MDS Visualization

Danielle Daidone and Ryan Lidster

4/11/26

This script creates visualizations of MDS results using dimension scores as coordinates. Because the initial coordinates are often hard to interpret, it also aids in rotating the solution so that you can find the one that makes the most sense visually. Matrix rotation code is from Brian Shalloway's blog post here: https://www.bryanshalloway.com/2020/02/20/visualizing-matrix-transformations-with-gganimate/#fn9

Note that rotation does not change the distances between points, and thus the Euclidean distances results from MDS_Analysis_new.Rmd will be unaffected. However, if you want to correlate dimension scores with acoustic or phonological properties, you should use the desired rotation for these analyses. The rotated dimension scores are saved at the bottom of this script.

Because of the length of the script, I recommend collapsing all the code chunks for easy of readability. Go to Edit –> Folding –> Collapse All. Alternately, the short cut is Alt + O (the letter).

Download and load relevant packages

```
## Loading required package: tidyverse
```

```
## Warning: package 'ggplot2' was built under R version 4.5.3
```

```
## Warning: package 'readr' was built under R version 4.5.3
```

```
## Warning: package 'purrr' was built under R version 4.5.3
```

```
## Warning: package 'dplyr' was built under R version 4.5.3
```

```
## Warning: package 'lubridate' was built under R version 4.5.3
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.2.1     v readr     2.2.0
## v forcats   1.0.1     v stringr   1.6.0
## v ggplot2   4.0.2     v tibble    3.3.1
## v lubridate 1.9.5     v tidyr     1.3.2
## v purrr     1.2.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
## Loading required package: ggrepel
```

```
## Warning: package 'ggrepel' was built under R version 4.5.3
```

Set working directory - *Change to match your filepath*

```
knitr::opts_knit$set(root.dir = "C:/Users/ddaid/OneDrive - UNC-Wilmington/CV/Website/FC analysis info/Ou
```

OPEN MDS DATA

Make sure you run the appropriate chunks below for visualization and rotation based on whether you pull in a 2D or 3D solution.

```
# read in the MDS dimension scores and view the file
MDS_points = read.table("MDS_3D_AllContexts.txt",sep="\t",header=TRUE)

view(MDS_points)
```

CREATE COLUMNS FOR SPEAKER AND CONDITION

Our row names are coded as condition_speaker, e.g., CVCV_M. Adjust as necessary. For additional manipulations, check out https://stringr.tidyverse.org/.

```
# creates a column that matches the row names
MDS_points$split = rownames(MDS_points)

# splits the column into two called "condition" and "speaker" based on the delimiter
MDS_points = separate_wider_delim(MDS_points, cols=split, delim="_", names=c("condition", "speaker"))

# Examples of other manipulations:

# make all condition names uppercase
# MDS_points$condition = str_to_upper(MDS_points$condition)

# replace condition name with another, e.g., with correct IPA symbol
# MDS_points$condition = str_replace_all(MDS_points$condition, "ae", "æ")
```

