

Marketing Methods and Analysis: R Examples used during Market Research Lectures

Marcel Lichters

2024-03-30 (Last modified: October 7, 2026)

Abstract

Goal: The students should practice analyzing data with R and RStudio. Another goal is to make students familiar with functions from modern packages used in applied market research tasks. These skills will serve them well throughout their studies and beyond. I intentionally move away from base R, and I will routinely update this script for newer developments (e.g., new package releases). The internet provides countless further examples. In addition, there are very good textbooks. Throughout the script, I reference the literature and the packages used. The script follows the structure of the lecture slides at the Otto von Guericke University Magdeburg (Germany). The examples are based on the lecture slides and the textbook by Sarstedt & Mooi (2019). I first present examples on analyzing cross-tabulation tables; then we present examples of one-way and Two-way ANOVA, followed by examples of OLS regression. After that, we cover factor analysis, and finally cluster analysis. The script is a work in progress and will be updated regularly. If you have any questions, please contact the author.

Abstract

Goal: The students should practice analyzing data with *R* and *RStudio*. Another goal is to make students familiar with functions from modern packages used in applied market research tasks. These skills will serve them well throughout their studies and beyond. I intentionally move away from base *R*, and I will routinely update this script for newer developments (e.g., new package releases). The internet provides countless further examples. In addition, there are very good textbooks. Throughout the script, I reference the literature and the packages used. The script follows the structure of the lecture slides at the Otto von Guericke University Magdeburg (Germany). The examples are based on the

lecture slides and the textbook by Sarstedt & Mooi (2019). I first present examples on analyzing cross-tabulation tables; then we present examples of one-way and Two-way ANOVA, followed by examples of OLS regression. After that, we cover factor analysis, and finally cluster analysis. The script is a work in progress and will be updated regularly. If you have any questions, please contact the author.

Further Information

R analysis script presenting the examples discussed in the lecture's slides.

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

Download the pdf version [here](#)

💡 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 1: Faculty of Economics and Management at the Otto von Guericke University in Magdeburg (Germany) Magdeburg

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 if the package `pacman` (Rinker & Kurkiewicz, 2018) is already installed to 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)). 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(abdiv, car, cv, datawizard, dendextend,
               descr, effectsize,
               formatR, fpc, ggplot2, ggpubr,
               ggstatsplot, haven, knitr, labelled,
               lmtest, lm.beta, naniar, NbClust,
               nhstplot, nomclust, marginaleffects,
               modelsummary, patchwork, PMCMRplus,
               psych, pwr, pwrss, qgraph, RESI,
               rstatix, sandwich, styler, tidyverse,
               vegan, webshot2, xml2)
```

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] parallel stats      graphics grDevices utils      datasets methods
[8] base
```

```
other attached packages:
```

```
[1] xml2_1.6.0          webshot2_0.1.2      vegan_2.7-5
[4] permute_0.9-10      lubridate_1.9.5     forcats_1.0.1
[7] stringr_1.6.0       dplyr_1.2.1         purrr_1.2.2
[10] readr_2.2.0         tidyr_1.3.2         tibble_3.3.1
[13] tidyverse_2.0.0     styler_1.11.0       sandwich_3.1-3
[16] rstatix_1.1.0       RESI_1.5.1          quarto_1.5.1
[19] qgraph_1.10.1       pwrss_1.2.0         pwr_1.3-0
[22] psych_2.6.5         PMCMRplus_1.9.12    patchwork_1.3.2
[25] modelsummary_2.6.0  marginaleffects_1.0.0 nomclust_2.8.1
[28] clValid_0.7         cluster_2.1.8.3     nhstplot_1.4.2
[31] NbClust_3.0.1       naniar_1.1.0        lm.beta_1.7-3
[34] lmtest_0.9-40       zoo_1.9-0           labelled_2.16.0
[37] knitr_1.51          haven_2.5.5         ggstatsplot_1.1.1
[40] ggpubr_1.0.0        ggplot2_4.0.3       fpc_2.2-15
[43] formatR_1.14        effectsize_1.0.3    downloadthis_0.5.0
[46] downlit_0.4.5       descr_1.1.9         dendextend_1.19.1
[49] datawizard_1.3.1    cv_2.0.6            doParallel_1.0.17
[52] iterators_1.0.14    foreach_1.5.2       car_3.1-5
[55] carData_3.0-6       abdiv_0.2.0         pacman_0.5.1
```

loaded via a namespace (and not attached):

[1] clubSandwich_0.7.0	later_1.4.8	splines_4.6.1
[4] R.oo_1.27.1	rpart_4.1.27	lifecycle_1.0.5
[7] Rdpack_2.6.6	processx_3.9.0	lattice_0.23-1
[10] prabclus_2.3-5	MASS_7.3-66	insight_1.5.4
[13] backports_1.5.1	magrittr_2.0.5	Hmisc_5.2-6
[16] rmarkdown_2.32	yaml_2.3.12	otel_0.2.0
[19] flexmix_2.3-21	chromote_0.5.1	pbapply_1.7-5
[22] minqa_1.2.8	RColorBrewer_1.1-3	multcomp_1.4-32
[25] abind_1.4-8	quadprog_1.5-8	R.cache_0.17.0
[28] R.utils_2.13.0	nnet_7.3-21	TH.data_1.1-5
[31] correlation_0.8.8	codetools_0.2-20	tidyselect_1.2.1
[34] SuppDists_1.1-9.9	farver_2.1.2	lme4_2.0-6
[37] gmp_0.7-5.1	viridis_0.6.5	stats4_4.6.1
[40] base64enc_0.1-6	jsonlite_2.0.0	Formula_1.2-6
[43] survival_3.8-11	emmeans_2.0.4	BWstest_0.2.3
[46] tools_4.6.1	Rcpp_1.1.2	glue_1.8.1
[49] mnormt_2.1.2	gridExtra_2.3.1	mgcv_1.9-4
[52] xfun_0.60	kSamples_1.2-12	websocket_1.4.4
[55] withr_3.0.3	fastmap_1.2.0	boot_1.3-32
[58] digest_0.6.39	timechange_0.4.0	R6_2.6.1
[61] estimability_2.0.0	colorspace_2.1-3	visdat_0.6.0
[64] gtools_3.9.5	jpeg_0.1-11	diptest_0.77-2
[67] R.methodsS3_1.8.2	generics_0.1.4	data.table_1.18.4
[70] corpcor_1.6.10	robustbase_0.99-7	class_7.3-24
[73] htmlwidgets_1.6.4	parameters_0.29.3	pkgconfig_2.0.3
[76] gtable_0.3.6	modeltools_0.2-24	Rmpfr_1.1-2
[79] statsExpressions_2.1.1	S7_0.2.2	htmltools_0.5.9
[82] lavaan_0.7-2	multcompView_0.1-12	scales_1.4.0
[85] png_0.1-9	reformulas_0.4.4	rstudioapi_0.19.0
[88] tzdb_0.5.0	reshape2_1.4.5	coda_0.19-4.1
[91] checkmate_2.3.4	nlme_3.1-170	nloptr_2.2.1
[94] cachem_1.1.0	foreign_0.8-91	pillar_1.11.1
[97] grid_4.6.1	vctrs_0.7.3	promises_1.5.0
[100] xtable_1.8-8	htmlTable_2.5.0	paletteer_1.7.0
[103] pbivnorm_0.6.0	evaluate_1.0.5	mvtnorm_1.4-2
[106] cli_3.6.6	compiler_4.6.1	rlang_1.3.0
[109] rstantools_2.7.1	ggsignif_0.6.4	fdrtool_1.2.18
[112] mclust_6.1.3	rematch2_2.1.2	plyr_1.8.9
[115] stringi_1.8.9	viridisLite_0.4.3	tables_0.9.35
[118] aod_1.3.3	CompQuadForm_1.4.4	bayestestR_0.19.0
[121] Matrix_1.7-6	glasso_1.11	hms_1.1.4
[124] kernlab_0.9-33	rbibutils_2.4.1	igraph_2.3.3
[127] broom_1.0.13	memoise_2.0.1	RcppParallel_6.2.1
[130] DEoptimR_1.2-1		

Cross Tabulation - Example: Gender and Mobile Phone Design.

Testing for Independence

200 respondents (100 males, 100 females) were surveyed regarding their preference for smartphone colors. Expected frequencies (in braces) and observed frequencies are reported in the following table:

Table 1: Color preferences for smartphones

	Male customers	Female customers	Σ
Black	28	12	40
Silver	48	36	84
White	24	52	76
Σ	100	100	200

In our example, the analysis draws on a (Pearson) χ^2 -test (without Yate's continuity correction (Yates, 1934)). Furthermore, we know from lectures in statistics that the df in a χ^2 -test equals: (number of rows -1) * (number of columns -1). Thus, $df = (3-1)*(2-1) = 2$. The degrees of freedom (df) are 2.

We first create the table within *R*:

```
smartphone_table <- matrix(  
  data = c(28, 12, 48, 36, 24, 52),  
  ncol = 2, nrow = 3,  
  byrow = TRUE,  
  dimnames = list(c("black", "silver", "white"), c("male",  
    "female"))  
)
```

Next, we ask for a χ^2 -test (without Yate's continuity correction). Before doing so, we ask for the critical value of the χ^2 test statistic at $\alpha=5\%$ error rate and $df=2$:

```
critical_value <- qchisq(p=0.95, df=2)  
  
critical_value
```

[1] 5.991465

Note

Note that this value perfectly mirrors the one discussed during the lecture. Usually, the market researchers interprets a statistical test results as to be *significant* when the empirical value of the test statistic exceeds the critical value. As seen in the lecture, however, most often research prefer to calculate the empirical *p-value*, which represents the probability of finding a value of the corresponding test statistic that is at minimum the observed one (i.e., the probability of finding such an extreme value when H_0 is true Field, Miles, & Field (2012, p. 54)).

Let us calculate both the test statistic (χ^2) and the corresponding *p-value*.

```
chi2_test <- smartphone_table |> chisq.test(correct = FALSE)

chi2_test
```

Pearson's Chi-squared test

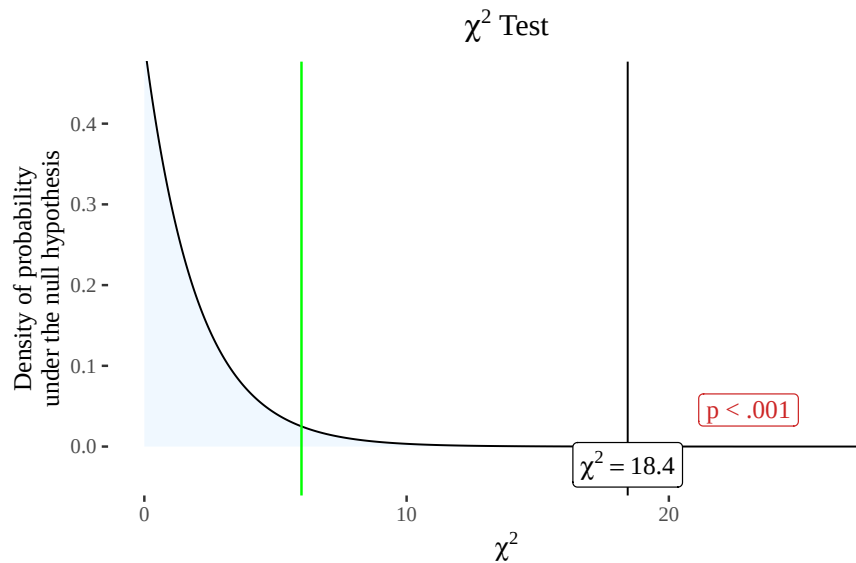
```
data:  smartphone_table
X-squared = 18.43, df = 2, p-value = 9.953e-05
```

Interpretation

The test results align to the lecture calculation. The χ^2 -value of 18.43 leads to a significant p-value of roughly 0. Thus, the data support the hypothesis that both genders prefer different colors for their smartphone.

Let's visualize the test actual statistic value, the critical value (green line) and the theoretical distribution.

```
chi2_test |> nhstplot::plotchisqtest() + ggplot2::geom_vline(
  xintercept = critical_value,
  col = "green"
)
```



We have seen in the lecture that we can also switch to Fisher's exact test ([Fisher, 1922](#)) or Yate's correction for the 2-test. These methods are also commonly used in market research problems ([Lichters, Bengart, Sarstedt, & Vogt, 2017](#); [Lichters, Brunnlieb, Nave, Sarstedt, & Vogt, 2016](#)). We will look at Fisher's exact test next. This test can almost always be applied when contingency tables are not too big.

```
Fishers_test <- smartphone_table |> fisher.test()

Fishers_test
```

Fisher's Exact Test for Count Data

```
data:  smartphone_table
p-value = 0.0001077
alternative hypothesis: two.sided
```

💡 Interpretation

Also, Fisher's exact test is significant based on the given data (the empirical p-value is 0). Note that the test does not come with a test statistic since it is an exact test that does not make assumptions about parameter distributions, etc.

Effect Sizes for Crosstabs tests

In the lecture, we also discussed a series of effect size measures that market researchers routinely apply when analyzing contingency tables (e.g., [Sawyer & Ball, 1981](#)). Here, we have distinguished between 2x2 tables and bigger tables.

2x2 tables

Some effect sizes are only applicable in the case of 2x2 tables. In the lecture, we have discussed the **Phi coefficient** ([Yule, 1912](#)): $\phi = \sqrt{\frac{\chi^2}{N}}$

We first construct a small 2x2 problem to illustrate its calculation:

Table 2: Beer consumption and passing the exam

	MMA exam passed	MMA exam failed	Σ
Regular beer drinker	28	14	42
Not drinking beer	67	94	161
Σ	95	108	203

```
MMA_exam <- matrix(  
  data = c(28, 14, 67, 94),  
  ncol = 2, nrow = 2,  
  byrow = TRUE,  
  dimnames = list(c("Regular beer drinker", "Not drinking beer"),  
    c("MMA exam passed", "MMA exam failed"))  
)
```

The corresponding χ^2 -test (with Yate's correction) would result in:

```
chi2_test_MMA_exam <- MMA_exam |> chisq.test(correct = TRUE)  
chi2_test_MMA_exam
```

Pearson's Chi-squared test with Yates' continuity correction

```
data: MMA_exam  
X-squared = 7.4205, df = 1, p-value = 0.006449
```

Now, we calculate ϕ . Here, we use the `effectsize` package (Ben-Shachar, Lüdtke, & Makowski, 2020), which also corrects phi for small sample sizes by default. It also provides a function (`interpret`) that allows us to interpret an effect strength according to established guidelines (Cohen, 1988).

i call explicitly from a certain package

Note, that below we explicitly use the notation `effectsize::phi`. This clarifies that we want to use the function `phi()` from the package `effectsize` because other packages provide functions with the same name.

```
MMA_exam |>
  effectsize::phi(adjust = TRUE) |>
  effectsize::interpret(rules = "cohen1988")
```

Phi (adj.)	95% CI	Interpretation
0.19	[0.05, 1.00]	small

- One-sided CIs: upper bound fixed at [1.00].
- Interpretation rule: cohen1988

Larger tables

Whereas the **Phi coefficient** is only useful for 2x2 tables, the following effect size measures could additionally be applied to larger tables.

- Next, we go for:

- **Pearson's Contingency Coefficient**: $C = \sqrt{\frac{\chi^2}{\chi^2 + N}}$
- **Cohen's omega** (which is not bound to +1): $\omega = \sqrt{\frac{C}{1-C}}$
- **Cramer's V**: $V = \sqrt{\frac{\chi^2}{n * (\min(k, l) - 1)}}$

```
smartphone_table |>
  effectsize::pearsons_c() |>
  effectsize::interpret(rules = "cohen1988")
```

Pearson's C	95% CI	Interpretation
0.29	[0.17, 1.00]	small

- One-sided CIs: upper bound fixed at [1.00].
- Interpretation rule: cohen1988

```
smartphone_table |>
  effectsize::cohens_w() |>
  effectsize::interpret(rules = "cohen1988")
```

Cohen's w	95% CI	Interpretation
0.30	[0.18, 1.00]	moderate

- One-sided CIs: upper bound fixed at [1.00].
- Interpretation rule: cohen1988

```
smartphone_table |>
  effectsize::cramers_v() |>
  effectsize::interpret(rules = "cohen1988")
```

Cramer's V (adj.)	95% CI	Interpretation
0.29	[0.15, 1.00]	small

- One-sided CIs: upper bound fixed at [1.00].
- Interpretation rule: cohen1988

Interpretation

We see that there is little difference in the different effect size measures for our introductory lecture example. However, we see even small differences can lead to relatively significant variation in the interpretation of effect strength according to Cohen (1988).

ANOVA - Example One-way ANOVA on effectiveness of three different in-store promotion campaigns

Import data from lecturer's homepage

First, we will download the data from our [Chair's homepage](#) at the [Otto von Guericke University](#).

The dataset is stored as a .rds file within the following directory: https://marketing.ovgu.de/marketing_media/L

The dataset's name is **table_6_1_sales.rds**.

We first set the url (Uniform Resource Locator) for the download.

```
url_of_file <-  
"https://marketing.ovgu.de/marketing_media/Downloads/MMA/table_6_1_sales.rds"
```

We then download the file to our current *R* project.

```
sales_data <- readRDS(url(url_of_file)) |> tibble::as_tibble()
```

If anything does not work out as expected on your machine, you can download the data by clicking on the following button.

Let's have a look at the dataset:

```
print(sales_data)
```

```
# A tibble: 10 x 4  
  service_type point_of_sale_display free_tasting_stand in_store_announcements  
  <chr>          <dbl>          <dbl>          <dbl>  
1 Personal      50            55            45  
2 Personal      52            55            50  
3 Personal      43            49            45  
4 Personal      48            57            46  
5 Personal      47            55            42  
6 Self-service  45            49            43  
7 Self-service  44            48            42  
8 Self-service  49            54            45  
9 Self-service  51            54            47  
10 Self-service 44            44            42
```

Check assumptions

Of course, the sample size is tiny. It would be better to have at least $n=30$ per group.

Test for normality via Shapiro-Wilk

For testing normality, we rely on the `shapiro_test()` function provided by the package `rstatix` (Kassambara, 2026), which conducts the Shapiro-Wilk test (Shapiro & Wilk, 1965).

```
rstatix::shapiro_test(sales_data$free_tasting_stand)
```

```
# A tibble: 1 x 3
  variable                statistic p.value
  <chr>                  <dbl>   <dbl>
1 sales_data$free_tasting_stand    0.879  0.127
```

```
rstatix::shapiro_test(sales_data$point_of_sale_display)
```

```
# A tibble: 1 x 3
  variable                statistic p.value
  <chr>                  <dbl>   <dbl>
1 sales_data$point_of_sale_display    0.936  0.507
```

```
rstatix::shapiro_test(sales_data$in_store_announcements)
```

```
# A tibble: 1 x 3
  variable                statistic p.value
  <chr>                  <dbl>   <dbl>
1 sales_data$in_store_announcements    0.896  0.197
```

Interpretation

As the empirical p-values for all three Shapiro-Wilk-tests are clearly above 5%, we conclude that it is likely enough to observe the group-wise sales data when sampling from a population in which H_0 is true. Thus, we see support for the normality of sales data within each group.

Check for Equality of Variances Assumption via Levene

To compute Levene's test for homogeneity of variance across groups, we rely on the function `leveneTest()` provided by the package `car` (Fox & Weisberg, 2019). One drawback of doing so is that we first have to flip the dataset from a wide into the long (sometimes also called stacked) format. The next code chunk does

this job with the help of the function `pivot_longer()`, provided by the [tidyr](#) package (Wickham, Vaughan, & Girlich, 2025).

Transform to long format

```
sales_data_long <- sales_data |> tidyr::pivot_longer(
  cols = point_of_sale_display:in_store_announcements,
  names_to = "promotion_type",
  values_to = "sales"
)

## arrange factor levels
sales_data_long$promotion_type <-
factor(sales_data_long$promotion_type,
  levels = c("point_of_sale_display", "free_tasting_stand",
    "in_store_announcements")
)

sales_data_long$promotion_type <-
labelled::to_labelled(sales_data_long$promotion_type)

sales_data_long$service_type <-
factor(sales_data_long$service_type,
  levels = c("Personal", "Self-service")
)

sales_data_long$service_type <-
labelled::to_labelled(sales_data_long$service_type)

## sort
sales_data_long <- sales_data_long |>
dplyr::arrange(promotion_type, service_type)

##see data in long format
print(sales_data_long, n = 30)
```

A tibble: 30 x 3

	service_type <dbl+lbl>	promotion_type <dbl+lbl>	sales <dbl>
1	1 [Personal]	1 [point_of_sale_display]	50
2	1 [Personal]	1 [point_of_sale_display]	52
3	1 [Personal]	1 [point_of_sale_display]	43
4	1 [Personal]	1 [point_of_sale_display]	48

5	1	[Personal]	1	[point_of_sale_display]	47
6	2	[Self-service]	1	[point_of_sale_display]	45
7	2	[Self-service]	1	[point_of_sale_display]	44
8	2	[Self-service]	1	[point_of_sale_display]	49
9	2	[Self-service]	1	[point_of_sale_display]	51
10	2	[Self-service]	1	[point_of_sale_display]	44
11	1	[Personal]	2	[free_tasting_stand]	55
12	1	[Personal]	2	[free_tasting_stand]	55
13	1	[Personal]	2	[free_tasting_stand]	49
14	1	[Personal]	2	[free_tasting_stand]	57
15	1	[Personal]	2	[free_tasting_stand]	55
16	2	[Self-service]	2	[free_tasting_stand]	49
17	2	[Self-service]	2	[free_tasting_stand]	48
18	2	[Self-service]	2	[free_tasting_stand]	54
19	2	[Self-service]	2	[free_tasting_stand]	54
20	2	[Self-service]	2	[free_tasting_stand]	44
21	1	[Personal]	3	[in_store_announcements]	45
22	1	[Personal]	3	[in_store_announcements]	50
23	1	[Personal]	3	[in_store_announcements]	45
24	1	[Personal]	3	[in_store_announcements]	46
25	1	[Personal]	3	[in_store_announcements]	42
26	2	[Self-service]	3	[in_store_announcements]	43
27	2	[Self-service]	3	[in_store_announcements]	42
28	2	[Self-service]	3	[in_store_announcements]	45
29	2	[Self-service]	3	[in_store_announcements]	47
30	2	[Self-service]	3	[in_store_announcements]	42

See value labels

```
sales_data_long$promotion_type |> labelled::val_labels()
```

point_of_sale_display	free_tasting_stand	in_store_announcements
1	2	3

```
sales_data_long$service_type |> labelled::val_labels()
```

Personal	Self-service
1	2

Levene's test

```
sales_data_long |> car::leveneTest(sales ~
labelled::to_factor(promotion_type),
                                data = _)
```

```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  2  0.8287 0.4474
      27
```

💡 Interpretation

As the empirical p-value is not significant, we conclude that we have little support for a violation of the assumption of homoscedasticity. Thus, we can proceed with One-way ANOVA

Conduct the One-way ANOVA

i use of `to_factor()`

Note that below, we apply the function `to_factor()` from the [labelled](#) package ([Larmarange, 2025](#)) to the variable `promotion_type`. We must do so because otherwise, the `aov()` function will not recognize that we have a categorical variable. Thus, we have to explicitly tell to handle this variable as a categorical factor variable.

```
one_way_ANOVA_results <- aov(
  formula = sales ~ labelled::to_factor(promotion_type),
  data = sales_data_long
) |> rstatix::anova_summary(detailed = TRUE)

one_way_ANOVA_results
```

	Effect	DFn	DFd	SSn	SSd	F	p	p<.05	ges
1		1	2	273.8	310.2	11.916	0.000195	1	0.469

💡 Interpretation

The results are identical to the lecture example. Particularly, the between-sum-of-squares (numerator) are 273.8, whereas the within-sum-of-squares (denominator) are 310.2. As a result, the F-Ratio is 11.92, with a corresponding p-value of approx. 0. Because the empirical p-value is below 0.05, we conclude that the data is very unlikely to come from a sample with an underlying population in which H_0 holds true. Thus, we find support for H_A ; the three promotion types result in different sales numbers.

Keep in mind that from the global F-test in ANOVA, we aren't able to tell which levels of the factor variable (i.e., which advertising groups) are different from each other. Thus, in the absence of hypotheses, we have to apply post-hoc tests (Sarstedt & Mooi, 2019, Chapter 6). For an application in market research, see Girard et al. (2019).

Post-hoc Test

One option, when the assumptions of normality and homoscedasticity are met, is to use Tukey's procedure (Tukey, 1949).

