HAMBURG SCHOOL OF
BUSINESS ADMINISTRATION

Digital Tool Box Data Business – Project Report

Project work as part of the Bachelor of Science (B.Sc.) in Business Informatics

Benedikt Kronhardt (5089)
Börge Meyer (5076)

Project Theme	Cost of Living Index: Does the classification of a country as a developing or industrialized country have a significant impact on the cost of living index?
Study Group	20A-BI2
Lecturer	Ulf Köther
Submitted	December 1, 2022
Team Number	1
Wordcount	11041

Contents

List of Figures

List of Tables

Acronyms

CRAN	Comprehensive R Archive Network
H_2O	water
IDE	integrated development environment
GLM	generalized linear model

1. Introduction

The standard of living became more and more important for the world population. But every standard of living comes at a price. How high the standard of living is in a country can be analyzed and compared between countries with the help of the cost of living index.

1.1. Task description

Our task was to analyze a data set and write a report about it using R, RStudio, RMarkdown and the procedures of literate programming to put together a PDF-manuscript. In our team we have received the "cost of living" data set and analyzed it with the research question "Is there a significant difference between industrialized countries and developing countries in cost of living?"

1.2. Structure

Our document is divided into five chapters. This chapter is the introduction, where we describe the task and how we imported our dataset, respectively added other datasets. In the chapter "Theoretical Background" the theoretical background of the dataset and the data collection is briefly discussed, before in the methods section the dataset is statistically described, including information on the variables' distribution, missing values, categories and the relationships between the variables. The results section comprise all necessary calculations, which are then discussed in connection with the research question in the following section ("Discussion").

1.3. Setup

After the required libraries, which will be worked with in the following, were installed, the libraries still had to be imported in order to be able to use them.

```
library(tidyverse)
library(dplyr)
library(stringr)
library(ggplot2)
library(maps)
library(janitor)
library(modelsummary)
library(car)
library(carData)
library(gpairs)
library(GGally)
```

Subsequently, the data had to be read in. This could be initialized with the following command, after the data set was added as a csv file in the folder "02-data". To be able to work better with the names of the columns and the dataset in general, the command "janitor::clean_names" was executed. With this, for example, the spaces were removed and the names were all written in small letters.

```
costOfLiving <- read_delim("02-data/cost-of-living-2017.csv",
                           delim = "\t", escape_double = FALSE,
                           trim_ws = TRUE)
costOfLiving <- janitor::clean_names(costOfLiving)
```

To make it easier to split the data by region, we imported a csv file that shows the names of the countries in this world and their corresponding regions.

First we had to import the dataset, which we named "continents". This dataset is from the website "kaggle", named "Country Mapping - ISO, Continent, Region". [1]

```
#import list of continents and countries
continents <- read_csv("02-data/continents2.csv")
continents <- janitor::clean_names(continents)
```

[1] Kaggle (2019)

To be able to do a join with the raw data, we had to rename the column “name” to “country”. After that, a left join could be performed on the renamed column. Since we only needed the column “region”, a select for this one column was performed within the join.

```
#rename attribute 'name' in 'country' to perform a left join.
continents <- rename(continents, country = name)
costOfLivingAndContinents <- left_join(costOfLiving, select(continents, country, region),
```

Now it was possible to check if a country was not assigned to a region. Since the country Kosovo could not be assigned to a region, this had to be done manually.

```
costOfLivingAndContinents[486, 12] <- "Europe"
```

To assign the different countries in our dataset to either a developing or an industrialized country, we also imported a new csv file, which we named “dd”. We created this file ourselves, based on data from UN (2014).

```
#import list of Countries and development status
dd <- read_delim("02-data/developed_and_developing_countries.csv",
                 delim = ";", escape_double = FALSE,
                 trim_ws = TRUE)
dd <- janitor::clean_names(dd)
```

To format the category as a double value, we executed the following commands.

```
dd$category[dd$category == "developed"] <- 1.0
dd$category[dd$category == "developing"] <- 0.0
dd$category <- as.double(dd$category)
```

Once this was done, we scanned the dataset for various capitalization errors and corrected them. Also we have renamed the column category to development.

```
dd$country[dd$country == "italy"] <- "Italy"
dd$country[dd$country == "Hong Kong SAR"] <- "Hong Kong"
dd$country[dd$country == "Taiwan Province of China"] <- "Taiwan"
dd$country[dd$country == "Russian Federation"] <- "Russia"
dd$country[dd$country == "Viet Nam"] <- "Vietnam"
dd$country[dd$country == "Bosnia and Herzegovina"] <- "Bosnia And Herzegovina"
dd$country[dd$country == "Kosovo"] <- "Kosovo (Disputed Territory)"
colnames(dd)[2] <- "development"
```

In the end, we were able to perform a left join and thus add the categorization of development countries to our dataset.

```
dataFinished <- left_join(costOfLivingAndContinents,dd, by="country")
```

2. Theoretical background

2.1. Cost of Living Index

Because of different prices, living standards, currencies and other factors, it is not possible to compare the cost of living in different countries properly.

To be able to compare the cost of living between different countries, the Cost of Living Index is used - also abbreviated as CLI in the following. The cost of living is the financial resources needed to cover, in a given place and in a given period of time, the basic expenses for a given standard of living, such as a shelter, food, medicines and others. The CLI enables the comparison of expenditures between different places in the world and at different times in history.[1]

In economics, the cost-of-living index describes the ratio of the minimum expenditure required to achieve a given indifference curve between two prices. The calculation not only requires two different price groups, but is also dependent on a preference order of the required living goods and on a basic indifference curve describing the utility of two products. Among the two prices needed, e.g., from two different places, one is called the comparison price and the other is called the reference price or the base price. The base price is then used to illustrate on which prices the Cost-Of-Living Index is based and calculated. The calculated index is then dependent on the comparison prices determined. Further, the general logic of the cost-of-living index is best understood when the index is interpreted in the multiple context of temporal and spatial comparisons.[2]

2.2. Industrialized, emerging and developing countries

In general, countries are divided into industrialized, emerging and developing countries. States in which the economic performance is supported by a large part of the resident companies are referred

[1] Banton (2021)

[2] Pollak (1989)

to as industrialized countries. Such countries stand out due to their high per capita income, which results from the available standard of education, high productivity in production, good external trade relations and usually a currency with low inflation.[3]

A country that is in the process of becoming an industrialized country is called an emerging country. These are nevertheless referred to the category of developing countries. Emerging countries are identifiable by their above-average economic growth. Nevertheless, emerging countries are similar to developing countries in the social structure, such as in the level of education, mortality and access to infrastructure.[4]

The third category is developing countries, which are associated with poor food supply, high poverty, poor health care and educational opportunities. In association with the characteristics, such countries have an overall low standard of living and a preponderance of labor in agriculture and external economic difficulties.[5]

To analyze the available data, developing countries were combined with emerging economies and contrasted with developed countries.

[3] BPB (2021)

[4] BMZ (n.d.b)

[5] BMZ (n.d.a)

3. Methods

3.1. Data Description

The provided data consists of 511 different datasets from 110 different states. The data was set up into City, State, Country, Cost of Living Plus Rent Index, CLI, Rent Index, Groceries Index, Restaurant Index, Local Purchasing Power Index, Leverage Model 1 and Leverage Model 2 attributes.

In Figure 3.1 can be seen how many datasets are available per region.

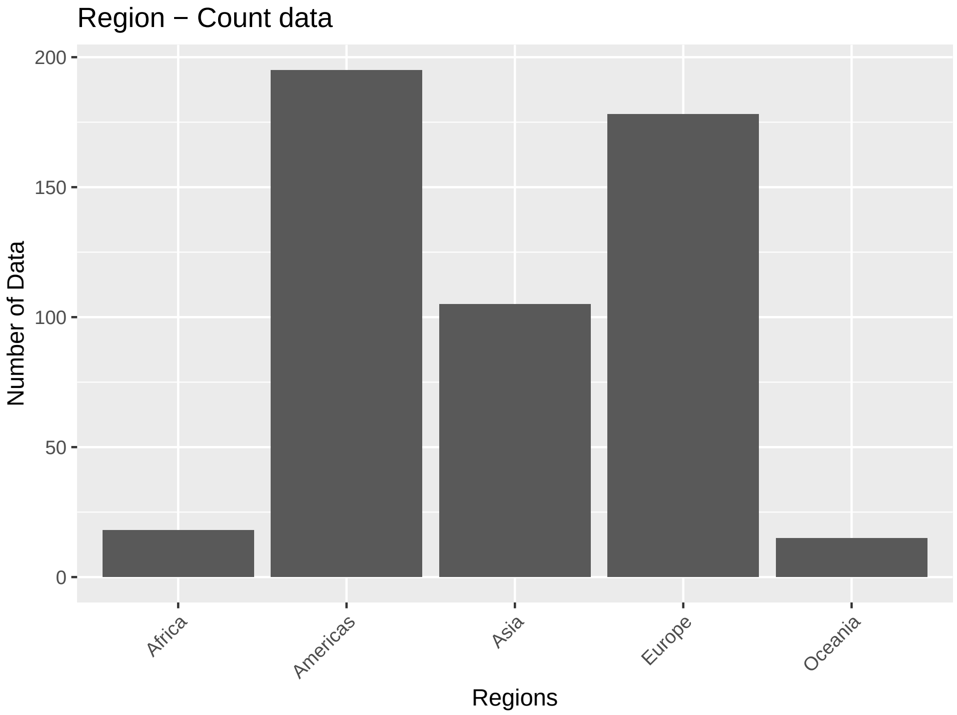


Figure 3.1.: Count of Data from different regions

–>TODO: Text zu Weltkarte schreiben

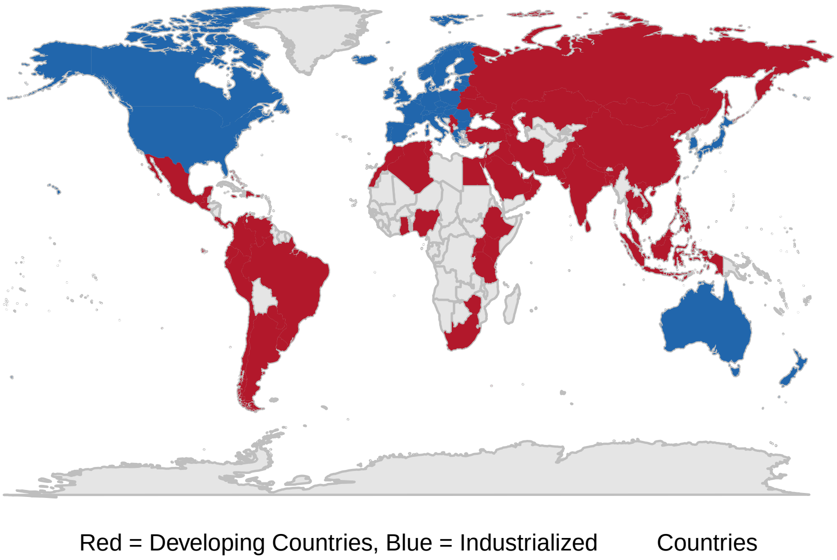

Red = Developing Countries, Blue = Industrialized   Countries

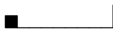
Figure 3.2.: Industrialized and developing countries

3.2. Exploratory Data Analysis

First of all, we had to check, if there are missing values inside of the data-set. Therefore we used the following code to proof this:

```
summary(is.na(dataFinished))
```

```
#>     city           state          country
#>  Mode :logical   Mode :logical   Mode :logical
#>  FALSE:511       FALSE:128       FALSE:511
#>                  TRUE :383
#>  cost_of_living_plus_rent_index    cli
#>  Mode :logical                    Mode :logical
#>  FALSE:511                        FALSE:511
#>
#>  rent_index      groceries_index restaurant_price_index
#>  Mode :logical   Mode :logical   Mode :logical
#>  FALSE:511       FALSE:511       FALSE:511
#>
#>  local_purchasing_power_index leverage_model_1
#>  Mode :logical                Mode :logical
#>  FALSE:511                    FALSE:511
#>
```

	Unique (#)	Missing (%)	Mean	SD	Min	Median	Max	
cost_of_living_plus_rent_index	490	0	46.3	19.0	13.6	48.6	133.2	
cli	489	0	65.2	22.4	21.8	70.6	149.5	
rent_index	485	0	26.8	17.7	3.1	24.5	119.6	
groceries_index	495	0	63.4	24.4	22.0	65.0	163.7	
restaurant_price_index	493	0	59.0	26.7	12.1	65.4	152.5	
local_purchasing_power_index	497	0	80.0	33.7	3.0	84.2	176.2	
development	2	0	0.7	0.5	0.0	1.0	1.0	

```
#>  leverage_model_2   region         development
#>  Mode :logical     Mode :logical   Mode :logical
#>  FALSE:511         FALSE:511       FALSE:511
#>
```

As it can be seen, there were 383 missing values inside the column “state”. However, since the column has no bearing on our research question, we decided to disregard this column. With the city column we have a more meaningful basis to answer our question. To disregard this column, we cut it off. To do this, we used to following R code chunk. Because it is the second column, we can just delete this column.

```
dataFinished <- dataFinished[-2]
```

We also truncated the leverage_model_1 and leverage_model_2 columns, since we did not work with these columns any further.

