
The SVD and the GSVD

Part 2: Single Table Techniques
Chapter 2: Principal Components Analysis
- Use with: quantitative data

Chapter 3: Inferences for Principal Components Analysis
- Extension of PCA focusing on inferential techniques of permutation testing and bootstrapping.

Chapter 4: Correspondence Analysis
- Use with: qualitative data, nominal data, contingency tables

Chapter 5: Multiple Correspondence Analysis
- Use with: qualitative data, contingency tables, disjuctly coded tables

Part 3: Two Table Techniques
Chapter 6: Barycentric Discriminant Analysis
- Use with: Quantitative data with predetermined groups

Chapter 7: Discriminant Correspondence Analysis
- Use with: Qualitative data where you need to break down the relationships between the variables

Chapter 8: Partial Least Squares Correlation
- Use with: Two tables of data with two different sets of observations on the same set of variables, Two tables of data with two of the same sets of observations on different variables

Part 4: Multi-Table Techniques
Chapter 9: DiSTATIS
- Use with: Grouping data, each column is a judge

Chapter 10: MFA/STATIS
- Use with: Multiple differently scaled or sized tables containing information on the same observations

Bibliography

Appendix: Mel-Frequency Cepstral Coefficients

2.2 Datasets

2.2.1 Music Features

The first dataset, and the on on which the majority of these analyses have been performed, is the "Music Features" dataset; a dataset of spectral decomposition of 1000 30-second samples of audio files. These data were created for a machine learning/spectral decomposition project. The project can be found here. The originator's goal was to identify the genre of a given audio file based on spectral components. The features were extracted using libROSA. A general outline of the the data are below.

- Rows: 1000 observations of 30 second samples, 100 each from 10 different genres of music The genres identified in the dataset are Blues, Classical, Country, Disco, Hiphop, Jazz, Metal, Pop, Reggae and Rock.
- Columns: 30 data points on each
  - Filename: an identifier for each observation. This variable was used only ever used as the row names for the data.
  - tempo: measured in beats per minute, extracted by the software
  - beats: number of beats included in the recording - how many beats appear in each 30 second file, based on the tempo determined in the decomposition.
  - chroma: a vector indicating the strength of representation of chroma (pitch class) in the audio file, averaged across both time and frequency domains.
  - RMS: Root Mean Square (listed in the dataset as rmse) - the root mean square of the signal; a measure of volume
  - Spectral Centroid: effectively a measure of timbre, measured in hertz
  - spectral bandwidth: the width of the auditory spectrum in the file, also measured in hertz
  - rolloff: frequency point above which the file keeps less and less data, also measured in hertz
  - zero-crossing rate: how often the signal crosses the zero threshold, how busy the data are (often indicates speech or percussion instruments)
  - 20 Mel-Frequency Cepstral Coefficients: measure of the strength of a given frequency spectrum across the time domain. More info here.
  - Label: Genre the goal of the algorithm - identifying audio files by spectral content. This variable is used as the design variable for these analyses, to compare groups.
- Each of these variables, except for Filename and Genre, are numerical. They all represent very different values, which means that in order to be analyzed, they will need to be scaled in addition to being normalized.

2.2.2 Vowels & Colors

This dataset was collected by Maxim Chastaing as part of his research on speech, language, and cognition.[1] In this experiment, French participants were asked to associate vowels with colors. Each participant was

[1] As of the date of final submission of this project, I have yet to be able to track down which specific publication it is, but I think it's this one

presented with six colors and six vowel sounds, used in words that highlighted the sound in question, and asked which vowel associated most closely with which color. The participants could place more than one vowel with each color, but could only put each vowel in a single color 'bin'. The data take the form of a contingency table, shown below.

	Yellow	Green	Orange	Blue	Red	Violet
i	46	17	2	11	42	5
y	18	48	6	12	6	6
e	17	20	13	29	4	8
a	8	7	5	17	30	6
o	18	9	19	19	21	10
u	1	2	15	14	16	16

2.2.3 Beer Sorting Data

This dataset comes from an experiment done in Mexico, where participants were asked to sort beers into groups of their own determing. The majority of the beers come from Mexico and Central America, but there are a few European beers thrown in. There are three rows containing data on the participants: participant number, gender, and whether the participant reported that they preferred "craft" beer or "industrial" beer for their own consumption. The rows are each of the beers, and the intersection of a row and a column shows the group in which a participant placed a beer.

2.3 Tools for Analysis

Research Questions:
For each analysis, guiding principles are provided to aid in forming research questions, and there are suggested questions that are informed by the data and the analysis technique. In general, these questions will be exploratory in nature, as these analyses do not function in the same way as an ANOVA or a T-test might. The results from these analyses and the answers to these questions serve as useful guidelines to form hypotheses for further testing.

Packages
The following packages are being used for the analyses that follow. The specific packages used for each analysis are specified at the top of each page. For more information on any of these, load the package and then type ?packagename in the command line

General packages for working with data:
library(tidyverse) *Tidyverse contains a number of useful packages designed to work with data.*
library(knitr) *knitr is useful for displaying data in tables, using kable. Note: Kable works well with HTML, not powerpoint, and be careful with PDF.*
library(kableExtra) *Allows for more flexible and powerful tables, using the pipe syntax (%>%)*
library(pander) *Another helpful package for simple, clean tables. Works well for pdf.*
library(Matrix)

Packages for Displaying or Visualizing Data:
library(corrplot) *Specifically used for correlation plots*
library(ggplotify) *Works with ggplot2 (tidyverse) Necessary for turning some objects into graphical objects (grobs)*
library(ggpubr) *Works with ggplot2*
library(grid) *Allows you to arrange plots for effective display. There are other packages that do this as well*

library(gridExtra) *Works with grid to create more functionality.*
library(wesanderson)

Packages for data analysis
(these should be updated regularly from Dr. Abdi's github)
To update: devtools::install_github("herveabdi/[package]")

library(PTCA4CATA)
library(ExPosition)
library(InPosition)
library(TExPosition)
library(TInPosition)
library(MExPosition)
library(factoextra)
library(data4PCCAR)
library(DistatisR)

Note: MExPosition must be updated from the Cran Repository, then manually installed using the GUI

2.3.1 A word on Plots

There are some plots that are shown in this book almost a full page, and some that are shown much smaller in the interest of saving space. One of the beautiful things about r markdown/bookdown is that because the images are embedded, not screenshotted or copied and pasted, they are visible full size and you can zoom in and see incredible detail. So if there are any plots that you think aren't visible enough, just zoom in for detail.

Part I: The Backbone

Chapter 3

SVD and GSVD

This is not intended as an exhaustive review of the process, but rather a basic conceptual breakdown allowing the reader to approach the topic. All of the information is from Hervé Abdi (2007b) and Hervé Abdi and Williams (2010c). Consult those references for more information on this technique and its application to the analyses in this book.

The analyses in this book are all based on linear and matrix algebra. The "backbone" of these multivariate analyses, specifically, is the Singular Value Decomposition (SVD) and the Generalized Singular Value Decomposition (GSVD). In each of these techniques, we perform eigen-decomposition of the data table or tables, viewed as a matrix or set of matrices (Hervé Abdi 2007b). These techniques differ, among other ways, in how the constraints are placed on the decomposition. However, both techniques focus on the orthogonality of the dimensions of decomposition.

The eigen-decomposition of the matrix gives us three matrices. While it may seem counter-intuitive in that we are creating *more* matrices, these new matrices represent a dimensionality reduction and allow us to later visualize the information in a way that is more intuitive than the initial plot. Of the three matrices extracted, two are orthogonal and one is a diagonal matrix of singular values. Specifically, we decompose matrix $\mathbf{X}$ so: $\mathbf{X} = \mathbf{P}\Delta\mathbf{Q}^{\mathrm{T}}$. Here, the columns of the two orthogonal matrices, P and Q, are the left singular vectors of X and the right singular vectors of X, respectively. They can also be understood as the normalized eigenvectors of X * X^{T} and X^{T} * X, respectively. Δ is the diagonal matrix and represents the square roots of the eigenvalues of X * X^{T} and X^{T} * X.

For analyses that use the SVD, such as PCA, we put orthogonality constraints on the SVD: $\mathbf{X} = \mathbf{P}\Delta\mathbf{Q}^{\mathrm{T}}$, such that $\mathrm{P}^{\mathrm{T}}\mathrm{P} = \mathrm{Q}^{\mathrm{T}}\mathrm{Q} = \mathrm{I}$[1]. That is to say, the rows and columns of the original matrix are constrained to orthogonality. In analyses like CA that use a *generalized* singular value decomposition (GSVD), we add additional constraints using the masses and weights of the rows and the columns, respectively, such that P * M * P^{T} = Q * W * Q^{T} = I, where M is the masses of the rows and W is the weights of the columns. This will make more sense conceptually once you read those chapters.

Conceptually what is happening here is that we are extracting variance, in the form of eigenvalues. These can be tested for significance using permutation testing (more on that in the Inference PCA chapter). Extracting this information has the effect of rotating the original plot of data such that the line representing the first eigenvalue becomes the X-axis, and by definition and virtue of orthogonality, the second eigenvalue becomes the Y-axis. This allows one to see what 'principal component', 'latent variable', or 'salience' is causing the majority of the variance of the data. These variables may or may not be explicitly indicated in the original data, and can give you better insight into what separates observations in a systematic manner.

[1] where I is the identity matrix

We also see how variables 'load' or drive the variance of the principal components. This can tell us what variables are positively correlated, which are negatively correlated, and which are unrelated, or orthogonal, to one another. Because we'll be looking at these graphically instead of numerically, it allows a more intuitive understanding of the relationships between the variables. As you'll see in the variable loadings maps later, the squared cosine of the angle between any two variables is the correlation coefficient of those variables.

This solves one of the problems of traditional cartesian graphs, namely that plotting more than three variables against one another in a single graph is basically impossible to visualize and interpret, and interpreting only three is extremely difficult. This technique allows you to compare many variables simultaneously in a way that allows easy interpretation. However, it is not a replacement for traditional regression or ANOVA analyses, as the results of these analyses don't offer conclusive results. They instead show us how things are related in ways we may not have been able to see otherwise, and offer suggestions for future hypothesis testing.

Part II: Single Table Techniques

Chapter 4

Principal Components Analysis

4.1 Intro to PCA

PCA (Principal Component Analysis) is a statistical technique used to analyze large datasets of multiple inter-correlated variables. It is the oldest and most commonly known and used multivariate analysis technique. It's fundamentally about dimensionality reduction and comparing variables, and is the most fundamental of the techniques presented in this cookbook. As such, many of the explanations found here are not repeated on other cookbook pages, which instead reference back here. PCA also helps us to analyze the relationships between variables visually as opposed to numerically, which helps us form more intuitive and informed responses to the results of such analyses.

It uses least squares regression modeling to extract latent orthogonal variables not explicitly included in the dataset. These new variables are called principal components. The first principal component is identified by minimizing the variance to the line of best fit. The second principal component is identified by maximizing the inertia, essentially maximizing the distance to a line orthogonal to the first component. Once the two principal components are identified, matrix rotations are used to establish new axes, and that all of the original observations are plotted onto the new "factor space".

The two dimensions of the original dataset (rows and columns) are viewed as observations and variables, respectively. The observations are plotted onto the factor space as "factor scores" and the variables are plotted onto the factor space as "loadings". The two identify similar information, but are interpreted differently. For a PCA, these two plots are separate.

Factor scores are interpreted as how extreme a given observation is in a given dimension of a factor, and the similarity between observations is determined by how close or distant they are in the factor space. Loadings indicate how much a given variable contributes to a dimension of information extracted by the PCA. Variables that lie further along the principal components contribute more to that dimension and observations closer to the barycenter[1] contribute less. The similarity or difference between two variables is determined by the angle between them, regardless of where they lie on the the factor space.[2]

> The distance from the origin is important in both maps, because squared distance from the mean is inertia (variance, information; see sum of squares as in ANOVA/regression). Because of the Pythagorean Theorem, the total information contributed by a data point (its squared distance to the origin) is also equal to the sum of its squared factor scores.
> - Michael Kriegsman

[1] Center of Gravity; origin.

[2] The correlation between to variables is equal to the squared cosine of the angle between them. Angles approaching 0 or 180 degrees are highly correlated, those approaching 90 degrees are less so. Angles that are exactly 0 or 180 degrees have a correlation of 1 or -1, respectively, and angles that are exactly 90 are completely uncorrelated, or orthogonal.

All of the information in this chapter comes from lecture notes and Hervé Abdi and Williams (2010c), consult those for more detail on this technique.

4.1.1 Strengths & Weakness

Strengths
PCA is a great tool for analyzing quantitative data, and serves as a good basis for beginning investigations of that type. It's also fundamental to many of the other analyses presented in this cookbook, so understanding the plots are vital to understanding the other analyses.
Weaknesses
It is limited in that it doesn't handle qualitative data, and it doesn't allow for easy comparison of variables and observations. Also, because of the nature of the analysis, all we're really doing is looking at what is really in the data, and therefore the results of a PCA are not easily generalizable. For those purposes we need to run bootstrapping and permutations tests on the PCA data. For more on bootstrapping and permutation testing, see Hesterberg (2011) and Berry, Johnston, and Mielke (2011), respectively. These techniques are also discussed in more detail in the Inference PCA chapter.

4.1.2 Dos and Don'ts

Do:
- Remember that this analysis is primarily about the variables. We are seeing in the factor plots how each of the observations relate to the variables, but we're looking at which variables are measuring what, how similar those measurements are, and how we can reduce the dimensionality of the data.
- Be aware of the structure of your data. PCA requires that data be centered, but not necessarily scaled. If the data you are inputting are already centered, you don't need to re-center. That won't do anything. Whether you want to scale or not depends on your data. If you have data that are vastly different, i.e. age, income, likert scales, reaction time, all in the same dataset, you need to scale, so that they can be compared. If you only have one type of data, you can decide whether you want to scale based on the dataset. - Make sure you understand your variables. If you don't have a comprehensive understanding of what the numbers mean in the original dataset, the analysis won't help.
- Be open-minded about what the data reveal. You might be surprised.
- Compare observation factor scores by distance.
- Compare variable factor scores/loadings by correlation coefficient (Squared cosine).

Don't:
- Try to compare factor scores for rows and columns (observations and variables) directly.
- Forget about how orthogonality works.
- Come into this analysis with preconceived notions of what the data are.
- Make the mistake of "over-fitting"

Research Questions:
The PCA technique lends itself to research questions that get at underlying, "invisible" components that separate the observations. The specific questions should be guided by the data themselves. Sometimes, a PCA can offer insights into what other questions we should ask. For this specific analysis, some useful questions might be:
- Do these audio files vary systematically?
- Are the underlying variations in the data due to musical features, spectral features, or some combination of the two?
- Are there significant, systematic spectral differences between genres of music, and if so, what are they?
- We can also rephrase that to: Are there genre-specific signal markers that allow us to classify the files by genre?
- Using what we know about music, can we make any guesses as to why these variations occur?

4.2 Data

```
mfdata <- read.csv("data.csv")
rownames(mfdata) <- c(as.character(mfdata$filename))
colnames(mfdata) <- c("f.n.", "bpm", "b", "ch", "rmse", "spec_c", "spec_b", "r_o", "zcr",
                      "mfcc1", "mfcc2", "3", "4", "5", "6", "7", "8", "9", "10",
                      "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "lbl")
mf.genre <- mfdata$lbl
mfdata <- mfdata[ ,-c(1,30)]
```

The dataset for this analysis is a dataset of spectral decomposition of 1000 30-second samples of audio files, aimed at identifying the genre of a given audio file based on spectral components source. The features were extracted using libROSA. Each row (observation) is the libROSA output for the file identified in the row name. There are 1000 rows, with 100 files from each of 10 genres: Blues, Classical, Country, Disco, Hip hop, Jazz, Metal, Pop, Reggae and Rock. There are 28 numerical variables and one factor/design variable in the dataset:

- tempo (bpm): measured in beats per minute, extracted by the software
- beats (b): number of beats included in the recording - how many beats appear in each 30 second file, based on the tempo determined in the decomposition.
- chroma (ch): a vector indicating the strength of representation of chroma (pitch class) in the audio file, averaged across both time and frequency domains.
- Root Mean Square Error (rmse): the root mean square of the signal; a measure of volume
- Spectral Centroid (spec_c): effectively a measure of timbre, measured in hertz
- spectral Bandwidth (spec_b): the width of the auditory spectrum in the file, also measured in hertz
- Roll-off (r_o): frequency point above which the file keeps less and less data, also measured in hertz
- Zero-Crossing Rate (zcr): how often the signal crosses the zero threshold, how busy the data are (often indicates speech or percussion instruments)
- 20 Mel-Frequency Cepstral Coefficients (mfcc1, mfcc2, 3, ...): measure of the strength of a given frequency spectrum across the time domain. More info here.
- Genre (lbl): the goal of the algorithm - identifying audio files by spectral content, used as our design variable.

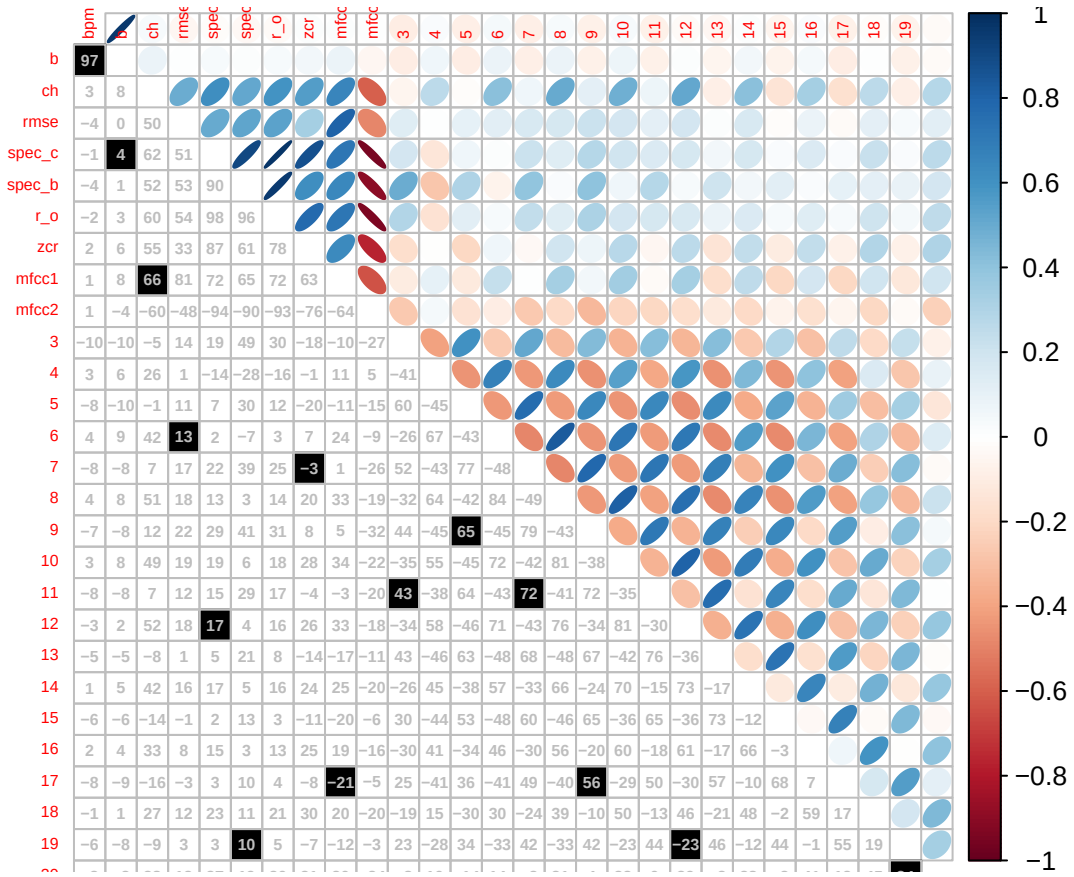
	b	ch	rmse	spec_c	spec_b	r_o	zcr	mfcc1	mfcc2	3
blues.00081.au	50	0.3802602	0.2482623	2116.943	1956.611	4196.108	0.1272725	-26.92978	107.33401	-46.809993
classical.00091.au	44	0.2237383	0.0447457	2192.798	1911.986	4066.467	0.1524379	-251.31345	88.72515	-41.767192
country.00013.au	63	0.3680927	0.0800457	2812.346	2842.053	6062.663	0.1295650	-126.16004	77.63675	3.165314
disco.00063.au	64	0.5478499	0.2939197	2583.278	2626.311	5855.473	0.0997726	-51.75267	70.33190	-3.919614
hiphop.00034.au	97	0.4699237	0.2279124	3191.045	2735.905	6353.995	0.1691660	5.87653	53.23913	-8.983690
jazz.00004.au	28	0.1717823	0.1087860	1039.623	1422.303	1838.214	0.0477688	-270.26482	137.57173	5.658533
metal.00088.au	54	0.4215711	0.1333055	1988.120	1986.989	4148.192	0.0924201	-129.70020	104.03440	-26.972440
pop.00024.au	57	0.4109899	0.2075838	3301.782	3175.657	7302.598	0.1445267	-40.80207	61.52205	9.868217
reggae.00009.au	76	0.3830216	0.0866705	1609.641	1990.699	3269.547	0.0605760	-232.62434	131.23940	2.697012
rock.00019.au	58	0.4359108	0.1122276	2321.186	2118.985	4702.063	0.1281995	-84.17958	102.33721	-41.332401

Note:
In this table, filename has been moved to the row names and is excluded from analyses. In the interest of saving space, only the first 3 MFCCs are shown. In the analyses to follow, the names of the MFCCs >2 have been shortened to just their numeral (i.e. MFCC3 = 3)

4.2.1 Data Visualization: Correlation Plot

In order to get an idea of what the data look like overall, we run a correlation analysis `cor()` and plot it using corrplot.

```
cor.res.full <- cor(mfdata)
corrplot(cor.res.full, diag = F, type = "upper",
         method = "ellipse", tl.cex = .5, tl.pos = "n") %>%
  corrplot(cor.res.full, add = TRUE, diag = F,
           type = "lower", method = "number",
           addCoefasPercent = T, col = "grey",
           tl.cex = .5, number.cex = .5, tl.pos = "l")
# If you want to record plots to a powerpoint, use recordPlot() to record whatever plot
# is active. Use the code at the end to save the plots.
#cor.plot.f <- recordPlot()
```



Reading this plot:

- This is a correlation plot showing us how the variables correlate with each other.
- There are other options for ordering the variables, using the parameter `order`. They include "FPC", which orders the variables they way they load on the first component, left to right. Explore these options for whichever makes the most sense for your data.
- The two halves of the plot display the same information.
- The top half uses color and shape to show the strength and direction of correlation.
- The bottom half uses values between -100 and 100, using the parameter `addCoefasPercent`, to show the correlation coefficient between the variables.

A couple of things to note:

- Tempo (bpm) and beats (b) are correlated with each other (they effectively measure the same thing) and basically with nothing else. This makes sense, as tempos are not unique to any given genre
- Many of the spectral elements (chroma, rmse) show a strong positive correlation with each other and a strong negative correlation with MFCC2.

- The MFCCs seem to be measuring the same thing - odd and even MFCCs are anti-correlated, which makes sense since the actual creation of the MFCCs involves a de-correlation process, so that neighboring triangular windows capture different information.
- That being said, MFCCs 1 and 2 seem to have a fairly strong negative correlation with each other, but they are approximately orthogonal to the other MFCCs.

4.3 Analysis

```
# This is the line of code that actually runs the PCA.
# The output from this is necessary for all of the other graphs and functions.
mfpca.res <- epPCA(mfdata, center = TRUE, scale = "SS1",
                   DESIGN = mf.genre, graphs = FALSE)
# This line runs both bootstraps and permutation tests on the data.
# Technically you could run only this line,
# as you get results for both fixed data and inference data.
mfpca.inf.res <- epPCA.inference.battery(mfdata, center = TRUE, scale = "SS1",
                                         DESIGN = mf.genre, graphs = FALSE,
                                         test.iters = 1000)
```

```
## [1] "It is estimated that your iterations will take 0.5 minutes."
## [1] "R is not in interactive() mode. Resample-based tests will be conducted. Please take note o
## ================================================================================
```

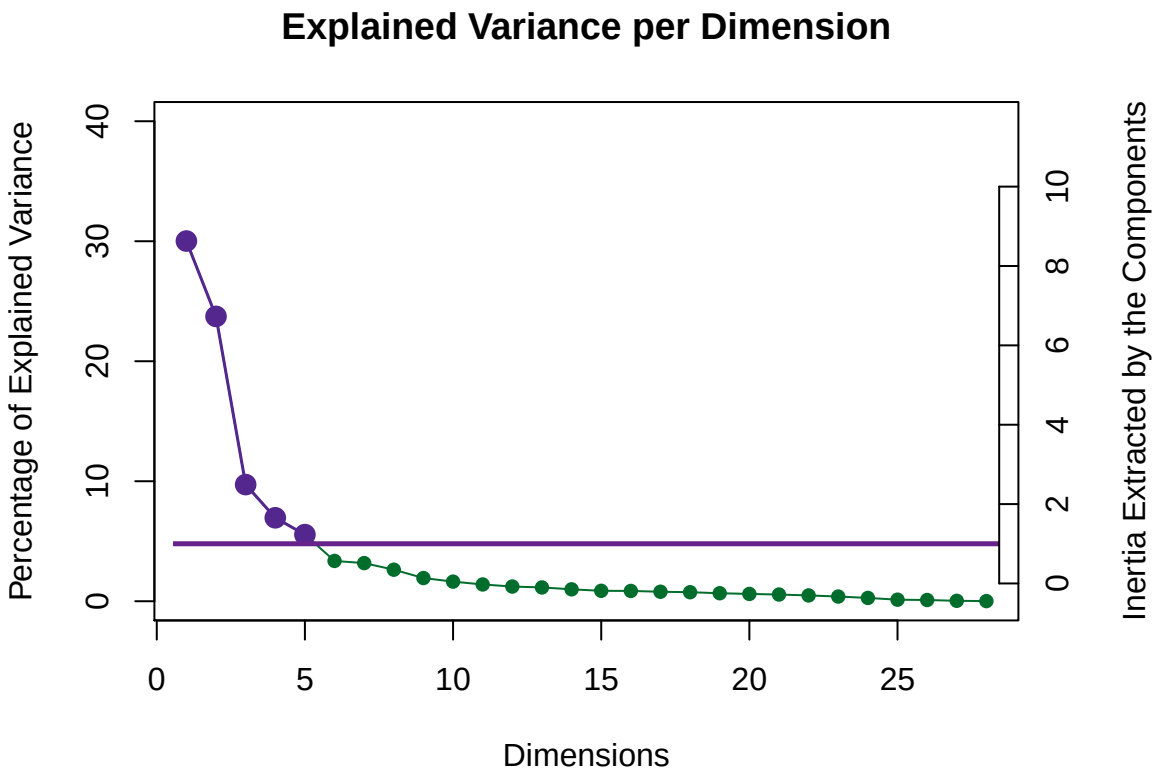
4.4 Results

4.4.1 Scree Plot

Reading this plot:
A scree plot plots eigenvalues by how much information there is in each component. Each of the dots on the scree plot identifies a dimension from the factor space in which there is variance, there are up to k - 1 dimensions from which variance can be extracted, where k is the the lower of either the number of variables in your analysis or the number of observations (i.e. min(nrow(DATA), ncol(DATA))), but your analysis of the dataset should focus on only the ones that take up the majority of the variance. A good rule of thumb for this is to look at the eigenvalues that fall above the "elbow", excluding the dimensions that fall below the noise threshold. A good way to visualize this is to connect the dots (as is done in the plot) and draw a straight line that extends from the bottom right hand corner all the way across the graph, and the point at which the dots start to land above this line is the noise threshold. On the plot below we see two other methods of determining the significance of each dimension. The first is the Kaiser criterion, where we look at the average of the eigenvalues, plotted as a horizontal line over the plot. This isn't a rule, by any means, but it does give us an idea of what dimensions are important. The second is the result of the permutation tests. The permutation tests tell us whether each of the eigenvalues fall within the most extreme 5% of values. Again, however, this doesn't tell us necessarily which eigenvalues are important, it just shows us what values are significant. It just so happens that in this case, the Kaiser criterion, the elbow test, and the permutation tests are showing us the same dimensionality. Probably a good clue that there are 5 dimensions of data worth looking at in this set. Bottom line is, scree plots give you two things: an idea of the true dimensionality of your data and a measure of the variance explained by the components. Remember that just because the numbers are small, doesn't mean that they can't be significant. The first eigenvalue is in a sense an omnibus test, it shows us whether or not there is any information in the data that isn't just noise. Beyond that, just because there are 28 dimensions in this analysis doesn't mean that there are 28 dimensions worth looking at, it's up to the researcher/observer to determine how many levels

we're going to investigate.

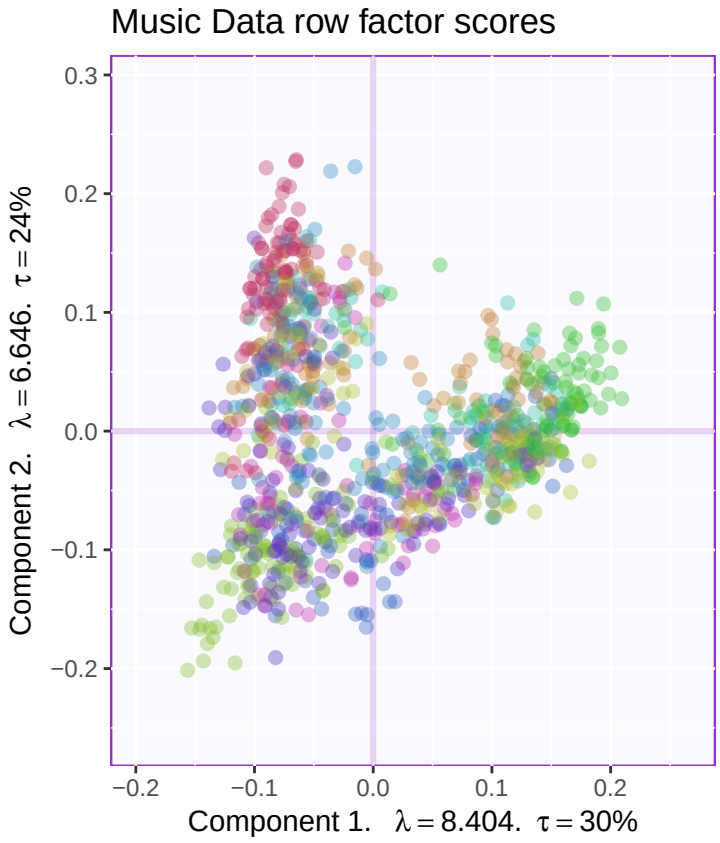
```
my.scree <- PlotScree(ev = mfpca.res$ExPosition.Data$eigs,
                      plotKaiser = TRUE,
                      p.ev = mfpca.inf.res$Inference.Data$components$p.vals)
```



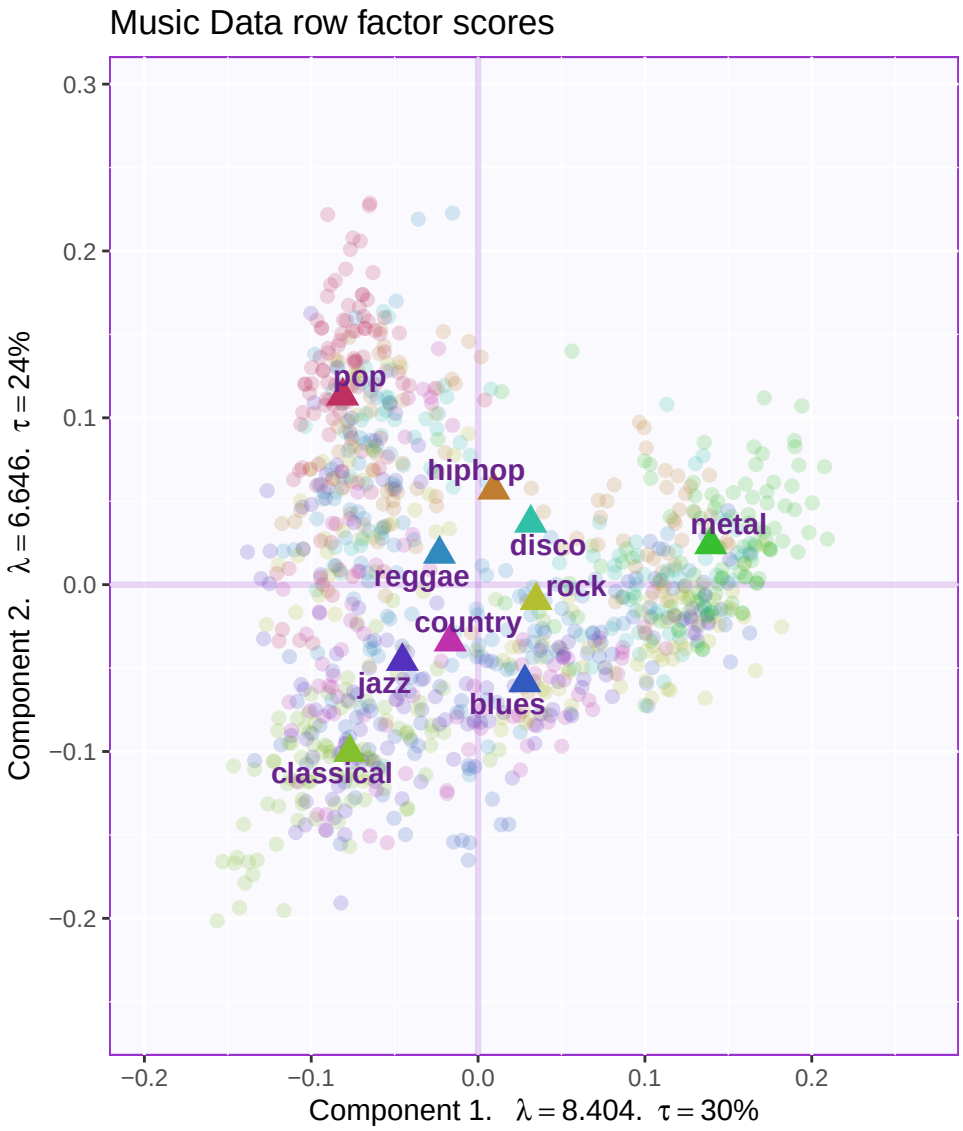
4.4.2 Row Factor scores

```
#####################################################################
# This is just sample code showing how a factor map is created.
# For more information on this, check out the actual RMD file,
# there will be more chunks dedicated to creating the factor plots.
# This section of code generates the plots for the factor scores
# for the observations and calls a basic version.
# The next chunk actually calls them.
#####################################################################

# Note that there are three sections:
# the first basically establishes all of the parameters for the factor scores plot.
# It allows you to plot of all data points in the factor space,
# using the first 2 eigenvectors as axes.

mfpca.fi.plot <- createFactorMap(mfpca.res$ExPosition.Data$fi,# factor scores
                            title = "Music Data row factor scores", # title of the plot
                            axis1 = 1, axis2 = 2, # which component for x and y axes
                            pch = 19, # the shape of the dots (google `pch`)
```

```
                              cex = 2, # the size of the dots
                              text.cex = 2.5, # the size of the text
                              col.points = mfpca.res$Plotting.Data$fi.col,
                              col.labels = mfpca.res$Plotting.Data$fi.col,
                              display.labels = FALSE,
                              alpha.points = .2
                              )
# The second creates axis labels for the plots
mfpca.fi.labels <- createxyLabels.gen(1,2,
                               lambda = mfpca.res$ExPosition.Data$eigs,
                               tau = round(mfpca.res$ExPosition.Data$t),
                               axisName = "Component "
                                )
# The third creates the actual plot using the parameters defined by the above code.
fp01.pca <- mfpca.fi.plot$zeMap + mfpca.fi.plot$zeMap_dots +
            mfpca.fi.plot$zeMap_text + mfpca.fi.labels
fp01.pca
```



Note that we've specified that the dots created in this plot should not have labels (`display.labels = FALSE`) for each observation. This is because with 1000 observations plotted closely on top of each other, it's impossible to read. Instead we make some adjustments and plot the group means on top of them.



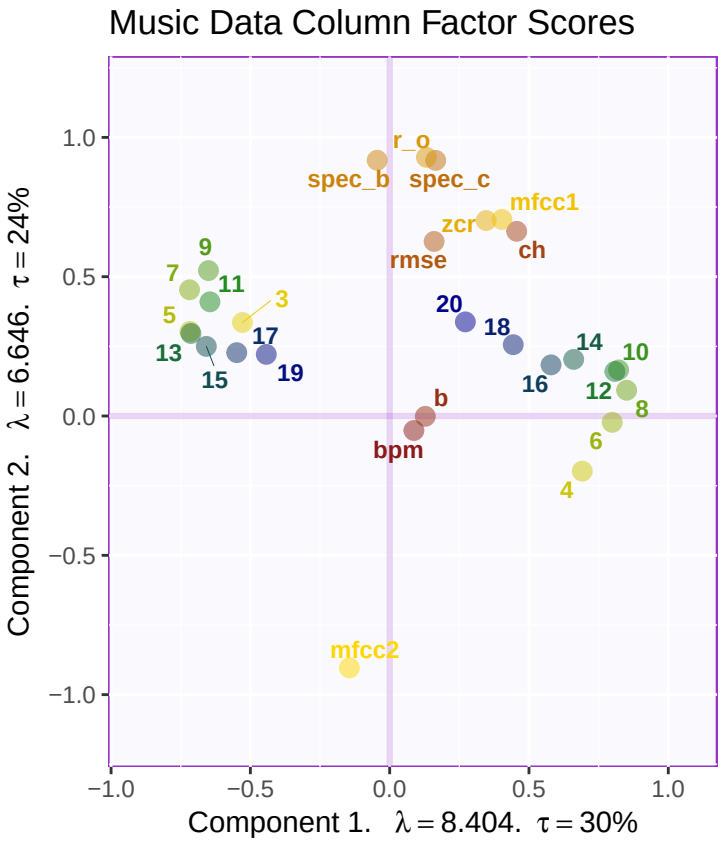
Reading this plot

As we stated above, the factor scores show us how each of the observations are represented relative to the first two principal components. Literally, how each observation 'scores' on each principal component. Observations that are closer together are more related, observations that are further apart are less related. The principal components then also show us the separation between the groups. Principal component 1 is primarily distinguishing between the genres of metal and pop and classical, while principal component two separates our classical and pop genres. This interpretation comes from how far away from the barycenter (origin) these group means are. In order to figure out what variables are driving these separations between variables, we'll have to look at the loadings plot.

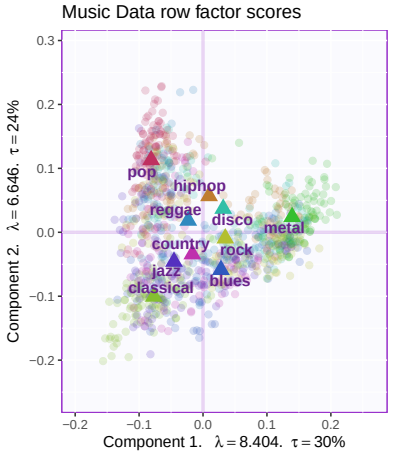
```
# This gets colors to use for the variables.
# ExPosition only gives us one color for the variables, but we need a total of 28 colors.
# colorRampPalette() inerpolates colors in between the ones that are listed
colfunc <- colorRampPalette(c("firebrick4","gold","forestgreen","darkblue"))
col4load <- colfunc(28)
```

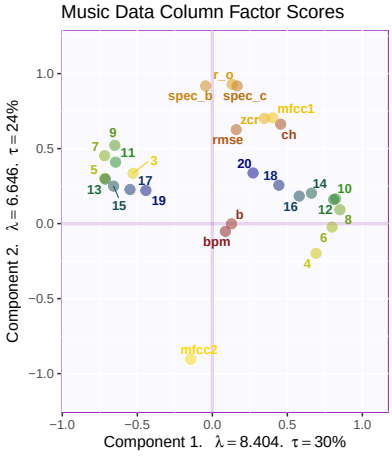
4.4.3 Column Factor Scores

We can make similar plots for the column factor scores, using `mfpca.res$ExPosition.Data$fj` instead of `$fi`:



Reading this plot:
This plot represents the factor scores for each of the columns of the analysis. Similarly to the row factor scores plot, we see how the variables score on the dimensions plotted. Likewise we see what variables drive the components. Principal Component 1 differentiates the MFCCs greater than 2 from the MFCCs greater than 1, while principal component 2 differentiates the spectral components roll of, spectral bandwidth and spectral centroid from MFCC2. Viewing the two plots side by side is very helpful in seeing how the observations and the variables are related.

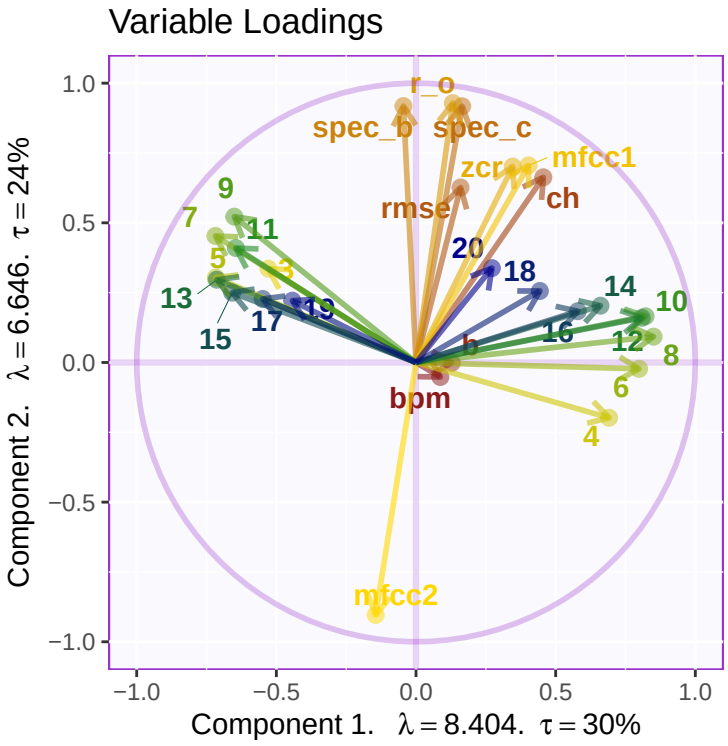




4.4.4 Loadings

We can then make a plot of the loadings of the variables, which shows us how much the variables load on the principal components, or how much variance each of the variables is contributing to the principal components:

```
# Look at the RMD for where we get the colors
# Constraints are set to [-1, 1] for both x and y dimensions because these are proportions
cor.loading <- cor(mfdata, mfpca.res$ExPosition.Data$fi)
colnames(cor.loading) <- rownames(cor.loading)
mfpca.fj.corr <- createFactorMap(cor.loading,
                                 col.points = col4load,
                                 col.labels = col4load,
                                 constraints = list(minx = -1, miny = -1,
                                                    maxx = 1, maxy = 1),
                                 title = "Variable Loadings"
                                 )
lp02.pca <- mfpca.fj.corr$zeMap +
            addArrows(cor.loading, color = col4load) +
            addCircleOfCor() + mfpca.fi.labels
lp02.pca
```



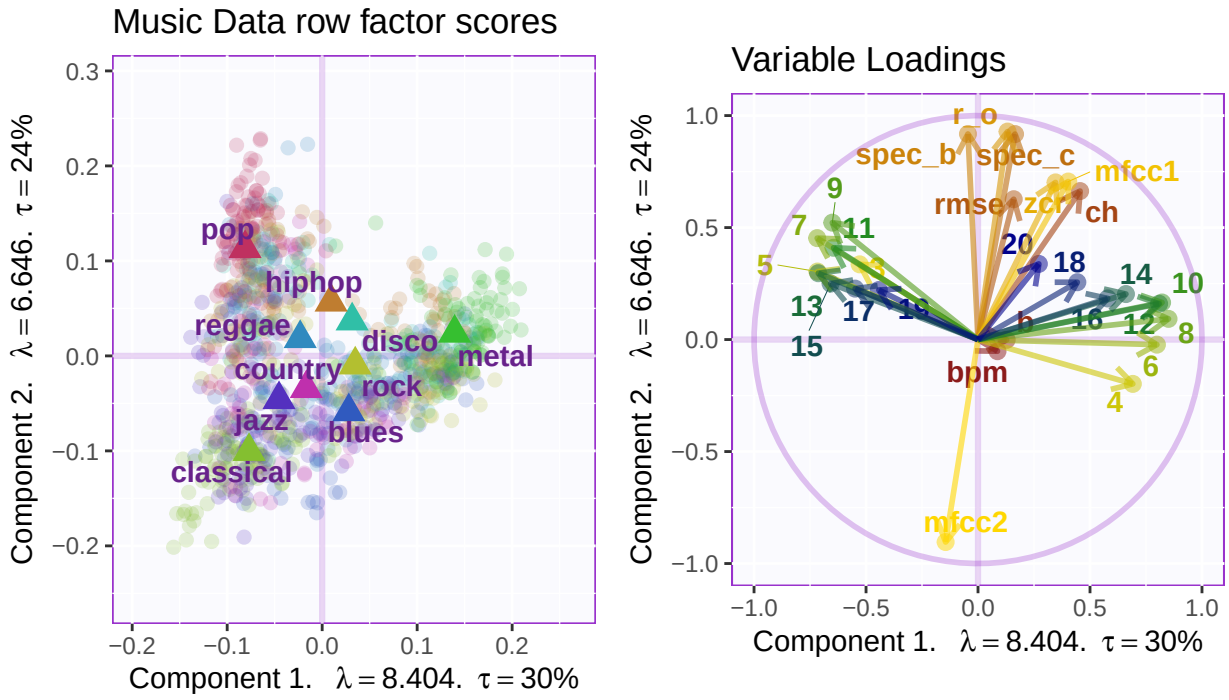
Reading this plot:

The loadings represent how each of the variables contribute to the various dimensions of the space.

- They are calculated as the correlation that each variable has with between the original data and the factor scores.
- The loadings are plotted inside of a circle of radius 1, representing the proportion of variance explained for a given variable.
- The length of an arrow (how close it is to the edge of the circle) shows you how much of that variable is accounted for in the plot shown.

- The angle between any two arrows/points show us the degree to which the two variables are related. The squared cosine of the angle is the correlation coefficient.
- Arrows that are at 0 or 180 degrees from each other represent variables that are completely correlated or completely uncorrelated.
- Arrows at 90 degrees are orthogonal, unrelated, and measure completely different things.

It's also helpful to see the loadings alongside the row factor scores. This allows an easier visual comparison between the variables and the observations or groups. When presenting your data, you should always try to present the data in a way that is most comprehensible.