We will do so next, by using the function `tukeyTest()`, provided by the `PMCMRplus` package (Pohlert, 2024). We present the results next to group-wise descriptive statistics.

```
sales_data_long |>
  dplyr::group_by(promotion_type) |>
  dplyr::summarise(
    mean = mean(sales),
    SD = sd(sales),
    n = n()
  )
```

```
# A tibble: 3 x 4
  promotion_type      mean    SD    n
  <dbl+lbl>      <dbl> <dbl> <int>
1 1 [point_of_sale_display]  47.3  3.20   10
2 2 [free_tasting_stand]    52    4.19   10
3 3 [in_store_announcements] 44.7  2.58   10
```

```
PMCMRplus::tukeyTest(
  formula = sales ~ labelled::to_factor(promotion_type),
  data = sales_data_long
)
```

	point_of_sale_display	free_tasting_stand
free_tasting_stand	0.01207	-
in_store_announcements	0.21797	0.00014

Interpretation

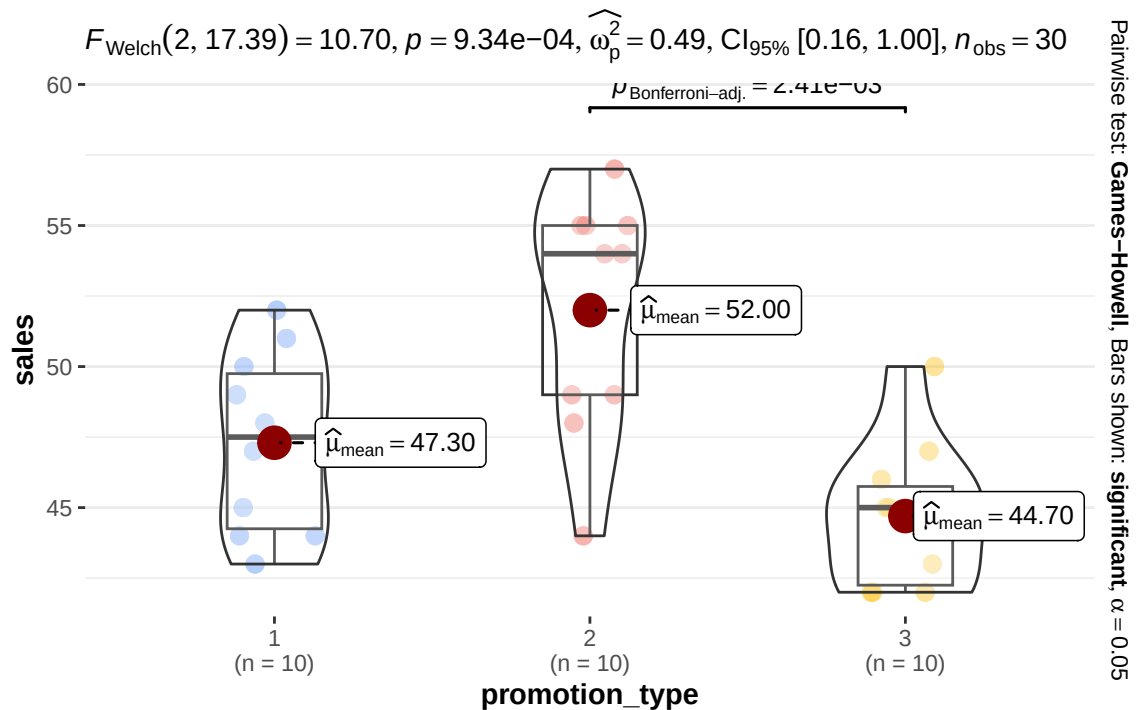
We see that the differences between the advertising types POS-display and Free tasting stand are statistically significant. Likewise, the difference

between the advertising types, Free tasting stand, and in-store announcements are significant. However, the test doesn't support a significant difference between POS-display and in-store announcements. As the Free tasting stand results in the highest sales numbers (significantly higher than the other two types), we would recommend implementing this type of advertising in the future (given that all three types do cost the same).

Doing One-way ANOVA graphically

As an alternative to the test procedures above, we can visualize the ANOVA results with the help of the function *ggbetweenstats()* provided by the package *ggstatsplot* (Patil, 2021).

```
sales_data_long |> ggstatsplot::ggbetweenstats(  
  y = sales,  
  x = promotion_type,  
  var.equal = TRUE,  
  p.adjust.method = "bonferroni",  
  bf.message = FALSE  
)
```



💡 Interpretation

The results mirror those seen before. Note, however, that the post-hoc procedure used here is a different one based on the well-known Bonferroni correction ([Bonferroni, 1936](#)).

ANOVA - Example Two-way ANOVA on effectiveness of three different in-store promotion campaigns and two service types

Uses the same data set as for One-way ANOVA

Let's have a look at the data set:

```
print(sales_data)
```

```
# A tibble: 10 x 4
  service_type point_of_sale_display free_tasting_stand in_store_announcements
  <chr>          <dbl>          <dbl>          <dbl>
1 Personal      50            55            45
2 Personal      52            55            50
3 Personal      43            49            45
4 Personal      48            57            46
5 Personal      47            55            42
6 Self-service  45            49            43
7 Self-service  44            48            42
8 Self-service  49            54            45
9 Self-service  51            54            47
10 Self-service 44            44            42
```

Check assumptions

Of course, the sample size is tiny. It would be better to have at least $n=30$ per group.

Test for normality (Shapiro-Wilk)

For testing normality, we rely on the `shapiro_test()` function provided by the package `rstatix` ([Kassambara, 2026](#)), which conducts the Shapiro-Wilk test ([Shapiro & Wilk, 1965](#)).

```
sales_data_long |>
  dplyr::group_by(service_type, promotion_type) |>
  rstatix::shapiro_test(sales)
```

```
# A tibble: 6 x 5
  service_type promotion_type variable statistic      p
  <dbl+lbl>    <dbl+lbl>    <chr>      <dbl> <dbl>
1 1 [Personal] 1 [point_of_sale_display] sales      0.978 0.921
2 1 [Personal] 2 [free_tasting_stand]    sales      0.768 0.0435
3 1 [Personal] 3 [in_store_announcements] sales      0.931 0.601
4 2 [Self-service] 1 [point_of_sale_display] sales      0.833 0.147
5 2 [Self-service] 2 [free_tasting_stand]    sales      0.900 0.410
6 2 [Self-service] 3 [in_store_announcements] sales      0.871 0.272
```

💡 Interpretation

The p-value for the cell personal service x Free tasting stand is lower than 5%. Thus, the assumption of normality is not met in our example. Because of the small sample size, we usually would prefer a non-parametric alternative to Two-way ANOVA such as Bootstrapped ANOVA or Aligned Rank Transform ANOVA via the [ARTool](#) package (Kay, Elkin, Higgins, & Wobbrock, 2025). However, for this introductory example, we will ignore this.

Check for Equality of Variances Assumption (Levene)

Levene's test

```
sales_data_long |>
  car::leveneTest(
    data = _,
    sales ~ labelled::to_factor(service_type) *
    labelled::to_factor(promotion_type)
  )
```

```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  5   0.383 0.8554
      24
```

💡 Interpretation

As the empirical p-value is not significant, we conclude that we have little support for a violation of the assumption of homoscedasticity.

Conduct the Two-way ANOVA

First, with the `aov()` function (stats package), then `anova_test()` provided by [rstatix](#) (Kassambara, 2026), which preserves factor names in output.

```
two_way_ANOVA_results <- aov(
  formula = sales ~ labelled::to_factor(service_type) *
  labelled::to_factor(promotion_type),
  data = sales_data_long
) |>
```

```
rstatix::anova_summary(detailed = TRUE)
```

```
two_way_ANOVA_results
```

	Effect	DFn	DFd	SSn	SSd	F	p	p<.05	ges
1	2	1	24	48.133	248.8	4.643	0.04100	2	0.162
2	1	2	24	273.800	248.8	13.206	0.00014	2	0.524
3	3	2	24	13.267	248.8	0.640	0.53600	1	0.051

```
two_way_ANOVA_results <- sales_data_long |>
  rstatix::anova_test(sales ~ labelled::to_factor(service_type) *
    labelled::to_factor(promotion_type),
    type = 3,
    detailed = TRUE
  )
```

```
two_way_ANOVA_results
```

ANOVA Table (type III tests)

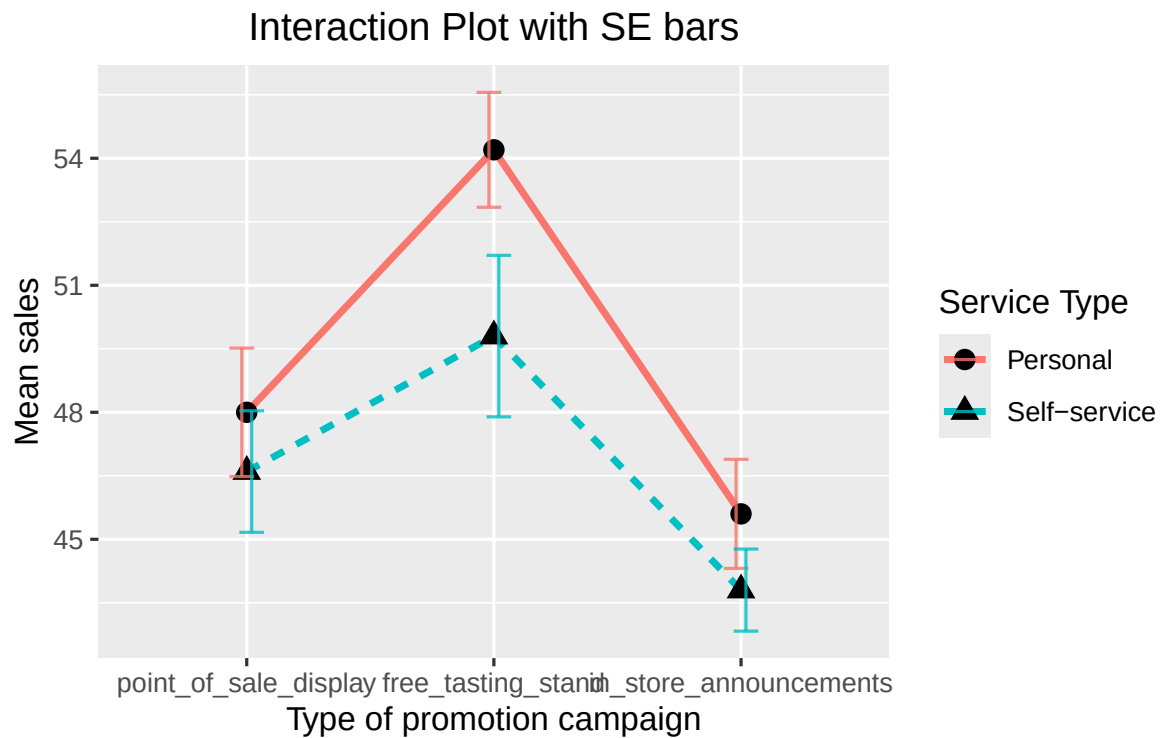
									Effect
1									(Intercept)
2									labelled::to_factor(service_type)
3									labelled::to_factor(promotion_type)
4	labelled::to_factor(service_type):labelled::to_factor(promotion_type)								
	SSn	SSd	DFn	DFd	F	p	p<.05	ges	
1	69120.000	248.8	1	24	6667.524	7.32e-31	*	0.996	
2	48.133	248.8	1	24	4.643	4.10e-02	*	0.162	
3	273.800	248.8	2	24	13.206	1.36e-04	*	0.524	
4	13.267	248.8	2	24	0.640	5.36e-01		0.051	

💡 Interpretation

The results are identical to the lecture example. In particular, the F-ratio is significant for the type of advertising factor (factor 1) and for the service factor (factor 2). In contrast, the interaction between both factors is not significant based on the p values, which clearly exceed 5%. Thus, the observed data does not support the claim that the influence that different advertising types have on sales depends on the service type provided (and vice versa).

To complete the picture , we can visualize the means of each condition in an interaction plot (although, we have found out that the interaction in our example is not significant).

```
sales_data_long |>
  ggplot2::ggplot(aes(
    x = labelled::to_factor(promotion_type),
    color = labelled::to_factor(service_type),
    group = labelled::to_factor(service_type),
    y = sales,
    linetype = labelled::to_factor(service_type),
    shape = labelled::to_factor(service_type)
  )) +
  ggplot2::stat_summary(fun = mean, geom = "path", linewidth =
1.1) +
  ggplot2::stat_summary(fun = mean, geom = "point", size = 3,
color = "black") +
  ggplot2::stat_summary(
    fun.data = "mean_se",
    geom = "errorbar",
    linetype = "solid",
    alpha = 0.8,
    width = 0.2,
    position = position_dodge(0.08)
  )+
  ggplot2::ggtitle("Interaction Plot with SE bars") +
  ggplot2::labs(
    x = "Type of promotion campaign",
    y = "Mean sales",
    color = "Service Type",
    shape = "Service Type",
    linetype = "Service Type"
  ) +
  ggplot2::theme(plot.title = element_text(hjust = 0.5))
```



Post-hoc Tests in two-way ANOVA

In case of Two-way ANOVA, we first need to establish a new variable indicating the cell membership for each case (i.e., 1, 2, ..., or 6).

```
sales_data_long <- sales_data_long |>
  dplyr::mutate(
    cell_membership = dplyr::case_when(
      service_type == 1 & promotion_type == 1 ~ 1,
      service_type == 1 & promotion_type == 2 ~ 2,
      service_type == 1 & promotion_type == 3 ~ 3,
      service_type == 2 & promotion_type == 1 ~ 4,
      service_type == 2 & promotion_type == 2 ~ 5,
      service_type == 2 & promotion_type == 3 ~ 6,
      TRUE ~ NA
    )
  )

labelled::val_labels(sales_data_long$cell_membership) <- c(
  "Personal | point_of_sale_display" = 1,
```

```

"Personal | free_tasting_stand" = 2,
"Personal | in_store_announcements" = 3,
"Self-service | point_of_sale_display" = 4,
"Self-service | free_tasting_stand" = 5,
"Self-service | in_store_announcements" = 6
)

sales_data_long$cell_membership |>
  labelled::val_labels()

```

Personal point_of_sale_display	Personal free_tasting_stand
1	2
Personal in_store_announcements	Self-service point_of_sale_display
3	4
Self-service free_tasting_stand	Self-service in_store_announcements
5	6

One option, when the assumptions of normality and homoscedasticity are met, is to use Tukey's procedure ([Tukey, 1949](#)).

We present the results next to group-wise descriptive statistics.

```

sales_data_long |>
  dplyr::group_by(service_type, promotion_type) |>
  dplyr::summarise(
    mean = mean(sales),
    SD = sd(sales),
    n = n()
  )

```

```

# A tibble: 6 x 5
# Groups:   service_type [2]
  service_type    promotion_type      mean    SD      n
  <dbl+lbl>      <dbl+lbl>    <dbl> <dbl> <int>
1 1 [Personal]    1 [point_of_sale_display]  48    3.39    5
2 1 [Personal]    2 [free_tasting_stand]    54.2  3.03    5
3 1 [Personal]    3 [in_store_announcements]  45.6  2.88    5
4 2 [Self-service] 1 [point_of_sale_display]  46.6  3.21    5
5 2 [Self-service] 2 [free_tasting_stand]    49.8  4.27    5
6 2 [Self-service] 3 [in_store_announcements]  43.8  2.17    5

```

```

post_hoc_results <- PMCMRplus::tukeyTest(
  formula = sales ~ labelled::to_factor(cell_membership),
  data = sales_data_long
)

```

```
)

post_hoc_results |> summary()
```

	q value
Personal free_tasting_stand - Personal point_of_sale_display == 0	4.306
Personal in_store_announcements - Personal point_of_sale_display == 0	-1.667
Self-service point_of_sale_display - Personal point_of_sale_display == 0	-0.972
Self-service free_tasting_stand - Personal point_of_sale_display == 0	1.250
Self-service in_store_announcements - Personal point_of_sale_display == 0	-2.917
Personal in_store_announcements - Personal free_tasting_stand == 0	-5.973
Self-service point_of_sale_display - Personal free_tasting_stand == 0	-5.278
Self-service free_tasting_stand - Personal free_tasting_stand == 0	-3.056
Self-service in_store_announcements - Personal free_tasting_stand == 0	-7.223
Self-service point_of_sale_display - Personal in_store_announcements == 0	0.694
Self-service free_tasting_stand - Personal in_store_announcements == 0	2.917
Self-service in_store_announcements - Personal in_store_announcements == 0	-1.250
Self-service free_tasting_stand - Self-service point_of_sale_display == 0	2.222
Self-service in_store_announcements - Self-service point_of_sale_display == 0	-1.945
Self-service in_store_announcements - Self-service free_tasting_stand == 0	-4.167
Pr(> q)	
Personal free_tasting_stand - Personal point_of_sale_display == 0	0.05532680
Personal in_store_announcements - Personal point_of_sale_display == 0	0.84251870
Self-service point_of_sale_display - Personal point_of_sale_display == 0	0.98165855
Self-service free_tasting_stand - Personal point_of_sale_display == 0	0.94674723
Self-service in_store_announcements - Personal point_of_sale_display == 0	0.33899903
Personal in_store_announcements - Personal free_tasting_stand == 0	0.00360819
Self-service point_of_sale_display - Personal free_tasting_stand == 0	0.01172692
Self-service free_tasting_stand - Personal free_tasting_stand == 0	0.29180266
Self-service in_store_announcements - Personal free_tasting_stand == 0	0.00040876
Self-service point_of_sale_display - Personal in_store_announcements == 0	0.99604508
Self-service free_tasting_stand - Personal in_store_announcements == 0	0.33899903
Self-service in_store_announcements - Personal in_store_announcements == 0	0.94674723
Self-service free_tasting_stand - Self-service point_of_sale_display == 0	0.62393032
Self-service in_store_announcements - Self-service point_of_sale_display == 0	0.74090587
Self-service in_store_announcements - Self-service free_tasting_stand == 0	0.06806693
.	
Personal free_tasting_stand - Personal point_of_sale_display == 0	.
Personal in_store_announcements - Personal point_of_sale_display == 0	.
Self-service point_of_sale_display - Personal point_of_sale_display == 0	.
Self-service free_tasting_stand - Personal point_of_sale_display == 0	.
Self-service in_store_announcements - Personal point_of_sale_display == 0	.
Personal in_store_announcements - Personal free_tasting_stand == 0	**
Self-service point_of_sale_display - Personal free_tasting_stand == 0	*
Self-service free_tasting_stand - Personal free_tasting_stand == 0	.

```

Self-service | in_store_announcements - Personal | free_tasting_stand == 0      ***
Self-service | point_of_sale_display - Personal | in_store_announcements == 0
Self-service | free_tasting_stand - Personal | in_store_announcements == 0
Self-service | in_store_announcements - Personal | in_store_announcements == 0
Self-service | free_tasting_stand - Self-service | point_of_sale_display == 0
Self-service | in_store_announcements - Self-service | point_of_sale_display == 0
Self-service | in_store_announcements - Self-service | free_tasting_stand == 0 .

```

```

post_hoc_results$p.value |> knitr::kable(digits = 3,
                                          caption = "Pair-wise adjusted
                                          p-values (undirected)")

```

Table 3: Pair-wise adjusted p-values (undirected)

	Personal point_of_sale_display	Personal free_tasting_stand	Personal in_store_announcements	Self-service point_of_sale_display	Self-service free_tasting_stand
Personal free_tasting_stand	0.055	NA	NA	NA	NA
Personal in_store_announcements	0.843	0.004	NA	NA	NA
Self-service point_of_sale_display	0.982	0.012	0.996	NA	NA
Self-service free_tasting_stand	0.947	0.292	0.339	0.624	NA
Self-service in_store_announcements	0.339	0.000	0.947	0.741	0.068

💡 Interpretation

From the table above, we can see which pair-wise group comparisons are significant (p-value below 5%) after applying a correction for multiple tests.

From the Shapiro-Wilk test, however, we have seen that the assumption of normality is violated, and we have a small sample size that isn't able to compensate for the lack of normality. Usually, we would, therefore, use a post-hoc procedure that is robust against violations of normality, such as the Dwass-Steel-Critchlow-Fligner procedure ([Douglas & Michael, 1991](#)). We can always use this procedure, but its downside is that it is conservative (i.e., it is more unlikely to find significant differences across groups).

Next, we apply this variant, which is also provided by the [PMCMRplus](#) package (Pohlert, 2024). The function is named `dscfAllPairsTest()`.

```
post_hoc_results <- PMCMRplus::dscfAllPairsTest(
  formula = sales ~ labelled::to_factor(cell_membership),
  data = sales_data_long
)

post_hoc_results |> summary()
```

	q value
Personal free_tasting_stand - Personal point_of_sale_display == 0	-3.140
Personal in_store_announcements - Personal point_of_sale_display == 0	-1.783
Self-service point_of_sale_display - Personal point_of_sale_display == 0	-0.741
Self-service free_tasting_stand - Personal point_of_sale_display == 0	-1.189
Self-service in_store_announcements - Personal point_of_sale_display == 0	-2.832
Personal in_store_announcements - Personal free_tasting_stand == 0	-3.450
Self-service point_of_sale_display - Personal free_tasting_stand == 0	-3.310
Self-service free_tasting_stand - Personal free_tasting_stand == 0	-3.009
Self-service in_store_announcements - Personal free_tasting_stand == 0	-3.750
Self-service point_of_sale_display - Personal in_store_announcements == 0	-0.150
Self-service free_tasting_stand - Personal in_store_announcements == 0	-1.932
Self-service in_store_announcements - Personal in_store_announcements == 0	-1.363
Self-service free_tasting_stand - Self-service point_of_sale_display == 0	-1.505
Self-service in_store_announcements - Self-service point_of_sale_display == 0	-2.087
Self-service in_store_announcements - Self-service free_tasting_stand == 0	-3.121
	Pr(> q)
Personal free_tasting_stand - Personal point_of_sale_display == 0	0.228177
Personal in_store_announcements - Personal point_of_sale_display == 0	0.806241
Self-service point_of_sale_display - Personal point_of_sale_display == 0	0.995252
Self-service free_tasting_stand - Personal point_of_sale_display == 0	0.959911
Self-service in_store_announcements - Personal point_of_sale_display == 0	0.340583
Personal in_store_announcements - Personal free_tasting_stand == 0	0.142739
Self-service point_of_sale_display - Personal free_tasting_stand == 0	0.177777
Self-service free_tasting_stand - Personal free_tasting_stand == 0	0.272703
Self-service in_store_announcements - Personal free_tasting_stand == 0	0.085254
Self-service point_of_sale_display - Personal in_store_announcements == 0	0.999998
Self-service free_tasting_stand - Personal in_store_announcements == 0	0.747418
Self-service in_store_announcements - Personal in_store_announcements == 0	0.929369
Self-service free_tasting_stand - Self-service point_of_sale_display == 0	0.895657
Self-service in_store_announcements - Self-service point_of_sale_display == 0	0.679885
Self-service in_store_announcements - Self-service free_tasting_stand == 0	0.234442
Personal free_tasting_stand - Personal point_of_sale_display == 0	
Personal in_store_announcements - Personal point_of_sale_display == 0	

```

Self-service | point_of_sale_display - Personal | point_of_sale_display == 0
Self-service | free_tasting_stand - Personal | point_of_sale_display == 0
Self-service | in_store_announcements - Personal | point_of_sale_display == 0
Personal | in_store_announcements - Personal | free_tasting_stand == 0
Self-service | point_of_sale_display - Personal | free_tasting_stand == 0
Self-service | free_tasting_stand - Personal | free_tasting_stand == 0
Self-service | in_store_announcements - Personal | free_tasting_stand == 0
Self-service | point_of_sale_display - Personal | in_store_announcements == 0
Self-service | free_tasting_stand - Personal | in_store_announcements == 0
Self-service | in_store_announcements - Personal | in_store_announcements == 0
Self-service | free_tasting_stand - Self-service | point_of_sale_display == 0
Self-service | in_store_announcements - Self-service | point_of_sale_display == 0
Self-service | in_store_announcements - Self-service | free_tasting_stand == 0

```

```

post_hoc_results$p.value |> knitr::kable(digits = 3,
                                         caption = "Pair-wise adjusted
                                         p-values (undirected)")

```

Table 4: Pair-wise adjusted p-values (undirected)

	Personal point_of_sale_display	Personal free_tasting_stand	Personal in_store_announcements	Self-service point_of_sale_display	Self-service free_tasting_stand
Personal free_tasting_stand	0.228	NA	NA	NA	NA
Personal in_store_announcements	0.806	0.143	NA	NA	NA
Self-service point_of_sale_display	0.995	0.178	1.000	NA	NA
Self-service free_tasting_stand	0.960	0.273	0.747	0.896	NA
Self-service in_store_announcements	0.341	0.085	0.929	0.680	0.234

Effect sizes in ANOVA

As learned during the lecture, we can generally recommend using the global and partial versions of epsilon² (ϵ^2) (Iacobucci, Popovich, Moon, & Román, 2023). In a situation where the total sample size n is at a minimum of 200, and we have only a few levels in the largest factor, we could rely on eta² (η^2). in market

research, we are, most of the time, rather interested in the explanatory power of the factors (the partial effect sizes). Let us do both (partial ϵ^2 and η^2). We, again, draw on the use of the `effectsize` package (Ben-Shachar et al., 2020).

```
two_way_ANOVA_results <- aov(
  formula = sales ~ labelled::to_factor(service_type) *
    labelled::to_factor(promotion_type),
  data = sales_data_long
)

two_way_ANOVA_results |> effectsize::eta_squared() |>
knitr::kable(digits = 3)
```

Parameter	Eta2_partial	CI_low	CI_high	
labelled::to_factor(service_type)	0.162	0.95	0.004	1
labelled::to_factor(promotion_type)	0.524	0.95	0.261	1
labelled::to_factor(service_type):labelled::to_factor(promotion_type)	0.051	0.95	0.000	1

```
two_way_ANOVA_results |> effectsize::epsilon_squared() |>
knitr::kable(digits = 3)
```

Parameter	Epsilon2_partial	CI_low	CI_high	
labelled::to_factor(service_type)	0.127	0.95	0.000	1
labelled::to_factor(promotion_type)	0.484	0.95	0.214	1
labelled::to_factor(service_type):labelled::to_factor(promotion_type)	0.000	0.95	0.000	1

💡 Interpretation

From the tables above, we see that the values for partial ϵ^2 and η^2 perfectly mirror those we calculated during the lecture. We conclude that the influence of the promotion type on sales far exceeds the influence of the service type.

OLS Regression

In the following section, we reproduce the regression example from the lecture within *R*. The example is modified from Table 7 of Chapter 7 in Sarstedt & Mooi (2019).

30 randomly sampled supermarkets of comparable size that sell “Organic Valley Milk” milk reported weekly sales of Organic Whole Milk, 0.25 Gallon Cartons, its price, and an index of promotional activities.

Question: Does the sample’s data support the idea that variance in price and promotion activities is related to different sales figures?

In total, we will go through the seven steps that we have discussed during the lecture:

1. Checking the regression analysis data requirements
2. Specifying and estimating the regression model
3. Testing the regression analysis assumptions
4. Interpreting the regression results and Goodness of Fit (GoF)
5. Validating the regression model
6. Using the regression model for predictions

However, we will only focus on the things that incorporate applications in *R*. Therefore, we will not repeat rules-of-thumb (for example, regarding sample sizes, etc.).

First, we will download the data from our [Chair’s homepage](#) at the [Otto von Guericke University](#).