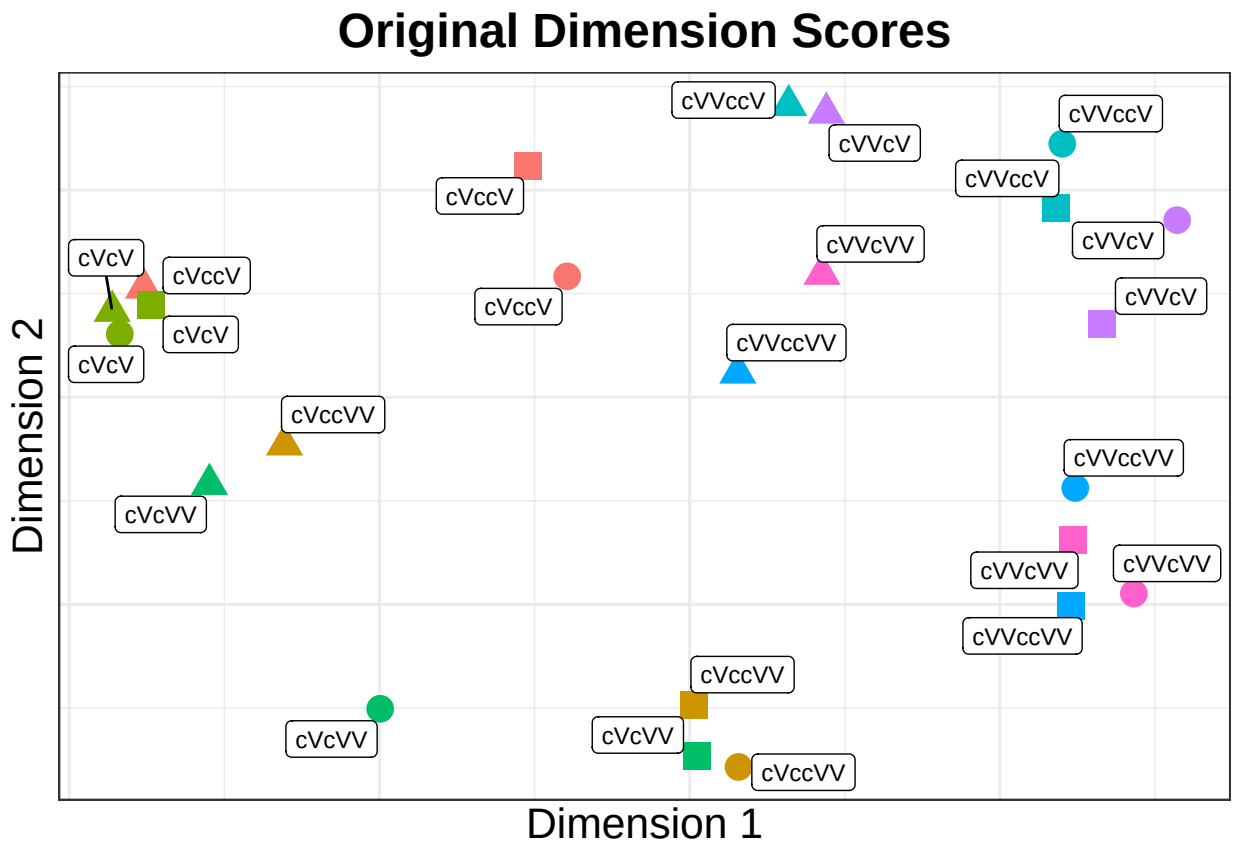
MDS VISUALIZATION

2D + 3D: CREATE MDS PLOT

Creates a plot with Dim 1 on x-axis and Dim 2 on y-axis

```
# sets the format of the plot to black and white
theme_set(theme_bw())

# indicate the data and the columns with the data you want to plot
# ggplot(data, aes(x = dimension 1, y = dimension 2))
mds_plot_1by2 <- ggplot(MDS_points, aes(x=D1, y=D2, shape=speaker, color=condition, size = 7)) +

  # make a scatterplot
  geom_point() +

  # add labels for points; if you want rownames as labels, use label=rownames(MDS_points)
  geom_label_repel(aes(label=condition), color="black", size=3) +
```

```
  # makes the labels for the title of the graph and x and y axis
  labs(
    title="Original Dimension Scores",
       x="Dimension 1",
       y="Dimension 2")



# change the look of the plot
mds_plot_1by2 <- mds_plot_1by2 + theme(

  # for title, change font size, make it bold, centered, and change line height
  plot.title=element_text(size=18,
                          face="bold",
                          hjust=0.5,
                          lineheight=1.2),

  # remove legend
  legend.position = "none",

  # change size of x axis title
  axis.title.x=element_text(size=16),

  # change size of y axis title
  axis.title.y=element_text(size=16),

  # remove x axis text
  axis.text.x=element_blank(),

  # remove y axis text
  axis.text.y=element_blank(),

  # remove x axis ticks
  axis.ticks.x=element_blank(),

  # remove y axis ticks
  axis.ticks.y=element_blank())

  # change size of text for x axis labels
  # axis.text.x=element_text(size=12, face="bold"),

  # change size of text for y axis labels
  # axis.text.y=element_text(size=12,face="bold"))

# plot figure
plot(mds_plot_1by2)
```

