

Sensory Marketing & Product Innovation: R Lecture Examples for Sensory Product Research

Marcel Lichters

2026-09-14

Abstract

Goal: This R analysis script presents solutions to the data analysis examples discussed in the lecture Sensory Marketing & Product Innovation at Otto von Guericke University Magdeburg (Germany). In particular, these examples help students practice analyzing data in R. These skills will serve them well throughout their studies and beyond. The internet provides countless additional examples. Excellent textbooks are also available (just search for sensory consumer science). Below, I intentionally moved beyond base R to provide students with state-of-the-art code using modern R packages. I will update these examples from time to time.

Abstract

Goal: This R analysis script presents solutions to the data analysis examples discussed in the lecture Sensory Marketing & Product Innovation at Otto von Guericke University Magdeburg (Germany). In particular, these examples help students practice analyzing data in R. These skills will serve them well throughout their studies and beyond. The internet provides countless additional examples. Excellent textbooks are also available (just search for sensory consumer science). Below, I intentionally moved beyond base R to provide students with state-of-the-art code using modern R packages. I will update these examples from time to time.

Further Information and Learning Objectives

Only modify (also not in part) with appropriate permission! See the license information at the end of the document.

Download the pdf version [here](#)



Figure 1: Follow me on ResearchGate

[Contact me via LinkedIn](#)

💡 If this grabs your attention

If this exercise grabs your attention, please check out our master study programs at the Otto-von-Guericke-University in Magdeburg (Germany) by clicking on the logo!



Figure 2: Faculty of Economics and Management at Otto-von-Guericke-University Magdeburg

In this exercise students should practice analyzing data with [R](#) and [RStudio](#). These skills will serve them well in the progress of their study time and in the future. The internet provides countless further examples. Besides, there exist very good text books ([Chapman & Feit, 2019](#); [Field, 2026](#); [Lê & Worch, 2018](#); [Næs, Brockhoff, & Tomic, 2010](#); [Worch, Delarue, De Souza, & Ennis, 2023](#)). Also, when studying Sensory Marketing & Product Innovation, students are assumed to have already passed Lecture Marketing Methods & Analysis. I also provide R examples for this lecture online ([Lichters, 2024](#)).

Learning Objectives for the students

This exercise is designed to help students learn how to:

- Use prominent *R* packages for sensory product research
 - Making statistical inference from the results of sensory Triangle discrimination tests
 - Use the *R* package `sensR` (Christensen & Brockhoff, 2023) to analyze discrimination test data
 - Use the *R* package `SensoMinreR` to conduct some basic analyses in the domain of sensory descriptive analysis (Husson, Le, & Cadoret, 2023; Lê & Husson, 2008).
-

Loading packages that we will use

i Note

R is a context-sensitive language. Thus, ‘data’ will be interpreted not in the same way as ‘Data’ will.

In *R* most functionality is provided by additional packages.

Most of the packages are well-documented, See: <https://cran.r-project.org/>

1. The code chunk below first evaluates whether the package `pacman` (Rinker & Kurkiewicz, 2018) is already installed on your machine. If yes, the corresponding package will be loaded. If not, *R* will install the package (as long as the package is still maintained on the Comprehensive **R** Archive Network *CRAN*).
2. Alternatively, you can do this manually first by executing `install.packages("pacman")` and then `library(pacman)`.
3. The second line then loads the package *pacman*
4. The third line uses the function `p_load()` from the *pacman* package to install (if necessary) and load all packages that we provide as arguments (e.g., `tidyverse` (Wickham et al., 2019), `sensR` (Christensen & Brockhoff, 2023), `SensoMineR` (Husson et al., 2023)). This only works if all packages are still available on *CRAN*

```

if (!"pacman" %in% rownames(installed.packages()))
install.packages("pacman")

library(pacman)

pacman::p_load(labelled, SensoMineR, sensR, tidyverse)

```

Here is the *R* session info which gives you information on my machine, all loaded packages and their version:

```
sessionInfo()
```

```

R version 4.6.1 (2026-06-24 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 11 x64 (build 26100)

```

```

Matrix products: default
  LAPACK version 3.12.1

```

```
locale:
```

```

[1] LC_COLLATE=German_Germany.utf8  LC_CTYPE=German_Germany.utf8
[3] LC_MONETARY=German_Germany.utf8 LC_NUMERIC=C
[5] LC_TIME=German_Germany.utf8

```

```

time zone: Europe/Berlin
tzcode source: internal

```

```
attached base packages:
```

```
[1] stats      graphics  grDevices  utils      datasets  methods    base
```

```
other attached packages:
```

```

[1] webshot2_0.1.2  tinytex_0.60    lubridate_1.9.5 forcats_1.0.1
[5] stringr_1.6.0   dplyr_1.2.1     purrr_1.2.2     readr_2.2.0
[9] tidyr_1.3.2     tibble_3.3.1    ggplot2_4.0.3   tidyverse_2.0.0
[13] styler_1.11.0   sensR_1.5-3     SensoMineR_1.28 FactoMineR_2.17
[17] quarto_1.5.1    labelled_2.16.0 formatR_1.14     pacman_0.5.1

```

```
loaded via a namespace (and not attached):
```

```

[1] tidyselect_1.2.1  farver_2.1.2      R.utils_2.13.0
[4] S7_0.2.2          fastmap_1.2.0     TH.data_1.1-5
[7] promises_1.5.0    digest_0.6.39     timechange_0.4.0
[10] estimability_2.0.0 lifecycle_1.0.5    cluster_2.1.8.3

```

```

[13] multcompView_0.1-12  survival_3.8-11      processx_3.9.0
[16] magrittr_2.0.5       compiler_4.6.1      rlang_1.3.0
[19] tools_4.6.1          yaml_2.3.12         knitr_1.51
[22] htmlwidgets_1.6.4    scatterplot3d_0.3-45 plyr_1.8.9
[25] showtextdb_3.0       RColorBrewer_1.1-3  websocket_1.4.4
[28] multcomp_1.4-32      KernSmooth_2.23-27  R.cache_0.17.0
[31] withr_3.0.3          numDeriv_2016.8-1.1 R.oo_1.27.1
[34] grid_4.6.1           AlgDesign_1.2.1.2   sysfonts_0.8.9
[37] xtable_1.8-8         emmeans_2.0.4       scales_1.4.0
[40] gtools_3.9.5         MASS_7.3-66         flashClust_1.1-4
[43] cli_3.6.6            mvtnorm_1.4-2       rmarkdown_2.32
[46] generics_0.1.4       otl_0.2.0           rstudioapi_0.19.0
[49] tzdb_0.5.0           reshape2_1.4.5      chromote_0.5.1
[52] splines_4.6.1        vctrs_0.7.3         Matrix_1.7-6
[55] sandwich_3.1-3       jsonlite_2.0.0      hms_1.1.4
[58] ggrepel_0.9.8        irlba_2.3.7         glue_1.8.1
[61] codetools_0.2-20     DT_0.34.0           stringi_1.8.9
[64] gtable_0.3.6         later_1.4.8         pillar_1.11.1
[67] htmltools_0.5.9      showtext_0.9-8      R6_2.6.1
[70] evaluate_1.0.5       lattice_0.23-1      haven_2.5.5
[73] R.methodsS3_1.8.2    leaps_3.2           Rcpp_1.1.2
[76] coda_0.19-4.1        xfun_0.60           zoo_1.9-0
[79] pkgconfig_2.0.3

```

Example for Discrimination test - Triangle test.