Import the regression data

The dataset is stored as a .rds file within the following directory: https://marketing.ovgu.de/marketing_media/Downloads/MMA/sales_data_regression.rds

The dataset’s name is **sales_data_regression.rds**.

We first set the url (Uniform Resource Locator) for the download.

```
url_of_file <-  
"https://marketing.ovgu.de/marketing_media/Downloads/MMA/sales_data_regression.rds"
```

Then, we download the file to our current *R* project.

```
sales_data_regression <- readRDS(url(url_of_file)) |>  
tibble::as_tibble()
```

If anything does not work out as expected on your machine, you can download the data by clicking on the following button.

Let's have a look at the dataset:

```
print(sales_data_regression, n=30)
```

```
# A tibble: 30 x 4
  supermarket promotion own_price sales
      <dbl>      <dbl>    <dbl> <dbl>
1           1      12.0       1.1  3454
2           2      22.0       1.08  3966
3           3      22.0       1.08  2952
4           4      22.0       1.08  3576
5           5      21.4       1.08  3692
6           6      22.0       1.08  3226
7           7      47.0       1.09  3776
8           8     117.       1.05 14134
9           9      92.0       1.1   5114
10          10      62.0       1.08  4022
11          11      87.0       1.12  4492
12          12      92.0       1.02 10186
13          13      61.4       1.08  7010
14          14      72.0       1.06  4162
15          15      47.0       1.13  3446
16          16      37.0       1.05  3690
17          17      12.0       1.1   3742
18          18      62.0       1.08  7512
19          19      62.0       1.08  9476
20          20      22.0       1.08  3178
21          21       2.04       1.12  2920
22          22      42.0       1.04  8212
23          23      17.0       1.09  3272
24          24       7.04       1.11  2808
25          25       2.04       1.12  2648
26          26       7.04       1.11  3786
27          27       2.04       1.12  2908
28          28      62.0       1.08  3395
29          29      82.0       1.04  4106
30          30     132.       1.02  8754
```

Step 1: Checking the regression analysis data requirements

Sample size (detectable R^2)

In addition to the things we have already discussed during the lecture, we could use a Power Analysis to evaluate which size of effect (in terms of overall R^2)

value) we can detect at $\alpha=5\%$ with a certain level of statistical power $(1-\beta)$, based on our $n=30$ sample size.

In essence, we are looking for the statistical power to detect an R^2 of a certain level as statistically significant in the global F-test of the regression. Think back to the general form of multiple regression: $Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \beta_n X_n + e$, $e \sim N(0, \sigma^2)$

Recall: the global F-test gets significant when it is improbable to find at least one of the sample's β_n s $\neq 0$ when, in reality, they are all 0 (if H_0 holds true in the underlying population). Further recall that in the lecture, we have used the following rules of thumb (Cohen, 1988, p. 79):

Table 5: Cohen's rules-of-thumb for interpreting R^2

	Interpretation
$R^2 < 0.10$	very weak model fit
$0.10 \leq R^2 < 0.30$	weak model fit
$0.30 \leq R^2 < 0.50$	moderate model fit
$R^2 \geq 0.50$	substantial fit

Thus, we could ask what chance we have to end with a significant F-test, when the underlying population's true model fit is weak (0.10) or moderate (0.30).

We use the `pwrss.f.reg()` function from the `pwrss` package (Bulus & Jentschke, 2026).

```
pwrss::pwrss.f.reg(n = 30, k = 2, alpha = 0.05, r2 = 0.10)
```

```
+-----+
|                POWER CALCULATION                |
+-----+
```

Linear Regression (F-Test)

```
-----
Hypotheses
-----
```

```

H0 (Null)           : R-squared = 0
H1 (Alternative)    : R-squared > 0

```

```
-----
Results
-----
```

```
Effect Size (R-squared) = 0.100
```

```

Sample Size           = 30
Type 1 Error (alpha)  = 0.050
Type 2 Error (beta)   = 0.680
Statistical Power     = 0.320  <<

```

```
pwrss::pwrss.f.reg(n = 30, k = 2, alpha = 0.05, r2 = 0.30)
```

```

+-----+
|               POWER CALCULATION               |
+-----+

```

Linear Regression (F-Test)

Hypotheses

```

H0 (Null)           : R-squared = 0
H1 (Alternative)    : R-squared > 0

```

Results

```

Effect Size (R-squared) = 0.300
Sample Size             = 30
Type 1 Error (alpha)    = 0.050
Type 2 Error (beta)     = 0.132
Statistical Power       = 0.868  <<

```

Interpretation

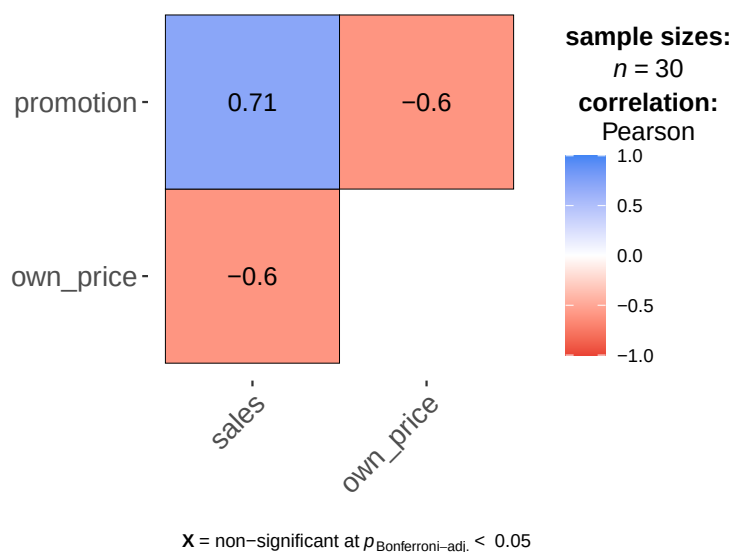
From the outputs above, we see that the corresponding values for the Power are 0.32 and 0.868. Therefore, we conclude that we have a fair chance of finding a significant F-test when the underlying R^2 represents a moderate level. However, we do not have a fair chance when the underlying true model fit is only weak.

Multicollinearity among independent variables

To assess whether our independent variables (IVs) are strongly correlated with each other, we will create a correlation matrix. Recall that in linear regression, we want each IV to be highly correlated with the dependent variable (DV) but not with other IVs.

We can visualize the correlation matrix with the help of the function `ggcorrmat()` provided by the package `ggstatsplot` (Patil, 2021).

```
sales_data_regression |>
  dplyr::select(sales, own_price, promotion) |>
  ggstatsplot::ggcorrmat(p.adjust.method = "bonferroni")
```



💡 Interpretation

From the outputs above, we see that the corresponding predictor variables **promotion** and **own_price** are substantially (and statistically significantly) correlated with each other. This indicates the presence of collinearity.

To assess whether the extend of collinearity among predictors is a problem, we further call for the Variance-inflation-factors (VIF). The package `car` (Fox & Weisberg, 2019) provides the variance inflation factor as the function `vif()` for this purpose. Below, `lm()` is a generic function that helps us to estimate linear regression models.

```
lm(formula = sales ~ own_price + promotion, data =
  sales_data_regression) |> car::vif()
```

```
own_price promotion
1.566884 1.566884
```

💡 Interpretation

We see that VIF values for both predictors are around 1.567 (note that they will always be the same size for models with only two independent variables). As this value is far below the threshold of 5, we conclude that the extend of multicollinearity is acceptable.

Step 2: Specifying and estimating the regression model

With regard to Step 2, we have discussed during lecture how to estimate β_n s mathematically through application of basic matrix manipulations.

For sake of completeness, we can subsequently let R do this job for us. Recall that:

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

Thus, we need to extract several things from our data set *sales_data_regression*. First, we create the design matrix $\mathbf{X}$, which contains the independent variables as columns. Keep in mind, that if we are willing to estimate the intercept α , we also need to include a first column only comprising 1s.

```
X <- cbind(1, sales_data_regression$own_price,  
sales_data_regression$promotion)
```

X

	[,1]	[,2]	[,3]
[1,]	1	1.10	12.04
[2,]	1	1.08	22.04
[3,]	1	1.08	22.04
[4,]	1	1.08	22.04
[5,]	1	1.08	21.42
[6,]	1	1.08	22.04
[7,]	1	1.09	47.04
[8,]	1	1.05	117.04
[9,]	1	1.10	92.04
[10,]	1	1.08	62.04
[11,]	1	1.12	87.04
[12,]	1	1.02	92.04
[13,]	1	1.08	61.42
[14,]	1	1.06	72.04
[15,]	1	1.13	47.04

```
[16,] 1 1.05 37.04
[17,] 1 1.10 12.04
[18,] 1 1.08 62.04
[19,] 1 1.08 62.04
[20,] 1 1.08 22.04
[21,] 1 1.12 2.04
[22,] 1 1.04 42.04
[23,] 1 1.09 17.04
[24,] 1 1.11 7.04
[25,] 1 1.12 2.04
[26,] 1 1.11 7.04
[27,] 1 1.12 2.04
[28,] 1 1.08 62.04
[29,] 1 1.04 82.04
[30,] 1 1.02 132.04
```

Finding the transposed form of $\mathbf{X}$ (X^T) is simple:

```
t(X)
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
[1,] 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
[2,] 1.10 1.08 1.08 1.08 1.08 1.08 1.09 1.05 1.10 1.08 1.12 1.02
[3,] 12.04 22.04 22.04 22.04 21.42 22.04 47.04 117.04 92.04 62.04 87.04 92.04
      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
[1,] 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
[2,] 1.08 1.06 1.13 1.05 1.10 1.08 1.08 1.08 1.12 1.04 1.09 1.11
[3,] 61.42 72.04 47.04 37.04 12.04 62.04 62.04 22.04 2.04 42.04 17.04 7.04
      [,25] [,26] [,27] [,28] [,29] [,30]
[1,] 1.00 1.00 1.00 1.00 1.00 1.00
[2,] 1.12 1.11 1.12 1.08 1.04 1.02
[3,] 2.04 7.04 2.04 62.04 82.04 132.04
```

We also have to extract the vector $\mathbf{y}$ containing the values for the dependent variable:

```
y <- cbind(sales_data_regression$sales)
```

Now, we are ready to multiply the elements. In *R*, multiplying a matrix $\mathbf{B}$ from the right side to a matrix $\mathbf{A}$ is expressed through the operator ‘`%%`’ (`‘A%%B’`). The function `solve()` is helpful in finding the inverse of matrix (i.e., to solve the equation system).

To find $(X^T X)^{-1}$, we apply:

```
solve(t(X) %*% X)
```

```

      [,1]      [,2]      [,3]
[1,] 77.96711074 -70.52149974 -3.569011e-02
[2,] -70.52149974 63.85887945 3.122106e-02
[3,] -0.03569011 0.03122106 4.219068e-05

```

Similarly, to find the vector projection $X^T y$, we apply:

```
t(X) %*% y
```

```

      [,1]
[1,] 147615.0
[2,] 158382.6
[3,] 8669899.4

```

That said, we find the estimated regression coefficients by:

```
solve(t(X) %*% X) %*% t(X) %*% y
```

```

      [,1]
[1,] 30304.05391
[2,] -25209.85801
[3,] 42.26605

```

i Note

Note that the first element is the intercept α , whereas the second and the third element represent the regression coefficients β_1 and β_2 for the independent variables *own_price* and *promotion*. Also note that the values perfectly mirror those that we have calculated during lecture manually.

We double-check via the function `lm()`, which helps us to estimate linear models.

```
lm(formula = sales ~ own_price + promotion,
   data=sales_data_regression)
```

Call:

```
lm(formula = sales ~ own_price + promotion, data = sales_data_regression)
```

Coefficients:

```

(Intercept)   own_price   promotion
  30304.05    -25209.86      42.27

```

Step 3: Testing the regression analysis assumptions

Gauss-Markov assumptions

We have to work on a list of assumptions:

1. linearity
2. bias-free errors
3. homoscedasticity
4. absence of autocorrelation
5. normally distributed errors (optional)

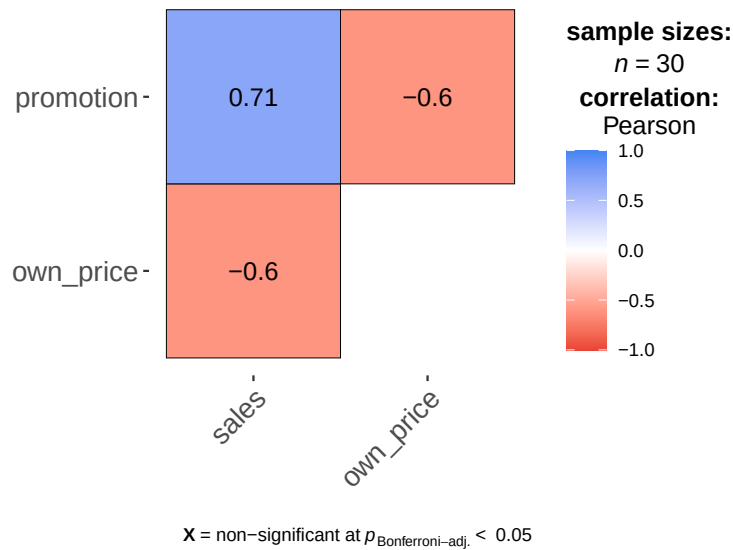
Let us start with **linearity**

Linearity in parameters

💡 Linearity is given

Note that this assumption is met, since we have formulated the regression equation in a linear way: $Y = \alpha + \beta_1 * price + \beta_2 * promotion + e$

However, often, researchers are using this step to additionally evaluate whether it makes sense to transform the independent (IV) or the dependent (DV) variables. This is useful when the relationship between the DV and an IV is not well approximated by linear one. To get further insight into this issue, we can build a scatter plot of each IV against the DV. Or, we ask for the correlation matrix of the variables involved (if the Pearson correlations are high, a linear relationship seems plausible). Above, we have still created such a correlation matrix.



💡 Interpretation

It seems that a linear relationship between both IVs and the DV fits well, as both absolute correlations are substantial and significantly different from 0.

In addition, we can apply **Ramsey's RESET test** ([Cook & Weisberg, 1983](#); [Ramsey, 1969](#)) to evaluate whether the relationship between the IVs and the DV is linear.

Let us first save the calibrated regression model:

```
regression_model <- lm(formula = sales ~ own_price + promotion,
data = sales_data_regression)
```

The package `lmtest` ([Zeileis & Hothorn, 2002](#)) provides Ramsey's RESET test in the form of the function `resettest()`.

```
regression_model |> lmtest::resettest()
```

RESET test

```
data: regression_model
RESET = 1.0532, df1 = 2, df2 = 25, p-value = 0.3638
```

💡 Interpretation

As the correlations are substantial and significant, whereas **Ramsey's RESET** test is not, it is plausible to assume that linear relationship between both IVs and the DV fits well.

Bias-free errors

In the absence of systematical errors (e), their expected value should be 0. We can only observe the empirical residuals for each observation in our regression model. Usually, one tests whether the observed errors lead to a significant difference from 0.

We can assess the observation-wise errors through:

```
regression_model$residuals
```

```
      1      2      3      4      5      6
371.90670 -42.95093 -1056.95093 -432.95093 -290.74598 -782.95093
      7      8      9     10     11     12
-1037.50351 5353.47890 -1349.37703 -1677.59279 -1255.84964 1705.83433
     13     14     15     16     17     18
1336.61216 -2464.45042 -359.10919 -1709.23737  659.90670 1812.40721
     19     20     21     22     23     24
3776.40721 -830.95093  764.76433 2349.33382 -273.52211 189.33551
     25     26     27     28     29     30
 492.76433 1167.33551  752.76433 -2304.59279 -3447.30804 -1416.80754
attr(,"label")
[1] "Weekly sales in USD"
```

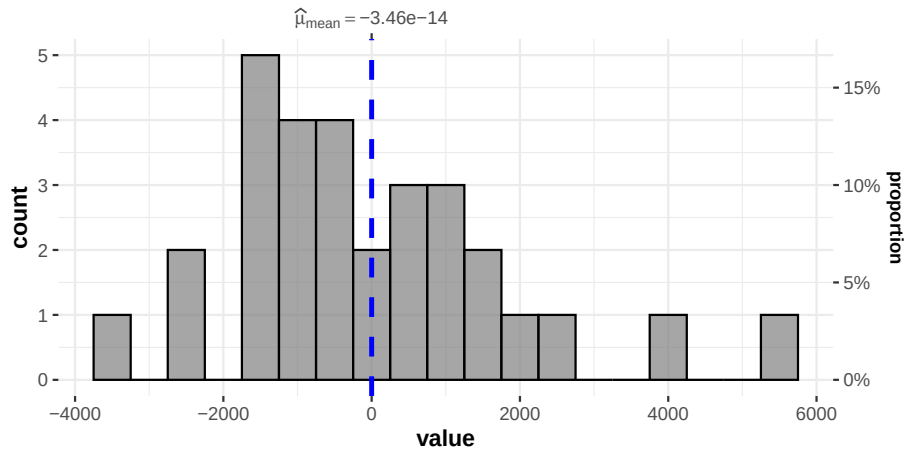
Next, we use the function `gghistostats()` (also from [ggstatsplot](#) (Patil, 2021)) to evaluate the residuals graphically:

```
regression_model$residuals |> tibble::as_tibble() |>
ggstatsplot::gghistostats(
  data = _,
  title = "Histogram of model residuals together with normal
curve \nand formal test of mean against the value of 0",
  x = value,
  type = "parametric",
  test.value = 0,
  bf.message = FALSE,
  k = 3L, #number of digits
```

```
binwidth = 500,  
subtitle = FALSE  
)
```

Histogram of model residuals together with normal curve and formal test of mean against the value of 0

$t_{\text{Student}}(29) = -1.03\text{e-}16$, $p = 1.00$, $\hat{g}_{\text{Hedges}} = -1.83\text{e-}17$, $\text{CI}_{95\%} [-0.35, 0.35]$, $n_{\text{obs}} = 30$

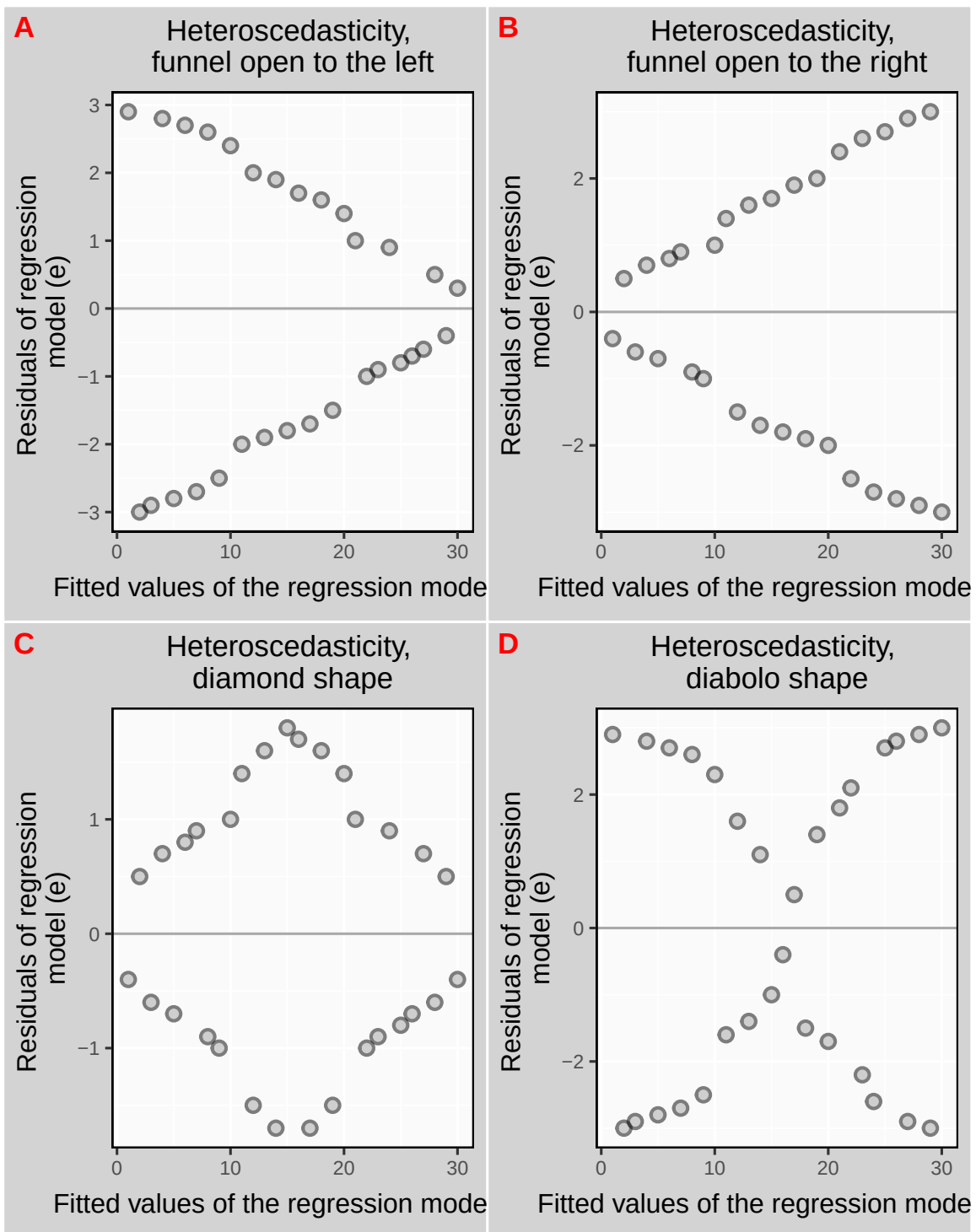


Interpretation

We see that the mean of the residuals almost perfectly approaches the value of 0. Likewise, a t-test against the constant 0 indicates that it is very likely that the observed mean comes from a underlying population where the true mean is also 0 (the empirical p-value approaches 1). Thus, we do not have problems with biased errors.

Homoscedasticity

Recall, homoscedasticity describes a constant error variance across the levels of the DV. A violation of the principle is called **heteroscedasticity**. Usually, market researchers first plot the residuals of the regression model e_i against the fitted values of the dependent variable $\hat{y}_i$, and then are searching for eye-catching patterns such as funnels, diabolos, or diamonds (see below).

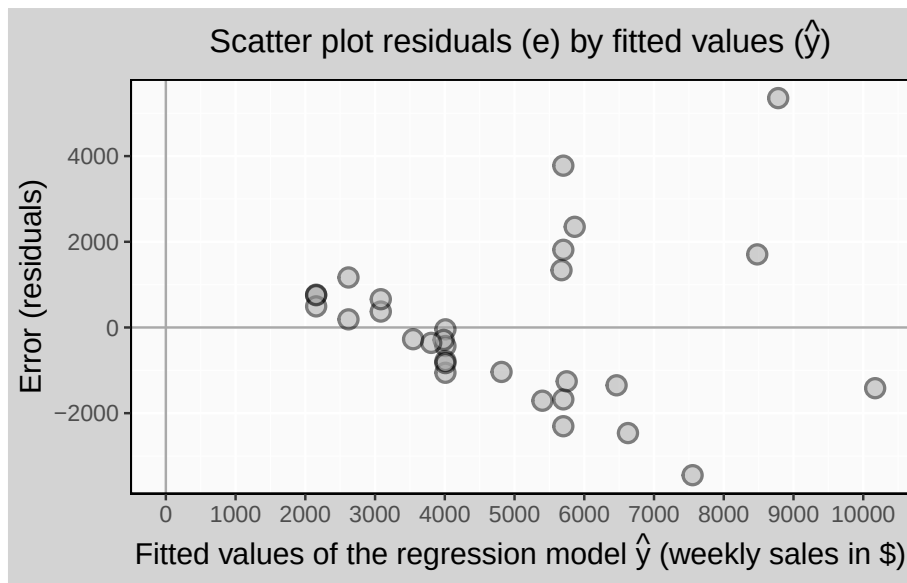


Next, we do so for our example. Below, we first create a new tibble housing the residuals and the fitted values of the regression model. Afterward, we create a scatterplot using *ggplot()* provided by the package *ggplot2* (Wickham, 2016).

```
fitted_and_residuals <- tibble::tibble(y_hat =
  regression_model$fitted.values, residuals =
  regression_model$residuals)

regression_heteroscedasticity_plot <- fitted_and_residuals |>
  ggplot2::ggplot(aes(y = residuals, x = y_hat)) +
  ggplot2::geom_vline(xintercept = 0, col = "darkgrey") +
  ggplot2::geom_hline(yintercept = 0, col = "darkgrey", ) +
  ggplot2::geom_point(alpha = 0.5, shape = 21, size = 3.2, colour
= "black", fill = "#A4A4A4", stroke = 1) +
  ggplot2::labs(y = "Error (residuals)", x =
  expression(paste("Fitted values of the regression model ",
  hat(y), " (weekly sales in $)"))) +
  ggplot2::ggtitle(expression(paste("Scatter plot residuals (e)
  by fitted values (", hat(y), ")"))) +
  ggplot2::scale_x_continuous(breaks = c(0, 1000, 2000, 3000,
  4000, 5000, 6000, 7000, 8000, 9000, 10000, 11000)) +
  ggplot2::theme(plot.title = element_text(hjust = 0.5)) +
  ggplot2::theme(
    plot.background = element_rect(fill = "lightgrey"),
    panel.background = element_rect(fill = "grey98"),
    panel.border = element_rect(colour = "black", fill = NA,
    linewidth = 0.7),
    text = element_text(size = 12),
    axis.title = element_text(size = 13)
  )

regression_heteroscedasticity_plot
```



💡 Interpretation

By tendency, we see a funnel shape which is open to the right. This indicates heteroscedasticity.

A formal test is given by the **Breusch-Pagan Test** (Breusch & Pagan, 1979), in an enhanced version (Koenker, 1981), which the package `lmtest` (Zeileis & Hothorn, 2002) provides the test as the function `bptest()`.

```
lmtest::bptest(regression_model)
```

studentized Breusch-Pagan test

```
data: regression_model
BP = 9.5337, df = 2, p-value = 0.008507
```

⚠️ Assumption violated

Drawing on both the visual inspection and the formal test, we conclude that the above regression model violates the assumption of equal error variance (i.e., the presence of heteroscedasticity).