In our customized dataset we still have seven numeric variables, which are shown in table **??**. Also the mean, median, minimum and maximum value of each numerical variable can be read here.

```
datasummary_skim(dataFinished, allign="center")
```

To determine if outliers exist within the data set, we chose to draw a boxplot.

As can be seen from the figure, there are several outliers within the data set. In order not to distort the result, we decided to keep these outliers and to continue working with them.

–> TODO: Text zu Correlation schreiben

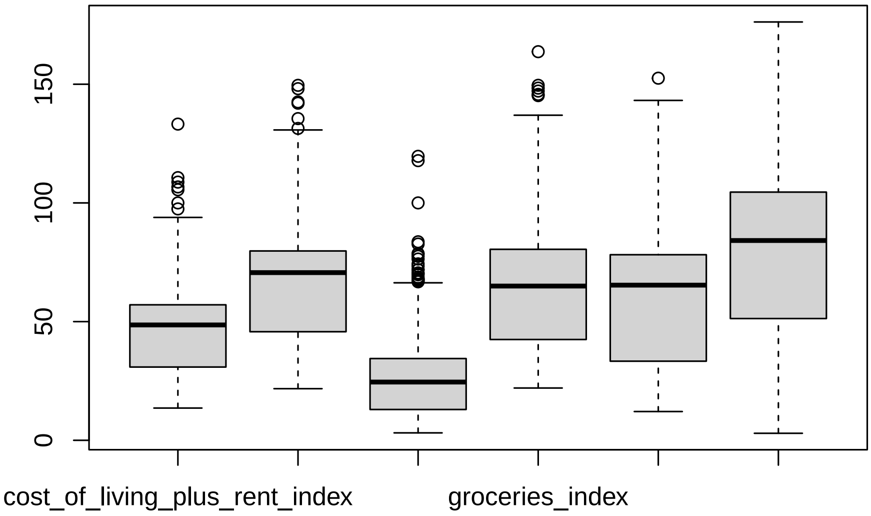


Figure 3.3.: Boxplot of Data to identify outliers

	cost_of_living_plus_rent_index	cli	rent_index	groceries_index	resta
cost_of_living_plus_rent_index	1	.	.	.	
cli	0.96	1	.	.	
rent_index	0.93	0.79	1	.	
groceries_index	0.92	0.95	0.77	1	
restaurant_price_index	0.91	0.95	0.75	0.85	
local_purchasing_power_index	0.66	0.64	0.60	0.65	
development	0.61	0.65	0.47	0.60	

```
datasummary_correlation(dataFinished)
```

3.3. Folgendes nur zur Übersicht/weiteren Aufbau(Gliederung)

zu wrangling gehören joining, merging und grouping Boxplot gehört zu profiling

3. Methods, for example:

- Descriptive statistics
- Variable distributions
- Statements about variances and co-variance and missing values

- Chosen analytical procedures to answer the research question

Inhalt/Aufbau: 1:EDA (Exploratory Data Analysis) auf Datenset anwenden 1.1: including information on the variables' distribution, missing values, categories / grouping factors (if applicable) and the relationships between the variables, especially regarding the variable in focus, for example, the dependent variable of a applied statistical method (if applicable) 2:Methoden/statistische Modelle aufstellen

4. Results

To determine whether there is a significant difference between developing and developed countries, we decided to run a multiple linear regression. This is to determine whether the classification into a developing country has a significant influence on the cost of living index or not.

4.1. Multiple linear regression

Within multiple linear regression, our dependent variable (y) is the cost of living index. Our independent variables (x) are the rent index, the groceries index, the restaurant price index, the local purchasing power index and the development status.

In order to perform a multiple linear regression, some conditions have to be fulfilled, which we will check in the following.

First, there must be a linear relationship between the x variables and the y variable. This is evident from the correlation shown in –>(TODO: Querverweis Kapitel Methods oder auch nur Tabelle/Figure Korrelation). Also, the y variable must be metrically scaled, which is given.

Third, the residuals should be approximately normally distributed. We proved this graphically with the help of a histogram.

First, we need to set up our model.

```
model <- lm(cli ~ rent_index + groceries_index + restaurant_price_index + local_purchasin
```

After that, we can create a histogram from our model.

From the histogram we can see that the distribution can be considered normally distributed, therefore this condition is also fulfilled. Scaling is also given, since the cost of living index is on a scale.

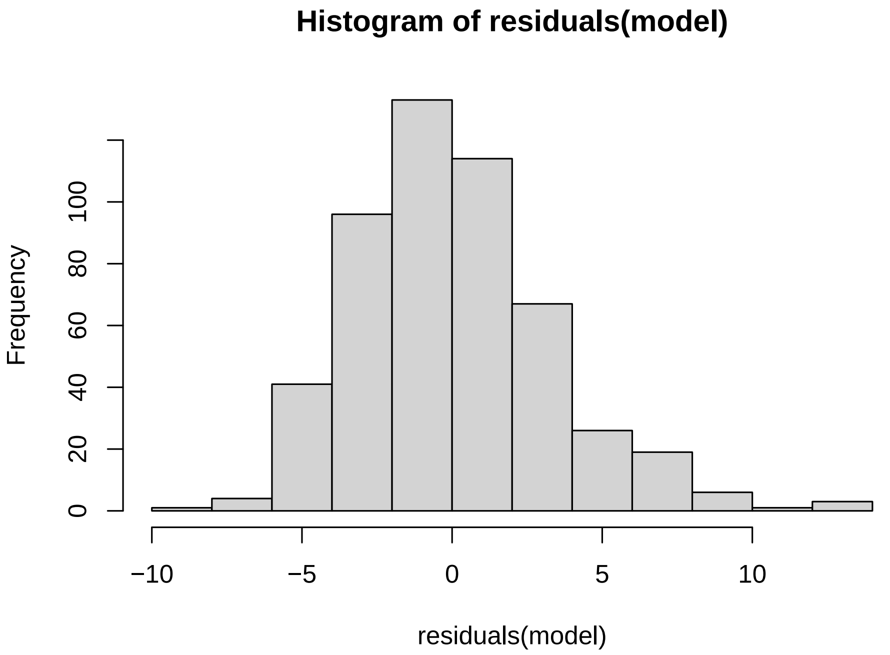


Figure 4.1.: Histogram of the multiple linear regression model

The last condition we checked is that there must be no multicollinearity within the independent variables. To check this, we created a correlation matrix. First, we generated a subset from the data in which the variables to be tested are stored. Then we created the correlation matrix from this subset and worked with the pearson method.

```
#>                               rent_index groceries_index
#> rent_index                     1.0000000       0.7674361
#> groceries_index                0.7674361       1.0000000
#> restaurant_price_index         0.7523090       0.8518550
#> local_purchasing_power_index   0.6000432       0.6458339
#> development                    0.4730879       0.5998356
#>                               restaurant_price_index
#> rent_index                                 0.7523090
#> groceries_index                            0.8518550
#> restaurant_price_index                     1.0000000
#> local_purchasing_power_index               0.6436926
#> development                                0.6838520
#>                               local_purchasing_power_index
#> rent_index                                       0.6000432
#> groceries_index                                  0.6458339
#> restaurant_price_index                           0.6436926
#> local_purchasing_power_index                     1.0000000
#> development                                      0.6425433
#>                               development
#> rent_index                      0.4730879
#> groceries_index                 0.5998356
#> restaurant_price_index          0.6838520
#> local_purchasing_power_index    0.6425433
#> development                     1.0000000
```

Since the correlation between restaurant price index and groceries index is $0.851855 >$ 0.8, this may indicate that there is multicollinearity. To confirm this, we used another method to check for multicollinearity, the method of Variance Inflation Factor values.

```
#>                   rent_index              groceries_index 
#>                     2.821033                     4.346802 
#>       restaurant_price_index local_purchasing_power_index 
#>                     4.863177                     2.187319 
#>                  development 
#>                     2.241055
```

Since according to this method none of the values is $>$10 we have rejected the theory of multicollinearity.

Now that all the assumptions can be accepted, we come to the actual evaluation of the model.

```
#> 
#> Call:
#> lm(formula = cli ~ rent_index + groceries_index + restaurant_price_index + 
#>     local_purchasing_power_index + development, data = dataFinished)
#> 
#> Residuals:
#>     Min      1Q  Median      3Q     Max 
#> -8.4054 -2.4098 -0.2694  1.8302 12.6586 
#> 
#> Coefficients:
#>                               Estimate Std. Error t value
#> (Intercept)                  10.785427   0.468691  23.012
#> rent_index                    0.028668   0.013976   2.051
#> groceries_index               0.479207   0.012561  38.152
#> restaurant_price_index        0.427674   0.012145  35.213
#> local_purchasing_power_index -0.028731   0.006438  -4.463
#> development                   0.466335   0.461661   1.010
#>                              Pr(>|t|)    
#> (Intercept)                   < 2e-16 ***
#> rent_index                     0.0408 *  
#> groceries_index               < 2e-16 ***
#> restaurant_price_index        < 2e-16 ***
#> local_purchasing_power_index 9.98e-06 ***
#> development                    0.3129
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 3.317 on 505 degrees of freedom
#> Multiple R-squared:  0.9782, Adjusted R-squared:  0.978
#> F-statistic:  4539 on 5 and 505 DF,  p-value: < 2.2e-16
```

The model makes a significant explanatory contribution, as the p-value is well below 0.05, and we can proceed with the interpretation of the further results.

As we can see, according to the p-values, all variables except the classification of development have a significant impact on the cost of living index.

5. Discussion

Stichpunkte/Überthemen zum diskutieren/kritisch hinterfragen:

##Critical assessment of the data The objective of this study was to determine to what degree the status as an industrialized or developing country has an influence on the Cost of Living Index.

The critical review allows first of all to scrutinize the available data. Most of the data sets that were used as a basis for this work did not include all existing countries. In addition, it must be mentioned that a large number of African countries in particular are not included in the initial data. This could have biased the results of the work (see Figure 3.1).

Furthermore, data were added that resulted in additional analysis possibilities, such as the representation of industrialized and developing countries. Data from the United Nations is considered to be trusted because the United Nations is an official and recognized organization.

Data produced by third parties are classified as less trustworthy, as this can lead to falsification. Since this was based on the regional allocation of the data provided for the different countries, the usage does not have a high weighting in the result.

Translated with www.DeepL.com/Translator (free version)

5.1. Woher stammen die Daten

-sowohl der eigentliche Datensatz als auch die hinzugefügten Datensätze (continents2 und dd) - Waren die Seiten vertrauensvoll, wie hätte man alternativ “bessere” Daten bekommen können?

5.2. Reicht es aus, eine einfache multiple lineare regression zu machen, um die Forschungsfrage zu beantworten? - Was hätte man zusätzlich machen können?

-Können wir eine Aussage dadurch treffen?

5.3. Ist die Forschungsfrage mit der Menge der Daten überhaupt aussagekräftig? - Gibt es genug Datensätze für Industrie und Entwicklungsländer?

5. Diskussion Muss noch überarbeitet werden!

Inhalt: Die vorher dargestellten Results diskutieren in Verbindung mit der Forschungsfrage!

References

Allaire, JJ, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. 2022. *Rmarkdown: Dynamic Documents for r*. https://CRAN.R-project.org/package=rmarkdown.

Banton, Caroline. 2021. "Cost of Living: Definition, How to Calculate, Index, and Example." March 2021. https://www.investopedia.com/terms/c/cost-of-living.asp.

BMZ. n.d.a. "Entwicklungsland." Bundesamt für wirtschaftliche Zusammenarbeit und Entwicklung. https://www.bmz.de/de/service/lexikon/entwicklungsland-14308.

———. n.d.b. "Schwellenland." Bundesamt für wirtschaftliche Zusammenarbeit und Entwicklung. https://www.bmz.de/de/service/lexikon/schwellenland-14810.

BPB. 2021. "Industrieländer." Bundeszentrale für politische Bildung. June 2021. https://www.bpb.de/kurz-knapp/lexika/lexikon-der-wirtschaft/19720/industrielaender/.

Kaggle. 2019. "Country Mapping - ISO, Continent, Region." December 2019. https://www.kaggle.com/datasets/andradaolteanu/country-mapping-iso-continent-region.

Pollak, Robert A. 1989. *The Theory of the Cost-of-Living Index*. Oxford University Press.

R Core Team. 2022. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

UN. 2014. *World Economic Situation and Prospects*. United Nations.

Xie, Yihui. 2022. *Bookdown: Authoring Books and Technical Documents with r Markdown*. https://CRAN.R-project.org/package=bookdown.

Declaration of Honor

We hereby declare that

1. we wrote this project report without the assistance of others;
2. we have marked direct quotes used from the literature and the use of ideas of other authors at the corresponding locations in the thesis;
3. we have not presented this thesis for any other exam. We acknowledge that a false declaration will have legal consequences.

Hamburg, dd.mm.yyyy

aaaaa, bbbbbb

We accept that the HSBA may check the originality of our work using a range of manual and computer based techniques, including transferring and storing our submission in a database for the purpose of data-matching to help detect plagiarism.