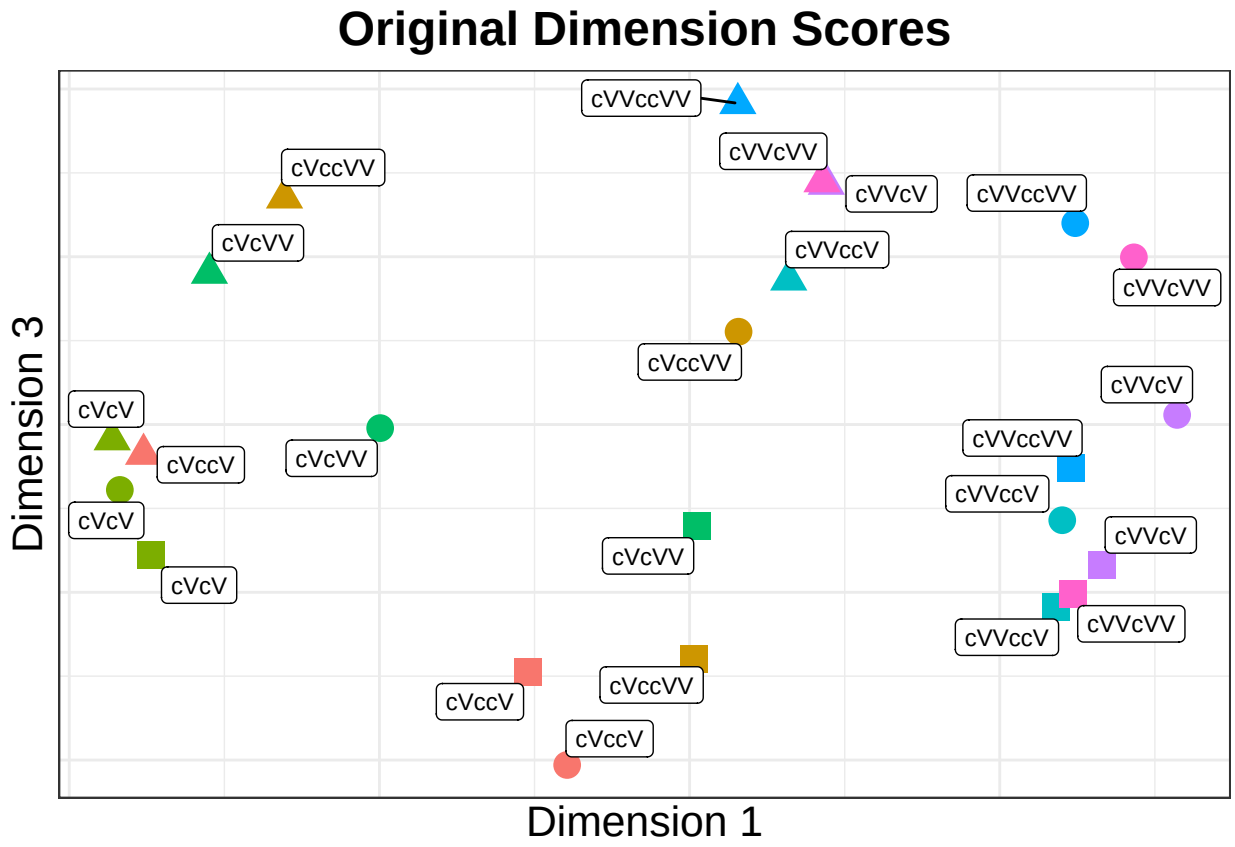


3D only: CREATE ADDITIONAL MDS PLOT

Creates a plot with Dim 1 on x-axis and Dim 3 on y-axis

```
# sets the format of the plot to black and white
theme_set(theme_bw())

# indicate the data and the columns with the data you want to plot
# ggplot(data, aes(x = dimension 1, y = dimension 2))
mds_plot_1by3 <- ggplot(MDS_points, aes(x=D1, y=D3, shape=speaker, color=condition, size = 7)) +

  # make a scatterplot
  geom_point() +

  # add labels for points; if you want rownames as labels, use label=rownames(MDS_points)
  geom_label_repel(aes(label=condition), color="black", size=3) +

  # makes the labels for the title of the graph and x and y axis
  labs(
    title="Original Dimension Scores",
       x="Dimension 1",
       y="Dimension 3")



# change the look of the plot
mds_plot_1by3 <- mds_plot_1by3 + theme(
```

```
  # for title, change font size, make it bold, centered, and change line height
  plot.title=element_text(size=18,
                          face="bold",
                          hjust=0.5,
                          lineheight=1.2),

  # remove legend
  legend.position = "none",

  # change size of x axis title
  axis.title.x=element_text(size=16),

  # change size of y axis title
  axis.title.y=element_text(size=16),

  # remove x axis text
  axis.text.x=element_blank(),

  # remove y axis text
  axis.text.y=element_blank(),

  # remove x axis ticks
  axis.ticks.x=element_blank(),

  # remove y axis ticks
  axis.ticks.y=element_blank())

  # change size of text for x axis labels
  # axis.text.x=element_text(size=12, face="bold"),

  # change size of text for y axis labels
  # axis.text.y=element_text(size=12,face="bold"))

# plot figure
plot(mds_plot_1by3)
```



ROTATE SOLUTION

Code pulled from https://github.com/brshallo/brshallo/blob/master/content/post/2020-02-20-visualizing-matrix-transformations-with-gganimate.Rmd

CREATE A FUNCTION FOR MATRIX ROTATION

`transform_df_coords()`: Given dataframe, column names of coordinates, and a transformation matrix, return dataframe with transformed coordinates.