As discussed during the lecture, we should, therefore, not trust the above regression model when interpreting significance tests for its regression coefficients.

Potential solutions include (see later applications in this script):

- estimation of robust standard errors (e.g., [White, 1980](#))
- bootstrapping (e.g., [Efron & Gong, 1983](#))

Absence of autocorrelation

We test for the presence of autocorrelation by means of a **Durbin-Watson test**. The package `car` ([Fox & Weisberg, 2019](#)) provides the D-W test as the function `durbinWatsonTest()`.

```
regression_model |> car::durbinWatsonTest()
```

```
lag Autocorrelation D-W Statistic p-value
1      0.08628019      1.805584      0.48
Alternative hypothesis: rho != 0
```

💡 Interpretation

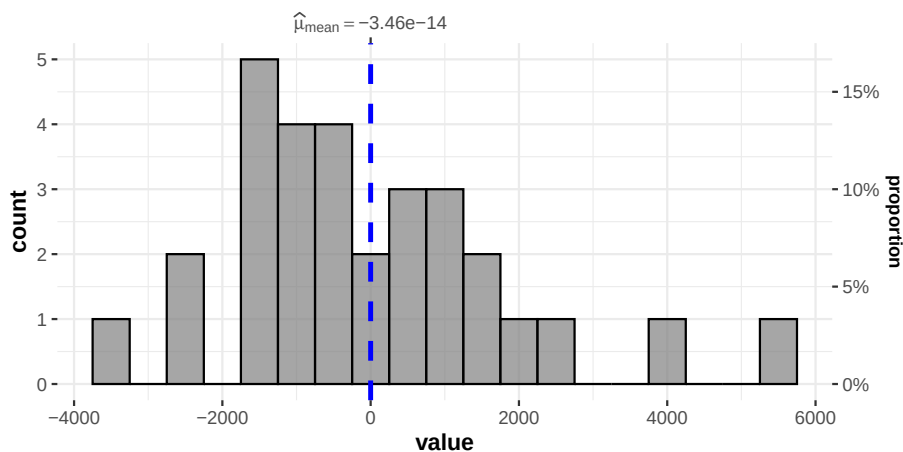
As the empirical p-value far exceeds the α -level 5%, the data supports the notion that we have no problems with autocorrelation.

Normally distributed errors (optional)

First, we can review the histogram of residuals again (for code see above):

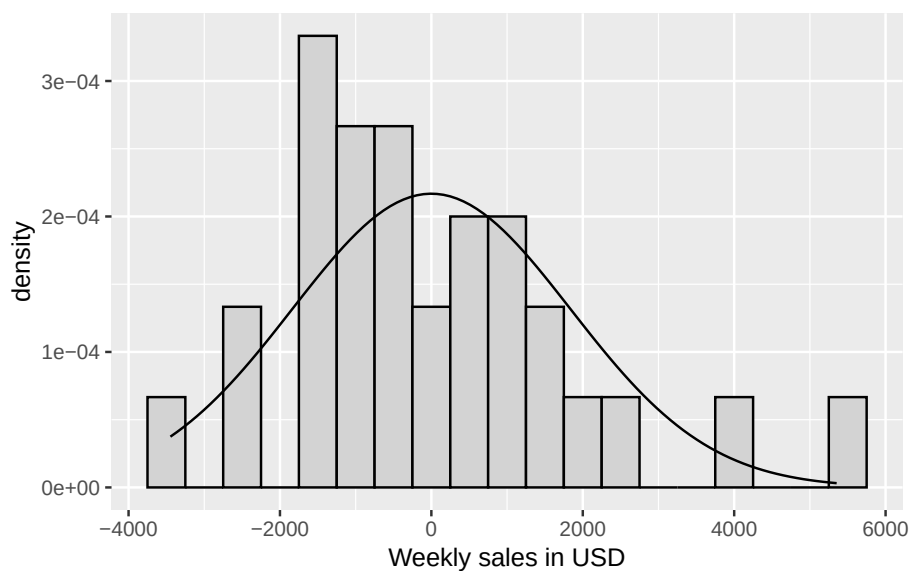
Histogram of model residuals together with normal curve and formal test of mean against the value of 0

$t_{\text{Student}}(29) = -1.03\text{e-}16$, $p = 1.00$, $\hat{g}_{\text{Hedges}} = -1.83\text{e-}17$, $\text{CI}_{95\%} [-0.35, 0.35]$, $n_{\text{obs}} = 30$



Next, we build a second one, also including a normal curve

```
regression_model$residuals |>
  tibble::as_tibble() |>
  ggplot2::ggplot(data = _, aes(value)) +
  ggplot2::geom_histogram(aes(y = ggplot2::after_stat(density)),
    fill = "lightgray",
    col = "black",
    binwidth = 500
  ) +
  ggplot2::stat_function(
    fun = dnorm,
    args = list(
      mean = mean(regression_model$residuals),
      sd = sd(regression_model$residuals)
    )
  )
```



💡 Interpretation

We already know, that the residuals' mean is not significantly different from 0. The black density curve in the plot furthermore represents a superimposed normal distribution. We see that the histogram is more or less well following the curve (take into account that our teaching example only has 30 observations). Eye-catching, however, is one observation with

a relatively high absolute residual (right side of the plot).

To investigate this further, we apply a Shapiro-Wilk-test ([Shapiro & Wilk, 1965](#)) for normality to the obtained residuals of the regression model. For example, the package [rstatix](#) ([Kassambara, 2026](#)) provides the Shapiro-Wilk-test as the function `shapiro_test()`.

```
regression_model$residuals |> rstatix::shapiro_test()
```

```
# A tibble: 1 x 3
  variable          statistic p.value
  <chr>             <dbl>   <dbl>
1 regression_model$residuals    0.957  0.257
```

Interpretation

As the empirical p-value in the Shapiro-Wilk-test exceeds the α -level 5%, the data supports the notion that we have no problems with non-normal errors.

Step 4: Interpreting the regression results and Goodness of Fit (GoF)

The process of interpreting the regression model's results typically covers:

1. Evaluating the statistical significance of the overall model based on the global F-test (lack of significance means that there is little support for the notion that at minimum one of the independent variables has a linear relationship to the dependent variable)
2. Evaluating global goodness of fit (GoF) drawing on R^2 and the adjusted R^2 (how much information of the dependent variable is explainable by the independent variables)
3. Evaluating local statistical significance of the regression coefficients through e.g., t-tests (and potentially: dropping of insignificant independent variables)
4. Interpreting the signs of the regression coefficient (i.e., the direction linear association between independent and dependent variable)
5. Comparing the independent variables' strength of association with the dependent variable (involving standardized regression coefficients when the independent variables are observed on a different scale)

Global F-test, R^2 , and adjusted R^2

Think back to our lecture, when interpreting the regression model, we usually begin with the **global F-Test** and proceed with the calculation of R^2 as well as the adjusted R^2 .

We can review all of these figures by simply calling:

```
regression_model |> summary()
```

Call:

```
lm(formula = sales ~ own_price + promotion, data = sales_data_regression)
```

Residuals:

Min	1Q	Median	3Q	Max
-3447.3	-1206.1	-282.1	761.8	5353.5

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	30304.05	16837.17	1.800	0.08307 .
own_price	-25209.86	15237.86	-1.654	0.10962
promotion	42.27	12.39	3.412	0.00204 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1907 on 27 degrees of freedom

Multiple R-squared: 0.5513, Adjusted R-squared: 0.5181

F-statistic: 16.59 on 2 and 27 DF, p-value: 2e-05

Interpretation

Usually, we first look at the global F-test to evaluate whether it is probable to obtain the observed model when in the underlying population all β_n s are 0. Based on the empirical p-value, which is almost 0, this is very unlikely. Thus, we continue with the R^2 value which is 55.13 percent. According to the common classification (see above), this indicates substantial model fit. The same is true for the adjusted R^2 : 51.81 percent. Note that these values perfectly mirror those seen in the lecture again.

Before continuing with interpreting the regression model's coefficients, let us calculate the global F-value, the R^2 , and the adjusted R^2 manually from the data (we are doing so to support the calculations we have familiarize our-self with during the lecture). Recall:

$$R^2 = \frac{SS_R}{SS_E + SS_R} = \frac{SS_R}{SS_T}$$

$$R_{adj}^2 = 1 - (1 - R^2) * \frac{n - 1}{n - m - 1}$$

$$F = \frac{R^2}{1 - R^2} * \frac{n - m - 1}{m}$$

where m is the number of predictor variables (2), n is the sample size (30). We need to calculate SS_T , and SS_R while keeping in mind:

$$SS_T = SS_E + SS_R$$

These components are calculated in the following way:

$$\sum_{i=1}^n (y_i - \bar{y})^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \sum_{i=1}^n (\hat{y}_i - \bar{y})^2$$

First, we calculate $SS_T = \sum_{i=1}^n (y_i - \bar{y})^2$ (the total variation of observations around the mean)

```
manual_calculation_tibble <- sales_data_regression |>
  dplyr::select(sales) # new tibble only containing y

#mean of dependent variable y
mean_y <- sales_data_regression |>
  dplyr::summarise(
    mean = mean(sales),
  ) |> as.numeric()

#Addends for SST
manual_calculation_tibble <- manual_calculation_tibble |>
  dplyr::mutate(
    y_minus_mean_squared = (sales - mean_y)^2
  )

SST <- sum(manual_calculation_tibble$y_minus_mean_squared)

SST
```

```
[1] 218804418
```

Second, we calculate $SS_R = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2$ (the information explained by the regression model)

```

manual_calculation_tibble$y_hat <- regression_model$fitted.values
# add fitted values from the regression model to the tibble

#Addends for SSR
manual_calculation_tibble <- manual_calculation_tibble |>
  dplyr::mutate(
    y_hat_minus_mean_squared = (y_hat - mean_y)^2
  )

SSR <- sum(manual_calculation_tibble$y_hat_minus_mean_squared)

SSR

```

```
[1] 120631815
```

To complete the picture, SS_E is given by:

```

SSE <- SST - SSR

SSE

```

```
[1] 98172602
```

Now, that we have SS_T and SS_R , we can calculate R^2 as:

```

R_squared <- SSR/SST

R_squared

```

```
[1] 0.5513226
```

Perfectly in line with the value that the summary of $lm()$ provides us with (see above). The adjusted R^2 is easily obtained by:

```

adjusted_R_squared <- 1-(1-R_squared)*((30-1)/(30-2-1))

adjusted_R_squared

```

```
[1] 0.5180872
```

Finally, F is given by:

```
F_value <- ((R_squared)/(1-R_squared))*((30-2-1)/(2))

F_value
```

```
[1] 16.58843
```

Interpretation of regression coefficients

Let us return to the summary of our regression model:

```
regression_model |> summary()
```

Call:

```
lm(formula = sales ~ own_price + promotion, data = sales_data_regression)
```

Residuals:

Min	1Q	Median	3Q	Max
-3447.3	-1206.1	-282.1	761.8	5353.5

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	30304.05	16837.17	1.800	0.08307 .
own_price	-25209.86	15237.86	-1.654	0.10962
promotion	42.27	12.39	3.412	0.00204 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1907 on 27 degrees of freedom

Multiple R-squared: 0.5513, Adjusted R-squared: 0.5181

F-statistic: 16.59 on 2 and 27 DF, p-value: 2e-05

Now that we have evaluated the global F-test as significant and found a substantial model fit, we proceed with testing of the individual regression coefficients.

We see that among the two independent variables, only the variable **promotion** is significantly different from 0 (column $\text{Pr}(>|t|)$, which shows the corresponding p-values from t-tests against the value of 0). Thus, we usually would drop the variable **promotion** from the model to estimate a new, more parsimonious model. However, we will keep both independent variables for illustrative purposes. Remember, we have issues with heteroscedasticity (see above). Due to the biased standard errors (column *Std. Error*), we should not trust the conventional t-tests for the regression coefficients. One way to tackle this problem

is to implement **robust standard errors** (e.g., [White, 1980](#)). Another would be to use bootstrapping (e.g., [Efron & Gong, 1983](#)).

For example, the package `lmtest` ([Zeileis & Hothorn, 2002](#)) provides the function `coefTest()`. In combination with the function `vcovHC()` provided by the package `sandwich` ([Zeileis, Koell, & Graham, 2020](#)), we can implement many different approaches for the estimation of significance tests relying on robust standard errors.

Let us show White's estimator ([White, 1980](#)), which is robust against violations of homoscedasticity (HC in the code below) :

```
regression_model |> lmtest::coefTest(vcov =
sandwich::vcovHC(x=regression_model, type = 'HC'))
```

t test of coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	30304.054	13825.120	2.1920	0.03718	*
own_price	-25209.858	12472.071	-2.0213	0.05326	.
promotion	42.266	13.594	3.1091	0.00439	**

Signif. codes:	0 '***'	0.001 '**'	0.01 '*'	0.05 '.'	0.1 ' ' 1

Interpretation

Note, that the signs and absolute values of the regression coefficients did not change! In contrast, standard errors and the associated t-test results changed. In the corrected test for the independent variables the coefficient for **promotion** remains significant at $p < 0.05$. However, what is interesting is that the p-value in the corrected test for the coefficient of **own_price** is now close to $\alpha = 5\%$. If one argues that the directed hypothesis “*price has a negative relationship with sales*” is justified from an economical perspective ([Sablotty-Wackershauser, Lichters, Guhl, Bengart, & Vogt, 2024](#)), one could also argue that we can divide the empirical p-value by 2 for mimicking a directed (one-sided) test. Against this background, also the price variable has a significant regression coefficient. It seems as if even the limited extent of heteroscedasticity in our example is enough to lead to a **suppression effect** (see lecture), where the correlation between two IVs suppresses the significant relationship of one of the variables with the DV in the regression model.

 Do not overgeneralize

Even if we would have come to the conclusion that **own_price** is not a significant predictor for sales, this would not mean that price is not related to sales. It only means that price has no correlation with sales in the ranges of prices that are recorded in the sample. That is, more extreme price changes might nevertheless influence sales.

For sake of completeness, below we repeat this analysis step ones with **robust confidence intervals** ($\alpha = 0.05$ and $\alpha = 0.10$) instead of t-tests and tests with standard errors that are robust against heteroscedasticity and autocorrelation simultaneously (Newey & West, 1987).

```
regression_model |> lmtest::coefci(level = 0.95, vcov =  
sandwich::vcovHC(x=regression_model, type = 'HC')) # undirected
```

		2.5 %	97.5 %
(Intercept)	1937.25062	58670.85720	
own_price	-50800.43323	380.71721	
promotion	14.37263	70.15947	

```
regression_model |> lmtest::coefci(level = 0.90, vcov =  
sandwich::vcovHC(x=regression_model, type = 'HC')) # directed
```

		5 %	95 %
(Intercept)	6755.88661	53852.22122	
own_price	-46453.39189	-3966.32413	
promotion	19.11085	65.42124	

```
regression_model |> lmtest::coeftest(vcov = sandwich::NeweyWest)
```

t test of coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	30304.054	15821.742	1.9153	0.066097 .
own_price	-25209.858	14438.704	-1.7460	0.092183 .
promotion	42.266	12.252	3.4498	0.001858 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Drawing on the above discussion, we leave both IV's in the model. Next we interpret their direction of association with the DV as well as their relative contribution in explaining variation in the DV. The IV **own_price** is measured in

USD (as the DV sales is). The IV **promotion**, however, is a complex index combining the level of promotional activities (e.g., product placement, advertising, and coupons) for the milk product. Here are the coefficients again

Call:

```
lm(formula = sales ~ own_price + promotion, data = sales_data_regression)
```

Coefficients:

(Intercept)	own_price	promotion
30304.05	-25209.86	42.27

We see that in the complete absence of promotional activities and a price of 0 USD (unrealistic!), supermarkets are estimated to generate 30304.05 USD of sales per week for selling the milk product. Further, with each increase of the product price for 1 USD, the model estimates that a supermarket generates 25209.86 USD less. Keep in mind that the average sales in our dataset are 4920.5 USD, whereas the mean price is around 1.08 USD. Therefore, it seems logical that supermarkets are estimated to generate approximately $30304.05 - 25209.86 = 5094.19$ USD in milk sales when the price increases from 0 to 1 USD. Furthermore, the model estimates that an increase in promotional activities for 1 index point will lead to an increase of sales for 42.27 USD.

 Never focus on the difference in p-values!

When comparing the independent variables' strength of association with the dependent variable, we should never base our conclusions on the relative differences between p-values! Rather, we should take the differences in the regression coefficients' absolute values into account. If different IVs are measures in different units/scales, we should compare the standardized regression coefficients instead.

Next, we will evaluate the IVs' relative contribution in explaining variation in the DV. For this purpose, we need the standardized regression coefficients. Fortunately, many packages such as the package `lm.beta` (Behrendt, 2025) provide suitable functions such as `lm.beta()` that we will use below:

```
regression_model |> lm.beta::lm.beta() |> summary()
```

Call:

```
lm(formula = sales ~ own_price + promotion, data = sales_data_regression)
```

Residuals:

Min	1Q	Median	3Q	Max
-3447.3	-1206.1	-282.1	761.8	5353.5

Coefficients:

	Estimate	Standardized	Std. Error	t value	Pr(> t)
(Intercept)	3.030e+04	NA	1.684e+04	1.800	0.08307 .
own_price	-2.521e+04	-2.670e-01	1.524e+04	-1.654	0.10962
promotion	4.227e+01	5.506e-01	1.239e+01	3.412	0.00204 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1907 on 27 degrees of freedom
Multiple R-squared: 0.5513, Adjusted R-squared: 0.5181
F-statistic: 16.59 on 2 and 27 DF, p-value: 2e-05

```
regression_model |> lm.beta::lm.beta()
```

Call:

```
lm(formula = sales ~ own_price + promotion, data = sales_data_regression)
```

Standardized Coefficients::

(Intercept)	own_price	promotion
NA	-0.2669624	0.5506476

Interpretation

Since the absolute value of standardized regression coefficient for **promotion** is higher than those for **own_price**, we conclude that promotion activities have a higher contribution in explaining variation in milk sales as compared to price changes.

A formal test is missing

Above, we compared the absolute values of the standardized regression coefficients. However, we did not conduct a formal test to evaluate whether the difference between the two regression coefficients is significant. In *R* many approaches exist to implement such a formal procedure. One popular approach is to test the standardized coefficients' difference by means of bootstrapping (e.g., [Efron & Gong, 1983](#)). Another approach tests both coefficients for significant differences in terms of a z-test. For example, the package [marginaleffects](#) ([Arel-Bundock, Greifer, & Heiss, 2024](#)) provides the function `hypothesis()` for this purpose. A third variant

is the comparison of the robust effect size index (RESI) for two predictor variables (Jones, Kang, & Vandekar, 2025). The package `RESI` (Jones et al., 2025) provides the function `resi()` for this purpose.

Below, we illustrate the function `hypothesis()` (Arel-Bundock et al., 2024) to test whether the difference between the two regression coefficients is significantly different from 0. We will also implement a robust variance-covariance matrix estimator (Newey & West, 1987) to account for heteroscedasticity and autocorrelation. Importantly, when implementing such a test, we need to account for different measurement units of the IV's. Therefore, we will re-estimate the regression model with standardized IV's (i.e., z-standardization). The function `scale()` provides this functionality. Within the function call, `b2` represents the regression coefficient for **own_price** and `b3` the regression coefficient for **promotion**. The null hypothesis is that both coefficients are equal (i.e., $b_2 - b_3 = 0$). The alternative hypothesis is that they are not equal (i.e., $b_2 - b_3 \neq 0$).

```
regression_model_standard <- lm(formula = sales ~
scale(own_price) + scale(promotion), data =
sales_data_regression)

regression_model_standard |>
  margineffects::hypotheses(model = _,
                           hypothesis = "b2 - b3 = 0",
                           vcov = sandwich::NeweyWest)
```

Hypothesis	Estimate	Std. Error	z	Pr(> z)	S	2.5 %	97.5 %
b2-b3=0	-2246	577	-3.89	<0.001	13.3	-3376	-1115

Interpretation

Based on this formal test and the standardized regression coefficients, we conclude that the regression coefficient for **promotion** is significantly higher than the regression coefficient for **own_price**. Put differently, promotion activities have a higher contribution in explaining variation in milk sales as compared to price changes.

Step 5: Validating the regression model

During the lecture we have learned that validating the regression model includes an assessment of the model's stability (i.e., do the model coefficients remain

stable if one changes the data basis or adds/removes predictor variables?) and predictive validity (does the model predict values for the DV of observations not used for calibration well?). While stability ensures us that our interpretation of the regression model's results observed for the drawn sample of stores does not hinge on a miss specified model, productive validity is important when the goal of the market research study is to predict sales for stores not included in the sample. Sometimes, market researchers just try to understand relationships among variables within a given sample, other times predictive validity is more important since the goal of the market research study is to find a suitable prediction model.

Evaluating the regression model's stability

For illustrative purpose, we apply a **Split-sample validation** with a 50:50 split into two subsamples. Remember from the lecture, such an approach involves splitting into two groups and re-running the model on each subset. Results are stable if both samples produced similar results (i.e., signs of the individual parameters should at least be consistent, and significant variables should remain so).

Keep in mind that our model has a great chance of not being stable since each sub sample only contains $n=15$ observations. Further, below, we use the function `sample_frac(.50)` (`dplyr` package (Wickham, François, Henry, Müller, & Vaughan, 2026)) to create random sub sample. Thus, results might differ depending on your machine. Below, we use the function `modelsummary()` (`modelsummary` package (Arel-Bundock, 2022)) for comparing both models side to side.

```
sales_data_regression
```

```
# A tibble: 30 x 4
  supermarket promotion own_price sales
    <dbl>         <dbl>    <dbl> <dbl>
1         1         12.0      1.1  3454
2         2         22.0     1.08  3966
3         3         22.0     1.08  2952
4         4         22.0     1.08  3576
5         5         21.4     1.08  3692
6         6         22.0     1.08  3226
7         7         47.0     1.09  3776
8         8        117.     1.05 14134
9         9         92.0     1.1   5114
10        10         62.0     1.08  4022
# i 20 more rows
```

```

# create id variable
sales_data_regression <- sales_data_regression |>
dplyr::mutate(id = dplyr::row_number())

# create estimation set
sales_data_regression_subsample_1 <- sales_data_regression |>
dplyr::sample_frac(.50)

# create validation set
sales_data_regression_subsample_2 <-
dplyr::anti_join(sales_data_regression,
sales_data_regression_subsample_1, by = 'id')

# estimate both regression models
model_subsample_1 <- lm(formula = sales ~ own_price + promotion,
data = sales_data_regression_subsample_1)

model_subsample_2 <- lm(formula = sales ~ own_price + promotion,
data = sales_data_regression_subsample_2)

#compare results with modelsummary
models <- list(
  "Model_1" = model_subsample_1,
  "Model_2" = model_subsample_2
)

modelsummary::modelsummary(models,
  statistic = c("std.error", "p.value"),
  shape = term ~ model + statistic,
  fmt = fmt_decimal(digits = 2, pdigits = 3),
  gof_omit = "AIC|BIC|Log.Lik|Deviance|RMSE",
  align = "lrrrrrr")

```

Evaluating the regression model's predictive validity

During lecture, we have learned about multiple approaches to evaluate predictive validity in linear regression. Below, we will exemplary apply **k-fold cross validation** and **leave-on-out validation** (i.e., n-fold cross validation). For example, the package **cv** (Fox & Monette, 2026) provides as function with the package's name. We will use this function for a simple 5-fold cross-validation first (see lecture slides). However, the package comes with one drawback. It is not apt in handling variables possessing variable labels. Therefore, below we re-estimate the regression model, but based on the data without variable labels

	Model_1			Model_2		
	Est.	S.E.	p	Est.	S.E.	p
(Intercept)	53 621.22	26 911.45	0.070	22 061.42	22 442.33	0.345
own_price	-46 487.30	24 479.93	0.082	-17 495.67	20 158.43	0.402
promotion	42.38	16.80	0.027	30.01	20.89	0.176
Num.Obs.	15			15		
R2	0.617			0.481		
R2 Adj.	0.553			0.395		
F	9.667			5.564		

(function `remove_labels()` from [labelled](#) ([Larmarange, 2025](#))).

Remember, when we judged the within sample GoF above (e.g., R^2), we based our conclusions in great part on: $\sum_{i=1}^n (y_i - \hat{y}_i)^2$ (the sum of squares between the fitted values of the regression model ($\hat{y}$) and the observed values for the dependent variable (y)). Thus, the lower the sum of squared models residuals, the higher the global GoF is.

Below, we apply a similar logic when using **5-fold cross validation**. More precisely, we have to tell the `cv()` function which measure of misfit for out-of-sample predictions to use when predicting each hold-out fold's values for the dependent variable. The default is the **mean squared error (MSE)** of prediction, calculated as $MSE = \frac{\sum_{j=1}^k \sum_{i=1}^d (y_{j,i} - \hat{y}_{j,i})^2}{n}$, where k is the number of folds and $1 \dots d$ indicates the number of cases within a fold. The lower the MSE, the higher the predictive validity of the model. Instead of the MSE, below we use the **root mean squared error of prediction (RMSE)**, given by: $RMSE = \sqrt{\frac{\sum_{j=1}^k \sum_{i=1}^d (y_{j,i} - \hat{y}_{j,i})^2}{n}}$. This measure has the nice property of possessing the same measurement unit as the dependent variable (sales in USD). Also, we set `reps = 3`. We rerun 5-fold cross-validation for 3 times to get an impression of how strongly the results depend on the concrete allocation of data into the five folds.

Note that due to the random elements in this procedure, your results might look slightly different from those below, if you are not setting the same seed for the random number generator.

