Reducing Data Complexity

Luis Francisco Gómez López

FAEDIS

2024-09-18

Table of contents I

- This presentation is based on (Chapman and Feit 2019, chap. 8)

- Apply data complexity reduction by using the principal component analysis technique

- On a scale from 1 to 10, where 1 is least and 10 is most, how `<perceptual adjective>` is `<brand>`?
- 100 respondents rate 10 brands on 9 perceptual adjectives
    - **perform**: has strong performance $(1, 2, \dots, 10)$
    - **leader**: is a leader in the field $(1, 2, \dots, 10)$
    - **latest**: has the latest products $(1, 2, \dots, 10)$
    - **fun**: is fun $(1, 2, \dots, 10)$
    - **serious**: is serious $(1, 2, \dots, 10)$
    - **bargain**: products are a bargain $(1, 2, \dots, 10)$
    - **value**: products are a good value $(1, 2, \dots, 10)$
    - **trendy**: is trendy $(1, 2, \dots, 10)$
    - **rebuy**: I would buy from `<brand>` again $(1, 2, \dots, 10)$
    - **brand**: coffee brand rated by a consumer $(a, b, \dots, j)$

- Import data

```r
consumer_brand <- read_csv("http://goo.gl/IQl8nc")
consumer_brand |> head(n = 5)
```

```
# A tibble: 5 x 10
  perform leader latest   fun serious bargain value trendy rebuy brand
    <dbl>  <dbl>  <dbl> <dbl>   <dbl>   <dbl> <dbl>  <dbl> <dbl> <chr>
1       2      4      8     8       2       9     7      4     6 a
2       1      1      4     7       1       1     1      2     2 a
3       2      3      5     9       2       9     5      1     6 a
4       1      6     10     8       3       4     5      2     1 a
5       1      1      5     8       1       9     9      1     1 a
```

- Transform data

```r
consumer_brand_scale <- consumer_brand |>
  mutate(across(perform:rebuy,
                .fns = ~ scale(x = .x,
                               center = TRUE,
                               scale = TRUE)[,1]))
consumer_brand_scale |> head()
```

```
# A tibble: 6 x 10
  perform leader latest    fun serious bargain  value trendy  rebuy brand
    <dbl>  <dbl>  <dbl>  <dbl>   <dbl>   <dbl>  <dbl>  <dbl>  <dbl> <chr>
1  -0.777 -0.160  0.586  0.704  -0.836  1.78    1.11  -0.445  0.893 a
2  -1.09  -1.31  -0.713  0.340  -1.20  -1.22   -1.39  -1.17  -0.679 a
3  -0.777 -0.543 -0.388  1.07   -0.836  1.78    0.276 -1.54   0.893 a
4  -1.09   0.607  1.24   0.704  -0.476 -0.0971  0.276 -1.17  -1.07  a
5  -1.09  -1.31  -0.388  0.704  -1.20   1.78    1.94  -1.54  -1.07  a
6  -0.777  1.37   0.911 -0.389  -0.476  1.40    1.11  -1.54  -0.679 a
```

- Summarize data
    - Ups the table is really big!!! Try it in your console to see the complete table

```
consumer_brand_scale |> skim()
```

- Correction matrices

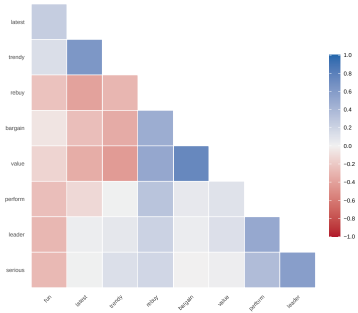
    - Pearson correlation coefficients for samples in a tibble

```
correlation_matrix <- consumer_brand_scale |>
  select(perform:rebuy) |>
  corrr::correlate(use = "pairwise.complete.obs", # There are NA values
                   method = "pearson",
                   diagonal = NA)
correlation_matrix # Ups!!! The tibble is wide. Check out the tibble in your console
```

```
# A tibble: 9 x 10
  term     perform  leader  latest     fun  serious  bargain   value   trendy
  <chr>      <dbl>   <dbl>   <dbl>   <dbl>    <dbl>    <dbl>   <dbl>    <dbl>
1 perform NA         0.500  -0.122  -0.256   0.359    0.0571   0.102   0.00873
2 leader   0.500    NA       0.0269 -0.290   0.571    0.0331   0.118   0.0665
3 latest  -0.122     0.0269 NA        0.245   0.00995 -0.254   -0.343   0.628
4 fun     -0.256    -0.290   0.245   NA      -0.281   -0.0666  -0.145   0.128
5 serious  0.359     0.571   0.00995 -0.281  NA       -0.00266  0.0238  0.121
6 bargain  0.0571    0.0331 -0.254   -0.0666 -0.00266 NA        0.740  -0.351
7 value    0.102     0.118  -0.343   -0.145   0.0238   0.740   NA       -0.435
8 trendy   0.00873   0.0665  0.628    0.128   0.121   -0.351   -0.435   NA
9 rebuy    0.307     0.209  -0.397   -0.237   0.181    0.467    0.506  -0.298
# i 1 more variable: rebuy <dbl>
```