Triangle tests are very popular in sensory product research as tests for discrimination, and help to evaluate whether consumers can significantly distinguish between product alternatives (see, e.g., [Lawless & Heymann, 2010, Chapter 4.4](#)). In our lecture example on Lasagna variants, we observe the following results based on n=100 consumers:

	Correctly identified odd sample	Incorrect response
Observed	43 (O_1)	57 (O_2)
Expected	33.3 (E_1)	66.6 (E_2)

In our example, analysis draws on a (Pearson) χ^2 -test (without continuity correction). Here, the test statistic is calculated as follows:

$$\chi^2_{(1)} = \sum \frac{(\text{observed} - \text{expected})^2}{\text{expected}} = \left[\frac{(O_1 - E_1)^2}{E_1} \right] + \left[\frac{(O_2 - E_2)^2}{E_2} \right] = \left[\frac{(43 - 33.3)^2}{33.3} \right] + \left[\frac{(57 - 66.6)^2}{66.6} \right] = 2.803 + 1.402 = 4.205$$

We know from lectures in statistics that the df in a χ^2 -test equals: (number of rows -1) * (number of columns -1). Thus, $df=(2-1)*(2-1)=1$. The degrees of freedom are 1.

The package [sensR](#) provides functions to handle nearly all types of discrimination tests, including Triangle tests ([Christensen & Brockhoff, 2023](#)). We will use the function **discrim()**. This function allows us to specify many arguments (see help by pressing 'F1'). We will do so for

- correct = the number of correct answers
- total = total number of responses
- conf.level = significance level (Type I error probability α)
- method = the discrimination protocol applied (see lecture slides for the various options)
- statistic = The analysis strategy to apply for the data (whether one wants to use normal distribution or χ^2 etc.)
- test = Whether a test for similarity or difference should be obtained.

Below, we assign the results of our Triangle Discrimination test to a new object named 'triangle_test'. Then, we call for its content.

```
triangle_test <- sensR::discrim(
  correct = 43,
  total = 100,
  conf.level = 0.95,
  method = "triangle",
  statistic = "score",
  test = "difference")

triangle_test
```

Estimates for the triangle discrimination protocol with 43 correct answers in 100 trials. One-sided p-value and 95 % two-sided confidence intervals are based on the Pearson and score statistics.

	Estimate	Std. Error	Lower	Upper
pc	0.430	0.04951	0.3333	0.5328
pd	0.145	0.07426	0.0000	0.2992
d-prime	1.075	0.30276	0.0000	1.6355

Result of difference test:

Pearson X-square statistic = 4.205, df = 1, p-value: 0.02015

Alternative hypothesis: d-prime is greater than 0

💡 Interpretation

As the results of the χ^2 -test is significant ($p = 0.02$), the obtained data is not supporting H_0 . Thus, consumers can reliably distinguish between the new and the old lasagna.

- In the output above, pc means the percentage of correct discriminators ($43/100$).
- pd stands for the estimated proportion of individuals in the (relevant) population that would detect the product difference, which is in the case of a Triangle test: $pd = \frac{pc-1/3}{2/3}$. Note that the confidence interval for pd has an upper limit of 29.92%. Thus, in the worst case, the true proportion of discriminators is almost as high as 30%. Schlich (1993) proposed the following limits for pd as *small*, *medium*, and *large* differences: 0.25, 0.375, and 0.50.
- The Thurstonian approach of transforming the number of correct answers into an estimate, called *d-prime* of the underlying (relative) sensory difference is an attempt to overcome the concepts of pc and pd , since these are dependent on the concrete test protocol one has applied (e.g., Triangle vs. 3-AFC tests). The higher the value, the higher the detectable degree of difference between the tested products.

Example of a Post-hoc Power Analysis for a Triangle Test

To get an idea about the statistical Power of a Similarity or Dissimilarity test protocol, the [sensR](#) package also provides functions that facilitate sample size planning for discrimination tests (i.e., statistical power analysis). To give an example, we will use the *d-prime* estimate from the Triangle test on lasagna above (1.075) as input to determine the statistical power of the Triangle test with $n=100$.

In the code chunk below, we use the obtained *d-prime* as input for the function *d.primePwr()*. Furthermore, we specify all other aspects of our exemplary Triangle test.

```
power_triangle <- sensR::d.primePwr(  
  d.primeA = 1.075,  
  sample.size = 100,  
  alpha = 0.05,
```

```

    method = "triangle",
    test = "difference", statistic = "exact"
  )

power_triangle

```

```
[1] 0.617578
```

Interpretation

With the observed effect size for the difference, our Triangle test protocol only has a power of 61.76%. Thus, if the observed effect is the true effect in the underlying population, then we would have this chance to find a significant difference in a Triangle test with $n=100$ and $\alpha=5\%$, which is insufficient.

Finally, we can estimate how many consumers should be tested to obtain sufficient statistical power based on the observed effect size for the difference, a Triangle test, and $\alpha=5\%$. For this purpose, we first establish a list of varying sample sizes below. Afterward, we use this list as input for the *sample.size* argument of the *d.primePwr()* function. Usually, this function does not accept a list of values as arguments. To handle this problem, we additionally apply the *map()* function provided by the *purrr* package (Wickham & Henry, 2023). In the corresponding call of *d.primePwr()* we set **sample.size** = ., to tell the map function to use each element of the input list as input for the sample size argument in repeated calls of *d.primePwr()*.

```

sample_sizes <- list(n50 = 50,
                    n100 = 100,
                    n150 = 150,
                    n200 = 200,
                    n250 = 250,
                    n300 = 300,
                    n350 = 350,
                    n400 = 400,
                    n450 = 450,
                    n500 = 500)

sample_sizes |>
  map(~ sensR::d.primePwr(
    d.primeA = 1.075,
    sample.size = .,
    alpha = 0.05,

```

```
method = "triangle",  
test = "difference", statistic = "exact"  
)
```

```
$n50  
[1] 0.3855667
```

```
$n100  
[1] 0.617578
```

```
$n150  
[1] 0.7447738
```

```
$n200  
[1] 0.8583079
```

```
$n250  
[1] 0.9206918
```

```
$n300  
[1] 0.9552637
```

```
$n350  
[1] 0.9804197
```

```
$n400  
[1] 0.9888657
```

```
$n450  
[1] 0.9936335
```

```
$n500  
[1] 0.9972308
```

Interpretation

We can see that with a sample size of $n=200$ and above, we realize a power of over 85%.