5-fold cross-validation

```
set.seed(26111985) # set seed for reproducibility

lm(formula = sales ~ own_price + promotion, data =
remove_labels(sales_data_regression)) |>
  cv::cv(k = 5, criterion = rmse, reps = 3) |>
```

```
summary()
```

```
Replicate 1:  
5-Fold Cross Validation  
method: Woodbury  
criterion: rmse  
cross-validation criterion = 1996.592  
full-sample criterion = 1808.983
```

```
Replicate 2:  
5-Fold Cross Validation  
method: Woodbury  
criterion: rmse  
cross-validation criterion = 2135.71  
full-sample criterion = 1808.983
```

```
Replicate 3:  
5-Fold Cross Validation  
method: Woodbury  
criterion: rmse  
cross-validation criterion = 2089.171  
full-sample criterion = 1808.983
```

```
Average:  
5-Fold Cross Validation  
method: Woodbury  
criterion: rmse  
cross-validation criterion = 2054.516 (69.53104)  
full-sample criterion = 1808.983
```

Interpretation

Above, we see that, on average, the model predicts the sales of a new store not included in calibration rather badly. The RMSE is around 2100 USD. Given that the mean sales within the sample are 4920.5 USD, this is a high number, but not surprising with an eye on our tiny sample size. The output portrays the results for each repetition and then averaged across repetitions, with the standard deviations of the CV criterion in parentheses. Based on these results we should not trust the model when doing predictions. We also see that the RMSE error within the complete sample is much lower. Such an adverse situation where the out-of-sample error of prediction is much higher than the in-sample error drawing on the whole sample is typically called **overfitting**.

Leave-one-out (LOO) n-fold cross-validation

```
set.seed(26111985) # set seed for reproducibility

lm(formula = sales ~ own_price + promotion, data =
remove_labels(sales_data_regression)) |>
  cv::cv(k = "loo", criterion = rmse)
```

cross-validation criterion (rmse) = 2065.1

Interpretation

The results of a loo cross-validation closely mirror those of the 5-fold cross validation.

When prediction is at the forefront

Note that when the main goal of a study is not significance testing but rather prediction, we should rather use cross-validation also to decide which independent variables should remain in the regression model (Hastie, Tibshirani, & Friedman, 2009, p. 246).

For sake of completeness, below we apply leave-one-out (loo) cross-validation to compare a model with and another without the **price** included as a independent variable. We first use cv's *models()* function to create a list of regression models to compare. Then, we use the *cv()* function to present the results of cross-validation for each list element.

```
set.seed(26111985) # set seed for reproducibility

model_list <- cv::models(
  regression_price_promotion = lm(formula = sales ~ own_price +
  promotion, data = remove_labels(sales_data_regression)),
  regression_promotion = lm(formula = sales ~ promotion, data =
  remove_labels(sales_data_regression))
)

model_list |> cv::cv(data = remove_labels(sales_data_regression),
  k = "loo",
  criterion = rmse)
```

Model regression_price_promotion:

```
cross-validation criterion = 2065.1
```

```
Model regression_promotion:  
cross-validation criterion = 2093.559
```

Interpretation

The out-of-sample prediction error (RMSE) is slightly lower for the model with both predictors included (**promotion** and **price**). This result delivers additional support for the decision to leave the independent variable **own_price** in the regression model.

Step 6: Using the regression model

Using the regression model means prediction! This is easy in *R*. Let us take the lecture example.

Prediction for the price set at 1.10 USD and promotional activities at 50

```
predictions <- regression_model |> predict.lm(  
  newdata = data.frame(promotion = 50, own_price = 1.10),  
  interval = "prediction",  
  level = 0.95  
)
```

```
predictions
```

	fit	lwr	upr
1	4686.512	658.6533	8714.372

Interpretation

Given the above values for the independent variables, the regression model estimates that the supermarket's milk weekly milk sales will be approximately 4686.51 USD. However, please pay attention to the extraordinary wide 95% prediction interval for this estimate. This showcases how much uncertainty we face when using regression models based on tiny sample sizes for predictions.

Factor Analysis and Principal Component Analysis

In the following section, we reproduce the principal component analysis (**PCA**) example from the lecture within *R*. The dataset stems from Sarstedt, Ringle, Raithel, & Gudergan (2014) and is also used as an introductory example in Chapter 8 in Sarstedt & Mooi (2019).

Fan satisfaction represents a vital management goal of any (professional) soccer club. Empirical studies often consider only the role of individual aspects of fan satisfaction on target variables. Therefore, A soccer club commissioned a questionnaire comprising 99 previously identified items through literature databases and focus groups of fans (high-dimensional data). Development of the FANSAT scale for measurement and prediction of soccer fan behavior. We use only 7 of the items (as was done during the lecture). Like in the lecture, we also use the term *factor* to refer to principal components (when conducting PCA).

Question (PCA): Which umbrella terms can we use to summarize the sets of variables that load highly on the extracted principal components? Question (Factor analysis aka, principal axis analysis, **FA**): What is the underlying construct (common reason) for the strong correlations between the variables and their corresponding factors?

In total, we will go through the seven steps that we have discussed during the lecture:

1. Checking the data requirements and conducting preliminary analyses
2. Extracting the factors
3. Determining the number of factors to retain
4. Interpreting the (rotated) factor solution
5. Evaluating the goodness-of-fit of the factor solution
6. Computing the factor scores (optional)

However, we will only focus on the things that incorporate applications in *R*. Therefore, we will not repeat rules-of-thumb (for example, regarding sample sizes, etc.).

Let us download the data from our [Chair's homepage](#) at the [Otto von Guericke University](#).

Import the data for Principal Component Analysis / Factor Analysis

The dataset is stored as a .rds file within the following directory: https://marketing.ovgu.de/marketing_media/L

The dataset's name is **Soccer.rds**.

We first set the url (Uniform Resource Locator) for the download.

```
url_of_file <-  
"https://marketing.ovgu.de/marketing_media/Downloads/MMA/Soccer.rds"
```

Then, we download the file to our current *R* project.

```
Soccer_data_PCA_FA <- readRDS(url(url_of_file)) |>  
tibble::as_tibble()
```

If anything does not work out as expected on your machine, you can download the data by clicking on the following button.

Let's have a look at the dataset:

```
print(Soccer_data_PCA_FA, n=25)
```

```
# A tibble: 251 x 7  
  x_1          x_2          x_3    x_4    x_5    x_6    x_7  
  <dbl+lbl>    <dbl+lbl>    <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>  
1 5          5          5      6      5      5      5  
2 7 [very dissatisfied] 7 [very dissa~ 7 [ver~ 7 [ver~ 7 [ver~ NA      1 [ver~  
3 5          5          4      2      1 [ver~ NA      6  
4 6          6          4      3      2      NA      6  
5 7 [very dissatisfied] 7 [very dissa~ 7 [ver~ 5      7 [ver~ 4      5  
6 3          3          3      2      2      4      6  
7 7 [very dissatisfied] 7 [very dissa~ 6      5      5      4      4  
8 7 [very dissatisfied] 7 [very dissa~ 4      1 [ver~ 1 [ver~ NA      7 [ver~  
9 3          4          4      7 [ver~ 6      NA      4  
10 7 [very dissatisfied] 5          4      4      6      4      3  
11 7 [very dissatisfied] 7 [very dissa~ 7 [ver~ 5      5      NA      4  
12 7 [very dissatisfied] 7 [very dissa~ 6      4      6      6      4  
13 7 [very dissatisfied] 7 [very dissa~ 7 [ver~ 7 [ver~ 7 [ver~ 6      6  
14 7 [very dissatisfied] 7 [very dissa~ 7 [ver~ 6      6      6      6  
15 7 [very dissatisfied] 7 [very dissa~ 6      6      6      7 [ver~ 7 [ver~  
16 6          6          5      4      4      3      4  
17 7 [very dissatisfied] 6          5      3      1 [ver~ 4      4  
18 7 [very dissatisfied] 7 [very dissa~ 7 [ver~ 7 [ver~ 7 [ver~ 2      3  
19 7 [very dissatisfied] 7 [very dissa~ 6      5      6      6      5
```

```

20 7 [very dissatisfied] 3          3          5          6          4          4
21 4          4          4          5          7 [ver~ 4          6
22 6          6          6          2          4          NA          5
23 6          6          5          1 [ver~ 1 [ver~ 4          3
24 7 [very dissatisfied] 7 [very dissa~ 7 [ver~ 3          2          NA          5
25 7 [very dissatisfied] 7 [very dissa~ 7 [ver~ 5          6          4          3
# i 226 more rows

```

We see that we have 251 cases and 7 variables (`x_1` to `x_7`). Unfortunately, we also spot missing values (*NA*) in the data. These missing value are no non-responses in the questionnaire. Rather, they represent respondents’ “**don’t know**” reaction. We can briefly summarize variable and the value labels, as well as the number of missing for each variable by using the `look_for()` function from the package `labelled` (Larmarange, 2025).

```
Soccer_data_PCA_FA |> labelled::look_for()
```

pos	variable	label	col_type	missing
1	x_1	Condition of the stadium	dbl+lbl	2
2	x_2	Outer appearance of the sta~	dbl+lbl	2
3	x_3	Interior design of the stad~	dbl+lbl	3
4	x_4	Quality of the team composi~	dbl+lbl	5
5	x_5	Number of stars in the team	dbl+lbl	12
6	x_6	Price of annual season tick~	dbl+lbl	47
7	x_7	Entry fees	dbl+lbl	3

```

values
[1] very satisfied
[7] very dissatisfied
[1] very satisfied
[7] very dissatisfied
[1] very satisfied
[7] very dissatisfied
[1] very satisfied
[7] very dissatisfied
[1] very satisfied
[7] very dissatisfied
[1] very satisfied
[7] very dissatisfied

```

```
[1] very satisfied
[7] very dissatisfied
```

Next to the labels, we see that the variable `x_6` (satisfaction with the price of the annual season ticket) has 47 missing values. After all, people can't tell about this aspect of fan satisfaction when they are not buying the annual ticket. Why is this important? PCA/ FA analysis uses the complete cases for calculations. Thus, the effective sample size in the following steps will not be $n=251$. More importantly, it will also not be $251-47=204$ since a case with no missing values in `x_6` can have missing values elsewhere. To get an overview, we subsequently analyze the structure of missing data using functions provided by the package `naniar` (Tierney & Cook, 2023).

```
Soccer_data_PCA_FA |>
  naniar::miss_case_table()
```

```
# A tibble: 6 x 3
  n_miss_in_case n_cases pct_cases
      <int>      <int>      <dbl>
1           0       195       77.7
2           1        47       18.7
3           2         5        1.99
4           3         1        0.398
5           4         1        0.398
6           5         2        0.797
```

Note!

The effective sample size in the PCR/FA will be $n=195$.

Step 1: Checking the data requirements and conducting preliminary analyses

Question: Are the variables sufficiently correlated with each other?

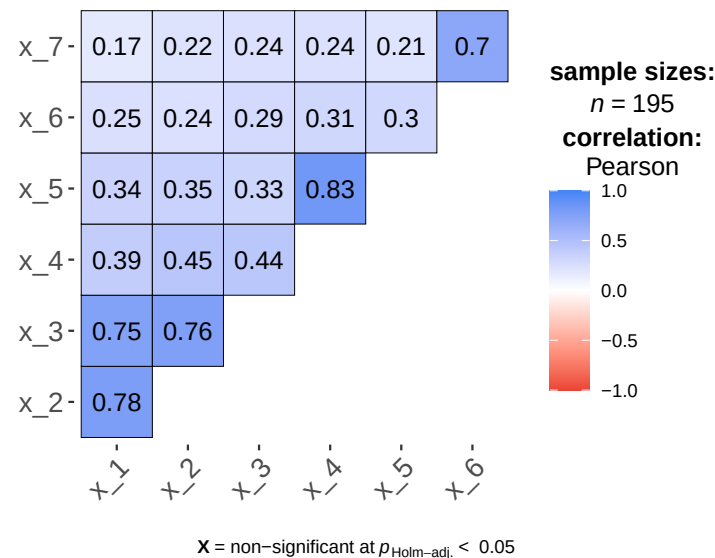
We will approach the answer to this question by using multiple analyses.

Correlation matrix and tests for significance

We begin with the pairwise Pearson correlations (r) and the corresponding tests whether these pairwise correlations are significantly different from 0. Like in other sections of this script, we visualize the correlation matrix with the help

of the function `ggcorrmat()` provided by the package `ggstatsplot` (Patil, 2021). Note, that for the following analysis steps we only use the complete cases, while dropping the incomplete ones. We do so by applying the function `drop_na()` (`rstatix` (Kassambara, 2026)).

```
Soccer_data_PCA_FA |>
  tidyr::drop_na() |>
  ggstatsplot::ggcorrmat(k=3L)
```



💡 Interpretation

Note that the values perfectly mirror those in the lecture (IBM SPSS outputs). All pairwise correlations differ significantly from 0 (none of the matrix elements is crossed out). Furthermore, the colors reveal some interesting patterns. x_1 , x_2 , and x_3 somehow are correlated highly. The same is true for the pairs x_4 & x_5 as well as for x_6 & x_7 .

Kaiser-Meyer-Olkin (KMO) criterion/ Measure of Sampling Adequacy (MSA)

Next, we use the **Kaiser-Meyer-Olkin (KMO)** statistic. The KMO statistic, also called the **measure of sampling adequacy (MSA)**, indicates whether the other variables in the dataset can explain the correlations between variables. Kaiser (1974) introduced the statistic and recommended threshold values for KMO/MSA:

Table 6: Interpretation guidelines KMO/MAS

KMO/MAS	Interpretation
≥ 0.90	excellent
≥ 0.80	deserving
≥ 0.70	quite good
≥ 0.60	moderate
≥ 0.50	miserable
< 0.50	unbearable

Thus, global the KMO/MAS value should exceed 0.5. We should consider deleting variables with local KMO/MAS values below 0.5.

Below, we use the function *KMO()* provided by the package [psych](#) (2026b). to ask for the MSA values. In general, this package [psych](#) provides many very useful functions for conducting PCAs/FAs.

```
Soccer_data_PCA_FA |>
  tidyr::drop_na() |>
  psych::KMO()
```

```
Kaiser-Meyer-Olkin factor adequacy
Call: psych::KMO(r = tidyr::drop_na(Soccer_data_PCA_FA))
Overall MSA = 0.72
MSA for each item =
  x_1 x_2 x_3 x_4 x_5 x_6 x_7
0.79 0.80 0.82 0.67 0.64 0.65 0.61
```

Interpretation

With a global KMO/MSA value of 0.72, the data shows a ‘quite good’ adequacy for PCA/FA. Also when looking at the local values, the minimum value is 0.61 (**x_7**), indicating ‘moderate’ adequacy of each of the seven questions.

Bartlett’s test of sphericity

Finally, we conduct Bartlett’s test of sphericity ([Bartlett, 1950](#)). Keep in mind, if the empirical p-value is insignificant (i.e., data is likely to come from a population with zero correlations between all variables), the appropriateness of factor analysis should be questioned.

The [psych](#) package also provides a function for the Bartlett test:

```
Soccer_data_PCA_FA |>
  tidyr::drop_na() |>
  psych::cortest.bartlett()
```

```
$chisq
[1] 809.3528
```

```
$p.value
[1] 9.435409e-158
```

```
$df
[1] 21
```

Interpretation

Since the empirical p-value is almost 0, the data is very unlikely to stem from a population where all pairwise correlations are 0. Thus, the data seems to be appropriate for PCA/FA.

Furthermore, the results of the KMO statistic and the Bartlett test are well-aligned with the IBM SPSS outputs we saw during our lecture.

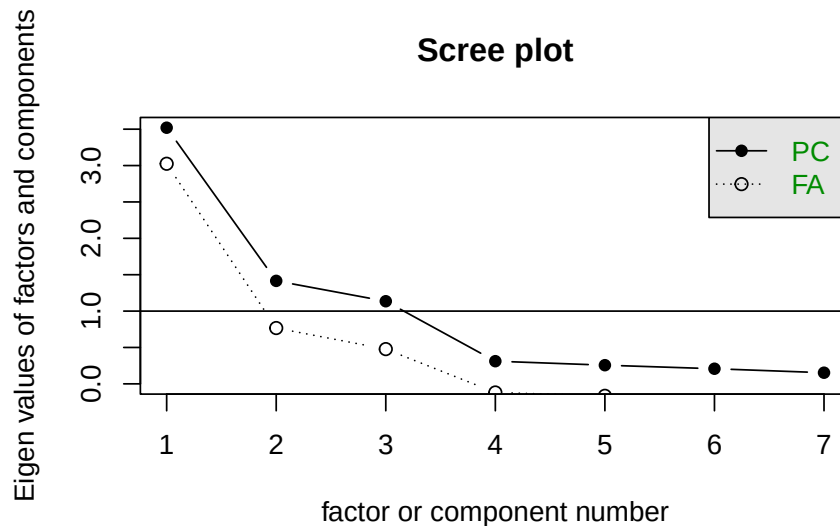
Step 2: Extracting the factors and Step 3: Determining the number of factors to retain

We will complete both steps combined. First, we decide how many factors to extract. Second, we only extract the recommended number of factors.

Kaiser criterion and Scree plot

The function `scree()` is also provided by the `psych` package. It delivers both information (a) how many eigenvalues are > 1 in PCA (Kaiser's criterion, (Kaiser & Dickman, 1959)), and (b) the scree plot hopefully indicating an “Elbow”. Note that the function, by default, provides the information for principle component analysis (PCA) and factor analysis (FA). In the latter case, the eigenvalues are generally lower since in FA (vs. PCA), not all information contained in the original variables is thought to be reproduced by the factor solution (i.e., the sum of communalities does not equal to the number of variables).

```
Soccer_data_PCA_FA |>
  tidyr::drop_na() |>
  psych::scree()
```



💡 Interpretation

We will interpret the PCA results of the figure (PC). Here, we see that according to the Kaiser criterion, we should retain only three factors. The first pronounced elbow appears at four factors. Thus, with the elbow criterion, we should extract one factor less (three factors in total) because the left-hand side eigenvalue is much higher than the eigenvalue seen for the fourth factor.

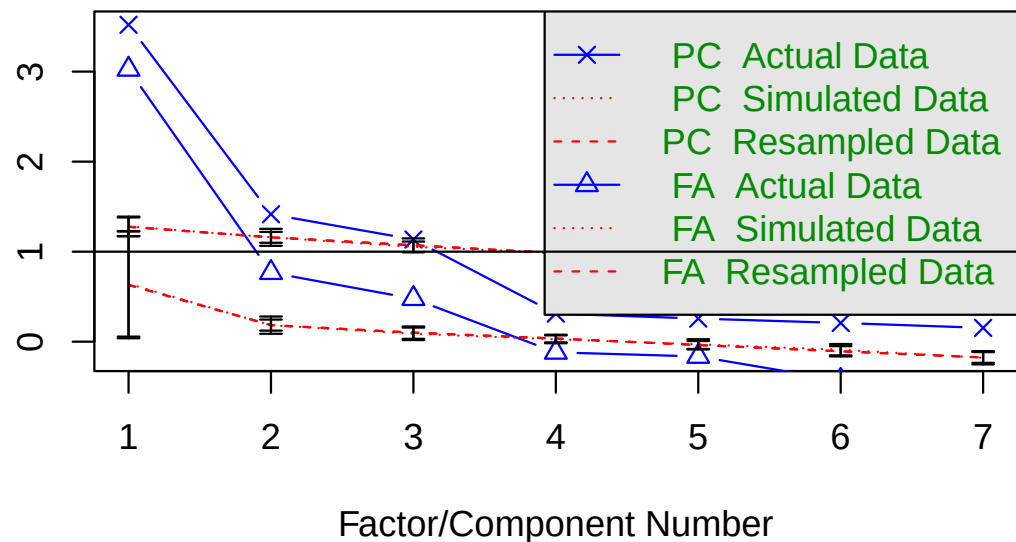
Parallel Analysis

Next, we proceed with parallel analysis ([Horn, 1965](#)), the recommended approach to determine the number of factors to retain. In parallel analysis, we extract as much factors as we observe eigenvalues greater than corresponding 95% percentile of eigenvalues from randomly generated datasets with the same sample size and number of variables than the original dataset. The [psych](#) package comes with a corresponding function (*fa.parallel()*).

```
Soccer_data_PCA_FA |>
  tidyr::drop_na() |>
  psych::fa.parallel(quant = 0.95, error.bars = TRUE)
```

envalues of principal components and factor an

Parallel Analysis Scree Plots



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

💡 Interpretation

Randomly generated datasets comes in two forms: randomly generated uncorrelated data following a normal distribution or resampled data (two separate red lines in the plot above indicate the mean eigenvalues for both random generation strategies). In the present case, the parallel analysis suggests to retain 3 factors (in case of PCA and FA).

Note that due to the small dataset and the random elements involved in parallel analysis your results might slightly look different. Further note that the error bars in the plot represent 1-standard deviation and not the 95% percentile of randomly drawn data. To apply the 95% percentile rule, we set $quant = 0.95$ in the above code chunk, which is not reflected by the plot.

Extraction of unrotated solution

In the subsequent step, we extract the factor solution. We remember from the lecture that we never interpret the unrotated factor solution. However,

for illustration, we will do so next. For factor extraction, we draw on [psych](#)'s *principal()* function.

```
PCA_results_unrotated <- Soccer_data_PCA_FA |>
  tidyr::drop_na() |>
  psych::principal(
    rotate = "none",
    nfactors = 3
  )

PCA_results_unrotated
```

Principal Components Analysis

Call: `psych::principal(r = tidyr::drop_na(Soccer_data_PCA_FA), nfactors = 3, rotate = "none")`

Standardized loadings (pattern matrix) based upon correlation matrix

	PC1	PC2	PC3	h2	u2	com
x_1	0.79	-0.40	0.25	0.85	0.150	1.7
x_2	0.81	-0.37	0.23	0.85	0.148	1.6
x_3	0.81	-0.33	0.26	0.83	0.168	1.5
x_4	0.75	0.08	-0.59	0.91	0.088	1.9
x_5	0.68	0.14	-0.66	0.92	0.077	2.1
x_6	0.56	0.69	0.25	0.84	0.157	2.2
x_7	0.49	0.72	0.33	0.86	0.142	2.2

	PC1	PC2	PC3
SS loadings	3.52	1.42	1.14
Proportion Var	0.50	0.20	0.16
Cumulative Var	0.50	0.71	0.87
Proportion Explained	0.58	0.23	0.19
Cumulative Proportion	0.58	0.81	1.00

Mean item complexity = 1.9

Test of the hypothesis that 3 components are sufficient.

The root mean square of the residuals (RMSR) is 0.03
with the empirical chi square 9.62 with prob < 0.022

Fit based upon off diagonal values = 0.99

Interpretation

The function's results provide us with many useful information. We will focus on the essential ones, which we also discussed during the lecture.

The first table in the output shows the matrix of factor loadings (columns PC1 to PC3, PC stands for principal component). Keep in mind that we never interpret the unrotated loadings. Also, note that the order of rows (the items) is not the same as in the IBM SPSS output we have seen during the lecture.

The numerical results indicate that the first three components together account for 87% of the variance of the initial items (column *PC3*, row *Cumulative Var*), which is clearly above the lower threshold of 0.5 and also above the recommended value of at least 0.75.

If we look at the item-specific communalities (*h2*), we see that the solution does exceed the common threshold of 50% for each of the items, which is satisfying (*h2* is simply the sum of squared factor loadings of a corresponding item). Since the communalities are the item-specific information accounted for by the factor solution, their sum divided by the number of items is again 87% ($[0.85+0.85+0.83+0.91+0.92+0.84+0.86]/7=0.866$).

The column *u2* presents the corresponding uniqueness values. Keep in mind that $1 - \text{communality} = \text{uniqueness}$. Thus, for *x_1*, 15% percent of the original variable's information is not accounted for by the three extracted factors.

Finally, the column *com* lists the items' complexity (Hofmann, 1977, 1978). This values always is allways between 1 and the number of factors extracted. If an item represents only one latent construct, then the value in this column is close to one. Higher values indicate that assigning an item to only one factor is challenging. Values higher than two suggest that an item shares more information with other factors than the one it is assigned to based on the highest factor loading. We also do not interpret these values for the unrotated factor solution.

Step 4: Interpreting the (rotated) factor solution

To enhance the interpretability of the factor solution, we next apply a factor rotation scheme to the unrotated matrix of factor loadings. As we have seen during the lecture, rotating a factor solution in an orthogonal way preserves the factors' interdependence. Furthermore, it does not change the total information accounted for by the factor solution. Instead, some information explained by the first few factors in the unrotated solution is shifting in the direction of the latter factors extracted. Mathematically, rotating is simply a matrix multiplication where a rotation matrix is multiplied from the right to the unrotated factor loadings matrix. During the lecture, we have learned that the most often applied orthogonal rotation is the **Varimax** rotation (Kaiser, 1958).

Below, we will use this rotation:

```

PCA_results_varimax <- Soccer_data_PCA_FA |> tidyr::drop_na() |>
psych::principal(
  rotate = "varimax",
  nfactors = 3
)

PCA_results_varimax

```

Principal Components Analysis

Call: `psych::principal(r = tidyr::drop_na(Soccer_data_PCA_FA), nfactors = 3, rotate = "varimax")`

Standardized loadings (pattern matrix) based upon correlation matrix

	RC1	RC3	RC2	h2	u2	com
x_1	0.90	0.17	0.08	0.85	0.150	1.1
x_2	0.89	0.20	0.11	0.85	0.148	1.1
x_3	0.88	0.18	0.15	0.83	0.168	1.1
x_4	0.28	0.90	0.14	0.91	0.088	1.2
x_5	0.17	0.94	0.13	0.92	0.077	1.1
x_6	0.14	0.18	0.89	0.84	0.157	1.1
x_7	0.10	0.08	0.92	0.86	0.142	1.0

	RC1	RC3	RC2
SS loadings	2.53	1.83	1.71
Proportion Var	0.36	0.26	0.24
Cumulative Var	0.36	0.62	0.87
Proportion Explained	0.42	0.30	0.28
Cumulative Proportion	0.42	0.72	1.00

Mean item complexity = 1.1

Test of the hypothesis that 3 components are sufficient.