- Correlation matrices
    - Pearson correlation coefficients for samples in a tibble

```
correlation_matrix |>
  autoplot(method = "HC", # Hierarchical clustering: More details in Chapter 11
           triangular = "lower")
```



- Mean ratings by brand

```
brand_mean <- consumer_brand_scale |>
  group_by(brand) |>
  summarise(across(everything(), .fns = mean))
brand_mean
```

```
# A tibble: 10 x 10
   brand perform leader latest    fun serious bargain   value trendy   rebuy
   <chr>   <dbl>  <dbl>  <dbl>  <dbl>   <dbl>   <dbl>   <dbl>  <dbl>   <dbl>
 1 a      -0.886 -0.528  0.411  0.657 -0.919   0.214   0.185  -0.525  -0.596
 2 b       0.931  1.07   0.726 -0.972  1.18    0.0416  0.151   0.740   0.237
 3 c       0.650  1.16  -0.102 -0.845  1.22   -0.607  -0.441   0.0255 -0.132
 4 d      -0.680 -0.593  0.352  0.187 -0.692  -0.881  -0.933   0.737  -0.494
 5 e      -0.564  0.193  0.456  0.296  0.0421  0.552   0.418   0.139   0.0365
 6 f      -0.0587 0.270 -1.26  -0.218  0.589   0.874   1.02   -0.813   1.36
 7 g       0.918 -0.168 -1.28  -0.517 -0.534   0.897   1.26   -1.28    1.36
 8 h      -0.0150 -0.298  0.502  0.715 -0.141  -0.738  -0.783   0.864  -0.604
 9 i       0.335 -0.321  0.356  0.412 -0.149  -0.255  -0.803   0.591  -0.203
10 j      -0.630 -0.789 -0.154  0.285 -0.602  -0.0971 -0.0738 -0.481  -0.962
```

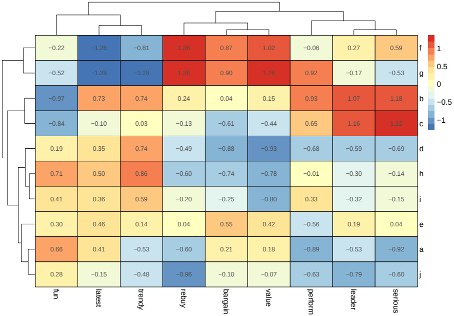
- Mean ratings by brand

```r
brand_mean_longer <- brand_mean |>
  pivot_longer(cols = perform:rebuy,
               names_to = "perceptual_adjectives",
               values_to = "value_mean") |>
  mutate(brand = fct_reorder(.f = brand, .x = value_mean),
         perceptual_adjectives = fct_reorder(.f = perceptual_adjectives, .x = value_mean))
brand_mean_longer
```

```
# A tibble: 90 x 3
   brand perceptual_adjectives value_mean
   <fct> <fct>                      <dbl>
 1 a     perform                   -0.886
 2 a     leader                    -0.528
 3 a     latest                     0.411
 4 a     fun                        0.657
 5 a     serious                   -0.919
 6 a     bargain                    0.214
 7 a     value                      0.185
 8 a     trendy                    -0.525
 9 a     rebuy                     -0.596
10 b     perform                    0.931
# i 80 more rows
```

- Heat map mean ratings by brand

```
library(tidyheatmaps)
tidyheatmap(df = brand_mean_longer,
           rows = brand, columns = perceptual_adjectives, values = value_mean,
           cluster_rows = TRUE, cluster_cols = TRUE,
           clustering_method = "complete", # See ?hclust and chapter 11
           display_numbers = TRUE, border_color = "black", fontsize = 12)
```