Example Sensory Descriptive Analysis

Attention

Ensure that the package *SensoMineR* (Husson et al., 2023; Lê & Husson, 2008) has been loaded (see above).

Loading datasets

The package *SensoMineR* provides two data sets that we will use in this exercise. The first one is called *sensochoc* and the second one is called *hedochoc*. We load them with the following command:

```
data(chocolates)
```

The data here refer to six varieties of dark LINDT chocolates sold in France (Pagès & Husson, 2001). The six chocolate varieties are:

1. Excellence Noir 70%
2. Qualité amère
3. Mi-doux
4. Amazonie
5. Pâtissier noir
6. Extra supérieur

There are two data frames:

1. *sensochoc*: a data frame with 348 rows and 19 columns: 4 qualitative variables (*Panelist*, *Session*, *Rank*, *Product*) and 14 sensory descriptors from sensory descriptive analysis (from 0 to 10 [10 for a very strong intensity])
2. *hedochoc*: a data frame with 6 rows and 222 columns: each row corresponds to a bar of chocolate and each column to the hedonic overall liking scores (scale from 0 (greatest liking) to 10 (greatest disliking)) given by one of the 222 consumers participating in an affective product acceptance test.

Let's have a look at the *sensochoc* data:

```
sensochoc |> tibble::as_tibble()
```

```
# A tibble: 348 x 18
  Panelist Session Rank Product CocoaA MilkA CocoaF MilkF Caramel Vanilla
  <fct>      <fct>   <fct> <fct>      <int> <int>   <int> <int>   <int>   <int>
1 1 1        1       1   choc6         7     6     6     5     5     3
2 1 1        1       6   choc3         6     7     2     7     8     4
3 1 1        1       3   choc2         8     6     5     4     7     4
4 1 1        1       5   choc1         7     8     8     3     3     2
5 1 1        1       2   choc4         8     5     4     4     4     4
6 1 1        1       4   choc5         7     5     3     5     6     2
7 2 1        1       1   choc4         6     1     8     1     0     0
8 2 1        1       4   choc3         4     2     3     4     0     0
9 2 1        1       3   choc6         5     1     8     1     0     0
10 2 1       1       5   choc2         5     2     8     1     0     0
# i 338 more rows
# i 8 more variables: Sweetness <int>, Acidity <int>, Bitterness <int>,
#   Astringency <int>, Crunchy <int>, Melting <int>, Sticky <int>,
#   Granular <int>
```

Next, we insert the actual product names.

```
levels(sensochoc$Product) <- c("Excellence Noir 70%",
                               "Qualité amère",
                               "Mi-doux",
                               "Amazonie",
                               "Pâtissier noir",
                               "Extra supérieur")
```

Basic statistics

i Note

We will not analyze properties of the dataset such as panel consensus or performance of each panelist. These are usually steps involved to ensure data quality of the trained panel. For more information, see the [SensoMineR](#) package documentation. These steps are usually carried-out by the panel leader or sensory scientist (often having a nutrition science background), before the results of the sensory descriptive analysis are used for further analyses by market researchers.

Get an overview of the mean values per product x descriptor. Below, first we transform the dataset *sensochoc* into a tibble. Then, we group the subsequent data by product and calculate the mean values for each sensory descriptor. Afterward, we transpose the dataset. Then, we add a new variable housing the

names of the sensory descriptors. Finally, we set the variables labels to the product names and deselect the first row.

```
sensochoc |>
  tibble::as_tibble() |>
  dplyr::group_by(Product) |>
  dplyr::summarise(dplyr::across(where(is.numeric),
                                ~ round(mean(.x,
                                              na.rm = TRUE),
                                              digits = 2))) |>
  t() |>
  tibble::as_tibble(, rownames = "descriptor") |>
  setNames( c("descriptor", levels(sensochoc$Product))) |>
  dplyr::slice(-1) |>
  print(width = Inf)
```

```
# A tibble: 14 x 7
  descriptor `Excellence Noir 70%` `Qualité amère` `Mi-doux` Amazonie
  <chr>      <chr>                  <chr>      <chr>      <chr>
1 CocoaA    7.09                    6.55        4.67        6.26
2 MilkA     3.59                    4.00        6.05        4.10
3 CocoaF    8.07                    6.91        3.38        6.69
4 MilkF     1.57                    2.38        7.71        2.59
5 Caramel   1.67                    2.78        6.33        2.67
6 Vanilla   1.10                    1.81        3.67        2.12
7 Sweetness 3.14                    4.62        7.60        4.29
8 Acidity   4.66                    3.14        1.57        3.93
9 Bitterness 7.07                   4.95        1.40        5.19
10 Astringency 4.76                   3.16        1.21        3.69
11 Crunchy   5.97                    7.71        2.98        6.10
12 Melting   4.74                    4.33        7.31        4.38
13 Sticky    3.76                    3.83        5.03        4.10
14 Granular  3.45                    3.16        1.60        3.55
  `Pâtissier noir` `Extra supérieur`
  <chr>            <chr>
1 6.79            6.36
2 4.17            4.57
3 6.79            6.22
4 3.12            3.36
5 3.41            3.26
6 1.79            1.91
7 5.22            5.62
8 3.09            2.67
9 4.88            4.19
10 3.10           2.76
```

11	6.64	7.33
12	4.74	4.21
13	3.22	3.93
14	3.07	3.17

Next, we test which descriptors significantly discriminate between products. The *SensoMineR* package provides the function *decat()* to perform a one-way ANOVA for each descriptor. The function returns a table with the F-statistic, degrees of freedom, and p-value for each descriptor. The function *decat()* requires the following arguments:

- *donnee*: the data frame containing the sensory data
- *formul*: a formula specifying the model to be fitted (e.g., “~Product+Panelist” for a one-way repeated measures ANOVA with Product as the independent variable and Panelist as a random effect [to account for the multiple observations per panelist])
- *firstvar*: the index of the first variable to be analyzed (in this case, 5, as the first descriptors start at the 5th column in the data frame)
- *graph*: whether to plot the results (default is TRUE, but we set it to FALSE here)

We save the results in a new object called *test_results* and then print the table with the F-statistics and p-values rounded to three decimal places. Note, that the F-value is called *Vtest* (for variance test) in the output of the *decat()* function.

```
test_results <- SensoMineR::decat(donnee = sensochoc,
                                formul = "~Product+Panelist",
                                firstvar = 5,
                                graph = FALSE)

test_results$tabF |> round(x=_, digits = 3)
```

	Vtest	P-value
CocoaA	7.070	0
MilkA	5.926	0
CocoaF	13.481	0
MilkF	16.399	0
Caramel	11.532	0
Vanilla	7.318	0
Sweetness	11.441	0
Acidity	7.723	0

Bitterness	13.343	0
Astringency	8.959	0
Crunchy	12.662	0
Melting	8.415	0
Sticky	3.563	0
Granular	4.371	0

💡 Interpretation

From the output above, we understand that all sensory descriptors significantly discriminate between the products ($p < 0.001$). The F-statistic (Vtest) indicates the strength of the discrimination, with higher values suggesting stronger differences between products.