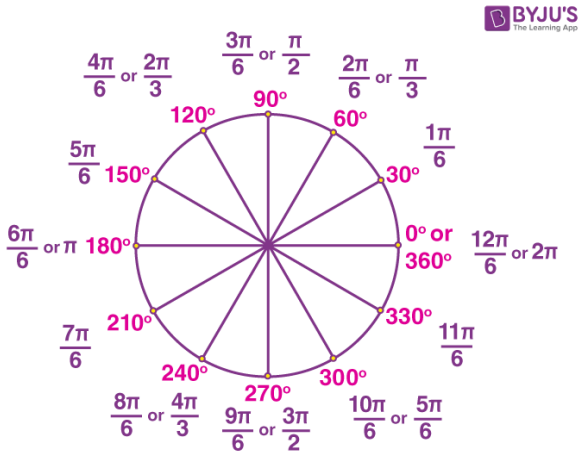
`transform_df_coords()` is just matrix multiplication, but facilitates applying matrix transformations on a dataframe where each row (in specified columns) represents a vector / coordinate point

```
transform_df_coords <- function(df, ..., m = diag(length(df))){

  df_names <- names(df)

  df_coords <- df %>%
    select(...)

  df_coords_names <- names(df_coords)

  df_matrix <- df_coords %>%
    as.matrix() %>%
    t()

  df_coords_new <- (m %*% df_matrix) %>%
    t() %>%
```

```
    as_tibble() %>%
    set_names(df_coords_names)

  df_other <- df %>%
    select(-one_of(df_coords_names))

  bind_cols(df_coords_new, df_other) %>%
    select(all_of(df_names))
}
```

EXPLANATION OF ROTATION

Rotation is calculated in radians, so you need to input your desired degree of rotation using pi. For example, pi/2 is a 90º rotation, pi is a 180º rotation, and (3 * pi)/2 is a 270º degree rotation. See the image for equivalents between degrees and fractions with pi: https://byjus.com/maths/relation-between-degree-and-radian/



-2D solutions can only be rotated along the z-axis like a clock on a wall (i.e. no rotation can be coming at you since 2D is only a flat surface)

-3D solutions can be rotated along the x-axis, y-axis, or z-axis

Visual for rotations along each axis:

https://jsfiddle.net/greggman/Lqg93y1v/

- x rotates down in front towards you
- y rotates counterclockwise like a spinning top on a table
- z rotates counterclockwise like on a wall

3D ROTATION

Note that the rotations are based on the original dimension scores. If you want to rotate your solution along more than one axis, change MDS_points to the output of the latest rotation (e.g., MDS_points_rotated).

3D only: DEFINE TRANSFORMATION MATRIX - x rotation

```
# input your desired angle of rotation using pi
x_rotation = pi / 2

# create transformation matrix
transformation_matrix <- tribble(~ x, ~ y, ~ z,
                                 1, 0, 0,
                                 0, cos(x_rotation), -sin(x_rotation),
                                 0, sin(x_rotation), cos(x_rotation)) %>%
  as.matrix()

# create new dimension scores based on multiplying by transformation matrix (i.e. rotation)
MDS_points_rotated <- transform_df_coords(MDS_points, D1, D2, D3, m = transformation_matrix)
```

```
## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if
## `.name_repair` is omitted as of tibble 2.0.0.
## i Using compatibility `.name_repair`.
## This warning is displayed once per session.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

3D only: DEFINE TRANSFORMATION MATRIX - y rotation

```
# input your desired angle of rotation using pi
y_rotation = pi/2

# create transformation matrix
transformation_matrix <- tribble(~ x, ~ y, ~ z,
                                 cos(y_rotation), 0, sin(y_rotation),
                                 0, 1, 0,
                                 -sin(y_rotation), 0, cos(y_rotation)) %>%
  as.matrix()

# create new dimension scores based on multiplying by transformation matrix (i.e. rotation)
MDS_points_rotated <- transform_df_coords(MDS_points, D1, D2, D3, m = transformation_matrix)
```

3D only: DEFINE TRANSFORMATION MATRIX - z rotation

```
# input your desired angle of rotation using pi
z_rotation = pi/2

# create transformation matrix
transformation_matrix <- tribble(~ x, ~ y, ~ z,
                                 cos(z_rotation), -sin(z_rotation), 0,
                                 sin(z_rotation), cos(z_rotation), 0,
                                 0, 0, 1) %>%
  as.matrix()