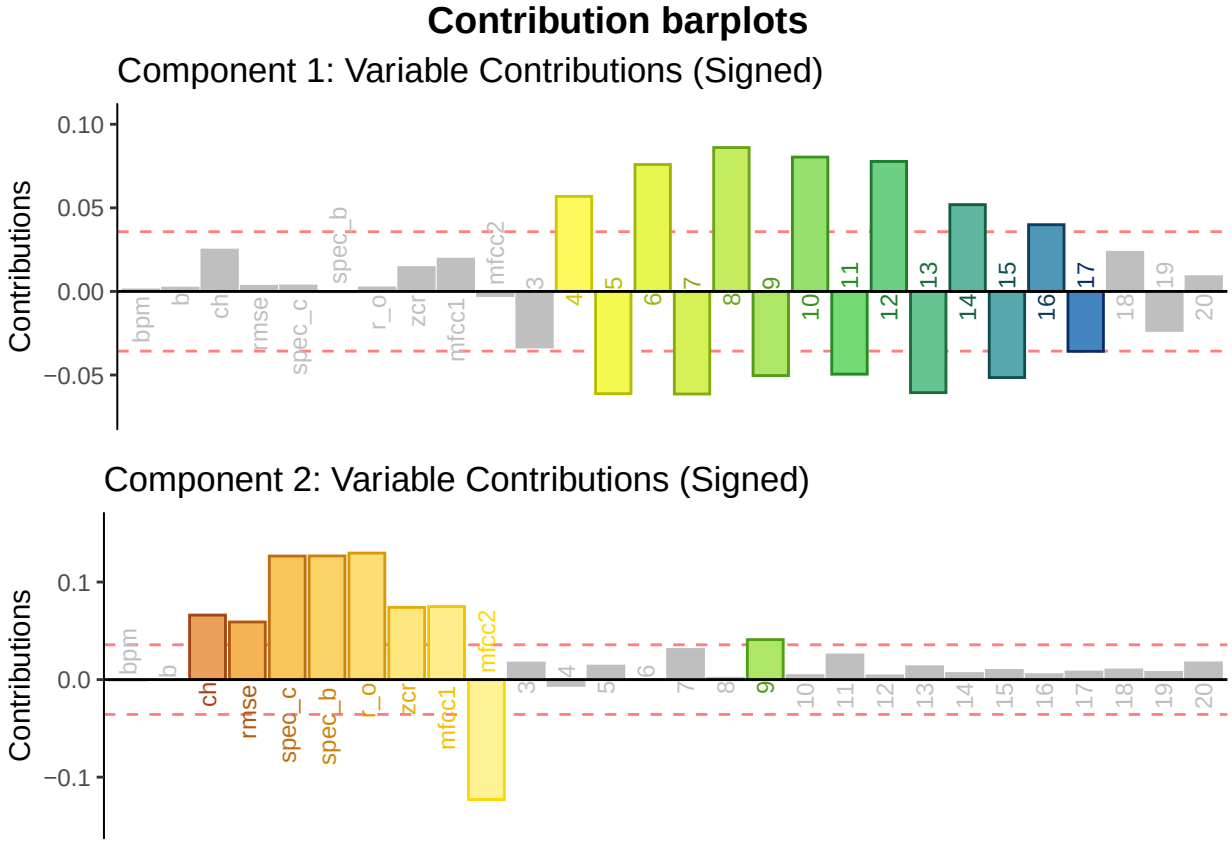
4.4.5 Contribution barplots

Reading this Plot

The contribution barplots show you how much each variable loading contributes to each principal component and identifies visually which of them are significant in this regard. The contributions plotted here are signed, which gives us both the magnitude and the direction of the contribution.

```
# This code creates the plots for the contributions of the loadings.
# The next block of code inserts them into the document.
mfpca.s.ctrJ <- mfpca.res$ExPosition.Data$cj * sign(mfpca.res$ExPosition.Data$fj)
# plot contributions for component 1
mfpca.ctrJ.1 <- PrettyBarPlot2(mfpca.s.ctrJ[,1],
                         threshold = 1 / NROW(mfpca.s.ctrJ),
                         font.size = 3,
                         color4bar = col4load, # we need hex code
                         main = 'Component 1: Variable Contributions (Signed)',
                         ylab = 'Contributions',
                         ylim = c(1.2*min(mfpca.s.ctrJ[,1]), 1.2*max(mfpca.s.ctrJ[,1]))
)
# plot contributions for component 2
```

```
mfpca.ctrJ.2 <- PrettyBarPlot2(mfpca.s.ctrJ[,2],
                           threshold = 1 / NROW(mfpca.s.ctrJ),
                           font.size = 3,
                           color4bar = col4load, # we need hex code
                           main = 'Component 2: Variable Contributions (Signed)',
                           ylab = 'Contributions',
                           ylim = c(1.2*min(mfpca.s.ctrJ[,2]), 1.2*max(mfpca.s.ctrJ[,2]))
)
mfpca.ctrs <- grid.arrange(as.grob(mfpca.ctrJ.1), as.grob(mfpca.ctrJ.2),
                         ncol=1, top = text_grob("Contribution barplots",
                                                 size = 14, face = "bold"))
```



4.5 Conclusions

- **Component 1**
  - The first principal component separates observations of Metal on one end and Pop and Classical on one end.
  - For variables, the first component separates odd and even MFCCs greater than 2. Knowing what we know about these variables, it makes sense that odd and even are separated.
  - Interpretation: Although metal seems to have different spectral components from classical and pop, there must be another variable or set of variables that are separating classical from pop on the second dimension. This may be a measure of distortion, or 'cleanliness' of signal. Rock is also trending to that side of the component. Perhaps further analyses will clarify what the first component represents.
- **Component 2**

  - The second principal component separates observations of Pop on one end and Classical on the other.
  - For variables, the second component separates MFCC 1 and the other named spectral components on one end and MFCC2 on the other.
  - Interpretation: Because we see the electronically produced genres like pop and hip hop tending towards one end of the second principal component and acoustically recorded genres like classical, blues, and jazz, we can gather that the acoustically produced music has lower extreme spectral components than electronically produced music.
- **About the Variables**: Remember that this technique is about understanding how the variables work at a deeper level. One thing that we can learn from this is that the MFCCs greater than two, odd and even, are anti-correlated with each other. This basically means that they're measuring the same thing. Likewise, since MFCC 1 and 2 are essentially orthogonal to the other MFCCs, that tells us they're measuring different things. If MFCCs 1 and 2 and the named spectral components measure the width and breadth of the audio spectrum in a file, maybe the first dimension (the other MFCCs) represent the wave forms of the files. While we might think that the first of those measurements would be more important than the second in terms of macro measurements for these files, but remember that this technique handles the data impartially, based on which variables ;provide the most inertia. In this case, because we have 20 variables (the MFCCs) that measure the wave form, and only 8 that are named and measure the shape of the spectrum, that preponderance of information is what's going to drive the first component. Luckily we have some other analyses further along in the cookbook that break the variables down a little more to tease apart those relationships.

Chapter 5

Inferences for Principal Components Analysis

5.1 Intro to Inferences

This is a recipe for Inferences for Principal Component Analysis. To understand this, you should have already read the chapter on Principal Components Analysis. Many of the topics presented here are the same, and thus are truncated to avoid redundancy. The analysis is run on a dataset of spectral decomposition of 1000 30-second samples of audio files, aimed at identifying the genre of a given audio file based on spectral components. The features were extracted using the libROSA package from python.

The inference PCA is an extension of the PCA technique. For more information on the background for the technique, check out the introduction to this recipe. The extensions specifically are some inferential statistics methods: permutation tests for the eigenvalues and bootstrapping for the means of the groups and the contributions to the factors. Permutation testing (Berry, Johnston, and Mielke 2011) breaks the connection between the observations and the variables, permuting the data into different possible arrangements. The result is a measure of how likely the observed data are to arise due to a random effect. The comparison of the observed value of the eigenvalues to the permuted eigenvlaues tells us what the p-value is for these data. Bootstrapping (Hesterberg 2011) uses resampling with replacement, keeping the observations and their data intact. It does not test the significance of the data, but the stability of the observations. Bootstrapping will by definition cluster to the barycenter or barycenters of the data. It thus gives us a measure a) of how stable the observations are, how likely they are to be generalizable to other observations, and thus whether or not we can make predictions based on the results and b) if there are any underlying deviations in the data distributions.

5.1.1 Dos and Don'ts

Do:
- Remember that bootstrapping and permutation testing fundamentally test different things.

Don't:
- Confuse the results of the bootstrapping tests and the permutation tests.

Research Questions:
The inferences techniques we're looking at here lend themselves to the specific questions of "What is real?", i.e. "What is different from chance?" and "What is important", i.e. "What is stable, reproducible, generalizable?". As such, those should be guiding principles in formulating our research questions.
- Are the results that we saw regarding variance explained by certain factors by the PCA significant?

- How consistent or stable are any of these factors?
- How would we expect other types of music to compare to the types here?
- If we were to add other genres to this analysis, how would those generalize?

5.2 Data

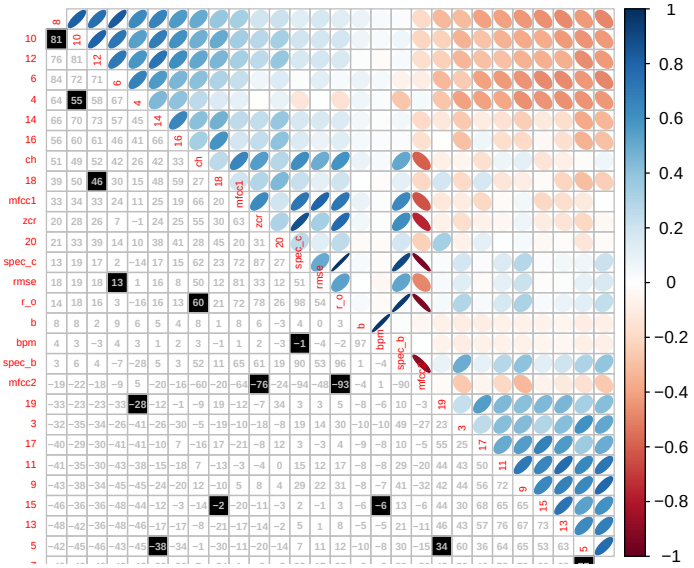
The dataset for this analysis is the same as the one used for PCA. Check there for more info. Below is a table to give you an idea of each of the variables. In the interest of saving space, only the first 3 MFCC's are shown. In the analyses to follow, the names of the MFCC's >2 have been shortened to just their numeral (i.e. MFCC3 = 3).

	b	ch	rmse	spec_c	spec_b	r_o	zcr	mfcc1	mfcc2	3
blues.00081.au	50	0.3802602	0.2482623	2116.943	1956.611	4196.108	0.1272725	-26.92978	107.33401	-46.809993
classical.00091.au	44	0.2237383	0.0447457	2192.798	1911.986	4066.467	0.1524379	-251.31345	88.72515	-41.767192
country.00013.au	63	0.3680927	0.0800457	2812.346	2842.053	6062.663	0.1295650	-126.16004	77.63675	3.165314
disco.00063.au	64	0.5478499	0.2939197	2583.278	2626.311	5855.473	0.0997726	-51.75267	70.33190	-3.919614
hiphop.00034.au	97	0.4699237	0.2279124	3191.045	2735.905	6353.995	0.1691660	5.87653	53.23913	-8.983690

5.2.1 Correlation Matrix

In order to get an idea of what the data look like overall, we run a correlation analysis `cor()` and plot it using corrplot.

```
cor.res <- cor(mfdata)
corrplot(cor.res, diag = F, type = "upper", method = "ellipse",
         order = "FPC", tl.cex = .5, tl.pos = "n") %>%
corrplot(cor.res, add = TRUE, diag = F, type = "lower", method = "number",
         addCoefasPercent = T, order = "FPC", col = "grey",
         tl.cex = .5, number.cex = .5, tl.pos = "ld")
# cor.plot.r <- recordPlot()
# If you want to save to a powerpoint later, uncomment this.
# It records whatever plot is in the current r device.
```



Reading this plot:
- This is a correlation plot showing us how the variables correlate with each other.

- There are other options for ordering the variables, using the parameter `order`. They include "FPC", which orders the variables they way they load on the first component, left to right. Explore these options for whichever makes the most sense for your data.
- The two halves of the plot display the same information.
- The top half uses color and shape to show the strength and direction of correlation.
- The bottom half uses values between -100 and 100, using the parameter `addCoefasPercent`, to show the correlation coefficient between the variables.

A couple of things to note:
- Tempo (bpm) and beats (b) are correlated with each other (they effectively measure the same thing) and basically with nothing else. This makes sense, as tempos are not unique to any given genre.
- Many of the spectral elements (chroma, rmse) show a strong positive correlation with each other and a strong negative correlation with MFCC2.
- The MFCC's seem to be measuring the same thing - odd and even MFCCs are anti-correlated, which makes sense since the actual creation of the MFCCs involves a de-correlation process, so that neighboring triangular windows capture different information.
- That being said, MFCCs 1 and 2 seem to have a fairly strong negative correlation with each other, but they are approximately orthogonal to the other MFCCs.

5.3 Analysis

The code below runs our PCA and PCA inference battery. Technically, for an inference PCA all we need is the `epPCA.inference.battery`, as the results for that function include both fixed data results and inference results (for comparison, see the outputs below).
- data is `mfdata` from above
- `center = TRUE` centers all of the data in the datset
- `scale = "SS1"` scales the data so that the sum of squares of the columns (variables) equals 1
- `DESIGN` tells the PCA/INFPCA what the groups that it should analyze by are
- `graphs = FALSE` suppresses the graph production so that the file will knit - we'll make our own graphs later
- `test.iters = 1000` tells the inference battery to run 1000 iterations of the permuation test. We'll do the bootstrapping later.

```
mf.pca <- epPCA(mfdata, center = TRUE, scale = "SS1", DESIGN = mf.genre, graphs = FALSE)
mf.pcainf <- epPCA.inference.battery(mfdata, center = TRUE, scale = "SS1",
                                     DESIGN = mf.genre, graphs = FALSE, test.iters = 1000)
```

```
## [1] "It is estimated that your iterations will take 0.17 minutes."
## [1] "R is not in interactive() mode. Resample-based tests will be conducted. Please take note o
## ============================================================================
```

```
mf.pca
```

```
## **ExPosition output data**
## *Contains the following objects:
##
##   name
## 1 "$ExPosition.Data"
## 2 "$Plotting.Data"
##   description
## 1 "All ExPosition classes output (data, factor scores, contributions, etc...)"
## 2 "All ExPosition & prettyGraphs plotting data (constraints, colors, etc...)"
```

`mf.pcainf$Fixed.Data` gives the same thing as `mf.pca`:

```
mf.pcainf$Fixed.Data
```

```
## **ExPosition output data**
## *Contains the following objects:
##
##   name
## 1 "$ExPosition.Data"
## 2 "$Plotting.Data"
##   description
## 1 "All ExPosition classes output (data, factor scores, contributions, etc...)"
## 2 "All ExPosition & prettyGraphs plotting data (constraints, colors, etc...)"
```

`mf.pcainf$Inference.Data` gives the inference results:

```
mf.pcainf$Inference.Data
```

```
## **Results for Principal Component Analysis Inference Battery**
## Permutation was performed on  28 components and bootstrap performed on  28  variables
## *The results are available in the following objects:
##
##   name
## 1 "$components"
## 2 "$fj.boots"
##   description
## 1 "p-values ($p.vals) and permutations ($eigs.perm) for each component."
## 2 "A list with bootstrap data and tests."
```

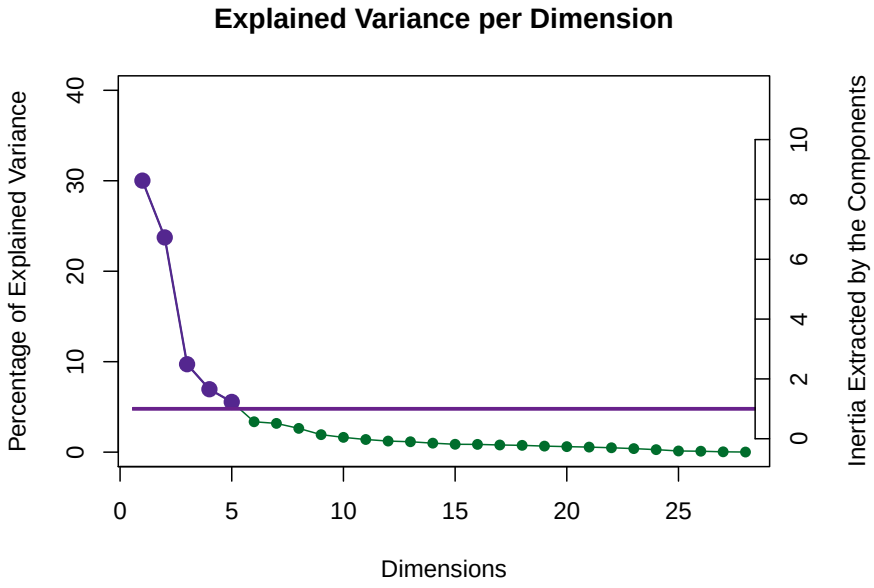
5.4 Results

5.4.1 Scree Plots

Reading this plot:
A scree plot plots eigenvalues by how much information there is in each component. Each of the dots on the scree plot identifies a dimension from the factor space in which there is variance, there are up to k - 1 dimensions from which variance can be extracted, where k is the the lower of either the number of variables in your analysis or the number of observations (i.e. min(nrow(DATA), ncol(DATA))), but your analysis of the dataset should focus on only the ones that take up the majority of the variance. A good rule of thumb for this is to look at the eigenvalues that fall above the "elbow", excluding the dimensions that fall below the noise threshold. A good way to visualize this is to connect the dots (as is done in the plot) and draw a straight line that extends from the bottom right hand corner all the way across the graph, and the point at which the dots start to land above this line is the noise threshold.
On the plot below we see two other methods of determining the significance of each dimension. The first is the Kaiser criterion, where we look at the average of the eigenvalues, plotted as a horizontal line over the plot. This isn't a rule, by any means, but it does give us an idea of what dimensions are important. The second is the result of the permutation tests. The permutation tests tell us whether each of the eigenvalues fall within the most extreme 5% of values. Again, however, this doesn't tell us necessarily what eigenvalues are important, it just shows us what values are significant. It just so happens that in this case, the Kaiser criterion, the elbow test, and the permutation tests are showing us the same dimensionality. Probably a good clue that there are 5 dimensions of data in this set.
Bottom line is, scree plots give you an idea of the true dimensionality of your data. The first eigenvalue is

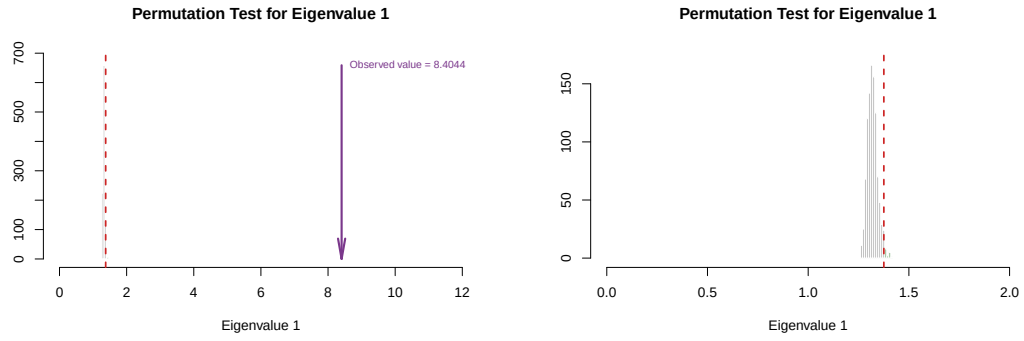
in a sense an omnibus test, it shows us whether or not there is any information in the data that isn't just noise. Beyond that, just because there are 28 dimensions in this analysis doesn't mean that there are 28 dimensions worth looking at, it's up to the researcher/observer to determine how many levels we're going to investigate.

```
my.scree <- PlotScree(ev =  mf.pcainf$Fixed.Data$ExPosition.Data$eigs,
                      p.ev = mf.pcainf$Inference.Data$components$p.vals, plotKaiser = T)
```



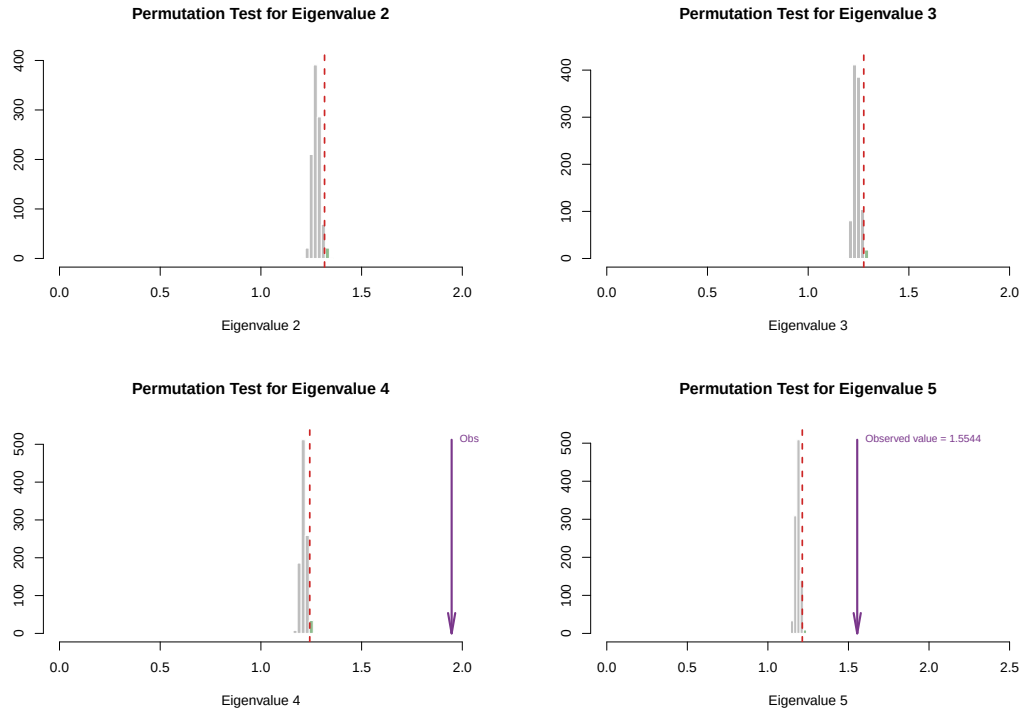
5.4.2 Testing the eigenvalues

Reading these plots: What we do here is visualize the permutations tests. The plots below show how the eigenvalues of the permutations fall into bins and stack up, and then draws a red dotted line to split the top 5% and the bottom 95%. If our eigenvalues don't fall above the line, then our specific observations and the eigenvalues we've observed are nonsignificant. If we see that our observed eigenvalue for the first dimension falls above the line, we know that we're looking at something besides noise in the data, and we can determine how many dimensions we want to investigate. As you can see, the first and second dimensions fall pretty far above the significance line. The plots below show both where the eigenvalues fall in the distribution and the overall distribution.



The plots below show us the permutation test results for our other five "significant" eigenvalues. Each of the piles shows the permutations of the test in histogram form. The observed results for the second - fifth eigenvalues are 6.646, 2.721, 1.947, and 1.554, respectively. As above, the results fall far enough below the

observed value and cluster tightly enough that it's hard to visualize both the histogram and the observed value in the same plot. The question yet remains as to whether those dimensions will be interperable. Looking above at the scree plot, we see that the 4th and 5th eigenvalues are extracting about 8% and 7% of the variance of the overall model, respectively.



5.5 Inferences

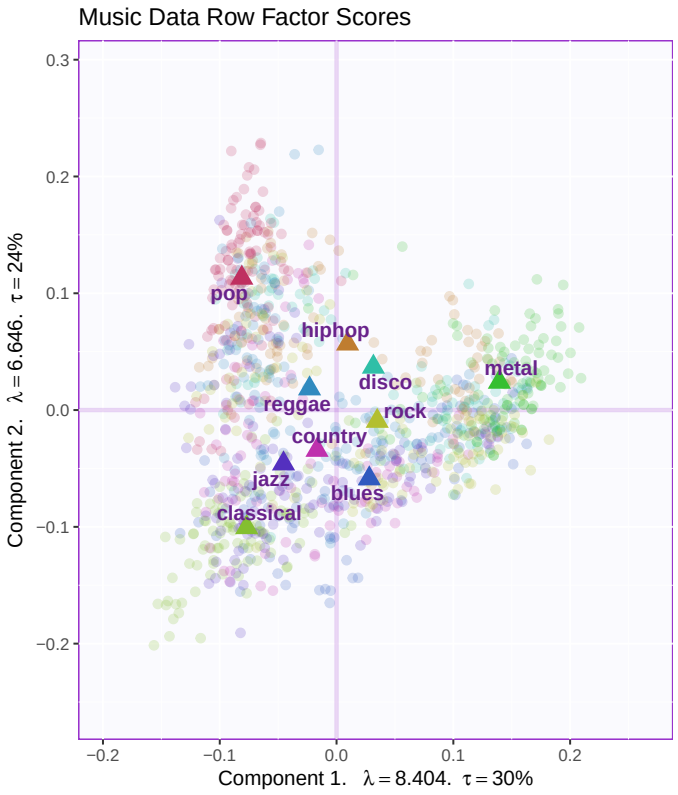
5.5.1 For the Observations

Because this section builds on the plots that we created for PCA, check there for the basic descriptions of the plots themselves. This section focuses on the inferential techniques for the plots.

We have factor scores plots for the observations (rows) similar to those we had in PCA:

```
# This is sample code showing how a factor map is created.
# Note that there are five sections:
# The first basically establishes all of the parameters for the factor scores plot.
# It allows you to plot of all data points in the factor space,
# using the first 2 eigenvectors as axes.
mfinfpca.fi.plot <- createFactorMap(mf.pcainf$Fixed.Data$ExPosition.Data$fi,# factor scores
                            title = "Music Data Row Factor Scores",# title of the plot
                            axis1 = 1, axis2 = 2, # which component for x and y axes
                            pch = 19, # the shape of the dots (google `pch`)
                            cex = 2, # the size of the dots
                            text.cex = 2.5, # the size of the text
                            col.points = mf.pcainf$Fixed.Data$Plotting.Data$fi.col,
                            col.labels = mf.pcainf$Fixed.Data$Plotting.Data$fi.col,
                            display.labels = FALSE,
                            alpha.points = .2
                            )
```

```
# The second creates axis labels for the plots
mfinfpca.fi.labels <- createxyLabels.gen(1,2,
                              lambda = mf.pcainf$Fixed.Data$ExPosition.Data$eigs,
                              tau = round(mf.pcainf$Fixed.Data$ExPosition.Data$t),
                              axisName = "Component "
                               )
# The third gets the means of the groups so we can plot them
mfinfpca.means <- getMeans(mf.pcainf$Fixed.Data$ExPosition.Data$fi, mf.genre)
# The fourth creates the factor map for the means, so we can plot them on the same plot
groupscolors <- unique(mf.pcainf$Fixed.Data$Plotting.Data$fi.col)
mfinfpca.fi.meansplot <- createFactorMap(mfinfpca.means,
                                  title = "Music Data Genre factor scores",
                                  axis1 = 1, axis2 = 2,
                                  pch = 17,
                                  cex = 4,
                                  text.cs = 2.5,
                                  col.points = groupscolors,
                                  alpha.points = 1,
                                  display.labels = TRUE
                                  )
# The fifth creates the actual plot using the parameters defined by the above code.
fp01.infpca <- mfinfpca.fi.plot$zeMap + mfinfpca.fi.meansplot$zeMap_dots +
                mfinfpca.fi.meansplot$zeMap_text + mfinfpca.fi.labels
fp01.infpca
```

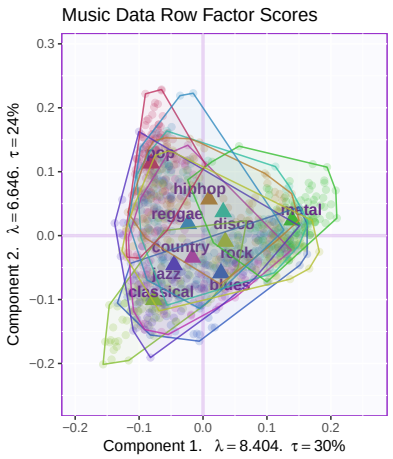


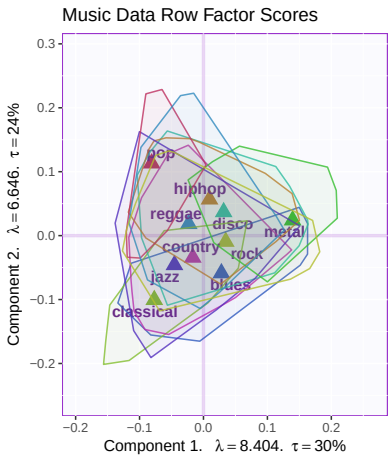
Tolerance intervals

Tolerance intervals basically outline the data by groups. On the left are the tolerance intervals plotted over

the observations by group, but that's super busy, so I plotted the graph on the right using only the means and the outline. The code below creates the tolerance intervals with a 95% coverage rate (`p.level = .95`). The specific code used to put all of the parts of the graphs below together can be viewed in the RMD file.

```
TIplot <- MakeToleranceIntervals(mf.pcainf$Fixed.Data$ExPosition.Data$fi,
                                 design = as.factor(mf.genre),
                                 # line below is needed
                                 names.of.factors =  c("Dim1","Dim2"), # needed
                                 col = groupscolors, p.level = .95,
                                 line.size = .50, line.type = 1,
                                 alpha.ellipse = .05, alpha.line    = .8,
                                 )
```





5.5.2 Bootstrapping!

Alright we're here. Let's talk about bootstrapping. Bootstrapping basically samples from the data to see if the group means are really what they say they are, or if they're actually different. So what happens here is that you take a predetermined number of samples (in this case we're taking 1000) from the data, each the same size as your original data (again, in this case, 1000), and calculate the group means each time, and then calculate confidence intervals from the 1000 samples to get an estimate of how consistent your data are.

Now, this doesn't just have to be done with the means. And to be clear, this isn't going to get you a "better" estimate of the mean. What it's going to do instead is give you an idea of how consistent and how accurate your data are, and it can help to expose underlying distribution issues in your data (i.e. if it's bimodal).

This is the actual boostrap procedure. Notice with the output below it specifies that the boot cube is a K x J x L brick of bootstrapped means. K is the number of groups, J is the number of variables, and L is the number of iterations. So in this case we have a 10 row by 28 column by 1000 page 3D matrix of means. The original group means are stored in the `$GroupMeans` table and the `$BootstrappedGroupMeans` is the means of groups from the BootCube. Because the dataset is so large, we're going to look at the first 2 pages of the bootcube.

```
## ------------------------------------------------------------------------
## Bootstrapped Means K groups, J Variables, L iterations
## ------------------------------------------------------------------------
## $BootCube                An K*J*L brick of Bootstrapped means
## $GroupMeans              The K*J table of means
## $BootstrappedGroupMeans  The K*J table of means of BootCube
```

```
## ------------------------------------------------------------------------

## , , 1
##
##                   bpm           b           ch         rmse
## blues      0.02793823 -0.05472405  0.012151113  0.011070100
## classical -0.07337849 -0.09531389 -0.039720504 -0.013750639
## country   -0.02109534 -0.03581121  0.012322503  0.015923919
## disco      0.03208195  0.03681967  0.014650681 -0.001482373
## hiphop     0.02167365  0.05656336 -0.022276511 -0.004202618
## jazz      -0.03498627 -0.05253368 -0.018512783 -0.009354446
## metal      0.14061759  0.01374145  0.005873241 -0.003729683
## pop       -0.07994886  0.11108234  0.011159045  0.002656081
## reggae    -0.02297884  0.01314266 -0.020922264 -0.022436885
## rock       0.04334710 -0.01021890  0.018526268  0.002555577
##
## , , 2
##
##                   bpm           b           ch         rmse
## blues      0.02425926 -0.061798069  0.026936930  0.011184134
## classical -0.07615768 -0.100481872 -0.040295777 -0.020605897
## country   -0.01315704 -0.036883484  0.014399106  0.014634921
## disco      0.03011844  0.032135677  0.005918271 -0.002889084
## hiphop     0.01136474  0.046907304 -0.015758096  0.013388110
## jazz      -0.03987313 -0.045543051 -0.022525849  0.003493807
## metal      0.14063590  0.021999650  0.005957486 -0.007021412
## pop       -0.07929219  0.118641319  0.013652853  0.003944758
## reggae    -0.01582602  0.015187004 -0.015633531 -0.017321526
## rock       0.02692589 -0.003488175  0.021626149  0.009428983
```

5.5.3 Plotting the Bootstrap CIs

Reading this plot The bootstrapped confidence intervals give us a measure of consistency, represented by how tight around the group mean the ellipses fall. Small ellipses tell us any number of things, and it depends on the data. It could be, as we hope, that the intervals represent very clear, consistent group means based on the bootstrapping. However, it could also just be an effect of having a large dataset. Remember that bootstrapping samples the same number of samples from your data as are in the original set, so a large dataset will see a lot of bootstrapping.

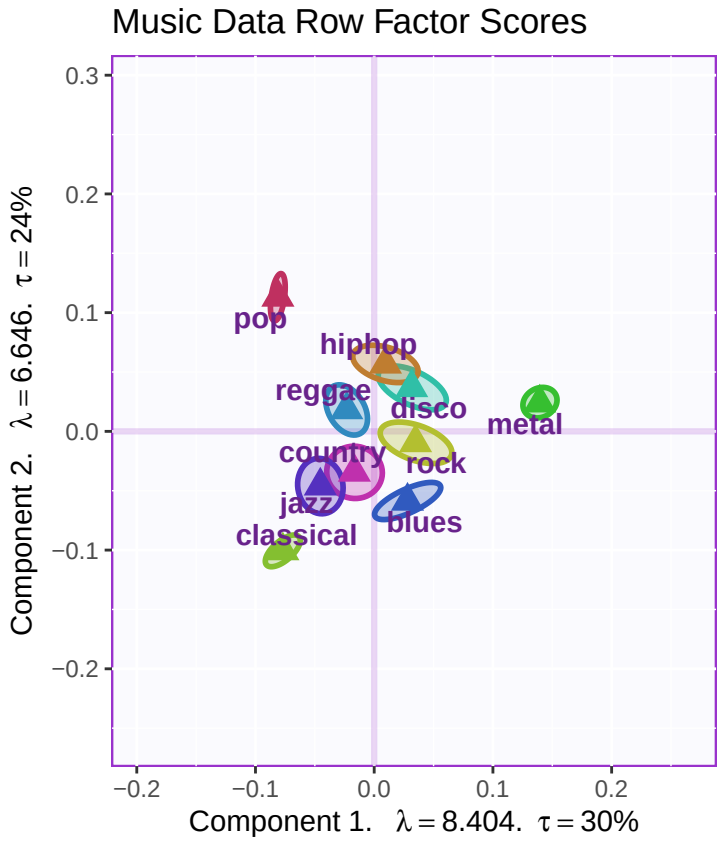
The plot below shows us the original factor scores plot with the group means and confidence intervals plotted over them. Generally, seeing that there is no overlap between group means suggests that we have clear separation between our groups. Note also that a few of the group means (pop, metal, classical) are surrounded by their factor scores dots, whereas some (reggae, hip-hop, disco) are not sitting on top of their factor scores dots. This suggests that they have a more diverse distribution of observations. However, this is difficult to read, so...

Below we once again have two graphs. The first is the same factor scores graph we saw above, with the original group means superimposed and the confidence intervals for those group means surrounding them. Below that, once again for visibility's sake, I've removed the dots and instead plotted the tolerance intervals with the group means and their Bootstrapped CIs superimposed. Note that the CIs are pretty tight around the means. This can be a for a couple of reasons. Either the means are super consistent, which is entirely possible, or because you have a large sample size. The larger the sample size, the easier it will be to get a consistent mean. Likewise, if you have a dataset that is small but is consistent, you'll get tighter confidence

intervals. Note also that the confidence intervals for these data assume similar shape to the underlying data distribution.

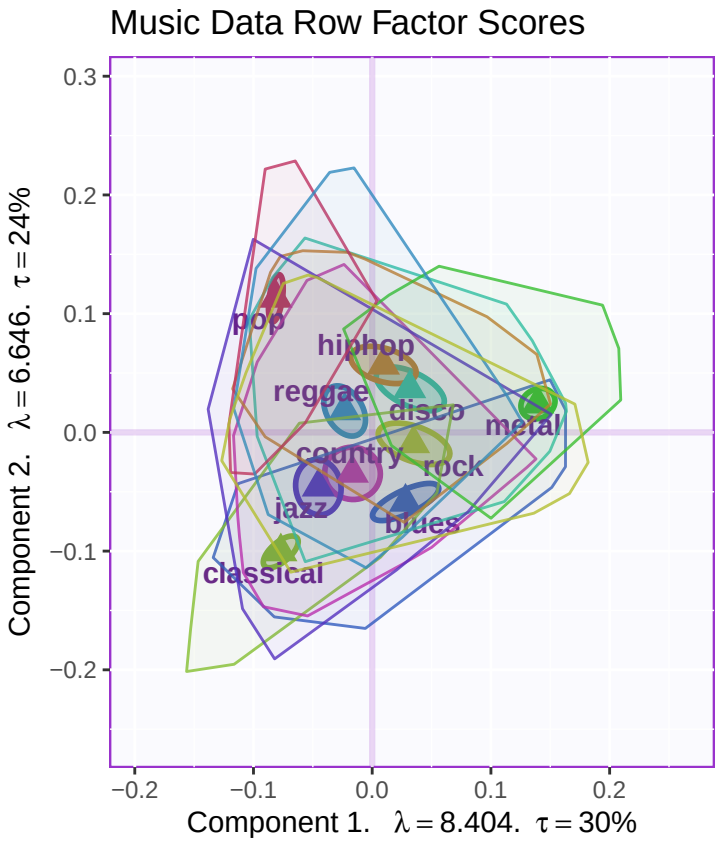
Note: When you plot your CIs, use the `p.level` parameter in `MakeCIEllipses` to adjust for multiple comparisons. A simple way to do that is to divide the standard p value of .05 by the number comparisons you need to make (in this case 45, resulting in a necessary p value of .001 (shown below)), or divide by the number of groups you have (in this case 10, resulting in a necessary p value of .005)

```
# Check other parameters you can change for this function
bootCI4mean <- MakeCIEllipses(fi.boot$BootCube[,c(1:2),], # get the first two components
                              col = unique(mf.pcainf$Fixed.Data$Plotting.Data$fi.col),
                              p.level = .999
                              )
fi.WithMeanCI <- mfinfpca.fi.plot$zeMap_background + bootCI4mean +
                 mfinfpca.fi.meansplot$zeMap_dots +
                 mfinfpca.fi.meansplot$zeMap_text + mfinfpca.fi.labels
fi.WithMeanCI
```



We can plot the group mean bootstrapped CIs on top of the tolerance intervals. It's important to note, though, that plotting the means and CIs on top of the tolerance intervals doesn't show that there is a strange dispersion for some of the groups, that we can see by plotting the means and CIs on top of the get observation factor scores.

```
fi.WithMeanCITI <- mfinfpca.fi.plot$zeMap_background +
                    bootCI4mean + mfinfpca.fi.meansplot$zeMap_dots +
                    mfinfpca.fi.meansplot$zeMap_text +
                    mfinfpca.fi.labels + TIplot
fi.WithMeanCITI
```

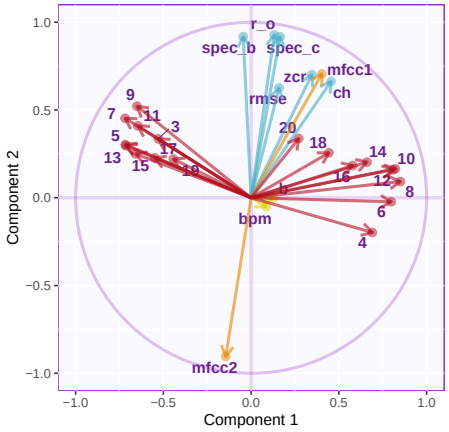


5.5.4 For the Variables

Below is the factor scores plots for the variables (columns), which uses `mfpca.res$ExPosition.Data$fj` instead of `$fi`. We can also make a plot of the loadings of the variables, which shows us how much the variables load on the principal components, or how much variance each of the variables is contributing to the principal components. Here we changed the color of the arrows to make it easier to read the names of the variables: Notice here that the variables are colored by groups. The colors come from the `wesanderson` package found here. These two plots are the same as in PCA, check there and the RMD file for more detailed descriptions.

```
mfinfpca.fj.plot <- createFactorMap(mf.pcainf$Fixed.Data$ExPosition.Data$fj ,
                            title = "Music Data Column Factor Scores",
                            axis1 = 1, axis2 = 2, # which component for x and y axes
                            pch = 19, # the shape of the dots (google `pch`)
                            cex = 3, # the size of the dots
                            text.cex = 3, # the size of the text
                            col.points = mf.pcainf$Fixed.Data$Plotting.Data$fj.col,
                            col.labels = mf.pcainf$Fixed.Data$Plotting.Data$fj.col,
                            )

fp02.infpca <- mfinfpca.fj.plot $zeMap + mfinfpca.fi.labels

cor.loading <- cor(mfdata, mf.pcainf$Fixed.Data$ExPosition.Data$fi)
colnames(cor.loading) <- rownames(cor.loading)
loading.plot <- createFactorMap(cor.loading,
                                constraints = list(minx = -1, miny = -1,
```

```
                                                maxx = 1, maxy = 1),
                                col.points = mf.pcainf$Fixed.Data$Plotting.Data$fj.col
                                )
LoadingMapWithCircles <- loading.plot$zeMap +
                         addArrows(cor.loading,
                                   color = mf.pcainf$Fixed.Data$Plotting.Data$fj.col) +
                         addCircleOfCor() +
                         xlab("Component 1") + ylab("Component 2")

fp02.infpca
LoadingMapWithCircles
```



5.5.5 Bootstrap Ratios for columns

Because we saw how to create the contribution barplots in PCA, the code for those plots isn't shown below. Instead we're focusing on the bootstrap ratios for those contributions, for the first dimension. There's more code in the RMD that creates the other plots.

Reading this plot:

The contributions tell us which of the variables load significantly on the first two principal components[1], and the bootstrap ratios tell us how consistently the the variables load on those components, or how generalizable we would expect those variables to be for a different sample. As you can see, pretty much everything but spectral bandwidth (spec_b) consistently loads on component 1, and bpm, beats (b), and MFCC 6 are consistently loading to the same degree on component 2. This reflects what we see on the loading plots, too. MFCC 6 basically lays flat along component 2.

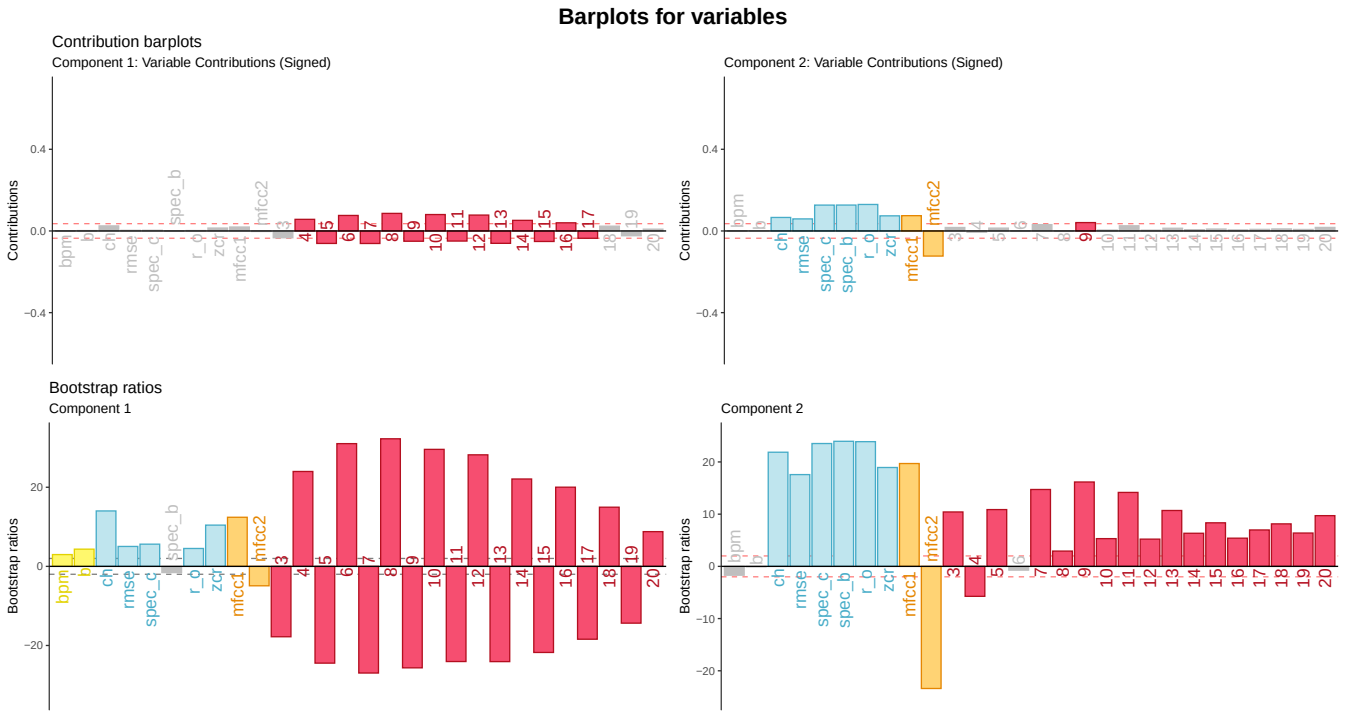
```
# Plot the bootstrap ratios for Dimension 1
BR <- mf.pcainf$Inference.Data$fj.boots$tests$boot.ratios
laDim = 1
ba001.BR1 <- PrettyBarPlot2(BR[,laDim],
                            threshold = 2,
                            font.size = 5,
                          color4bar = mf.pcainf$Fixed.Data$Plotting.Data$fj.col,
                        ylab = 'Bootstrap ratios',
                        line.col = "black"
```

[1] If you looked at PCA before this and are wondering why the contributions look less here than there, it's because the limits are different. In that analysis, the limits were set based on only the first dimension, whereas here they are set based on the entire sample. In this, they are set based on the min and max of all the contributions, not just the first dimension

```
                    ) + ggtitle("Bootstrap ratios", subtitle = paste0('Component ', laDim))
```

This arranges the plots:

```
grid.arrange(
  as.grob(ctrJ.1),
  as.grob(ctrJ.2),
  as.grob(ba001.BR1),
  as.grob(ba002.BR2),
  ncol = 2,nrow = 2,
  top = text_grob("Barplots for variables", size = 18, face = "bold"))
```



5.6 Conclusions

- **General**
  - First of all, based on the bootstrapped group means it looks like my data are both large and fairly consistent, and that for the most part, there are real significant differences between the group means.

  - The permutation testing shows us that the first few eigenvalues we extracted represent something of significance, not just noise.

- **Component 1**
  - The first principal component separates observations of Metal on one end and Pop and Classical on one end.
  - For variables, the first component separates odd and even MFCCs greater than 2. Knowing what we know about these variables, it makes sense that odd and even are separated.
  - Interpretation: Although metal seems to have different spectral components from classical and pop, there must be another variable or set of variables that are separating classical from pop on the second dimension. This may be a measure of distortion, or 'cleanliness' of signal. Rock is

also trending to that side of the component. Perhaps further analyses will clarify what the first component represents.

- **Component 2**
  - The second principal component separates observations of Pop on one end and Classical on the other.
  - For variables, the second component separates MFCC 1 and the other named spectral componentes on one end and MFCC2 on the other.
- **Interpretation:** Because we see the electronically prouduced genres like pop and hip hop tending towards one end of the second principal component and acoustically recorded genres like classical, blues, and jazz, we can gather that the acoustically produced music has lower extreme spectral components than electronically produced music.

Chapter 6

Correspondence Analysis

6.1 Intro to CA

Correspondence Analysis (CA) performs a generalized singular value decomposition on a data table to get principal components. The principal components calculated in this analysis are then used to find factor scores for both the rows and the columns. Because of the organization of the data, the decomposition allows for a visualization of both rows and columns in a single factor space. The CA is a flexible technique originally intended for use on a contingency table, and which has since been adapted for use with other types of qualitative data. The result of this technique is a factor space informed by a polygon with vertices representing the columns and points within that space representing the rows. The Correspondence Analysis is also useful if you find that your data are driven by a single extremely strong dimension. For more information on Correspondence Analysis, see Hervé Abdi and Williams (2010b) and Hervé Abdi and Béra (2018).

6.1.1 Strengths & Weaknesses

Strengths
- This technique allows you to analyze qualitative data.
- It allows you to visualize row factor scores and column factor scores in the same space.

Weaknesses
- Interpreting plots is complicated and requires an in-depth understanding of the technique

6.1.2 Dos and Don'ts

Do:
- Remember when you can and cannot compare row factor scores and column factor scores on the same plot.
- Make sure that you correctly determine the constraints for plotting your factor scores.

Don't:
- Confuse the PCA-type results in terms of principal components and factor scores that compare the rows and columns separately and the scaled 'asymmetric' plots that compare the two directly.

Research Questions:
Remember that research questions for this analysis should be guided by the fact that we can now compare variables and observations on the same space, in addition to the questions we may want to apply from PCA.
- What determines a person's separation of vowel sounds from one another?
- Are these factors physical or associational?
- Do these factors deal with phonemic information syntactic information, both, or neither?

- What guides the color association?
- How are colors and vowels related?

6.2 Data

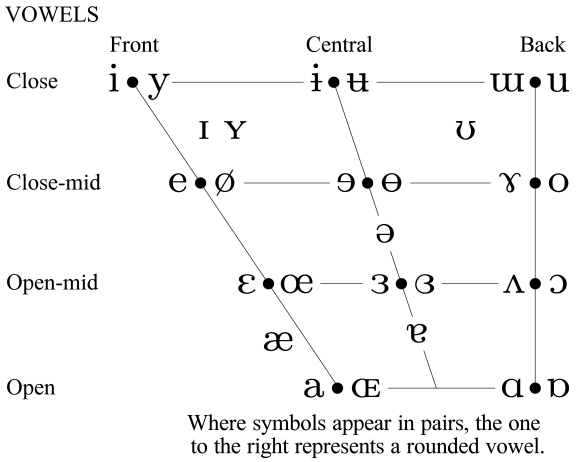
As stated above, the technique requires data similar to a contingency table. A contingency table is a table that represents the frequency distributions of observations or variables. Because of the nature of this table, the cells can be interpreted either as count data (the number of times a certain observation occurs for a certain variable) or as probability data. It's important to remember that the values represented in the cells, at the intersection of the row and the column represent count data for both the row observation and the column variable simultaneously, the designation of 'observation' and 'variable' is actually dependent on experiment design and whether we use the rows or columns as variables (by transposing the table before analysis) will have effects on our interpretation of the results of the analysis.

In this specific dataset, French participants were asked to associate vowels with colors. The participants were presented with six colors and six vowel sounds in words. The participants could associate as many vowels as they wanted with a particular color, but could assign each vowel to only one color. Below is the contingency table that shows how many participants associated each color with each vowel sound. To view these as probabilities, we simply need to look at the numbers in the table as values out of a whole. For example, participants were less likely to associate the "u" sound with yellow than they were to make any other vowel/color association.

	Yellow	Green	Orange	Blue	Red	Violet
i	46	17	2	11	42	5
y	18	48	6	12	6	6
e	17	20	13	29	4	8
a	8	7	5	17	30	6
o	18	9	19	19	21	10
u	1	2	15	14	16	16

6.2.1 Colors and Vowels

While not necessary for the Correspondence Analysis per se, these two visuals are important for understanding and interpreting the results. Top is a table showing the french names of the colors and the bottom is the International Phonetic Association's vowel quadrangle. The quadrangle represents a simplified version of where and how in the vocal apparatus a given vowel sound is produced.