Hamburg, dd.mm.yyyy

aaaaa, bbbbbb

6. Data-set

```
library(tidyverse) # This includes readr!
library(xtable) # For displaying LaTeX tables
library(modelsummary) # For displaying regression models in tables
library(stargazer) # For displaying regression models in tables
```

```
#>
#> Please cite as:
#>  Hlavac, Marek (2022). stargazer: Well-Formatted Regression and Summary Statistics Tables.
#>   R package version 5.2.3. https://CRAN.R-project.org/package=stargazer
```

```
library(jtools) # For displaying regression models in tables
library(kableExtra) # For displaying or changing tables
```

```
#>
#> Attache Paket: 'kableExtra'
#> Das folgende Objekt ist maskiert 'package:dplyr':
#>
#>     group_rows
```

```
library(gt) # For displaying tables
```

```
#>
#> Attache Paket: 'gt'
#> Das folgende Objekt ist maskiert 'package:modelsummary':
#>
#>     escape_latex
```

```
library(gtsummary) # For model reporting inline and in tables
library(broom) # For working with statistical models
library(car) # For type-III anova tests
library(report) # For automated text-based model reporting
library(effects) # For working with statistical models / visualize effects
```

```
#> lattice theme set by effectsTheme()
#> See ?effectsTheme for details.
```

```
library(ggeffects) # For working with statistical models / visualize effects
library(patchwork) # For putting different visualizations in one figure
library(janitor)

dataset <- read_delim("02-data/cost-of-living-2017.csv",
                      delim = "\t", escape_double = FALSE,
                      trim_ws = TRUE)
```

```
#> Rows: 511 Columns: 11
#> -- Column specification --------------------------
#> Delimiter: "\t"
#> chr (3): City, State, Country
#> dbl (8): Cost of Living Plus Rent Index, CLI, Rent Index...
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
continents <- read_csv("02-data/continents2.csv")
```

```
#> Rows: 249 Columns: 11
#> -- Column specification --------------------------
#> Delimiter: ","
#> chr (7): name, alpha-2, alpha-3, iso_3166-2, region, sub...
#> dbl (4): country-code, region-code, sub-region-code, int...
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(dataset)
```

```
#> # A tibble: 6 x 11
#>   City   State Country Cost ~1   CLI Rent ~2 Groce~3 Resta~4
#>   <chr>  <chr> <chr>     <dbl> <dbl>   <dbl>   <dbl>   <dbl>
#> 1 Zurich <NA>  Switze~   109.   150.    66.8    164.    141.
#> 2 Hamil~ <NA>  Bermuda   133.   148.   118.     145.    153.
#> 3 Zug    <NA>  Switze~   106.   143.    67.4    148.    143.
#> 4 Geneva <NA>  Switze~   107.   142.    70.2    147.    139.
#> 5 Basel  <NA>  Switze~    97.5  142.    51.5    150.    132.
#> 6 Bern   <NA>  Switze~    91.1  136.    45.3    146.    122.
#> # ... with 3 more variables:
#> #   `Local Purchasing Power Index` <dbl>,
#> #   `Leverage Model 1` <dbl>, `Leverage Model 2` <dbl>, and
#> #   abbreviated variable names
#> #   1: `Cost of Living Plus Rent Index`, 2: `Rent Index`,
#> #   3: `Groceries Index`, 4: `Restaurant Price Index`
```

```
dataset <- janitor::clean_names(dataset)

#Die Spalte region bei Dataset hinzufügen; Kosovo muss separat hinzugefügt werden, ist ni
manipulated_continents <- rename(continents, country=name)

manipulated_data <- left_join(dataset, select(manipulated_continents, country, region), b

#Prüfen, ob eine Region na ist -> Ergebnis: Kosovo ist na
filter(manipulated_data, is.na(region))
```

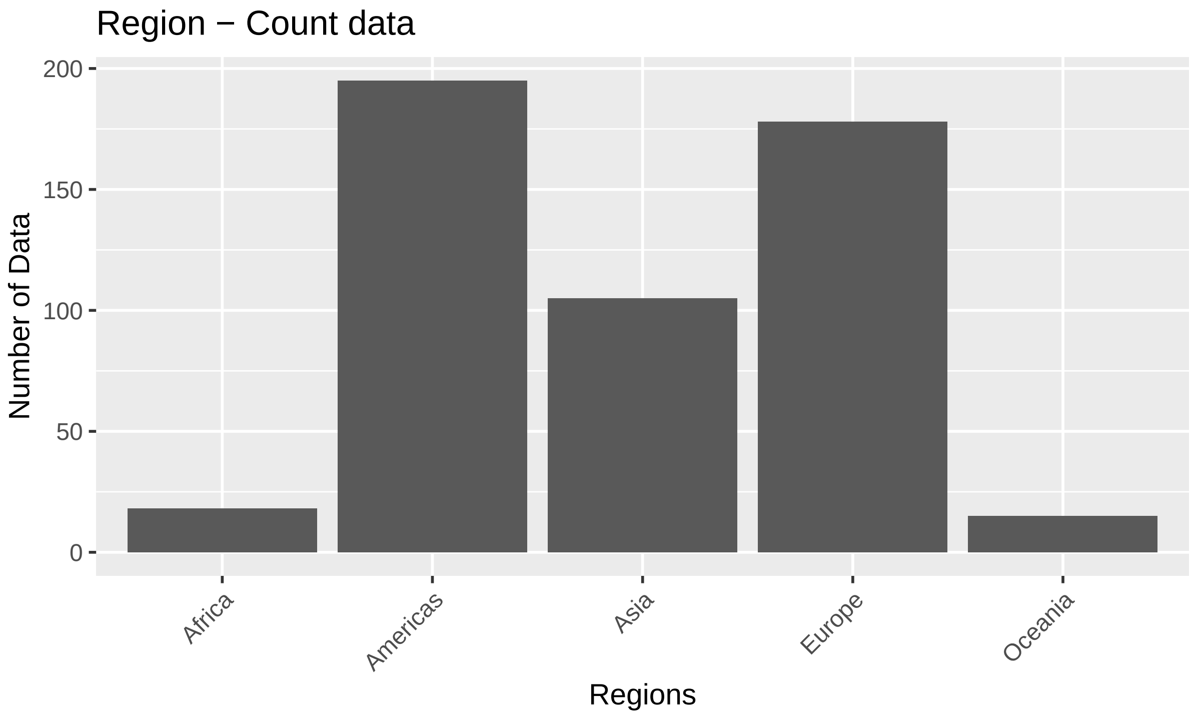
```
#> # A tibble: 1 x 12
#>   city   state country cost_~1   cli rent_~2 groce~3 resta~4
#>   <chr>  <chr> <chr>     <dbl> <dbl>   <dbl>   <dbl>   <dbl>
#> 1 Prist~ <NA>  Kosovo~    19.3  29.4     8.9    26.6    22.3
#> # ... with 4 more variables:
#> #   local_purchasing_power_index <dbl>,
#> #   leverage_model_1 <dbl>, leverage_model_2 <dbl>,
#> #   region <chr>, and abbreviated variable names
#> #   1: cost_of_living_plus_rent_index, 2: rent_index,
#> #   3: groceries_index, 4: restaurant_price_index
```

```
#Da Kosovo na ist, die Zelle in "Europe" ändern
manipulated_data[486, 12] <- "Europe"

dd <- read_delim("02-data/developed_and_developing_countries.csv",
                 delim = ";", escape_double = FALSE,
                 trim_ws = TRUE)
```

```
#> Rows: 172 Columns: 2
#> -- Column specification --------------------------
#> Delimiter: ";"
#> chr (2): country, category
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
dd <- janitor::clean_names(dd)
dd$category[dd$category == "developed"] <- 1.0
dd$category[dd$category == "developing"] <- 0.0
dd$category <- as.double(dd$category)
```

```
dd$category[dd$category == "developing"] <- 0.0
dd$country[dd$country == "italy"] <- "Italy"
dd$country[dd$country == "Hong Kong SAR"] <- "Hong Kong"
dd$country[dd$country == "Taiwan Province of China"] <- "Taiwan"
dd$country[dd$country == "Russian Federation"] <- "Russia"
dd$country[dd$country == "Viet Nam"] <- "Vietnam"
dd$country[dd$country == "Bosnia and Herzegovina"] <- "Bosnia And Herzegovina"
dd$country[dd$country == "Kosovo"] <- "Kosovo (Disputed Territory)"


colnames(dd)[2] <- "development"

dataWithCategory <- left_join(manipulated_data,dd, by="country")

#Prüfen, ob irgendwo ein 'NA' ist
dataToCorrectCity <- filter(dataWithCategory, is.na(city))
dataToCorrectCountry <- filter(dataWithCategory, is.na(country))
dataToCorrectCliPlusRentIndex <- filter(dataWithCategory, is.na(cost_of_living_plus_rent_
dataToCorrectCli <- filter(dataWithCategory, is.na(cli))
dataToCorrectRentIndex <- filter(dataWithCategory, is.na(rent_index))
dataToCorrectGroceriesIndex <- filter(dataWithCategory, is.na(groceries_index))
dataToCorrectRestaurantPriceIndex <- filter(dataWithCategory, is.na(restaurant_price_inde
dataToCorrectLocalPurchasingPowerIndex <- filter(dataWithCategory, is.na(local_purchasing
dataToCorrectRegion <- filter(dataWithCategory, is.na(region))
dataToCorrectDevelopment <- filter(dataWithCategory, is.na(development))

#source for data-hange: https://www.laenderdaten.info/entwicklungslaender.php



#Output nach Regionen und Anzahl an Datensätzen anzeigen
ggplot(data = manipulated_data, aes(x = region)) +
  geom_bar() +
  ggtitle("Region - Count data") +
  xlab("Regions") +
  ylab("Number of Data") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
```



```
#Als Tabelle mit Werten:
manipulated_data %>% count(region)
```

```
#> # A tibble: 5 x 2
#>   region       n
#>   <chr>    <int>
#> 1 Africa      18
#> 2 Americas   195
#> 3 Asia       105
#> 4 Europe     178
#> 5 Oceania     15
```

```
#Split into two datasets with developing countries and industrial countries
developingCountries <- filter(dataWithCategory, development==0)

industrialCountries <- filter(dataWithCategory, development==1)


# t-Test for difference between developing and industrial countries:
test1 <- t.test(cost_of_living_plus_rent_index ~ development, data=dataWithCategory, var.
test1_glance <- glance(test1)
test1
```

```
#> 
#> 	Two Sample t-test
#> 
#> data:  cost_of_living_plus_rent_index by development
#> t = -17.319, df = 509, p-value < 2.2e-16
#> alternative hypothesis: true difference in means between group 0 and group 1 is not equal to
#> 95 percent confidence interval:
#>  -26.99355 -21.49325
#> sample estimates:
#> mean in group 0 mean in group 1 
#>        30.41876        54.66216
```

```
test1_glance
```

```
#> # A tibble: 1 x 10
#>   estimate estima~1 estim~2 stati~3  p.value param~4 conf.~5
#>      <dbl>    <dbl>   <dbl>   <dbl>    <dbl>   <dbl>   <dbl>
#> 1    -24.2     30.4    54.7   -17.3 3.61e-53     509   -27.0
#> # ... with 3 more variables: conf.high <dbl>, method <chr>,
#> #   alternative <chr>, and abbreviated variable names
#> #   1: estimate1, 2: estimate2, 3: statistic, 4: parameter,
#> #   5: conf.low
```

```
#TODO: interpret these results
```

A. Useful code examples

A.1. Use of acronyms

To begin, let's see how the list of acronyms is working. The acronyms are defined in the file `abbreviations.tex`, see an explanation of how to do that in the file itself. Regarding the use of these acronyms inside of the text: In the source code, one writes something as follows, using standard LaTeX commands:

```
First use: \Ac{cran}. Second use: \ac{cran}. First use of another abbreviation:
\ac{ide}, then the plural use: \acp{ide}. And here you see that \acs{H2O}
is actually \ac{H2O}.
```

This displays as follows:

> First use: Comprehensive R Archive Network (CRAN). Second use: CRAN. First use of another abbreviation: integrated development environment (IDE), then the plural use: IDEs. And here you see that $\mathrm{H_2O}$ is actually water ($\mathrm{H_2O}$).

A.2. Load packages and read data into R

At the beginning of your book, you should put all necessary packages in one chunk, then loading the data in the next chunk so that these steps are properly organized:

```r
library(tidyverse) # This includes readr!
library(xtable) # For displaying LaTeX tables
library(modelsummary) # For displaying regression models in tables
library(stargazer) # For displaying regression models in tables
library(jtools) # For displaying regression models in tables
library(kableExtra) # For displaying or changing tables
library(gt) # For displaying tables
library(gtsummary) # For model reporting inline and in tables
library(broom) # For working with statistical models
library(car) # For type-III anova tests
library(report) # For automated text-based model reporting
library(effects) # For working with statistical models / visualize effects
library(ggeffects) # For working with statistical models / visualize effects
library(patchwork) # For putting different visualizations in one figure
```

One of the most common ways to get your data into R is to place them into your project directory as a CSV-file and read them into the current session, that is, the session of the produced book, using readr (https://cran.r-project.org/web/packages/readr/).

```
df1 <- read_csv("02-data/mpg_data_as_csv.csv", lazy = FALSE)
```

```
#> Rows: 234 Columns: 11
#> -- Column specification --------------------------
#> Delimiter: ","
#> chr (6): manufacturer, model, trans, drv, fl, class
#> dbl (5): displ, year, cyl, cty, hwy
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

You can then use the function arguments to change necessary things like the delimiter (also switching to `readr::read_delim()` instead of `readr::read_csv()` to do this), or see the specifics of the column types using `read::spec()` and change them accordingly if needed.