The root mean square of the residuals (RMSR) is 0.03
with the empirical chi square 9.62 with prob < 0.022

Fit based upon off diagonal values = 0.99

Interpretation

Notably, through the Varimax rotation, many results changed. When focusing on the matrix of factor loadings, we see a much clearer picture (RC now stands for rotated component). The variables x_1 to x_3 only possess high loadings on the first factor. The information explained in x_4 and x_5 is best summarized by the second factor, whereas the third factor mainly explains x_6 and x_7 . Note that the rotation did not change

anything in the communalities or the uniqueness values. However, in general, the explained information shifts from the first to the latter factors when comparing the unrotated to the rotated solution (row *Proportion Var*). The first two factors contain 62% of the original items' information. Furthermore, since the assignment of items to factors is now clearer, the item-specific complexity (*com*) values as well as the mean complexity is now approaching 1. Overall, the factor solution explains enough information of the original questions while being easy to interpret.

For completeness, let us make a case for the rotation matrix involved in the above Varimax rotation. Remember, rotation in factor analysis means multiplying a rotation matrix from the right to the unrotated matrix of factor loadings. Therefore, we are searching for the rotation matrix V that satisfies:

$$L' = L \times V$$

Here, L' stands for the rotated matrix of loadings while L is the unrotated matrix of factor loadings. We know L' and L from the results above.

```
L <- PCA_results_unrotated$loadings |> unclass()
L_rotated <- PCA_results_varimax$loadings |> unclass()
```

L

	PC1	PC2	PC3
x_1	0.7921634	-0.39808163	0.2531100
x_2	0.8142628	-0.36936331	0.2301943
x_3	0.8127470	-0.32629723	0.2555649
x_4	0.7501428	0.07826233	-0.5855445
x_5	0.6802674	0.13532182	-0.6649990
x_6	0.5552329	0.68795614	0.2477984
x_7	0.4851842	0.71841318	0.3264339

L_{rotated}

	RC1	RC3	RC2
x_1	0.9030880	0.16663668	0.08198102
x_2	0.8948374	0.20117132	0.10601309
x_3	0.8806297	0.18450185	0.15096776
x_4	0.2801160	0.90168908	0.14210380
x_5	0.1658047	0.93731104	0.13136293
x_6	0.1397769	0.17702581	0.88999751
x_7	0.1028291	0.07749895	0.91733332

In *R* we can solve a non-quadratic matrix equation by using the function `qr.solve()` (base *R*).

```
V <- qr.solve(a = L, b = L_rotated)

V
```

	RC1	RC3	RC2
PC1	0.7414698	0.5385021	0.4002975
PC2	-0.5391272	0.1229690	0.8331990
PC3	0.3994552	-0.8336031	0.3814989

V above is the rotation matrix that was used in the Varimax rotation. To complete the picture, we can recalculate the rotated loadings matrix L' by means of matrix multiplication:

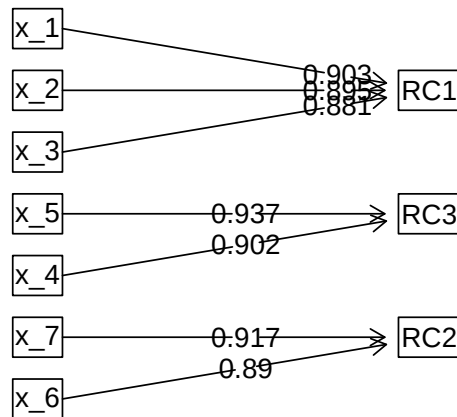
```
L%*%V
```

	RC1	RC3	RC2
x_1	0.9030880	0.16663668	0.08198102
x_2	0.8948374	0.20117132	0.10601309
x_3	0.8806297	0.18450185	0.15096776
x_4	0.2801160	0.90168908	0.14210380
x_5	0.1658047	0.93731104	0.13136293
x_6	0.1397769	0.17702581	0.88999751
x_7	0.1028291	0.07749895	0.91733332

The `psych` package also provides a graphical path diagram to guide market researchers in interpreting the final factor solution (function `fa.diagram()`). Usually, when interpreting the factor solution, researchers apply the rule of thumb that they blind out every factor loading (and cross-loading) that falls below the absolute value of 0.5. Therefore, we set `cut = 0.5` in the code chunk below.

```
psych::fa.diagram(PCA_results_varimax,
  digits = 3,
  simple = FALSE,
  cut = 0.5,
  sort = TRUE,
  adj = 3
)
```

Components Analysis



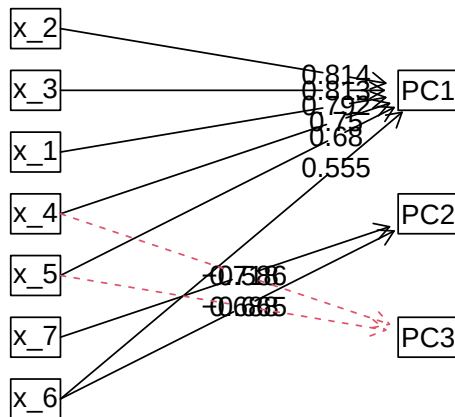
💡 Interpretation

The above path diagram reinforces the interpretation of the factor solutions. The first factor summarizes fans' satisfaction with the stadium, the second the satisfaction with the team, and the third the satisfaction with the ticket prices.

In more complex situation, these path diagrams come in very handy to identify unwanted cross-loadings between factors (red dashed lines below). To illustrate this, consider the path diagram for the unrotated solution.

```
psych::fa.diagram(PCA_results_unrotated,  
  digits = 3,  
  simple = FALSE,  
  cut = 0.50,  
  sort = TRUE,  
  adj = 3  
)
```

Components Analysis



The 50% cutoff criterion is a rough guideline. Keep in mind, that the factor loadings are Pearson correlations between an item and the corresponding factor. Therefore, in an old version of their text book, Hair et alii (2006, p. 128) came up with the idea to look at loadings (cross-loadings) only when they are significantly different from 0 at $\alpha = 0.05$ and $\beta = 0.20$ (i.e., a statistical power of 80%).

Below, we use the function `pwrss.z.corr()` function from the `pwrss` package (Bulus & Jentschke, 2026) to produce a table presenting the necessary sample sizes for different (absolute) loading sizes to be considered as significant:

```
necessary_n <- tibble::tibble(sample_size = numeric())

i <- 0.15

while (i <= 0.95) {
  temp <- pwrss::pwrss.z.corr(
    n = NULL,
    r = i,
    alpha = 0.05,
    power = 0.80,
    alternative = "not equal",
    verbose = FALSE
  )

  values <- list(loading_size = as.character(i), sample_size =
    round(temp$n, digits = 0))
}
```

```
necessary_n <- rbind(necessary_n, values)

  i <- i + 0.05
}

necessary_n
```

	loading_size	sample_size
1	0.15	347
2	0.2	194
3	0.25	124
4	0.3	85
5	0.35	62
6	0.4	47
7	0.45	37
8	0.5	30
9	0.55	24
10	0.6	20
11	0.65	17
12	0.7	14
13	0.75	12
14	0.8	10
15	0.85	8
16	0.9	7

Since our effective sample size is $n=195$, we could have also used **0.20** as absolute cutoff when interpreting factor loadings and cross-loadings. One sees that these guidelines are more beneficial when operating with relatively small sample size. This is because, with high sample size, also very low loadings get significant quickly.

Alternatively, we can apply the function *pwr.r.test()* from the [pwr](#) package ([Champely, 2020](#)) to calculate the actual threshold for significance given a specific sample size.

```
pwr::pwr.r.test(
  n = 195,
  r = NULL,
  sig.level = 0.05,
  power = 0.80
)
```

approximate correlation power calculation (arctangh transformation)

```
n = 195
r = 0.1990192
sig.level = 0.05
power = 0.8
alternative = two.sided
```

Step 5: Evaluating the goodness-of-fit of the factor solution

As we have discussed during lecture, we will focus on four sources of information to evaluate the factor solution:

1. The total variance explained
2. The congruence of the initial and reproduced Correlations
3. Variables' communality/ uniqueness
4. The measure of item complexity / total complexity

We have seen most of these things in our analyses above:

Principal Components Analysis

```
Call: psych::principal(r = tidyr::drop_na(Soccer_data_PCA_FA), nfactors = 3,
  rotate = "varimax")
```

Standardized loadings (pattern matrix) based upon correlation matrix

	RC1	RC3	RC2	h2	u2	com
x_1	0.90	0.17	0.08	0.85	0.150	1.1
x_2	0.89	0.20	0.11	0.85	0.148	1.1
x_3	0.88	0.18	0.15	0.83	0.168	1.1
x_4	0.28	0.90	0.14	0.91	0.088	1.2
x_5	0.17	0.94	0.13	0.92	0.077	1.1
x_6	0.14	0.18	0.89	0.84	0.157	1.1
x_7	0.10	0.08	0.92	0.86	0.142	1.0

	RC1	RC3	RC2
SS loadings	2.53	1.83	1.71
Proportion Var	0.36	0.26	0.24
Cumulative Var	0.36	0.62	0.87
Proportion Explained	0.42	0.30	0.28
Cumulative Proportion	0.42	0.72	1.00

Mean item complexity = 1.1

Test of the hypothesis that 3 components are sufficient.

The root mean square of the residuals (RMSR) is 0.03

with the empirical chi square 9.62 with prob < 0.022

Fit based upon off diagonal values = 0.99

The total variance explained is clearly above the threshold of 75% (see 0.87 in the row *Cumulative Var*). The variables' communality values (h^2) are clearly exceeding 0.50, and the item-specific complexities approach 1 (also the total complexity). Note, that these values are well aligned to the values we have discussed during lecture.

Let us focus on the correlation residuals (below, the function *lowerMat()* also comes from the [psych](#) package..

```
PCA_results_varimax$residual |> psych::lowerMat(digits = 3)
```

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	0.150						
x_2	-0.067	0.148					
x_3	-0.085	-0.080	0.168				
x_4	-0.020	-0.002	0.010	0.088			
x_5	0.021	-0.001	-0.010	-0.081	0.077		
x_6	0.017	-0.017	-0.003	-0.014	-0.003	0.157	
x_7	-0.009	0.019	-0.008	0.012	0.004	-0.149	0.142

The output shows us the lower off-diagonal matrix of residuals between the original correlations and the reproduced residuals. Note, that the complete matrix will be symmetric. Thus, we only have to focus on the lower part. The diagonal elements, nonetheless, present the item-specific uniqueness values again. We now have to count all off-diagonal elements that exceed an absolute value of 0.05, to see whether more than 50% are doing so.

```
residuals <- PCA_results_varimax$residual |>  
psych::lowerMat(digits = 3)
```

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	0.150						
x_2	-0.067	0.148					
x_3	-0.085	-0.080	0.168				
x_4	-0.020	-0.002	0.010	0.088			
x_5	0.021	-0.001	-0.010	-0.081	0.077		
x_6	0.017	-0.017	-0.003	-0.014	-0.003	0.157	
x_7	-0.009	0.019	-0.008	0.012	0.004	-0.149	0.142

```
sum(residuals > 0.05 | residuals < -0.05)/length(residuals)
```

```
[1] 0.2380952
```

Interpretation

We see that fewer than 50% of the correlation residuals have an absolute value of higher than 0.05. The original correlations are reproduced with sufficient precision.

Step 6: Computing the factor scores

Computing factor scores is very easy. Actually, we just have them calculated when extracting the factor solution. The scores are saved with the results:

```
PCA_results_varimax$scores |> head()
```

	RC1	RC3	RC2
[1,]	-0.339646371	0.55154286	0.45810123
[2,]	0.847307756	0.61906876	-0.04454389
[3,]	-1.189578985	-1.46718026	0.89315875
[4,]	0.835139521	0.08730112	-0.35566928
[5,]	-0.001201816	0.34022781	-0.67542413
[6,]	0.732337861	0.04429829	0.32984885

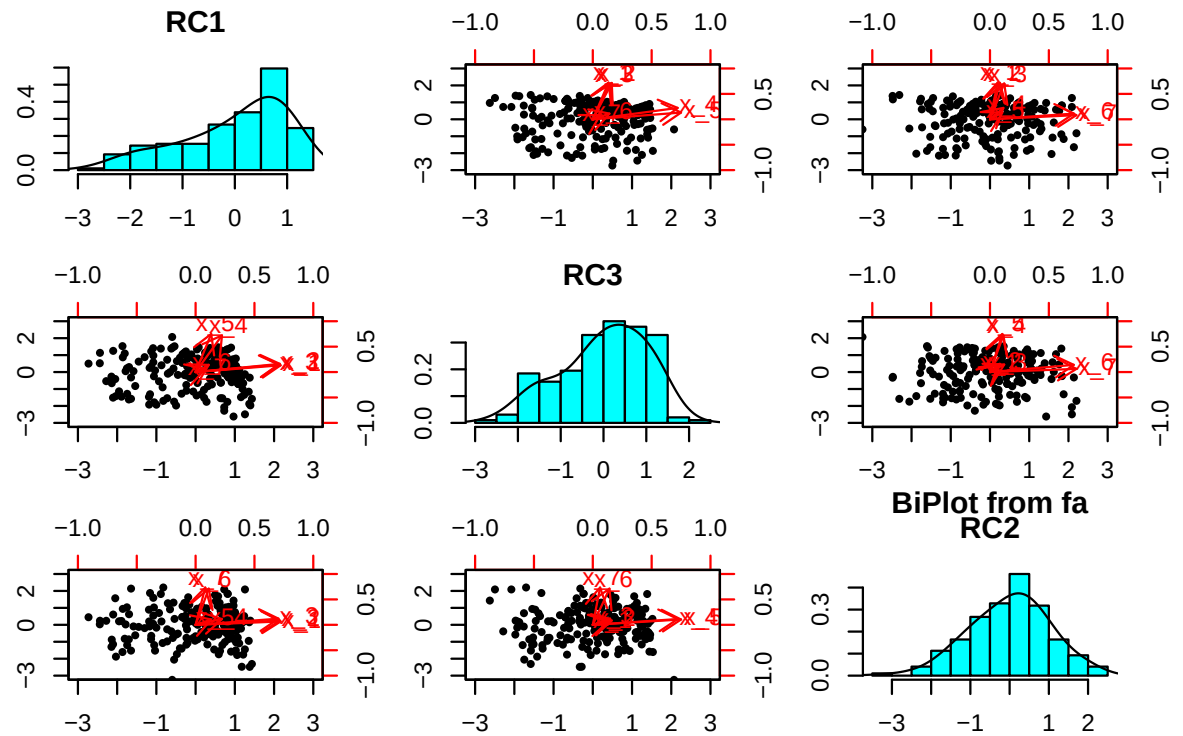
Interpretation

The factor scores in this example are standardized. Thus, they have a mean of 0 and a SD of 1. The interpretation for the above output is: Fan 1 has a higher-than-average score for factors 2 [team] and 3 [price] (i.e., a factor score of > 0) and lower-than-average score for factor 1 [Satisfaction with the stadium] (i.e., a factor score of < 0).

Keep in mind, in factor analysis there exist several methods on how to calculate factor scores. However, this goes beyond the scope of our introduction.

As noted during the lecture, often market researchers are using the factor scores to visualize the cases (fans in this case) in low dimensional space. We will do so next by using `psych`'s `biplot.psych()` function.

```
PCA_results_varimax |> psych::biPlot(x = _)
```

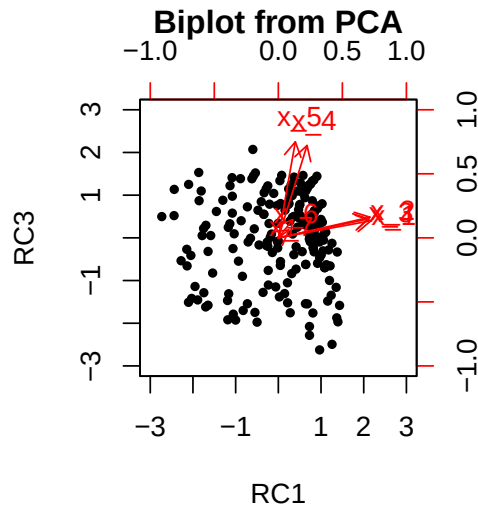


💡 Interpretation

Since we have a complex factor solution with 3 factors, the resulting graphics are rather complex. The histograms summarize the distribution of factor scores across all consumers. For example, the histogram in the upper-left position indicates that many consumers possess above average satisfaction with the stadium (factor 1), whereas others are extremely dissatisfied with this component. The plot in the lower-left position portrays the biplot between the factor scores of factors 1 (stadium, abscissa) and 2 (team, ordinate). Each point represents one case in the analysis. The red vectors indicate the variables. For example, one could interpret that few consumers are extremely satisfied by the stadium (would be high values on the abscissa), independently of their satisfaction level with the team (ordinate).

To get a clearer picture, we could also only ask for the middle-left plot.

```
PCA_results_varimax |> psych::biPlot(x = _,
                                     main="Biplot from PCA",
                                     choose = c("RC1", "RC3"))
```

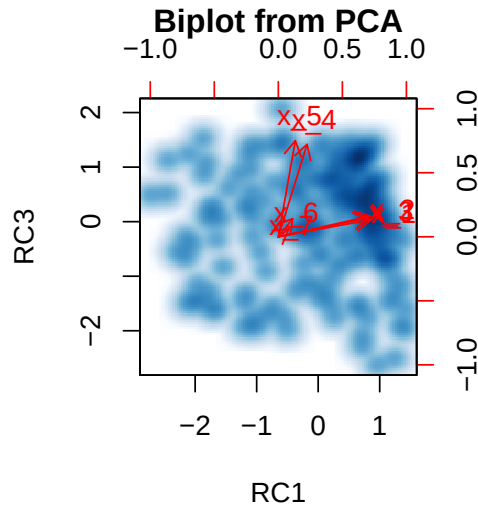


💡 Interpretation

With this bigger picture, we could argue that the variables x_4 and x_5 are relatively well explained by the rotated components RC1 and RC3 as their absolute loadings are pronounced (equivalent to saying that their distance from 0 is relatively high).

If the number of consumers in a analysis gets higher, it is often useful to switch to a heatmap-like visualization that helps to assess the mass of consumers in the plot.

```
PCA_results_varimax |> psych::biPlot(x = _, smoother = TRUE,
                                     main="Biplot from PCA",
                                     choose = c("RC1", "RC3"))
```



Basic Cluster Analysis

In the following section, we start reproducing the cluster example from the lecture within R . The example is modified from Table 9.1 of Chapter 9 in Sarstedt & Mooi (2019).

Two variables measured for seven customers on a scale from 0 to 100 with higher values denoting a higher degree of price consciousness and brand loyalty..

Question: How can we segment a market based on customers' price consciousness (x) and brand loyalty (y)?

In total, we will go through the 5 steps that we have discussed during the lecture:

1. Selecting the clustering variables (already done)
2. Selecting the clustering procedure (we will apply hierarchical clustering and partitioning k-means for illustration)
3. Conducting clustering methods
4. Deciding on the number of clusters to retain
5. Validating and interpreting the cluster solution

However, we will only focus on the things that incorporate applications in *R*. Therefore, we will not repeat rules-of-thumb (for example, regarding sample sizes, etc.).

First, we will establish the data in *R*.

Creating the data in R and doing some basic data analyses to recap lecture content

```
Brand_Loyalty_Price_Consciousness <- tibble::tibble(  
  customer = LETTERS[1:7],  
  x = c(33, 82, 66, 30, 79, 50, 10),  
  y = c(95, 94, 80, 67, 60, 33, 17)  
)  
  
labelled::var_label(Brand_Loyalty_Price_Consciousness) <- list(  
  customer = "Customer",  
  x = "Price consciousness",  
  y = "Brand loyalty"  
)  
  
print(Brand_Loyalty_Price_Consciousness)
```

```
# A tibble: 7 x 3  
  customer      x      y  
  <chr>    <dbl> <dbl>  
1 A          33     95  
2 B          82     94  
3 C          66     80  
4 D          30     67  
5 E          79     60  
6 F          50     33  
7 G          10     17
```

Keep in mind, that we usually would evaluate, whether our clustering variables show high (≥ 0.90) pair-wise correlations.

```
Brand_Loyalty_Price_Consciousness |> dplyr::select(x, y) |> cor()
```

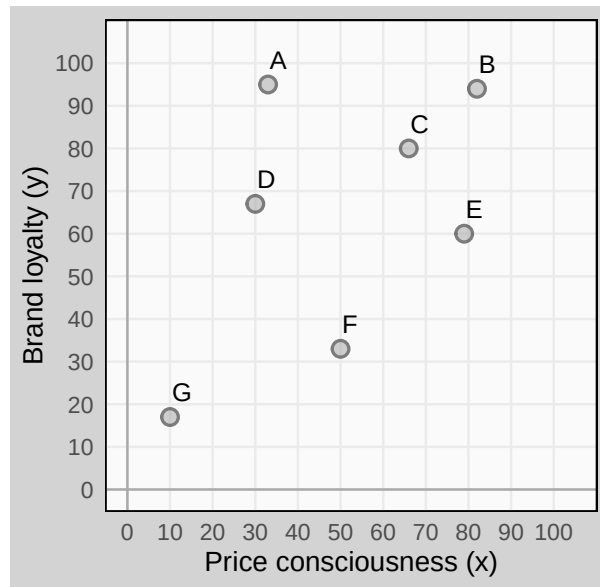
```
      x      y  
x 1.0000000 0.4944447  
y 0.4944447 1.0000000
```

💡 Interpretation

In the present case, we have a somewhat “moderate” correlation ($r=0.49$) between both clustering variables (Cohen, 1988. p. 79). Usually, we handle more than two variables. In this case, we can use `ggstatsplot::ggcorrmat()` to get an impression of the pair-wise correlations (see, Section).

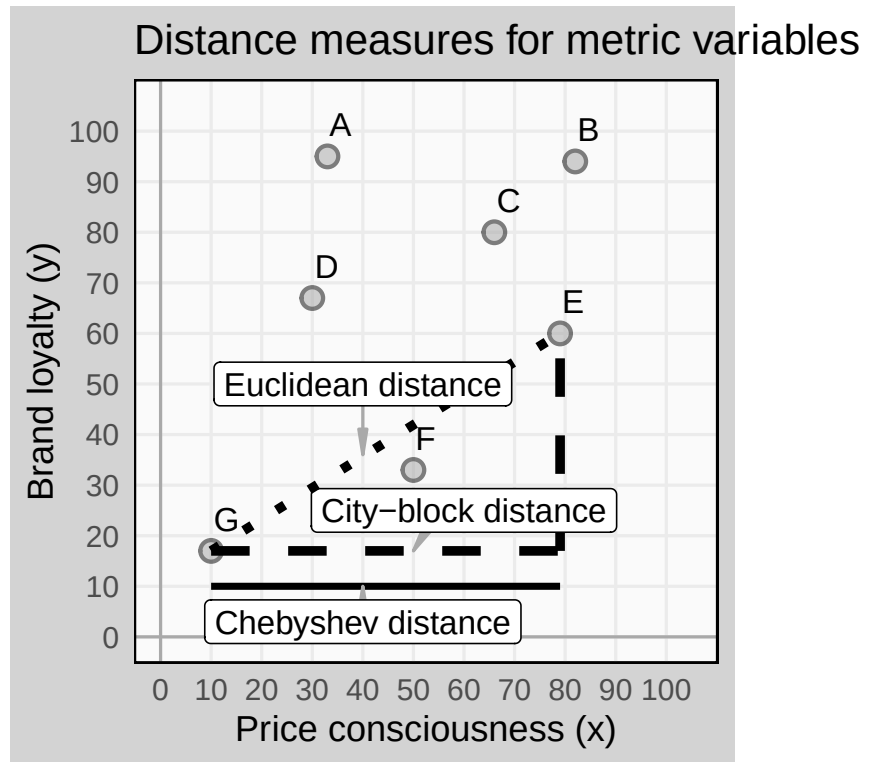
With only two variables in the example, it is easy to portray the data in a two-dimensional scatterplot

```
scatterplot_basic_cluster_analysis <-  
ggplot2::ggplot(Brand_Loyalty_Price_Consciousness, aes(x = x, y =  
y)) +  
  ggplot2::geom_point(alpha = 0.5, shape = 21, size = 2.7, colour =  
= "black", fill = "#A4A4A4", stroke = 1) +  
  ggplot2::geom_hline(yintercept = 0, col = "darkgrey") +  
  ggplot2::geom_vline(xintercept = 0, col = "darkgrey") +  
  ggplot2::labs(y = "Brand loyalty (y)", x = "Price consciousness  
(x)") +  
  ggplot2::coord_fixed() +  
  ggplot2::theme_minimal() +  
  ggplot2::scale_x_continuous(breaks = seq(0, 100, 10), limits =  
c(0, 105), minor_breaks = NULL) +  
  ggplot2::scale_y_continuous(breaks = seq(0, 100, 10), limits =  
c(0, 105), minor_breaks = NULL) +  
  ggplot2::geom_text(aes(label = customer), hjust = -0.1, vjust =  
-0.85) +  
  ggplot2::theme(  
    plot.background = element_rect(fill = "lightgrey"),  
    panel.background = element_rect(fill = "grey98"),  
    panel.border = element_rect(colour = "black", fill = NA,  
    linewidth = 0.7),  
    text = element_text(size = 12)  
  )  
  
scatterplot_basic_cluster_analysis
```



Keep in mind, that with the graphical representation above, it is also easy to identify customer **G** as a multivariate outlier. Likewise, it appears as if **B**, **C**, and **E** form a relatively homogeneous group, while **A** and **D** represent another one.

We first will calculate the distance matrix between the seven customers. Think back to what we have learned during lecture. There are several distance measures available.



We will use the **Euclidean distance** (L_2 -norm) for illustration purpose, which is defined between two objects A and B and k clustering variables as

$$\sqrt{\sum_{i=1}^k (A_i - B_i)^2}$$

(A_i represents A's coordinates at the i^{th} variable).