	English	French
1	Yellow	Jaune
2	Green	Vert
3	Orange	Orange
4	Blue	Bleu
5	Red	Rouge
6	Violet	Violette

6.2.2 Chi²

The χ^2 test evaluates how different from a null distribution the data are.

```
chi2    <-  chisq.test(vcmat)
```

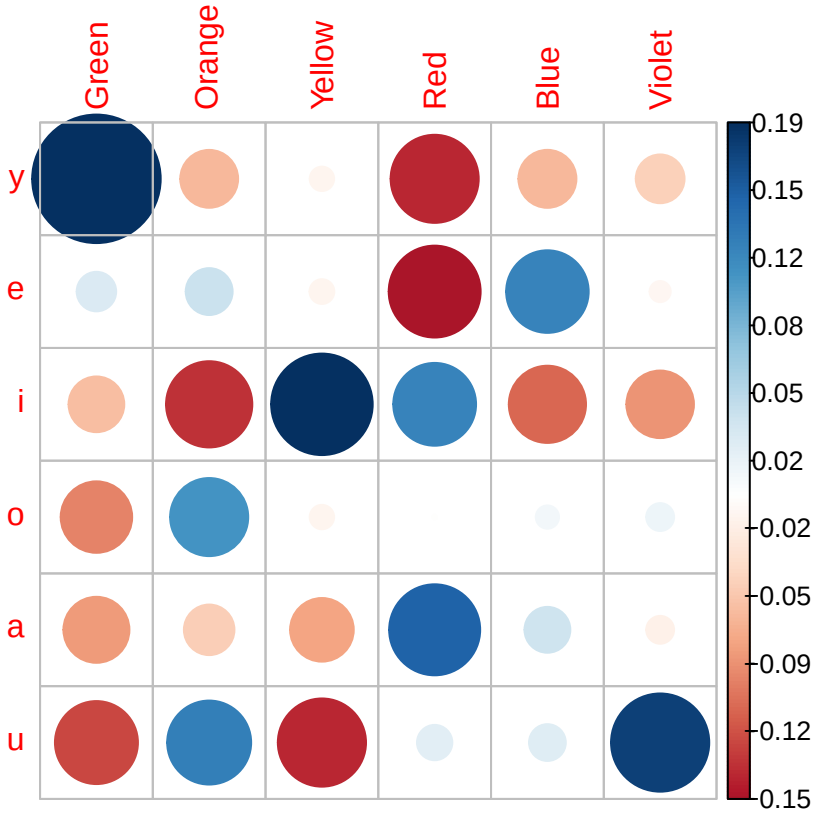
But remember, χ^2 is in counts, but CA analyzes probabilities (i.e., the profiles). So, we need to divide the χ^2 statistics by the total sum of the data. Also, the χ^2 statistic adds the chi-squares in all cells to give one number. In CA, however, we keep the *pattern* of chi-squares instead of adding all of them up. What we end up with is a table that shows us how much more or less than the average any given cell has occurred. Below are two ways of calculating the χ^2 inertia cells. The top uses the values from the results of the test above, the bottom computes them directly from the data.

```
# Components of chi2: the chi-squares for each cell
#                     before we add them up to compute the chi2
Inertia.cells <- chi2$residuals / sqrt(sum(vcmat))

# You can also compute it directly from the data
Z <- vcmat / sum(vcmat) # observed
r <- as.matrix(rowSums(Z)) # expected for each row
c <- as.matrix(colSums(Z)) # expected for each column
# Inertia.cells
test.Inertia.cells <- diag( as.vector(r^(-1/2)) ) %*%
                      (Z - r%*%t(c) ) %*% diag(as.vector(c^(-1/2)))
```

6.2.3 Correlation Table

Reading this Plot
The correlation table shows us the results of the χ^2 analysis. Each of the circles represents the strength (size and opacity) and the direction (color) of the correlation for a column (color) and row (vowel). These results should be interpreted as how much more or less likely any participant is to associate a given color with a vowel *than the average*, and vice versa. I've used `order = "FPC"` here to show the order in which they load on the first component. As you will see later, both Green and Y are furthest left, and they share a strong positive correlation. Violet and U are furthest right, and also share a strong positive correlation. Notice that Green's correlation values with the vowels gradually move down across the rows, such that it shares a moderate negative association with U.

```
vc.corr <- corrplot(Inertia.cells, is.cor = FALSE, method = "circle", order = "FPC")
```



6.3 Analysis

Below is the code for the correspondence analysis. Because of how CA works, we have the option of analyzing the data in an asymmetrical or a symmetrical plot. For both of those options, we want to run an Inference battery to test the distributions of our data.

```
library(InPosition)
vca.sym <- epCA(vcdata, symmetric = TRUE, graphs = FALSE)
vca.asym <- epCA(vcdata, symmetric = FALSE, graphs = FALSE)
vcainf.symI <- epCA.inference.battery(vcdata, symmetric = TRUE,
                                      test.iters = 1000, graphs = FALSE)
```

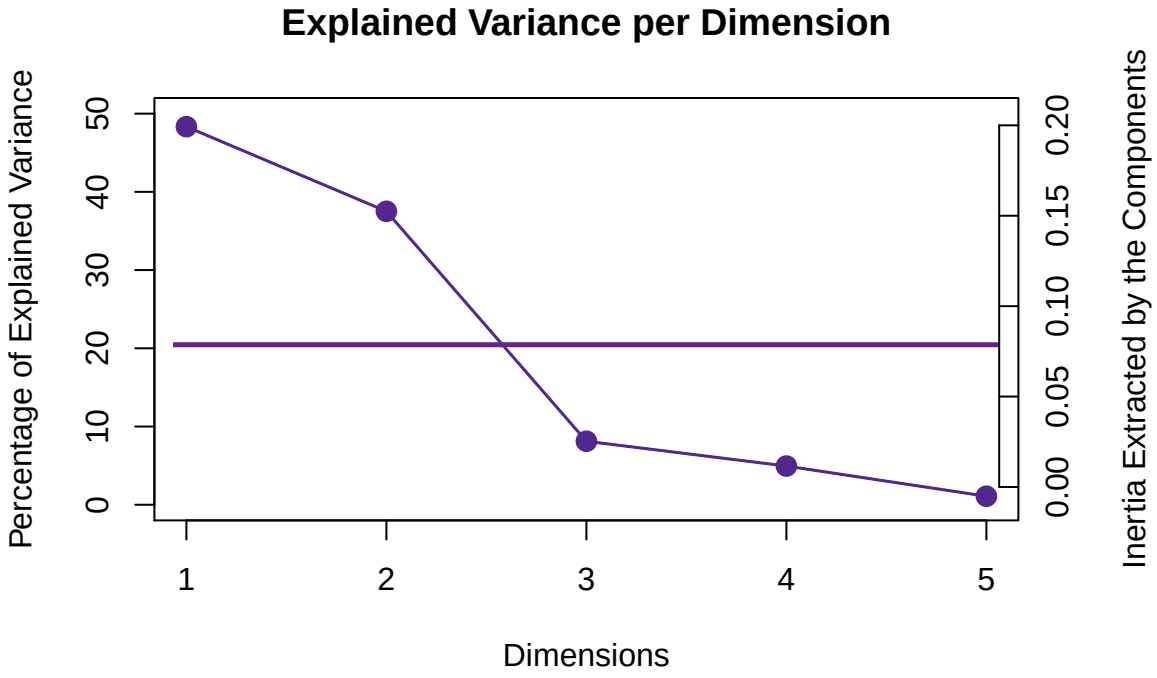
```
## [1] "It is estimated that your iterations will take 0 minutes."
## [1] "R is not in interactive() mode. Resample-based tests will be conducted. Please take note c
## ============================================================================
```

```
vcainf.symJ <- epCA.inference.battery(t(vcdata), symmetric = FALSE,
                                      test.iters = 1000, graphs = FALSE)
```

```
## [1] "It is estimated that your iterations will take 0.17 minutes."
## [1] "R is not in interactive() mode. Resample-based tests will be conducted. Please take note c
## ============================================================================
```

6.4 Results

6.4.1 Scree Plot

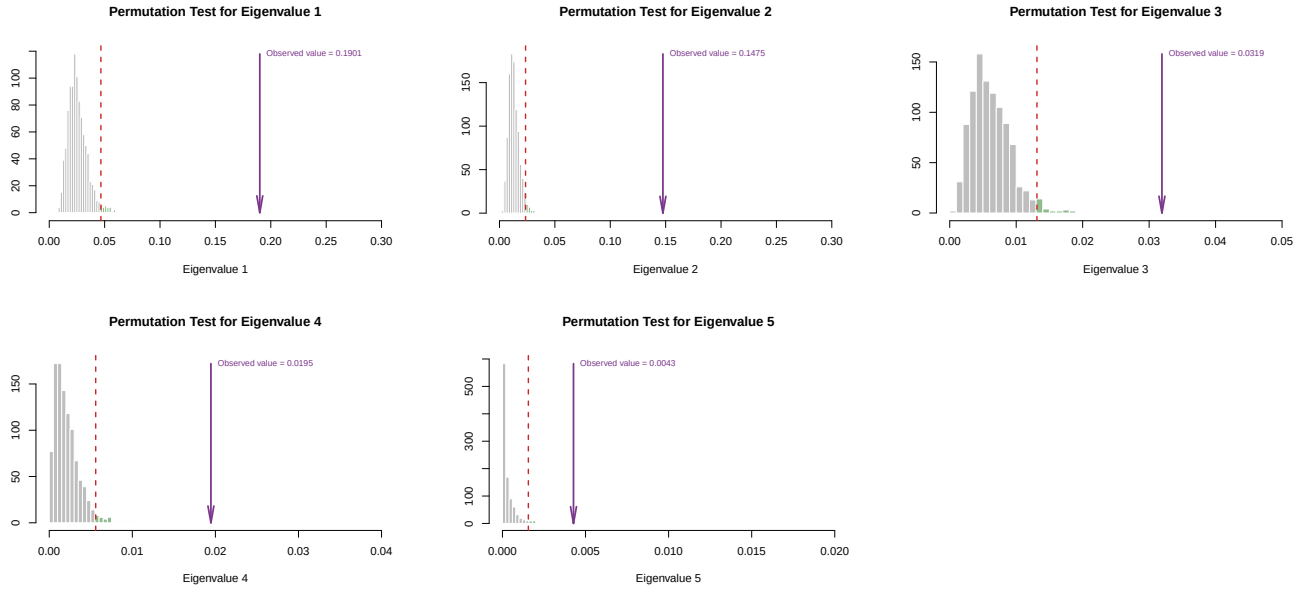


Reading this plot:
A scree plot plots eigenvalues by how much information there is in each component. Each of the dots on the scree plot identifies a dimension from the factor space in which there is variance, there are up to k - 1 dimensions from which variance can be extracted, where k is the the lower of either the number of variables in your analysis or the number of observations (i.e. min(nrow(DATA), ncol(DATA))), but your analysis of the dataset should focus on only the ones that take up the majority of the variance. A good rule of thumb for this is to look at the eigenvalues that fall above the "elbow", excluding the dimensions that fall below the noise threshold. A good way to visualize this is to connect the dots (as is done in the plot) and draw a straight line that extends from the bottom right hand corner all the way across the graph, and the point at which the dots start to land above this line is the noise threshold.
On the plot below we see two other methods of determining the significance of each dimension. The first is the Kaiser criterion, where we look at the average of the eigenvalues, plotted as a horizontal line over the plot. This isn't a rule, by any means, but it does give us an idea of what dimensions are important. The second is the result of the permutation tests. The permutation tests tell us whether each of the eigenvalues fall within the most extreme 5% of values. Again, however, this doesn't tell us necessarily what eigenvalues are important, it just shows us what values are significant. It just so happens that in this case, the Kaiser criterion, the elbow test, and the permutation tests are showing us the same dimensionality. Probably a good clue that there are 5 dimensions of data in this set.
Bottom line is, scree plots give you two things: an idea of the true dimensionality of your data and a measure of the variance explained by the components. Remember that just because the numbers are small, doesn't mean that they can't be significant. The first eigenvalue is in a sense an omnibus test, it shows us whether or not there is any information in the data that isn't just noise. Beyond that, just because there are multiple significant dimensions in this analysis doesn't mean that all of them are worth looking at. It's up to the researcher/observer to determine how many levels we're going to investigate.
The scree plot shows an interesting case that is an important lesson. The Kaiser criterion (that the

eigenvalues must be "above average" to be considered significant) is not accurate here. Testing the p values using our inference battery shows us that all five eigenvalues in this case are significant, only the top two are above the average.

6.4.1.1 Permutation Testing

As you can see below, each of the eigenvalues was tested against the permutations. All are significant in that it is unlikely that they arise by chance. Note that the scale on each of the plots below is different. They've all been scaled so it's possible to see both the permutation bins and the observed values. See the RMD for more info on creating these.



6.4.2 Factor Plots

6.4.2.1 Constraints

This next chunk shows us how we make the constraints (scale) for the X and Y axes for the factor plots. We're using `minmaxHelper` to get the constraints, and assigning them to `sym` (for symmetrical) and `asym` (for asymmetrical) respectively. For this analysis we're not doing any supplementary analyses, but if we were to do so, the code for those is supplied below. We're setting these up so that the plots display the axes correctly.

```
# Here are the factor scores you need
Fj.a <- vca.asym$ExPosition.Data$fj
Fi   <- vca.sym$ExPosition.Data$fi
Fj   <- vca.sym$ExPosition.Data$fj

# constraints -----
# first get the constraints correct
constraints.sym  <- minmaxHelper(mat1 = Fi, mat2  = Fj)
constraints.asym <- minmaxHelper(mat1 = Fi, mat2  = Fj.a)
# constraints.sup  <- minmaxHelper(mat1 = rbind(Fi, HA.sup$fii),
#                                  mat2  = rbind(Fj, punct.sup$fjj) )

colnames(Fi) <- paste("Dimension ", 1:ncol(Fi))
```

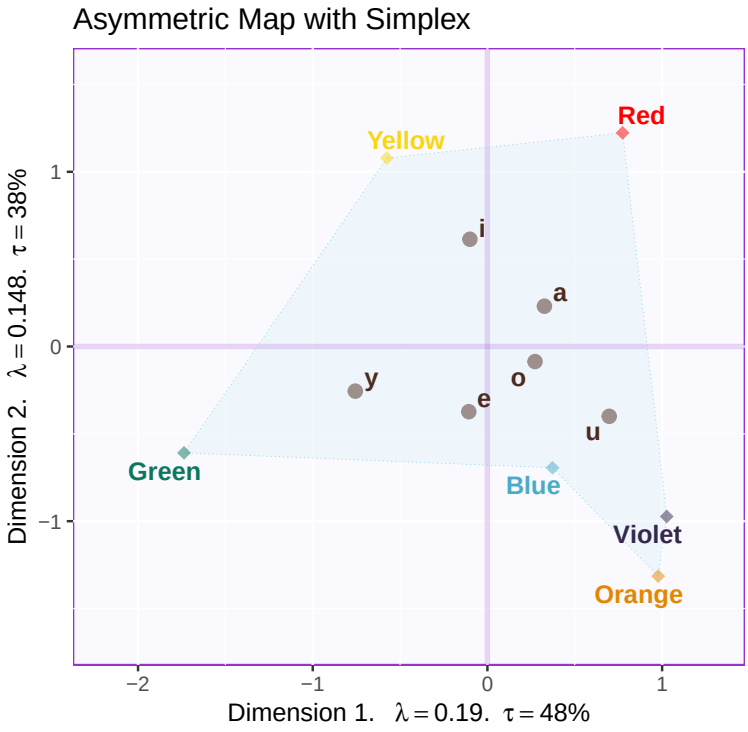
```
colnames(Fj) <- paste("Dimension ", 1:ncol(Fj))
colnames(Fj.a) <- paste("Dimension ", 1:ncol(Fj.a))
```

6.4.2.2 Asymmetric Plot

Reading this plot

Like a PCA, this still plots all of the scores on the principal components, so we can still look at the axes and try to interpret what these principal components are, and how each row and column scores on each factor. Unlike a PCA, the asymmetric plot shows us the factor scores of the rows as they relate to the column scores, with the interpretation being that the row scores show how far along the simplex (shaded) in any direction the rows observations score in that column. The simplex facilitates our understanding of the relationship between the rows and the columns in a visual/geometric sense. For more on this specific point, see Hervé Abdi and Williams (2010b), p. 8. In this case, it shows us how much more likely than average any given participant was to select certain color for a certain vowel. With this plot, we are interpreting the distances from the row points to the column points, relative to the barycenter, as scores along that axis of the simplex. We can always compare rows directly to rows and columns directly to columns, in terms of proximity and squared cosine. It's also worth noting that the functions we're using are different: `createFactorMapIJ` and `ggdrawPolygon` from the `PTCA4CATA` package. In code creating the polygon, we can specify the order in which we connect the dots. The order in which the dots are connected automatically is determined by the order in which the variables appear in the data, what we've done here is specify the number of the order in which they appear using the `order2draw` function.

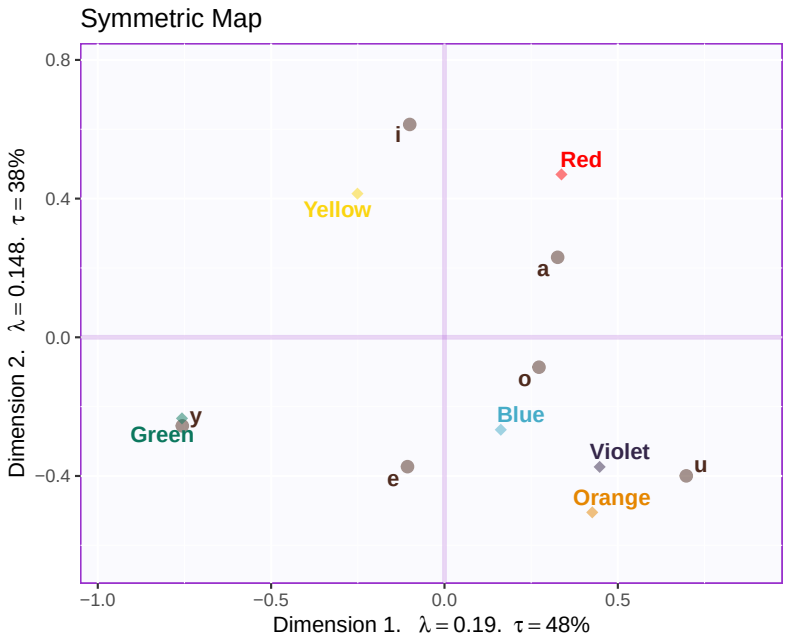
```
asymMap <- createFactorMapIJ(Fi,Fj.a,
                             col.points.i = wes_palettes$BottleRocket1[5],
                             col.points.j = columncolors,
                             alpha.points.i = 1, alpha.points.j = 1,
                             col.labels.j = columncolors,
                             col.labels.i = wes_palettes$BottleRocket1[5])
zePoly.J <- PTCA4CATA::ggdrawPolygon(Fj.a, order2draw = c(1, 2, 4, 3, 6, 5),
                                     color = wesanderson::wes_palettes$Moonrise3[1],
                                     fill = wesanderson::wes_palettes$Moonrise3[1],
                                     alpha = .1, size = .2,)
vc.labels <- createxyLabels(resCA = vca.asym)
vcmapiasym <- asymMap$baseMap + zePoly.J + asymMap$I_points + asymMap$I_labels +
              asymMap$J_points + asymMap$J_labels + vc.labels +
              ggtitle('Asymmetric Map with Simplex')
vcmapiasym
```



6.4.2.3 Symmetric Plot

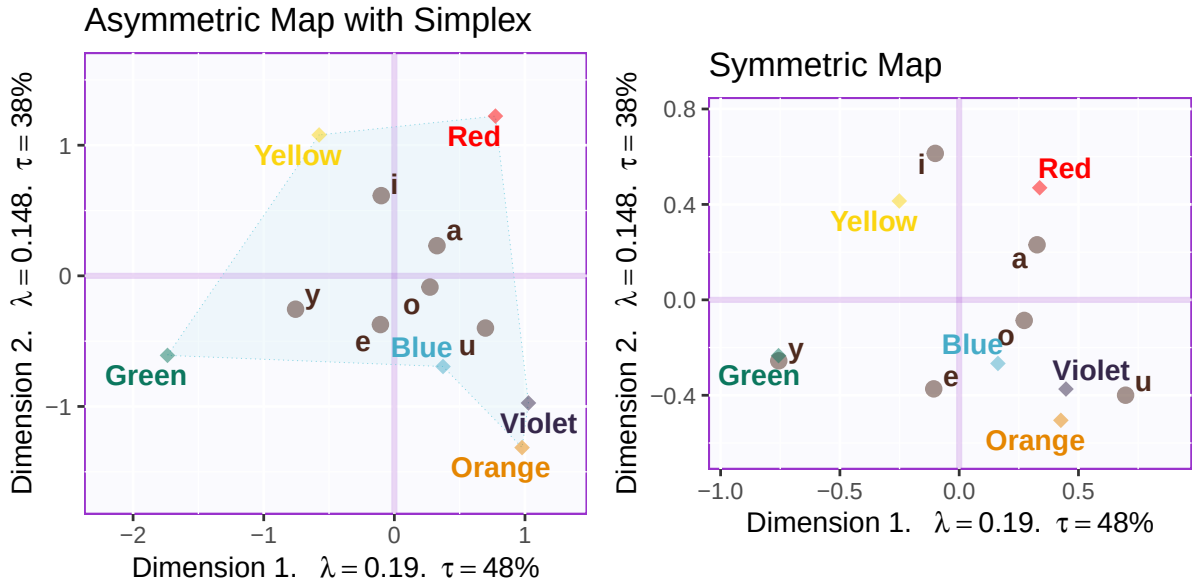
Reading this plot:
In the symmetric plot, we use the transition formula to plot both the row scores and the column scores in the same space. Now, we can interpret distance between row points as similarity and distance between column points as similarity, but we must rely on angles (squared cosines) and distance from the barycenter to interpret the similarity of the rows and columns. Green and Y are a great example of this - a strong relationship, very small angle and large distance from the origin. If you refer back to the data table at the top, you'll see that the participants were more than twice as likely to pick Green for Y than any other letter, and likewise more likely to pick Y for Green than any other color. Likewise also Yellow and U. Only one person picked U for Yellow, so they are shown to have a strong negative relationship. They are far apart and almost 180 degrees from one another around the origin. Notice also that the scale for the colors (columns) has changed. In the plot above, the vowels (values for the rows) are essentially plotted inside the column space, but here we've scaled so that both sets of scores are in the same space.

```
symMap <- createFactorMapIJ(Fi, Fj,
                            col.points.i = wes_palettes$BottleRocket1[5],
                            col.points.j = columncolors,
                            alpha.points.i = 1, alpha.points.j = 1,
                            col.labels.j = columncolors,
                            col.labels.i = wes_palettes$BottleRocket1[5])
map.IJ.sym <- symMap$baseMap +
  symMap$I_labels + symMap$I_points +
  symMap$J_labels + symMap$J_points +
  ggtitle('Symmetric Map') +
  vc.labels

map.IJ.sym
```



Side By Side

Below are the symmetric and asymmetric plots side by side, so we can compare and contrast the two. This highlights how different types of data can be better or worse suited for different kinds of plots. Because we're looking at how participants assigned vowels to colors, it makes the most sense to interpret this data in the asymmetric map. If the data were different, maybe more stereotypical count data, it might make more sense to plot the rows and columns in the symmetric map so that they can be compared more directly.



6.4.3 Contributions

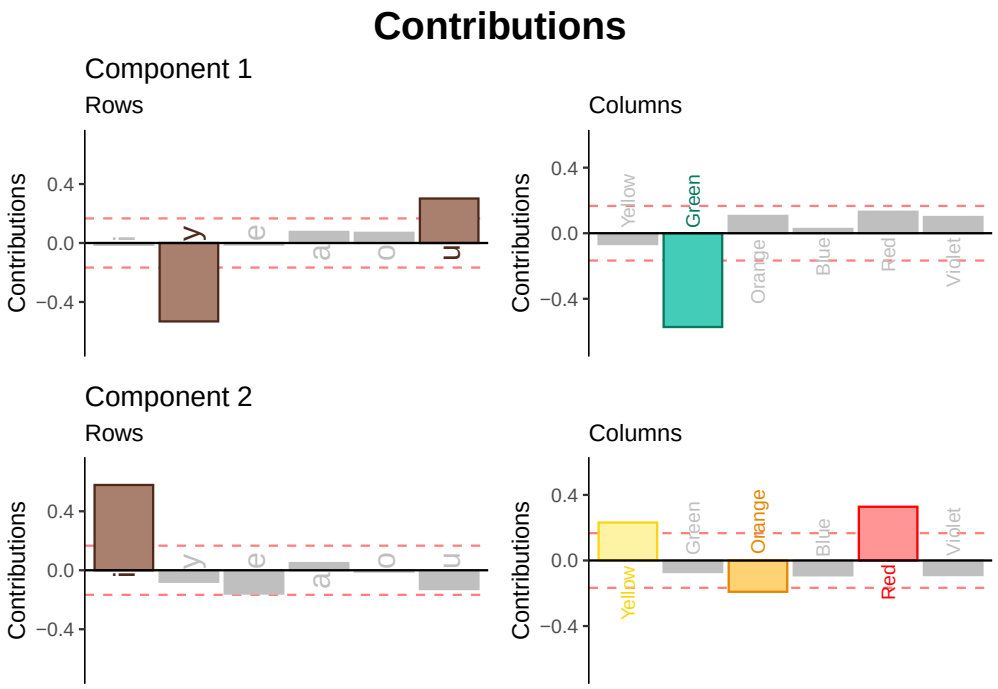
The section below shows us which of the rows and columns are contributing significantly to the principal components. Where as in PCA, we plotted only the contributions of the columns, in CA, because of the nature of the calculations and the original dataset, we need to calculate the contributions of both the rows and the columns. Note that because of the factor plots we have above, there's no need to plot the circle of loadings for the variables as we do in other analyses. We only need to do the contribution barplots.

Reading this Plot

The contribution barplots show you how much each row and column loading contributes to each principal component and identifies visually which of them are significant in this regard. The contributions plotted here are signed, which gives us both the magnitude and the direction of the contribution.

This next section arranges the four plots we created in the last chunk in a simple, easy to read grid. As we can see, Y and U seem to be the rows driving component 1, which gives a good clue of how to interpret this, especially in reference to the IPA vowel chart above, and Green is the column driving the first component. Component two is carried primarily by I loading positively on it where as the columns see significant positive contributions from Yellow and Red and significant negative contributions from Orange.

```
#Get signed components
signed.ctrI <- vca.sym$ExPosition.Data$ci * sign(vca.sym$ExPosition.Data$fi)
signed.ctrJ <- vca.sym$ExPosition.Data$cj * sign(vca.sym$ExPosition.Data$fj)
#Sample contributions code
ctrI.1 <- PrettyBarPlot2(signed.ctrI[,1],
                         threshold = 1 / NROW(signed.ctrI),
                         font.size = 5,
                         color4bar = sixbrowns , # we need hex code
                         ylab = 'Contributions',
                         ylim = c(1.2*min(signed.ctrI), 1.2*max(signed.ctrI))
) + ggtitle("Component 1", subtitle = 'Rows')
```



6.4.4 Bootstrapping

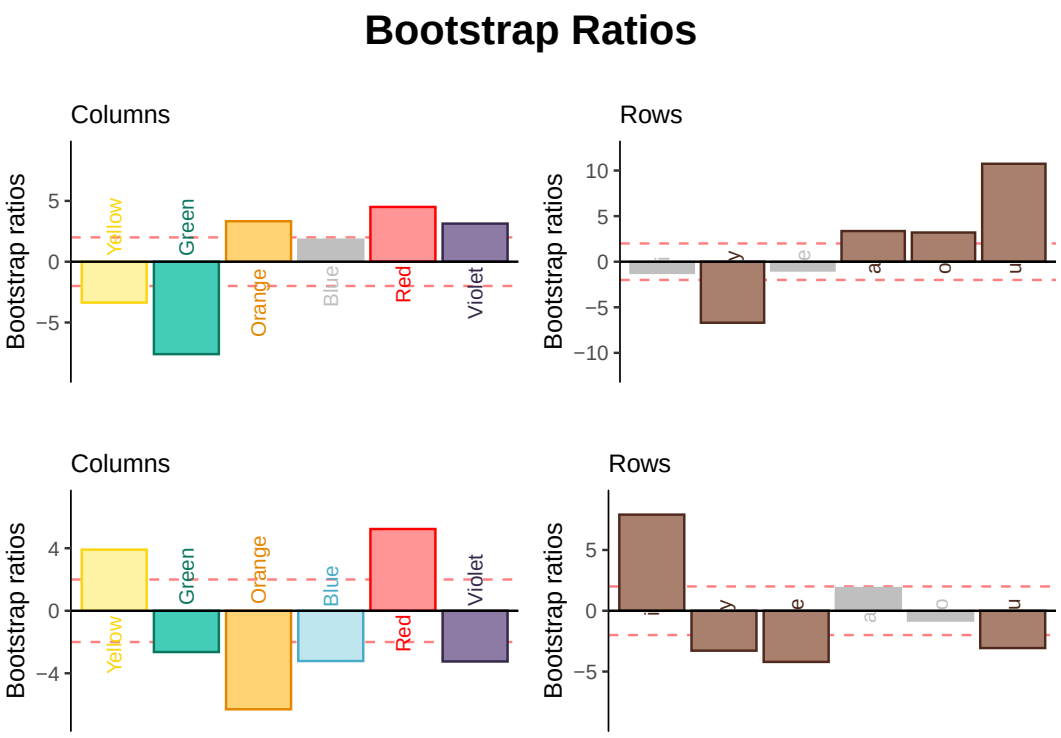
There's more on boostrapping in the Inference PCA cookbook page, but bottom line is, with this we're looking to see what the underlying distribution of the data are, and how consistent or generalizable they are; whether or not we can make inferences based on the data. For deeper discussion on bootstrapping, see Hesterberg (2011).

This is the code that calls our bootstrapping data from the inference battery we ran above and plots those data on a similar plot as we see above for the contributions.

```
#Get Bootstrap Ratios
BR.I <- vcainf.symI$Inference.Data$fj.boots$tests$boot.ratios
BR.J <- vcainf.symJ$Inference.Data$fj.boots$tests$boot.ratios
#Sample Bootstrap Ratio Plot code
laDim = 1
ba001.BR1.I <- PrettyBarPlot2(BR.I[,laDim],
                          threshold = 2,
                          font.size = 3,
                       color4bar = columncolors, # we need hex code
                      ylab = 'Bootstrap ratios'
                      #ylim = c(1.2*min(BR[,laDim]), 1.2*max(BR[,laDim]))
)  + ggtitle("", subtitle = 'Columns')
```

6.4.4.1 Bootstrap Ratios Plot

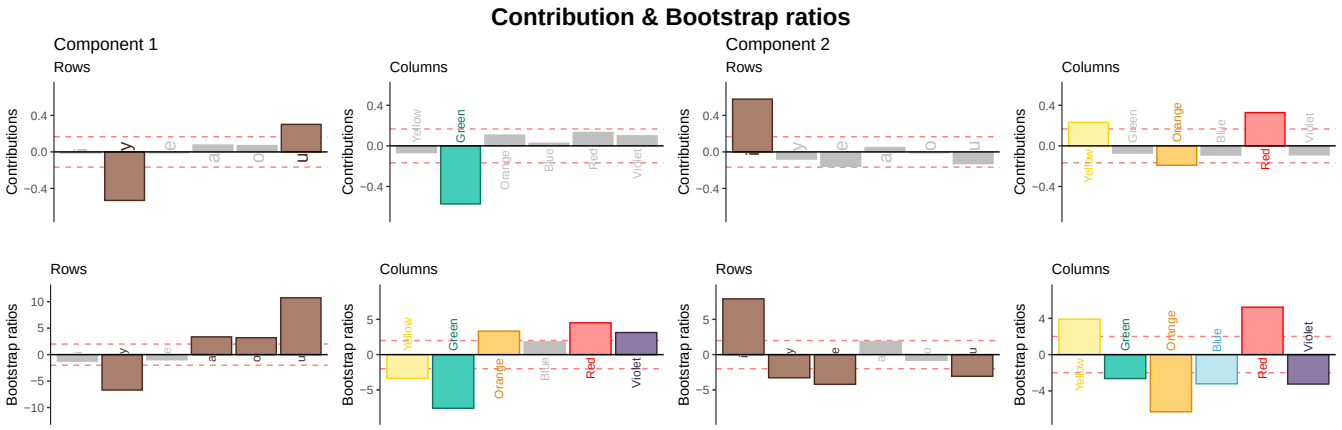
Reading this plot: The bootstrap ratios are seen below. All of the columns but Blue load consistently on the first component, likewise all of the columns load consistently on the second component. Similarly, all of the rows but I and E load consistently on the first component, and all of the rows but A and O load consistently on the second component.



6.4.4.2 All Contributions & Boostrap Ratios

Reading this Plot

The code below plots all eight graphs together for visibility and interpretation. Remember when reading bootstrap ratios and contribution plots that they present different information, even though they look very similar. The contributions represent how much variance each row or column contributes to the principal component. The bootstrap ratios are calculated by computing the mean of the bootstraps divided by the standard deviation (BR = M/S). Essentially, they tell us whether or not the the groups are consistent in how the variable or observation scores. Observations or variables for which the bootstrap ratio is bigger than two are considered stable, and therefore generalizable.



6.5 Conclusions

- **Component 1**
  - It looks like the first component, with regard to the rows/vowels, differentiates along the dimension of vowel placement, whether the vowel is placed further backward or forward in the oral mechanism. Because this is fundamental to the creation of the formants of the vowel, we are in some sense seeing that people are using the timbre of the vowel to differentiate between how they associate vowels with colors.

  - The interpretation of the column factor scores for the first component is a little bit more abstract. There could be a number of things that I am not aware of with this dataset, for example, cultural associations with colors in French culture. However, it does look like there's something like the influence of the complexity of the color word on the first component.
- **Component 2**
  - The rows seem to be lining up with Rhyme or sound association. More abstract than the first component for the vowels, but still interesting.

  - The columns seem to be driving color similarity on the second component, with one exception. Because Orange is almost completely anti-correlated with Yellow, it looks like people are likely to pick *either* yellow or orange for any vowel, but because of experimental design, they can't choose both.
- **Interpretation**
  - The two dimensions seem to have to do with timbral aspects of the sounds of the colors and their associated sounds and the semantic associations of colors. I also find it incredibly interesting that the placement of the vowels on the principal component axes that we've created is reflective of that of the placement map.

- **Symmetric vs. Asymmetric plot**
  - There are important differences between the two factor plots we've put together above. Each compares the data in different ways, and each is suited to different types of data and analyses. As is stated above, because we're looking at how participants assigned vowels to colors, and there is a sense that the rows do in fact depend on the column space, it makes the most sense to interpret this data in the asymmetric map.
  - If the data were different, maybe more stereotypical count data, it might make more sense to plot the rows and columns in the symmetric map so that they can be compared more directly. There's a great example of this in Hervé Abdi and Williams (2010b).

Chapter 7

Multiple Correspondence Analysis

7.1 Intro to MCA

MCA is effectively an extension of what we saw in CA, and allows us to decompose some of the relationships between categorical or nominal variables, or binned quantitative variables. Like the CA, we'll need something similar to a contingency table. In the analysis below, R treats each level of our variables as a single variable. Obviously with 1000 possible values for each variable, this would make our dataset and its analysis prohibitively large (possibly as large as 1000 x 28000). Instead, we'll bin the variables and run the MCA on an indicator matrix. An indicator matrix is a complete disjunctly coded matrix with observations being represented by one and only one '1' for any level of a variable. There's more on that later, in section titled Binning and the Complete Disjunct Table. While the overall output with regard to factor scores for the variables, look similar to CA (and therefore PCA), it's necessary to make some adjustments in our interpretation. For more on Multiple Correspondence, see: Hervé Abdi (2018). For more on Correspondence Analysis, see Hervé Abdi and Williams (2010b) and Hervé Abdi and Béra (2018).

7.1.1 Strengths & Weaknesses

Strengths
- Great for breaking down variables into bins, so if you have data or factor plots from a PCA that are non-linear, you can see how the various levels of the variables load onto the factor space.
- Pre-processing (histograms, etc.) force you to take a close look at your data.
- It's great to be able to see which variables are loading on which dimensions, which ones are not, and why.

Weaknesses
- The histograms are helpful, but they take some time to interpret individually.
- The visually complicated plots can be difficult to pull apart for interpretation.

7.1.2 Dos and Don'ts

Do:
- Once you've got the variables and their loadings, check out which variables load on which dimension. If you have to, pull sets of variables out and plot them individually. See which variables are measuring similar things, and which are not.
- Make sure that if you apply a correction (Greenacre or Benzécri), you apply it to both the regular MCA and the inferences for the MCA. If you don't, it won't spit an error, but you won't have the correct eigenvalues or percentage of variance extracted.

Don't:
- Get bogged down in information overload when interpreting the variables. There are number of tools that

allow us to interpret these, so use the histograms for initial overview, look for things that stand out, but come back once the disjunct factor maps are plotted to see if there's more information that can be gleaned from them.

Research Questions
Questions for this analysis should be guided by the fact that we are are looking at levels of variables relative to our observations.
- How do different levels of variables combine to create the group means?
- How do different levels of variables separate the group means along the two principal axes?
- What combinations of variables and their levels combine to create a typical observation from a certain group?

7.2 Data

We're using the music features dataset for this analysis. As stated above, we'll need an indicator matrix to run the MCA on, which means we'll need to do some editing of our data to make it work. Below is our original data. The rows show observations of one audio file each from each of six of the ten genres used in this data set. The columns show 10 of the 28 variables. For more on what each of these are, check out the explanations in the previous pages of the cookbook.

7.2.1 Data Table

	bpm	b	ch	rmse	spec_c	spec_b	r_o	zcr	mfcc1	mfcc2
blues.00081.au	103.35938	50	0.3802602	0.2482623	2116.943	1956.611	4196.108	0.1272725	-26.92978	107.33401
classical.00091.au	92.28516	44	0.2237383	0.0447457	2192.798	1911.986	4066.467	0.1524379	-251.31345	88.72515
country.00013.au	129.19922	63	0.3680927	0.0800457	2812.346	2842.053	6062.663	0.1295650	-126.16004	77.63675
disco.00063.au	129.19922	64	0.5478499	0.2939197	2583.278	2626.311	5855.473	0.0997726	-51.75267	70.33190
hiphop.00034.au	198.76803	97	0.4699237	0.2279124	3191.045	2735.905	6353.995	0.1691660	5.87653	53.23913
jazz.00004.au	64.59961	28	0.1717823	0.1087860	1039.623	1422.303	1838.214	0.0477688	-270.26482	137.57173

7.2.2 Data Preparation

We have a number of steps to prepare before we can actually run the analysis. Because an MCA evaluates a table of disjuctively coded data, we need to take a couple of steps to make sure the data is prepared to run the MCA. We do the following:

1. Evaluation
2. Binning
3. Spearman Rank testing
4. Nominalizing

At the end of this, you should end up with a dataset with the same number of rows, but more columns, because of the binning and nominalizing. If you're lucky enough to be able to use the same number of bins for each variable, it'll be the number of bins times the number of variables. For this dataset, that would be 4 (the default for the function we use below) times 28, or 112 columns.

7.2.2.1 Evaluation

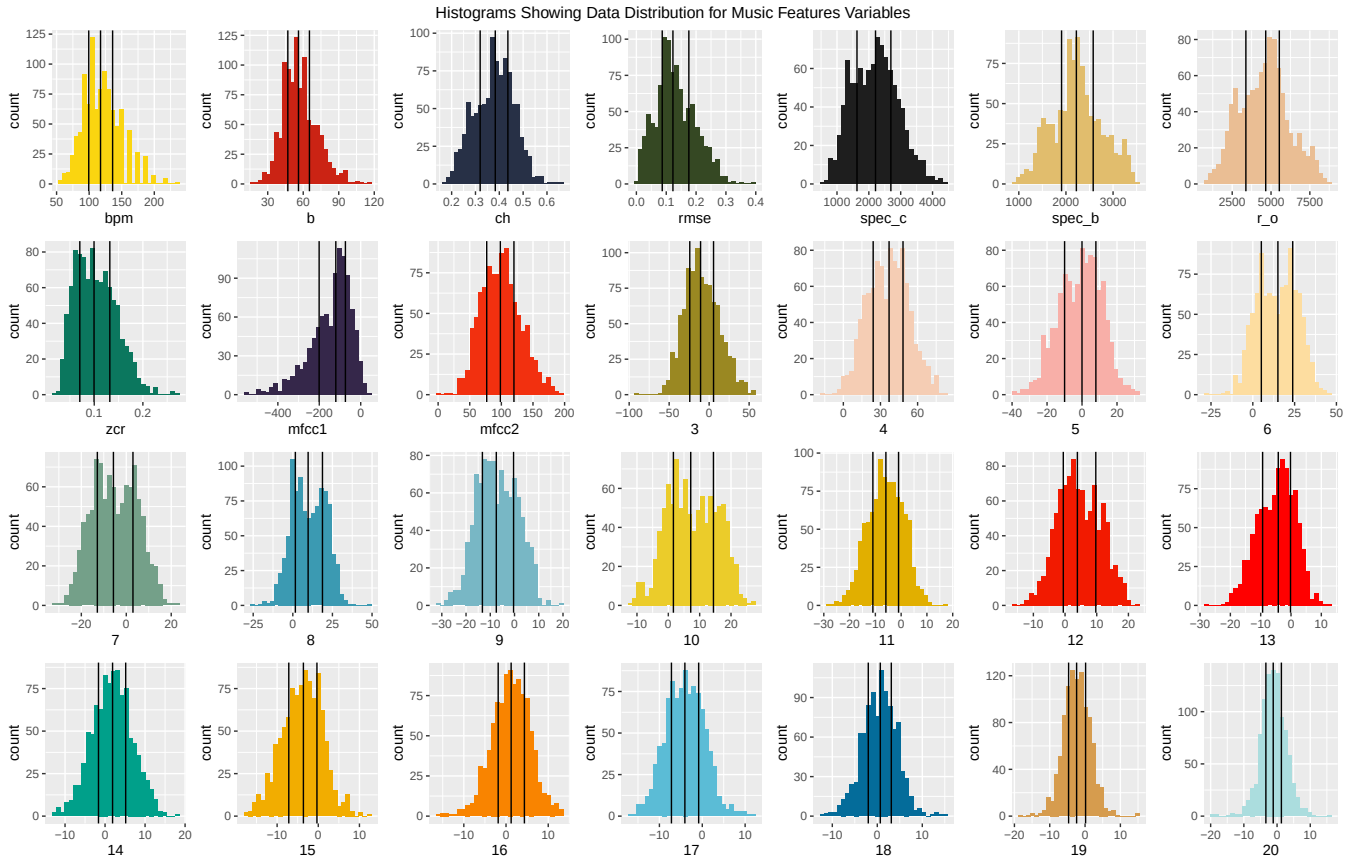
First thing we do before we bin is check out our data a little more closely, looking at range, mins and maxes, and distribution of our data. We're looking for anything out of the ordinary that would make us want to bin our data in a special way. One effective way we can visualize our data is to check out histograms of each of the variables.

`summary()` below shows us a summary of the data, and `str()` shows us the structure. The histograms below show us how each of the variables is distributed. The code we use to create each histogram is `h1 <- ggplot(data = mfdata, aes(x = bpm)) + geom_histogram(bins = 30) + geom_vline(xintercept = quantile(mfdata$bpm)[2:4])`, where mfdata is our data and bpm is the variable we're drawing a histogram for. The majority of the variables look like they have an approximately normal distribution, but you should always check to see if any look like they might have a bimodal distribution or some kurtosis or skewness.If you have a variable that has a different kind of distribution, like a Poisson distribution or a lomax distribution, you may need to do some extra pre-processing before binning or analysis. In the example we have here, the zcr variable looks slightly skewed towards the left, which suggests that the majority of the audio files contain large amounts of percussion, which makes sense given our genres. It also makes sense given our factor map, which shows that ZCR is negatively correlated with the Classical genre, which may be more likely to contain only string, brass, or woodwind instruments, and positively correlated with Metal and Hip-Hop, which tend to feature percussive sounds more prominently and consistently.

Reading this Plot

The histograms below all show the distributions of the data. The x-axis shows the values and the y-axis shows us the frequency of occurrence of any of those values. The vertical lines represent the quartiles of the distribution, with the center line representing the mean. The code listed above `quantile(mfdata$bpm)[2:4]` omits the first and fifth quantile, the minimum and maximum, from plotting. The MFCC values on the x-axes of these histograms does not represent frequency values in Hz, but relative power in that frequency bin (see more here). The rest of the values are raw values for their units (Bandwidth, for example, is in Hz, and bpm is beats per minute).

```
# Sample code used to create the histograms
h1  <- ggplot(data = mfdata, aes(x = bpm)) +
            geom_histogram(bins = 30, fill = cfv[1]) +
            geom_vline(xintercept = quantile(mfdata$bpm)[2:4])
```



7.2.2.2 Binning and the Complete Disjunct Table

Because of the size of the dataset (1000 observations), I don't really see anything that makes me think I should specify any specific binnings for these data, so we're going to go with the default for `BinQuant()` (from the data4PCCAR package), which is four.

First thing I'm going to do is create a dataframe to put our binned variables in. Then I'm going to run a for loop binning the variables, and then another loop over those data using the Spearman Rank Correlation to test how well they correlate with the original data. The function we use for the Spearman rank correlation is `cor(x, y, method = "spearman")`. This correlation is only helpful when the variable is linear. If there's a nonlinear relationship, the Spearman rank correlation isn't a helpful piece of information. That's why it's important to look at the histograms.

Finally, I'm going to take the bins and make them nominal. This creates as many variables as there are levels of all the variables. For this dataset, I used four bins for each variable, so I ended up with four variables for each original variable (e.g. bpm.1, bpm.2, bpm.3, bpm.4). As stated in the introduction, each observation (audio file/song) has one and only one '1' for each of the original variables, corresponding to the bin (quartile) in which the value for that observation fell.

7.3 Distributions Table

The table we get from the output below shows us what the ranges of each of the bins are, how many of each variable are binned in each bin, and how accurately these bins align with the original data. Below are the first 5 variables. In a perfect world, we'd see 250 observations in each variable, which we very nearly do. Things won't always be this neat.

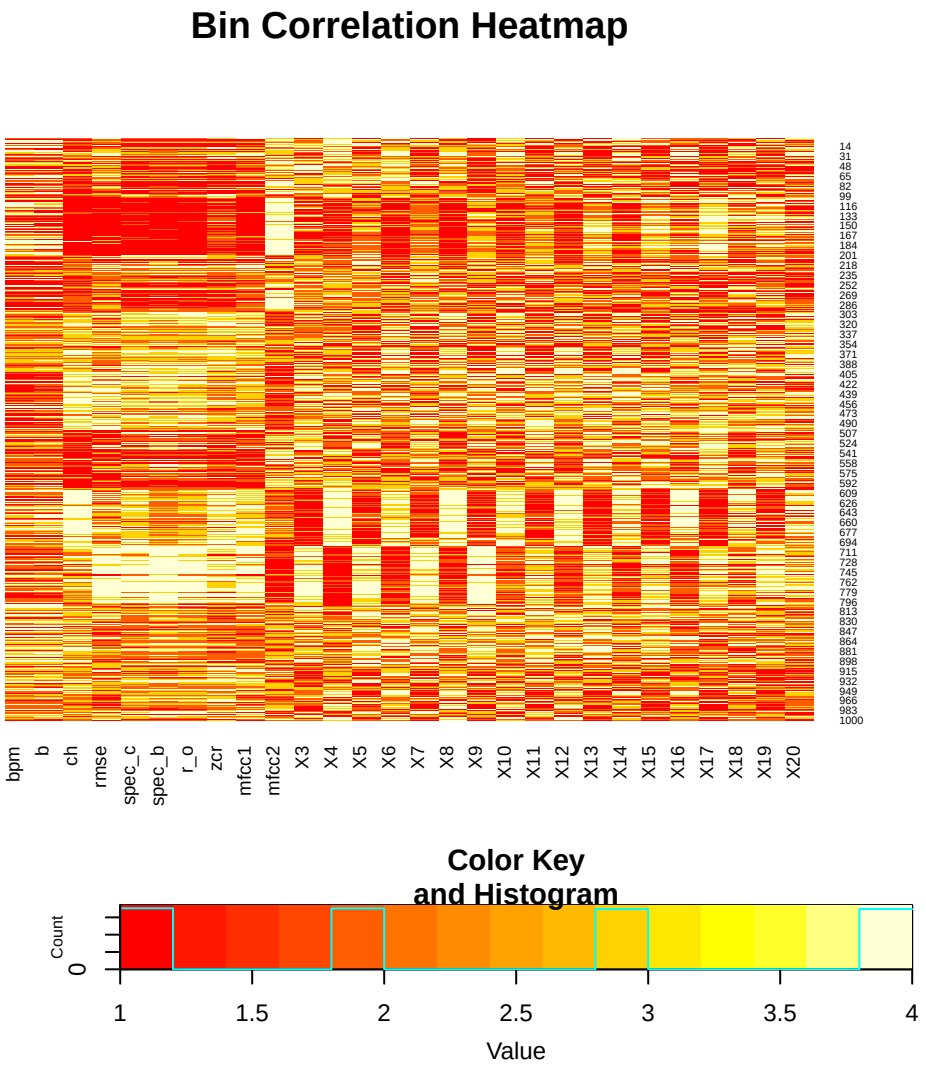
	bpm	b	ch	rmse	spec_c	spec_b
Minimum	54.9783910	18.0000000	0.1717823	0.0052756	569.9307210	897.9943189
Cutpoint 1	99.3840144	47.0000000	0.3196414	0.0866255	1627.7939309	1907.1365050
Cutpoint 2	117.4538352	56.0000000	0.3830746	0.1224477	2209.4687803	2221.4089826
Cutpoint 3	135.9991776	65.2500000	0.4359742	0.1757926	2691.9697019	2578.4743520
Maximum	234.9076705	117.0000000	0.6635727	0.3980119	4434.4394443	3509.5786771
Bin 1	286.0000000	263.0000000	250.0000000	250.0000000	250.0000000	250.0000000
Bin 2	263.0000000	266.0000000	250.0000000	250.0000000	250.0000000	251.0000000
Bin 3	234.0000000	221.0000000	250.0000000	250.0000000	250.0000000	249.0000000
Bin 4	217.0000000	250.0000000	250.0000000	250.0000000	250.0000000	250.0000000
Spearman	0.9689706	0.9680422	0.9682464	0.9682464	0.9682464	0.9682456

Bin Correlation Heatmap

The heatmap below shows us the what our data looks like post-binning. There are some pretty clear distinctions between genres and between spectral markers, although some of the spectral markers do seem to be moving in blocks. There are clear differences between the 600 and 700 groups, which are the rows of metal and pop. Likewise, MFCCs are very clearly alternating high and low values by odds and evens.