```
spec(df1)
```

```
#> cols(
#>   manufacturer = col_character(),
#>   model = col_character(),
#>   displ = col_double(),
#>   year = col_double(),
#>   cyl = col_double(),
#>   trans = col_character(),
#>   drv = col_character(),
#>   cty = col_double(),
#>   hwy = col_double(),
#>   fl = col_character(),
#>   class = col_character()
#> )
```

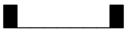
```
df1 <- read_delim("02-data/mpg_data_as_csv.csv",
                  delim = ",",
                  col_types = cols(
                      manufacturer = col_character(),
                      model = col_character(),
                      displ = col_double(),
                      year = col_double(),
                      cyl = col_double(),
                      trans = col_character(),
                      drv = col_character(),
                      cty = col_double(),
                      hwy = col_double(),
                      fl = col_character(),
                      class = col_character()
                      ), lazy = FALSE
                  )
```

A.3. Displaying different types of tables from modelsummary

The data set is part of the ggplot2 package (https://cran.r-project.org/web/packages/ggplot2/) and is originally named mpg. To display a summary of the data set in your work, you may use, for example, the modelsummary package with its function modelsummary::datasummary_skim(), from which the output can be seen in table A.1. Don't forget to name the chunk and specify the table caption using the argument title in the function call! Without these two adjustments, you cannot cross-reference this table and it does not appear in the list of tables in the frontmatter. Another source of error is the naming of the chunks: Never use underscores in these, only letters and minus signs! And always place a blank line in front and after a code chunk.

```
# You can reference this table now with "\@ref(tab:skim-df1)".
# The part with kable_styling is coming from the kableExtra package and
# changes the position of the table and its appearance
datasummary_skim(df1, output = 'kableExtra', booktabs = TRUE,
        title = "Overview of the numerical variables in data set df1") %>%
    kable_styling(latex_options = c("striped", "HOLD_position"))
```

Table A.1.: Overview of the numerical variables in data set df1

	Unique (#)	Missing (%)	Mean	SD	Min	Median	Max	
displ	35	0	3.5	1.3	1.6	3.3	7.0	
year	2	0	2003.5	4.5	1999.0	2003.5	2008.0	
cyl	4	0	5.9	1.6	4.0	6.0	8.0	
cty	21	0	16.9	4.3	9.0	17.0	35.0	
hwy	27	0	23.4	6.0	12.0	24.0	44.0	

Another useful table can be produced by using the function `datasummary_correlation()` from the same package. Its output can be seen in table A.2, which in addition is changed to show how to use footnotes using `kableExtra`. The content of the footnote is also true for table A.1, so be careful and adjust variable types before doing something with them such as in table A.3.

```r
df1_footnote <- df1
names(df1_footnote)[names(df1_footnote) == "year"] <-
    paste0(names(df1_footnote)[names(df1_footnote) == "year"],
           footnote_marker_symbol(1, "latex"))

datasummary_correlation(df1_footnote, output = 'kableExtra',
        booktabs = TRUE, escape = FALSE, # 'escape = FALSE' here is important!
        title = "Correlations of numerical variables in data set df1") %>%
    kable_styling(latex_options = c("striped", "HOLD_position")) %>%
    footnote(symbol = paste0("Using the categorical variable",
                             " 'year' here in a correlation is :-("),
             threeparttable = TRUE) # This last options enables the line break!
```

Table A.2.: Correlations of numerical variables in data set df1

	displ	year*	cyl	cty	hwy
displ	1	.	.	.	.
year*	0.15	1	.	.	.
cyl	0.93	0.12	1	.	.
cty	−0.80	−0.04	−0.81	1	.
hwy	−0.77	0.002	−0.76	0.96	1

* Using the categorical variable 'year' here in a correlation is :-(

Adjusting all categorical variables to the correct type:

```
df1 <- df1 %>%
    mutate(manufacturer = factor(manufacturer),
           model = factor(model),
           year = factor(year),
           cyl = ordered(cyl,
                         levels = c(4,5,6,8),
                         labels = c("4 Cylinders",
                                    "5 Cylinders",
                                    "6 Cylinders",
                                    "8 Cylinders")),
           trans = factor(trans),
           drv = factor(drv,
                        levels = c("f", "r", "4"),
                        labels = c("Front wheel drive",
                                   "Rear wheel drive",
                                   "4 wheel drive")),
           fl = factor(fl),
           class = factor(class))
```

Now using another kind of table from the `modelsummary` package:

```
datasummary_crosstab(drv ~ year,
                     data = df1, booktabs = TRUE,
        title = "Cross tabulations for year and type of drive") %>%
    kable_styling(latex_options = c("striped", "HOLD_position"))
```

Table A.3.: Cross tabulations for year and type of drive

drv		1999	2008	All
Front wheel drive	N	57	49	106
	% row	53.8	46.2	100.0
Rear wheel drive	N	11	14	25
	% row	44.0	56.0	100.0
4 wheel drive	N	49	54	103
	% row	47.6	52.4	100.0
All	N	117	117	234
	% row	50.0	50.0	100.0

See the `datasummary` vignettes for more possibilities:

https://vincentarelbundock.github.io/modelsummary/articles/datasummary.html

A.4. Reporting statistical models

A.4.1. t-test

In the following paragraphs, I want to give some examples on how to report a statistical model. Let's start with a simple one, an independent two sample t-test:

$$t = \frac{m_A - m_B}{\sqrt{\frac{s^2}{n_A} + \frac{s^2}{n_B}}} \tag{A.1}$$

where s^2 is an estimator of the common variance of the two samples. It can be calculated as

$$s^2 = \frac{\sum (x - m_A)^2 + \sum (x - m_B)^2}{n_A + n_B - 2} \tag{A.2}$$

Once the t-test statistic value is calculated, one uses the critical value of Student's t-distribution corresponding to the significance level alpha of your choice (5%). The degrees of freedom (df) used in this test are $df = n_A + n_B - 2$. We can simulate some data according to equation (A.1) and apply the function `t.test()` from the `stats` package:

```
N <- 100
delta <- 5
same_sd <- 5
df_ttest <- tibble(class = gl(n = 2, k = N/2, labels = c("Class A", "Class B")),
                   exam_score = c(rnorm(N/2, mean = 50, sd = same_sd),
                                  rnorm(N/2, mean = 50 + delta, sd = same_sd)))
test1 <- t.test(exam_score ~ class, data = df_ttest, var.equal = TRUE)
test1_glance <- glance(test1)
test1
```

```
#>
#>  Two Sample t-test
#>
#> data:  exam_score by class
#> t = -5.9078, df = 98, p-value = 5.025e-08
#> alternative hypothesis: true difference in means between group Class A and group Class B is
#> 95 percent confidence interval:
#>  -8.056795 -4.005108
#> sample estimates:
#> mean in group Class A mean in group Class B
#>              49.66084              55.69179
```

This console output is not very pleasant and should not be reported as this. Better to use the package broom and its function broom::glance() to extract everything you need using inline code chunks, which gives you a significant difference of ≈ -6.03 between class A $(M = 49.66, SD = 4.42)$ and class B $(M = 55.69, SD = 5.71)$ in this case, $t(98) = -5.908$, $p < .001$. You should read the source code of this paragraph carefully to see how everything in the inline chunks fits together to produce such an output.

A.4.2. χ^2-test

The same procedure can be used to report a χ^2-test using the function chisq.test() from the stats package. This time, let's use the mpg data set imported above in section A.2:

```
# Build a contingency table for year and cylinders
tbl_df <- table(df1$year, df1$cyl)
chi_test <- chisq.test(tbl_df)
chi_result <- glance(chi_test)
tbl_df
```

```
#>
#>        4 Cylinders 5 Cylinders 6 Cylinders 8 Cylinders
#>   1999          45           0          45          27
#>   2008          36           4          34          43
```

```
chi_test
```

```
#>
#>  Pearson's Chi-squared test
#>
#> data:  tbl_df
#> X-squared = 10.189, df = 3, p-value = 0.01703
```

```
chi_result
```

```
#> # A tibble: 1 x 4
#>   statistic p.value parameter method
#>       <dbl>   <dbl>     <int> <chr>
#> 1      10.2  0.0170         3 Pearson's Chi-squared test
```

In this toy example, we can report that the cell means appear to be significantly different from each other, in other words, that they are not independent, $\chi^2(3) = 10.189$, $p = 0.017$.

Because it is very tedious to use the `ifelse()` command inside an inline code chunk to print the p-value correctly, one can define a function for printing the p-value that simplifies the source code a little bit like in the following code chunk.

```
print_p_value <- function(p.num, DIGITS = 3) {
    if (abs(p.num) < 0.001) {
        number <- '<~.001'
    } else if (abs(p.num) > 0.9) {
        number <- '>~.9'
    } else {
        number <- paste0('=~',
                         stringr::str_replace(round(p.num, dig = DIGITS),
                                              '0\\.', '.'))
    }
    return(as.character(number))
}
```

This gives you the possibility to write the test as follows, $\chi^2(3) = 10.189$, $p = .017$. Please look at the source code where the χ^2-test is reported for the second time to see the difference to the inline code chunk from before.

A.4.3. Linear regression models

In this section, we will again use the data set loaded in section A.2, where we formulate different linear regression models using the function `lm()` from the `stats` package to predict the outcome hwy (highway miles per gallon). The first one will be a more complicated model including a continuous predictor, `displ` (displacement), two categorical predictors, `year` and `cyl` (number of cylinders), and the interaction between the two categorical independent variables. The second one will be a model with all three predictors but without the interaction, and the third one will also omit the continuous predictor variable. Because there are only some cars with five cylinders, these observations are excluded beforehand to simplify the models. After fitting the three models that are all nested (the second is a simpler versions of the first, the third a simpler version of the second model), all three models are compared using the `anova()` function from the `stat` package, see table A.4, which is produced using the `broom` package and some `tidyverse` and `kableExtra` magic (again).

```r
df1_small <- df1 %>% filter(cyl != "5 Cylinders") %>%
    mutate(cyl = droplevels(cyl))

m1 <- lm(hwy ~ year + cyl + displ + year:cyl,
         data = df1_small,
         contrasts = list(year = "contr.treatment",
                          cyl = "contr.treatment"))
m2 <- update(m1, . ~ . - year:cyl)
m3 <- update(m2, . ~ . - displ)

tidy(anova(m1, m2, m3)) %>%
    mutate(across(where(is.numeric), ~ as.character(round(.x, digits = 3)))) %>%
    mutate(across(where(is.character), ~ if_else(is.na(.x), "", .x))) %>%
    rename("Residual $df$" = `df.residual`,
           "RSS" = rss,
           "$df$" = df,
           "Sum of Squares" = sumsq,
           "Statistic" = statistic,
           "$p$-value" = p.value) %>%
    mutate(Model = c(formula(m1)[[3]], formula(m2)[[3]], formula(m3)[[3]])) %>%
    select(Model, everything(), -term) %>%
    kbl(caption = "Analysis of variance table for three linear models",
        booktabs = TRUE, escape = FALSE, align = "lcccrrr",
        centering = TRUE) %>%
    kable_styling(latex_options = c("striped", "HOLD_position",
                                    "scale_down")) %>%
    add_footnote(label = paste0("cyl = cylinder; ",
                                "displ = displacement; ",
                                "$df$ = degrees of freedom; ",
                                "RSS = Residual Sum of Squares"),
                 notation = "none", escape = FALSE)
```

Table A.4.: Analysis of variance table for three linear models

Model	Residual df	RSS	df	Sum of Squares	Statistic	p-value
year + cyl + displ + year:cyl	223	3117.436				
year + cyl + displ	225	3122.895	-2	-5.459	0.195	0.823
year + cyl	226	3364.355	-1	-241.46	17.272	0

cyl = cylinder; displ = displacement; df = degrees of freedom; RSS = Residual Sum of Squares

If you want to know more about the underlying logic behind the statistical part, for example the meaning of analysis of variance in this situation, or what the change of the contrasts here is supposed to do, or what the heck contrasts are to begin with, I can only recommend reading! A good starting point would be the chapter about linear models from the manual *An Introduction into*

R (**introR2022?**), the book *An R Companion to Applied Regression* by (**fox2019?**), or the book *Regression Modeling Strategies* by (**harrell2019?**). There are a lot of books about linear models out there and many of them are dealing with them using R, so it's up to you. If you want to know what the above code does, read it carefully and omit steps to see what element does what!

From the output of table A.4 we can see that in this toy data set case, the smallest model seems to be significantly better fitting the data than the intermediate model with one more predictor, and that there is no significant difference between the second and the most complicated model including the interaction. That means, model three would be better than the others.

But, let's use different packages to produce a side by side table for all three models. The first example, see table A.5, is produced using the `modelsummary` package:

```
modelsummary(
    list("Model 1" = m1, "Model 2" = m2, "Model 3" = m3),
    output = "kableExtra", fmt = 2, booktabs = TRUE,
    escape = FALSE, statistic = NULL, stars = TRUE,
    estimate = "{estimate} ({std.error}){stars}",
    coef_map = c(
        "(Intercept)" = "Constant",
        "year2008" = "Year = 2008",
        "cyl6 Cylinders" = "6 Cylinders",
        "cyl8 Cylinders" = "8 Cylinders",
        "displ" = "Displacement",
        "year2008:cyl6 Cylinders" = "2008 $\\times$ 6 Cylinders",
        "year2008:cyl8 Cylinders" = "2008 $\\times$ 8 Cylinders"
    ),
    gof_map = tribble(
        ~raw, ~clean, ~fmt,
        "nobs", "\\# Observations", 0,
        "adj.r.squared", "$adj.~R^2$", 2,
        "aic", "$AIC$", 2,
        "rmse", "$RMSE$", 2,
        "F", "$F$", 2
    ),
    notes = paste0("{\\\\small \\\\textsl{Notes:}~",
                   "$+~p~\\\\leq~.1$; ",
                   "$*~p~\\\\leq~.05$; ",
                   "$**~p~\\\\leq~.01$; ",
                   "$***~p~\\\\leq~.001$}"),
    title = "Comparison of three linear models using modelsummary") %>%
    kable_styling(latex_options = c("striped", "HOLD_position"))
```

Table A.5.: Comparison of three linear models using modelsummary

	Model 1	Model 2	Model 3
Constant	33.11 (1.26)***	33.03 (1.22)***	28.34 (0.49)***
Year = 2008	1.21 (0.84)	1.22 (0.50)*	1.04 (0.52)*
6 Cylinders	−3.28 (1.03)**	−3.15 (0.90)***	−5.97 (0.61)***
8 Cylinders	−4.39 (1.89)*	−4.74 (1.70)**	−11.35 (0.64)***
Displacement	−2.26 (0.54)***	−2.22 (0.53)***	
2008 × 6 Cylinders	0.39 (1.19)		
2008 × 8 Cylinders	−0.40 (1.25)		
# Observations	230	230	230
$adj.\ R^2$	0.61	0.61	0.58
AIC	1268.25	1264.65	1279.78
$RMSE$	3.68	3.68	3.82
F	59.95	90.48	107.07

Notes: $+\ p\ \leq\ .1;\ *\ p\ \leq\ .05;\ **\ p\ \leq\ .01;\ ***\ p\ \leq\ .001$

Another variant would be to use the `stargazer` package, but here one has to apply several small extra steps to make this work in the bookdown environment we are working in. Firstly, you need to set the code chunk option `results = "asis"`, then you must set the label option in the stargazer function to the exact label you want to cross-reference to, including the "tab:" part, e.g., `label = "tab:m123-stargazer"`, and lastly you must specify the argument `header = FALSE` in the `stargazer` function call to suppress the message "Table built by...". Everything else is a question of using the correct options to tweak the output in the direction you want. Stargazer does have sensible defaults and useful settings, but one drawback is that it cannot be adjusted using `kableExtra` at the end. Have a look at the result in table A.6 and compare this to the corresponding source code to see how it works.

```r
stargazer(m1, m2, m3,
          title = "Comparison of three linear models using stargazer",
          label = "tab:m123-stargazer",
          covariate.labels = c("Constant",
                               "Year = 2008",
                               "6 Cylinders",
                               "8 Cylinders",
                               "Displacement",
                               "2008 $\\times$ 6 Cylinders",
                               "2008 $\\times$ 8 Cylinders"),
          ci = TRUE, ci.level = 0.95, header = FALSE, digits = 2,
          intercept.bottom = FALSE, intercept.top = TRUE,
          table.placement = "H", font.size = "small")
```

Table A.6.: Comparison of three linear models using stargazer

	Dependent variable:		
	hwy		
	(1)	(2)	(3)
Constant	33.11***	33.03***	28.34***
	(30.65, 35.58)	(30.64, 35.42)	(27.39, 29.29)
Year = 2008	1.21	1.22**	1.04**
	(−0.43, 2.85)	(0.24, 2.20)	(0.03, 2.05)
6 Cylinders	−3.28***	−3.15***	−5.97***
	(−5.30, −1.26)	(−4.91, −1.40)	(−7.16, −4.77)
8 Cylinders	−4.39**	−4.74***	−11.35***
	(−8.09, −0.68)	(−8.07, −1.41)	(−12.60, −10.10)
Displacement	−2.26***	−2.22***	
	(−3.31, −1.20)	(−3.27, −1.18)	
2008 × 6 Cylinders	0.39		
	(−1.95, 2.72)		
2008 × 8 Cylinders	−0.40		
	(−2.84, 2.04)		
Observations	230	230	230
R^2	0.62	0.62	0.59
Adjusted R^2	0.61	0.61	0.58
Residual Std. Error	3.74 (df = 223)	3.73 (df = 225)	3.86 (df = 226)
F Statistic	59.95*** (df = 6; 223)	90.48*** (df = 4; 225)	107.07*** (df = 3; 226)
Note:			*p<0.1; **p<0.05; ***p<0.01

If you want to produce an inline result from a specific model you can do this as before by hand, or you can use the package `gtsummary`, which gives you the possibility of directly reporting statistical results using inline functions like `tbl_summary()` and `inline_text()`. As an example, the coefficient for six cylinders in model 3 is $-6.0\ (0.95\%\ CI\ -7.2,\ -4.8;\ p < 0.001)$. Have a look in the source code to see how this is accomplished, but be aware that the argument `pattern` within the function call to `inline_text()` must be provided in this case because we want to render a PDF-file and `gtsummary` is currently mainly aiming at HTML-output. The difference between the numbers in the inline output and table A.6 are due to rounding. If you want to look into the given capabilities of `gtsummary` regarding inline reporting, check out this presentation and the package documentation.

A.4.4. Generalized linear regression models

Generalized linear models (GLMs), first introduced in this comprehensive manner by (**nelder1972?**) are models in which the outcome does not need to belong to a normal distribution, but where the simple linear model is just a special case. For example, the outcome can belong to a $Poisson$ or a $negative - binomial$ distribution (for count data, i.e., only integer values and no value less than zero), to a $Gamma$ distribution (for only positive decimal values, not only integers), to a $binomial$ distribution (for the number of success in a fixed set of trials), or to a $bernoulli$ distribution, which is a simple case of the $binomial$ distribution, where the outcome can only be either one or zero (i.e., yes/no, true/false). There are a lot of possible ways and also user provided packages which make it feasible to model such outcomes using the most appropriate distribution for the data at hand, but the starting point is mainly the function `glm()` from the stats package.

This document can not introduce you to the statistical concepts behind GLMs in general, but I want to give one example of logistic regression to demonstrate the general workflow and to give examples of plotting marginal means. For this purpose, I will use the famous titanic data set (**dawson1995?**), which was further greatly updated and improved by Thomas Cason using the Encyclopedia Titanica to create a new dataset called `titanic3`, which can be obtained here and which can also be obtained through the `Hmisc` package (**Hmisc2022?**).

```
titanic <- read_csv("02-data/titanic3.csv",
    col_types = cols(
        pclass = col_double(),
        survived = col_double(),
        name = col_character(),
        sex = col_character(),
        age = col_double(),
        sibsp = col_double(),
        parch = col_double(),
        ticket = col_character(),
        fare = col_double(),
        cabin = col_character(),
        embarked = col_character(),
        boat = col_character(),
        body = col_double(),
        home.dest = col_character()
    ), lazy = FALSE)
titanic$pclass <- factor(titanic$pclass,
                         levels = c(1,2,3),
                         labels = c("1st", "2nd", "3rd"))
```

The original (non-enhanced) dataset based on data originally collected by the British Board of Trade also comes with the datasets package in R, but provides only the variables class, sex, age, and survived for each person on board of the Titanic. The enhanced version additionally includes variables like age, the ticket fare, the cabin or the place where the person embarked and some others. Next to that, the variable age is changed from a categorical one (child vs. adult) to the actual numeric age in years. The data set comprises 1309 observations.

In the next step, we want to predict if someone survived the Titanic based on the variables pclass, sex and age and the interaction between sex and age. The dependent variable is survived, encoded as 0 or 1, which means the model predicts the mean probability of surviving the Titanic.

```
mlog1 <- glm(survived ~ pclass + sex + age + sex:age,
             data = titanic, family = binomial)
summary(mlog1)
```

```
#> 
#> Call:
#> glm(formula = survived ~ pclass + sex + age + sex:age, family = binomial,
#>     data = titanic)
#> 
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -2.3844  -0.6721  -0.4063   0.7041   2.5440
#> 
#> Coefficients:
#>              Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  2.790834   0.362823   7.692 1.45e-14 ***
#> pclass2nd   -1.424583   0.241513  -5.899 3.67e-09 ***
#> pclass3rd   -2.388177   0.236380 -10.103  < 2e-16 ***
#> sexmale     -1.029755   0.358593  -2.872  0.00408 **
#> age         -0.004084   0.009461  -0.432  0.66600
#> sexmale:age -0.052891   0.012025  -4.398 1.09e-05 ***
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> 
#> (Dispersion parameter for binomial family taken to be 1)
#> 
#>     Null deviance: 1414.62  on 1045  degrees of freedom
#> Residual deviance:  961.92  on 1040  degrees of freedom
#>   (263 Beobachtungen als fehlend gelöscht)
#> AIC: 973.92
#> 
#> Number of Fisher Scoring iterations: 5
```

Again, the default console output is not feasible for a written report, but is printed here to explain some points about the model. The estimated numbers here are on the scale of the linear predictor, that is, they cannot be interpreted as probability values, which is why you see negative numbers. In logistic regression, a linear combination of covariate values (which can take values between $\pm\infty$) is converted to the scale of a probability (between 0 and 1) using the logit link function, which is the inverse of the standard logistic function $\sigma(x) = 1/(1+e^{-x})$, with its definition stated in see equation (A.3).

$$logit(\theta_i) = ln(\frac{\theta_i}{1-\theta_i}) = \beta_0 + \beta_1 \times x_1 + ... + \beta_p \times x_p \tag{A.3}$$

That means that after exponentiating both sides we have the odds, see equation (A.4), which are easier to interpret.

$$odds = \theta_i = \frac{\theta_i}{1-\theta_i} = e^{\beta_0+\beta_1\times x_1+...+\beta_p\times x_p} \tag{A.4}$$

In a logistic regression, the response being modeled is the $log(odds)$ that $Y = 1$. Therefore, the regression coefficients give the change in log(odds) in the response for a unit change in the predictor variable, holding all other predictor variables constant. The odds ratio can then be defined as in equation (A.5).

$$OR = \frac{\text{odds}(x+1)}{\text{odds}(x)} = \frac{e^{\beta_0+\beta_1(x+1)}}{e^{\beta_0+\beta_1 x}} = e^{\beta_1} \tag{A.5}$$

This exponential relationship, see equation (A.5), provides an interpretation for β_1: The odds multiply by e^{β_1} for every 1-unit increase in x. So, if we take the coefficient for being male in the above model (-1.0297546) and take the exponential of it ($\approx$ 0.3570946), we can say that a male passenger has a chance of surviving when compared to a female passenger which is reduced by a factor of 0.3570946. That roughly means that a male passenger only has a 64% lower chance of survival than a female passenger. The probability value for the female passenger is not directly obvious from the above model output, because it serves as a baseline category in this model and is not explicitly named in the above output but is indicated by the intercept.

Using package jtools (which utilizes package huxtable for table formatting), one can get another decent model summary table (see table A.7), but you can as well use the aforementioned packages like modelsummary or stargazer.

```
export_summs(mlog1, digits = 2, error_pos = "right", exp = TRUE,
             error_format = "[{conf.low}, {conf.high}]",
             model.names = "Dependent variable: Survived",
             coefs = c("Intercept" = "(Intercept)",
                       "2nd Passenger Class" = "pclass2nd",
                       "3rd Passenger Class" = "pclass3rd",
                       "Male Passenger" = "sexmale",
                       "Age" = "age",
                       "Male Passenger x Age" = "sexmale:age")) %>%
  huxtable::set_caption('A logistic regression model for the titanic data set using packa
```

Table A.7.: A logistic regression model for the titanic data set using package jtools

	Dependent variable: Survived	
Intercept	16.29 ***	[8.00, 33.18]
2nd Passenger Class	0.24 ***	[0.15, 0.39]
3rd Passenger Class	0.09 ***	[0.06, 0.15]
Male Passenger	0.36 **	[0.18, 0.72]
Age	1.00	[0.98, 1.01]
Male Passenger x Age	0.95 ***	[0.93, 0.97]
N	1046	
AIC	973.92	
BIC	1003.63	
Pseudo R2	0.47	

*** p < 0.001; ** p < 0.01; * p < 0.05.

Another way to produce such a model table is using package gtsummary in conjunction with kableExtra, see table A.8. The only problem here is that you have to “rewrite” the column names because on the way from gtsummary output to kableExtra, the superscripts are lost which renders the footnote a little bit without meaning.