# create new dimension scores based on multiplying by transformation matrix (i.e. rotation)
MDS_points_rotated <- transform_df_coords(MDS_points, D1, D2, D3, m = transformation_matrix)
```

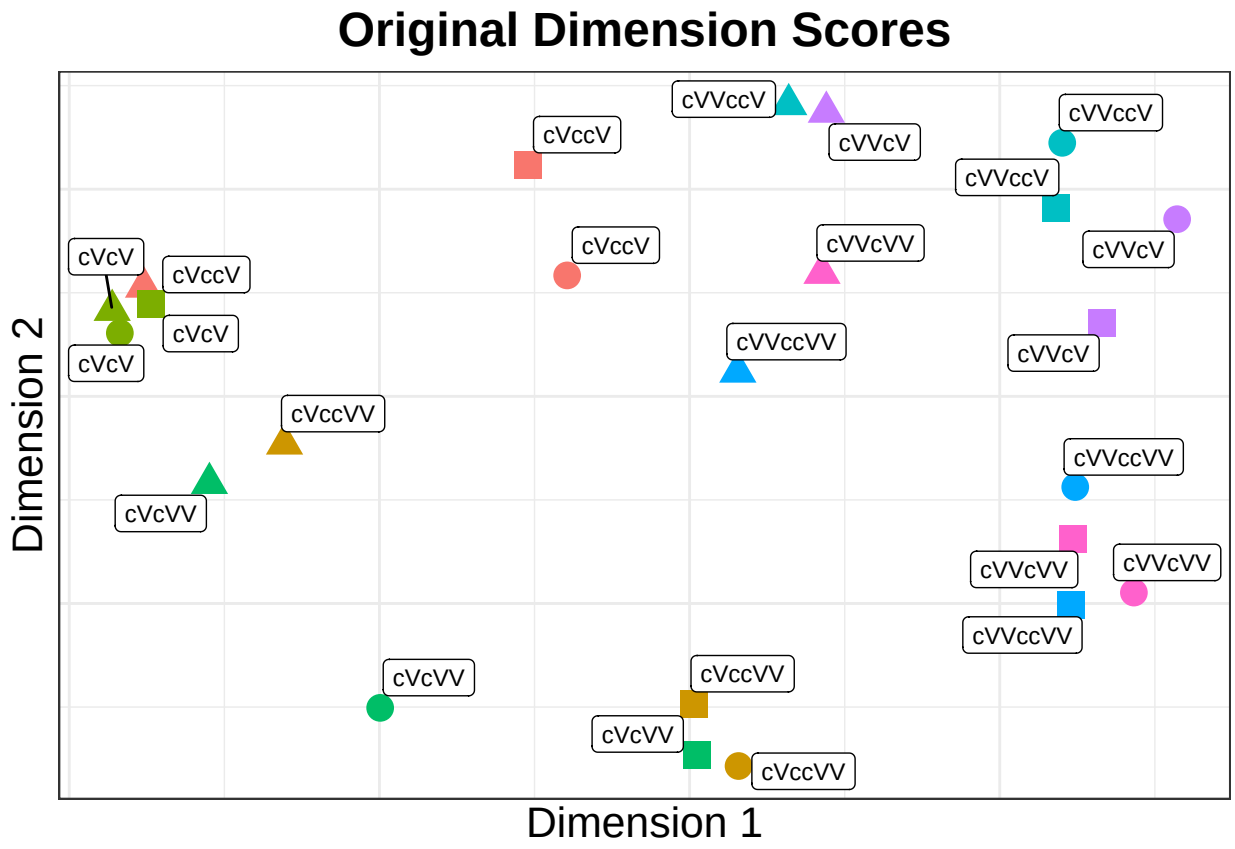
PLOT ROTATED MDS SOLUTION

2D + 3D: CREATE MDS PLOT WITH ROTATED DIMENSION SCORES - Dim 1 by Dim 2

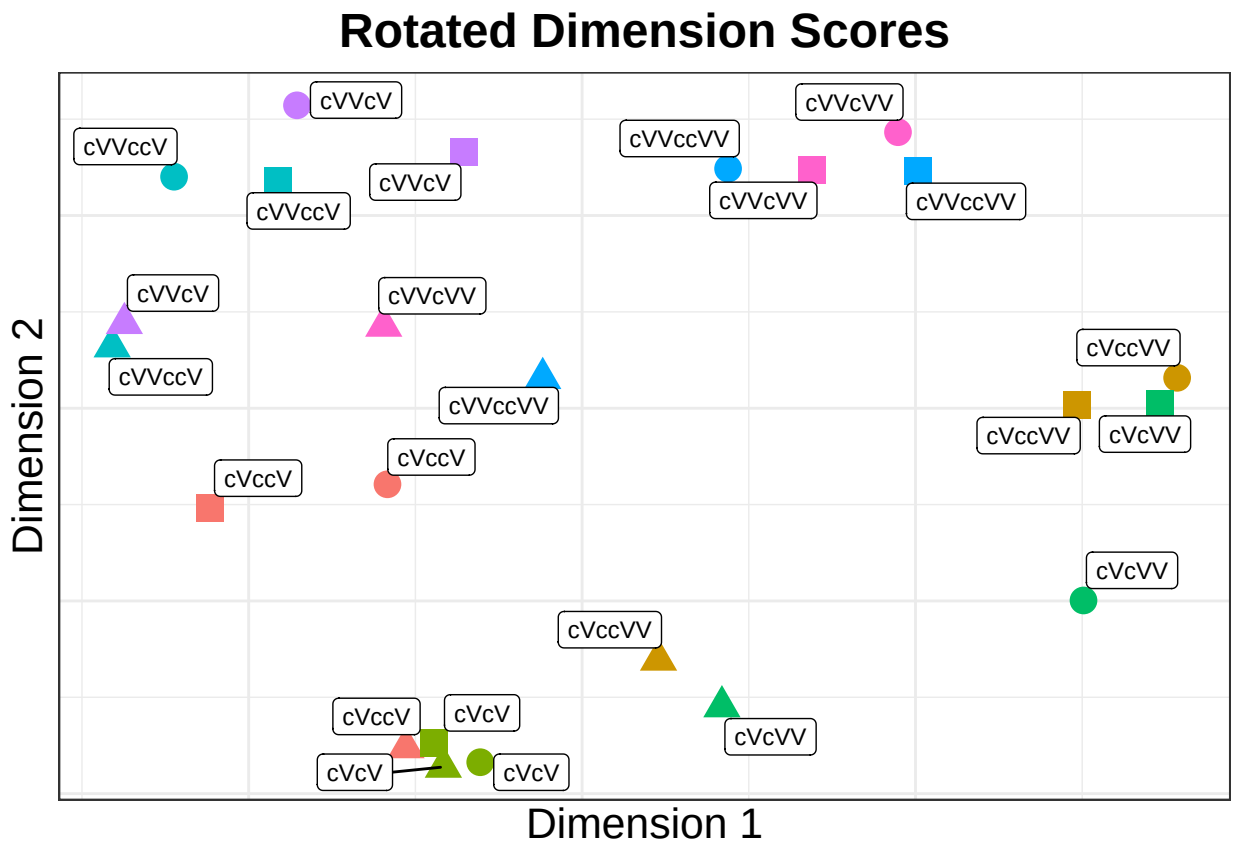
NOTE: Plot only based on z-rotation in this example