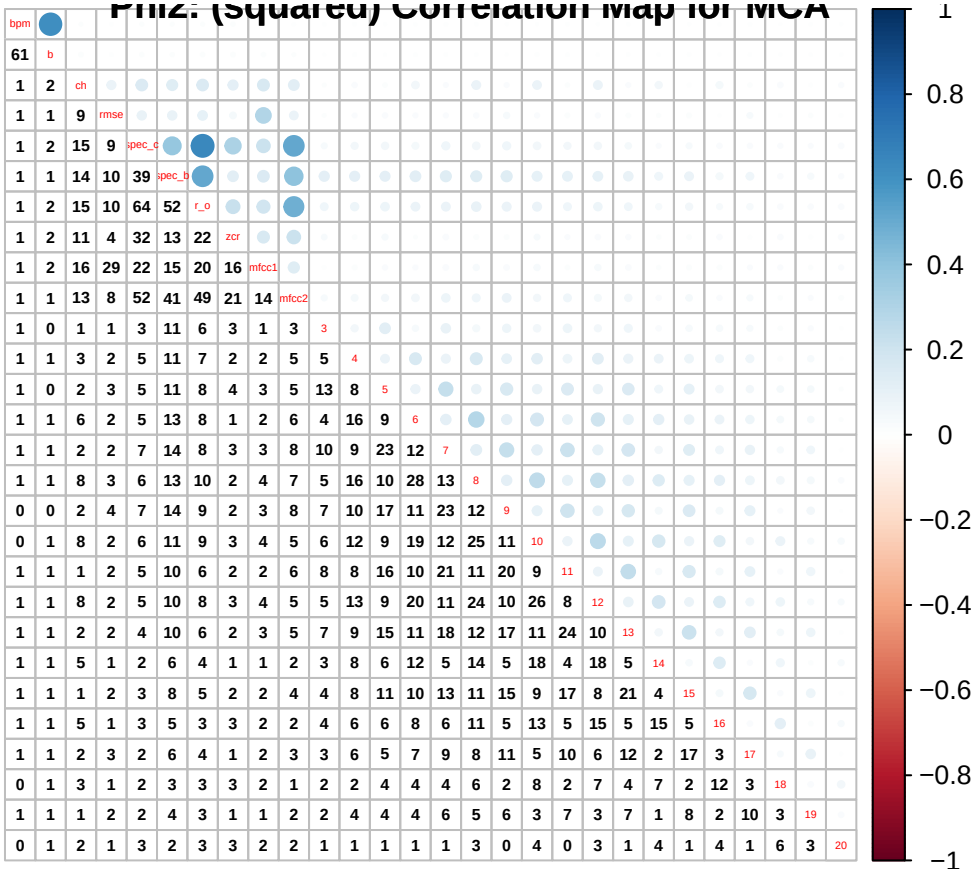
For reference, the genre labels and the numbers corresponding to those values are listed below:

Genre	Numbers	Genre	Numbers	Genre	Numbers	Genre	Numbers
blues	1-100	disco	301-400	metal	601-700	rock	901-1000
classical	101-200	hip hop	401-500	pop	701-800		
country	201-300	jazz	501-600	reggae	801-900		



7.3.1 Phi2 squared & Burt Table

The next step for an MCA is to compute a Burt table, which is computed by multiplying our nominal (complete disjunct) table by its transpose to get a contingency table. For this we use the function `phi2mat4BurtTable` from the `data4PCCAR` package. The function gives us two outputs: the phi^2 matrix and the Burt table. The Burt table is a contingency table for all of our nominalized variables, and the phi^2 table shows us the squared correlation between the variables. We can then visualize this using a correlation plot. The correlation plot below is the phi^2 correlation.



7.4 Analysis

The code below shows us the MCA and the MCA inference batteries, respectively.

```
mfMCA <- epMCA(mfnom, make_data_nominal = FALSE,
               DESIGN = music.genre,
               graphs = FALSE, correction = "bg"
               )
```

```
mfMCA.inf <- epMCA.inference.battery(DATA = mfnom,
                                     make_data_nominal = FALSE,
                                DESIGN = music.genre,
                                make_design_nominal = TRUE,
                                graphs =  FALSE, # TRUE first pass only
                                correction = "bg"
                                )
```

```
## [1] "It is estimated that your iterations will take 0.23 minutes."
## [1] "R is not in interactive() mode. Resample-based tests will be conducted. Please take note o
## ================================================================================
```

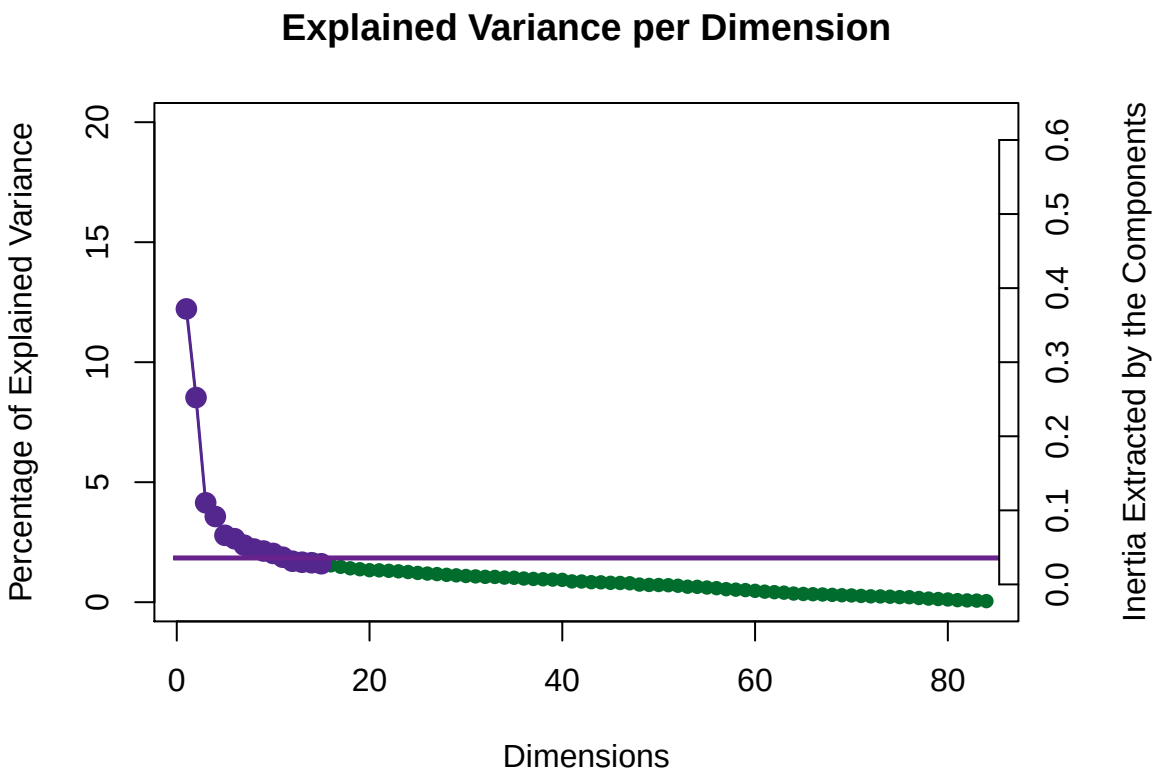
7.5 Results

7.5.1 Scree Plot

The scree plot shows us which of the eigenvalues/dimensions of variance extracted are significant. The dimensionality of this analysis is so much higher than the previous one because we have artificially inflated the number of variables, and therefore the dimensionality of the analysis, by binning. We therefore need to make a correction regarding the eigenvalues (Hervé Abdi 2018). There are two possible corrections, that of Benzécri and that of Greenacre, which can be specified in the `epMCA` function. Hervé Abdi (2018) suggests that Greenacre (1993) is a slightly better correction, and we've used that here.

Here we have 15 significant dimensions out of a possible 26. The Kaiser criterion shows us that the average value for the eigenvalues is approximately 6, and that we have multiple significant dimensions below that. See PCA or CA for more on interpreting scree plots. As always, remember that significant doesn't necessarily mean important.

```
PlotScree(ev = mfMCA$ExPosition.Data$eigs,
          p.ev = mfMCA.inf$Inference.Data$components$p.vals,
          plotKaiser = TRUE,
          )
```

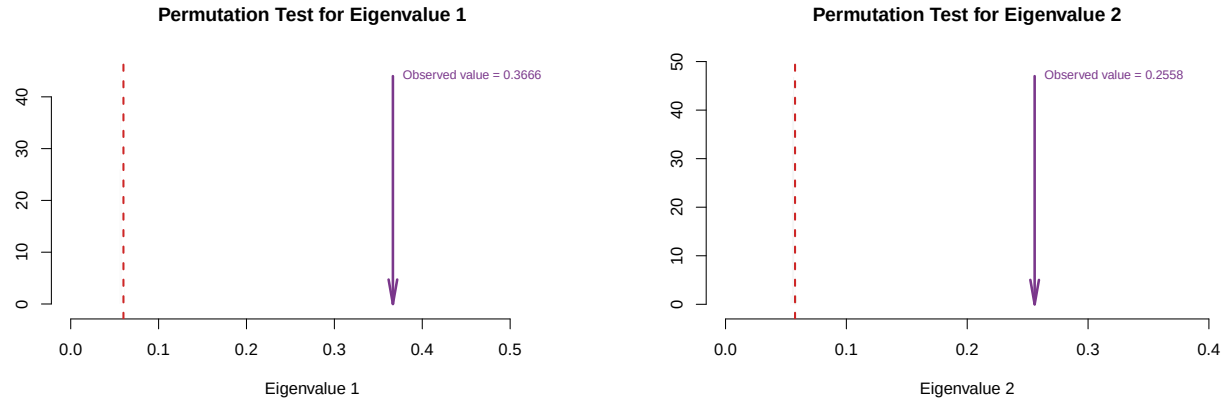



```
a001a.screePlot <- recordPlot()
```

7.5.1.1 Permutation Testing

As you can see below, the first two eigenvalues was tested against the permutations. All are significant in that it is unlikely that they arise by chance. Remember that the first value is an omnibus test to tell us whether or not there's something in the data. The histogram of the distribution of the permuted eigenvalues

is difficult to see because it's stacked up right along the significance line. Zooming in makes it possible to see. See PCA for more on interpreting these histograms.



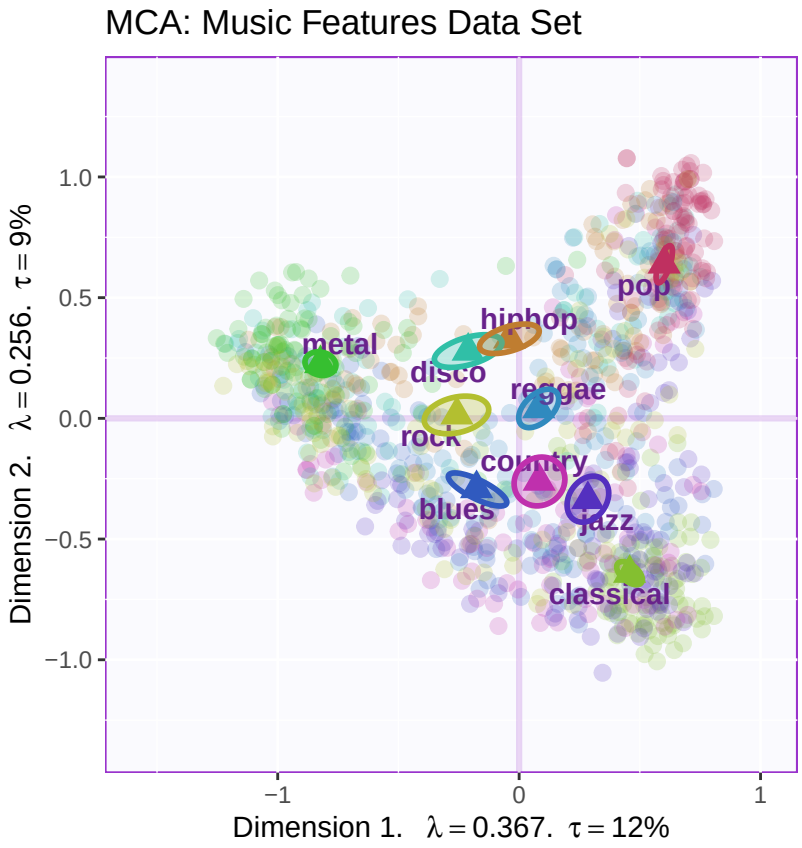
7.5.2 Factor Maps

Next we need to visualize all of our data on the principal components space. The first thing we're going to do is visualize all of our observations, including their means and confidence intervals for their means as we have done before. This will help us to understand how the observations map onto the variables and what patterns we can discern from that. If you've seen the cookbooks on PCA, Inference PCA, and CA, this should look familiar.

You may notice that these plots are flipped across the center. In PCA, we saw Metal on the right and Pop and Classical on the left, but here they've reversed. That's largely an artifact of coincidence. Running the same analyses on different computers, different operating systems, etc., will often give flipped results.[1]

```
# I-set map ----
# a graph of the observations
mf.Imap <- PTCA4CATA::createFactorMap(
  title = 'MCA: Music Features Data Set',
  mfMCA$ExPosition.Data$fi,
  col.points = mfMCA$Plotting.Data$fi.col,
  display.labels = FALSE,
  alpha.points = .2
)
# Labels
label4Map <- createxyLabels.gen(1,2,
                    lambda = mfMCA$ExPosition.Data$eigs,
                    tau = mfMCA$ExPosition.Data$t)
# Put it together
a002.Map.I <- mf.Imap$zeMap + label4Map
```

[1] More important than the specific factor or loading directions is how they relate to one another. For example, regardless of the direction in which a given variable loads, as long we see the variables consistently loading with the same variables and against the same variables, we're getting consistent results.



7.5.3 Disjunct Factor Maps

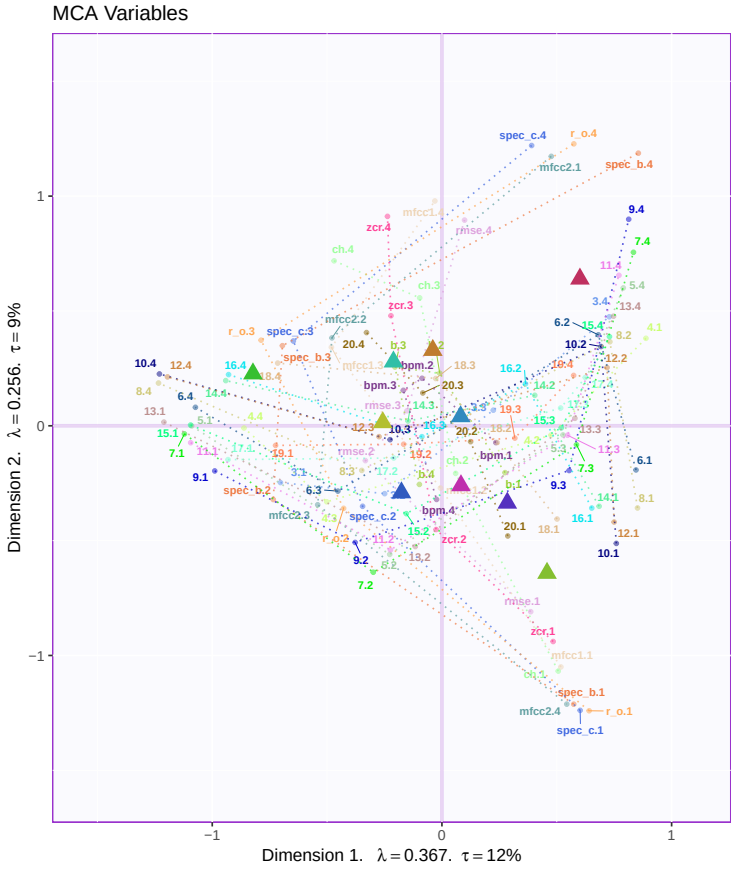
Reading this Plot

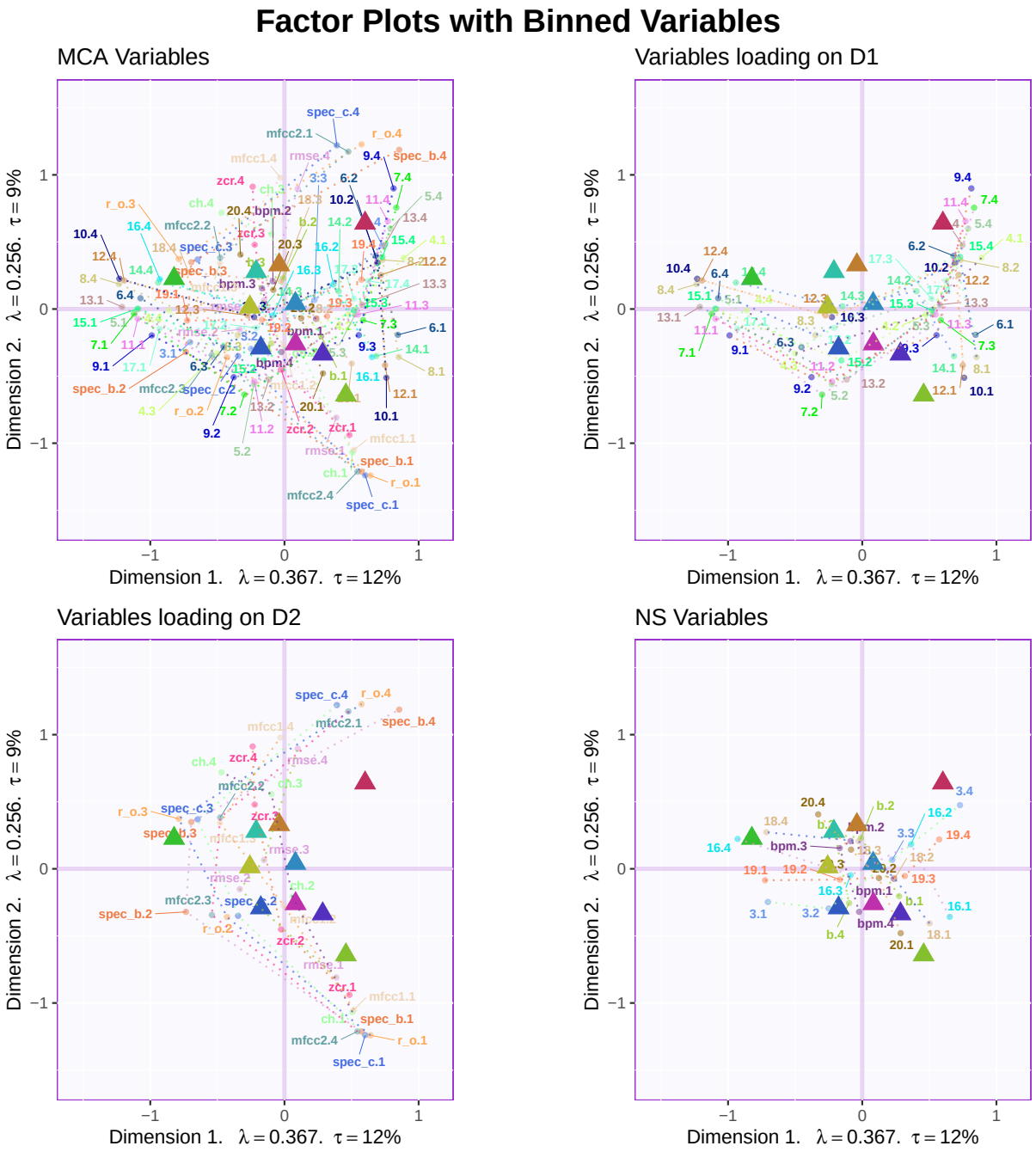
These factor maps show us how the observations map onto the factor space. Because we're using disjunctively coded data, we can also see what levels of those variables map onto which areas of the factor space. There are four maps below, each of which show certain variables. The first map shows all of the variables, the second shows just the variables that load significantly on the first dimension, the third shows the variables that load significantly on the second dimension, and the fourth shows the variables that don't load significantly on either one of those variables. I did this by specifying which of the rows of `mfMCA$ExPosition.Data$fj` I wanted to display on each graph. I selected them after looking at the contribution plots (below) to see which variables contributed what to which dimensions. I've set the constraints (mins and maxes for the axes) on all of them to be the same, so the scale is the same, and it's easier to see how much they're loading relative to one another.

Note that the first dimension is made up entirely of the MFCCs, which makes sense because of the structure of the measurement and the fact that they're basically measuring the same thing. The second dimension is driven by the other named spectral components and MFCCs 1 and 2, which means that they're measuring something different to the other MFCCs. The fourth plot, the non-significant variables, consists of beats, beats per minute, and the last few MFCCs. Interestingly, none of these variables are plotting in a linear fashion on these factor maps, and comparing the disjunct factor maps to the observation factor plot shows us similar results to what we would expect to see given the heatmap we saw above. As an example, MFCC 2 loads such that the lowest values of the variable correspond with the placement of the pop genre on the observation factor plot, and likewise the highest values load along with the classical genre.

```
#The disjunct factor maps are created the same way the other factor maps are:
# First you make the base map using the FJs,
# the factor scores for the levels of binned variables
```

```
BaseMap.Fj.all <- createFactorMap(X = Fj.all, axis1 = axis1, axis2 = axis2,
                                  title = 'MCA Variables', cex = 1,
                                  col.points = col4Levels$color4Levels,
                                  col.labels = col4Levels$color4Levels,
                                  text.cex = 2.5, force = 2)
# Assign the parts of the map, depending on what you want to display
###### This one uses just the Fjs
b001.BaseMap.Fj.all <- BaseMap.Fj.all$zeMap + label4Map
###### This one creates the maps using the group means also
b002.BaseMapNoDot.Fj.all  <- BaseMap.Fj.all$zeMap_background +
                             BaseMap.Fj.all$zeMap_text + label4Map +
                             fi.plot.redux$zeMap_dots + fi.plot.redux$zeMap_text
# Then you create the lines that connect the levels of the binned variables
lines4J.all <- addLines4MCA(Fj.all, col4Var = col4Var)
# Then you put it all together
b003.MapJ.allwm <-  b001.BaseMap.Fj.all + lines4J.all + fi.plot.redux$zeMap_dots
b003.MapJ.all <-  b001.BaseMap.Fj.all + lines4J.all
# This is just the first map. Check the .rmd file for the creation of the other
# plots, as well as the grid.arrange line we use to put them all together.

# The first plot is displayed here by itself and with all the other plots below.
b003.MapJ.allwm
```





7.5.4 Contributions

Reading this Plot The plot below shows us contributions for each dimension, as well as the bootstrap ratios for each of the disjunctively coded variables. The contributions are showing us simply how much each variable loads, irrespective of the sign of that loading. This is because for this type of analysis, we already broke down how much each component loads and in what direction for what value with the binning and the disjunct factor plots we did above. We also see those loadings in the bootstrap ratios. The bootstrap ratios on top show us how much consistently each bin of each variable loads on each component, and in what direction. It's easy to see the differences between the spectral components and the MFCCs in this, with the MFCCs driving component 1 and spectral components loading on variable 2. You can also see if you look closely that most of the variables load in order - the first bin loads the most positively while the fourth bin loads most negatively (or vice versa) and the middle two bins fall somewhere in between.

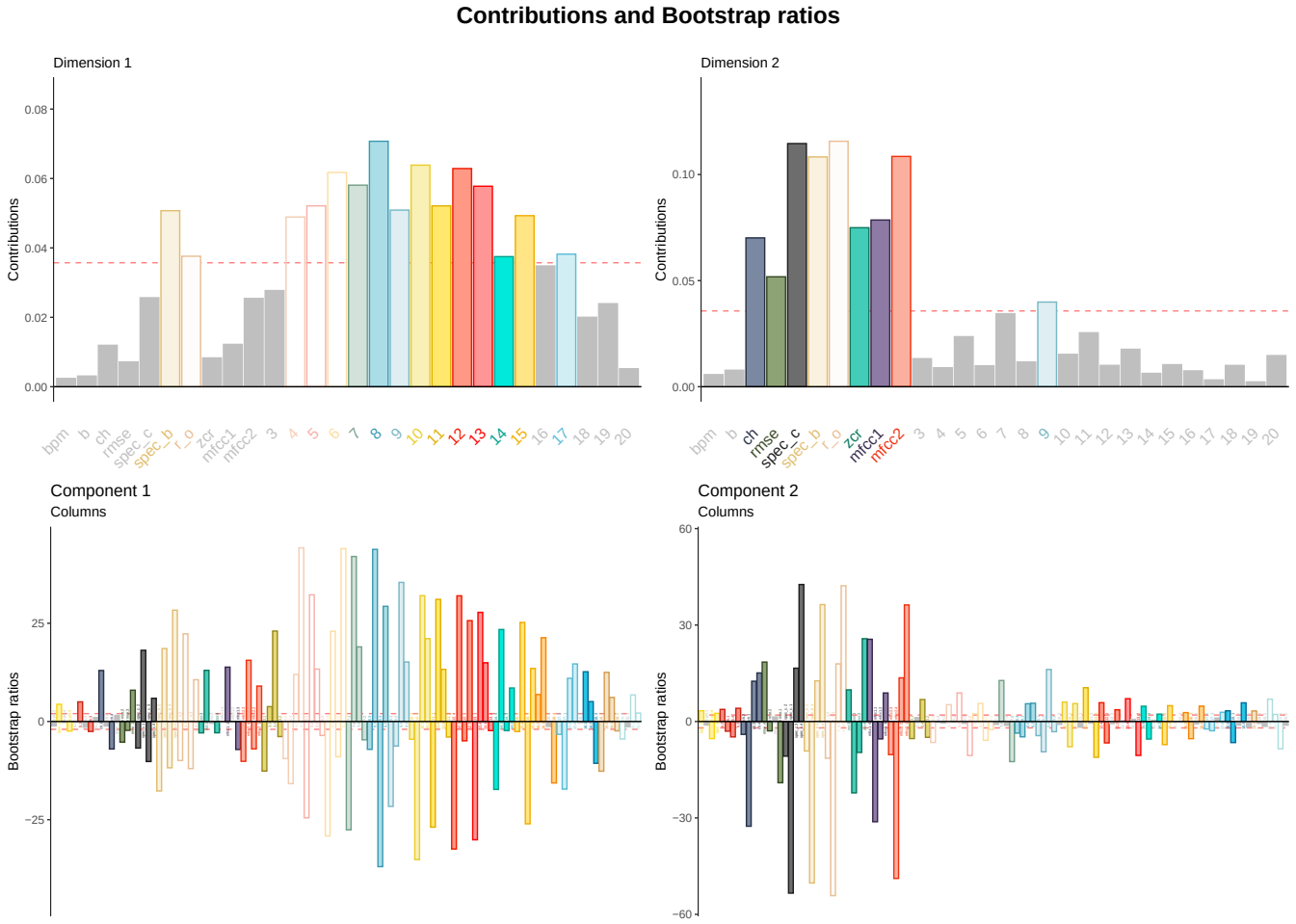
Sometimes they don't, which happens in the case where some variables load in a non-linear fashion. Beats and BPM are examples.

The code below creates the plots for the contributions and the bootstrap ratios. See the rmd file for how we arrange them on the page. The labels for each of the variables on the bottom plot are small, but visible if you zoom in. If you're looking at this on a webpage, open in a new tab and zoom in. If you're looking at a PDF, just zoom in.

```
mfcont <- ctr4Variables(mfMCA$ExPosition.Data$cj)
mfcont.s <- ctr4Variables(mfMCA$ExPosition.Data$cj)*sign(mfMCA$ExPosition.Data$fj)
# Here we plot contributions of columns for component 1
varCtr1 <- mfcont[,1]
names(varCtr1) <-rownames(mfcont)
ctrJ.1 <- PrettyBarPlot2(varCtr1,
                         ylim =c(0, 1.2*max(varCtr1)),
                         ylab = 'Contributions',
                         font.size = 4,
                         threshold = 1/ nrow(mfcont),
                         color4bar = cfv,
                         angle.text = 45
                         )+ ggtitle("", subtitle = 'Dimension 1')
# and for component 2
varCtr2 <- mfcont[,2]
names(varCtr2) <- rownames(mfcont)
ctrJ.2 <- PrettyBarPlot2(varCtr2,
                         threshold = 1 / NROW(varCtr2),
                         font.size = 4,
                         color4bar = cfv, # we need hex code
                         ylab = 'Contributions',
                         ylim = c(0, 1.2*max(varCtr2)),
                         angle.text = 45,
                         ) + ggtitle("", subtitle = 'Dimension 2')
```

```
BR.I <- mfMCA.inf$Inference.Data$fj.boots$tests$boot.ratios
laDim = 1
# Plot the bootstrap ratios for Dimension 1
ba001.BR1 <- PrettyBarPlot2(BR.I[,laDim],
                            threshold = 2,
                            font.size = 1,
                     color4bar = t(cfb), # we need hex code
                    ylab = 'Bootstrap ratios', #sortValues = TRUE
                    #ylim = c(1.2*min(BR[,laDim]), 1.2*max(BR[,laDim]))
) + ggtitle(paste0('Component ', laDim), subtitle = 'Columns')
# Plot the bootstrap ratios for Dimension 2
laDim = 2
ba002.BR2 <- PrettyBarPlot2(BR.I[,laDim],
                            threshold = 2,
                            font.size = 1,
                     color4bar = t(cfb), # we need hex code
                    ylab = 'Bootstrap ratios', #sortValues = TRUE
                    #ylim = c(1.2*min(BR[,laDim]), 1.2*max(BR[,laDim]))
```

```
) + ggtitle(paste0('Component ', laDim), subtitle = 'Columns')
```



7.6 Conclusions

One thing that this does is it shows how the levels (bins) of each of the variables are reflected in each of the groups. To visualize this more effectively, we can look at the factor maps with the means of the groups plotted as well.

- **Component 1**
  - This explains why the variables we saw before loaded the way they did:
  - On the negative side, we have all of the variables driving the horizontal component, but on the positive side, the variables also seem to load a bit on the second component.
  - We can also see how the Odd MFCCs have lower values on the negative end of component 1 and higher values on the positive side, while even MFCCs have lower values on the positive side of component 1 and higher values on the positive end.
  - This does really help separate which of the genres tend to load in which way on these spectral components.
- **Component 2**
  - Component two really helps separate out how music recording techniques differ between genres.
  - The highest values of: spectral centroid, roll off, and spectral bandwidth are all more associated with pop than any other genre.
  - Likewise, the lowest values for every single component is more highly associated with classical than any other genre.

  - We can see how many genres cluster in the middle, suggesting that there aren't too many differences in their recording techniques.
- Interpretation: Acoustically recorded music has lower extreme spectral components than electronically produced music. This interpretation isn't new, and is the same as we saw for PCA, but offers the additional value of showing how the levels of the variables contribute to the placement of the observations in the factor space. This is similar to what we will see in DiCA.

Part III: Two-Table Techniques

Chapter 8

Barycentric Discriminant Analysis

8.1 Intro to BADA

Discriminant analysis is a method of analysis that analyzes a dataset in which multiple measures (variables) describe observations. BADA is a method used on a dataset in which each observation belongs to or can be assigned to a specific pre-determined category. At first glance, it appears that we're only using a single table, but the way BADA works is to create a separate table of the group means by variables that is used to create the factor space. This explains the difference in appearance between the axes of our PCA and the ones created here. After the PCA is performed on the barycenters of the groups, the individual observations are projected onto the group factor plot as supplementary observations. The BADA then determines, relative to the barycenters, which of the individual observations fit into which groups. The analysis below also creates a confusion matrix, which shows us how many of the observations from each group get classified into each group, and to which groups they get classified. An ideal result would show each result classified in its respective group. For more on BADA, see Hervé Abdi and Williams (2010a), Hervé Abdi, Williams, and Béra (2018).

8.1.1 Strengths & Weaknesses

Strengths

- Even if the groups aren't very well differentiable, that information alone can be a useful metric. If you're expecting groups to be different, and they aren't, or vice-versa, that can tell you that there's something to dig in to, or that you did your analysis wrong. Either way, useful information.
- The group discrimination is useful for figuring out whether the groups are real or not. You can see whether you're just looking at a large dataset, in which case the confidence intervals (from bootstrapping) for the means will be fairly tight, and it will appear like there are group differences, even if there is a lot of overlap and there may not be real differences between the groups.

Weaknesses

- Because the groups are discriminated relative to the barycenters of the groups, if there are tightly clustered group means and lots of observations, it's difficult to assign the observations to groups. Looking at the tolerance intervals is helpful in this regard because it can give you an idea of how much the observations overlap and how likely you are to be able to discriminate between the groups.

8.1.2 Dos and Don'ts

Do:

- Remember that the accuracy of the confusion matrix is relative to how many groups you have. If you have two groups, 50% is chance. If you have three groups, 33.3% is chance, and so on.

- Remember that although group means represent our best estimate for a given group, tolerance intervals show you how much the observations for the groups really overlap.

Don't:
- Fool yourself into thinking that your results are truly generalizable. When you're evaluating your fixed and random effects, you're still only generalizing within your sample. Things like demographic data, which won't have changed between your full sample and your leave-one-out model, will still need to inform your interpretation.

Research Questions
Questions for BADA should be guided by the idea of group differences.
- How are we separating groups across the extracted principal components?
- Are these group differentiable enough for accurate classification of observations?
- What can the confusion matrix tell us about the distribution of our observations?

8.2 Data

This dataset is the music features dataset, it's the same as the one used for the PCA. Here's what the data table looks like. I've pulled out one of each of the first five genres in the table to show what the variables/measures and the values/observations look like in the interest of saving space, only 8 variables are shown. Call `head(mfdata)` to see more.

	ch	rmse	spec_c	spec_b	r_o	zcr	mfcc1	mfcc2
blues.00081.au	0.3802602	0.2482623	2116.943	1956.611	4196.108	0.1272725	-26.92978	107.33401
classical.00091.au	0.2237383	0.0447457	2192.798	1911.986	4066.467	0.1524379	-251.31345	88.72515
country.00013.au	0.3680927	0.0800457	2812.346	2842.053	6062.663	0.1295650	-126.16004	77.63675
disco.00063.au	0.5478499	0.2939197	2583.278	2626.311	5855.473	0.0997726	-51.75267	70.33190
hiphop.00034.au	0.4699237	0.2279124	3191.045	2735.905	6353.995	0.1691660	5.87653	53.23913

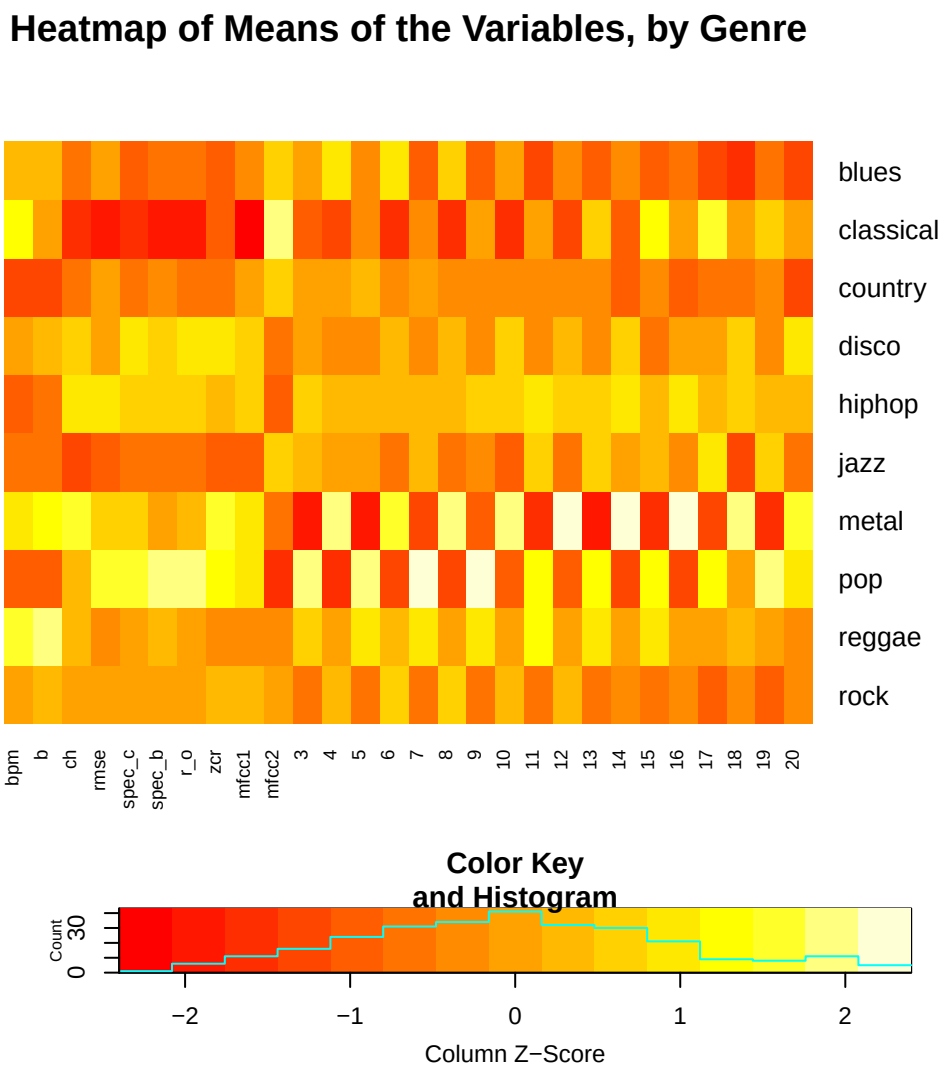
8.2.1 Raw Data Visualization: Heat Map

The heatmap below shows the average normalized values of each of the variables by group. On the left side, we have the named spectral components and on the right we have the MFCCs. Each of the variables shows something interesting about music genre. For bpm/b (the first two columns) we see higher average values (and therefore higher tempos) for Reggae, Classical, and Metal. Higher tempos in Metal seems self-explanatory, but Classical and Reggae may be a little bit less intuitive. Because these were electronically extracted signal data, the program analyzes the periodicity of the volume cycles (stronger beats are generally louder) to extract the number of beats in the file and therefore the tempo. Although reggae does tend to be slower, there may be less variability in the volume across beats compared to say, country, so the program likely extracted more beats as significant in its analysis. Reggae also uses a different beat cycle for strong and weak beats than the other genres here. In most blues, rock, jazz, country, etc., there are accents on the 'backbeat', or beats 2 and 4. In Reggae, that cycle is turned around, and the accents go on 1 and 3. This may account for the difference in analysis between the genres. For classical, it may surprise many people to hear that may classical tunes are quite quick and have high bpms. Also, because the signal extraction for beats often relies on percussive signals (wide frequency bands & sharp attack envelope), and classical music has less consistent percussion instruments, it has to rely on other signals, which may contribute to this value.

Notice that classical and pop have very different levels of Spectral Bandwidth, Spectral Centroid, and Roll off. This makes sense in light of the understanding how the respective genres are recorded. Acoustic instruments are able to produce less extreme spectral components than can be produced electronically.

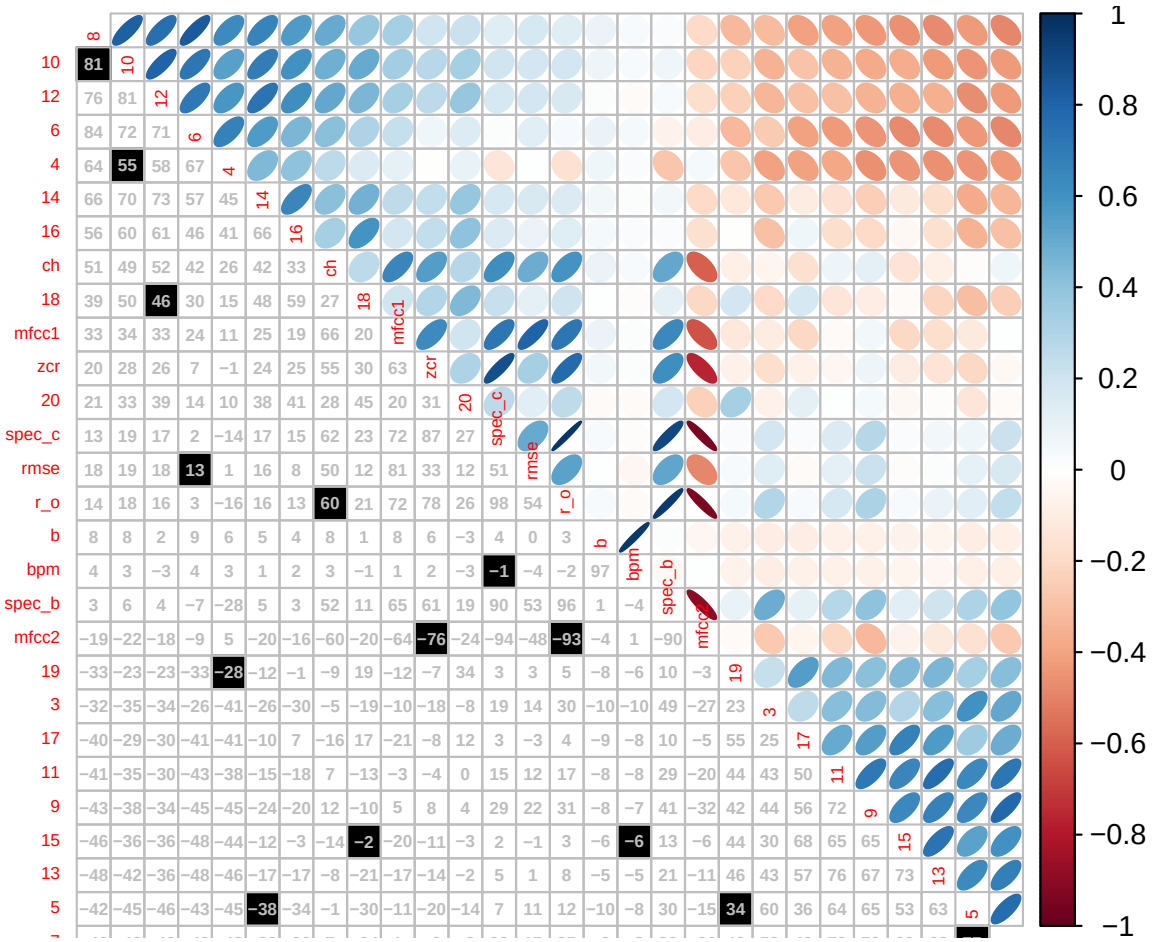
Also, seeing that metal has the highest average Zero Crossing Rate makes sense knowing that the ZCR is used as a measure of how busy a signal is, and is often a good indicator of percussion instruments and distortion, and Metal is a percussion - heavy genre.

Because the heatmap below compares the means of the groups/variables, they are being compared (as the key shows us) by z-scores. This means that the 'pure orange' are the most average values and likely reflect genres that sit closest to the barycenter. Those with more extreme values (darker reds or lighter yellows) are further from the mean and likely indicate that the group sits further from the barycenter.



8.2.2 Correlation Plots

The standard correlation plot below shows the correlations visually (top) and numerically (bottom). The variables are ordered according to their loadings on the first component. This makes sense given our results from the PCA, namely that the even MFCCs greater than 2 load positively on the first component and the odd MFCCs greater than 1 load negatively on the first component. It makes sense that these are opposed to one another as they are intentionally decorrelated. Likewise, the variables that load more heavily on the second component (or, in the case of b and bpm, the 3rd or 4th component) are more central in the plot.



8.3 Analysis

This next section runs the actual BADA analysis. `tepBADA` runs the BADA and `tepBADA.inference.battery` runs the inference battery that allows us to check variables.

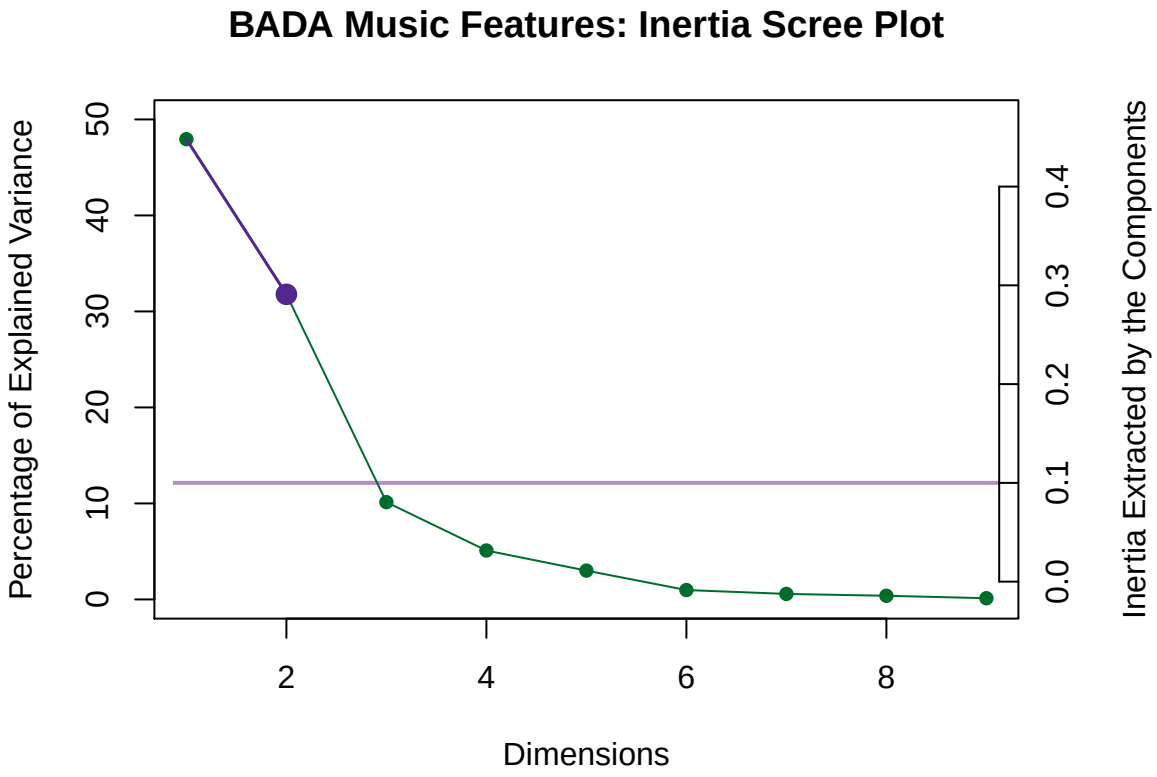
```
# Run BADA  ----
mfBADA <- tepBADA(mfdata, DESIGN = music.genre,
                  graphs = FALSE)
# Inferences ----
set.seed(70301) # we have a problem with the inference part
  # it will be addressed soon. In the meantime we fix the seed
  # for random
nIter = 100
mfBADA.inf <- tepBADA.inference.battery(mfdata,
                  DESIGN = music.genre,
                  test.iters = nIter,
                  graphs = FALSE)
```

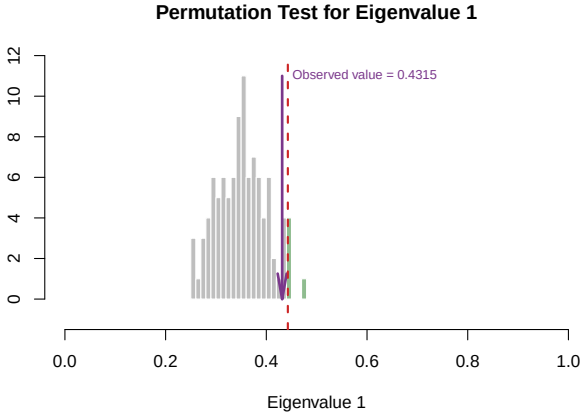
```
## [1] "It is estimated that your iterations will take 0.2 minutes."
## [1] "R is not in interactive() mode. Resample-based tests will be conducted. Please take note o
## ================================================================================
```

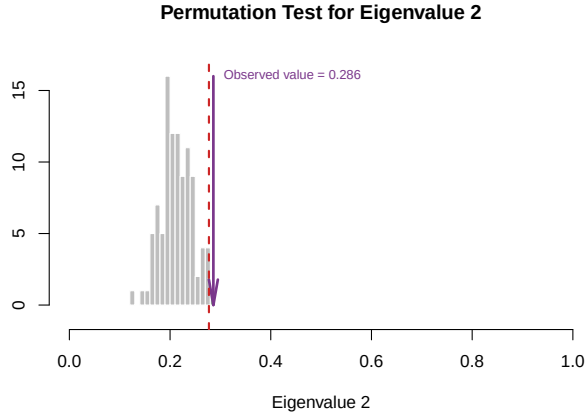
8.4 Results

8.4.1 Scree

The scree plot shows us which of the eigenvalues extracted by the BADA are significant. The dimensions here (9) are one less than the number of groups. The p values associated with the eigenvalues show us which values (via permutation testing in `tepBADA.inference.battery`) are significant. See Inference PCA for more on interpreting scree plots and their permutations. Here we have an interesting example where the first eigenvalue, although the strongest, seems to be not significant. As we discussed in earlier pages, the first eigenvalue is essentially an omnibus test, the non-significance of which seems to carry through this entire analysis.