```
tbl_regression(mlog1, exponentiate = TRUE,
               pvalue_fun = ~style_pvalue(.x, digits = 2),
               label = list(pclass ~ "Passenger Class",
                            sex ~ "Sex",
                            age ~ "Age"
                            )) %>%
    add_global_p() %>%  # add global p-value
    add_nevent() %>%    # add number of events of the outcome
    add_q() %>%         # adjusts global p-values for multiple testing
    bold_p() %>%        # bold p-values under a given threshold (default 0.05)
    bold_p(t = 0.05, q = TRUE) %>% # bold q-values under the threshold of 0.05
    bold_labels() %>%
    italicize_levels() %>%
    as_kable_extra(format = "latex", booktabs = TRUE,
                   linesep = "", escape = FALSE,
      caption = 'A logistic regression model for the titanic data set using package gtsum
      col.names = c("Coefficient", "$\\mathrm{Event}~N$", "$OR^{1}$",
                    "$95~\\mathrm{CI}^{1}$",
                    "$p-\\mathrm{value}$",
                    "$q-\\mathrm{value}^{2}$")) %>%
    kable_styling(latex_options = c("striped", "HOLD_position"))
```

Table A.8.: A logistic regression model for the titanic data set using package gtsummary

Coefficient	Event N	OR[1]	95 CI[1]	$p-$value	$q-$value[2]
Passenger Class	427			**<0.001**	**<0.001**
1st		—	—		
2nd		0.24	0.15, 0.38		
3rd		0.09	0.06, 0.14		
Sex	427			**0.004**	**0.005**
female		—	—		
male		0.36	0.18, 0.72		
Age	427	1.00	0.98, 1.01	0.67	0.67
Sex * Age	427			**<0.001**	**<0.001**
*male * Age*		0.95	0.93, 0.97		

[1] OR = Odds Ratio, CI = Confidence Interval

[2] False discovery rate correction for multiple testing

As you have seen, there are a lot of possibilities to generate model and summary tables. It depends on what you need to do, what amount of time you are willing to invest into getting what you want and how satisfied you are with the output generated by one of these helper packages. There are lots of possibilities to tweak the output in more than one way and I would guess that in 90% of the cases, a pre-generated model table is doing the job just great. But, in case it does not, you can always resort to writing down a custom LaTeX table on your own and fill in the cells with inline R chunks that give you exactly the number you want. In this way you can completely decide how your table should look like.

A.4.5. Automated report generation

Another automated way to report a model is using the `report` package (**Makowski2021?**), for example:

```
report(mlog1)
```

We fitted a logistic model (estimated using ML) to predict survived with pclass (formula: survived ~ pclass + sex + age + sex:age). The model's explanatory power is substantial (Tjur's R2 = 0.39). The model's intercept, corresponding to pclass = 1st, is at 2.79 (95% CI [2.09, 3.51], p < .001). Within this model:

- The effect of pclass [2nd] is statistically significant and negative (beta = -1.42, 95% CI [-1.91, -0.96], p < .001; Std. beta = -1.42, 95% CI [-1.91, -0.96])
- The effect of pclass [3rd] is statistically significant and negative (beta = -2.39, 95% CI [-2.86, -1.93], p < .001; Std. beta = -2.39, 95% CI [-2.86, -1.93])
- The effect of sex [male] is statistically significant and negative (beta = -1.03, 95% CI [-1.74, -0.33], p = 0.004; Std. beta = -2.61, 95% CI [-2.96, -2.28])
- The effect of age is statistically non-significant and negative (beta = -4.08e-03, 95% CI [-0.02, 0.01], p = 0.666; Std. beta = -0.06, 95% CI [-0.32, 0.21])
- The interaction effect of age on sex [male] is statistically significant and negative (beta = -0.05, 95% CI [-0.08, -0.03], p < .001; Std. beta = -0.76, 95% CI [-1.11, -0.43])

Standardized parameters were obtained by fitting the model on a standardized version of the dataset. 95% Confidence Intervals (CIs) and p-values were computed using a Wald z-distribution approximation., We fitted a logistic model (estimated using ML) to predict survived with sex (formula: survived ~ pclass + sex + age + sex:age). The model's explanatory power is substantial (Tjur's R2 = 0.39). The model's intercept, corresponding to sex = female, is at 2.79 (95% CI [2.09, 3.51], p < .001). Within this model:

- The effect of pclass [2nd] is statistically significant and negative (beta = -1.42, 95% CI [-1.91, -0.96], p < .001; Std. beta = -1.42, 95% CI [-1.91, -0.96])
- The effect of pclass [3rd] is statistically significant and negative (beta = -2.39, 95% CI [-2.86, -1.93], p < .001; Std. beta = -2.39, 95% CI [-2.86, -1.93])
- The effect of sex [male] is statistically significant and negative (beta = -1.03, 95% CI [-1.74, -0.33], p = 0.004; Std. beta = -2.61, 95% CI [-2.96, -2.28])
- The effect of age is statistically non-significant and negative (beta = -4.08e-03, 95% CI [-0.02, 0.01], p = 0.666; Std. beta = -0.06, 95% CI [-0.32, 0.21])
- The interaction effect of age on sex [male] is statistically significant and negative (beta = -0.05, 95% CI [-0.08, -0.03], p < .001; Std. beta = -0.76, 95% CI [-1.11, -0.43])

Standardized parameters were obtained by fitting the model on a standardized version of the dataset. 95% Confidence Intervals (CIs) and p-values were computed using a Wald z-distribution approximation. and We fitted a logistic model (estimated using ML) to predict survived with age (formula: survived ~ pclass + sex + age + sex:age). The model's explanatory power is substantial (Tjur's R2 = 0.39). The model's intercept, corresponding to age = 0, is at 2.79 (95% CI [2.09, 3.51], p < .001). Within this model:

- The effect of pclass [2nd] is statistically significant and negative (beta = -1.42, 95% CI [-1.91, -0.96], p < .001; Std. beta = -1.42, 95% CI [-1.91, -0.96])
- The effect of pclass [3rd] is statistically significant and negative (beta = -2.39, 95% CI [-2.86, -1.93], p < .001; Std. beta = -2.39, 95% CI [-2.86, -1.93])
- The effect of sex [male] is statistically significant and negative (beta = -1.03, 95% CI [-1.74, -0.33], p = 0.004; Std. beta = -2.61, 95% CI [-2.96, -2.28])
- The effect of age is statistically non-significant and negative (beta = -4.08e-03, 95% CI [-0.02, 0.01], p = 0.666; Std. beta = -0.06, 95% CI [-0.32, 0.21])
- The interaction effect of age on sex [male] is statistically significant and negative (beta = -0.05, 95% CI [-0.08, -0.03], p < .001; Std. beta = -0.76, 95% CI [-1.11, -0.43])

Standardized parameters were obtained by fitting the model on a standardized version of the dataset. 95% Confidence Intervals (CIs) and p-values were computed using a Wald z-distribution approximation.

All the text since the last code chunk was automatically generated by this one function call. have a look at the chunk options to see how this is printed as normal text and not as console output.

A.5. Plotting statistical models

To really understand what a regression model is telling you, you should not just stare at a summary table but you should plot the predictions of a model. There, you can see how the dependent variable is changing when a predictor variable is changing, normally while holding all other predictors constant (e.g., at 0 or the sample mean). That means that you need to calculate the predicted values from the model given different input values for the predictors, calculate the lower and upper confidence intervals according to some level of confidence (normally 95%) and plot all these together. In the case of a GLM one should decide if these numbers should be transformed from the scale of the linear predictor to the scale of the response, for example in case of a logistic regression model, if you should plot the probability instead of the log odds.

There are several packages available that make this task easier. Some of them are `effects` (**fox2019?**), `ggeffects` (**Luedecke2018?**), `jtools` (**jtools?**), `see` (**Luedecke-see2021?**) or `emmeans` (for estimated marginal means, **Lenth2022?**).

A first example using `ggeffects`:

```
eff1 <- ggpredict(mlog1, terms = "pclass")
plot(eff1) + labs(x = "Passenger Class", y = "Probability of Survival",
                  title = element_blank()) +
    scale_y_continuous(labels = scales::percent, limits = c(0,0.65))
```

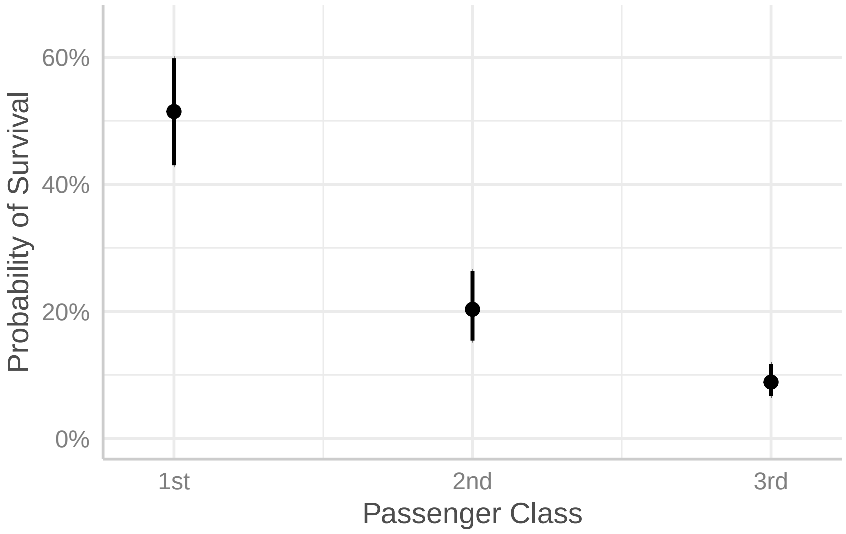


Figure A.1.: Predicted probabilities of survival for each passenger class

Here, you can see another example to visualize the interaction between sex and age, that is, an interaction between a continuous and a categorical predictor.

```
eff2 <- ggpredict(mlog1, terms = c("age [all]", "sex"))
plot(eff2) + labs(x = "Age", y = "Probability of Survival",
                  title = element_blank()) +
    theme(legend.position = "bottom") +
    guides(color = guide_legend(title = "Sex"))
```

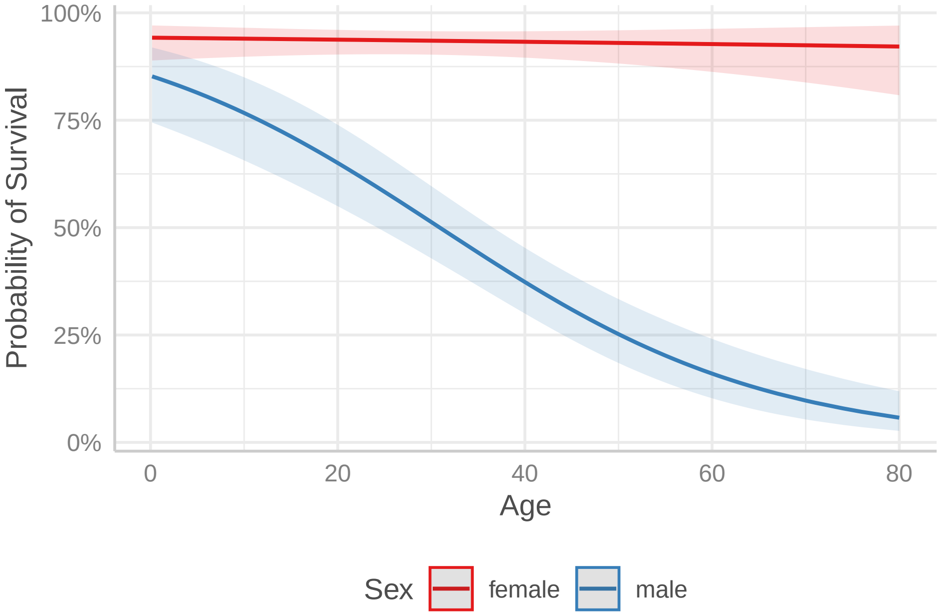


Figure A.2.: Predicted probabilities of survival dependend on sex and age

Coming back to the multiple regression model from above (see section A.4.3), especially model 1 (see table A.5), we can generate one visualization for all the main effects and one for the interaction. In a "real" scientific work you would not present the main effects for themselves in presence of an interaction involving these main effects, but for demonstration purposes, we want to generate four visualizations and put them together in one plot using the package `patchwork` (**Pedersen2022?**). The code for this is placed on the following page, and the figure it produces comes on the next one. By the way, have a look into the chunk options for the last chunk, where both the arguments `fig.cap` and `fig.scap` are used to produce a caption and a short caption for figure A.3, where the former is printed under the figure itself and the latter is used in the list of figures to have a smaller output in the frontmatter.