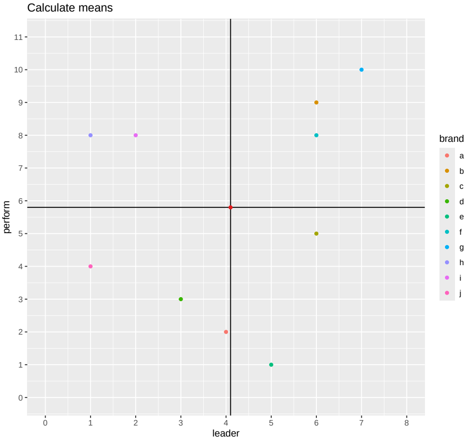
	fun	latest	trendy	rebuy	bargain	value	perform	leader	serious
f	−0.22	−1.26	−0.81	1.36	0.87	1.02	−0.06	0.27	0.59
g	−0.52	−1.28	−1.28	1.36	0.90	1.26	0.92	−0.17	−0.53
b	−0.97	0.73	0.74	0.24	0.04	0.15	0.93	1.07	1.18
c	−0.84	−0.10	0.03	−0.13	−0.61	−0.44	0.65	1.16	1.22
d	0.19	0.35	0.74	−0.49	−0.88	−0.93	−0.68	−0.59	−0.69
h	0.71	0.50	0.86	−0.60	−0.74	−0.78	−0.01	−0.30	−0.14
i	0.41	0.36	0.59	−0.20	−0.25	−0.80	0.33	−0.32	−0.15
e	0.30	0.46	0.14	0.04	0.55	0.42	−0.56	0.19	0.04
a	0.66	0.41	−0.53	−0.60	0.21	0.18	−0.89	−0.53	−0.92
j	0.28	−0.15	−0.48	−0.96	−0.10	−0.07	−0.63	−0.79	−0.60

- Principal component analysis (PCA) and perceptual maps
    - PCA reduced example

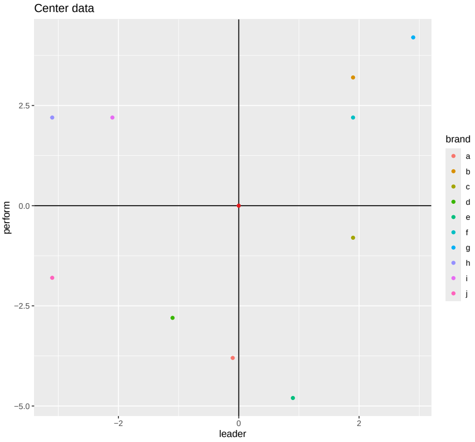
```r
set.seed(seed = 1234)
consumer_brand_sample <- consumer_brand |>
  slice_sample(n = 1, by = brand) |>
  select(brand, perform, leader)
consumer_brand_sample
```

```
# A tibble: 10 x 3
   brand perform leader
   <chr>   <dbl>  <dbl>
 1 a           2      4
 2 b           9      6
 3 c           5      6
 4 d           3      3
 5 e           1      5
 6 f           8      6
 7 g          10      7
 8 h           8      1
 9 i           8      2
10 j           4      1
```

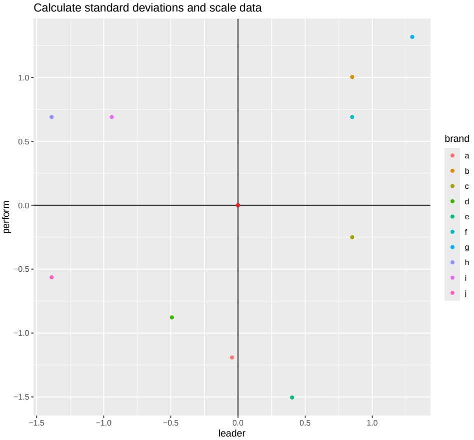
- Principal component analysis (PCA) and perceptual maps
    - Visualizing original data



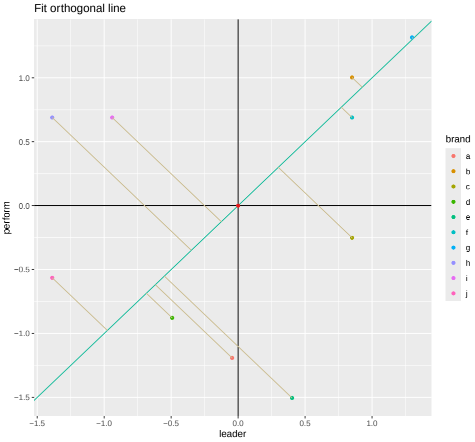
- Principal component analysis (PCA) and perceptual maps
  - Centering data using the mean



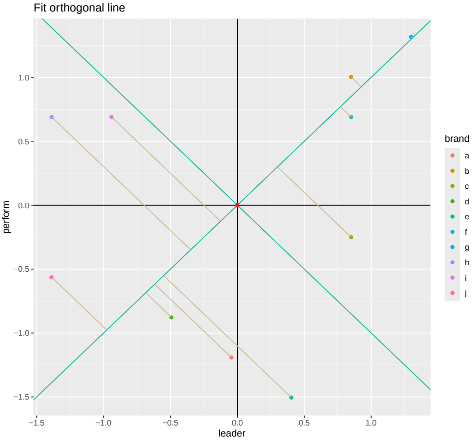
- Principal component analysis (PCA) and perceptual maps
  - Scaling data using the standard deviation