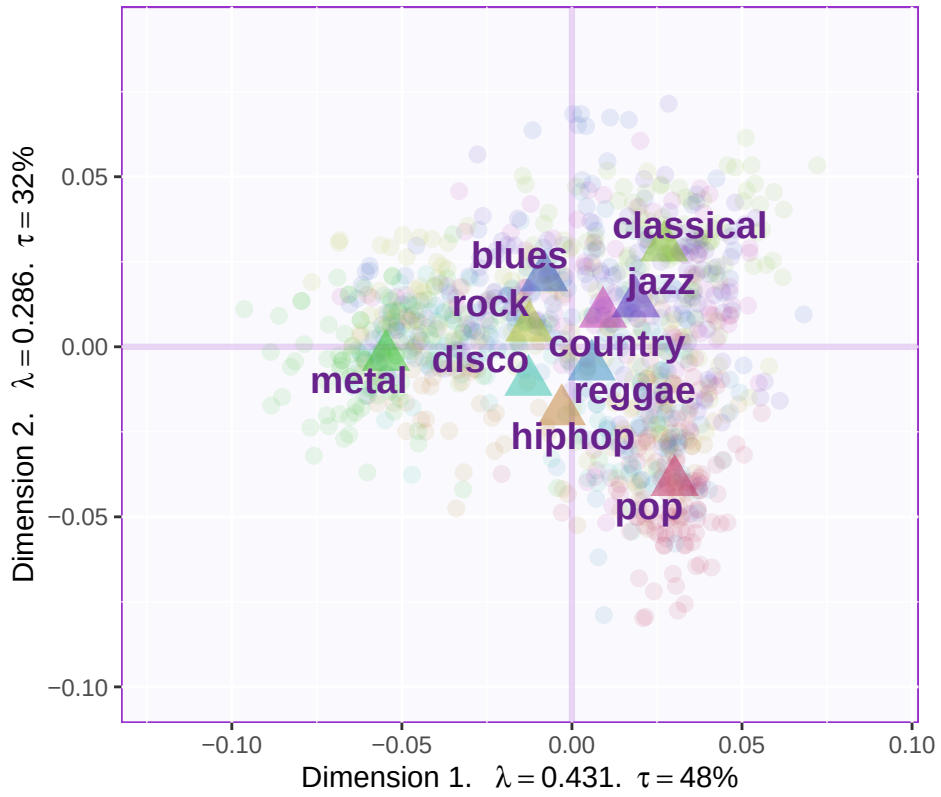




8.4.2 Factor Map & Observations

The map produced below shows us the factor scores of all of the observations and the means of the groups plotted on top of the principal components. It looks similar to each of the factor maps produced before, but is flipped around both axes from the factor map we saw in PCA. As discussed in MCA, this simply an artifact of coincidence.

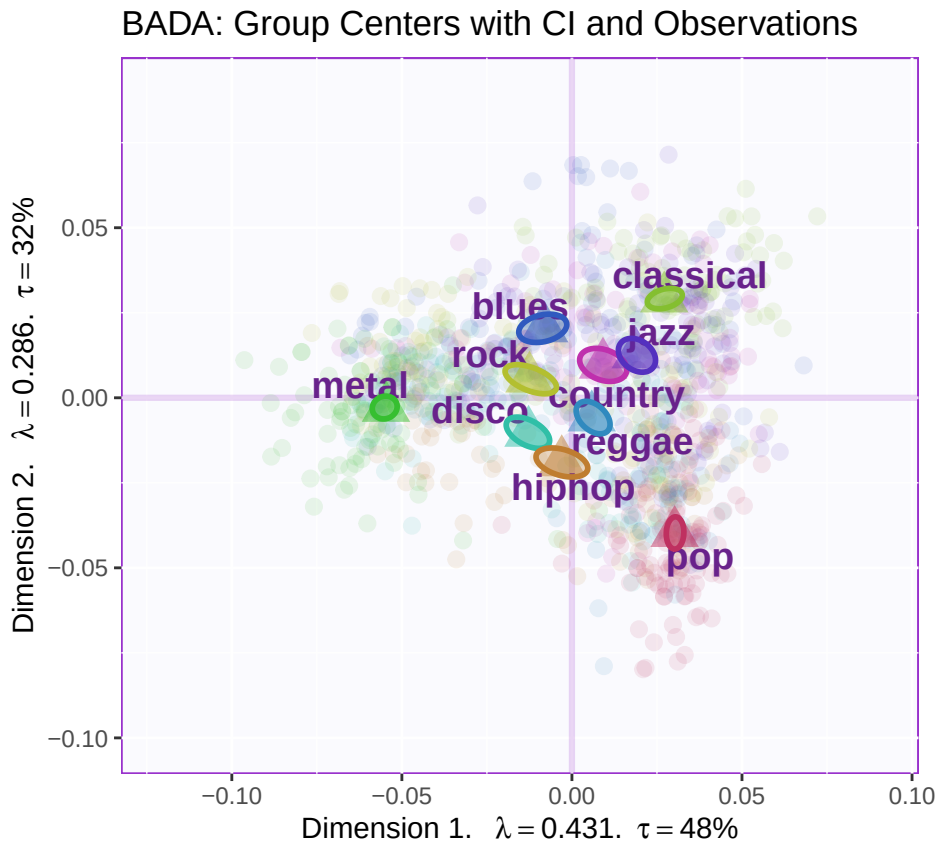
```
#  Factor map for the observations
Imap <- createFactorMap(mfBADA$TExPosition.Data$fii, alpha.points = .1,
                        col.points = mfBADA$Plotting.Data$fii.col, display.labels = FALSE
                        )
# make labels for the graph
label4Map <- createxyLabels.gen(1,2,
                                lambda = mfBADA$TExPosition.Data$eigs,
                                tau = mfBADA$TExPosition.Data$t)
# Get the means for the next plot using genre as a design variable
genremeans <- getMeans(mfBADA$TExPosition.Data$fii, music.genre)
# Map for the means of the genres/groups
MapGroup <- createFactorMap(genremeans, col.points = unique(mfBADA$Plotting.Data$fii.col),
                            # use the constraint from the main map
                            constraints = Imap$constraints,
                            pch = 17, cex = 6, text.cex = 5
                            )
# Put it all together:
a003.bada <- Imap$zeMap + label4Map + MapGroup$zeMap_dots + MapGroup$zeMap_text
a003.bada
```



8.4.3 Confidence Intervals for Means

The graph below shows us the same factor map as above, but with the confidence intervals for the means, as determined by the inference battery, plotted as an oval around the group mean. As you can see, the confidence intervals are pretty tight, which means, basically, that I have a large, consistent dataset.

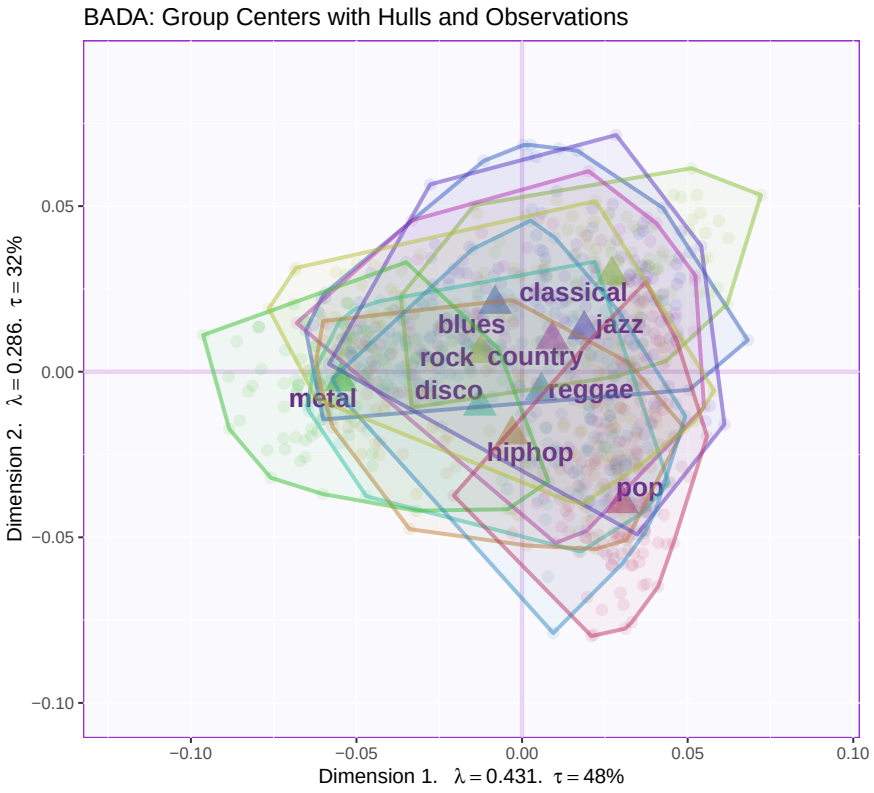
```
# Create Confidence Interval Plots
fi.boot <- mfBADA.inf$Inference.Data$boot.data$fi.boot.data$boots
# This helps us get the colors by replacing the punctuation in the rownames with nothing,
# hence the "". The effect is the deletion of the punctuation.
rownames(fi.boot) <- sub("[[:punct:]]","",rownames(fi.boot))
# use function MakeCIEllipses from package PTCA4CATA
GraphElli <- MakeCIEllipses(mfBADA.inf$Inference.Data$boot.data$
                              fi.boot.data$boots[,c(1,2),],
                            col = unique(mfBADA$Plotting.Data$fii.col),
                            p.level = .95
                            )
# Put everything together
a004.bada.withCI <-  Imap$zeMap_background + Imap$zeMap_dots +
                     MapGroup$zeMap_dots + MapGroup$zeMap_text +
                     GraphElli + label4Map +
                     ggtitle('BADA: Group Centers with CI and Observations')

a004.bada.withCI
```



8.4.4 Tolerance Intervals

The tolerance intervals surround all of the datapoints in the data. They show you explicitly what the boundaries of the distributions of your data are. This is useful especially in this example and in DiCA for comparing to our group barycenters. If the dispersion is wide and the group mean CI is tight, then it's likely that the tightness of that confidence interval is due to the size of the data. That's the case here, where we have the observations for these groups overlapped, even though the means and confidence intervals are not. That suggests that we may have issues in differentiating between groups.

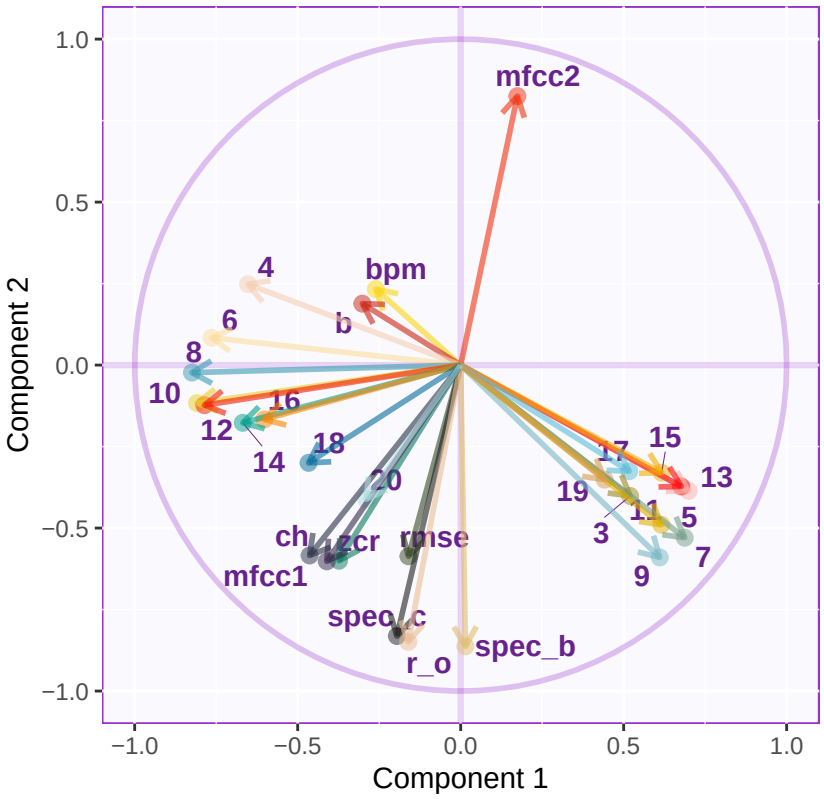
```
# This creates hulls for the groups
Fii <- mfBADA$TExPosition.Data$fii
colnames(Fii) <- paste0('Dimension ', 1:ncol(Fii))
# getting the color correct: an ugly trick
col4Hull <- unique(mfBADA$Plotting.Data$fii.col)
GraphHull <- MakeToleranceIntervals(Fii,
                                    design = music.genre,
                                    col = col4Hull,
                                    # the next line is required
                                    names.of.factors =  c("Dim1","Dim2"),
                                    p.level = 1.00,
                                    alpha.ellipse = .05
                                    )
a006.bada.withHull <-  Imap$zeMap_background + Imap$zeMap_dots +
            MapGroup$zeMap_dots + MapGroup$zeMap_text +
            GraphHull + label4Map +
            ggtitle('BADA: Group Centers with Hulls and Observations')

a006.bada.withHull
```



8.4.5 Variable Loadings

The graph below shows us what variables load on what component and to what extent. The closer an arrow is to the edge, the more variance is extracted. The angle between the variables determines the strength of their relation. Approaching 0 or 180 degrees indicates a strong correlation, and the direction of that correlation. Approaching 90 degrees indicates a weak correlation, and 90 degrees is perfectly uncorrelated.

```
cor.loading <- cor(mfdata, mfBADA.inf$Fixed.Data$TExPosition.Data$fii)
#colnames(cor.loading) <- rownames(cor.loading)

loading.plot <- createFactorMap(cor.loading,
                                constraints = list(minx = -1, miny = -1,
                                                   maxx = 1, maxy = 1),
                                col.points = cfv)

LoadingMapWithCircles <- loading.plot$zeMap +
  addArrows(cor.loading, color = cfv) +
  addCircleOfCor() + xlab("Component 1") + ylab("Component 2")

LoadingMapWithCircles
```

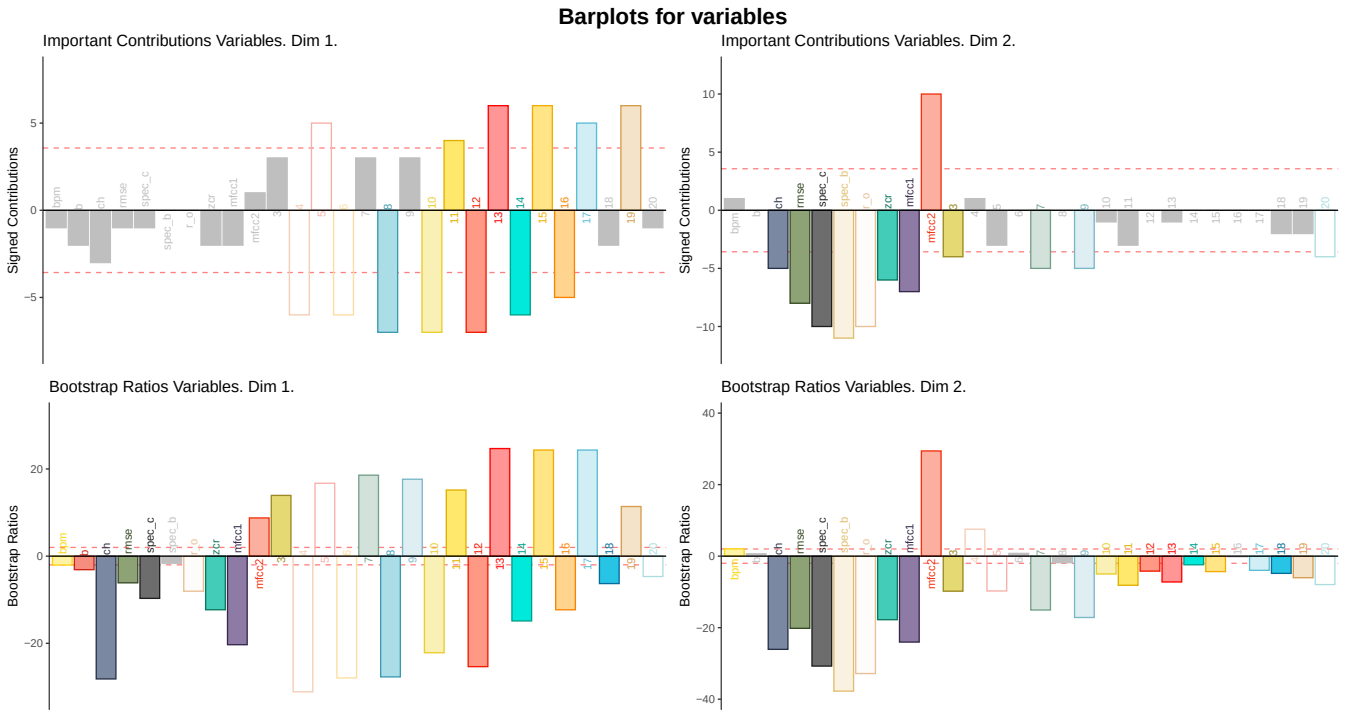


8.4.6 Contributions and Bootstrap Ratios

The plot below shows which variables contribute significantly to which components, and in what direction they contribute. The Bootstrap ratios show the same directionality, but they indicate the proportion of bootstrap ratios that significantly load on those components. The results we see below are much the same as we've seen in previous pages. See for example, the page on Inference PCA for more on reading these

plots.

```
#_____________________________________________________________________
#  Contributions Plots
#     Dimension 1
ctrj <- mfBADA$TExPosition.Data$cj
signed.ctrj <- ctrj * sign(Fj)
# BR1
c001.plotCtrj.1 <- PrettyBarPlot2(
                        bootratio = round(100*signed.ctrj[,1]),
                        threshold = 100 / nrow(signed.ctrj),
                        ylim = NULL,
                        color4bar = gplots::col2hex(cfv),
                        color4ns = "gray75",
                        plotnames = TRUE,
                        main = 'Important Contributions Variables. Dim 1.',
                        ylab = "Signed Contributions")
#     Dimension 2
c002.plotCtrj.2 <- PrettyBarPlot2(
  bootratio = round(100*signed.ctrj[,2]),
  threshold = 100 / nrow(signed.ctrj),
  ylim = NULL,
  color4bar = gplots::col2hex(cfv),
  color4ns = "gray75",
  plotnames = TRUE,
  main = 'Important Contributions Variables. Dim 2.',
  ylab = "Signed Contributions")
#_____________________________________________________________________
# Bootstraps
BRj <- mfBADA.inf$Inference.Data$boot.data$fj.boot.data$tests$boot.ratios
#     Dimension 1
d001.plotBRj.1 <- PrettyBarPlot2(
  bootratio = BRj[,1],
  threshold = 2,
  ylim = NULL,
  color4bar = gplots::col2hex(cfv),
  color4ns = "gray75",
  plotnames = TRUE,
  main = 'Bootstrap Ratios Variables. Dim 1.',
  ylab = "Bootstrap Ratios")
#     Dimension 2
d003.plotBRj.2 <- PrettyBarPlot2(
  bootratio = BRj[,2],
  threshold = 2,
  ylim = NULL,
  color4bar = gplots::col2hex(cfv),
  color4ns = "gray75",
  plotnames = TRUE,
  main = 'Bootstrap Ratios Variables. Dim 2.',
  ylab = "Bootstrap Ratios")
```



8.5 Evaluation

The two tables below show us how well the BADA model represents, or classifies, the data. The question each table answers is, for the fixed effects, "Can I classify the data that I have?", and for the random effects, "Can I classify new data?".In a perfect world, the cell at the intersection of each genre, actual and predicted cells would show 100% of the audio files in that cell. The top table is the fixed effects model, and has a descriptive accuracy of 26.4%, which is very low, even considering that chance for this many groups is 10%. This is probably because the process of classifying music by genre is very difficult. Looking at the tolerance intervals for the groups best represents the overlap between the groups and the inherent difficulty of this task. The bottom table is the Random Effects (leave one out) Model, and it has an accuracy of 25.4%. This is also pretty terrible.

What sticks out to me here is how many the predicted observations there are for Country, Disco, Reggae, and Rock. Looking back at the factor map, we see that these genres are the ones with means that are closest to the barycenter, so it makes sense that the observations that cluster there are reflected in that. It's also remarkable how few Classical, Metal, and Pop tunes were accurately predicted. This makes sense in light of how far from the barycenter their means are relative to the other genres.

	.blues	.classical	.country	.disco	.hiphop	.jazz	.metal	.pop	.reggae	.rock	mffrowsums
.blues	12	0	3	0	0	2	0	0	1	3	21
.classical	0	3	0	0	0	0	0	0	0	0	3
.country	20	13	44	11	17	29	2	16	16	22	190
.disco	7	1	1	38	20	2	52	7	9	10	147
.hiphop	0	3	9	10	43	7	3	59	14	1	149
.jazz	10	55	3	1	0	27	0	1	1	3	101
.metal	0	0	0	0	0	0	3	0	0	0	3
.pop	0	0	0	0	0	0	0	1	0	0	1
.reggae	13	21	15	11	12	19	2	15	46	14	168
.rock	38	4	25	29	8	14	38	1	13	47	217

	.blues.actual	.classical.actual	.country.actual	.disco.actual	.hiphop.actual	.jazz.actual	.metal.actual	.pop.actual	.reggae.actual	.rock.actual	mfrrowsums
.blues.predicted	11	0	3	0	0	3	0	0	1	3	21
.classical.predicted	0	3	0	0	0	0	0	0	0	0	3
.country.predicted	21	13	43	11	18	29	2	16	16	22	191
.disco.predicted	7	1	1	37	22	2	53	7	9	10	149
.hiphop.predicted	0	3	9	10	39	7	3	59	14	1	145
.jazz.predicted	10	55	3	1	0	26	0	1	1	3	100
.metal.predicted	0	0	0	0	0	0	2	0	0	0	2
.pop.predicted	0	0	0	0	0	0	0	1	0	0	1
.reggae.predicted	13	21	15	12	13	19	2	15	46	15	171
.rock.predicted	38	4	26	29	8	14	38	1	13	46	217

8.6 Conclusions

- **Groups**
  - Groups have a lot of overlap and it's difficult to distinguish them.
  - Music Genre identification depends on a number of factors that, although hypothetically quantifiable, and therefore perceptible to a computer, are not present in this dataset.
- **Component 1**
  - Not actually significant. Makes sense given the groups and how close they are and how much they overlap. Also, considering that we're looking at multiple measures of the same thing (MFCCs) it looks like BADA has picked out that they don't accurately distinguish between genres.
- **Component 2**
  - Significant: separates weight of fundamental tone of audio from other MFCCs and shows how MFCC2 is anti-correlated with other spectral components.
- **Interpretation**
  - MFCCs have been driving the PC's to this point, but BADA has pulled that out of the model. Although it's the strongest source of variance, it doesn't differentiate significantly enough between the groups of observations.

  - As I said in the introduction, being able to differentiate between groups or not being able to differentiate between groups are equally valuable pieces of information. One of the things that we see here is that the model doesn't predict well what group the observations should belong to. What this tells us is that the process of quantifying to which genre a musical excerpt or sample belongs is a difficult process and involves processing information that the we don't have in this dataset. This information could be things like language, specific scale features, and the way in which harmony is used.

Chapter 9

Discriminant Correspondence Analysis

9.1 Intro to DiCA

DiCA is an extension of what we saw in MCA and CA, in that it allows us to decompose some of the relationships between multiple categorical variables, or levels of quantitative variables. It's also similar to what we did in BADA, in that one of the goals is to group the observations.
Like the CA, we'll need something similar to a contingency table. In the analysis below, R treats each level of our variables as a single variable. Obviously with 1000 possible values for each variable, this would make our dataset and its analysis prohibitively large. In order to do this, we'll bin the variables and run the DiCA on an indicator matrix. An indicator matrix is a complete disjunctly coded matrix with observations being represented by one and only one '1' for any level of a variable. There's more on that later in Binning and the Complete Disjunct Table. While the overall output with regard to factor scores for the variables looks similar to CA, it's necessary to make some adjustments in our interpretation. For more on DiCA, see Hervé Abdi (2007a). Once the variables are binned, a correspondence analysis is performed on the groups by binned variables matrix, and the group means and the variables are plotted in the new factor space. Just like BADA, we then project the original observations and classify the observations into groups based on the nearest barycenter.

9.1.1 Strengths & Weaknesses

Strengths
- This is a useful combination technique that combines some of the most useful things about BADA and CA.
- Because you can compare group means and variables in the same space, it provides an intuitive understanding of what levels of which variables are loading in what ways on the factor space and how that affects group separation.
- Great for breaking down variables into bins, so if you have data or factor plots from a PCA that are non-linear, you can see how the various levels of the variables load onto the factor space.
- Pre-processing (histograms, etc.) force you to take a close look at your data.
- It's great to be able to see which variables are loading on which dimensions, which ones are not, and why.

Weaknesses
- Plots can be visually very busy and require some time to construct and interpret. May require extra steps to create additional plots with things broken down.
- Similar issue to BADA regarding group differentiability. If the groups have differentiable means but lots of overlap for the observations, it may not be a good tool for classification. However, how well the groups are differentiable alone can also tell you something about the groups.

9.1.2 Dos and Don'ts

Do:
- Remember that this is a combination of techniques and offers some of the strengths and drawbacks of both. - Remember that the accuracy of the confusion matrix is relative to how many groups you have. If you have two groups, 50% is chance. If you have three groups, 33.3% is chance, and so on.
- Remember that although group means represent our best estimate for a given group, tolerance intervals show you how much the observations for the groups really overlap.

Don't:
- Forget to consider both techniques when interpreting the results of this analysis.
- Fool yourself into thinking that your results are truly generalizable. When you're evaluating your fixed and random effects, you're still only generalizing within your sample. Things like demographic data, which won't have changed between your full sample and your leave-one-out model, will still need to inform your interpretation.

Research Questions
Questions for DiCA can focus on two separate major concepts: how bins quantitative variables, or levels of qualitative/nominal variables load on the principal axes, and how the groups differ relative to those loadings.
- How do different levels of variables combine to create the group means?
- How do different levels of variables separate the group means along the two principal axes?
- What combinations of variables and their levels combine to create a typical observation from a certain group?
- How are we separating groups across the extracted principal components?
- Are these group differentiable enough for accurate classification of observations?
- What can the confusion matrix tell us about the distribution of our observations?

9.2 Data

We're using the music features dataset for this analysis, the same as the one used for the PCA. As stated above, we'll need an indicator matrix to run the MCA on, which means we'll need to do some editing of our data to make it work. Below is our original data. I've pulled out one of each of the first five genres in the table to show what the variables/measures and the values/observations look like in the interest of saving space, only 8 variables are shown. Call `head(mfdata)` to see more.

	ch	rmse	spec_c	spec_b	r_o	zcr	mfcc1	mfcc2
blues.00081.au	0.3802602	0.2482623	2116.943	1956.611	4196.108	0.1272725	-26.92978	107.33401
classical.00091.au	0.2237383	0.0447457	2192.798	1911.986	4066.467	0.1524379	-251.31345	88.72515
country.00013.au	0.3680927	0.0800457	2812.346	2842.053	6062.663	0.1295650	-126.16004	77.63675
disco.00063.au	0.5478499	0.2939197	2583.278	2626.311	5855.473	0.0997726	-51.75267	70.33190
hiphop.00034.au	0.4699237	0.2279124	3191.045	2735.905	6353.995	0.1691660	5.87653	53.23913

9.2.1 Data Preparation

Because an DiCA evaluates a table of disjuctively coded data, we need to take a couple of steps to make sure the data is prepared to run the MCA. We do the following:

1. Evaluation
2. Binning
3. Spearman Rank testing
4. Nominalizing

At the end of this, you should end up with a dataset with the same number of rows, but more columns, because of the binning and nominalizing. If you're lucky enough to be able to use the same number of bins for each variable, it'll be the number of bins times the number of variables. For this dataset, that would be 4, (the default for the function we use below) times 28, or 112 columns.

9.2.1.1 Evaluation

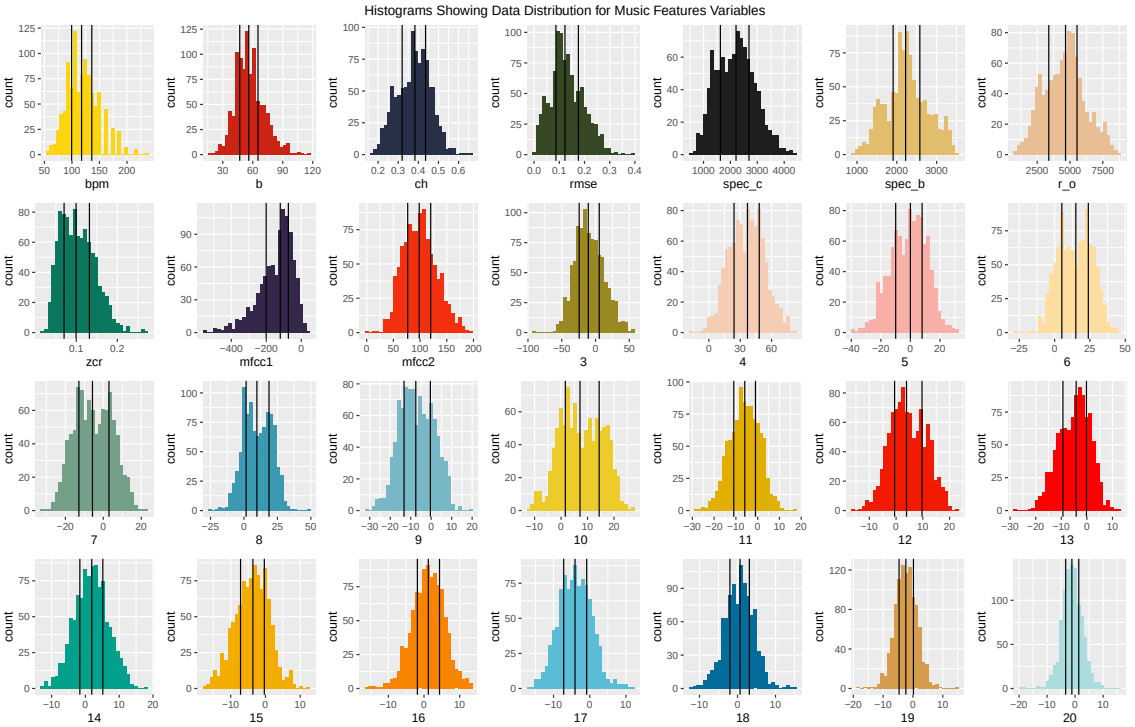
This chunk gets us colors for the variables we'll be using throughout. `cfv` stands for color for variables and `cfb` stands for color for bootstraps. You'll see why `cfb` is four of `cfv` later.

```
cfv <- unique(c(wes_palettes$BottleRocket2, wes_palettes$Rushmore1,
                wes_palettes$Royal2, wes_palettes$Zissou1,
                wes_palettes$Darjeeling1,
                wes_palettes$Darjeeling2[2:4]))
cfb <- matrix(nrow = 28, ncol = 4)
cfb[,1:4] <- as.matrix(cfv)
```

First thing we do before we bin is check out our data a little more closely, looking at range, mins and maxes, and distribution of our data. To that end, `summary()` shows us a summary of the data, and `str()` shows us the structure. The histograms below show us how each of the variables is distributed. We're looking for anything out of the ordinary that would make us want to bin our data in a special way. One effective way we can visualize our data is to check out histograms of each of the variables.
Sample code to create the histograms:

```
h1  <- ggplot(data = mfdata, aes(x = bpm)) +  geom_histogram(bins = 30, fill = cfv[1]) +
        geom_vline(xintercept = quantile(mfdata$bpm)[2:4])
```

The majority of the variables look like they have an approximately normal distribution, but you should always check to see if any look like they might have a bimodal distribution or some kurtosis or skewness.If you have a variable that has a different kind of distribution, like a Poisson distribution or a lomax distribution, you may need to do some extra pre-processing before binning or analysis. In the example we have here, the zcr variable looks slightly skewed towards the left, which suggests that the majority of the audio files contain large amounts of percussion, which makes sense given our genres. It also makes sense given our factor map, which shows that ZCR is negatively correlated with the Classical genre, which may be more likely to contain only string, brass, or woodwind instruments, and positively correlated with Metal and Hip-Hop, which tend to feature percussive sounds more prominently and consistently.



9.2.1.2 Binning and the Complete Disjunct Table

This chunk is actually steps 2, 3, and 4 listed above.

Because of the size of the dataset (1000 observations), I don't really see anything that makes me think I should specify any specific binnings for these data, so we're going to go with the default for `BinQuant()`, which is four.

First thing I'm going to do is create a dataframe to put our binned variables in. Then I'm going to run a for loop binning the variables, and then another loop over those data using the Spearman Rank Correlation to test how well they correlate with the original data. The function we use for the Spearman rank correlation is `cor(x, y, method = "spearman")`. This correlation is only helpful when the variable is linear. If there's a nonlinear relationship, the Spearman rank correlation isn't a helpful piece of information. That's why it's important to look at the histograms.

Finally, I'm going to take the bins and make them nominal. This creates as many variables as there are levels of all the variables. For this dataset, I used four bins for each variable, so I ended up with four variables for each original variable (e.g. bpm.1, bpm.2, bpm.3, bpm.4). As stated in the introduction, each observation (audio file/song) has one and only one '1' for each of the original variables, corresponding to the bin (quartile) in which the value for that observation fell.

9.2.1.3 Distributions Table

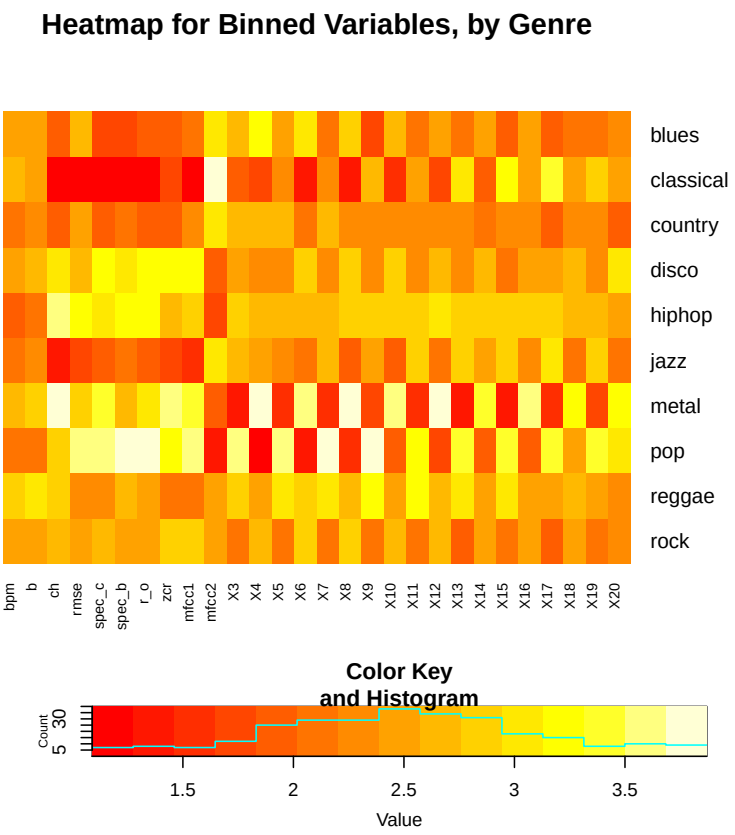
The table we get from the output below shows us what the ranges of each of the bins are, how many of each variable are binned in each bin, and how accurately these bins align with the original data. Below are the first six variables. In a perfect world, we'd see 250 observations in each variable, which we very nearly do. Things won't always be this neat.

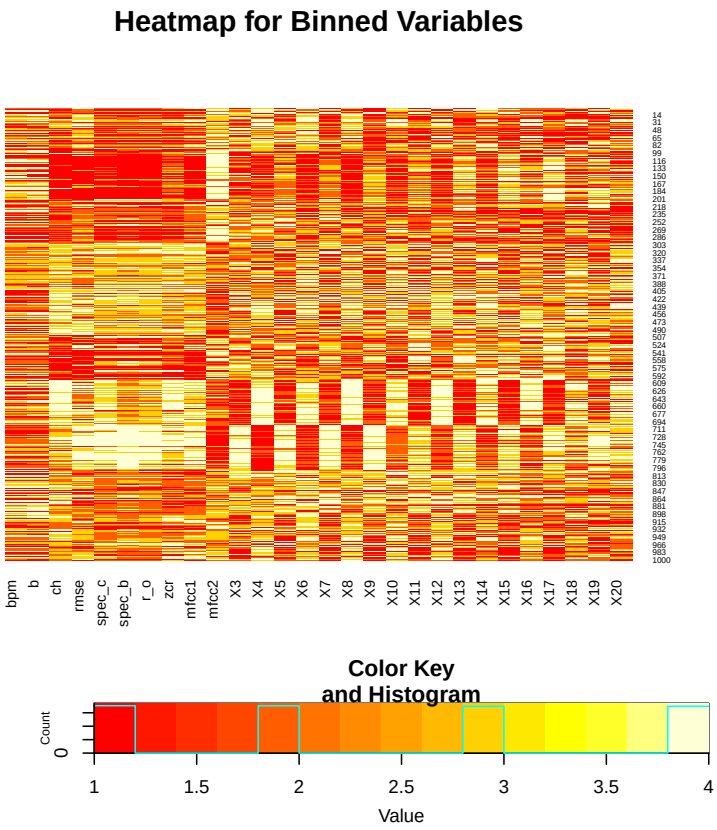
9.2.2 Processed Data Visualization: Heat Map

There are two heatmaps below that show us the what our data looks like post-binning. The first shows us what variables look like by genre (group means) and the second shows us what each of the individual

observations look like. One thing that we can take away from these maps, above and beyond how the data is distributed, is how the groups seem to really highlight the differences between the groups by smoothing out the variation within the groups. There are some pretty clear distinctions between genres and between spectral markers, although some of the spectral markers do seem to be moving in blocks. There are clear differences between the 600 and 700 groups, which are the rows of metal and pop. Likewise, MFCCs are very clearly alternating high and low values by odds and evens.

```
mfbin4heatmap <- lapply(mfbin, as.numeric)
gm4hm <- getMeans(mfbin4heatmap, music.genre)
# Sets layout parameters for the heatmap
lmat = rbind(c(0,3,0),c(2,1,0),c(0,4,0))
lwid = c(.5,3,.5)
lhei = c(1,4,1.5)
# Heatmap for group means
mfbinheatmap <- heatmap.2(as.matrix(data.frame(lapply(gm4hm, as.numeric))),
                          Rowv = NA, Colv = NA,
                         dendrogram = 'none', trace = 'none',
                         lmat = lmat, lhei = lhei, lwid = lwid,
                          main = "Heatmap for Binned Variables, by Genre",
                         labRow = unique(music.genre))
# Heatmap for individual observations
mfbinheatmap.m <- heatmap.2(as.matrix(data.frame(lapply(mfbin4heatmap, as.numeric))),
                            Rowv = NA, Colv = NA,  dendrogram = 'none', trace = 'none',
                            lmat = lmat, lhei = lhei, lwid = lwid,
                            main = "Heatmap for Binned Variables")
```

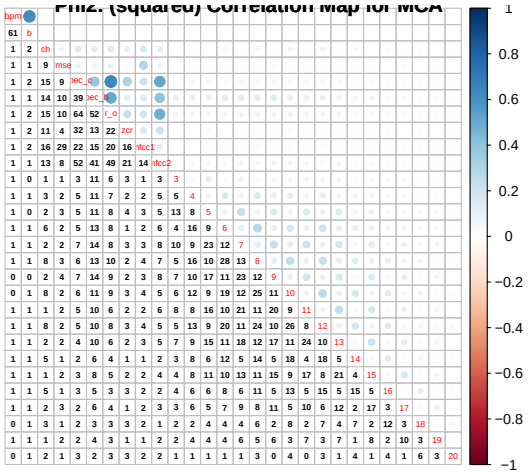




9.2.3 Phi^2 & Burt Table

The next step for DiCA is just like MCA, we compute a Burt table, which is computed by multiplying our nominal (complete disjunct) table by its transpose to get a contingency table. For this we use the function `phi2mat4BurtTable` from the `data4PCCAR` package. The function gives us two outputs: the phi^2 matrix and the Burt table. The Burt table is a contingency table for all of our nominalized variables, and the phi^2 table shows us the squared correlation between the variables. We can then visualize this using correlation plots. The plot below below shows the phi^2 (left) correlation.

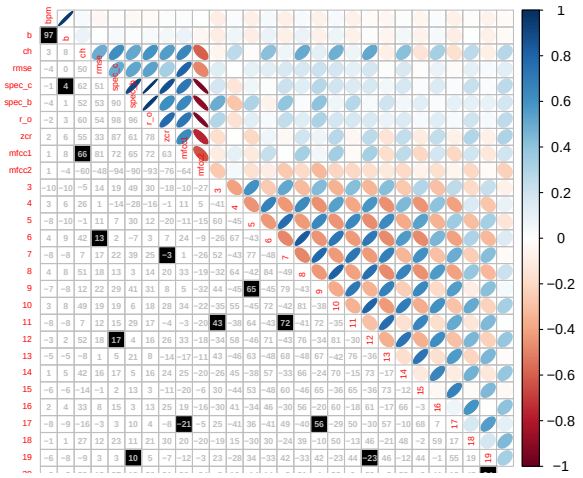
```
# Pseudo Heat Map ----
corrMatBurt.list <- phi2Mat4BurtTable(mfnom, make_data_nominal = FALSE)

corr4MCA <- corrplot.mixed(as.matrix(corrMatBurt.list$phi2.mat),
                           lower.col = "black", tl.cex = .5,
                           title = "Phi2: (squared) Correlation Map for MCA",
                           addCoefasPercent = TRUE, number.cex = .5)
```



9.2.4 Correlation Plots

The standard correlation plot below shows the correlations visually (top) and numerically (bottom). The variables are ordered according to their loadings on the first component. This makes sense given our results from the PCA, namely that the even MFCCs greater than 2 load positively on the first component and the odd MFCCs greater than 1 load negatively on the first component. It makes sense that these are opposed to one another as they are intentionally decorrelated. Likewise, the variables that load more heavily on the second component (or, in the case of b and bpm, the 3rd or 4th component) are more central in the plot.

```
mfcor <- cor(mfdata)

corrplot(mfcor, diag = F, type = "upper", method = "ellipse",
         tl.cex = .5, tl.pos = "n") %>%
corrplot(mfcor, add = TRUE, diag = F, type = "lower",
         method = "number", addCoefasPercent = T,
         col = "grey", tl.cex = .5, number.cex = .5, tl.pos = "ld")
```



9.3 Analysis

The code below shows us the DiCA and the DiCA inference batteries, respectively.

```
mfdica <- TExPosition::tepDICA(mfnom,
                               make_data_nominal = FALSE,
```

```
                                  DESIGN = music.genre,
                                  graphs = FALSE)
```

```
mfdica.inf <- TInPosition::tepDICA.inference.battery(DATA = mfnom,
                                                     make_data_nominal = FALSE,
                                   DESIGN = music.genre, make_design_nominal = TRUE,
                                   graphs =  FALSE # TRUE first pass only
                                   )
```

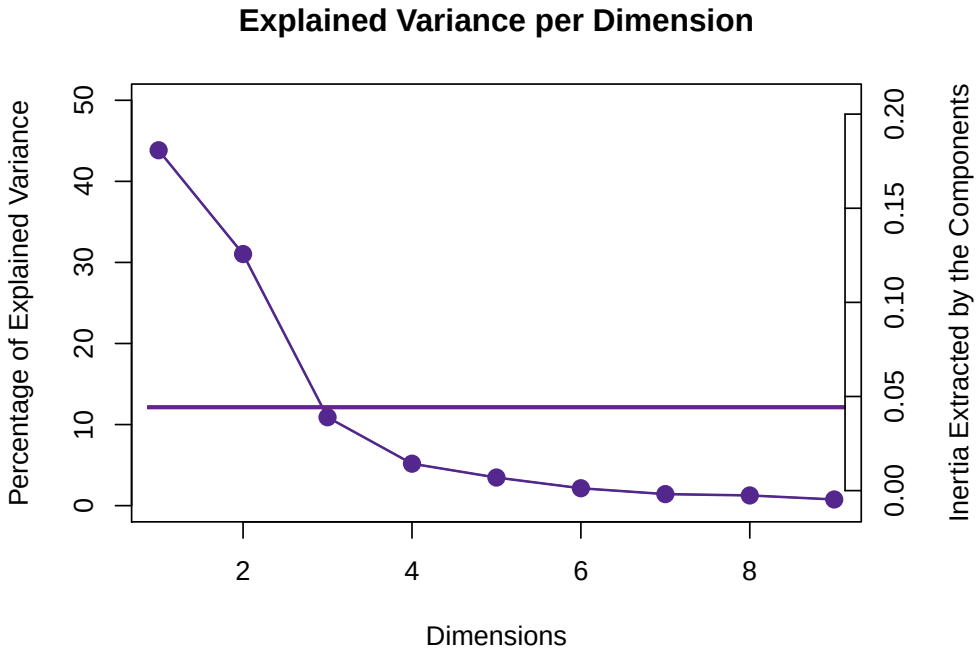
```
## [1] "It is estimated that your iterations will take 0.35 minutes."
## [1] "R is not in interactive() mode. Resample-based tests will be conducted. Please take note o
## ============================================================================
```

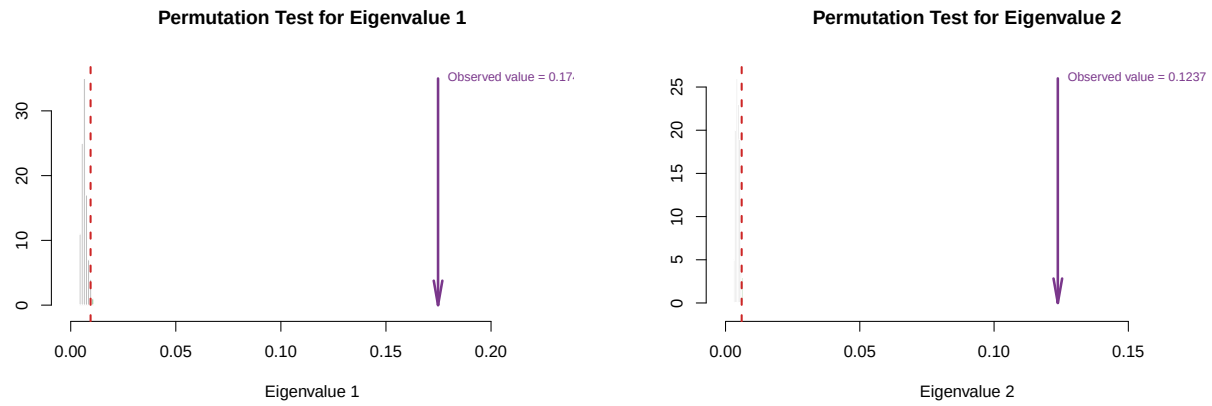
9.4 Results

9.4.1 Scree Plot

The scree plot shows us which of the eigenvalues/dimensions of variance extracted are significant. The dimensions here (9) are one less than the number of groups. The p values associated with the eigenvalues show us which values (via permutation testing in `tepDICA.inference.battery`) are significant. We have 9 out of 9 significant dimensions of variance extracted. The Kaiser criterion shows us that the average value for the eigenvalues is approximately 11, but that based on permutation testing, each of the eigenvalues for the data are significant, even though they fall below that average. The plots below, titled "Permutation Test for Eigenvalue" show how far removed from the random distribution the first two observed values are. More on reading scree plots and their permutations can be found here

```
PlotScree(ev = mfdica$TExPosition.Data$eigs,
          p.ev = mfdica.inf$Inference.Data$components$p.vals,
          plotKaiser = TRUE)
```



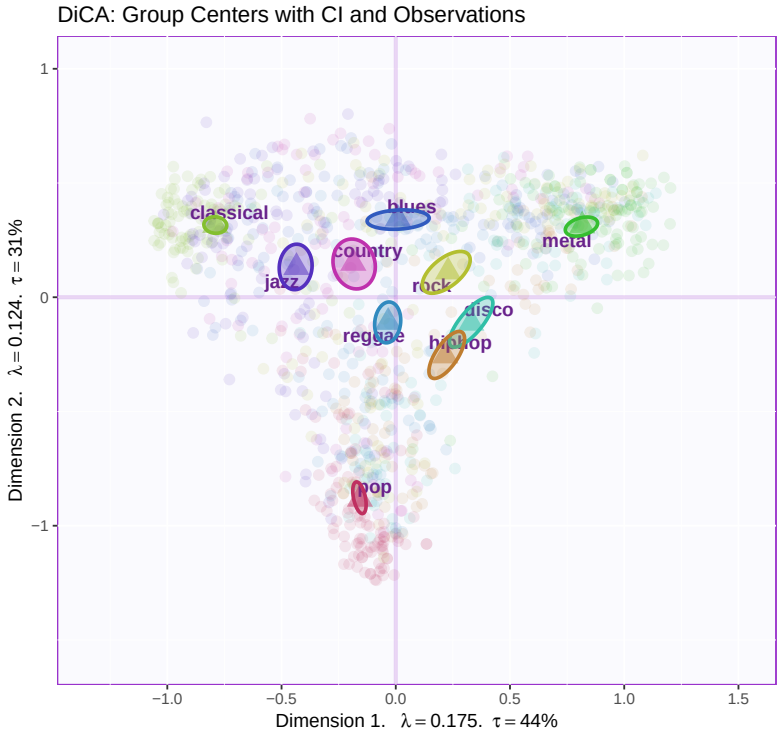


9.4.2 Factor Maps

Next we need to visualize all of our data on the principal components space. The first thing we're going to do is visualize all of our observations, including their means and confidence intervals for their means as we have done before. This will help us to understand how the observations map onto the variables and what patterns we can discern from that. If you've seen the cookbooks on PCA, Inference PCA, and CA, this should look familiar, but binning has made some slight changes to the appearance of our plots.

```
# Create a factor map for the individual observations
mf.Imap <- createFactorMap(title = 'DiCA: Group Centers with CI and Observations',
                           mfdica$TExPosition.Data$fii,
                           col.points = mfdica$Plotting.Data$fii.col,
                           display.labels = F,
                           alpha.points = .1)
# This helps us clean up the group names by replacing the punctuation in the rownames
# with nothing, hence the "". The effect is the deletion of the punctuation.
rownames(mfdica$TExPosition.Data$fi) <- sub("[[:punct:]]","",
                                            rownames(mfdica$TExPosition.Data$fi))
# Create a factor map for the group means of observations
mf.gmap <- createFactorMap(mfdica$TExPosition.Data$fi,
                           col.points = mfdica$Plotting.Data$fi.col,
                           distplay.labels = T,
                           pch = 17,
                           alpha.points = .5,
                           cex = 5)
# Create labels
label4Map <- createxyLabels.gen(1,2,
                    lambda = mfdica$TExPosition.Data$eigs,
                    tau = mfdica$TExPosition.Data$t)
# Put together the observations map, the labels, the group means,
#       and the labels for the group means
a002.Map.I <- mf.Imap$zeMap + label4Map + mf.gmap$zeMap_dots + mf.gmap$zeMap_text
# Create confidence intervals for the group means (from the PTCA4CATA Package)
GraphElli <- MakeCIEllipses(
            mfdica.inf$Inference.Data$boot.data$fi.boot.data$boots[,c(1,2),],
                           col = unique(mfdica$Plotting.Data$fi.col),
                           p.level = .95)
# Puts together the group means map, the CI ellipses, and the labels for the axes
```

```
# Note that because you're using the mf.gmap map, you don't get points for
# the individual observations.
a003.Map.gm <- mf.gmap$zeMap + GraphElli + label4Map
# Puts together the everything from a002, plus the ci ellipses
a004.dica.ci <- a002.Map.I + GraphElli
a004.dica.ci
```