Data visualizations

Now, we prepare a series of data visualizations. For this purpose, we will use the function `panellipse()` from the *SensoMineR* package. This function allows us to create a PCA biplot with confidence ellipses for each product. The function requires the following arguments:

- *donnee*: the data frame containing the sensory data
- *col.p*: the index of the column containing the product names (in this case, 4)
- *col.j*: the index of the column containing the panelist names (in this case, 1)
- *firstvar*: the index of the first descriptor variable (in this case, 5)
- *nbsimul*: the number of simulations to perform for the confidence ellipses (default is 500)
- *scale.unit*: whether to scale the data to unit variance (default is TRUE)
- *variability.variable*: whether to plot the variability of each descriptor (default is TRUE)
- *graph.type*: the type of graph to produce (in this case, “ggplot” for a ggplot2-based graph)

We save the results in a new object called *graphical_results*. The function returns a list containing the PCA results and the ggplot2 graphs.

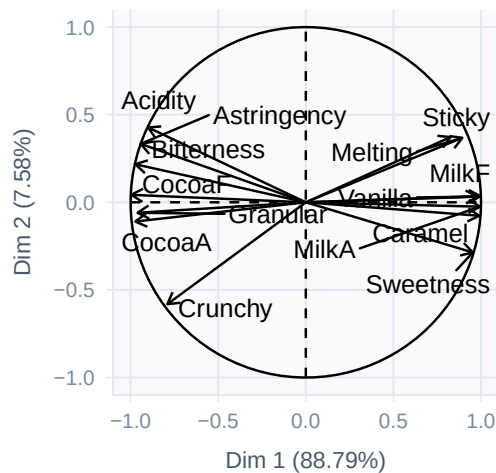
```
graphical_results <- SensoMineR::panellipse(donnee =
sensochoc[,c("Panelist", "Session", "Rank", "Product",
"CocoaA", "MilkA", "CocoaF", "MilkF", "Caramel", "Vanilla",
"Sweetness",
"Acidity", "Bitterness", "Astringency", "Crunchy", "Melting",
"Sticky",
```

```
"Granular"]],
col.p = 4,
col.j = 1,
firstvar = 5,
nbsimul = 1000,
scale.unit = TRUE,
variability.variable = TRUE,
graph.type = "ggplot")
```

Let us begin with the first plot:

```
graphical_results$graph$plotVar
```

PCA graph of variables



💡 Interpretation

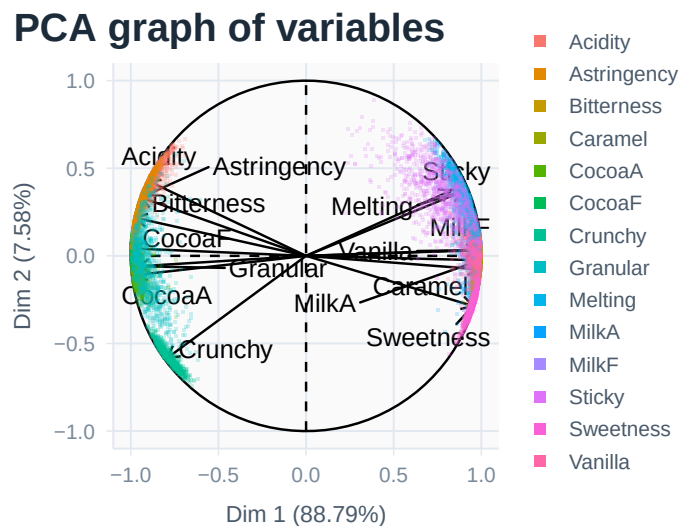
We see a PCA-Plot (based on a principal component analysis, please review our lecture examples in [Marketing Methods and Analysis](#)) (Lichters, 2024/2025) of the perceptual space of the trained panel. We see all descriptors as position vectors (these are the factor/component loadings). Two descriptors are highly positively correlated if the angle between them is slight. Two perfectly negatively correlated descriptors would point in opposite directions (e.g., have a look at *Acidity* vs. *Sweetness*). Further, we see how much information about the panel's product evaluations is represented by the first two principal components. Generally, both axes should

explain at least 75-80% of the information in the panelists' perceptions. Otherwise, a 2D map may miss important information on the products' dissimilarity. The length of each vector further corresponds to the *communality* in the PCA. Put differently, descriptors possessing short vectors are not well explained in terms of the first two principal components.

The map above helps to understand which sensory descriptors usually go hand in hand with the evaluated product category. For example, see the slight angle between *Vanilla*, *Caramel*, *Sweetness* in taste.

Next, we call for the second plot:

```
graphical_results$graph$plotVarVariab
```



💡 Interpretation

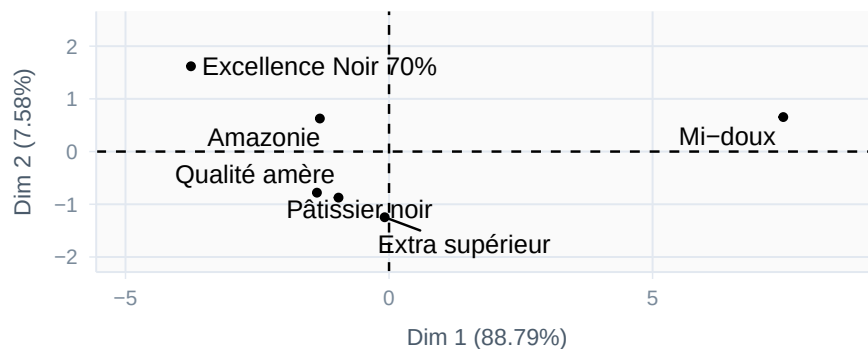
This graphic is very similar to the first one. It, again, presents the first two principal components in the space of descriptors. This time, however, the plot also provides an impression of the uncertainty, or better said, the (dis)agreement in the panelists' scoring concerning each descriptor, drawing on *bootstrapping* (Efron & Gong, 1983; Efron & Tibshirani, 1994). Here, we can see that the panel's agreement is excellent for some descriptors (e.g., *Sweetness*). In contrast, it is relatively low for others, as indicated by an expansive cloud of individual evaluations (e.g., *Sticky*).