For this purpose, we use the function `dist()` which comes with base *R*. Since any distance matrix is symmetric, let us focus only on lower-diagonal matrix (below, the function `lowerMat()` comes from the [psych](#) package).

```
distance_matrix <- Brand_Loyalty_Price_Consciousness |>
  dplyr::select(x, y) |>
  dist(method = "euclidean") |>
  as.matrix()

rownames(distance_matrix) <-
Brand_Loyalty_Price_Consciousness$customer
colnames(distance_matrix) <-
Brand_Loyalty_Price_Consciousness$customer
```

```
distance_matrix |> psych::lowerMat(digits = 2)
```

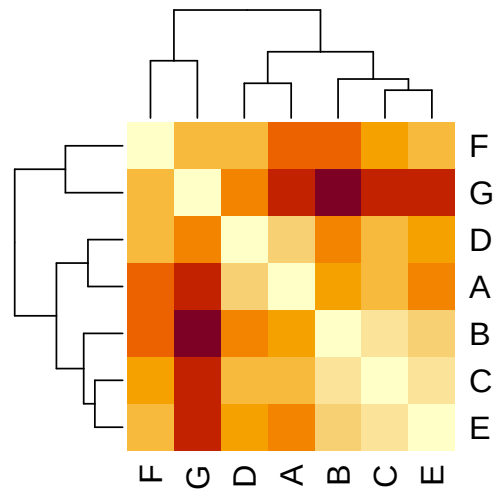
	A	B	C	D	E	F	G
A	0.00						
B	49.01	0.00					
C	36.25	21.26	0.00				
D	28.16	58.59	38.28	0.00			
E	57.80	34.13	23.85	49.50	0.00		
F	64.29	68.88	49.65	39.45	39.62	0.00	
G	81.32	105.42	84.29	53.85	81.30	43.08	0.00

💡 Interpretation

As with the graphical representation above, we see that **G** has a relatively high distance to all other objects.

Often, market researchers use data visualization techniques for better insights into distance/ similarity matrices when research problems become bigger. Below, we use the *heatmap()* function which comes with base *R* to do so.

```
distance_matrix |> heatmap(scale="none", revC=TRUE)
```



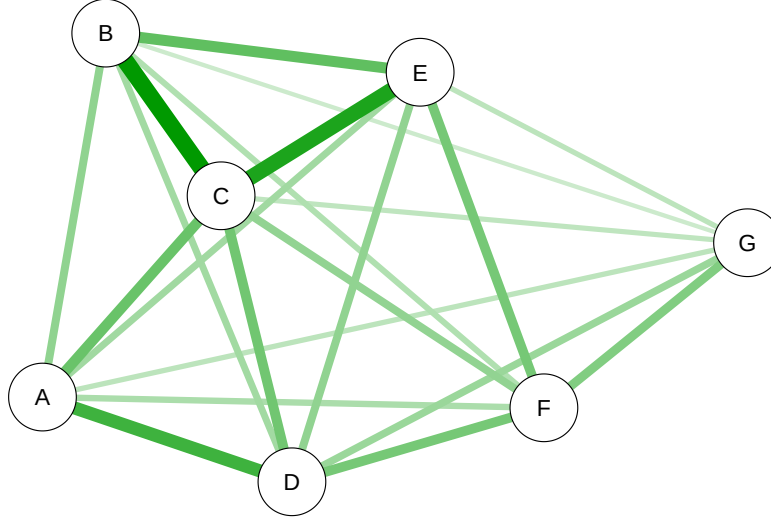
💡 Interpretation

With the heatmap representation, understanding the dissimilarity between the seven customers becomes straightforward. The more the color shifts towards red, the more distant the two objects are from each other (e.g., B and G). Conversely, the more yellowish the color, the higher the similarity (proximity in feature space). It's worth noting that the function automatically arranges columns and rows based on the Euclidean distance. This arrangement groups similar objects together, making it easy to identify patterns. For instance, we can quickly see that B, C, and E form a region of high similarity in the lower-right part of the matrix. A and D, on the other hand, form a separate region. The graphic also includes dendrograms for a corresponding clustering of data, a topic we will discuss in detail later. By default, *heatmap()* uses complete linkage as a hierarchical clustering method.

Another approach to visualize distances between cases in cluster analysis is to use network graphs. Below we use the function *qgraph()* which the [qgraph](#) package (Epskamp, Cramer, Waldorp, Schmittmann, & Borsboom, 2012) provides.

We first convert the distance matrix into a similarity matrix. Keep in mind that $similarity = \frac{1}{distance}$, since the similarity of two objects is the reciprocal of their distance/dissimilarity. By doing so, we accept that each object has an infinite similarity with itself.

```
similarity_matrix <- 1/distance_matrix  
  
similarity_matrix |> qgraph::qgraph(layout="spring")
```



💡 Interpretation

In the network graph above, objects that are more similar to each other (e.g., B, C, and E) as compared to others (e.g., A and G) are placed in a narrow distance. Similarity between customers is represented by the thickness and the shade of green of connection lines. This kind of visualization is particularly useful with a manageable number of cases but a high number of clustering variables.

For sake of completeness, we subsequently will also use the **City-block distance/ Manhattan metric**, and the **Chebychev distance** for compiling the distance matrix. Note, that there exist also ample other measures (e.g., [Kasambara, 2017](#); [Lichters, Möslin, Sarstedt, & Scharf, 2021](#)).

The **City-block distance/ Manhattan metric** is defined between two objects A and B and k clustering variables as

$$\sum_{i=1}^k |A_i - B_i|$$

(A_i represents A's coordinates at the i^{th} variable).

The largest distance of two objects A and B on any clustering variable k defines to **Chebychev distance**:

$$\max_{i=1}^k |A_i - B_i|$$

Let us start with the **City-block distance/ Manhattan metric**.

```

distance_matrix <- Brand_Loyalty_Price_Consciousness |>
  dplyr::select(x, y) |>
  dist(method="manhattan") |>
  as.matrix()

rownames(distance_matrix) <-
Brand_Loyalty_Price_Consciousness$customer
colnames(distance_matrix) <-
Brand_Loyalty_Price_Consciousness$customer

distance_matrix |> psych::lowerMat(digits = 2)

```

	A	B	C	D	E	F	G
A	0						
B	50	0					
C	48	30	0				
D	31	79	49	0			
E	81	37	33	56	0		
F	79	93	63	54	56	0	
G	101	149	119	70	112	56	0

Then, the **Chebychev distance**.

```

distance_matrix <- Brand_Loyalty_Price_Consciousness |>
  dplyr::select(x, y) |>
  dist(method="maximum") |>
  as.matrix()

rownames(distance_matrix) <-
Brand_Loyalty_Price_Consciousness$customer
colnames(distance_matrix) <-
Brand_Loyalty_Price_Consciousness$customer

distance_matrix |> psych::lowerMat(digits = 2)

```

	A	B	C	D	E	F	G
A	0						
B	49	0					
C	33	16	0				
D	28	52	36	0			
E	46	34	20	49	0		
F	62	61	47	34	29	0	
G	78	77	63	50	69	40	0

Conducting Hierarchical Cluster Analysis

Single Linkage based on Euclidean distance

We only will illustrate the application of **Single Linkage** algorithm based **Euclidean distance**.

There are several good packages in *R* that help with conducting hierarchical cluster analysis. For the subsequent clustering, we are staying with a basic variant which the *hclust()* function provides ([stats](#) package ([2026a](#))). We first recreate the distance matrix again. Since we have already seen the visualizations of the distance matrix above, we will not repeat this steps here.

```
distance_matrix_euclidean <- Brand_Loyalty_Price_Consciousness |>
  dplyr::select(x, y) |>
  dist(method="euclidean")
```

Next, we apply the *hclust()* function to the distance matrix and subsequently set the labels (i.e., the customers' names in our example).

```
h_c_euclidean_single_linkage <-
hclust(d=distance_matrix_euclidean, method="single")

h_c_euclidean_single_linkage$labels <-
Brand_Loyalty_Price_Consciousness$customer
```

Decide on the number of clusters

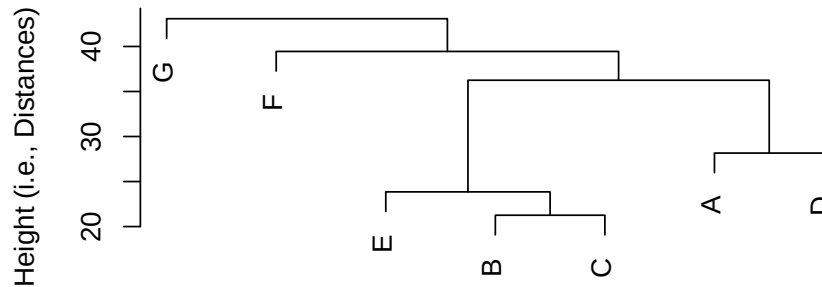
i Note

Keep in mind that one should never rely only on statistical considerations when making the final decision on clusters. As discussed during lecture, often expert judgment has a valid stake in the final decision.

As seen during the lecture, in hierarchical clustering it is particularly useful to look at the corresponding dendrogram to select the final number of clusters to retain.

```
plot(h_c_euclidean_single_linkage, ylab="Height (i.e.,
Distances)")
```

Cluster Dendrogram



```
distance_matrix_euclidean
hclust (*, "single")
```

💡 Interpretation

In the dendrogram we can understand the step-wise approach of clustering. Specifically, we see that first **B** and **C** are merged. Afterward, this new cluster fusions with **E** to **{B, C, E}**. Then, **{A, D}** is created. Both clusters are next merged to **{A, B, C, D, E}**, before **F** and finally **G** are joining. Most interestingly, we see that the main increase in distance (i.e., the main increase in within-cluster heterogeneity) occurs when **{B, C, E}** is merged with **{A, D}**. Thus, we should stop before this step, leaving the final 4-cluster solution comprising **{A, D}**, **{B, C, E}**, **F**, and **G**. As **F** and **G** are non merged singular observations in the final solution, one could also think of them as **multivariate outliers** (recall that Single Linkage is often used because of its chaining tendencies to detect such outliers).

The cluster assignments for each observation are given by:

```
cutree(h_c_euclidean_single_linkage, k=4)
```

```
A B C D E F G
1 2 2 1 2 3 4
```

We can also add the assignments to the original data:

```
Brand_Loyalty_Price_Consciousness$h_c_single_linkage <-
cutree(h_c_euclidean_single_linkage, k=4)
```

```
Brand_Loyalty_Price_Consciousness
```

```
# A tibble: 7 x 4
  customer      x      y h_c_single_linkage
  <chr>    <dbl> <dbl>          <int>
1 A         33    95              1
2 B         82    94              2
3 C         66    80              2
4 D         30    67              1
5 E         79    60              2
6 F         50    33              3
7 G         10    17              4
```

Usually, with larger datasets, one looks at the number of observations in each segment as well as the cluster profiles regarding the clustering variables.

```
Brand_Loyalty_Price_Consciousness |>
  dplyr::group_by(h_c_single_linkage) |>
  dplyr::summarise(
    price_consciousness = mean(x),
    brand_loyalty = mean(y),
    n = n()
  )
```

```
# A tibble: 4 x 4
  h_c_single_linkage price_consciousness brand_loyalty      n
          <int>             <dbl>          <dbl> <int>
1             1             31.5             81      2
2             2             75.7             78      3
3             3              50              33      1
4             4              10              17      1
```

Other packages enable for a more appealing visualization of dendrograms. To illustrate this, we use functions included in the package [dendextend](#) package (Galili, 2015).

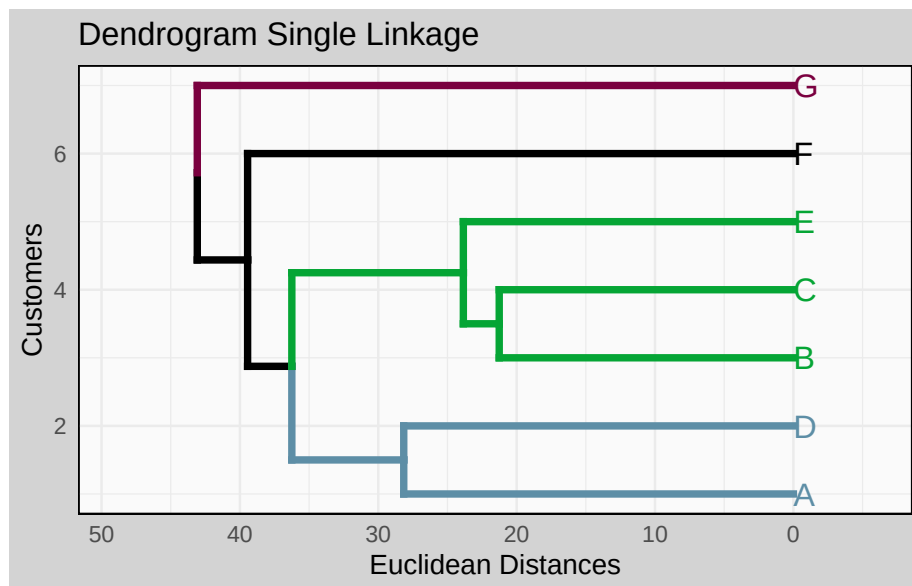
```
dendrogram <- h_c_euclidean_single_linkage |>
  as.dendrogram() |>
  sort() |> # sort dendrogram
  dendextend::set("labels_colors", value = c("#5d8ea6",
"#05a535", "black", "#7a003f"), k = 4) |> # set color for labels
```

```

dendextend::set("branches_k_color", value = c("#5d8ea6",
"#05a535", "black", "#7a003f"), k = 4) |> # set color of lines in
dendrogram
dendextend::set("branches_lwd", 1.5) |> # set width of lines
dendextend::as.ggdend(horiz = TRUE, theme = theme_minimal()) ##
convert to ggplot compatibility

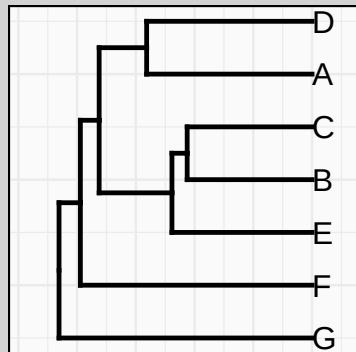
dendrogram |> ggplot2::ggplot(horiz = TRUE, theme =
theme_minimal()) +
  ggplot2::ggtitle("Dendrogram Single Linkage") +
  ggplot2::xlab("Customers") +
  ggplot2::ylab("Euclidean Distances") +
  ggplot2::theme(
    plot.background = element_rect(fill = "lightgrey"),
    panel.background = element_rect(fill = "grey98"),
    panel.border = element_rect(colour = "black", fill = NA,
    linewidth = 0.7),
    text = element_text(size = 12))

```

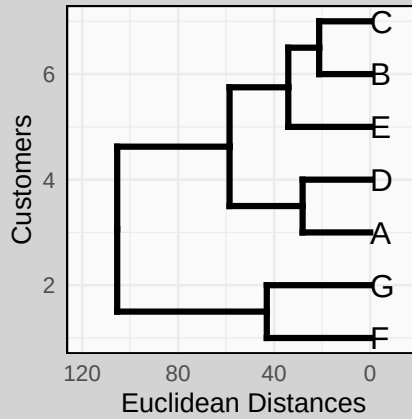


For sake of completeness, we spot the different dendrograms, which result from different fusion algorithms (but always Euclidean distance) below.

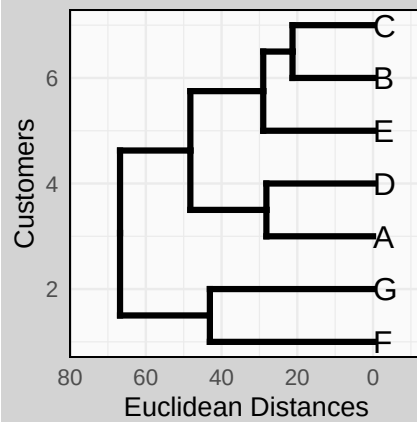
Dendrogram Single Linkage



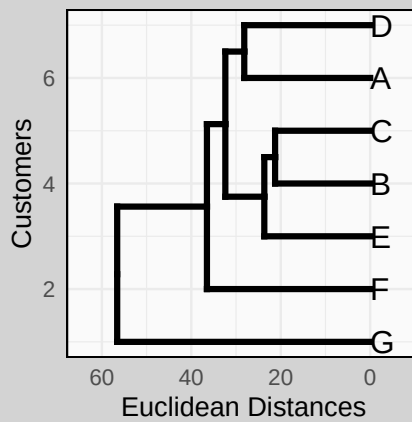
Dendrogram Complete Linkage



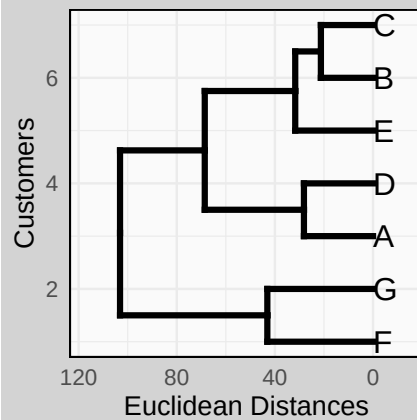
Dendrogram Average Linkage



Dendrogram Centroid Linkage



Dendrogram Ward's Linkage



💡 Interpretation

Above, we see that only **Single Linkage** and **Centroid Linkage** are able to detect **F** and **G** as multivariate outliers.

Another way of guiding the selection of the final number of clusters is to look at the invariance of the cluster solution (i.e., the assignment of cluster labels to the observations) with regard to different fusion algorithms or/and distance measures. As a rule of thumb, less than 20% of cluster assignments should vary across different reasonable clustering approaches (Sarstedt & Mooi, 2019, p. 332). Note, that only reasonable clustering approaches should be evaluated. For example, it makes no sense to switch to matching coefficients as similarity/distance measures if one handles clustering variables that are all interval-scaled.

In the following, we stay with the Euclidean distance, but evaluate **Ward's Linkage** (Ward, 1963) and **Average Linkage** next to the already implemented **Single Linkage**.

```
#Distance matrix
distance_matrix_euclidean <- Brand_Loyalty_Price_Consciousness |>
  dplyr::select(x, y) |>
  dist(method = "euclidean")

# Single Linkage
single_linkage <- hclust(d=distance_matrix_euclidean,
  method="single")
single_linkage$labels <-
Brand_Loyalty_Price_Consciousness$customer

# Average Linkage
average_linkage <- hclust(d=distance_matrix_euclidean,
  method="average")
average_linkage$labels <-
Brand_Loyalty_Price_Consciousness$customer

# Ward's Linkage
ward_linkage <- hclust(d=distance_matrix_euclidean,
  method="ward.D2")
ward_linkage$labels <- Brand_Loyalty_Price_Consciousness$customer
```

After having conducted these segmentation, we save the segment assignments for solutions with 3 to 5 clusters separately.

```

assignments <- tibble::tibble(.rows =
  nrow(Brand_Loyalty_Price_Consciousness))

assignments$Single_Linkage_3 <- cutree(single_linkage, k=3)
assignments$Single_Linkage_4 <- cutree(single_linkage, k=4)
assignments$Single_Linkage_5 <- cutree(single_linkage, k=5)

assignments$Average_Linkage_3 <- cutree(average_linkage, k=3)
assignments$Average_Linkage_4 <- cutree(average_linkage, k=4)
assignments$Average_Linkage_5 <- cutree(average_linkage, k=5)

assignments$Ward_Linkage_3 <- cutree(ward_linkage, k=3)
assignments$Ward_Linkage_4 <- cutree(ward_linkage, k=4)
assignments$Ward_Linkage_5 <- cutree(ward_linkage, k=5)

```

Doing so is helpful to understand how stable the Single Linkage solution is at varying numbers of clusters. The next table shows the assignments in case of 3-cluster solutions.

```

assignments |>
  dplyr::select(Single_Linkage_3, Average_Linkage_3,
    Ward_Linkage_3)

```

```

# A tibble: 7 x 3
  Single_Linkage_3 Average_Linkage_3 Ward_Linkage_3
      <int>          <int>          <int>
1             1             1             1
2             1             2             2
3             1             2             2
4             1             1             1
5             1             2             2
6             2             3             3
7             3             3             3

```

```

assignments |>
  dplyr::group_by(Single_Linkage_3, Average_Linkage_3) |>
  dplyr::summarise(
    n = n(),
    percent = n()/nrow(assignments)*100
  )

```

```

# A tibble: 4 x 4
# Groups:   Single_Linkage_3 [3]

```

	Single_Linkage_3	Average_Linkage_3	n	percent
	<int>	<int>	<int>	<dbl>
1	1	1	2	28.6
2	1	2	3	42.9
3	2	3	1	14.3
4	3	3	1	14.3

We see that for the solution with 3 clusters, we only have a limited concordance of cluster assignments between Single and other linkage algorithms. In contrast, the concordance between **Average Linkage** and **Ward** is perfect. This small example provides an essential learning. First, the less than 20 % discordance rule mentioned above is misleading when multivariate outliers are present, and the fusion algorithms under usage are sensitive to outliers to varying degrees. We have $42.9\% + 14.3\% = 57.2\%$ mismatch between the clustering variants in the above example. This is because fusion algorithms with chaining properties (such as **Single Linkage**) will seldom match the outcomes of other algorithms disturbed by outliers, when the number of clusters is low. At the same time, with multivariate outliers present, we should not trust the concordance of two fusion algorithms that are sensitive to outliers. These outliers affect the final cluster assignments so strongly that two fusion algorithms sensitive to outliers will almost always end in the same assignments. Second, the only proper way in the presence of multivariate outliers is to exclude them before clustering the data and then evaluate the concordance of varying solutions. Alternatively, one could collect the outliers in a kind of noise cluster (Vigneau, Qannari, Navez, & Cottet, 2016), as it is done somehow by the final steps in **Single Linkage**.

Often, market research also are preparing cross tabs of competing cluster solutions. We will do so below with the help of the *CrossTable()* function provided by the *descr* package (Aquino, Enzmann, Schwartz, Jain, & Kraft, 2026).

```
descr::CrossTable(x = assignments$Single_Linkage_3,
y = assignments$Average_Linkage_3,
digits = 2,
prop.r = FALSE,
prop.c = FALSE,
prop.t = TRUE,
prop.chisq = FALSE,
dnn = c("Single Linkage", "Average Linkage"))
```

Cell Contents	

	N
N / Table Total	

Single Linkage	Average Linkage			Total
	1	2	3	
1	2	3	0	5
	0.29	0.43	0.00	
2	0	0	1	1
	0.00	0.00	0.14	
3	0	0	1	1
	0.00	0.00	0.14	
Total	2	3	2	7

For illustration, we next focus on the comparison between 3 to 5 clusters and **Single Linkage** and **Ward**.

```
assignments |>
  dplyr::group_by(Single_Linkage_3, Ward_Linkage_3) |>
  dplyr::summarise(
    n = n(),
    percent = n()/nrow(assignments)*100
  )
```

```
# A tibble: 4 x 4
# Groups:   Single_Linkage_3 [3]
  Single_Linkage_3 Ward_Linkage_3      n percent
      <int>         <int> <int>   <dbl>
1           1             1     2    28.6
2           1             2     3    42.9
3           2             3     1    14.3
4           3             3     1    14.3
```

```
assignments |>
  dplyr::group_by(Single_Linkage_4, Ward_Linkage_4) |>
  dplyr::summarise(
    n = n(),
    percent = n()/nrow(assignments)*100
  )
```

```
# A tibble: 4 x 4
# Groups:   Single_Linkage_4 [4]
```

	Single_Linkage_4	Ward_Linkage_4	n	percent
	<int>	<int>	<int>	<dbl>
1	1	1	2	28.6
2	2	2	3	42.9
3	3	3	1	14.3
4	4	4	1	14.3

```
assignments |>
  dplyr::group_by(Single_Linkage_5, Ward_Linkage_5) |>
  dplyr::summarise(
    n = n(),
    percent = n()/nrow(assignments)*100
  )
```

```
# A tibble: 6 x 4
```

```
# Groups:   Single_Linkage_5 [5]
```

	Single_Linkage_5	Ward_Linkage_5	n	percent
	<int>	<int>	<int>	<dbl>
1	1	1	1	14.3
2	2	2	2	28.6
3	2	3	1	14.3
4	3	1	1	14.3
5	4	4	1	14.3
6	5	5	1	14.3

Interpretation

In short, the discordance between both fusion algorithms in case of 3 clusters is 57.2%. In case of 4 clusters it shrinks to 0%. Finally, with 5 clusters the mismatch is 28.6 %. Thus, according to this stability assessment we would also favor the 4 cluster solution.

Market researchers also often apply cluster quality indices to decide on an appropriate number of clusters in the final solution. Researchers, came up with many different of these criteria - all have their merits and drawbacks. As a consequence, many good *R* packages implement a plethora of the criteria. Examples include the **NbClust** (Charrad, Ghazzali, Boiteau, & Niknafs, 2014), the **fpc** (Hennig, 2026), and the **clusterCrit** (Desgraupes, 2023) package.

During the lecture, we focused on two classes of criteria: (a) the **Variance Ratio Criterion** proposed by Calinski & Harabasz (1974), and (b) the **Duda-Hart-index** (Duda & Hart, 1973). We will use both classes subsequently, because both worked well in the seminal work of Milligan & Cooper (1985).

We start with the **Variance Ratio Criterion** (VRC) proposed by Calinski & Harabasz (1974). Remember from the lecture that the calculation of the VRC is similar to computations we have seen in ANOVA:

$$VRC_k = \frac{\frac{SS_B}{k-1}}{\frac{SS_W}{n-k}}$$

where n is the total sample size and k the number of clusters in a solution. SS_B indicated the between-cluster sum of squares and SS_W the within-group sum of squares. Usually, we are searching for the k that maximizes VRC, since at this point the variation regarding the clustering variables between the groups is high (i.e., clusters are well separated), while the variation within the groups is small (i.e., the clusters are very compact). However, as discussed during lecture, VRC tend to decrease with increasing number of clusters. Therefore, a more nuanced approach is to use the minimal Omega, defined as searching for the minimal the differences in VRC values for successive numbers of clusters:

$$\omega_k = (VRC_{k+1} - VRC_k) - (VRC_k - VRC_{k-1})$$

Clearly, ω_k is not defined for $k < 3$ and $k > n-2$. Why? Because ω_k , in case of 2 clusters, would ask for a calculation of VRC_1 . However, VRC_1 is not defined, since we do not have a SS_B in a 1-cluster solution. Similarly, in case of $k=n-1$, ω_k would ask for a calculation of VRC_n . However, the term $\frac{SS_W}{n-k}$ in the VRC calculation above is not defined for $n=k$.