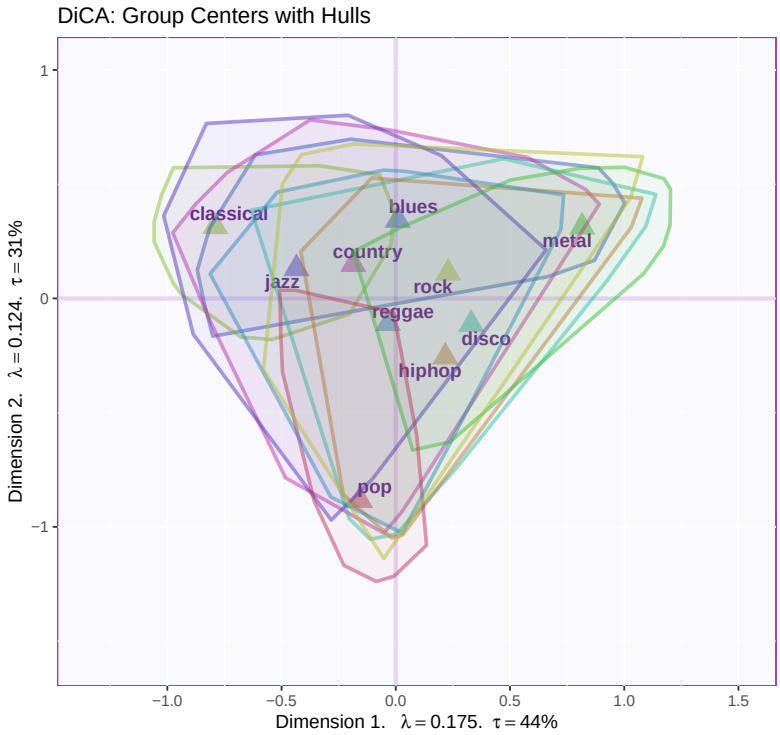


9.4.2.1 Tolerance Intervals

The tolerance intervals surround all of the datapoints in the data. They show you explicitly what the boundaries of the distributions of your data are. This is useful especially in this example and in BADA for comparing to our group barycenters. If the dispersion is wide and the group mean CI is tight, then it's likely that the tightness of that confidence interval is due to the size of the data, the number of observations. That's the case in this example, where we have the observations for these groups pretty consistently overlapped, even though the means and confidence intervals are not. That suggests that we may have issues in differentiating between groups.

```
# Gets values for the observations
Fii <- mfdica$TExPosition.Data$fii
# Adds 'Dimension' to each column of the observations
colnames(Fii) <- paste0('Dimension ', 1:ncol(Fii))
# Pulls the colors for the hulls from the fii plotting data
col4Hull <- unique(mfdica$Plotting.Data$fii.col)
GraphHull <- PTCA4CATA::MakeToleranceIntervals(
                                mfdica.inf$Fixed.Data$TExPosition.Data$fii,
                                design = as.factor(music.genre),
                                col = col4Hull,
                                # the next line is required
                                # for some strange unknown reasons
```

```
                                names.of.factors =  c("Dim1","Dim2"),
                                p.level = 1.00,
                                alpha.ellipse = .05)
# Puts all of the group dots on the map with labels and hulls
a006.dica.withHull <-  mf.Imap$zeMap_background + label4Map +
                       mf.gmap$zeMap_dots + mf.gmap$zeMap_text +
                       GraphHull + ggtitle('DiCA: Group Centers with Hulls')
a006.dica.withHull
```



9.4.3 Disjunct Factor Maps

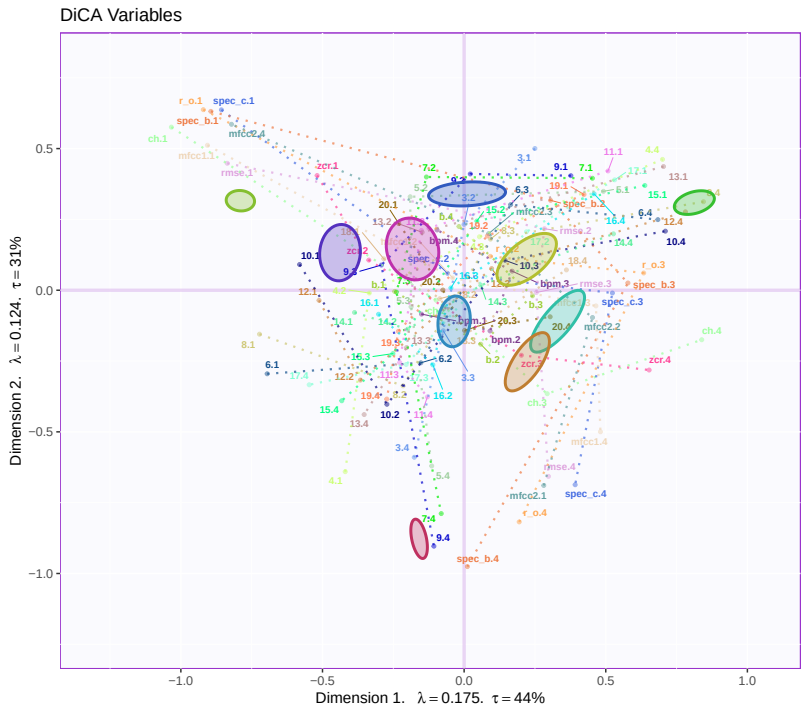
These factor maps show us how the observations map onto the factor space. Because we're using disjunctively coded data, we can also see what levels of those variables map onto which areas of the factor space. There are four maps below, each of which show certain variables. The first map shows all of the variables, the second shows just the variables that load significantly on the first dimension, the third shows the variables that load significantly on the second dimension, and the fourth shows the variables that don't load significantly on either one of those variables. I did this by specifying which of the rows of `mfMCA$ExPosition.Data$fj` I wanted to display on each graph. I selected them after looking at the contribution plots (below) to see which variables contributed what to which dimensions.

These are plotted with the group mean CI's to give an idea of how the variables and the observations are related.

First we pull set up the colors and make the original map.

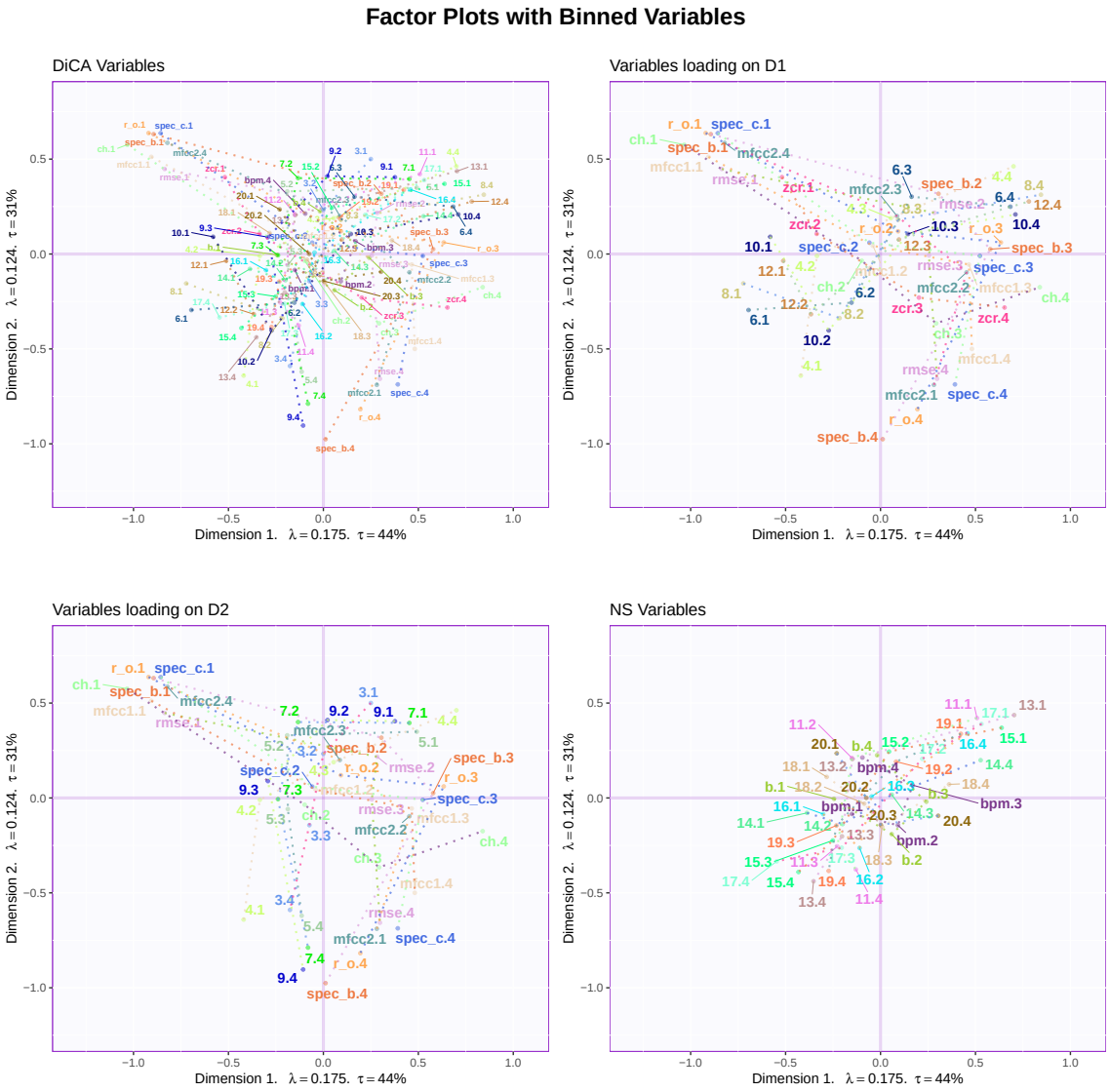
```
col4Var <- c('orange','orange4','red','red4')
# Levels
col4Levels <- coloringLevels(rownames(mfdica$TExPosition.Data$fj), col4Var)
# To save typing, we soft-code the axes we want so we can change it
# for all of the plots at once if we want to look at other dimensions
```

```
axis1 = 1
axis2 = 2
# Assign the FJs
Fj.all <- mfdica$TExPosition.Data$fj
#Creates the Map
BaseMap.Fj.all <- createFactorMap(X = Fj.all, axis1 = axis1, axis2 = axis2,
                                  title = 'DiCA Variables',
                                  col.points = col4Levels$color4Levels,
                                  cex = 1, text.cex = 2.5, force = 2,
                                  col.labels = col4Levels$color4Levels,
                                  )
#Assigns all the map parts
b001.BaseMap.Fj.all <- BaseMap.Fj.all$zeMap + label4Map
#Assigns the map so that there are no dots for the binned variables factor scores
b002.BaseMapNoDot.Fj.all  <- BaseMap.Fj.all$zeMap_background +
                             BaseMap.Fj.all$zeMap_text + label4Map + GraphElli
# creates the lines for the DiCA
# notice we're using the same code as we did for MCA
lines4J.all <- addLines4MCA(Fj.all, col4Var = col4Var, size = .75)
# Adds the lines to the plots with mean CIs and without, respectively
b003.MapJ.allwm <-  b001.BaseMap.Fj.all + lines4J.all  + GraphElli
b003.MapJ.all <-  b001.BaseMap.Fj.all + lines4J.all
# Just calls the map with all the bits
b003.MapJ.allwm
```



But this is incredibly busy, so we break it down by pulling the variables that load on each dimension and that don't load at all, which we've gotten from our loadings barplots below, and putting those each on

separate plots. Note that for each of these plots, the constraints are assigned to be the same as the original (all variables) factor plot, so we can easily and intuitively compare the plots to one another.



9.5 Contributions

The contributions are showing us simply how much each variable loads, irrespective of the sign of that loading. This is because for this type of analysis, we're actually going to break down how much each component loads with the binning we did above, and we see those in the bootstrap ratios. The variables that don't load significantly on either variable are Beats, BPM, and MFCCs 3, 16, 18, 19, and 20. Note that 16 is just shy of achieving significance on component 1.

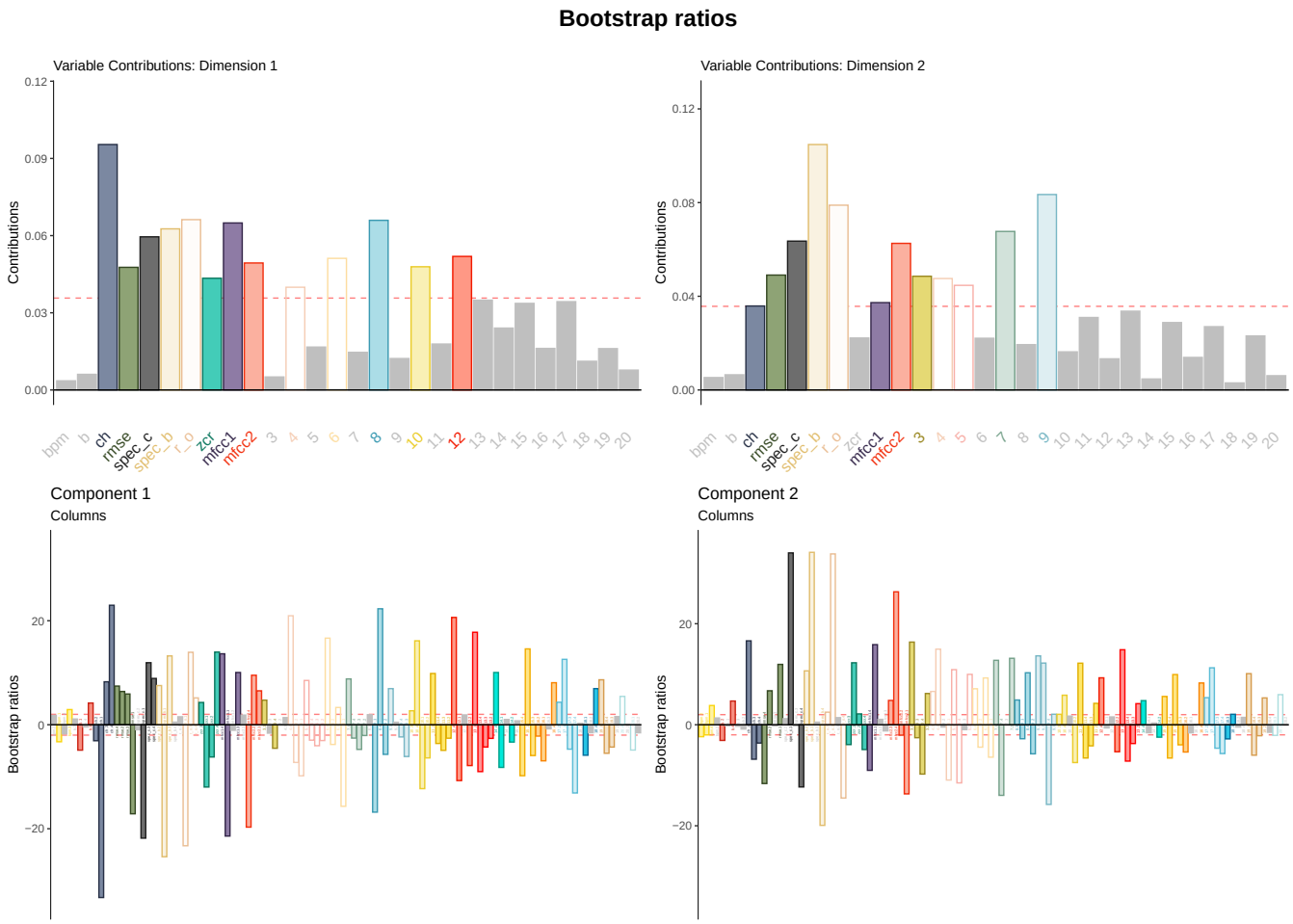
The contributions for each dimension, as well as the bootstrap ratios for each of the disjunctively coded variables. The bootstrap ratios on top show us how much consistently each bin of each variable loads on each component, and in what direction. It's easy to see the differences between the spectral components and the MFCCs in this, with the MFCCs driving component 1 and spectral components loading on variable 2. You can also see if you look closely that most of the variables load in order - the first bin loads the most positively while the fourth bin loads most negatively (or vice versa) and the middle two bins fall somewhere in between. Sometimes they don't, which happens in the case where some variables load in a non-linear

fashion. Beats and BPM are examples.

The labels for each of the variables on the bottom plot are small, but visible if you zoom in. If you're looking at this on a webpage, open in a new tab and zoom in. If you're looking at a PDF, just zoom in.

```
# contributions for variables, without their signs
mfcont <- ctr4Variables(mfdica$TExPosition.Data$cj)
# plot contributions of columns for component 1
varCtr1 <- mfcont[,1]
names(varCtr1) <-rownames(mfcont)
ctrJ.1 <- PrettyBarPlot2(varCtr1,
                         ylim =c(0, 1.2*max(varCtr1)),
                         ylab = 'Contributions',
                         font.size = 4,
                         threshold = 1/ nrow(mfcont),
                         color4bar = cfv,
                         angle.text = 45
) + ggtitle("", subtitle = 'Variable Contributions: Dimension 1')

# plot contributions of columns for component 2
varCtr2 <- mfcont[,2]
names(varCtr2) <- rownames(mfcont)
ctrJ.2 <- PrettyBarPlot2(varCtr2,
                         threshold = 1 / NROW(varCtr2),
                         font.size = 4,
                         color4bar = cfv, # we need hex code
                         ylab = 'Contributions',
                         ylim = c(0, 1.2*max(varCtr2)),
                         angle.text = 45,
) + ggtitle("", subtitle = 'Variable Contributions: Dimension 2')
```

```
BR.J <- mfdica.inf$Inference.Data$boot.data$fj.boot.data$tests$boot.ratios
# Plot the bootstrap ratios for Dimension 1
laDim = 1
ba001.BR1 <- PrettyBarPlot2(BR.J[,laDim],
                        threshold = 2,
                        font.size = 1,
                    color4bar = t(cfb), # we need hex code
                   ylab = 'Bootstrap ratios', #sortValues = TRUE
                   #ylim = c(1.2*min(BR[,laDim]), 1.2*max(BR[,laDim]))
) + ggtitle(paste0('Component ', laDim), subtitle = 'Columns')

# Plot the bootstrap ratios for Dimension 2
laDim = 2
ba002.BR2 <- PrettyBarPlot2(BR.J[,laDim],
                        threshold = 2,
                        font.size = 1,
                    color4bar = t(cfb), # we need hex code
                   ylab = 'Bootstrap ratios', #sortValues = TRUE
                   #ylim = c(1.2*min(BR[,laDim]), 1.2*max(BR[,laDim]))
) + ggtitle(paste0('Component ', laDim), subtitle = 'Columns')
```



9.6 Model Evaluation

The confusion matrices below evaluate how accurate the model is in assigning the observations to the a priori groups. On top is the fixed model, which has an accuracy of 47.6%. On the bottom is the random effects (leave one out) model, which has an accuracy of 43%. This isn't great, but it's well above chance, which in this case is only 10% (100/10 groups). This is likely due to the fact that a) music genre identification is an incredibly complex task and b) looking at the group tolerance intervals, there's a whole lot of overlap between genres.

What sticks out to me here is how high the predicted observation values are for Country, Disco, Reggae, and Rock. Looking back at the factor map, we see that these genres are the ones with means that are closest to the barycenter, so it makes sense that the observations that cluster there are reflected in that. It's also remarkable how few Classical, Metal, and Pop tunes were accurately predicted. This makes sense in light of how far from the barycenter their means are relative to the other genres.

	.blues	.classical	.country	.disco	.hiphop	.jazz	.metal	.pop	.reggae	.rock	mfdicafrowsums
.blues	22	4	8	0	1	9	2	0	5	6	57
.classical	4	84	7	1	0	26	0	1	2	0	125
.country	10	2	38	3	2	6	0	3	5	14	83
.disco	2	0	0	23	3	1	1	1	6	5	42
.hiphop	0	0	1	6	31	1	2	3	10	3	57
.jazz	24	8	11	0	1	32	0	2	4	5	87
.metal	20	0	3	23	23	1	83	0	1	18	172
.pop	0	0	13	26	24	12	1	81	11	14	182
.reggae	1	1	5	3	9	6	0	9	53	6	93
.rock	17	1	14	15	6	6	11	0	3	29	102

	.blues.actual	.classical.actual	.country.actual	.disco.actual	.hiphop.actual	.jazz.actual	.metal.actual	.pop.actual	.reggae.actual	.rock.actual	mfdicarrowsums
.blues.predicted	15	4	13	0	1	10	2	0	6	8	59
.classical.predicted	4	84	7	1	0	30	0	1	2	0	129
.country.predicted	12	2	28	3	2	7	0	3	5	15	77
.disco.predicted	3	0	0	19	4	1	1	2	8	5	43
.hiphop.predicted	0	0	1	8	22	1	2	4	10	3	51
.jazz.predicted	25	8	14	0	1	26	0	2	5	5	86
.metal.predicted	20	0	3	23	25	1	83	0	1	19	175
.pop.predicted	0	0	13	26	26	12	1	79	11	14	182
.reggae.predicted	1	1	7	3	13	6	0	9	49	6	95
.rock.predicted	20	1	14	17	6	6	11	0	3	25	103

9.7 Conclusions

One thing DiCA manages to do is disentangle how the levels (bins) of each of the variables are reflected in each of the groups. To visualize this more effectively, we can look at the factor maps with the means of the groups plotted as well.

- **Component 1**
  - This explains why the variables we saw before loaded the way they did:
  - On the positive side, we have all of the variables driving the horizontal component, but on the negative side, the variables also seem to load a bit on the second component.
  - We can also see how the Odd MFCCs have lower values on the negative end of component 1 and higher values on the positive side, while even MFCCs have lower values on the positive side of component 1 and higher values on the positive end.
- This does really help separate which of the genres tend to load in which way on these spectral components.
- **Component 2**
  - Component two really helps separate out how music recording techniques differ between genres.
  - The highest values of: spectral centroid, roll off, and spectral bandwidth are all more associated with pop than any other genre.
  - Likewise, the lowest values for every single component is more highly associated with classical than any other genre.
  - We can see how many genres cluster in the middle, suggesting that there aren't too many differences in their recording techniques.
- Interpretation: Although the interpretation for this dataset doesn't offer us any new information, it does look like it's telling us that this model (or at least this code) is more effective at discriminating between groups than our BADA model. Still not great, even though it's well above chance.

Chapter 10

Partial Least Squares Correlation

10.1 Intro to PLSC

This technique allows for the analysis of multiple separate data tables that contain distinct data. It's commonly used in brain imaging analysis using matrices of imaging data and matrices of behavioral, task, or other data. A full enumeration of the types and uses of PLS can be found in "Partial Least Squares (PLS) methods for neuroimaging: A tutorial and review" (Krishnan et al. 2011). Also check out Hervé Abdi and Williams (2013) and Hervé Abdi (2010).

In this technique, we will be looking at covariance and correlation between the variables in the two tables, and we use the techniques to extract "latent variables" (instead of the "principal components" of the PCA technique), and we look at the "saliences" of the variables (as opposed to "loadings").

The null hypothesis for the PLSC technique is that there is absolutely nothing that is similar between the two tables. One thing we know from experience, though, is that in a dataset large enough, you're going to find *something* that is similar, and therefore correlated. This has been shown a few times with people looking at just noise and finding significant eigenvalues to extract. This suggests that the technique is overpowered, so it's important to interpret the results of such an analysis carefully.

10.1.1 Strengths & Weaknesses

Strengths
- Maximizing covariance makes this technique really good at extracting similarities between tables. That's what makes it so good for analyzing brain imaging and behavioral data. If there's a signal, PLSC will find it.
- Another thing that makes it useful is the fact that it can handle datasets that are hundreds or thousands (or hundreds of thousands) of variables long. (Pancake data)
- If you're dealing with a dataset where a single variable drives an entire dimension, it may make sense to break the dataset up as I've done here (MFCC2 is the culprit), so that the first dimension can extract that variance and the second dimension can explain a little more about the other variables.

Weaknesses
- It may in fact be overpowered, so be careful when interpreting results.
- No inherent predictive power. For that, use the sister technique, PLSR.

10.1.2 Dos and Don'ts

Do:
- Remember what the null hypothesis for PLSC is when analyzing the data and interpreting the factor plots.

Table 10.1: X matrix

	spec_b	r_o	4	6	8	10	11	12	13
blues.00081.au	1956.611	4196.108	40.93263	24.8162749	15.2534595	12.262823	-15.2340499	14.336612	-13.8217688
classical.00091.au	1911.986	4066.467	37.91978	21.6378512	25.7257307	14.300963	-7.5613448	10.469501	-4.9211583
country.00013.au	2842.053	6062.663	19.95351	0.7278421	-0.4991002	5.391963	-3.7306883	-3.500606	-5.3320550
disco.00063.au	2626.311	5855.473	49.74851	35.7078097	24.7918958	22.044862	-6.1189723	17.185402	-2.7709661
hiphop.00034.au	2735.905	6353.995	26.93997	15.5495557	7.6611179	8.607721	3.4188215	8.532096	1.1353375
jazz.00004.au	1422.303	1838.214	42.47749	9.9179535	8.2343280	5.467933	-5.4344488	-5.186267	-6.9909234
metal.00088.au	1986.989	4148.192	68.27110	22.6089120	17.0547311	13.668835	-2.8086952	5.377908	-3.0042006
pop.00024.au	3175.657	7302.598	10.09793	5.3679174	3.5069437	5.883242	-0.1018416	-0.568772	0.5524485
reggae.00009.au	1990.699	3269.547	18.37370	22.9268087	16.7547936	15.531519	-8.3772216	8.026785	-3.7762516
rock.00019.au	2118.985	4702.063	47.02087	26.2307412	18.7908188	11.045203	-19.2316279	5.228019	-16.7577640

Table 10.2: Y matrix

	ch	rmse	spec_c	zcr	mfcc1	mfcc2	3	5	7	9
blues.00081.au	0.3802602	0.2482623	2116.943	0.1272725	-26.92978	107.33401	-46.809993	-21.4637758	-18.945571	-15.050104
classical.00091.au	0.2237383	0.0447457	2192.798	0.1524379	-251.31345	88.72515	-41.767192	-12.0239733	-5.854145	-5.196483
country.00013.au	0.3680927	0.0800457	2812.346	0.1295650	-126.16004	77.63675	3.165314	10.8347912	5.216033	-1.002268
disco.00063.au	0.5478499	0.2939197	2583.278	0.0997726	-51.75267	70.33190	-3.919614	1.5853422	-4.096545	-9.710883
hiphop.00034.au	0.4699237	0.2279124	3191.045	0.1691660	5.87653	53.23913	-8.983690	0.4364055	1.650808	4.983282
jazz.00004.au	0.1717823	0.1087860	1039.623	0.0477688	-270.26482	137.57173	5.658533	-7.9488311	-6.799091	-6.188287
metal.00088.au	0.4215711	0.1333055	1988.120	0.0924201	-129.70020	104.03440	-26.972440	2.4353629	-12.100199	-5.673693
pop.00024.au	0.4109899	0.2075838	3301.782	0.1445267	-40.80207	61.52205	9.868217	8.4562927	8.377963	8.253806
reggae.00009.au	0.3830216	0.0866705	1609.641	0.0605760	-232.62434	131.23940	2.697012	11.3516613	-10.464260	-15.247327
rock.00019.au	0.4359108	0.1122276	2321.186	0.1281995	-84.17958	102.33721	-41.332401	-16.9485922	-18.950190	-21.683568

- Remember that the factor plots represent the maximum covariance extracted on the first two dimensions of both matrices, and what the variables are in each table that may be driving the plot.

Don't:
- Get lost in the sauce. Just because something is important, doesn't mean it's significant.

Research Questions
Questions for this technique should focus on how the sets of variables are similar and different, and how they're distinguishing groups of observations along the latent variables. If you consider that this technique is commonly used for correlating brain imaging data and behavioral data, finding the proverbial needle in the haystack, that helps in terms of clarification.

10.2 Data

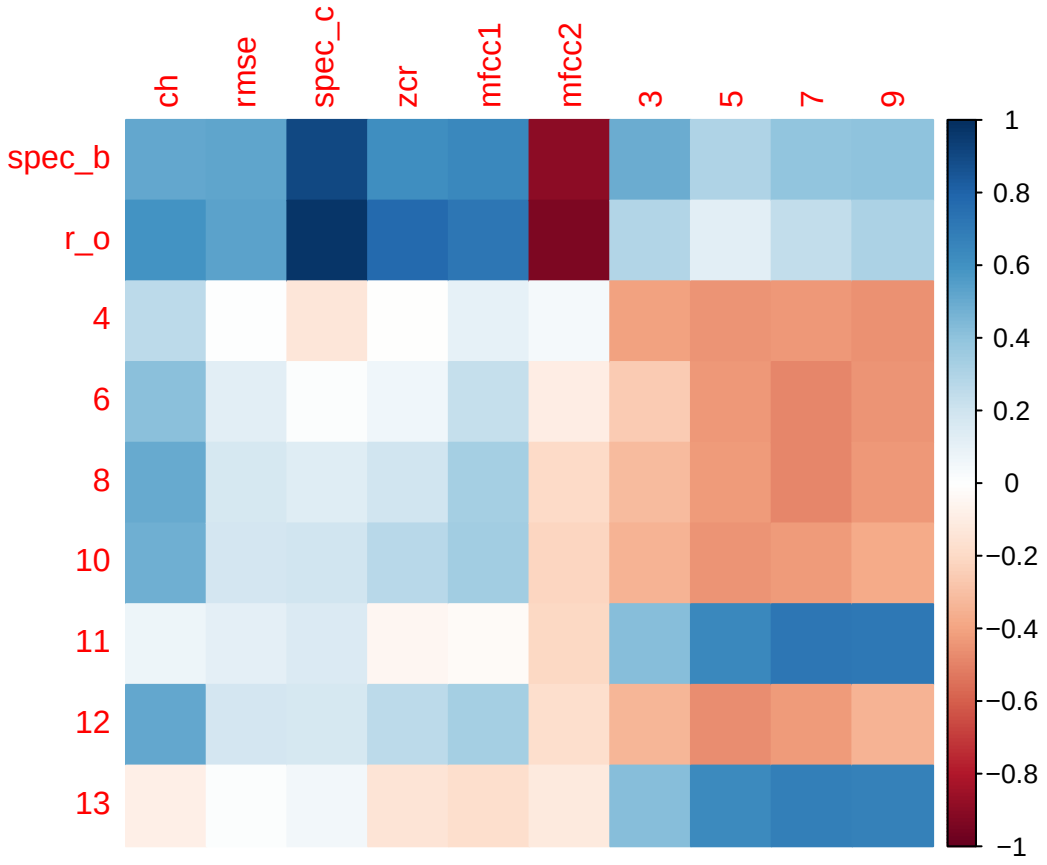
As stated above, for this technique we need two separate datasets. Instead of adding new data to what we have already, we're going to break up the Music Features dataset into 3 smaller matrices, with 9, 10, and 9 of the original variables in each, respectively.

- For the X Matrix, we're going to use the following variables: Spectral Bandwidth (spec_b), roll off (r_o), and MFCCs 4, 6, 8, 10, 11, 12, and 13.
- For the Y Matrix, we're going to use: Chroma (ch), RMSE, Spectral Centroid (spec_C), Zero Crossing Rate (zcr), and MFCCs 1, 2, 3, 5, 7, and 9.
- The rest of the variables, BPM, Beats (b), and MFCCs 14 - 20, were assigned to a third matrix and not used for this analysis.

The tables below shows the X and Y matrix variables with 10 rows of observations, one from each genre represented in the dataset.

10.3 Data Visualization

The plot below shows us the covariance of the data; the centered dot products of the rows and columns of matrices X and Y. We see some of the same patterns that we saw during previous analyses, namely that odd MFCCs are anti-correlated with even MFCCs, and correlated with other odd MFCCs, and vice versa for even MFCCs. We also see strong relationships between the spectral content and MFCCs 1 and 2, but weaker relationships between the spectral content and other MFCCs.



10.4 Analysis

The code below runs the actual analysis on the two matrices. Note that there are two data tables as inputs (X and Y), whereas in previous analyses (PCA, CA, MCA, BADA), there was only one data table to be analyzed. We still have a design vector (music genre) that groups our observations for the analysis.

```
pls.res <- tepPLS(Xmat,Ymat, DESIGN = music.genre,
                  make_design_nominal = TRUE, graphs = FALSE)
resPerm4PLSC <- perm4PLSC(Xmat, # First Data matrix
                          Ymat, # Second Data matrix
                          nIter = 1000) # How mny iterations
resPerm4PLSC # to see what results we have
```

```
## ----------------------------------------------------------------------------
##  Results of Permutation Test for PLSC of X'*Y = R
##  for Omnibus Inertia and Eigenvalues
## ----------------------------------------------------------------------------
## $ fixedInertia      the Inertia of Matrix X
```

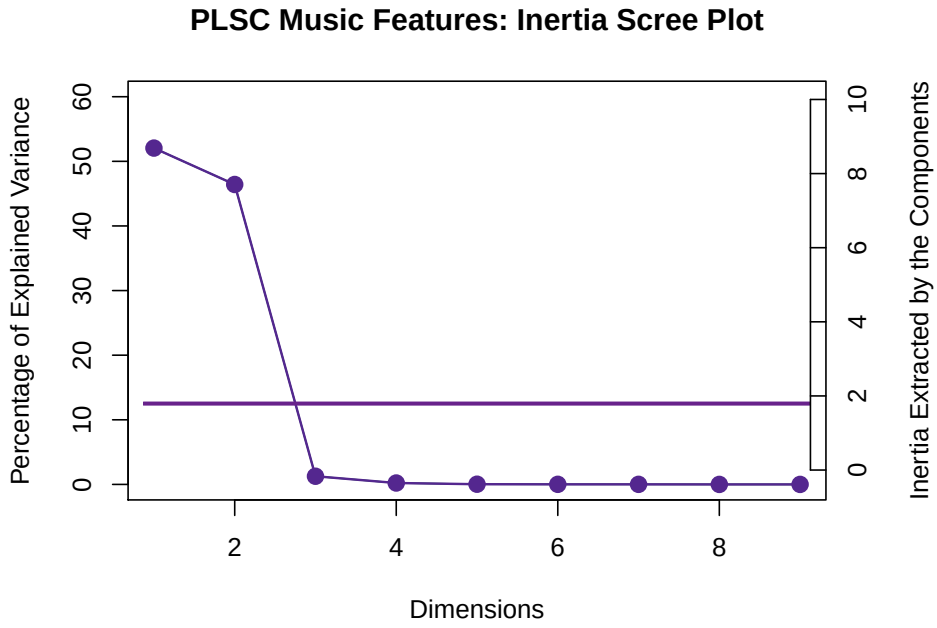
```
## $ fixedEigenvalues  an L*1 vector of the eigenvalues of X
## $ pOmnibus          the probablity associated to the Inertia
## $ pEigenvalues      an L* 1 matrix of p for the eigenvalues of X
## $ permInertia       vector of the permuted Inertia of X
## $ permEigenvalues   matrix of the permuted eigenvalues of X
## -------------------------------------------------------------------------------
```

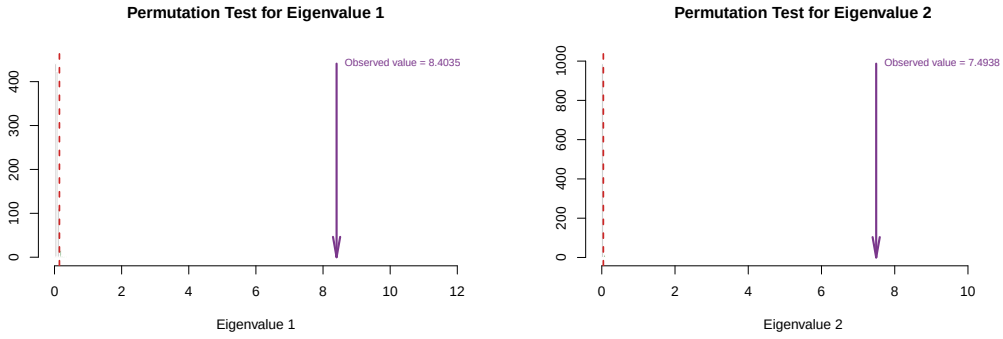
10.5 Results

10.5.1 Scree Plot

The plot below shows our eigenvalues for this analysis. Note that there are 9 dimensions, equal to the least number of variables in either matrix. The permutations tests show us that almost all of our eigenvalues are significant, but looking at the scree plot, it looks like it would be best to look at the eigenplane created by the first two eigenvalues. The permutation tests indicate that these two eigenvalues are definitely above the threshold of p = .05. See the chapter on Inferences for PCA for more on reading scree plots and their permutations.

```
PlotScree(ev = pls.res$TExPosition.Data$eigs,
          p.ev = resPerm4PLSC$pEigenvalues, plotKaiser = TRUE,
          title = 'PLSC Music Features: Inertia Scree Plot')
```





10.5.2 Factor Plots

In previous cookbook pages, the plots were organized slightly differently. All of the computations were done first, and then all of the factor maps were created. Here, we've done all of the calculations for each of the plots with the plot immediately after. First, though, we create the colors.

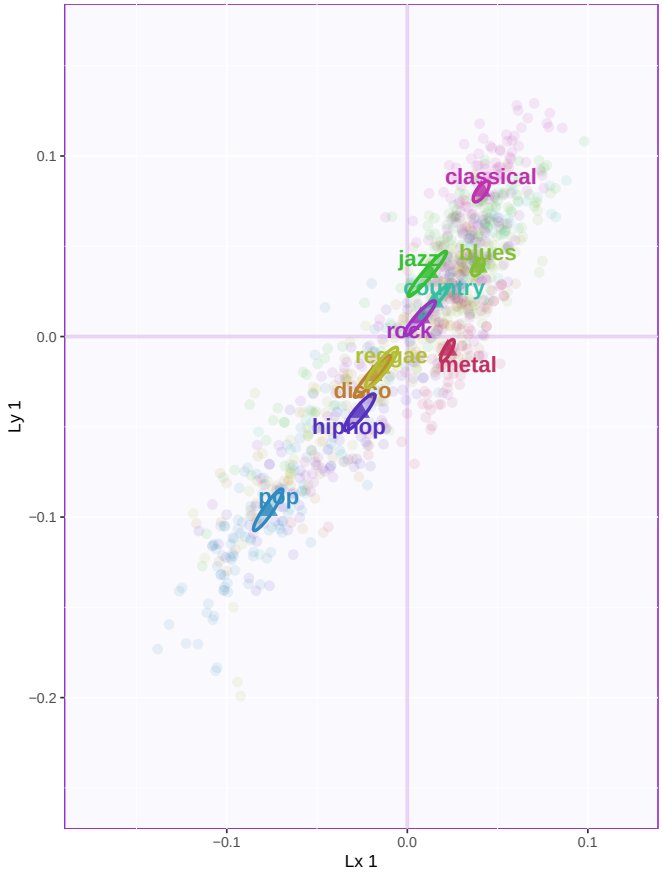
10.5.2.1 Factor Plot 1

The first plot shows us the observations observed using the first latent variables of each table as the axes. Because of the technique we used above, extracting maximal covariance, these two axes are orthogonal.

Because each of these latent variables is orthogonal, we see the observations appear along a diagonal. This plot includes the group means and the confidence intervals for those means as well, and we can see that the genres are effectively separated along the latent variables from both tables. The separation achieved by the LVs from either table works better than the separation along either axis alone.

```
# First, given how CreateFactorMap works, you need to
# create a matrix with observations on the rows, and
# whatever you want to put as the x-axis in the first column,
# and whatever you want to put as the y-axis in the second column.

# For the first plot, the first component of the latent variable of X
# is the x-axis, and the first component of the latent variable of Y is the y-axis
latvar.1 <- cbind(pls.res$TExPosition.Data$lx[,1],pls.res$TExPosition.Data$ly[,1])
colnames(latvar.1) <- c("Lx 1", "Ly 1")

# compute means
lv.1.group <- getMeans(latvar.1, music.genre)

# get bootstrap intervals of groups
lv.1.group.boot <- Boot4Mean(latvar.1, music.genre)
colnames(lv.1.group.boot$BootCube) <- c("Lx 1", "Ly 1")

#Next, we can start plotting:
# Basic factor map
plot.lv1 <- createFactorMap(latvar.1,
                            col.points = pls.res$Plotting.Data$fii.col,
                            col.labels = pls.res$Plotting.Data$fii.col,
                            alpha.points = 0.1,
                            )
```

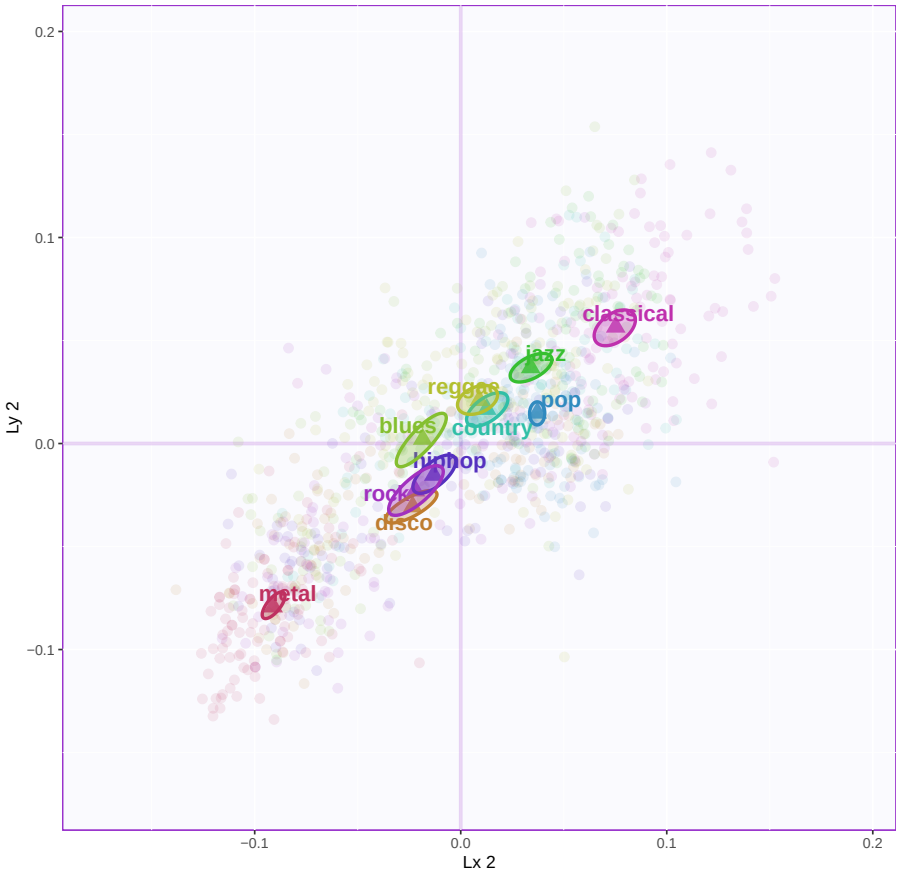
```
# Factor Map of group means
plot1.mean <- createFactorMap(lv.1.group,
                              col.points = grpcol[rownames(lv.1.group),],
                              col.labels = grpcol[rownames(lv.1.group),],
                              cex = 4,
                              pch = 17,
                              alpha.points = 0.8,
                              text.cex = 5, force = 3)
# Factor map for group confidence intervals
# remember we only need the 1st and 2nd columns of each
# page of the bootcube
plot1.meanCI <- MakeCIEllipses(lv.1.group.boot$BootCube[,c(1:2),],
                              col = grpcol[rownames(lv.1.group.boot$BootCube),],
                              names.of.factors = c("Lx 1", "Ly 1")
                              )
# Put together everything we want in a plot:
plot1stlvs <- plot.lv1$zeMap_background + plot.lv1$zeMap_dots +
         plot1.mean$zeMap_dots + plot1.mean$zeMap_text + plot1.meanCI
plot1stlvs
```



10.5.2.2 Factor Plot 2

The second plot shows us the observations using the second latent variables of each table as the axes. Note that the plot retains the same overall diagonal shape, but the groups have rearranged. There is also greater

dispersion than in the first factor plot. Also, the separation between genres appears more effectively on the second component of the Y matrix than the X matrix. Disco, rock, hip-hop and blues all group on the X2, as do reggae and country, and jazz and pop. Metal and classical drive the extremes of both X2 and Y2.



10.5.3 Column Loadings

The plots below shows us the column loadings of the first components of X and Y and the second components of X and Y, separated into their respective matrices and latent variables. Unlike our other column loadings plots, we're not creating the correlation circles here. We know which variables are going to load on which components because we've assigned them to load on those components. What we're seeing here is more about how much they load, and how much variance they contribute. These also represent how much variance is explained in each of the latent variable. The variables that extend further from the axis are more important for that set of latent variables. For this and all of the other barplots, all of the variables from table 1 are shown in reds and yellows, and all of the variables from table 2 are shown in blues and greens. The loadings are shown loading in the direction of the axis they represent: X variables load on the X axis and are shown horizontally, Y variables load on the Y axis and are shown here vertically.

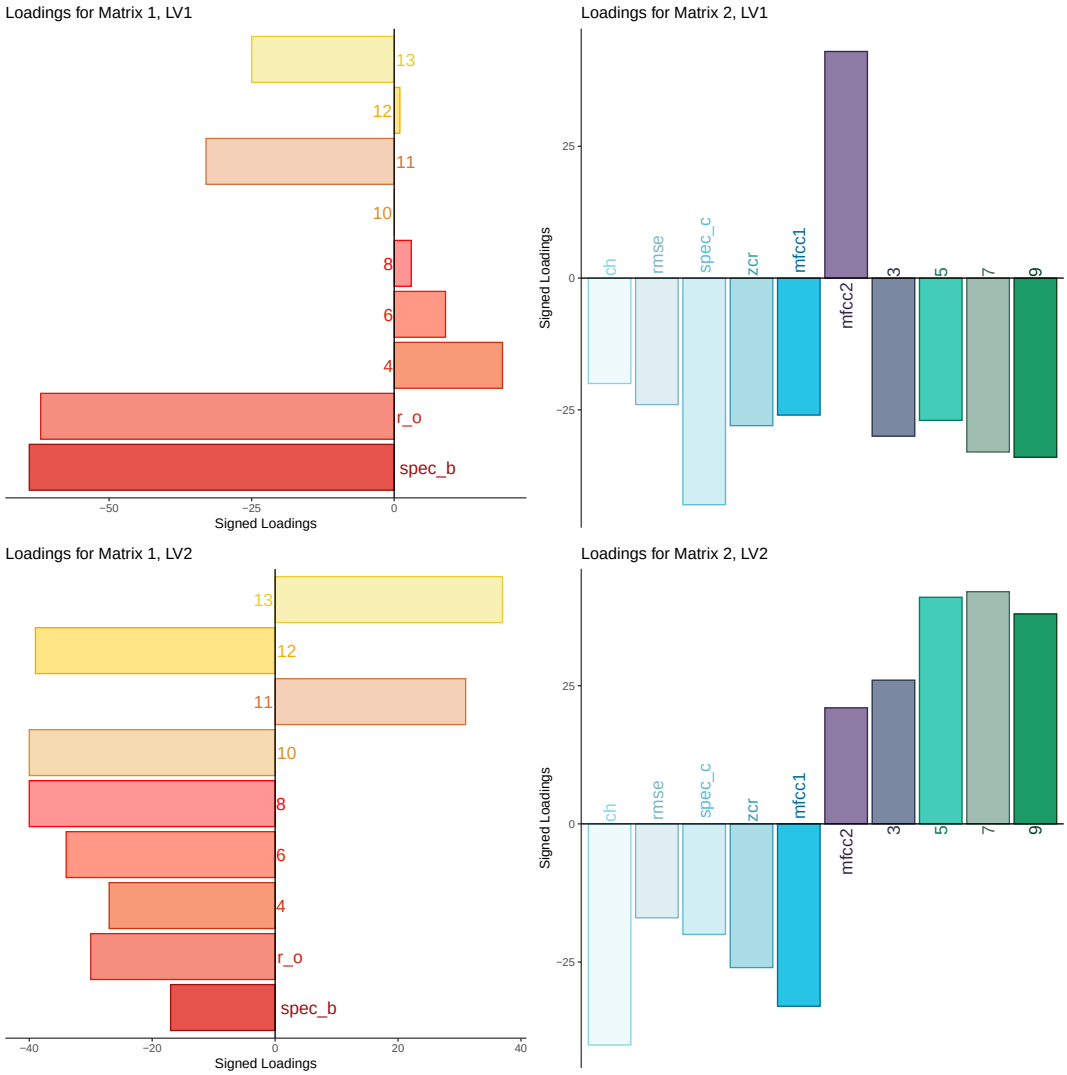
There are a couple of things to talk about here:

First of all, this definitely makes sense in terms of our factor plots. We see that the classical/pop separation on the plots of LV1 is really being driven by MFCC2 and the spec_b and roll-off spectral components. This is consistent with all of the other analyses we've done, if you're not sure what that means, go ahead and check out the PCA page.