Product developers should trust the panel's ability to use descriptors with low variance (vs. high variance) more.

We move on to the next plot:

```
graphical_results$graph$plotInd
```

PCA graph of individuals



💡 Interpretation

This biplot provides us with information on the product's location in the space of the sensory perception as perceived by the panel. More precisely, we see the factor/ component scores of each product. The logic is that products that are very far away from each other on a particular axis are very dissimilar in sensory perception. Products that are very close to each other are very similar in terms of sensory descriptors.

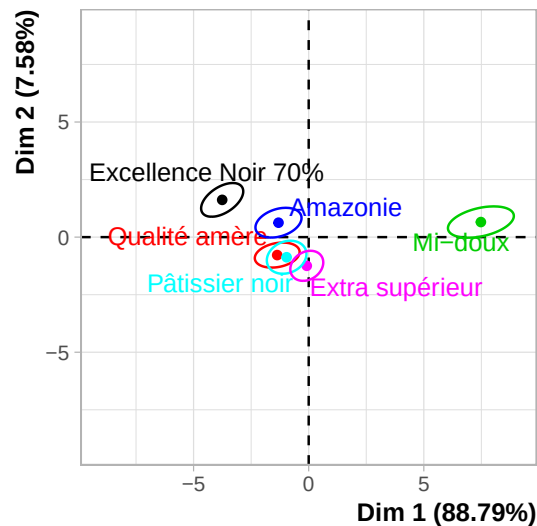
In practice, this information is essential to understanding the competitive landscape. It provides companies with information on their products' sensory (dis)similarity in the assortment, R&D samples, and competitors.

Empty spaces in the product perception map may also encourage product innovation. Above, we also see that the trained panelists perceive the product *Mi-doux* as very different from other products. One plot above, we already noticed that this product is characterized by a very *Sticky* and *Melting* mouth feel, but it's all but *Crunchy*. If, in a sensory product acceptance test, it turns out that a profitable segment of consumers strongly prefers *Mi-doux* over other products, this would be an opportunity for Lindt's competitors to "attack" with a product innovation.

In addition, we can enrich the above data visualization with uncertainty estimates in the form of 95% bootstrapped confidence ellipses. In the code chunk below, we also set a new x- and y-axis limit to also visually communicate that the products are not very dissimilar in terms of their factor scores concerning the second dimension. We also added a new title to the plot.

```
graphical_results$graph$plotIndEll +
  xlim(-9, +9) +
  ylim(-9, 9) +
  ggtitle("PCA of trained panel perception with confidence
  ellipses")
```

PCA of trained panel perception with confidence ellipses



💡 Interpretation

This helps us understand which products do, overall, significantly differ from each other in terms of their factor scores (which the perception of the trained panel constitutes) and which do not. For example, the confidence ellipse for *Excellence Noir 70%* does not overlap with other products. The product is, therefore, very sensory different from other products. This is not true for the three products at the bottom of the plot, which have a lot of overlap.

Combining Sensory Descriptive Analysis and Affective Product Acceptance Tests through External Preference Mapping

Finally, we built a bridge between the trained panel's perceptions and the naive consumer liking judgment coming from a sensory product acceptance test.

Let us first explore the hedonic dataset containing the product likings.

```
hedochoc |> tibble::as_tibble()
```

```
# A tibble: 6 x 222
      j1    j2    j3    j4    j5    j6    j7    j8    j9   j10   j11   j12   j13
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1     5     3     4     6     3     3     2     2     2     5     6     7     4
2     8     6     6     6     2     7     5     6     4     3     6     4     4
3     4     5     8     2     7     6     5     7     4     4     3     4     6
4     3     6     5     1     5     8     4     5     7     1     4     8     2
5     3     7     6     5     7     7     3     5     7     7     2     8     4
6     8     7     6     4     2     8     3     6     7     2     4     2     4
# i 209 more variables: j14 <dbl>, j15 <dbl>, j16 <dbl>, j17 <dbl>, j18 <dbl>,
#   j19 <dbl>, j20 <dbl>, j21 <dbl>, j22 <dbl>, j23 <dbl>, j24 <dbl>,
#   j25 <dbl>, j26 <dbl>, j27 <dbl>, j28 <dbl>, j29 <dbl>, j30 <dbl>,
#   j31 <dbl>, j32 <dbl>, j33 <dbl>, j34 <dbl>, j35 <dbl>, j36 <dbl>,
#   j37 <dbl>, j38 <dbl>, j39 <dbl>, j40 <dbl>, j41 <dbl>, j42 <dbl>,
#   j43 <dbl>, j44 <dbl>, j45 <dbl>, j46 <dbl>, j47 <dbl>, j48 <dbl>,
#   j49 <dbl>, j50 <dbl>, j51 <dbl>, j52 <dbl>, j53 <dbl>, j54 <dbl>, ...
```

We see that each row corresponds to a bar of chocolate and each column to the overall liking scores (scale from 0 (greatest liking) to 10 (greatest disliking)). We first change this rather untypical coding (i.e., highest score should be 10 and lowest 0).

in the code chunk below, we use the *mutate()* function from the *dplyr* package (Wickham, François, Henry, Müller, & Vaughan, 2023) to change the coding of the liking scores. We use the *across()* function to apply the transformation to all numeric columns (is.numeric) in the dataset. The transformation is done by subtracting each score from 10, effectively reversing the scale.

```
hedochoc <- hedochoc |>
  dplyr::mutate(dplyr::across(where(is.numeric), ~ 10 - .x))
```

Then, we transpose the dataset to understand the mean product's liking. We also assign a unique case ID to each consumer.

```
hedochoc_transposed <- hedochoc |> t() |> tibble::as_tibble() |>
tibble::rowid_to_column("ID")
```

In a subsequent step, we are inserting the actual product names.

```
names(hedochoc_transposed) <- c("ID",
                                "Excellence Noir 70%",
                                "Qualité amère",
                                "Mi-doux",
                                "Amazonie",
                                "Pâtissier noir",
                                "Extra supérieur")
```

Afterward, we tell *R* to handle the case *ID* as a factor.

```
hedochoc_transposed$ID <-
haven::as_factor(hedochoc_transposed$ID)
```