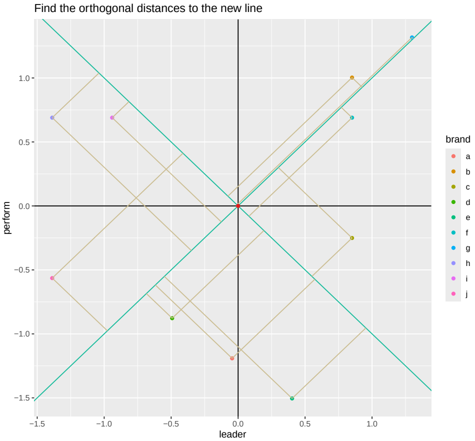
- Principal component analysis (PCA) and perceptual maps
  - Fitting a line by performing an orthogonal regression



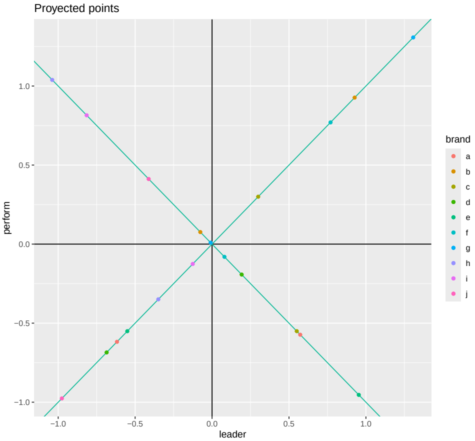
- Principal component analysis (PCA) and perceptual maps
  - Find a line orthogonal to the fitted line



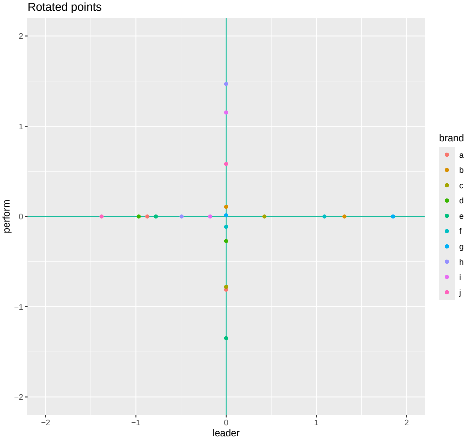
- Principal component analysis (PCA) and perceptual maps
  - Find the orthogonal distances between the points and the new line



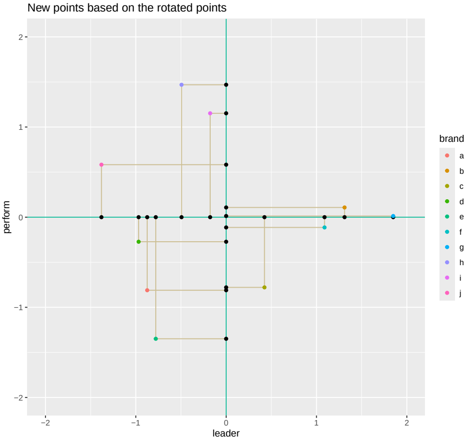
- Principal component analysis (PCA) and perceptual maps
  - Project the points onto each line



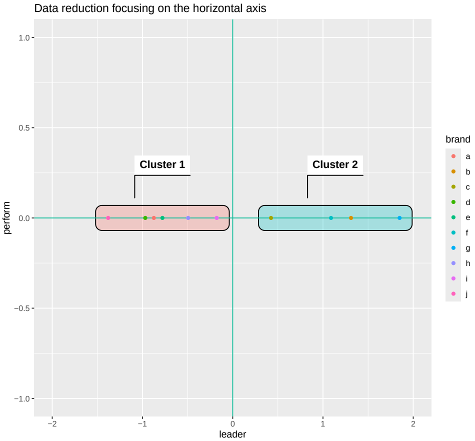
- Principal component analysis (PCA) and perceptual maps
  - Rotate the fitted line and the projected points around $(0,0)$



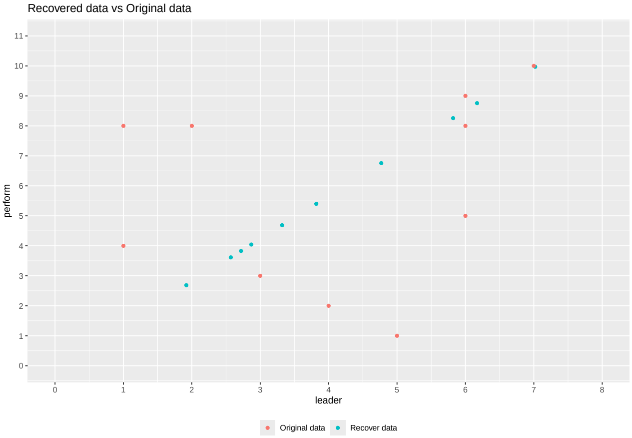
- Principal component analysis (PCA) and perceptual maps
  - Fix the new points based on the projected points