```
# sets the format of the plot to black and white
theme_set(theme_bw())

# indicate the data and the columns with the data you want to plot
# ggplot(data, aes(x = dimension 1, y = dimension 2))
mds_plot_rotated_1by2 <- ggplot(MDS_points_rotated, aes(x=D1, y=D2, shape=speaker, color=condition, siz

  # make a scatterplot
  geom_point() +

  # add labels for points; if you want rownames as labels, use label=rownames(MDS_points)
  geom_label_repel(aes(label=condition), color="black", size=3) +

  # makes the labels for the title of the graph and x and y axis
  labs(
    title="Rotated Dimension Scores",
       x="Dimension 1",
       y="Dimension 2")



# change the look of the plot
mds_plot_rotated_1by2 <- mds_plot_rotated_1by2 + theme(

  # for title, change font size, make it bold, centered, and change line height
  plot.title=element_text(size=18,
                          face="bold",
                          hjust=0.5,
                          lineheight=1.2),

  # remove legend
  legend.position = "none",

  # change size of x axis title
  axis.title.x=element_text(size=16),

  # change size of y axis title
  axis.title.y=element_text(size=16),

  # remove x axis text
  axis.text.x=element_blank(),

  # remove y axis text
  axis.text.y=element_blank(),

  # remove x axis ticks
  axis.ticks.x=element_blank(),

  # remove y axis ticks
  axis.ticks.y=element_blank())

  # change size of text for x axis labels
```

```
  # axis.text.x=element_text(size=12, face="bold"),

  # change size of text for y axis labels
  # axis.text.y=element_text(size=12,face="bold"))

# plot rotated figure and original figure for comparison
plot(mds_plot_1by2)
```




```
plot(mds_plot_rotated_1by2)
```

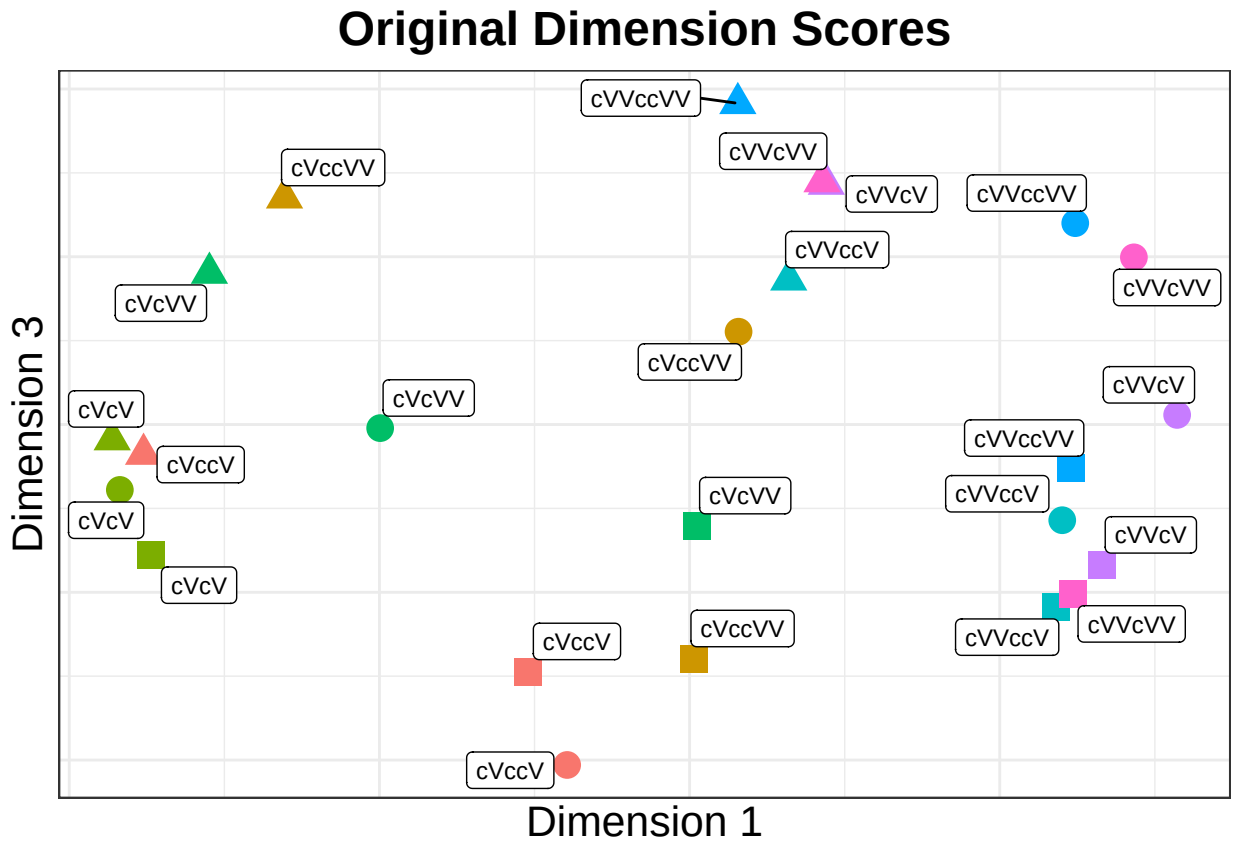


3D only: CREATE MDS PLOT WITH ROTATED DIMENSION SCORES - Dim 1 by Dim 3

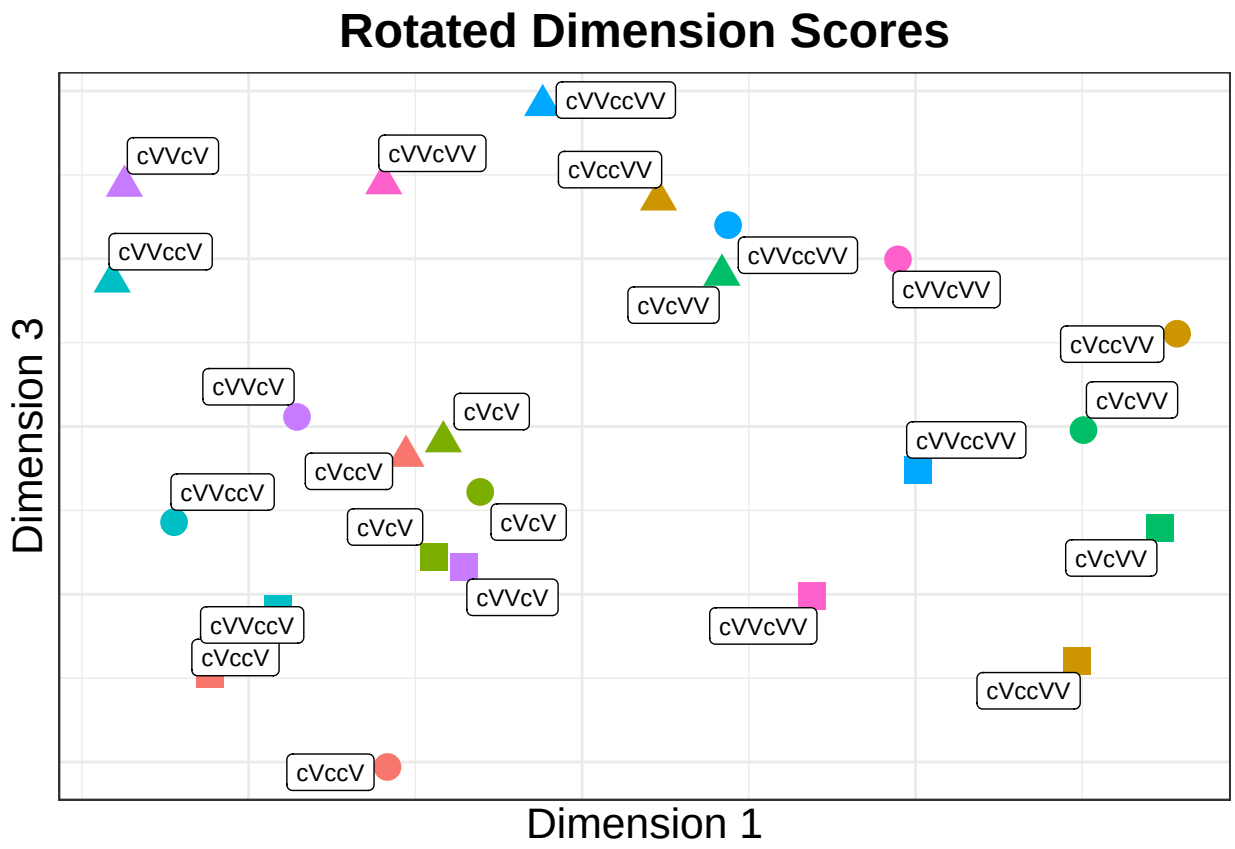
NOTE: Plot only based on z-rotation in this example