Lets look at the transposed dataset again

```
hedochoc_transposed
```

```
# A tibble: 222 x 7
   ID      `Excellence Noir 70%` `Qualité amère` `Mi-doux` Amazonie
  <fct>                <dbl>         <dbl>      <dbl>      <dbl>
1 1      5                    2            6          7
2 2      7                    4            5          4
3 3      6                    4            2          5
4 4      4                    4            8          9
5 5      7                    8            3          5
6 6      7                    3            4          2
7 7      8                    5            5          6
8 8      8                    4            3          5
9 9      8                    6            6          3
10 10     5                    7            6          9
# i 212 more rows
# i 2 more variables: `Pâtissier noir` <dbl>, `Extra supérieur` <dbl>
```

Next, we use the function *pivot_longer* (Wickham, Vaughan, & Girlich, 2024) to restructure our dataset from the **wide format** (each consumer is one row) to the **long format** (each product judgment is one row). Thus, we are *stacking* the data. Within the function call, we hand in *hedochoc_transposed* for the data

argument. Further, we tell the function to use the columns named “Excellence Noir 70%” to “Extra supérieur” to create the longer format. The *names_to* argument tells the function which name to use for the new stacked variable collecting all product variant names. Finally, the *values_to* argument sets the name for the stacked product evaluations. The results are stored in a new object called *d_stacked*.

```
hedochoc_stacked <- tidyr::pivot_longer(
  data = hedochoc_transposed,
  cols = "Excellence Noir 70%":"Extra supérieur",
  names_to = "product_sample",
  values_to = "liking")

hedochoc_stacked
```

```
# A tibble: 1,332 x 3
  ID   product_sample   liking
  <fct> <chr>           <dbl>
1 1     Excellence Noir 70%     5
2 1     Qualité amère         2
3 1     Mi-doux              6
4 1     Amazonie             7
5 1     Pâtissier noir        7
6 1     Extra supérieur       2
7 2     Excellence Noir 70%     7
8 2     Qualité amère         4
9 2     Mi-doux              5
10 2    Amazonie             4
# i 1,322 more rows
```

Now, we can ask for some descriptive statistics.

```
hedochoc_stacked |>
  dplyr::group_by(product_sample) |>
  dplyr::summarise(
    n = n(),
    mean = mean(liking),
    median = median(liking),
    sd = sd(liking)
  )
```

```
# A tibble: 6 x 5
  product_sample      n mean median    sd
  <chr>          <int> <dbl> <dbl> <dbl>
```

1 Amazonie	222	4.23	4	2.30
2 Excellence Noir 70%	222	4.67	5	2.55
3 Extra supérieur	222	4.09	4	2.00
4 Mi-doux	222	4.23	4	2.66
5 Pâtissier noir	222	4.37	4	2.05
6 Qualité amère	222	4.31	4	2.22

Interpretation

On average, the product *Excellence Noir 70%* is the most liked product with a mean of 4.67 and a standard deviation of 2.55. In contrast, the product *Extra supérieur* is the least liked product with a mean of 4.09 and a standard deviation of 2. The standard deviations indicate a lot of variance in the liking scores, especially for the products *Mi-doux* and *Extra supérieur*, a sign of preference heterogeneity (Lichters, Möslin, Sarstedt, & Scharf, 2021). Thus, searching for distinct consumer segments is indicated.

In the last step, we will achieve multiple analysis goals. First, we cluster the consumers according to their idiosyncratic taste preferences, using this segment-wise information to superimpose an external preference mapping (Naughton, Schramm, & Lichters, 2025) on the space spanned by the sensory descriptors (Greenhoff & MacFie, 1994).

In an initial step, we need to do some data wrangling on the data provided by the sensory panelists.

```
#first save means by descriptor
sensochoc_aggregated <- sensochoc |>
  tibble::as_tibble() |>
  dplyr::group_by(Product) |>
  dplyr::summarise(dplyr::across(where(is.numeric), ~ mean(.x,
na.rm = TRUE)))

# delete first row which contains the product names
sensochoc_aggregated <- sensochoc_aggregated |>
dplyr::select(-Product)

# Conduct principal component analysis on descriptive panel data
sensochoc_aggregated_PCA <- FactoMineR::PCA(sensochoc_aggregated,
graph = FALSE)

# Save coordinates of the PCA's first 2 dimensions (the component
scores)
senso_data <- sensochoc_aggregated_PCA$ind$coord[,1:2] |>
as.data.frame()
```

```
# adjust row names in the data set containing the sensory
coordinates
row.names(senso_data) <- c("Excellence_Noir",
                           "Qualité_amère",
                           "Mi_doux",
                           "Amazonie",
                           "Pâtissier_noir",
                           "Extra_supérieur")

#adjust row names in the data set containing the consumers'
liking scores
row.names(hedochoc) <- c("Excellence_Noir",
                        "Qualité_amère",
                        "Mi_doux",
                        "Amazonie",
                        "Pâtissier_noir",
                        "Extra_supérieur")
```

Basically, we have conducted a PCA on the mean values for the descriptors provided by the panel (see *sensochoc_aggregated*). The component scores of products of this PCA constitute the starting point for the preference mapping.

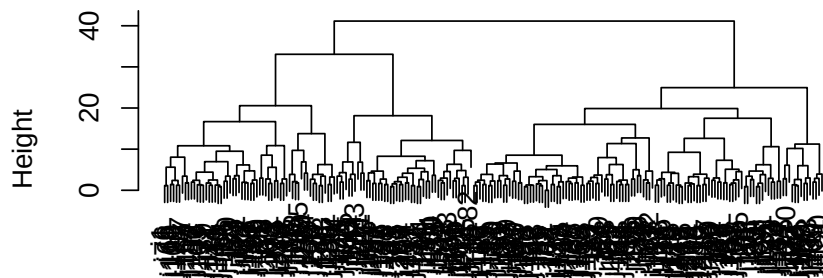
Next, we use the function *carto()* from the *SensoMineR* package to perform the external preference mapping. The function requires the following arguments:

- *Mat*: the matrix containing the PCA results of the sensory panel (the component scores corresponding to the products)
- *MatH*: the data frame containing the hedonic scores (the consumer liking scores)
- *graph.tree*: whether to plot a tree graph (i.e., the dendrogram) of the clusters (default is TRUE)
- *graph.corr*: whether to plot a correlation circle (default is TRUE)
- *graph.carto*: whether to plot the preference mapping (default is TRUE)

When executing this function, we will also obtain a dendrogram of the clusters and a correlation circle. The dendrogram helps us to understand how the consumers cluster according to their liking scores. The correlation circle visualizes the relationship between the segment-wise consumer preferences. Regarding the clustering, the function automatically applies a hierarchical clustering algorithm (*Ward's method* (Ward, 1963)) to the consumer liking scores. Then it selects an optimal number of clusters based on the *silhouette* method (Rousseeuw, 1987). In a subsequent step, the function takes the obtained cluster centers for further optimization of the cluster solution, drawing on *k-means* clustering (Hartigan & Wong, 1979).