- Principal component analysis (PCA) and perceptual maps
  - Apply data complexity reduction by focusing on the horizontal axis



- Principal component analysis (PCA) and perceptual maps
  - Recover the data that was reduced when focusing in the horizontal axis



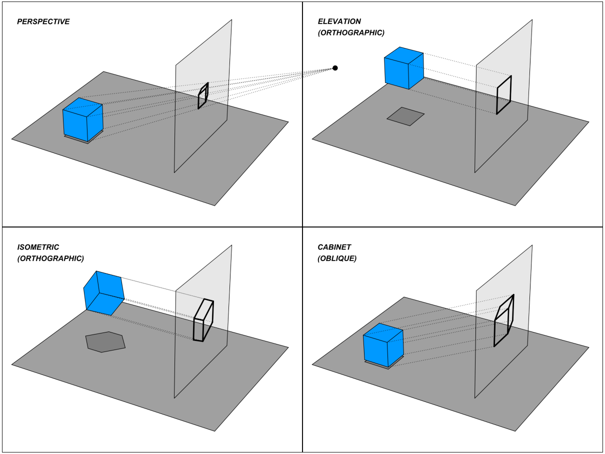


Figure 1: Various projections of cube above plane by Michael Horvath (aka Datumizer) 2021-05-10

- Principal component analysis (PCA) and perceptual maps
  - Using an image to understand data complexity reduction

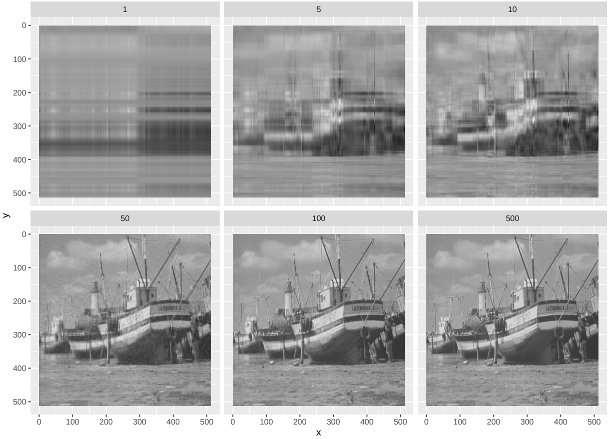
- Principal component analysis (PCA) and perceptual maps
  - Represent and image as data
    - `x,y`: position of a point in a cartesian plane (x, y)
    - `value`: a gray scale where 0 is white, 1 is black and $(0, 1)$ is an intermediate color between white and black

```
# A tibble: 262,144 x 3
       x     y value
   <int> <int> <dbl>
 1     1     1 0.498
 2     2     1 0.482
 3     3     1 0.490
 4     4     1 0.471
 5     5     1 0.494
 6     6     1 0.482
 7     7     1 0.498
 8     8     1 0.502
 9     9     1 0.490
10    10     1 0.506
# i 262,134 more rows
```

- Principal component analysis (PCA) and perceptual maps
  - Prepare data for PCA

```
# A tibble: 512 x 513
       x   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`  `10`  `11`  `12`
   <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1     1 0.498 0.502 0.502 0.486 0.494 0.490 0.498 0.482 0.494 0.486 0.478 0.494
 2     2 0.482 0.494 0.486 0.498 0.490 0.498 0.498 0.529 0.502 0.502 0.494 0.498
 3     3 0.490 0.502 0.502 0.502 0.502 0.494 0.494 0.471 0.486 0.498 0.502 0.490
 4     4 0.471 0.478 0.494 0.506 0.494 0.494 0.486 0.502 0.502 0.486 0.494 0.478
 5     5 0.494 0.490 0.498 0.475 0.494 0.502 0.471 0.475 0.490 0.498 0.482 0.490
 6     6 0.482 0.490 0.471 0.502 0.490 0.502 0.498 0.482 0.482 0.475 0.498 0.475
 7     7 0.498 0.478 0.502 0.506 0.498 0.502 0.502 0.494 0.502 0.502 0.486 0.498
 8     8 0.502 0.506 0.506 0.502 0.502 0.494 0.494 0.494 0.510 0.510 0.506 0.502
 9     9 0.490 0.498 0.502 0.506 0.514 0.510 0.502 0.502 0.502 0.518 0.514 0.514
10    10 0.506 0.502 0.514 0.522 0.498 0.506 0.514 0.522 0.518 0.522 0.525 0.514
# i 502 more rows
# i 500 more variables: `13` <dbl>, `14` <dbl>, `15` <dbl>, `16` <dbl>,
#   `17` <dbl>, `18` <dbl>, `19` <dbl>, `20` <dbl>, `21` <dbl>, `22` <dbl>,
#   `23` <dbl>, `24` <dbl>, `25` <dbl>, `26` <dbl>, `27` <dbl>, `28` <dbl>,
#   `29` <dbl>, `30` <dbl>, `31` <dbl>, `32` <dbl>, `33` <dbl>, `34` <dbl>,
#   `35` <dbl>, `36` <dbl>, `37` <dbl>, `38` <dbl>, `39` <dbl>, `40` <dbl>,
#   `41` <dbl>, `42` <dbl>, `43` <dbl>, `44` <dbl>, `45` <dbl>, `46` <dbl>, ...
```