```
# sets the format of the plot to black and white
theme_set(theme_bw())

# indicate the data and the columns with the data you want to plot
# ggplot(data, aes(x = dimension 1, y = dimension 2))
mds_plot_rotated_1by3 <- ggplot(MDS_points_rotated, aes(x=D1, y=D3, shape=speaker, color=condition, siz

  # make a scatterplot
  geom_point() +

  # add labels for points; if you want rownames as labels, use label=rownames(MDS_points)
  geom_label_repel(aes(label=condition), color="black", size=3) +

  # makes the labels for the title of the graph and x and y axis
  labs(
    title="Rotated Dimension Scores",
       x="Dimension 1",
       y="Dimension 3")



# change the look of the plot
mds_plot_rotated_1by3 <- mds_plot_rotated_1by3 + theme(
```

```
  # for title, change font size, make it bold, centered, and change line height
  plot.title=element_text(size=18,
                          face="bold",
                          hjust=0.5,
                          lineheight=1.2),

  # remove legend
  legend.position = "none",

  # change size of x axis title
  axis.title.x=element_text(size=16),

  # change size of y axis title
  axis.title.y=element_text(size=16),

  # remove x axis text
  axis.text.x=element_blank(),

  # remove y axis text
  axis.text.y=element_blank(),

  # remove x axis ticks
  axis.ticks.x=element_blank(),

  # remove y axis ticks
  axis.ticks.y=element_blank())

  # change size of text for x axis labels
  # axis.text.x=element_text(size=12, face="bold"),

  # change size of text for y axis labels
  # axis.text.y=element_text(size=12,face="bold"))

# plot rotated figure and original figure for comparison
plot(mds_plot_1by3)
```




```
plot(mds_plot_rotated_1by3)
```



SAVE ROTATED MDS SOLUTION

2D + 3D: SAVE ROTATED DIMENSION SCORES

If you want to compare acoustic or phonological properties to dimension scores, this will save the new rotated ones. If you've rotated along multiple axes, check that you're saving the correct version.

```
write.csv(MDS_points_rotated, "MDS_rotated.csv")
```

SAVE PLOTS

2D + 3D: SAVE PLOT

You can replace mds_plot_1by2 with whichever plot you want to save: mds_plot_rotated_1by2, mds_plot_1by3, mds_plot_rotated_1by3

```
# save MDS plot
ggsave("MDS_1by2.png", mds_plot_1by2, width = 160, height = 120, units=c("mm"), dpi = 600)
```