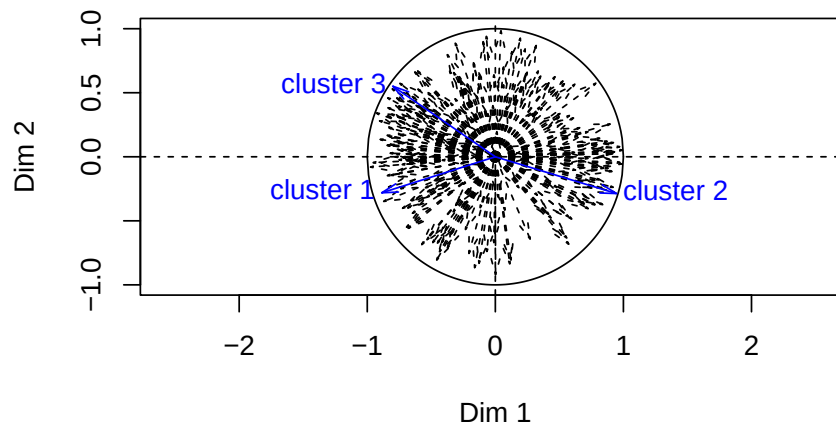
```
# simultaneous clustering and preference mapping
preference_mapping_choc <- SensoMineR::carto(
  Mat = senso_data,
  MatH = as.data.frame(hedochoc),
  regmod = 2,
  asp = 3,
  graph.tree = TRUE,
  graph.corr = TRUE,
  graph.carto = TRUE)
```

Cluster Dendrogram

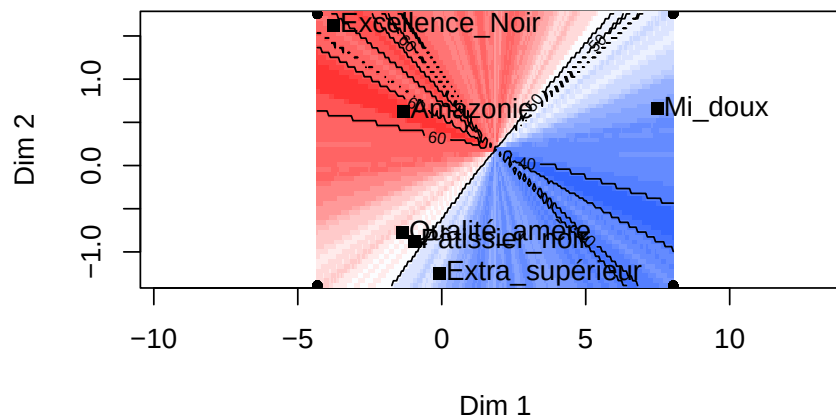


Panelists
Agglomerative Coefficient = 0.95

Correlation circle



Preference mapping



💡 Interpretation

First, the deprogram shows us that the consumers cluster optimally into three groups, because when moving further upward in the tree, the within-cluster heterogeneity is rising rapidly.

Second, the correlation circle visualizes the relationship between the

segment-wise consumer preference vectors and the individual consumers' preferences. What we see is that consumers in cluster 1 hold very different preferences as compared to consumers in clusters 2 and 3, because their preference vectors point almost in the opposite direction. In contrast, consumers in clusters 2 and 3 have somewhat similar preferences, as their preference vectors point more or less in the same direction. Importantly, this circle plot is not based on a PCA drawing on the sensory panel's perception, such as above. Instead, it is based on consumer preferences. Therefore, it makes no sense to compare the direction of the segment-wise vectors with the sensory descriptors. Instead, we can only compare the direction of the segment-wise vectors with each other. The further apart two vectors are, the more different the consumers' preferences are.

Finally, we see the external preference mapping in the heat map. The heat map visualizes areas of estimated high product liking in the space which is defined by the trained panel's product perception. The color intensity indicates the level of preference, with darker reddish colors indicating higher liking scores.

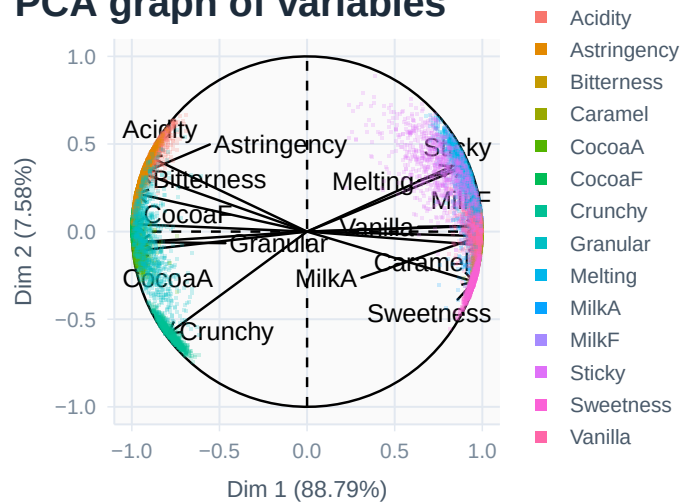
Interestingly, we see that products Excellence_Noir_70 and Amazonie are located in dark reddish areas, whereas products Extra_supérieur and especially Mi-doux are less liked.

Also, keep in mind that the colours are determined by the liking scores of the individual consumers. It's a majority vote on preferences. As a consequence, it does not mean that Mi-doux is not liked at all. Instead, it means that the majority of consumers do not like this product.

We can compare the obtained heat map to the PCA plot including the component loadings of the descriptors above.

```
graphical_results$graph$plotVarVariab
```

PCA graph of variables



💡 Interpretation

The PCA plot shows the sensory panel's perception of the products, while the heat map shows the consumer preferences. The two plots can be complementary, as they provide insights into how the sensory attributes of the products relate to consumer preferences. We can conclude that the well-received products (e.g., Excellence Noir 70% and Amazonie) are characterized by high *cocoa* content, impression of *bitterness* and high *Acidity* and low *Sweetness*, while the less liked products (e.g., Extra supérieur and Mi-doux) are characterized by lower cocoa content and higher sweetness. This suggests that consumers prefer products with a higher cocoa content and lower sweetness.