- Principal component analysis (PCA) and perceptual maps
  - Data complexity reduction



- Principal component analysis (PCA) and perceptual maps
    - Applying to the reduced example

```
consumer_brand_sample_matrix <- consumer_brand_sample |>
  select(-brand) |>
  as.matrix()
consumer_brand_sample_matrix |> head()
```

```
     perform leader
[1,]       2      4
[2,]       9      6
[3,]       5      6
[4,]       3      3
[5,]       1      5
[6,]       8      6
```

- Principal component analysis (PCA) and perceptual maps
  - prcomp output from R

```r
consumer_brand_sample_matrix_pca <- consumer_brand_sample_matrix |>
  prcomp(center = TRUE, scale. = TRUE)
consumer_brand_sample_matrix_pca
```

```
Standard deviations (1, .., p=2):
[1] 1.1051789 0.8823716

Rotation (n x k) = (2 x 2):
              PC1        PC2
perform 0.7071068  0.7071068
leader  0.7071068 -0.7071068
```

- Principal component analysis (PCA) and perceptual maps
    - Structure of prcompfrom R

```r
consumer_brand_sample_matrix_pca |> str()
```

```
List of 5
 $ sdev    : num [1:2] 1.105 0.882
 $ rotation: num [1:2, 1:2] 0.707 0.707 0.707 -0.707
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2] "perform" "leader"
  .. ..$ : chr [1:2] "PC1" "PC2"
 $ center  : Named num [1:2] 5.8 4.1
  ..- attr(*, "names")= chr [1:2] "perform" "leader"
 $ scale   : Named num [1:2] 3.19 2.23
  ..- attr(*, "names")= chr [1:2] "perform" "leader"
 $ x       : num [1:10, 1:2] -0.874 1.311 0.424 -0.969 -0.779 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "PC1" "PC2"
 - attr(*, "class")= chr "prcomp"
```

- Principal component analysis (PCA) and perceptual maps
    - Extracting scores: principle components space

```
scores <- consumer_brand_sample_matrix_pca$x
scores
```

```
            PC1         PC2
 [1,] -0.8739101 -0.81059416
 [2,]  1.3107664  0.10776349
 [3,]  0.4241852 -0.77881770
 [4,] -0.9688445 -0.27236914
 [5,] -0.7789757 -1.34881917
 [6,]  1.0891211 -0.11388181
 [7,]  1.8489914  0.01282907
 [8,] -0.4937775  1.46901678
 [9,] -0.1771978  1.15243706
[10,] -1.3803587  0.58243559
```

- Principal component analysis (PCA) and perceptual maps
  - Extracting loadings: map from principle components space back into the original space

```
loadings <- consumer_brand_sample_matrix_pca$rotation
loadings
```

```
              PC1        PC2
perform 0.7071068  0.7071068
leader  0.7071068 -0.7071068
```

- Principal component analysis (PCA) and perceptual maps
  - Extracting loadings: map from principle components space back into the original space

```
consumer_brand_sample_matrix_center_scale <- consumer_brand_sample_matrix |>
  scale(center = TRUE, scale = TRUE)
consumer_brand_sample_matrix_center_scale
```

```
         perform      leader
 [1,] -1.1911244 -0.04477113
 [2,]  1.0030521  0.85065153
 [3,] -0.2507630  0.85065153
 [4,] -0.8776706 -0.49248246
 [5,] -1.5045782  0.40294020
 [6,]  0.6895983  0.85065153
 [7,]  1.3165059  1.29836285
 [8,]  0.6895983 -1.38790512
 [9,]  0.6895983 -0.94019379
[10,] -0.5642168 -1.38790512
attr(,"scaled:center")
perform  leader
    5.8     4.1
attr(,"scaled:scale")
 perform   leader
3.190263 2.233582
```