Likewise the metal/classical separation is visible on the plots and loadings of LV2. Here we see the zero-crossing rate, MFCC1, and the even MFCCs greater than 2 driving the separation between Classical

and Metal. We also now have the odd MFCCs greater than 1 loading positively, which helps to move pop up closer to classical on the second plot.

Secondly, the loadings for the MFCCs all switch their sign for the load out on the second latent variable. This indicates that after partialing out the variance from the first component, we are able to see how the variables relate otherwise.

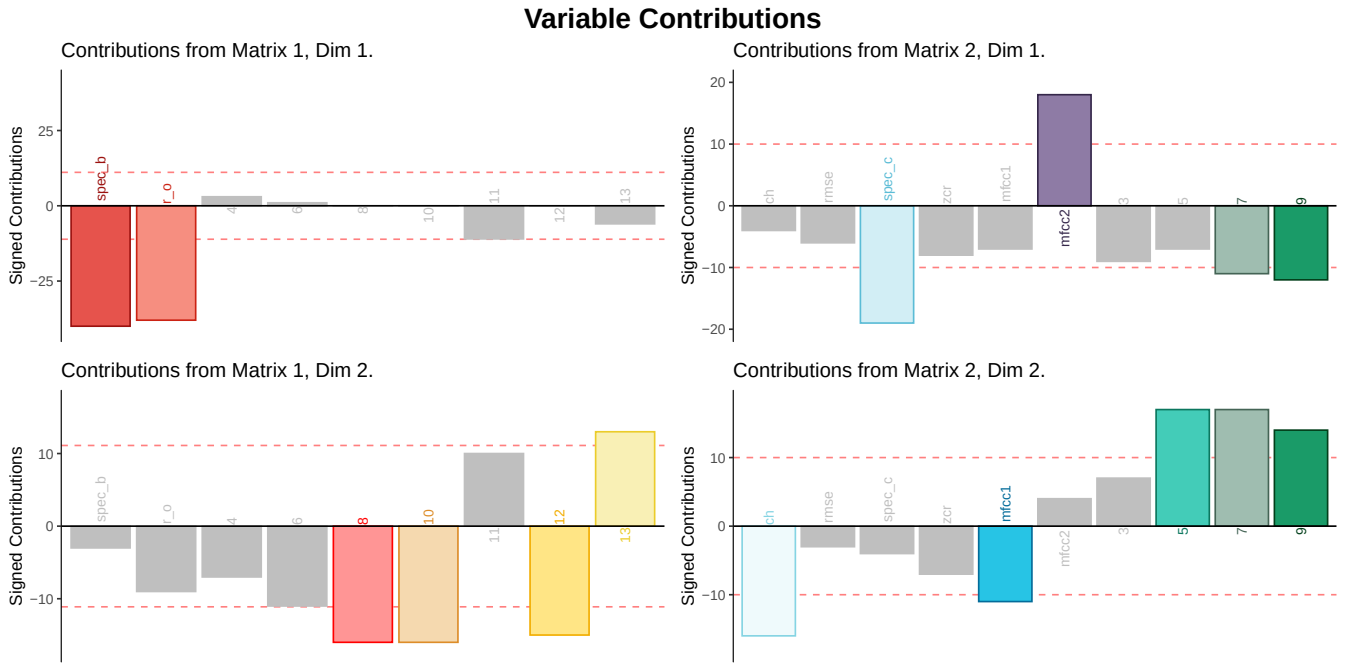


10.5.4 Contributions

For PLSC, we also plot the contributions for both rows and columns. The plot below shows us the contributions for each matrix and dimension. These are the variables (columns) of each of the matrices that drive the variance for each of the latent variables for the matrices. The first is shown as an example, the rest and the means to arrange them are in the rmd file, see there for more.

```
# Assign values for efficiency: cis & cjs are contributions,
# from the PLSC results
ctri <- pls.res$TExPosition.Data$ci
signed.ctri <- ctri * sign(pls.res$TExPosition.Data$fi)
ctrj <- pls.res$TExPosition.Data$cj
signed.ctrj <- ctrj * sign(pls.res$TExPosition.Data$fj)
# Creates the plot
```

```
c001.plotCtri.1 <- PrettyBarPlot2(bootratio = round(100*signed.ctri[,1]),
                                  threshold = 100 / nrow(signed.ctri),
                                  ylim = NULL,
                                  color4bar = cfp,
                                  color4ns = "gray75",
                                  plotnames = TRUE,
                                  main = 'Contributions from Matrix 1, Dim 1.',
                                  ylab = "Signed Contributions")
```



10.5.5 Bootstrap Ratios

The code below shows us how consistently the variables load the way they do on the latent variables. Note that most of the variables are pretty consistent, but some of the MFCCs are not consistent on the first component.

Because PLSC is only good for predicting fixed effects (effectively, you're predicting what is in the dataset; PLSR is useful for predicting the random effects), we need a measure to help us figure out what might happen if we look at observations that are not in the dataset. So we look to bootstrapping to help us to understand how generalizable the PLSC variables are to random effects. Again, the code we use to create the first plot is below, check out the rmd file for the rest. Also see the chapter on Inferences for PCA for more on bootstrapping and reading the plots.

First we need to compute the bootstrap ratios, using `resBoot4PLSC`:

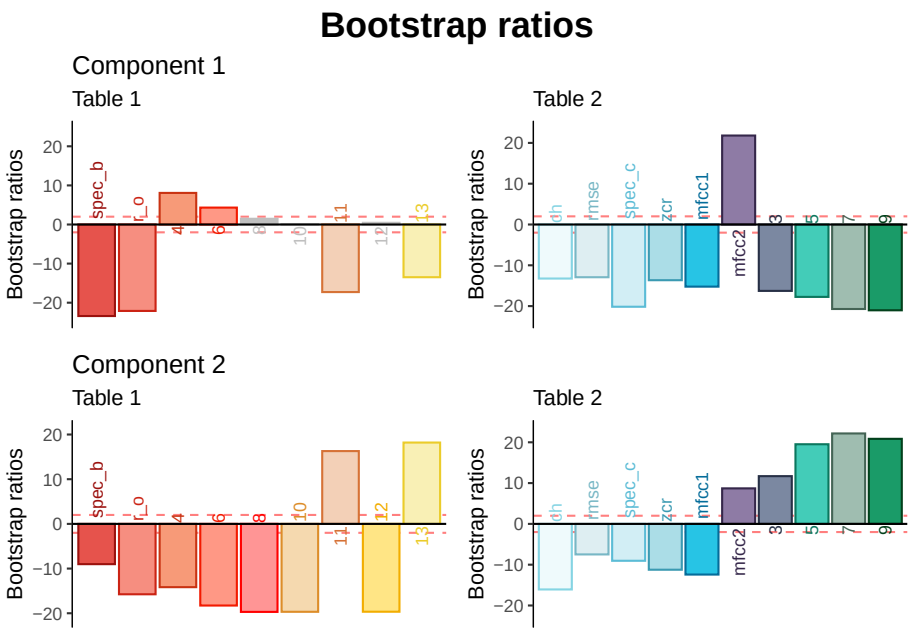
```
resBoot4PLSC <- Boot4PLSC(Xmat, # First Data matrix
                          Ymat, # Second Data matrix
                          nIter = 1000, # How many iterations
                      Fi = pls.res$TExPosition.Data$fi,
                      Fj = pls.res$TExPosition.Data$fj,
                      nf2keep = 3, critical.value = 2,
                      # To be implemented later
                      # has no effect currently
```

```
                      alphaLevel = .05)
print(resBoot4PLSC)
```

```
## ------------------------------------------------------------------------------
##  Bootstraped Factor Scores (BFS) and Bootstrap Ratios  (BR)
##  for the I and J-sets of a PLSC (obtained from multinomial resampling of X & Y)
## ------------------------------------------------------------------------------
## $ bootstrapBrick.i          an I*L*nIter Brick of BFSs  for the I-Set
## $ bootRatios.i              an I*L matrix of BRs for the I-Set
## $ bootRatiosSignificant.i   an I*L logical matrix for significance of the I-Set
## $ bootstrapBrick.j          a  J*L*nIter Brick of BFSs  for the J-Set
## $ bootRatios.j              a  J*L matrix of BRs for the J-Set
## $ bootRatiosSignificant.j   a  J*L logical matrix for significance of the J-Set
## ------------------------------------------------------------------------------
```

Then, we use those bootstrap ratios to create our barplots:

```
BR.I <- resBoot4PLSC$bootRatios.i
BR.J <- resBoot4PLSC$bootRatios.j
laDim = 1
# Plot the bootstrap ratios for Dimension 1
ba001.BR1.I <- PrettyBarPlot2(BR.I[,laDim],
                         threshold = 2,
                         font.size = 3,
                      color4bar = cfp, # we need hex code
                     ylab = 'Bootstrap ratios'
                     #ylim = c(1.2*min(BR[,laDim]), 1.2*max(BR[,laDim]))
) + ggtitle(paste0('Component ', laDim), subtitle = 'Table 1')
```



10.6 Summary

- **From the latent variables:**

  - The first set of latent variables seem to be driving the Classical/Pop distinction we've seen throughout these analyses. They also do a good job of grouping by genre. Classical is on its own, Jazz & Blues are grouped together, with country overlapping with rock and close to blues. Metal is closest to rock, and disco, reggae, and hip-hop all are grouped together. Pop is on the opposite end of the plot from classical. These groupings make a lot of sense in terms of genre relations.
  - The second set of latent variables separate metal and classical. This moves pop much closer to classical, and seems to capture a different set of information than I think of when I think of genre separation. It seems to separate genres based on the actual shape of the signal one might see in a given genre.

- **From the scores of Table 1:**
  - Component 1: According to the bootstrap ratios, all of these measures except for MFCC 8, 10, and 12, seem to be generalizable to genre separation, even though the first component is driven primarily by bandwidth and roll-off, which, based on our previous analyses, are also likely correlated
  - Component 2: MFCC's 8, 10, 12, and 13 drive the variance here. Honorable mention to MFCC 11, which is almost significant in components 1 and 2.

- **From the scores of Table 2:**
  - Component 1: As we've seen before, it's MFCC2 against everything else, with only spectral centroid, and MFCC's 2, 7, and 9 driving the variance here.
  - Component 2: This seems to bring MFCC 2 into check, allows us to take a look at how it's related to the other variables. Chroma and MFCCs 1, 5, 7, and 9 are significant. The higher values of the MFCCs have flipped their sign for the bootstrap ratios.

Part IV: Multi-Table Techniques

Chapter 11

DiSTATIS

11.1 Intro to DiSTATIS

This technique is a generalization of the STATIS technique (Abdi et al. 2006) that allows us to analyze a set of distance matrices for similarity between raters/judges and the objects being rated. STATIS is an acronym for "Structuration des Tableaux à Trois Indices de la Statistique". There are a number of background steps that are involved in this procedure, for more information check out this article and this article. From this technique we can evaluate clustering, similarity, and factor scores maps similar to those presented in MFA.

The data for this example require a specific kind of preparation. In an experiment, survey, or other, you ask judges to rate a set of objects by putting them into groups. The judges have to use at least two groups, but can use as many groups as they like. Once you have your set of ratings (in this case we have 51 judges rating 30 beers), you nominalize the data so that you have a matrix that has as many columns as the judge used groups, and you use that matrix as one "page" or "slice" in a three dimensional set (cube) of matrices that measure the similarity and difference of the the objects.

The sum of the times that a judge rates two objects as being in a different category or group then acts as the value for the difference (or rather, distance) between the objects. However, because any beer can only ever be in the same group as itself, technically we end up with 0s on the diagonal. To solve this we double-center the matrix and multiply by -.5.

Additionally, if you allow the judges to rate the objects qualitatively, you end up with a set of words to describe those objects, which can be projected onto the factor space to see how they correspond to the objects themselves.

You can also use qualitative information on the judges as a design factor for groups the analysis.

11.1.1 Strengths & Weaknesses

Strengths
- This is a great technique to compare similarity between objects without necessitating the use of an absolute scale.
- Allows for biplots that contain object barycenters with rater projections, as well as attribute barycenters based on rater uses of the descriptors.
- Can be used to create dendrograms and k-means clusters, which are also useful tools for assessing similarity.

Weaknesses
- Not really a weakness, per se, but the amount of plots and analyses possible using this technique is quite

high, so it takes some time to go through the results and find what's useful and what's not in terms of forming your interpretation.

11.1.2 Dos and Don'ts

Do:
- Make sure you're sure of your grouping variables.
- Make sure you check through your data when processing/preprocessing.

Don't:
- Try to cram too much information into a single plot. The visualizations should clarify, not obfuscate, the data/the results.

Research Questions Questions for this technique can be about either (both) the objects being observed and the raters rating them. A comprehensive analysis of the data will involve both group descriptors for the raters and analysis of the attributes the raters assigned to the objects.
- What fundamental features of these objects cause them to be differentiated along these principal components?
- Are there any systematic ways in which the object ratings by the judges differ? What causes those systematic differences?
- What can we learn about the tendencies of the judges from the rating data and the systematic differences identified in the previous question?
- What groups arise consistently as a function of these ratings and are there characteristics that aren't immediately apparent that are causing those groups to arise?

11.2 Data

As mentioned above, this dataset is a set of 30 beers rated by 51 participants. The beers are mostly Mexican and Central American beers, with a few European beers thrown in.

The are no qualitative descriptors assigned by the judges for the beers, but we have two pieces of information for the judges. We have their gender, and whether or not the judges prefer "industrial" or "craft" beers. I think it would be more interesting to see whether or not there are rating differences based on drink preference, so we're going to use that variable as our design variable.

	C1	C2	C3	C4	C5
Minerva PA	1	1	1	1	1
Cucapa Miel	2	1	2	1	2
Tempus Clasica	2	1	3	1	2
Tempus DM	3	1	3	1	2
Calavera MIS	4	1	2	1	2

11.3 Data Processing and Analysis

11.3.1 The judges

This first bit of code runs the nominalization that we needed from earlier. It also sets our design variable to the 2nd column of the judges' info (beer preference) and assigns colors to the judges based on their groups. We'll see what that looks like later when we plot the similarity of the judges.

```
# our design variable - remember we only have 2 pieces
# of information the judges, this selects the second column
```

```
# of the design_data matrix as our design variable
k <- 2
descJudges <- Design_Data[,k ]
nominal.Judges <- makeNominalData(as.data.frame(descJudges))
# get the colors using this function from the prettygraphs package
color4Judges.list <- createColorVectorsByDesign(nominal.Judges)
```

11.3.2 3.1 Distance Cube & DiSTATIS

This creates our distance cube we need to run the DiSTATIS and runs the analysis on it, saving the results in Distatis.res

```
DistanceCube <- DistanceFromSort(Sorting_Data)
Distatis.res <- distatis(DistanceCube)
```

11.3.3 Inference

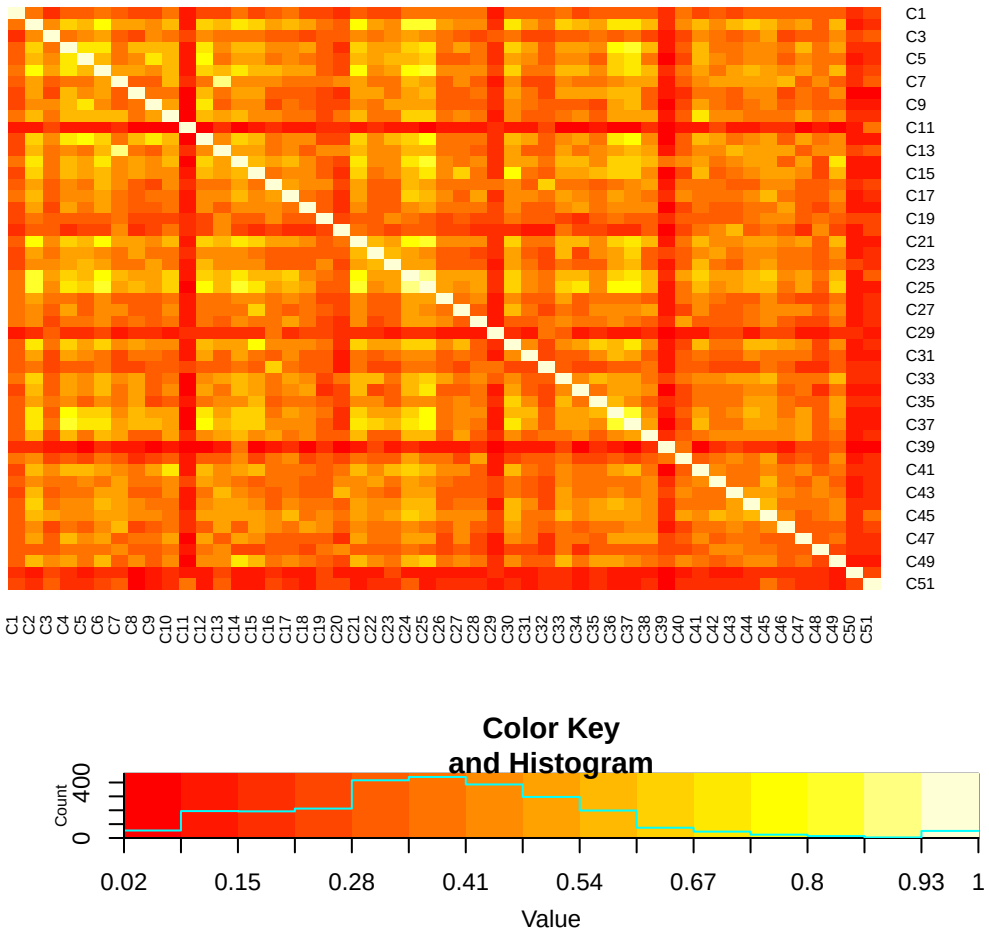
In order to run our bootstrapping for the judge means, we need to get the factors from the DiSTATIS results. Then we aggregate the judges and run a bootstrap analysis on the judge means.

```
# Get the factors from the Cmat analysis
G <- Distatis.res$res4Cmat$G
# Compute the mean by groups of HJudges
JudgesMeans.tmp <- aggregate(G, list(descJudges), mean)
JudgesMeans <- JudgesMeans.tmp[,2:ncol(JudgesMeans.tmp )]
rownames(JudgesMeans) <- JudgesMeans.tmp[,1]
# Get the bootstrap estimates using this function from PTCA4CATA package
BootCube <- Boot4Mean(G, design = descJudges,
                      niter = 100, suppressProgressBar = TRUE)
```

11.4 Results

11.4.1 Heatmap

Before we used heatmaps to look at frequency of occurrence, or relative values of observations/variables in a given dataset to look at trends in the data. here we're doing something slightly different. In the plot below, we see how the judges rated the beers, and the relative distance between each of the judges at the intersection. The darker cells mean that the judges are further away from one another, i.e. they rated the beers more differently, and the lighter cells indicate that the judges rated the beers more similarly to one another. The key shows us those values and also shows us what the density distribution of similarity is. There's a lot that went into constructing this plot, so check out the RMD for the specifics on how this was modified.



11.4.2 Partial map by judges

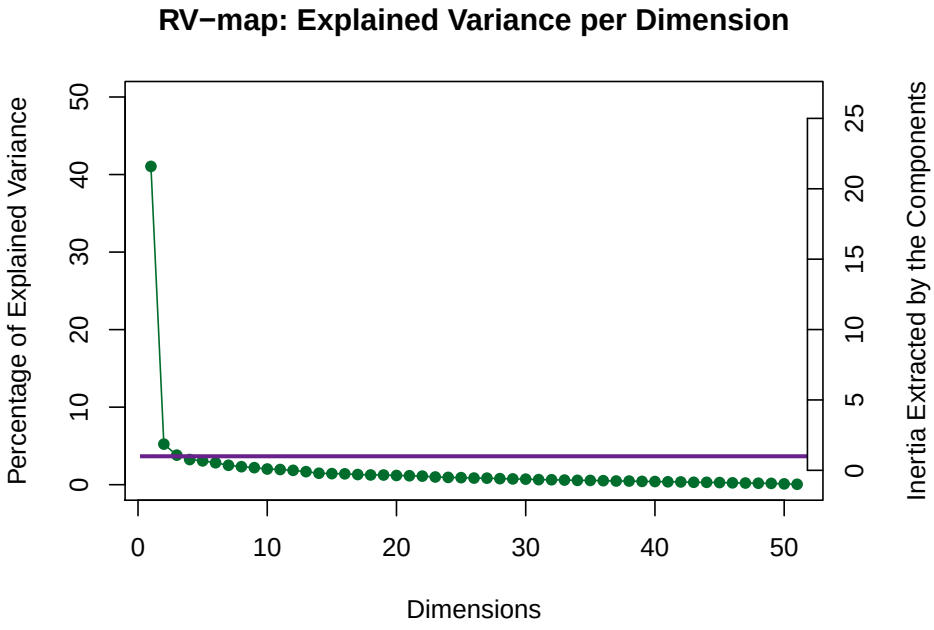
This section of the analysis helps us to get the correct loadings for our partial factor scores. We'll use the results from it later.

```
F_j <- Distatis.res$res4Splus$PartialF
alpha_j <- Distatis.res$res4Cmat$alpha
# create the groups of Judges
#groupsOfJudges <- substr(names(alpha_j),1,1)
groupsOfJudges <- descJudges
code4Groups <- unique(groupsOfJudges)
nK <- length(code4Groups)
# initialize F_K and alpha_k
F_k <- array(0, dim = c(dim(F_j)[[1]], dim(F_j)[[2]],nK))
dimnames(F_k) <- list(dimnames(F_j)[[1]], dimnames(F_j)[[2]], code4Groups)
alpha_k <- rep(0, nK)
names(alpha_k) <- code4Groups
Fa_j <- F_j
```

```
# A horrible loop
for (j in 1:dim(F_j)[[3]]){ Fa_j[,,j] <- F_j[,,j] * alpha_j[j] }
for (k in 1:nK){
                lindex <- groupsOfJudges == code4Groups[k]
                alpha_k[k] <- sum(alpha_j[lindex])
                F_k[,,k] <- (1/alpha_k[k])*apply(Fa_j[,,lindex],c(1,2),sum)
                }
```

11.4.3 Scree for Judges

Here we have our scree plot for the judges. The dimensionality of this screeplot is determined by the number of judges included in the analysis. However, it looks like there's really one one dimension worth looking at, perhaps 2, and we'll see how that shakes out in our factor plots. See PCA for more on reading scree plots.

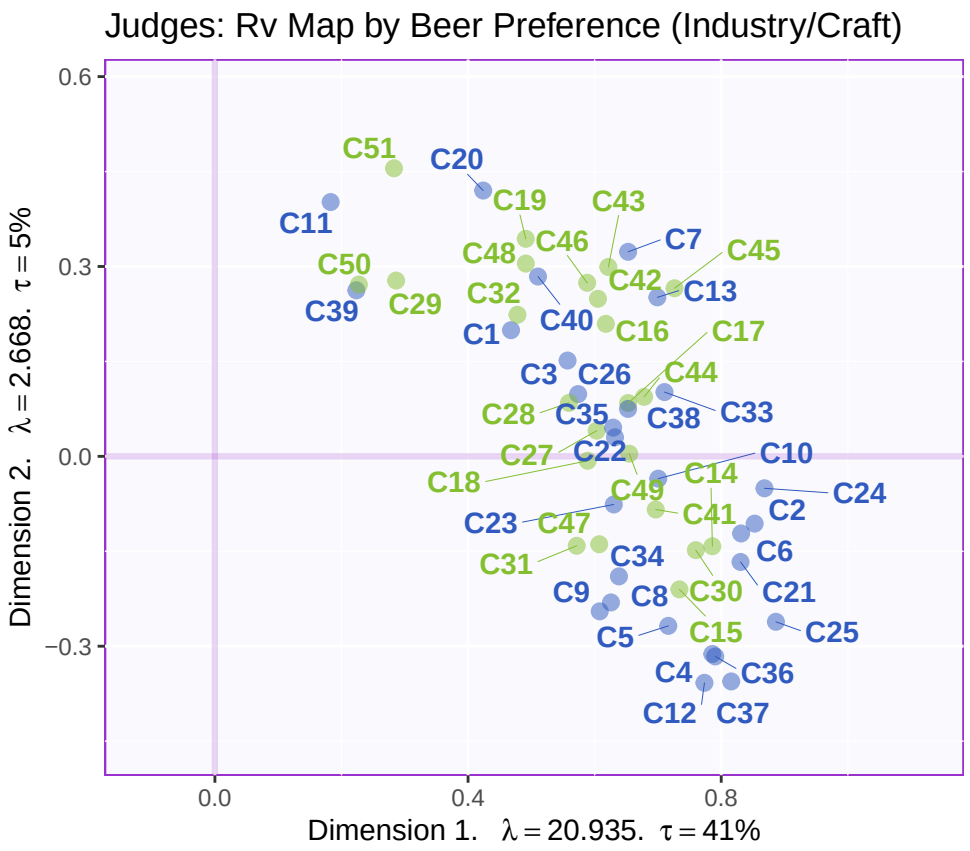


11.4.4 Factor Map for the Judges

This plot shows us how similarly the judges rated the beers. The blue dots indicate the judges that said they preferred 'Industrial' type beers and the green dots indicate judges that said they preferred 'craft' type beers. Although the dots seem pretty well mixed, it does look the the majority of the Industrial preferring judges are loaded negatively on the second dimension and the majority craft preferring judges are loaded positively on the second component.

```
gg.rv.graph.out <- createFactorMap(X = Distatis.res$res4Cmat$G,
                                   axis1 = 1, axis2 = 2,
        title = "Judges: Rv Map by Beer Preference (Industry/Craft)",
                                   col.points = color4Judges.list$oc,
                                   col.labels = color4Judges.list$oc)
# create the labels for the dimensions of the RV map
labels4RV <- createxyLabels.gen(lambda = Distatis.res$res4Cmat$eigValues,
                                tau = Distatis.res$res4Cmat$tau,
                                axisName = "Dimension ")
```

```
# # Create the map from the layers
# Here with labels and dots
a2a.gg.RVmap <- gg.rv.graph.out$zeMap + labels4RV
# Here with colored dots only
a2b.gg.RVmap <- gg.rv.graph.out$zeMap_background +
                gg.rv.graph.out$zeMap_dots + labels4RV
a2a.gg.RVmap
```

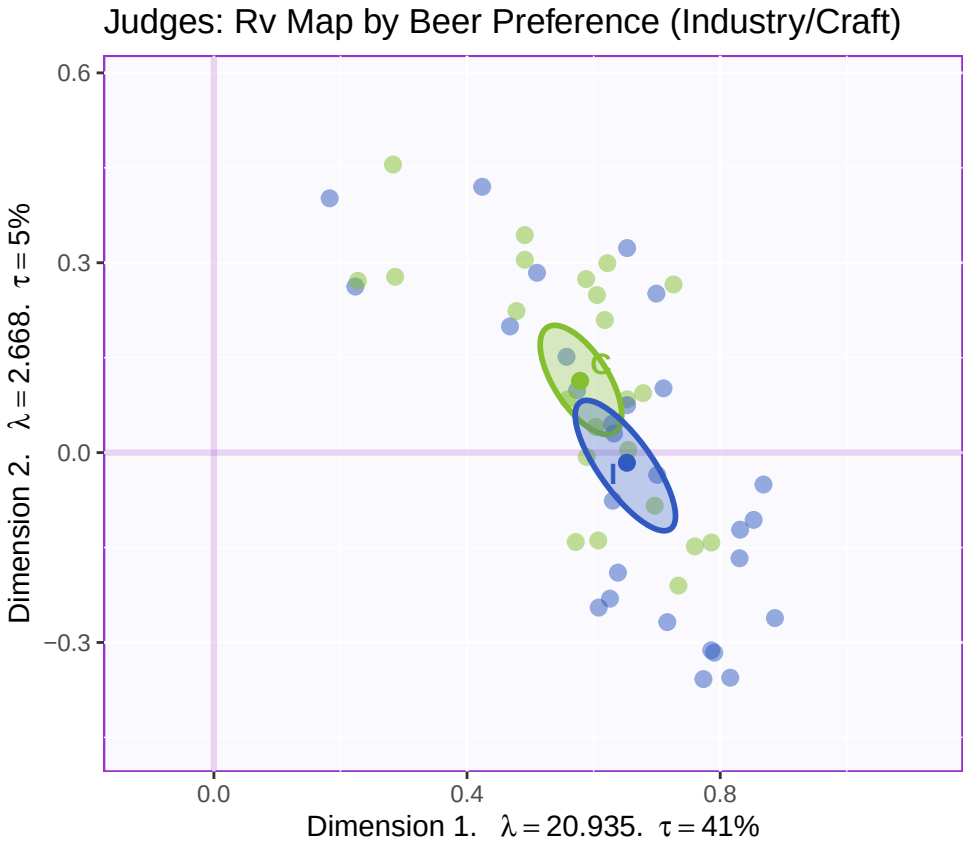


11.4.4.1 Factor Map for the Judges with Bootstrapped Confidence interval

As we can see below, the confidence intervals for the means of the groups overlap, which indicates that there really isn't much difference in how the judges grouped the beers overall. We'll see later why that might be. Ch

```
# First the means
# A tweak for colors
in.tmp <- sort(rownames(color4Judges.list$gc), index.return = TRUE)$ix
col4Group <- color4Judges.list$gc[in.tmp]
#
gg.rv.means <- createFactorMap(JudgesMeans,
                               axis1 = 1, axis2 = 2,
                               constraints = gg.rv.graph.out$constraints,
                               col.points = col4Group ,
                               alpha.points = 1, # no transparency
                               col.labels = col4Group,
                               display.labels = TRUE
```

```
                                  )
# Luckily R is smart enough to know when we've intentionally broken a line
# We've done this here so you can see how we rename
# the dimensions of the bootcube.
dimnames(BootCube$BootCube)[[2]] <- paste0('dim ',1:
                                           dim(BootCube$BootCube)[[2]])
GraphElli.rv <- MakeCIEllipses(BootCube$BootCube[,1:2,],
                               names.of.factors = c("dim 1","dim 2"),
                               col = col4Group,
                               p.level = .95)
a2d.gg.RVMap.CI <- a2b.gg.RVmap + gg.rv.means$zeMap_dots +
                   GraphElli.rv + gg.rv.means$zeMap_text
a2d.gg.RVMap.CI
```



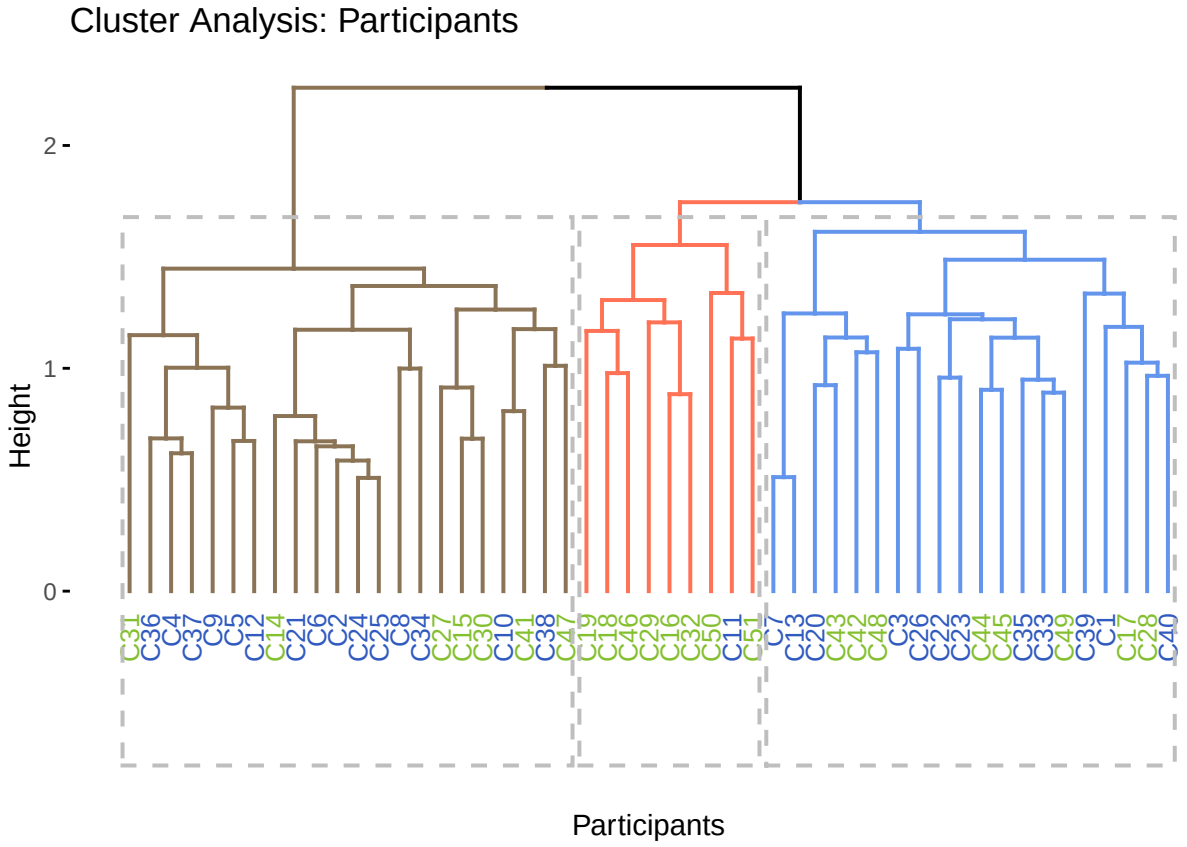
11.5 Cluster analysis

11.5.1 Tree Plot

The plot below is a tree diagram showing how the groups of judges rated the beers. Note that the main branch point at the top of the chart shows you the two main groups. In a perfect world, the design variable we chose would be reflected by that split. Needless to say, that is not what happened. The `fviz_dend` allows us to select groups for the dendrogram using the `rect` parameter, so we can see what groupings arise. We get groups for a dendrogram not by following the lines to where they split, but by drawing horizontal lines through the plot, and seeing what splits off. It would be possible to select anywhere between 2 and 6 groups, but three makes the most logical sense. The closer the splits are, the harder it is to justify. It's also

important generally to make as few assumptions as possible, but the more groups you try to break apart, the more assumptions you're making. In a larger dataset, it might make more sense to make more divisions (things like demographics, age groups, etc.) but here we don't have any of that information so it wouldn't make sense to make those assumptions.

```
D <- dist(Distatis.res$res4Cmat$G, method = "euclidean")
fit <- hclust(D, method = "ward.D2")
a05.tree4participants <- fviz_dend(fit, k = 3,
                          k_colors = c('burlywood4', 'coral1', 'cornflowerblue'),
                          label_cols = color4Judges.list$oc[fit$order],
                          cex = .7, xlab = 'Participants',
                          main = 'Cluster Analysis: Participants', rect = TRUE)
a05.tree4participants
```

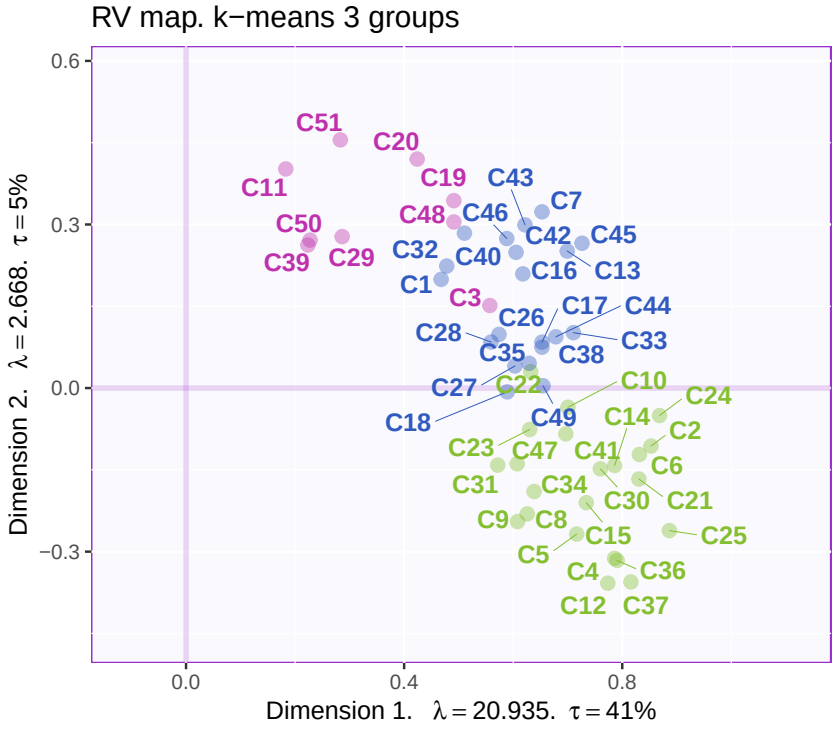


11.5.2 K-Means

The plot below finds three barycenters of the judges and assigns them to colors groups based on that similarity. Note, however, that these groups aren't the same groups as we see above in the dendrogram. With more information on the participants, these differences might be more interpretable, but I'm not entirely sure what's going on here.

```
# First plain k-means
set.seed(42)
participants.kMeans <- kmeans(x = G , centers = 3)
# Again, breaking lines...
col4Clusters <- createColorVectorsByDesign(
                makeNominalData(as.data.frame(participants.kMeans$cluster)))
```

```
# We use the colors created in theline above to show us
# where the groups are in the factor map, which is otherwise
# created similarly to usual.
baseMap.i.km <- createFactorMap(G, title = "RV map. k-means 3 groups",
                                col.points = col4Clusters$oc,
                                col.labels = col4Clusters$oc,
                                constraints = gg.rv.graph.out$constraints,
                                alpha.points = .4)

a06.aggMap.i.km <- baseMap.i.km$zeMap_background +
                   baseMap.i.km$zeMap_dots +
                   baseMap.i.km$zeMap_text + labels4RV
a06.aggMap.i.km
```



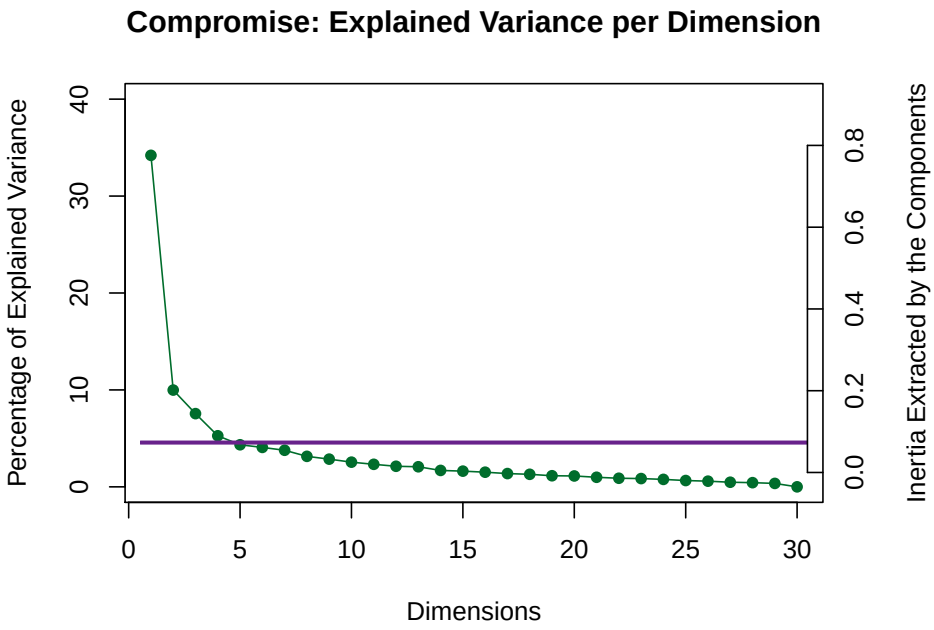
11.6 Analysis by beers

Remember that a this analysis allows us to investigate not only how the raters were similar or different, (which may have been more interesting if we had more data on the participants) but also how similarly or differently those raters rated the beers.

11.6.1 Scree Plot

The DiSTATIS function doesn't give us eigenvalues for the Splus matrix (the rows/observations), so to get our eigenvalues for this analysis, we run `eigen` on `Distatis.res$res4Splus$Splus`.
The eigenvalues for this are the eigenvalues of the compromise matrix. It looks like there is once again a single main dimension that we're really going to be interested in, but there is definitely a bit more variance extracted in the second, third, and fourth dimensions that may be worth analyzing. See PCA for more on interpreting scree plots.

```
comp.eigs <- eigen(Distatis.res$res4Splus$Splus)
scree.S.out <- PlotScree(ev = comp.eigs$values,
                         title = "Compromise: Explained Variance per Dimension",
                         plotKaiser = T)
```



11.6.2 Beer Bootstrapping

Below are two functions for bootstrapping our beer data. These are specific to this analysis. Check out the documentation for more info on them. We're actually using the results from the first option in the next section. Because it's so quick, I've shown both options. The first function is very fast but only bootstraps from the factor scores, it doesn't calculate factor scores by bootstrapping the original distance cube, and therefore may be too liberal if the number of assessors is very large. Our data is only 51, so not that big, so it should be fine. The second function bootstraps the distance cube (in this case; check out MFA for what a compromise matrix is) and computes factor scores from there. The results are more robust.

```
#  Option 1: bootstrap from factor scores, default is 1000 iterations
BootF <- BootFactorScores(Distatis.res$res4Splus$PartialF)
```

```
## [1] Bootstrap On Factor Scores. Iterations #:
## [2] 1000
```

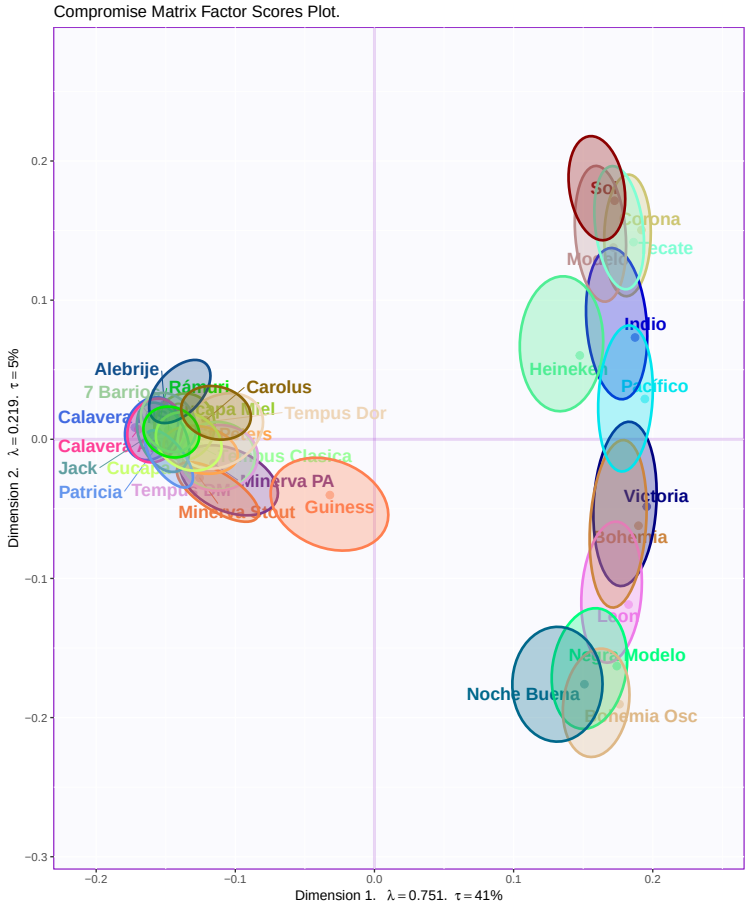
```
#
# Option 2: Full bootstrap, default is also 1000 iterations
F_fullBoot <- BootFromCompromise(DistanceCube)
```

```
## [1] Starting Full Bootstrap. Iterations #:
## [2] 1000
```

11.6.3 Beer Factor Scores Plot

There's a lot going on in the global factor scores plot below. This is the same as any of the other factor score plots in this book, so interpretation hasn't changed. However, what we need to understand about

this is that these factor scores represent 'compromise' factor scores between all of the matrices created by recoding the original grouping data.
If you look at the RMD file, there's a graph with ellipses. Because we haven't used groups to group the beers, the ellipses aren't terribly informative. Instead of showing us bootstrapped group means, they show us approximations of where the individual observations group together. This doesn't really show us anything that isn't obvious in the original plot, and creates more distraction. The code is included regardless so that for future analyses, you can include groups and bootstrapping.
The plot below shows us the global factor scores of how the judges grouped the beers. There are two obvious groupings, an industrial group on the positive end of component 1, and a more crafted group on the negative end of component 1. There is also a spread of the industrial group loading positively to negatively corresponding to light vs. dark beers, respectively.
What it looks like is people tended to grouped beers in three groups: unfamiliar, familiar light, and familiar dark. On the right side we see the differentiation between light and dark, but on the left, we have APAs and imperial stouts grouped together, which suggests that the judges weren't sure what the beers actually were. Guinness doesn't seem to fit into either of those groups, which shows that people weren't sure whether to put it with the 'craft' beers or the 'industrial' beers.

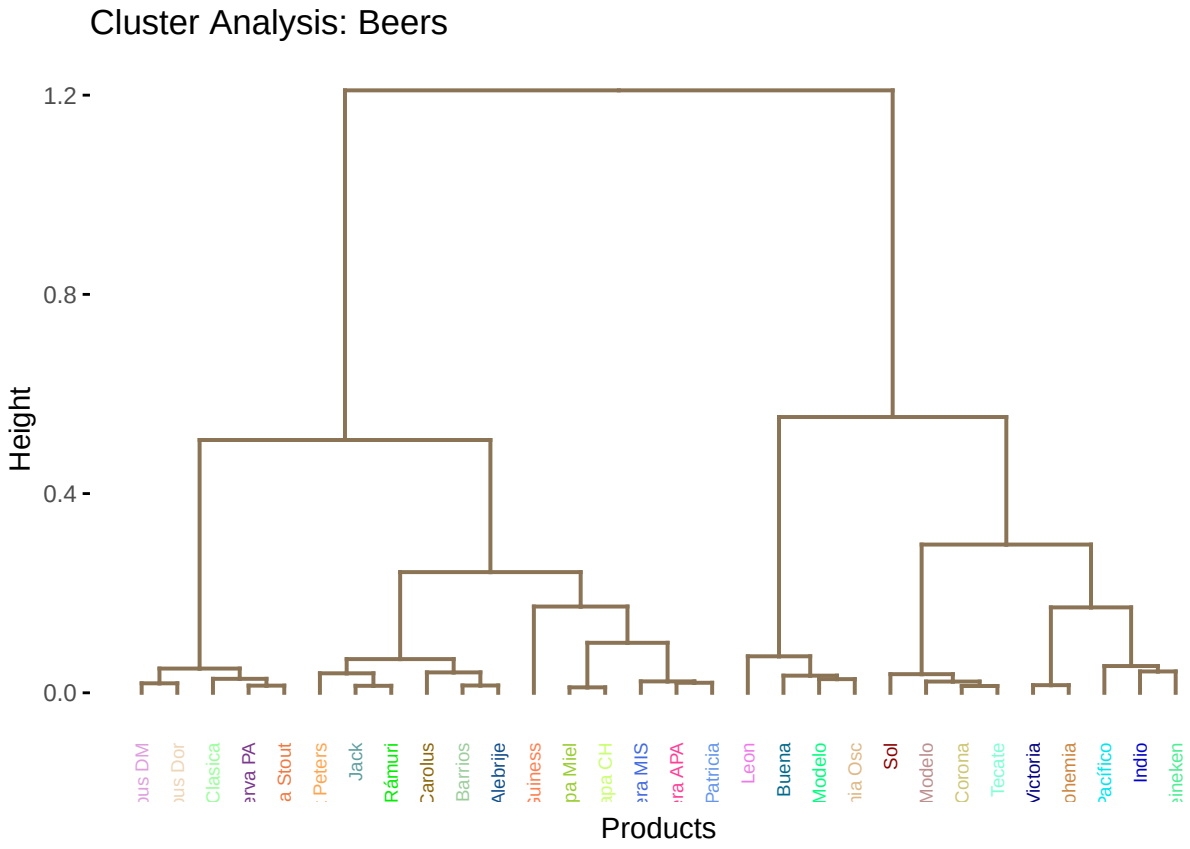


11.6.4 Beer Tree

The plot below shows us how the beers are grouped together. Because there is so much overlap in the factor scores plot, this actually serves as a great tool in determining how the beers were judged. We see clusters that may offer some insight into the thought process of the judges.