```r
m1_year <- ggpredict(m1, terms = "year")
m1_cyl <- ggpredict(m1, terms = "cyl")
m1_displ <- ggpredict(m1, terms = "displ [all]")
m1_year_cyl <- ggpredict(m1, terms = c("cyl", "year"))

m1_year_plot <- plot(m1_year) + labs(x = "Year",
    y = "Highway Miles per Gallon", title = element_blank()) +
    scale_y_continuous(limits = c(0, NA),
                       breaks = seq(from = 0, to = 50, by = 5))
m1_cyl_plot <- plot(m1_cyl) + labs(x = "",
    y = "Highway Miles per Gallon", title = element_blank()) +
    scale_y_continuous(limits = c(0, NA),
                       breaks = seq(from = 0, to = 50, by = 5))
m1_displ_plot <- plot(m1_displ) + labs(x = "Displacement",
    y = "Highway Miles per Gallon", title = element_blank()) +
    scale_y_continuous(limits = c(0, NA),
                       breaks = seq(from = 0, to = 50, by = 5))
m1_year_cyl_plot <- plot(m1_year_cyl) + labs(x = "",
    y = "Highway Miles per Gallon", title = element_blank()) +
    guides(color = guide_legend(title = "Year")) +
    scale_y_continuous(limits = c(0, NA),
                       breaks = seq(from = 0, to = 50, by = 5))

m1_year_plot + m1_cyl_plot + m1_displ_plot + m1_year_cyl_plot +
    plot_annotation(tag_levels = 'A') +
    plot_layout(guides = 'collect') &
    theme(legend.position = 'bottom')
```

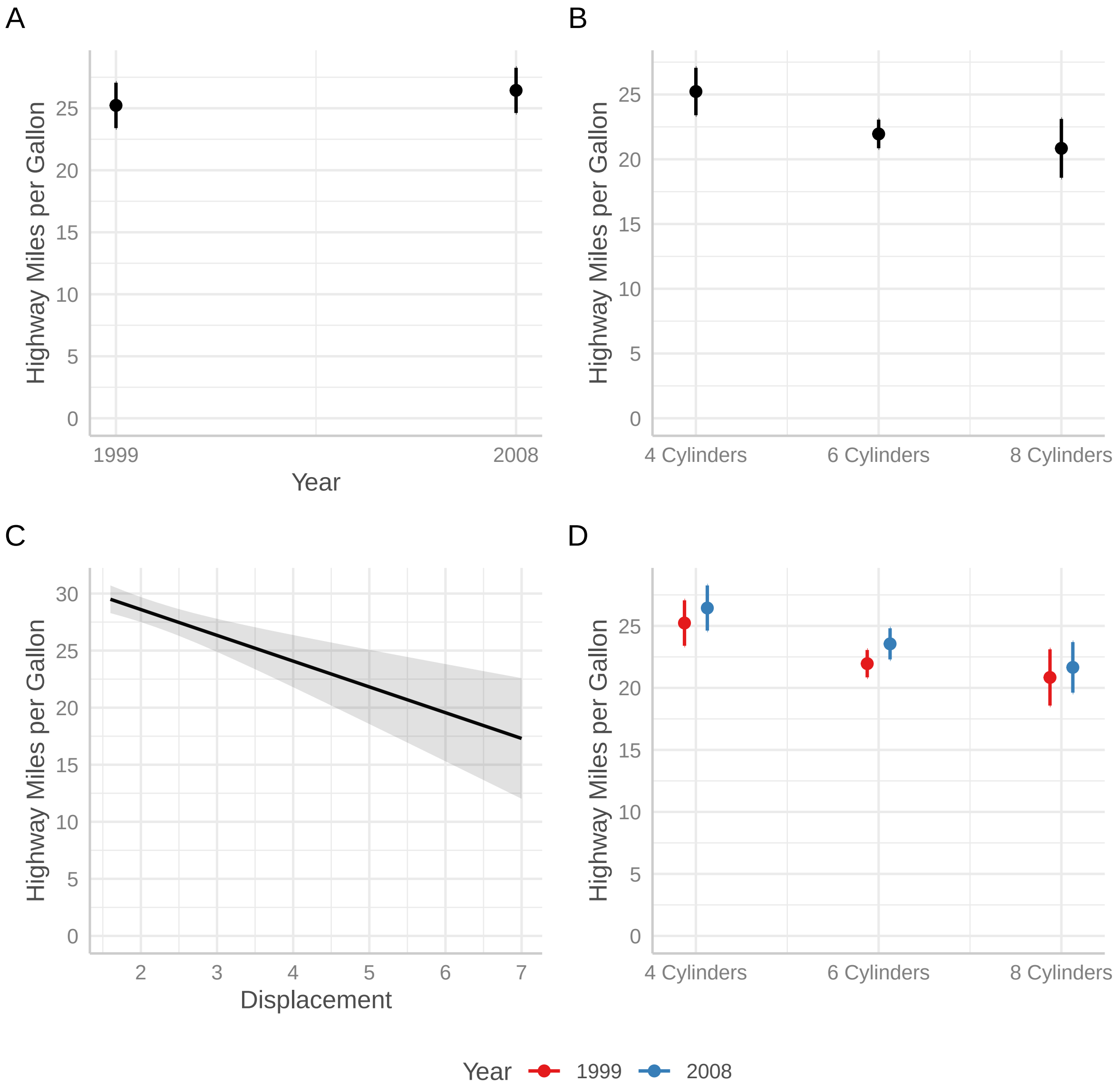


Figure A.3.: One overall plot with four subplots: (A) predictor Year, (B) predictor Cylinders, (C) predictor Displacement, and (D) the interaction between Year x Cylinders

This is the end! Good luck with writing your own report or thesis using this template!

B. Tipps and explanations about RBookdown

B.1. Introduction

B.1.1. Usage

math equation $a^2 + b^2 = c^2$.

Each **bookdown** chapter is an .Rmd file, and each .Rmd file can contain one (and only one) chapter. A chapter *must* start with a first-level heading: `# A good chapter`, and can contain one (and only one) first-level heading.

Use second-level and higher headings within chapters like: `## A short section` or `### An even shorter section`.

The `index.Rmd` file is required, and is also your first book chapter. It will be the homepage when you render the book.

B.1.2. Render book

You can render the HTML version of this example book without changing anything:

1. Find the **Build** pane in the RStudio IDE, and

2. Click on **Build Book**, then select your output format, or select "All formats" if you'd like to use multiple formats from the same book source files.

Or build the book from the R console:

```
bookdown::render_book()
```

To render this example to PDF as a `bookdown::pdf_book`, you'll need to install XeLaTeX. You are recommended to install TinyTeX (which includes XeLaTeX): https://yihui.org/tinytex/.

B.2. Preview book

As you work, you may start a local server to live preview this HTML book. This preview will update as you edit the book when you save individual .Rmd files. You can start the server in a work session by using the RStudio add-in "Preview book", or from the R console:

```
bookdown::serve_book()
```

B.2.1. Hello bookdown

All chapters start with a first-level heading followed by your chapter title, like the line above. There should be only one first-level heading (#) per .Rmd file.

B.2.2. A section

All chapter sections start with a second-level (##) or higher heading followed by your section title, like the sections above and below here. You can have as many as you want within a chapter.

An unnumbered section

Chapters and sections are numbered by default. To un-number a heading, add a `{.unnumbered}` or the shorter `{-}` at the end of the heading, like in this section.

B.3. Cross-references

Cross-references make it easier for your readers to find and link to elements in your book.

B.3.1. Chapters and sub-chapters

There are two steps to cross-reference any heading:

1. Label the heading: `# Hello world {#nice-label}`.

    - Leave the label off if you like the automated heading generated based on your heading title: for example, `# Hello world` = `# Hello world {#hello-world}`.

- To label an un-numbered heading, use: `# Hello world {-#nice-label}` or `{# Hello world .unnumbered}`.

2. Next, reference the labeled heading anywhere in the text using `\@ref(nice-label)`; for example, please see Chapter B.3.

- If you prefer text as the link instead of a numbered reference use: any text you want can go here.

B.3.2. Captioned figures and tables

Figures and tables *with captions* can also be cross-referenced from elsewhere in your book using `\@ref(fig:chunk-label)` and `\@ref(tab:chunk-label)`, respectively.

See Figure B.1.

```
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

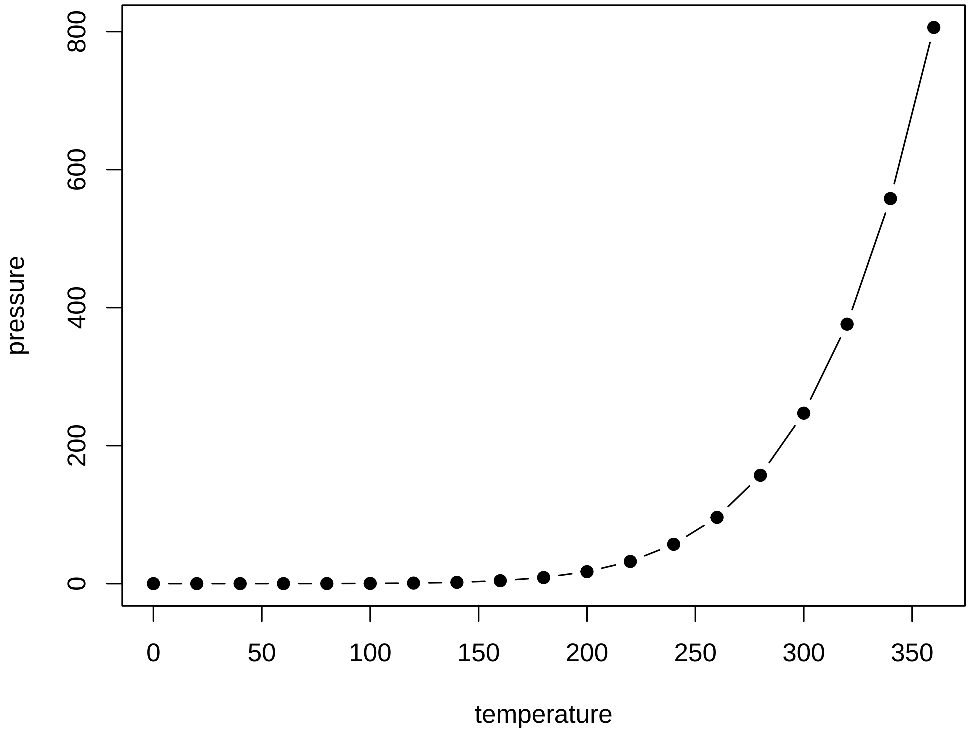


Figure B.1.: Here is a nice figure!

Don't miss Table B.1.

Table B.1.: Here is a nice table!

temperature	pressure
0	0.0002
20	0.0012
40	0.0060
60	0.0300
80	0.0900
100	0.2700
120	0.7500
140	1.8500
160	4.2000
180	8.8000

```
knitr::kable(
  head(pressure, 10), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

B.4. Parts

You can add parts to organize one or more book chapters together. Parts can be inserted at the top of an .Rmd file, before the first-level chapter heading in that same file.

Add a numbered part: `# (PART) Act one {-}` (followed by `# A chapter`)

Add an unnumbered part: `# (PART\*) Act one {-}` (followed by `# A chapter`)

Add an appendix as a special kind of un-numbered part: `# (APPENDIX) Other stuff {-}` (followed by `# A chapter`). Chapters in an appendix are prepended with letters instead of numbers.

B.5. Footnotes

Footnotes are put inside the square brackets after a caret `^[]`. Like this one [1].

[1] This is a footnote.

B.5.1. Citations

Reference items in your bibliography file(s) using `@key`.

For example, we are using the **bookdown** package (Xie 2022) (check out the last code chunk in index.Rmd to see how this citation key was added) in this sample book, which was built on top of R Markdown and **knitr** (**xie2015?**) (this citation was added manually in an external file book.bib). Note that the `.bib` files need to be listed in the index.Rmd with the YAML `bibliography` key.

The RStudio Visual Markdown Editor can also make it easier to insert citations: https://rstudio.github.io/visual-markdown-editing/#/citations

C. Data-set

```r
library(tidyverse) # This includes readr!
library(xtable) # For displaying LaTeX tables
library(modelsummary) # For displaying regression models in tables
library(stargazer) # For displaying regression models in tables
library(jtools) # For displaying regression models in tables
library(kableExtra) # For displaying or changing tables
library(gt) # For displaying tables
library(gtsummary) # For model reporting inline and in tables
library(broom) # For working with statistical models
library(car) # For type-III anova tests
library(report) # For automated text-based model reporting
library(effects) # For working with statistical models / visualize effects
library(ggeffects) # For working with statistical models / visualize effects
library(patchwork) # For putting different visualizations in one figure

dataset <- read_csv("02-data/cost-of-living-2017.csv", lazy= FALSE)
```

```
#> Rows: 511 Columns: 1
#> -- Column specification --------------------------
#> Delimiter: ","
#> chr (1): City    State   Country Cost of Living Plus Rent Ind...
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