- Principal component analysis (PCA) and perceptual maps
  - Using matrix multiplication, %*%, the original centered and scaled data, $X_{c,s}$, and the loadings, L, `loadings` to obtain the scores, S

$$S = X_{c,s}L$$

```
consumer_brand_sample_matrix_center_scale %*% loadings
```

```
            PC1         PC2
 [1,] -0.8739101 -0.81059416
 [2,]  1.3107664  0.10776349
 [3,]  0.4241852 -0.77881770
 [4,] -0.9688445 -0.27236914
 [5,] -0.7789757 -1.34881917
 [6,]  1.0891211 -0.11388181
 [7,]  1.8489914  0.01282907
 [8,] -0.4937775  1.46901678
 [9,] -0.1771978  1.15243706
[10,] -1.3803587  0.58243559
```

- Principal component analysis (PCA) and perceptual maps
  - Recovering original centered and scaled data, X, using loadings, L,[1] and scores, S

$$SL^t = X_{c,s}LL^t = X_{c,s}I = X_{c,s}$$

```r
(scores %*% t(loadings)) |> set_colnames(c("perform", "leader"))
```

```
        perform      leader
 [1,] -1.1911244 -0.04477113
 [2,]  1.0030521  0.85065153
 [3,] -0.2507630  0.85065153
 [4,] -0.8776706 -0.49248246
 [5,] -1.5045782  0.40294020
 [6,]  0.6895983  0.85065153
 [7,]  1.3165059  1.29836285
 [8,]  0.6895983 -1.38790512
 [9,]  0.6895983 -0.94019379
[10,] -0.5642168 -1.38790512
```

[1] L is an orthogonal matrix, which means that L is a real square matrix such that $L^tL = LL^t = I$ where I is the identity matrix.

- Principal component analysis (PCA) and perceptual maps
  - Reconstructing original centered and scaled data using the first principal component, X_{c,s,p_1}

$$S_{p_1} L_{p_1}^t = X_{c,s,p_1}$$

```
scores[, 1] %*% t(loadings[, 1])
```

```
        perform     leader
 [1,] -0.6179478 -0.6179478
 [2,]  0.9268518  0.9268518
 [3,]  0.2999442  0.2999442
 [4,] -0.6850765 -0.6850765
 [5,] -0.5508190 -0.5508190
 [6,]  0.7701249  0.7701249
 [7,]  1.3074344  1.3074344
 [8,] -0.3491534 -0.3491534
 [9,] -0.1252977 -0.1252977
[10,] -0.9760610 -0.9760610
```

- Principal component analysis (PCA) and perceptual maps
  - Reconstructing original centered data using the first principal component, X_{c,p_1}

```
scores[, 1] %*% t(loadings[, 1]) |>
  scale(center = FALSE, scale = 1/consumer_brand_sample_matrix_pca$scale)
```

```
         perform     leader
 [1,] -1.9714158 -1.3802370
 [2,]  2.9569010  2.0701996
 [3,]  0.9569010  0.6699501
 [4,] -2.1855743 -1.5301747
 [5,] -1.7572574 -1.2302994
 [6,]  2.4569010  1.7201372
 [7,]  4.1710595  2.9202620
 [8,] -1.1138912 -0.7798628
 [9,] -0.3997327 -0.2798628
[10,] -3.1138912 -2.1801123
attr(,"scaled:scale")
  perform    leader
0.3134538 0.4477113
```

- Principal component analysis (PCA) and perceptual maps
    - Reconstructing original data using the first principal component, X_{p_1}

```
scores[, 1] %*% t(loadings[, 1]) |>
  scale(center = FALSE, scale = 1/consumer_brand_sample_matrix_pca$scale) |>
  scale(center = -consumer_brand_sample_matrix_pca$center, scale = FALSE)
```

```
       perform   leader
 [1,] 3.828584 2.719763
 [2,] 8.756901 6.170200
 [3,] 6.756901 4.769950
 [4,] 3.614426 2.569825
 [5,] 4.042743 2.869701
 [6,] 8.256901 5.820137
 [7,] 9.971059 7.020262
 [8,] 4.686109 3.320137
 [9,] 5.400267 3.820137
[10,] 2.686109 1.919888
attr(,"scaled:scale")
  perform    leader
0.3134538 0.4477113
attr(,"scaled:center")
perform  leader
   -5.8    -4.1
```