💡 Further remarks

Please also note: applying the **VRC** and the **Omega** criterion does not make much sense when working the fusion algorithms with chaining properties in the presence of multivariate outliers. This is because such algorithms (e.g., Single Linkage) tend to produce clusters with only one observation per cluster. Thus, the corresponding SS_W -components contributed by these clusters in a solution will be 0, increasing the VRC artificially. Both criteria are also only useful when applied to fusion algorithms where the sum of squares has a valid interpretation. For example, we can't use them when we have created the cluster in a way that only considered categorical clustering variables, since squared differences between them have no meaningful interpretation.

Below, we start with the VRC criterion. First, we use the function *NbClust()* from the package with the same name ([Charrad et al., 2014](#)). This function asks us to specify a set of arguments as parameters.

- **data**: is the dataset providing the clustering variables (e.g., a tibble)
- **diss**: is the distance matrix that we might have calculated before
- **distance**: set to *NULL*, because we already have a distance matrix. Otherwise, we would have had specified a distance measure
- **min.nc** & **max.nc**: minimal and maximal number of clusters to consider
- **method**: a fusion algorithm (Ward's Linkage in the present case)

- **index**: a cluster quality index. Below, “*ch*” stands for Calinski & Harabasz (set to “*all*” to calculate all available indexes)

```
clustering_vars <- Brand_Loyalty_Price_Consciousness |>
dplyr::select(x, y)

suggestion_CH <- NbClust::NbClust(data = clustering_vars,
  diss = distance_matrix_euclidean,
  distance = NULL,
  min.nc = 1,
  max.nc = 6,
  method = "single",
  index = "ch")[1:2]

suggestion_CH
```

```
$All.index
  1      2      3      4      5      6
NA 4.1559 3.5922 7.6400 6.1707 8.4030
```

```
$Best.nc
Number_clusters  Value_Index
          6.000           8.403
```

Interpretation

If we would follow the **VRC**, than we would select six clusters in the final solution as the VRC has the maximum value 8.4. However, based on the above-dendrograms, we know that this is mainly driven by the multivariate outliers **G** and **F**. Calinski & Harabasz (1974, p. 12) knew about this potential problem with the application of the **VRC**. Therefore, in this situation, they advocated against the use of the globally maximum, and for the use of the first local maximum that usually occurs for smaller *k*. Would be 4 clusters in the present case.

Next, we will apply the omega criterion instead. Unfortunately, when compiling this script, I was not able to find a convenient package that provides a function for calculating omega (if you know one, just give me an email). Therefore, I have written one on my own, which our chair’s page lists. We use the following code to load the script providing the function named *Calinski_Harabasz_omega*.

```
url_of_file <-
  "https://marketing.ovgu.de/marketing_media/Downloads/MMA/cluster_functions_mma.R"
```

```
source(url(url_of_file))
```

If anything does not work out as expected on your machine, you can download the data by clicking on the following button.

Below, we apply the function. Keep in mind, that for $k > n-2$ omega is undefined. Therefore, we set max_k to 5 (with 6, the function will throw an error)..

```
Calinski_Harabasz_omega(max_k = 5,
                        distance_matrix =
                        distance_matrix_euclidean,
                        hierarchical_cluster_solution =
                        single_linkage)
```

	clusters	VRC	VRC_omega
1	2	4.155899	NA
2	3	3.592170	4.611549
3	4	7.639990	-5.517113
4	5	6.170697	3.701631
5	6	8.403034	NA

Interpretation

As discussed during the lecture, we are searching for the global minimum of omega to determine the optimal k. Doing so, would suggest retaining 4 clusters in the final solution.

We proceed with the **Duda-Hart index** (Duda & Hart, 1973) applied to the same clustering problem. When using the **Duda-Hart index**, one has to acknowledge that different literature sources provide different suggestions on how to use the index. For example, Sarstedt & Mooi (2019) suggests choosing the number of clusters k to maximize the **Duda-Hart index**. The original paper (Duda & Hart, 1973), as well as other authors (Charrad et al., 2014; Gordon, 1999; Milligan & Cooper, 1985) advise selecting the first k , which is accompanied by an index value that exceeds a k-specific critical value.

Remember from lecture, in the i^{th} step of a hierarchical clustering procedure, when two clusters A and B are up to be merged, the index is calculated as follows:

$$DH = \frac{SS_{W_A} + SS_{W_B}}{SS_{W_{A \cup B}}}$$

Here, SS_{W_A} is the within-group sum of squares of cluster A. The critical value calculates as follows (Charrad et al., 2014, p. 6):

$$crit_{DH} = 1 - \frac{2}{\pi * p} - 3.2 \sqrt{\frac{2(1 - \frac{8}{\pi^2 p})}{n_{A \cup B} * p}}$$

Without getting into too much detail here, the critical value takes into account the number of clustering variables involved (p), and the number objects in the joined cluster $A \cup B$, and provides an estimate of what one would expect if the data would not have any cluster structure but is randomly distributed in an p -dimensional space.

Again, we use the function `NbClust()` from the package with the same name (Charrad et al., 2014) to calculate **Duda-Hart index**.

```
clustering_vars <- Brand_Loyalty_Price_Consciousness |>
dplyr::select(x, y)

suggestion_DH_index <- NbClust::NbClust(data = clustering_vars,
    diss = distance_matrix_euclidean,
    distance = NULL,
    min.nc = 1,
    max.nc = 6,
    method = "single",
    index = "duda")[1:3]

suggestion_DH_index
```

```
$All.index
      1      2      3      4      5      6
0.5486 0.6821 0.3236 6.5738 1.2495 0.7522

$All.CriticalValues
      1      2      3      4      5      6
-0.2510 -0.3258 -0.4219 -1.0633 -0.7431 -1.0633

$Best.nc
Number_clusters  Value_Index
          1.0000          0.5486
```

Interpretation

As discussed during the lecture, we are searching for the first DH index value that exceeds the corresponding critical value to determine the opti-

mal k . Doing so, would suggest a final solution with only 1 cluster (DH = 0.55). In contrast, following the maximum DH index rule would lead to the selection of 4 clusters (DH = 6.57)

Finally, we proceed with the **Duda-Hart's Pseudo t^2** (Duda & Hart, 1973).

$$Pseudo\ t^2 = \frac{SS_{W_{A \cup B}} - SS_{W_A} - SS_{W_B}}{\frac{SS_{W_A} + SS_{W_B}}{n_A + n_B}}$$

The big numerator in the formula above ($SS_{W_{A \cup B}} - SS_{W_A} - SS_{W_B}$) represents the part of the within-group sum of squares that merging A and B adds to $A \cup B$ above the sum of squares already present in A and B ($SS_{W_A} + SS_{W_B}$). Thus, the lower the numerator the better. The denominator $\frac{SS_{W_A} + SS_{W_B}}{n_A + n_B}$ represents the mean sum of within-group squares of A and B. The **Pseudo t^2** , therefore, accounts for the number of objects in the A and B ($n_A + n_B$) before merging. The **Duda-Hart index** does not have this feature. The higher the mean sum of within-group squares of A and B (i.e., high heterogeneity within clusters A and B, even before merging), the lower the denominator gets. In conclusion, **Pseudo t^2** tends to be low when merging A and B in the next step of hierarchical clustering step only marginally increases the within-group sum of squares and when the heterogeneity within A and B is already high before merging to $A \cup B$.

Similar to the **Duda-Hart index**, some sources are recommending to search for the global minimal **Pseudo t^2** to determine the appropriate k (Sarstedt & Mooi, 2019), while others recommend to search for the smallest k at which the **Pseudo t^2** is smaller or equal the corresponding critical value (Charrad et al., 2014; Gordon, 1999).

```
clustering_vars <- Brand_Loyalty_Price_Consciousness |>
dplyr::select(x, y)

suggestion_pseudo_t2 <- NbClust::NbClust(data = clustering_vars,
  diss = distance_matrix_euclidean,
  distance = NULL,
  min.nc = 1,
  max.nc = 6,
  method = "single",
  index = "pseudot2")[1:3]

suggestion_pseudo_t2
```

```
$All.index
      1      2      3      4      5      6
4.1138 1.8640 6.2701 0.0000 -0.1997 0.0000
```

```

$All.CriticalValues
      1      2      3      4      5      6
-24.9172 -16.2785 -10.1102  0.0000 -2.3458  0.0000

$Best.nc
Number_clusters  Value_Index
              4              0

```

Interpretation

We are searching for the smallest k at which the **Pseudo t^2** is smaller than, or equal to the corresponding critical value. Doing so, would suggest a 4-cluster solution (Pseudo $t^2 = 0$).

Advanced Cluster Analysis

Conducting Partitioning Clustering

We will only illustrate the application of the **k-means** algorithm (e.g., [Hartigan & Wong, 1979](#)).

Think back to the lecture. The general procedure in k-means follows a fixed sequence:

1. **Initialize k cluster centers** (e.g., by randomly selecting k coordinates in the space of clustering variables).
2. **Creating a starting partition** (i.e., assigning each object to the closest cluster in terms of [Euclidean distance](#) between the objects and the cluster centers).
3. **Re-computing the k cluster centers – the cluster centroids** (in terms of their coordinates regarding the clustering variables as the mean of assigned objects)
4. **Apply exchange algorithm(s)**. That is, check for every object whether reassignment to another cluster would decrease the Euclidean distance of this object to its corresponding cluster center.
5. **Repeat Steps 3 and 4**. Then stop when (a) no reassignments are beneficial or (b) when a certain number of iterations are reached, or when (c) a specific threshold for the decrease in goal function to be minimized is not exceeded.

The goal in k-means is typically to minimize the total within-group sum of squares (SS_W):

$$\min! \quad SS_W = \sum_{i=1}^k \sum_{j=1}^{n_i} \sum_{m=1}^v (center(i)_m - j_m)^2$$

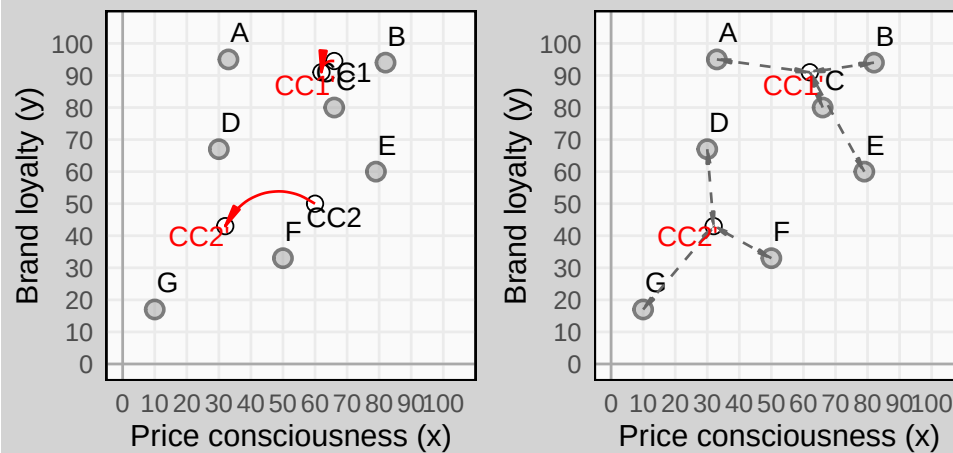
Where $\mathbf{k}$ is the number of clusters, n_i is the number of objects in cluster $\mathbf{i}$, $\mathbf{v}$ is the number of clustering variables, with $\mathbf{m}$ being a specific variable. The term $center(i)_m$ represents the values of the clustering variable $\mathbf{m}$ regarding the center of cluster $\mathbf{i}$. j_m is the value of the clustering variable $\mathbf{m}$ regarding object $\mathbf{j}$ in cluster $\mathbf{i}$.

The figure below visualizes these steps using the data we have used before. If you rather prefer cartoon-like explanations, I strongly recommend Allison Horst's fantastic [Data Science & Statistics Artwork](#).

Step 1: random cluster centers Step 2: assignment to clo



Step 3: recomputing cluster centers Step 4: reassignment to clusters



There are several good packages in *R* that help with conducting partitioning cluster analysis such as **k-means**.

- Aquino, J., Enzmann, D., Schwartz, M., Jain, N., & Kraft, S. (2026). *Descr: Descriptive statistics*. Retrieved from <https://CRAN.R-project.org/package=descr>
- Arel-Bundock, V. (2022). {Modelsummary}: Data and model summaries in {r}. *Journal of Statistical Software*, 103(1). doi: [10.18637/jss.v103.i01](https://doi.org/10.18637/jss.v103.i01)
- Arel-Bundock, V., Greifer, N., & Heiss, A. (2024). How to interpret statistical models using **marginaleffects** for *R* and *Python*. *Journal of Statistical Software*, 111(9). doi: [10.18637/jss.v111.i09](https://doi.org/10.18637/jss.v111.i09)
- Bartlett, M. S. (1950). Tests of significance in factor analysis. *British Journal of Statistical Psychology*, 3(2), 77–85. doi: [10.1111/j.2044-8317.1950.tb00285.x](https://doi.org/10.1111/j.2044-8317.1950.tb00285.x)
- Behrendt, S. (2025). *Lm.beta: Add standardized regression coefficients to linear-model-objects*. Retrieved from <https://CRAN.R-project.org/package=lm.beta>
- Ben-Shachar, M. S., Lüdtke, D., & Makowski, D. (2020). *{E}ffectsize: Estimation of effect size indices and standardized parameters*. 5, 2815. doi: [10.21105/joss.02815](https://doi.org/10.21105/joss.02815)
- Bonferroni, C. E. (1936). *Teoria statistica delle classi e calcolo delle probabilità*. Firenze: Seeber.
- Breusch, T. S., & Pagan, A. R. (1979). A simple test for heteroscedasticity and random coefficient variation. *Econometrica*, 47(5), 1287. doi: [10.2307/1911963](https://doi.org/10.2307/1911963)
- Bulus, M., & Jentschke, S. (2026). *Pwrss: Statistical power, sample size, and detectable effect calculations*. Retrieved from <https://CRAN.R-project.org/package=pwrss>
- Calinski, T., & Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics - Theory and Methods*, 3(1), 1–27. doi: [10.1080/03610927408827101](https://doi.org/10.1080/03610927408827101)
- Champely, S. (2020). *Pwr: Basic functions for power analysis*. Retrieved from <https://CRAN.R-project.org/package=pwr>
- Charrad, M., Ghazzali, N., Boiteau, V., & Niknafs, A. (2014). **NbClust**: An R package for determining the relevant number of clusters in a data set. *Journal of Statistical Software*, 61(6). doi: [10.18637/jss.v061.i06](https://doi.org/10.18637/jss.v061.i06)
- Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Lawrence Erlbaum.
- Cook, R. D., & Weisberg, S. (1983). Diagnostics for heteroscedasticity in regression. *Biometrika*, 70(1), 1–10. doi: [10.1093/biomet/70.1.1](https://doi.org/10.1093/biomet/70.1.1)
- Desgraupes, B. (2023). *clusterCrit: Clustering indices*. Retrieved from <https://CRAN.R-project.org/package=clusterCrit>
- Douglas, C. E., & Michael, F. A. (1991). On distribution-free multiple comparisons in the one-way analysis of variance. *Communications in Statistics - Theory and Methods*, 20(1), 127–139. doi: [10.1080/03610929108830487](https://doi.org/10.1080/03610929108830487)

- Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. UK: Wiley.
- 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)
- Epskamp, S., Cramer, A. O. J., Waldorp, L. J., Schmittmann, V. D., & Borsboom, D. (2012). **qgraph**: Network visualizations of relationships in psychometric data. *Journal of Statistical Software*, 48(4). doi: [10.18637/jss.v048.i04](https://doi.org/10.18637/jss.v048.i04)
- Field, A. P., Miles, J., & Field, Z. (2012). *Discovering statistics using r*. London; Thousand Oaks, Calif: Sage.
- Fisher, R. A. (1922). On the interpretation of χ^2 from contingency tables, and the calculation of p. *Journal of the Royal Statistical Society*, 85(1), 87. doi: [10.2307/2340521](https://doi.org/10.2307/2340521)
- Fox, J., & Monette, G. (2026). Cv: An R package for cross-validating regression models. *Journal of Statistical Software*, 116(8). doi: [10.18637/jss.v116.i08](https://doi.org/10.18637/jss.v116.i08)
- Fox, J., & Weisberg, S. (2019). *An {r} companion to applied regression*. Retrieved from <https://socialsciences.mcmaster.ca/jfox/Books/Companion/>
- Galili, T. (2015). *Dendextend: An r package for visualizing, adjusting, and comparing trees of hierarchical clustering*. doi: [10.1093/bioinformatics/btv428](https://doi.org/10.1093/bioinformatics/btv428)
- Girard, A., Lichters, M., Sarstedt, M., & Biswas, D. (2019). Short- and long-term effects of nonconsciously processed ambient scents in a servicescape: Findings from two field experiments. *Journal of Service Research*, 22(4), 440–455. doi: [10.1177/1094670519842333](https://doi.org/10.1177/1094670519842333)
- Gordon, A. D. (1999). *Classification* (2nd ed.). Boca Raton, London, New York, Washington D.C.: Chapman & Hall/CRC.
- Hair, J. F., Jr., Black, W. C., Babin, B. J., Anderson, R. E., & Tatham, R. L. (2006). *Multivariate data analysis* (6th, Pearson International Edition). New Jersey: Pearson.
- 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)
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning*. doi: [10.1007/978-0-387-84858-7](https://doi.org/10.1007/978-0-387-84858-7)
- Hennig, C. (2026). *Fpc: Flexible procedures for clustering*. Retrieved from <https://CRAN.R-project.org/package=fpc>
- Hofmann, R. J. (1977). Indices descriptive of factor complexity. *The Journal of General Psychology*, 96(1), 103–110. doi: [10.1080/00221309.1977.9920803](https://doi.org/10.1080/00221309.1977.9920803)
- Hofmann, R. J. (1978). Complexity and simplicity as objective indices descriptive of factor solutions. *Multivariate Behavioral Research*, 13(2), 247–250. doi: [10.1207/s15327906mbr1302_9](https://doi.org/10.1207/s15327906mbr1302_9)
- Horn, J. L. (1965). A rationale and test for the number of factors in factor analysis. *Psychometrika*, 30(2), 179–185. doi: [10.1007/bf02289447](https://doi.org/10.1007/bf02289447)
- Iacobucci, D., Popovich, D. L., Moon, S., & Román, S. (2023). How to calculate, use, and report variance explained effect size indices and not die trying. *Journal of Consumer Psychology*, 33(1), 45–61. doi: [10.1002/jcpy.1292](https://doi.org/10.1002/jcpy.1292)
- Jones, M., Kang, K., & Vandekar, S. (2025). **RESI**: An R package for robust ef-

- fect sizes. *Journal of Statistical Software*, 112(3). doi: [10.18637/jss.v112.i03](https://doi.org/10.18637/jss.v112.i03)
- Kaiser, H. F. (1958). The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, 23(3), 187–200. doi: [10.1007/bf02289233](https://doi.org/10.1007/bf02289233)
- Kaiser, H. F. (1974). An index of factorial simplicity. *Psychometrika*, 39(1), 31–36. doi: [10.1007/bf02291575](https://doi.org/10.1007/bf02291575)
- Kaiser, H. F., & Dickman, K. W. (1959). Analytic determination of common factors. *American Psychologist*, 14(7), 425.
- Kassambara, A. (2017). *Practical guide to cluster analysis in r: Unsupervised machine learning*. Retrieved from <https://www.datanovia.com/en/product/practical-guide-to-cluster-analysis-i>
- Kassambara, A. (2026). *Rstatix: Pipe-friendly framework for basic statistical tests*. Retrieved from <https://CRAN.R-project.org/package=rstatix>
- Kay, M., Elkin, L. A., Higgins, J. J., & Wobbrock, J. O. (2025). *ARTool: Aligned rank transform*. doi: [10.32614/CRAN.package.ARTool](https://doi.org/10.32614/CRAN.package.ARTool)
- Koenker, R. (1981). A note on studentizing a test for heteroscedasticity. *Journal of Econometrics*, 17(1), 107–112. doi: [10.1016/0304-4076\(81\)90062-2](https://doi.org/10.1016/0304-4076(81)90062-2)
- Larmarange, J. (2025). *Labelled: Manipulating labelled data*. Retrieved from <https://CRAN.R-project.org/package=labelled>
- Lichters, M., Bengart, P., Sarstedt, M., & Vogt, B. (2017). What really matters in attraction effect research: When choices have economic consequences. *Marketing Letters*, 28(1), 127–138. doi: [10.1007/s11002-015-9394-6](https://doi.org/10.1007/s11002-015-9394-6)
- Lichters, M., Brunnlieb, C., Nave, G., Sarstedt, M., & Vogt, B. (2016). The influence of serotonin deficiency on choice deferral and the compromise effect. *Journal of Marketing Research*, 53(2), 183–198. doi: [10.1509/jmr.14.0482](https://doi.org/10.1509/jmr.14.0482)
- 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)
- Milligan, G. W., & Cooper, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2), 159–179. doi: [10.1007/bf02294245](https://doi.org/10.1007/bf02294245)
- Newey, W. K., & West, K. D. (1987). A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica*, 55(3), 703. doi: [10.2307/1913610](https://doi.org/10.2307/1913610)
- Patil, I. (2021). Visualizations with statistical details: The 'ggstat-splot' approach. *Journal of Open Source Software*, 6(61), 3167. doi: [10.21105/joss.03167](https://doi.org/10.21105/joss.03167)
- Pohlert, T. (2024). *PMCMRplus: Calculate pairwise multiple comparisons of mean rank sums extended*. Retrieved from <https://CRAN.R-project.org/package=PMCMRplus>
- R Core Team. (2026a). *R: A language and environment for statistical computing*. doi: [10.32614/R.manuals](https://doi.org/10.32614/R.manuals)
- Ramsey, J. B. (1969). Tests for specification errors in classical linear least-squares regression analysis. *Journal of the Royal Statistical Society: Series B (Methodological)*, 31(2), 350–371. doi: [10.1111/j.2517-6161.1969.tb00796.x](https://doi.org/10.1111/j.2517-6161.1969.tb00796.x)
- Rinker, T. W., & Kurkiewicz, D. (2018). *{Pacman}: {P}ackage management*

- for $\{r\}$. Retrieved from <http://github.com/trinker/pacman>
- Sablotny-Wackershauser, V., Lichters, M., Guhl, D., Bengart, P., & Vogt, B. (2024). Crossing incentive alignment and adaptive designs in choice-based conjoint: A fruitful endeavor. *Journal of the Academy of Marketing Science*, 52(3), 610–633. doi: [10.1007/s11747-023-00997-5](https://doi.org/10.1007/s11747-023-00997-5)
- Sarstedt, M., & Mooi, E. (2019). *A concise guide to market research*. doi: [10.1007/978-3-662-56707-4](https://doi.org/10.1007/978-3-662-56707-4)
- Sarstedt, M., Ringle, C. M., Raithel, S., & Gudergan, S. P. (2014). In pursuit of understanding what drives fan satisfaction. *Journal of Leisure Research*, 46(4), 419–447. doi: [10.1080/00222216.2014.11950335](https://doi.org/10.1080/00222216.2014.11950335)
- Sawyer, A. G., & Ball, A. D. (1981). Statistical power and effect size in marketing research. *Journal of Marketing Research*, 18(3), 275–290. doi: [10.1177/002224378101800302](https://doi.org/10.1177/002224378101800302)
- Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4), 591. doi: [10.2307/2333709](https://doi.org/10.2307/2333709)
- Tierney, N., & Cook, D. (2023). *Expanding tidy data principles to facilitate missing data exploration, visualization and assessment of imputations*. 105. doi: [10.18637/jss.v105.i07](https://doi.org/10.18637/jss.v105.i07)
- Tukey, J. W. (1949). Comparing individual means in the analysis of variance. *Biometrics*, 5(2), 99. doi: [10.2307/3001913](https://doi.org/10.2307/3001913)
- Vigneau, E., Qannari, E. M., Navez, B., & Cottet, V. (2016). Segmentation of consumers in preference studies while setting aside atypical or irrelevant consumers. *Food Quality and Preference*, 47, 54–63. doi: [10.1016/j.foodqual.2015.02.008](https://doi.org/10.1016/j.foodqual.2015.02.008)
- 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)
- White, H. (1980). A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica*, 48(4), 817. doi: [10.2307/1912934](https://doi.org/10.2307/1912934)
- Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis*. Retrieved from <https://ggplot2.tidyverse.org>
- 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. (2026). *Dplyr: A grammar of data manipulation*. Retrieved from <https://CRAN.R-project.org/package=dplyr>
- Wickham, H., Vaughan, D., & Girlich, M. (2025). *Tidyr: Tidy messy data*. Retrieved from <https://CRAN.R-project.org/package=tidyr>
- William Revelle. (2026b). *Psych: Procedures for psychological, psychometric, and personality research*. Retrieved from <https://CRAN.R-project.org/package=psych>
- Yates, F. (1934). Contingency tables involving small numbers and the χ^2 test. *Supplement to the Journal of the Royal Statistical Society*, 1(2), 217. doi: [10.2307/2983604](https://doi.org/10.2307/2983604)

- Yule, G. U. (1912). On the methods of measuring association between two attributes. *Journal of the Royal Statistical Society*, 75(6), 579. doi: [10.2307/2340126](https://doi.org/10.2307/2340126)
- Zeileis, A., & Hothorn, T. (2002). *Diagnostic checking in regression relationships*. 2. Retrieved from <https://CRAN.R-project.org/doc/Rnews/>
- Zeileis, A., Koell, S., & Graham, N. (2020). Various versatile variances: An object-oriented implementation of clustered covariances in r. *Journal of Statistical Software*, 95(1), 1–36. doi: <http://dx.doi.org/10.18637/jss.v095.i01>