Finally, we should look at the segment-wise results more deeply. To do so, we first extract the cluster assignments from the *preference_mapping_choc* object. Then, we add the cluster assignments to the transposed hedonic dataset. next, we calculate the mean liking score for each product by cluster.

```
cluster_assignments <- preference_mapping_choc$clusters |>
  as.numeric()

hedochoc_transposed <- hedochoc_transposed |>
  dplyr::mutate(cluster = cluster_assignments)

hedochoc_transposed$cluster <-
  haven::as_factor(hedochoc_transposed$cluster)
```

```
hedochoc_transposed |>
  dplyr::group_by(cluster) |>
  dplyr::summarise(
    n = n(),
    Excellence_Noir = mean(`Extra supérieur`),
    Qualité_amère = mean(`Qualité amère`),
    Mi_doux = mean(`Mi-doux`),
    Amazonie = mean(`Amazonie`),
    Pâtissier_noir = mean(`Pâtissier noir`),
    Extra_supérieur = mean(`Extra supérieur`)
  )
```

```
# A tibble: 3 x 8
  cluster      n Excellence_Noir Qualité_amère Mi_doux Amazonie Pâtissier_noir
  <fct>    <int>          <dbl>          <dbl>    <dbl>    <dbl>          <dbl>
1 1         89           3.16           3.04     2.75     3.06           3.20
2 2         50           5.44           5.26     7.48     4.22           4.84
3 3         83           4.29           5.10     3.87     5.51           5.35
# i 1 more variable: Extra_supérieur <dbl>
```

Interpretation

We can see that the first consumer segment is the smallest one, while the second segment is the biggest. We also see why focussing on consumer segments is important. The mean liking scores for the products are very different across the segments. For example, consumers in cluster 1 like the sensory unique product *Mi-doux* the most. This highlights the importance of understanding consumer preference heterogeneity and segmenting the market accordingly.

-
- Chapman, C., & Feit, E. M. (2019). *R for marketing research and analytics*. doi: [10.1007/978-3-030-14316-9](https://doi.org/10.1007/978-3-030-14316-9)
- Christensen, R. H. B., & Brockhoff, P. B. (2023). *sensR: Thurstonian models for sensory discrimination*. Retrieved from <https://CRAN.R-project.org/package=sensR>
- Efron, B., & Gong, G. (1983). A leisurely look at the bootstrap, the jack-knife, and cross-validation. *The American Statistician*, 37(1), 36–48. doi: [10.1080/00031305.1983.10483087](https://doi.org/10.1080/00031305.1983.10483087)

- Efron, B., & Tibshirani, R. J. (1994). *An introduction to the bootstrap*. doi: [10.1201/9780429246593](https://doi.org/10.1201/9780429246593)
- Field, A. P. (2026). *Discovering statistics using r & rstudio* (2nd ed.). Retrieved from <https://study.sagepub.com/dsur2>
- Greenhoff, K., & MacFie, H. J. H. (1994). *Preference mapping in practice*. doi: [10.1007/978-1-4615-2171-6_6](https://doi.org/10.1007/978-1-4615-2171-6_6)
- Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Applied Statistics*, 28(1), 100. doi: [10.2307/2346830](https://doi.org/10.2307/2346830)
- Husson, F., Le, S., & Cadoret, M. (2023). *SensoMineR: Sensory data analysis*. Retrieved from <https://CRAN.R-project.org/package=SensoMineR>
- Lawless, H. T., & Heymann, H. (2010). *Sensory evaluation of food: Principles and practices* (2nd ed.). New York, Dordrecht, Heidelberg, London: Springer.
- Lê, S., & Husson, F. (2008). SENSOMINER: A package for sensory data analysis. *Journal of Sensory Studies*, 23(1), 14–25. doi: [10.1111/j.1745-459x.2007.00137.x](https://doi.org/10.1111/j.1745-459x.2007.00137.x)
- Lê, S., & Worch, T. (2018). *Analyzing sensory data with r*. doi: [10.1201/9781315373416](https://doi.org/10.1201/9781315373416)
- Lichters, M. (2024). *Marketing methods and analysis*. doi: [10.24352/UB.OVGU-2024-085](https://doi.org/10.24352/UB.OVGU-2024-085)
- Lichters, M. (2025). *Marketing methods and analysis: R examples used during market research lectures*. doi: [10.24352/UB.OVGU-2024-085](https://doi.org/10.24352/UB.OVGU-2024-085) (Original work published 2024)
- Lichters, M., Möslin, R., Sarstedt, M., & Scharf, A. (2021). Segmenting consumers based on sensory acceptance tests in sensory labs, immersive environments, and natural consumption settings. *Food Quality and Preference*, 89, 104138. doi: [10.1016/j.foodqual.2020.104138](https://doi.org/10.1016/j.foodqual.2020.104138)
- Næs, T., Brockhoff, P. B., & Tomic, O. (2010). *Statistics for sensory and consumer science*. doi: [10.1002/9780470669181](https://doi.org/10.1002/9780470669181)
- Naughton, P., Schramm, J. B., & Lichters, M. (2025). The eyes eat first: Improving consumer acceptance of plant-based meat alternatives by adjusting front-of-pack labeling. *Food Quality and Preference*, 131, 105567. doi: [10.1016/j.foodqual.2025.105567](https://doi.org/10.1016/j.foodqual.2025.105567)
- Pagès, J., & Husson, F. (2001). Inter-laboratory comparison of sensory profiles. *Food Quality and Preference*, 12(5-7), 297–309. doi: [10.1016/s0950-3293\(01\)00015-5](https://doi.org/10.1016/s0950-3293(01)00015-5)
- Rinker, T. W., & Kurkiewicz, D. (2018). *{Pacman}: {P}ackage management for {r}*. Retrieved from <http://github.com/trinker/pacman>
- Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20, 53–65. doi: [10.1016/0377-0427\(87\)90125-7](https://doi.org/10.1016/0377-0427(87)90125-7)
- Schlich, P. (1993). Risk tables for discrimination tests. *Food Quality and Preference*, 4(3), 141–151. doi: [10.1016/0950-3293\(93\)90157-2](https://doi.org/10.1016/0950-3293(93)90157-2)
- Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301), 236–244. doi: [10.1080/01621459.1963.10500845](https://doi.org/10.1080/01621459.1963.10500845)

- Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., ... Yutani, H. (2019). *Welcome to the {tidyverse}*. 4, 1686. doi: [10.21105/joss.01686](https://doi.org/10.21105/joss.01686)
- Wickham, H., François, R., Henry, L., Müller, K., & Vaughan, D. (2023). *Dplyr: A grammar of data manipulation*. Retrieved from <https://CRAN.R-project.org/package=dplyr>
- Wickham, H., & Henry, L. (2023). *Purrr: Functional programming tools*. Retrieved from <https://CRAN.R-project.org/package=purrr>
- Wickham, H., Vaughan, D., & Girlich, M. (2024). *Tidyr: Tidy messy data*. Retrieved from <https://CRAN.R-project.org/package=tidyr>
- Worch, T., Delarue, J., De Souza, V. R., & Ennis, J. (2023). *Data science for sensory and consumer scientists*. doi: [10.1201/9781003028611](https://doi.org/10.1201/9781003028611)