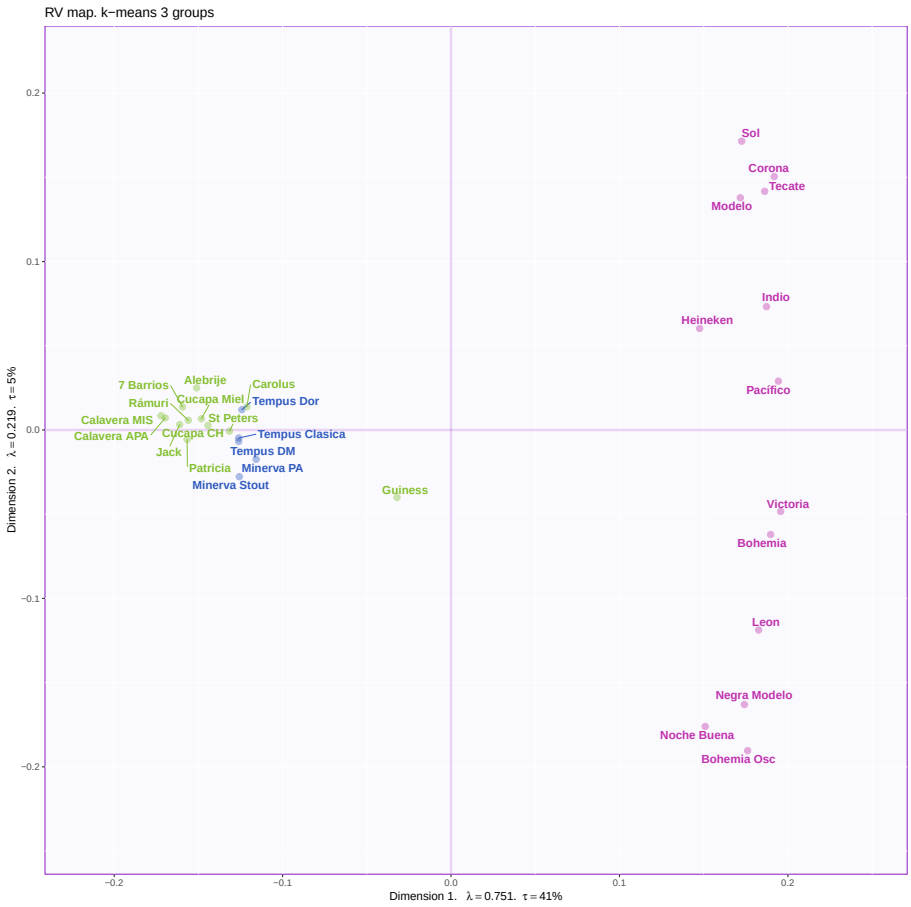
```
nFac4Prod = 3
D4Prod <- dist(Distatis.res$res4Splus$F[,1:nFac4Prod], method = "euclidean")
```

```
fit4Prod <- hclust(D4Prod, method = "ward.D2")
b3.tree4Product <- fviz_dend(fit4Prod, k = 1,
                             k_colors = 'burlywood4',
                             label_cols = color4Products[fit4Prod$order],
                             cex = .5, xlab = 'Products',
                             main = 'Cluster Analysis: Beers')
b3.tree4Product
```



11.6.5 K-means: Beers

This plot does the same thing the k-means plot above does, but this time its for the beers, not the judges. We're using three centers. Notice that there are a few points that look they are in the wrong group: Guinness, for example, and Tempus Dor. this is likely because the grouping is grouping the variables using three dimensions, not just 2.



11.6.6 Beer Partial Factor Scores Map

This plot adds the partial factor scores of the groups of judges to the global factor scores map. It shows us how the groups of judges differed in how they rated the beers, and may give us an idea of how the groups trended in terms of grouping beers that they may or may not have been familiar with. A couple of points that highlight this difference are Heineken and Pacifico, and the grouping of Negra Modelo, Noche Buena, and Bohemia Oscuro. Judges who preferred craft beers were more likely to rank Heineken and Pacifico as lighter, and closer to the grouping of Sol, Modelo, Corona, and Tecate, whereas the industrial beer drinkers were more likely to create a separation between those two groups. For the trio of darker lagers, we see a similar pattern. The industrial beer drinkers were more likely to group the three darker beers closer to the X axis, perhaps grouping them with Leon, Bohemia, and Victoria. This suggests that they were focusing on the brand of the beer, rather than the content. On the other hand, the craft beer drinkers seem to create a separation between the Bohemia and Bohemia Oscura, suggesting that they're focusing on the actual style or content of the beer.

```
# get the partial map
map4PFS <- createPartialFactorScoresMap(factorScores = Distatis.res$res4Splus$F,
                                        partialFactorScores = F_k,
                                        axis1 = 1, axis2 = 2,
                                        colors4Items = as.vector(color4Products),
                                        names4Partial = dimnames(F_k)[[3]], #
                                        font.labels = 'bold')
# This gives us a factor map where the partial scores colored by the two levels
# of the judge design variable
d1.partialFS.map.byblocks <- gg.compromise.graph.out$zeMap +
```

```
                              map4PFS$mapColByBlocks + label4S
# This gives us a factor map where the partial scores are colored by the levels
# of beers. The partial factor scores are the same, but that just means that the
# partial scores and the lines are the same color as the beer dots. This isn't useful
# information. The plot created by the code above is more interpretable.
d2.partialFS.map.byCategories <- gg.compromise.graph.out$zeMap +
                                 map4PFS$mapColByItems + label4S
d1.partialFS.map.byblocks
```

11.6.7 Contributions calculations

The `distatis` function doesn't provide us with contributions from each of the judges or from each of the beers, so we have to calculate those ourselves. The code below does that.

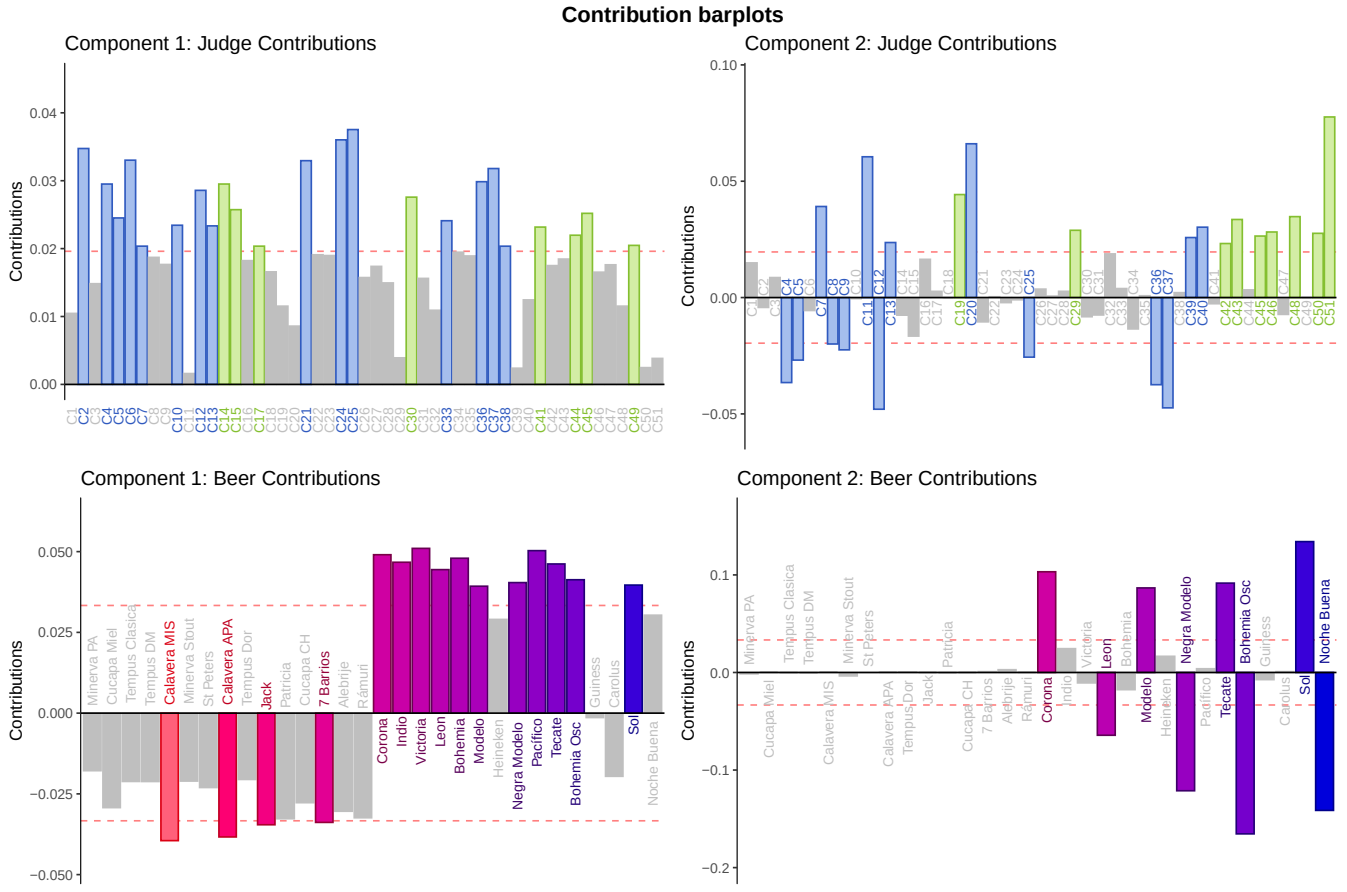
11.6.8 Contributions Barplots

The values we calculate using the above code is visualized below. It shows us which of the beers and which of the judges drive the dimensions that we've plotted, or that we're evaluating. The plots below show us that all of the contributions of the first dimension happen to be positive. Also there are just under half of the observations that contribute significantly to the dimension, and of those, 15/23 are industrial beer drinkers, so it looks like that group contributes more to the first dimension. For the second dimension, we see a similar thing, that the industrial beer drinkers represent 14/23 of the significant observations. This is interesting, in that it suggests that the industrial beer drinkers overall contribute more to the significant dimensions of the factor space.

The contributions for the beers are similar in that very clearly see the first dimension represented as craft vs. industrial beers (or rather known vs. unknown beers), and in the second dimension the darkest and the lightest of the well known beers contribute almost all of the variance.

The sample code below shows how we're getting the colors and also the contributions for the first dimension of the columns, which are the judges. See the rmd file for the rest of the code used to create the plot below. Note that 0 is hard-coded as the minimum of the y axis. That's because all of our contributions for the first dimension happen to be positive and avoids ggplot errors.

```
colfunc <- colorRampPalette(c("red","darkblue"))
col4rows <- colfunc(NROW(Ctr))
colconts1 <- PrettyBarPlot2(
                          bootratio = RvCtr[,1],
                          threshold = 1 / NROW(RvCtr),
                          font.size = 3,
                          color4bar = color4Judges.list$oc, # we need hex code
                          main = 'Component 1: Judge Contributions',
                          ylab = 'Contributions',
                          ylim = c(0, 1.2*max(RvCtr[,1]))
                          )
```



11.7 Conclusions

- **Component 1**
  - This primarily differentiates our data into the 'familiar' and 'unfamiliar' beers. It makes sense

that even craft beer drinkers can't be familiar with *every* beer on the list.

- **Component 2**
  - This seems to differentiate the 'familiar' beers along a light/dark spectrum, with the light beers on the positive end and the dark beers on the negative. We don't see the same spread between the 'unfamiliar' beers, though.
- **Judges**
  - Although there was a wide dispersion in how the judges rated the beers, it was largely not systematic in terms of beer drinking preference. This suggests that although people identify as craft or industrial beer differences, they may not be super aware of the entire variety of beers on the market.
- **Beers**
  - Where as there was a clear distinction in terms of the 'familiar' beers, there was no clear distinction in terms of the 'unfamiliar' beers. This suggests that the raters were likely to simply group the beers together by appearance of the label or something like that, if they didn't clearly know what the beer was or what it tasted like. And there were enough beers that were unfamiliar that this was the primary systematic distinction between the beers.
- **Interpretation**: Although there was a separation between the means of the judges groups, it looks like there was a systematic difference only in how they rated the industrial beers. Also it looks like overall, there is a massive lack of familiarity with the craft beers presented in this experiment. I would also be interested in seeing how the judgings were made, i.e. whether they were sorted by picture or by taste. My guess is that they were sorted by image only, because that's the only way the craft beer cluster makes sense. There are pale ales and stouts all grouped together,

Chapter 12

Multiple Factor Analysis

12.1 Intro to MFA

MFA allows us to compute disjunct factor scores from multiple tables, or multiple blocks within a table, which describe various bits of information on a single set of observations. Essentially it works by computing a PCA on each of the matrices to get partial factor scores for each matrix, and then combining all of the tables to compute global factor scores for the observations, which we then plot, PCA style, using the principal components as axes. In this regard, it is very similar to STATIS, which is discussed in the previous chapter as the basis for DiSTATIS.

The balancing done by the technique is called weighting, and the weights are distributed such that the variables that are closer to the barycenter are weighted more heavily than those that are further away. This has important implications for how we approach this specific dataset. Specifically, the weights are determined by scaling the tables such that their first singular values are equal to 1. To paraphrase Ju-Chi: "MFA normalization is equal rights. We don't ignore people because they're small". What this does is reduces the inertia of the table that has the most, whether that be because of a larger number of variables or otherwise, such that it is scaled to the same level as the table that has the least inertia. After the initial PCA, each original dataset is scaled such that each item in the table is normalized by a value equal to one divided by the first singular value of the table. This has the effect of scaling the singular values of each of the tables to one. This allows for a more effective, intuitive, and accurate comparison of data that otherwise would be unintelligible.

Check out This article (Hervé Abdi, Williams, and Valentin 2013) for more information on the details of MFA.

12.1.1 Strengths & Weaknesses

Strengths
- Allows you to break down large data sets into constituent groups.
- If you have groups of variables in a certain table that are related, not necessarily that measure the same thing, but things like personality variables or socioeconomic variables or education variables, you can see how those affect the observations as a whole.
- If you have a group of variables or even a single variable that is driving the majority of the inertia, this allows you to scale the inertia of the various tables such that they are all comparable.

Weaknesses
- Because of the inherent visual complexity of the factor maps, it may be necessary to look at the graphs in terms of group means, which may or may not be helpful or useful in terms of analysis. (That being said,

it's possible that the plots with the partial factor scores will actually show linear trends (remember Sabina's plot).)

12.1.2 Dos and Don'ts

Do:
- Make sure that the structure of your data make sense. If the variables for each of the tables aren't all describing the same observations, then an MFA won't make sense.

Don't:
- Confuse the Rv matrix and the weighting. The Rv Matrix is a measure of similarity between the individual matrices and the weights are determined by the first singular value of each matrix.
- If you're going to analyze group means, don't calculate the group means from the original variables. Run the analysis first, then find the group means of the factor scores, to plot those. This preserves the original data and provides a more accurate result.

Research Questions
Questions for this technique should be guided by the fact that we're breaking down multiple tables, looking for differences between tables/datasets/matrices that contribute to our global/overall group analyses.
- What can the disjunct factor scores tell us about what the principal components represent?
- Can we identify trends based on how different groups of variables (or different tables/matrices, etc.) are plotting in the factor space?
- Do the subtables vary systematically based on our divisions of information/variables?
- Are there individual observations, variables, or tables, that are driving a disproportionate amount of the variance, and is there anything interesting about that?

12.2 Data

As stated above, for this technique we need multiple datasets. What we're going to do is going to be similar to PLSC. Instead of adding new data to what we have already, we're going to break up the Music Features dataset into 3 smaller matrices, with 9, 10, and 9 of the original variables in each, respectively. Additionally, because the plots below would have so much information as to be incomprehensible were we to use all 1000 observations for the factor scores plot, we're just going to take the average of each genre and use those 10 genres as our observations. However, it would be possible to do the analysis using the original set of 1000 observations. In fact, I recommend not doing what I've done here (running the analysis on the group means of each variable) unless you have a deep understanding of the fundamental structure of your dataset. A better option might be to run analyses on each of the groups, or to break the table by groups and select a different analysis. I've only done this as an exercise, and you can look back at some of the other analyses in this cookbook to see how these results line up with those.

To select which variables are going to be assigned to which matrices, I went back and looked at the MCA and the DiCA techniques to see which variables loaded on which components systematically. These were then grouped into matrices to see if we could break down any of that original loading into smaller constituent parts.

- For the first matrix, we're going to use the variables that have consistently loaded on the 1st component of these analyses: Spectral Bandwidth (spec_b), roll off (r_o), and MFCCs 4, 6, 8, 10, 11, 12, and 13.

- For the second matrix, we're going to use the variables that have consistently loaded on the 2nd component of these analyses: Chroma (ch), RMSE, Spectral Centroid (spec_C), Zero Crossing Rate

(zcr), and MFCCs 1, 2, 3, 5, 7, and 9.

- For the third matrix, we're going to use the variables that have not consistently loaded on either component, or have had non-significant loadings. An interesting effect of MFA will allow the effect of these variables to be magnified relative to the others. BPM, Beats (b), and MFCCs 14 - 20, were assigned to a third matrix.

The three tables are shown below with three rows of observations from each table. The tables haven't been divided by observations also, I have just chosen to show different files in each table. Also, I've used different code to create these tables than the other ones. check out the rmd file for how we create these tables.

	spec_b	r_o	4	6	8	10	11	12	13
blues.00081.au	1956.61	4196.11	40.93	24.82	15.25	12.26	−15.23	14.34	−13.82
classical.00091.au	1911.99	4066.47	37.92	21.64	25.73	14.3	−7.56	10.47	−4.92
country.00013.au	2842.05	6062.66	19.95	0.73	−0.5	5.39	−3.73	−3.5	−5.33

	ch	rmse	spec_c	zcr	mfcc1	mfcc2	3	5	7	9
disco.00063.au	0.55	0.29	2583.28	0.1	−51.75	70.33	−3.92	1.59	−4.1	−9.71
hiphop.00034.au	0.47	0.23	3191.04	0.17	5.88	53.24	−8.98	0.44	1.65	4.98
jazz.00004.au	0.17	0.11	1039.62	0.05	−270.26	137.57	5.66	−7.95	−6.8	−6.19

	bpm	b	14	15	16	17	18	19	20
metal.00088.au	117.45	54	2.98	−5.84	3.94	−6	−1.97	2.39	−3.85
pop.00024.au	117.45	57	−1.98	1.92	0.08	−1.27	1.23	1.49	0.95
reggae.00009.au	161.5	76	3.12	−1.41	3.26	−4.1	2.94	0.38	0.01

12.3 Extra code we need

The code below allows us to pull a number of colors from the spectrum whenever we need new colors. It's a super useful function, and is used twice below.

```
colfunc <- colorRampPalette(c("firebrick4","gold","forestgreen","darkblue"))
```

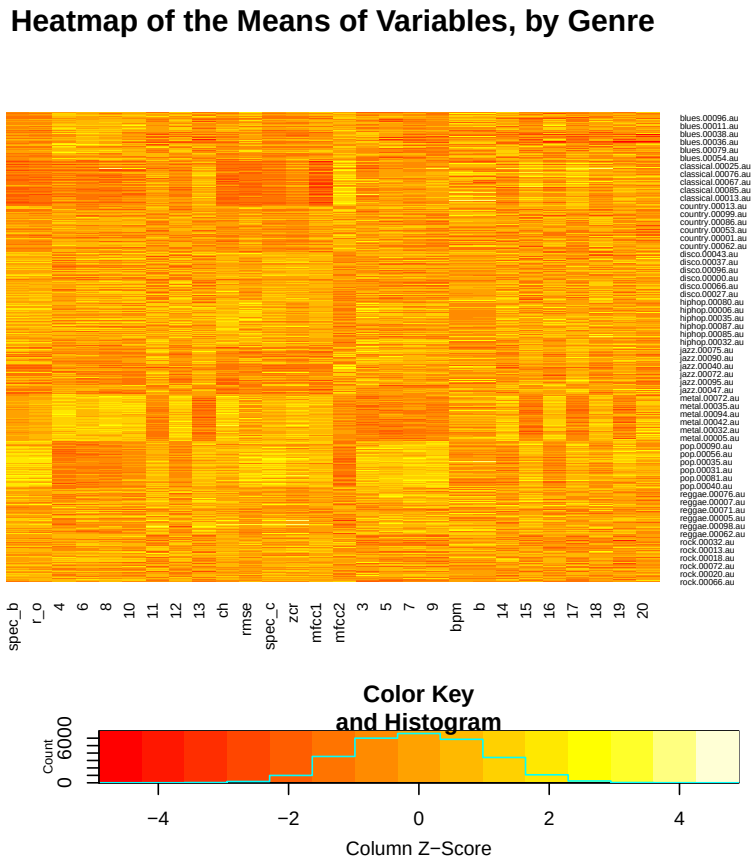
This function is a little shortcut that we need when we're making our multiple factor scores plot. If we don't have the dimensions of each of the tables named the same, ggplot kicks an error saying that it can't find the right thing.

```
renameCols <- function(x){
   colnames(x) <- paste("Dimension", 1:ncol(x))
   x
}
```

12.4 Data Visualization

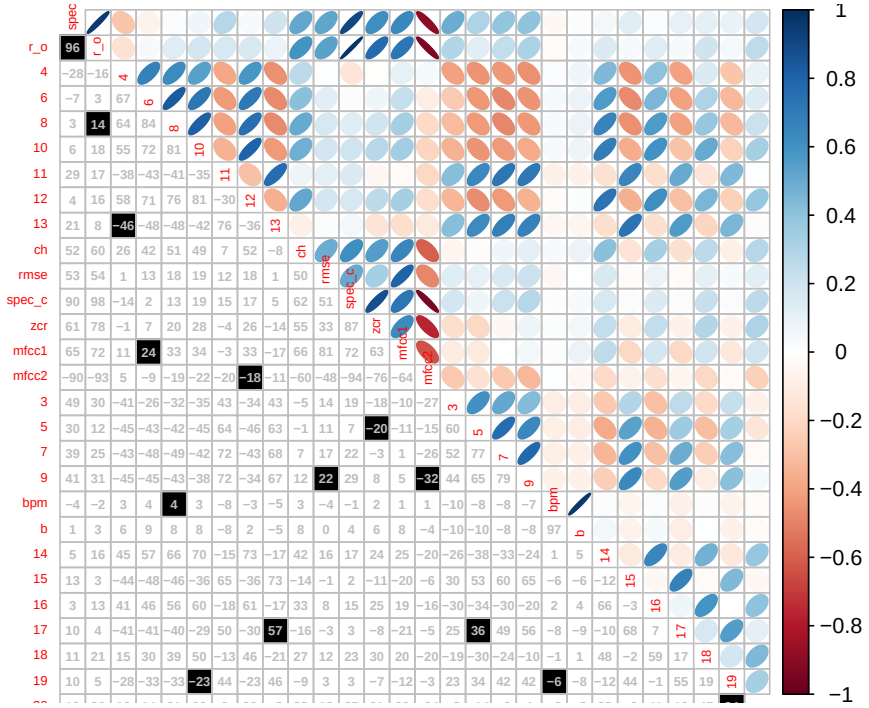
12.4.1 Heatmap

This heatmap shows us our heatmap of reordered data using the genre means for each variable. Gives us a good idea of structural contrast between the variables and observations. A couple of things worth noting: Metal and Pop genres seem to be incredibly different across the majority of our variables, and classical seems to also differ greatly from most genres. This is consistent with what we've up with all of the other analyses we've done so far.



12.4.2 Correlation Plot

The correlation plot below is again similar to those we've seen. This is useful for visualizing our data. If you've sen the other analysis, you'll notice that this one has the variables grouped by the matrix of which they are a part. We use `corrplot` like we've done before, check out the rmd file for the code.



12.5 Analysis

The code below is the actual MFA (`mpMFA`). The code above the analysis sets up a matrix where the variables are the column names, so that it aligns with column names for `mfdatareo`. The single row of the matrix is then filled "M1", "M2", and "M3" to indicate which matrix the variables belong to.

```
mfagroups <- data.frame(matrix(nrow = 1, ncol = 28))
row.names(mfagroups) <- "group"
colnames(mfagroups) <- colnames(mfdata)
mfagroups[, c(6,7, 12, 14, 16, 18:21)] <- "M1"
mfagroups[, c(3:5, 8:11, 13, 15, 17)] <- "M2"
mfagroups[, c(1,2,22:28)] <- "M3"
mfaresults <- mpMFA(mfdatareo, mfagroups,
                    DESIGN = music.genre, graphs = FALSE)
```

```
## [1] "Preprocessed the Rows of the data matrix using:  None"
## [1] "Preprocessed the Columns of the data matrix using:  Center_1Norm"
## [1] "Preprocessed the Tables of the data matrix using:  MFA_Normalization"
## [1] "Preprocessing Completed"
## [1] "Optimizing using:  None"
## [1] "Processing Complete"
```

12.6 Results

12.6.1 Rv Matrix

The Rv Matrix is the matrix of Rv values. The way we interpret the Rv coefficients it that it measures the level of similarity between two matrices. Although not technically (computationally) precise, we can view these values as the non-centered squared correlation coefficients. Each of the squares in the plot below approximates the shared variance between the two matrices for which the square in the matrix corresponds.

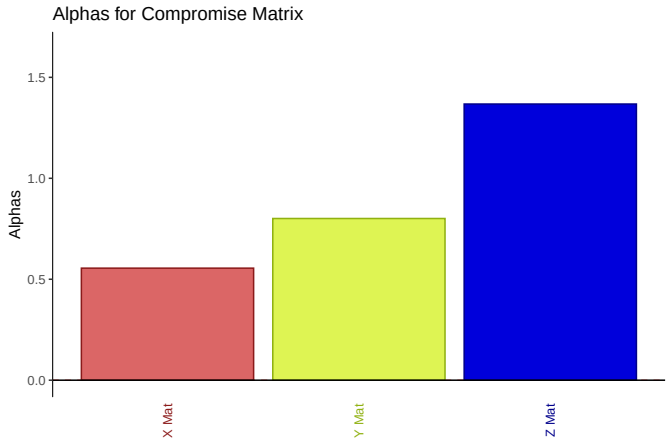
Check out the code below for how we create this matrix.

```
matnames <- c("X Mat", "Y Mat", "Z Mat") # The matrix doesn't supply names
# Check out what's in this next line to see how it creates the
RVmat <- mfaresults$mexPosition.Data$InnerProduct$RVMatrix
rownames(RVmat) <- matnames # These two lines assign the names of the matrices
colnames(RVmat) <- matnames # To both rows and columns
corrplot(RVmat, method = "color", addCoefasPercent = TRUE,
         tl.srt = 45, addCoef.col = "white")
```

12.6.2 Alphas

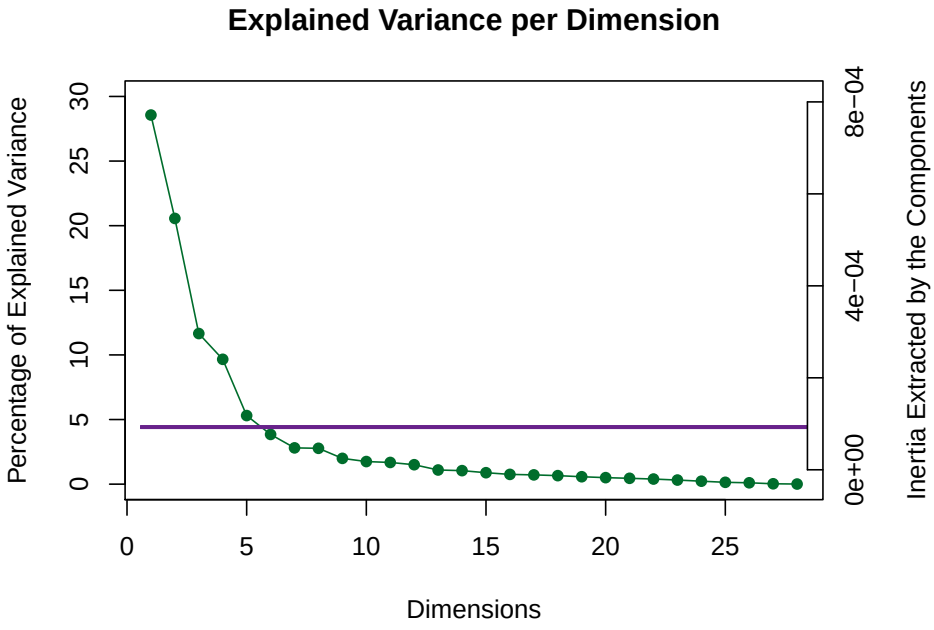
The alphas are the values that reflect the weights for the compromise matrix. The alpha value for each matrix is the calculated as 1 divided by the square root of the first singular value of the matrix. Because we're giving more weight to the matrix with more average values, the Z Matrix, which consists of the variables that haven't significantly loaded on our first principal components yet, gets the most weight, while the other two matrices, which consist of the variables that have been loading significantly, are weighted less.

```
col4alphas <- colfunc(3)
Eig.tab <- mfaresults$mexPosition.Data$Compromise$compromise.eigs
Alpha <- as.data.frame(1/sqrt(Eig.tab))
row.names(Alpha) <- matnames
alphmat <- PrettyBarPlot2(Alpha,
                          threshold = 0,
                          font.size = 3,
                          color4bar =  col4alphas,
                          main = "Alphas for Compromise Matrix",
                          ylab = "Alphas",
                          ylim = c(0, 1.2*max(Alpha)),
                          )
alphmat
```



12.6.3 Scree Plot

The scree plot below shows us the eigenvalues, as before. The dimensionality of this data set is the minimum of the number of columns of any of the matrices included in the analysis (in this case, 9). Although the Kaiser criterion (the average value of the eigenvalues) cuts out the third dimension, it looks like there would be three clear dimensions that explain a large amount of variance using the "elbow test", so we'll just focus on those. No permutations this time, but as a reminder, that only tells us whether or not the values are significant, not whether or not they're interesting. See the chapter on PCA for more on reading this plot.



12.7 Factor Scores Plots

The code and descriptions for both the global and the partial factor scores maps are shown first, then the two plots are shown next to each other below.

12.7.1 Global Factor Scores Plots

Factor scores plot! With this analysis, unlike previous analyses, we're only working with the group means for the variables, so we only have 10 total observations. Those are plotted on the principal components as we did for PCA. as you can see, the plot looks remarkably similar to the analyses we have done before, with pop, metal, and classical driving the first and second principal components.

Notice in the code for `createFactorMap`, we've specified the constraints manually. This is because if we don't, the constraints cut out some of the partial factor scores once we plot those. This might not happen every time, but if there are errors in the partial factor scores plot (# values omitted, missing information), it might be because they fall outside of the constraints specified here. There's also no way to specify constraints in the `createPartialFactorScoresMap` function, so we do it here.

```
fimeans <- getMeans(mfaresults$mexPosition.Data$Table$fi, music.genre)
fimeans <- renameCols(fimeans)

d1 <- getMeans(mfaresults$mexPosition.Data$Table$partial.fi.array[,,1], music.genre)
d2 <- getMeans(mfaresults$mexPosition.Data$Table$partial.fi.array[,,2], music.genre)
d3 <- getMeans(mfaresults$mexPosition.Data$Table$partial.fi.array[,,3], music.genre)

pfi4pfs <- abind(d1, d2, d3, along = 3)
pfi4pfs <- renameCols(pfi4pfs)

rownames(pfi4pfs) <- unique(music.genre)
dimnames(pfi4pfs)[[3]] <- c("M1", "M2", "M3")

MFA_FMap <- createFactorMap(fimeans,
                            col.points = unique(mfaresults$Plotting.Data$fi.col),
                            col.labels = unique(mfaresults$Plotting.Data$fi.col),
                            alpha.points = .6, pch = 17, cex = 5,
                            display.labels = TRUE,
                            constraints = minmaxHelper4Partial(
                                            FactorScores = fimeans,
                                            partialFactorScores = pfi4pfs)
                           )
label4Map <- createxyLabels.gen(1,2,
                                lambda = mfaresults$mexPosition.Data$Table$eigs,
                                tau = mfaresults$mexPosition.Data$Table$t,
                                axisName = "Dimension ")

a003.mfa <- MFA_FMap$zeMap + label4Map
```

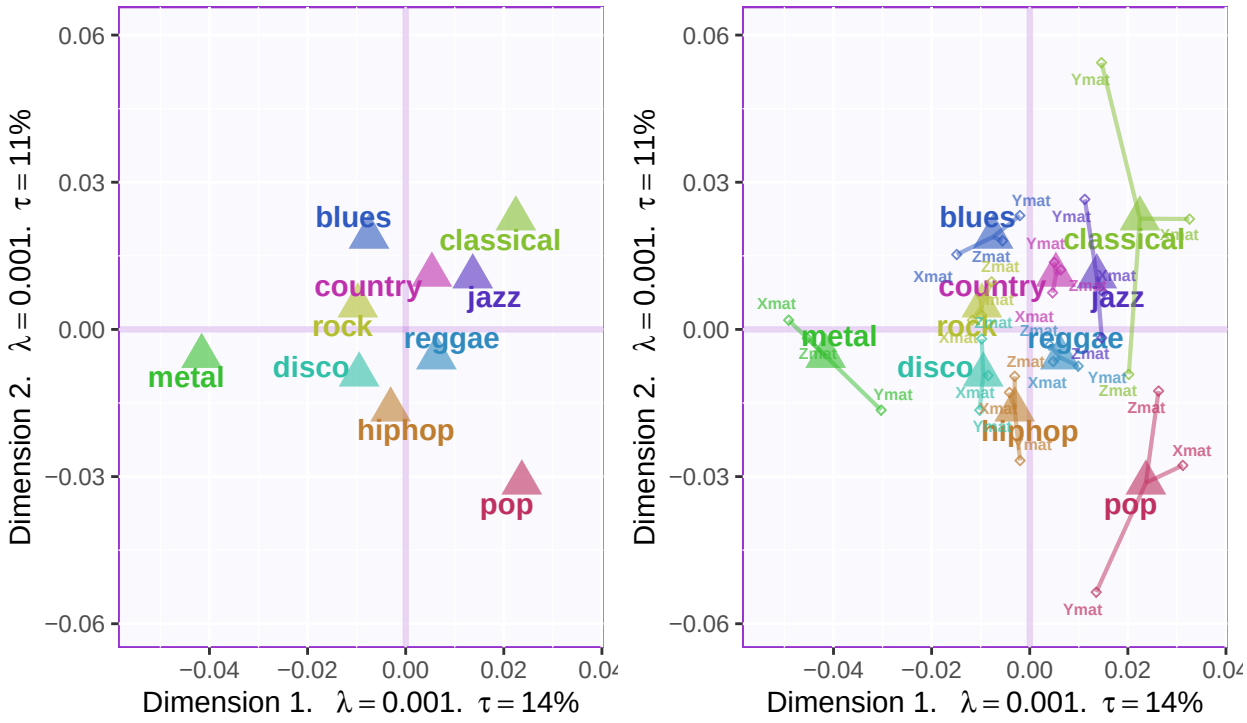
12.7.2 Partial Factor Scores Plot

Here we have the partial factor scores map. it shows us where each of the tables for each of the variables falls on the map and how they relate to the global factor scores we plotted in the previous plot.

In order to get this to work, you need to make sure that all of the dimensions of each of the matrices all match, otherwise ggplot will kick an error telling you it can't find which dimension you're trying to specify. So get around that, we use the function `renameCols` from above and assign the data to other variables to make our code for our partial factor scores map a little cleaner. Also, notice that we're using the `createPartialFactorScoresMap` instead of the `createFactorMap` that we've used for the rest of these

analyses.

```
map4PFS <- createPartialFactorScoresMap(
                          factorScores = fimeans, # renamed in previous chunk
                          partialFactorScores = pfi4pfs,
                          axis1 = 1, axis2 = 2,
                          colors4Items = unique(mfaresults$Plotting.Data$fi.col),
                          names4Partial = c("Xmat", "Ymat", "Zmat"),
                          font.labels = 'bold'
                                        )
partialfsmap <- MFA_FMap$zeMap + label4Map +
                map4PFS$linesColByItems + map4PFS$pointsColByItems +
                map4PFS$labelsColByItems
```



##

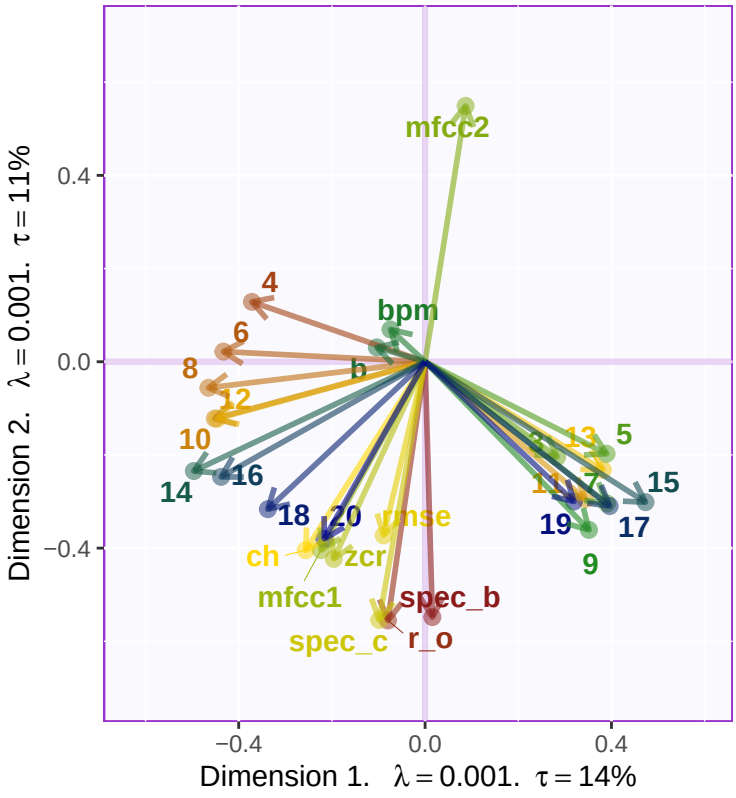
Loadings & Contributions

12.7.3 Loadings

Here are the loadings for each of the variables. The MFA Normalization process seems to have adjusted how much each of these variables loads relative to one another. Before, for example in PCA, we saw that a few of the variables drove the majority of the loadings, and the variables loaded almost all the way to the edge of the correlation circle. In the case below, check out the constraints of the graph. Besides b and bpm, thanks to MFA normalization the variables have balanced out a little bit, with no one variable positioned that much further from the barycenter of the graph.

```
load.vars <- mfaresults$mexPosition.Data$Table$Q
col4cols <- colfunc(28)
loading.plot <- createFactorMap(load.vars,
```

```
                                col.points = col4cols,
                                col.labels = col4cols,
                                )
LoadingMap <- loading.plot$zeMap +
              addArrows(load.vars, color = col4cols) +
              label4Map
LoadingMap
```



12.7.4 Contributions

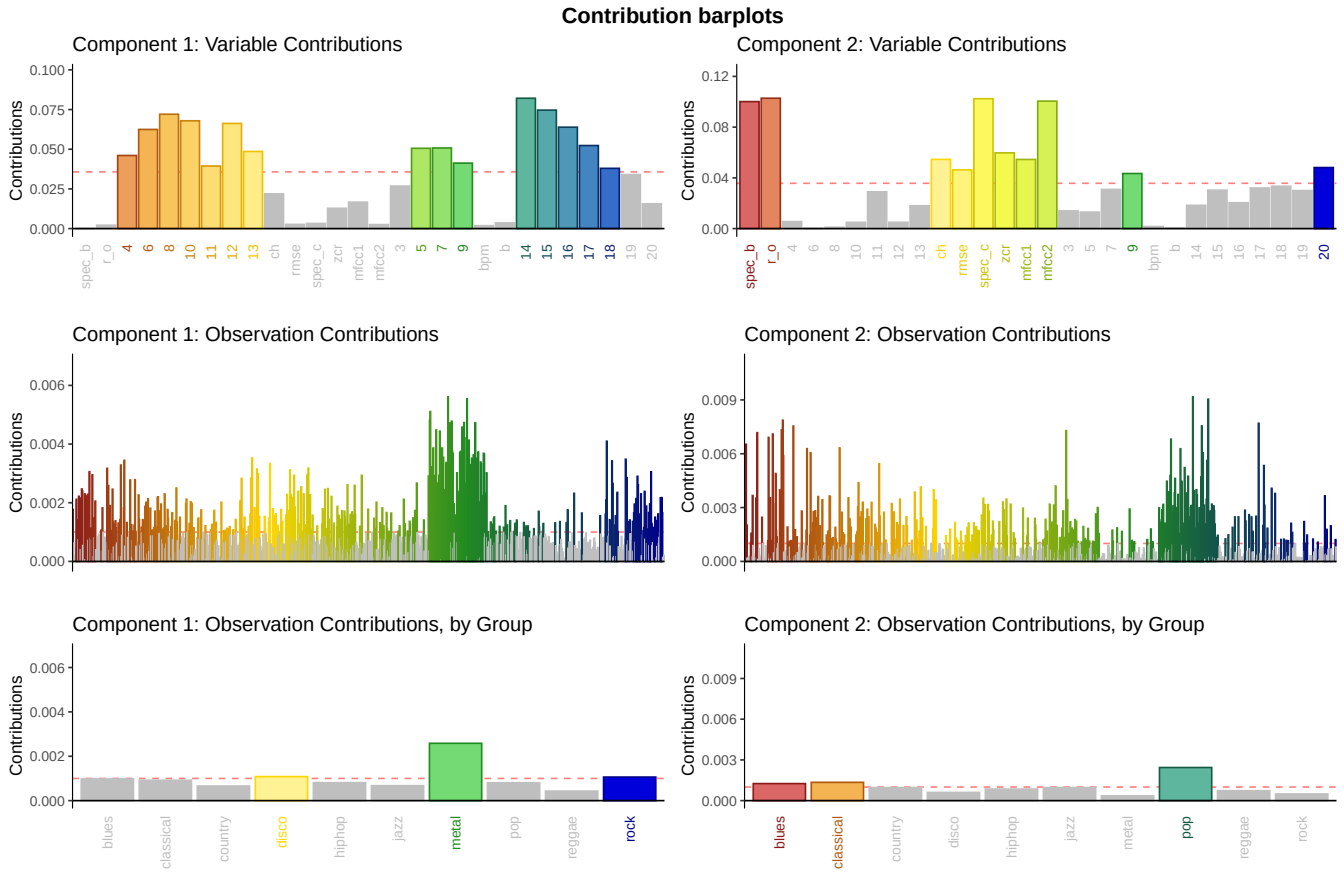
The contributions below show us how much each of the variables and each of the genres contribute to each of the dimensions. For the genres it's pretty obvious which genres load on which dimension, we can see from the factor scores plot. As per the factor scores plot, metal, pop, and classical drives the first dimension, and pop and classical, and to a lesser extent, blues and hip-hop drive the second dimension. The rest of the genres don't seem to load significantly on either dimension.

Looking at the contributions for the variables, it looks like we've finally brought out something from the higher MFCCs, but also we're looking at them loading on the same dimension. Note that this specific plot doesn't account for the sign of the loadings, and are simply presented by their magnitude.

Like the previous chapters of this book, the code for the first plot is shown, check out the rmd file for the rest of the plots, including the arrangement code.

```
# These first two lines get us colors for our observation contributions
col4ci <- colfunc(1000) # For the individual observations
col4cim <- colfunc(10) # For the group means
colconts1 <- PrettyBarPlot2(
```

```
                    bootratio = mfaresults$mexPosition.Data$Table$cj[,1],
                    threshold = 1 / NROW(mfaresults$mexPosition.Data$Table$cj),
                    font.size = 3,
                    color4bar = col4cols, # we need hex code
                    main = 'Component 1: Variable Contributions',
                    ylab = 'Contributions',
                    ylim = c(0, 1.2*max(mfaresults$mexPosition.Data$Table$cj[,1]))
                    )
```



12.7.5 Conclusions

There are a few interesting things that occur thanks to the MFA that we haven't seen before. It shows us, similarly to DiCA, how the groups of variables combine to create the factor scores we see for these observations. However, we can also see something that may have only been visible in the heatmap before. It definitely shows us which genres have more average spectral content. Contrasting disco with pop or classical is a great example. Where classical and pop have values from the three different groups of variables that are pulling them away from the center, all of disco's partial factor scores are pretty tight in on the barycenter.

Otherwise, our factor scores see some interesting effects. Where Spectral Bandwidth and Roll Off have been loading on dimension one (and dimension two to a certain extent), we're now seeing them only loading on dimension two. It looks like the only variables that are still insignificant are b, bpm, and MFCCs 3, 11, and 18. What we can learn from that is that those are likely the spectral components that are common to all of the genres and therefore don't contribute much to the dimensionality of the genre identification space.

Bibliography & Appendices

Chapter 13

Bibliography

Abdi, Hervé. 2007a. "Discriminant Correspondence Analysis." In *Encyclopedia of Measurement and Statistics*, edited by Neil Salkind. Thousand Oaks: Sage. https://doi.org/10.4135/9781412952644.n140.

———. 2007b. "Singular and Generalized Singular Value Decomposition." In *Encyclopedia of Measurement and Statistics*, edited by Neil Salkind, 1–14. Thousand Oaks: Sage. https://doi.org/10.4135/9781412952644.n413.

———. 2010. "Partial least squares regression and projection on latent structure regression (PLS Regression) Wiley Interdisciplinary Reviews: Computational Statistics Volume 2, Issue 1." *Wiley Interdisciplinary Reviews: Computational Statistics* 2 (1): 97–106. https://doi.org/10.1002/wics.051.

———. 2018. "Multiple correspondence analysis." In *Multiple Correspondence Analysis*, 31–55. Thousand Oaks: Sage. https://doi.org/10.4324/9781315516257-3.

Abdi, Hervé, and Michel Béra. 2018. "Correspondence Analysis." Springer Verlag. https://doi.org/10.1007/978-3-642-04898-2_195.

Abdi, Hervé, and Lynne J. Williams. 2010a. "Barycentric Discriminant Analysis (BADIA)." Edited by Neil Salkind. Thousand Oaks: Sage.

———. 2010b. "Correspondence Analysis." Edited by Neil Salkind. Thousand Oaks: Sage.

———. 2010c. "Principal component analysis Tutorial Review." *Wiley Interdisciplinary Reviews: Computational Statistics* 2 (August): 1–16.

———. 2013. "Partial Least Squares Methods: Partial Least Squares Correlation and Partial Least Square Regression." Edited by Brad Reisfeld and Arthur N. Mayeno. Springer Science+Business Media, LLC. https://doi.org/10.1007/978-1-62703-059-5.

Abdi, Hervé, Lynne J. Williams, and Michel Béra. 2018. "Barycentric Discriminant Analysis."

Abdi, Hervé, Lynne J. Williams, and Domininique Valentin. 2013. "Multiple factor analysis: Principal component analysis for multitable and multiblock data sets." *Wiley Interdisciplinary Reviews: Computational Statistics* 5 (2): 149–79. https://doi.org/10.1002/wics.1246.

Abdi, H., A. J. O'Toole, D. Valentin, and B. Edelman. 2006. "DISTATIS: The Analysis of Multiple Distance Matrices," 42–42. https://doi.org/10.1109/cvpr.2005.445.

Berry, Kenneth J., Janis E. Johnston, and Paul W. Mielke. 2011. "Permutation methods." *Wiley Interdisciplinary Reviews: Computational Statistics* 3 (6): 527–42. https://doi.org/10.1002/wics.177.

Hesterberg, Tim. 2011. "Bootstrap." *Wiley Interdisciplinary Reviews: Computational Statistics* 3 (6): 497–526. https://doi.org/10.1002/wics.182.

Krishnan, Anjali, Lynne J. Williams, Anthony Randal McIntosh, and Hervé Abdi. 2011. "Partial Least Squares (PLS) methods for neuroimaging: A tutorial and review." *NeuroImage* 56 (2): 455–75. https://doi.org/10.1016/j.neuroimage.2010.07.034.

Chapter 14

Mel-Frequency Cepstral Coefficients

Because it's important to understand the variables in order to understand the story the data are telling, I've included the discussion and links below to help guide our understanding of Mel-Frequency Cepstral Coefficients.

14.1 What are they?

Essentially, Mel-Frequency Cepstral Coefficients (MFCCs) measure how well the audio we hear from a recording fits into certain spectral bins. They are a useful tool for detecting spoken words in audio files, and there have been other interesting uses for them that have been found recently.

14.2 How do we get them?

The majority of the information I've for the MFCCs comes from the LibROSA documentation and a great online resource found here.

According to the libROSA documentation, the output for the MFCC function is a vector of 20 values, each representing the discrete cosine transform (DCT) of the mel-frequency decomposition of the audio signal in a short time window of the audo file. The number of values depends on the number of triangular windows the decomposer elects to do. The default for the function in the libROSA documentation is 20. These windows are set over the spectrum domain, not the time domain, and allow you to analyze the power of the spectrum in a given mel-frequency bin. The mel-frequencies are a mathematical analogue for how we hear (effectively, the log of the frequency), so the mel-frequency bins help us to analyze the frequency in terms of what we hear, not just in terms of the signal. Now, because these triangular windows overlap with the windows adjacent, they need to be decorrelated using the DCT so that they're only capturing the information that's unique to the individual window.

Steps to Calculating MFCCs from Practical Cryptography's MFCC Tutorial

- First you divide the audio file into short time windows, each one between 20 - 40 ms. Too short ($<$ 10 ms) and there isn't enough information present to get an accurate read on the spectral content of the window. Too long ($>$ 40 ms), and there will be too much noise to accurately determine what the spectral data are for that window.
- After you divide the audio into the time windows, you run a spectral analysis, calculating the periodogram of the power spectrum
- Apply the triangular mel filterbank and sum the energy in each filter, with the first filterbank at 0

- Take the discrete cosine transform of the log of the filterbank energies
- Once that's done, you're left with a vector of MFCCs for a given time slice of the audio file that represents the relative power of each mel-frequency spectral component in that slice. From there, the spectral power for each slice is averaged across the time domain, so that (for example) we get the average power for all of the MFCCs in the entire file. What this results in is a big picture idea of where the tonal energy is in the file. There is one value for each spectral window, hence 20 MFCCs.