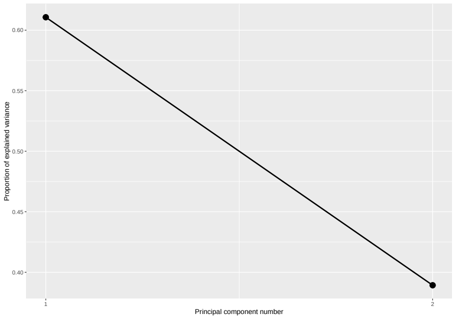
- Principal component analysis (PCA) and perceptual maps
  - Eingevalues, in this case `variance`, represent the variance explained by each principal component

```
eigenvalues <- consumer_brand_sample_matrix_pca |>
  tidy(matrix = "eigenvalues") |>
  mutate(variance = std.dev^2, .after = std.dev)
eigenvalues
```

```
# A tibble: 2 x 5
     PC std.dev variance percent cumulative
  <dbl>   <dbl>    <dbl>   <dbl>      <dbl>
1     1   1.11     1.22    0.611      0.611
2     2   0.882    0.779   0.389      1
```

- Principal component analysis (PCA) and perceptual maps
    - Eingevalues, in this case `variance`, represent the variance explained by each principal component

```
library(ggbiplot)
consumer_brand_sample_matrix_pca |>
  ggscreeplot() +
  scale_x_continuous(breaks = 1:2)
```

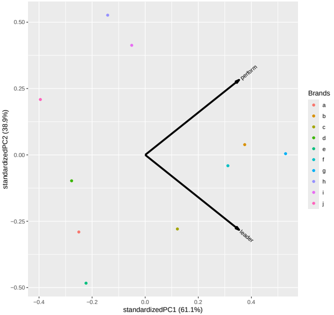


- Principal component analysis (PCA) and perceptual maps
  - A **biplot** represents visually the scores of the first, x-axis, and second, y-axis, of the principal components and the corresponding loadings both **scaled by a factor**[2]
  - In the case of principal component analysis there are many different ways to produce a biplot
    - For the differents ways to build a biplot check out Positioning the arrows on a PCA biplot

[2] For specific details check out `?stats:::biplot.prcomp`, `?ggbiplot::ggbiplot` and `?ggbiplot::get_SVD`

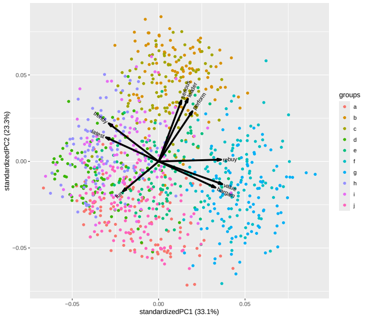
- Principal component analysis (PCA) and perceptual maps
  - Building a biplot using the package ggbiplot

```
ggbiplot(pcobj = consumer_brand_sample_matrix_pca,
         groups = consumer_brand_sample$brand,
         scale = 1, pc.biplot = FALSE) +
  labs(color = "Brands")
```



- Bibplot for all the consumer brand perception survey

```r
consumer_brand_pca <- consumer_brand |>
  select(-brand) |>
  prcomp(center = TRUE, scale. = TRUE)
consumer_brand_pca |>
  ggbiplot(groups = consumer_brand$brand, scale = 1, pc.biplot = FALSE)
```



- A biplot is a generalization of a scatterplot of 2 variables for the case of many variables (Greenacre 2010, 9)
- Variables of the brands that are grouped together are positively correlated to each other
  - For example `serious`, `leader` and `perform` or `trendy` and `latest`
- Variables of the brands that are displayed to the opposite sides of the biplot origin are negatively correlated to each other
  - For example `fun` in relation to `serious`, `leader` and `perform` or `trendy` and `latest` in relation to `value` and `bargain`

- A biplot is an **approximated** representation of a data table ordered by rows which represents some observations and columns which represents some variables
    - By the term **approximated** it means that the representation is not exact
    - In our case the last biplot was used to represent the data table `consumer_brand_sample` by reducing its complexity
- In a biplot the distance between points represent some measure of similarity
    - In the case of the last biplot for example brand g, that is colored in blue, tend to be spatially grouped in the plot

- To my family that supports me
- To the taxpayers of Colombia and the **UMNG students** who pay my salary
- To the **Business Science** and **R4DS Online Learning** communities where I learn **R** and π**-thon**
- To the **R Core Team**, the creators of **RStudio IDE**, **Quarto** and the authors and maintainers of the packages **tidyverse**, **skimr**, **corrr**, **tidyheatmaps tidymodels**, **ggforce**, **ggbiplot** , **imager** and **tinytex** for allowing me to access these tools without paying for a license
- To the **Linux kernel community** for allowing me the possibility to use some **Linux distributions** as my main **OS** without paying for a license

References I

Chapman, Chris, and Elea McDonnell Feit. 2019. *R For Marketing Research and Analytics*. 2nd ed. 2019. Use R! Cham: Springer International Publishing : Imprint: Springer. https://doi-org.ezproxy.umng.edu.co/10.1007/978-3-030-14316-9.

Greenacre, Michael J. 2010. *Biplots in Practice*. Bilbao: Fundación BBVA. https://www.fbbva.es/microsite/multivariate-statistics/biplots.html.