D. Discussion

Mal schauen, ob diese Datei funktioniert

```
library(tidyverse)
library(dplyr)
library(stringr)
library(ggplot2)
library(maps)
library(janitor)

#Initial data
rawData <- read_delim("02-data/cost-of-living-2017.csv",
                                  delim = "\t", escape_double = FALSE,
                                  trim_ws = TRUE)
```

```
#> Rows: 511 Columns: 11
#> -- Column specification --------------------------
#> Delimiter: "\t"
#> chr (3): City, State, Country
#> dbl (8): Cost of Living Plus Rent Index, CLI, Rent Index...
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
rawData <- janitor::clean_names(rawData)
rawData
```

```
#> # A tibble: 511 x 11
#>    city  state country cost_~1   cli rent_~2 groce~3 resta~4
#>    <chr> <chr> <chr>     <dbl> <dbl>   <dbl>   <dbl>   <dbl>
#>  1 Zuri~ <NA>  Switze~   109.   150.    66.8    164.    141.
#>  2 Hami~ <NA>  Bermuda   133.   148.   118.     145.    153.
#>  3 Zug   <NA>  Switze~   106.   143.    67.4    148.    143.
#>  4 Gene~ <NA>  Switze~   107.   142.    70.2    147.    139.
#>  5 Basel <NA>  Switze~    97.5  142.    51.5    150.    132.
#>  6 Bern  <NA>  Switze~    91.1  136.    45.3    146.    122.
#>   7 Laus~ <NA>  Switze~    93.6  131.    54.6    137.    128.
#>   8 Reyk~ <NA>  Iceland    93.9  131.    55.9    128.    141.
#>   9 Luga~ <NA>  Switze~    88.6  124.    51.7    121.    128.
#> 10 Stav~ <NA>  Norway     77.8  117.    37.4    108.    143.
#> # ... with 501 more rows, 3 more variables:
#> #   local_purchasing_power_index <dbl>,
#> #   leverage_model_1 <dbl>, leverage_model_2 <dbl>, and
#> #   abbreviated variable names
#> #   1: cost_of_living_plus_rent_index, 2: rent_index,
#> #   3: groceries_index, 4: restaurant_price_index
```

```
#Outliers
sum(is.na(rawData))
```

```
#> [1] 383
```

```
library(tidyverse)
library(dplyr)
library(stringr)
library(ggplot2)
library(maps)
library(janitor)

#Initial data
rawData <- read_delim("02-data/cost-of-living-2017.csv",
                                  delim = "\t", escape_double = FALSE,
                                  trim_ws = TRUE)
```

```
#> Rows: 511 Columns: 11
#> -- Column specification --------------------------
#> Delimiter: "\t"
#> chr (3): City, State, Country
#> dbl (8): Cost of Living Plus Rent Index, CLI, Rent Index...
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

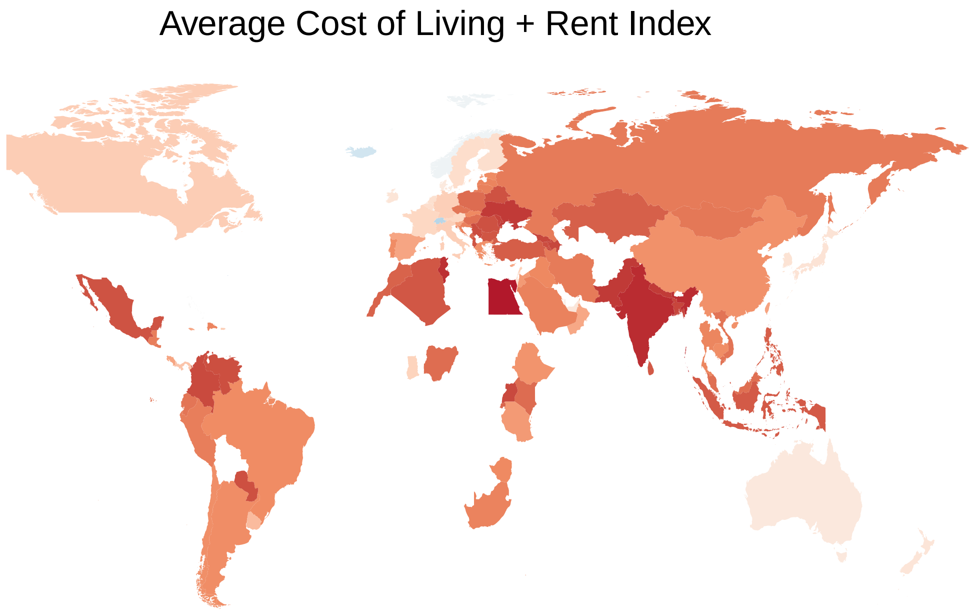
```
rawData <- janitor::clean_names(rawData)
manipulatedData <- rawData[,3:4] %>% group_by(country) %>% summarise(average = mean(cost_

colnames(manipulatedData)[1] <- "region"

world <- map_data("world")
worldSubset <- left_join(manipulatedData, world, by="region")
worldSubset
```

```
#> # A tibble: 66,008 x 7
#>    region  average  long   lat group order subregion
#>    <chr>     <dbl> <dbl> <dbl> <dbl> <int> <chr>
#>  1 Albania    24.4  20.1  42.5     6   770 <NA>
#>  2 Albania    24.4  20.1  42.5     6   771 <NA>
#>  3 Albania    24.4  20.2  42.4     6   772 <NA>
#>  4 Albania    24.4  20.2  42.3     6   773 <NA>
#>  5 Albania    24.4  20.3  42.3     6   774 <NA>
#>  6 Albania    24.4  20.4  42.3     6   775 <NA>
#>  7 Albania    24.4  20.5  42.2     6   776 <NA>
#>  8 Albania    24.4  20.5  42.2     6   777 <NA>
#>  9 Albania    24.4  20.6  42.0     6   778 <NA>
#> 10 Albania    24.4  20.6  41.9     6   779 <NA>
#> # ... with 65,998 more rows
```

```
plain <- theme(
  axis.text = element_blank(),
  axis.line = element_blank(),
  axis.ticks = element_blank(),
  panel.border = element_blank(),
  panel.grid = element_blank(),
  axis.title = element_blank(),
  panel.background = element_rect(fill = "white"),
  plot.title = element_text(hjust = 0.5)
)
worldCLI <- ggplot(data = worldSubset, mapping = aes(x = long, y = lat, group = group)) +
  coord_fixed(1.3) +
  geom_polygon(aes(fill = average)) +
  scale_fill_distiller(palette ="RdBu", direction = 1) + # or direction=1
  ggtitle("Average Cost of Living + Rent Index") +
  plain +
  guides(fill = FALSE)
```

```
#> Warning: `guides(<scale> = FALSE)` is deprecated. Please use
#> `guides(<scale> = "none")` instead.
```

```
worldCLI
```



```r
library(tidyverse)
library(dplyr)
library(stringr)
library(ggplot2)
library(maps)
library(janitor)

#initial country data
dd <- read_delim("02-data/developed_and_developing_countries.csv",
                                  delim = ";", escape_double = FALSE,
                                  trim_ws = TRUE)
```

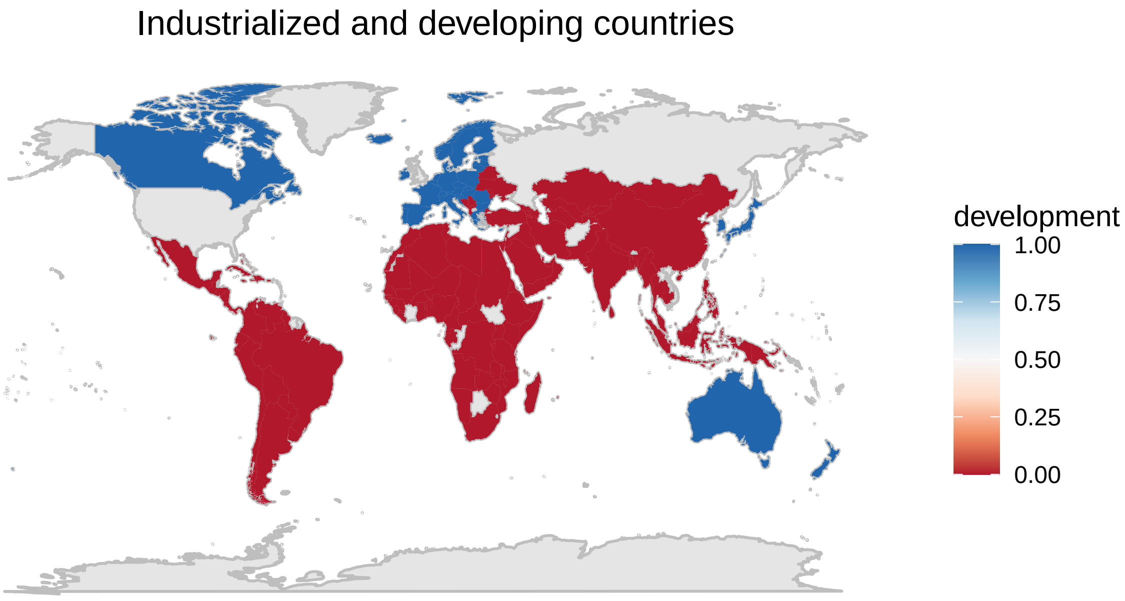
```
#> Rows: 172 Columns: 2
#> -- Column specification --------------------------
#> Delimiter: ";"
#> chr (2): country, category
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
dd <- janitor::clean_names(dd)
dd$category[dd$category == "developed"] <- 1.0
dd$category[dd$category == "developing"] <- 0.0
dd$category <- as.double(dd$category)

dd$category[dd$category == "developing"] <- 0.0
```

```
dd$country[dd$country == "italy"] <- "Italy"

colnames(dd)[1] <- "region"
colnames(dd)[2] <- "development"

world <- map_data("world")
worldSubset <- left_join(dd, world, by="region")

worldSubset
```

```
#> # A tibble: 71,310 x 7
#>    region  development  long   lat group order subregion
#>    <chr>         <dbl> <dbl> <dbl> <dbl> <int> <chr>
#>  1 Austria           1  17.0  48.6   181  9108 <NA>
#>  2 Austria           1  16.9  48.6   181  9109 <NA>
#>  3 Austria           1  16.9  48.6   181  9110 <NA>
#>  4 Austria           1  16.9  48.5   181  9111 <NA>
#>  5 Austria           1  16.9  48.4   181  9112 <NA>
#>  6 Austria           1  16.9  48.4   181  9113 <NA>
#>  7 Austria           1  17.0  48.2   181  9114 <NA>
#>  8 Austria           1  17.1  48.1   181  9115 <NA>
#>  9 Austria           1  17.1  48.0   181  9116 <NA>
#> 10 Austria           1  17.1  48.0   181  9117 <NA>
#> # ... with 71,300 more rows
```

```
plain <- theme(
  axis.text = element_blank(),
  axis.line = element_blank(),
  axis.ticks = element_blank(),
  panel.border = element_blank(),
  panel.grid = element_blank(),
  axis.title = element_blank(),
  panel.background = element_rect(fill = "white"),
  plot.title = element_text(hjust = 0.5)
)
worldDD <- ggplot(data = worldSubset, mapping = aes(x = long, y = lat, group = group)) +
  borders("world", fill="grey90",colour="grey") +
  coord_fixed(1.3) +
  geom_polygon(aes(fill = development)) +
  scale_fill_distiller(palette ="RdBu", direction = 1) + # or direction=1
  ggtitle("Industrialized and developing countries") +
  plain
worldDD
```



```
library(tidyverse)
library(dplyr)
library(stringr)
library(ggplot2)
library(maps)
library(janitor)

#Initial data
rawData <- read_delim("02-data/cost-of-living-2017.csv",
                                  delim = "\t", escape_double = FALSE,
                                  trim_ws = TRUE)
```

```
#> Rows: 511 Columns: 11
#> -- Column specification --------------------------
#> Delimiter: "\t"
#> chr (3): City, State, Country
#> dbl (8): Cost of Living Plus Rent Index, CLI, Rent Index...
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
rawData <- janitor::clean_names(rawData)
rawData
```

```
#> # A tibble: 511 x 11
#>    city  state country cost_~1   cli rent_~2 groce~3 resta~4
#>    <chr> <chr> <chr>     <dbl> <dbl>   <dbl>   <dbl>   <dbl>
#>  1 Zuri~ <NA>  Switze~   109.   150.    66.8    164.    141.
#>  2 Hami~ <NA>  Bermuda   133.   148.   118.     145.    153.
#>  3 Zug   <NA>  Switze~   106.   143.    67.4    148.    143.
#>  4 Gene~ <NA>  Switze~   107.   142.    70.2    147.    139.
#>  5 Basel <NA>  Switze~    97.5  142.    51.5    150.    132.
#>  6 Bern  <NA>  Switze~    91.1  136.    45.3    146.    122.
#>  7 Laus~ <NA>  Switze~    93.6  131.    54.6    137.    128.
#>  8 Reyk~ <NA>  Iceland    93.9  131.    55.9    128.    141.
#>  9 Luga~ <NA>  Switze~    88.6  124.    51.7    121.    128.
#> 10 Stav~ <NA>  Norway     77.8  117.    37.4    108.    143.
#> # ... with 501 more rows, 3 more variables:
#> #   local_purchasing_power_index <dbl>,
#> #   leverage_model_1 <dbl>, leverage_model_2 <dbl>, and
#> #   abbreviated variable names
#> #   1: cost_of_living_plus_rent_index, 2: rent_index,
#> #   3: groceries_index, 4: restaurant_price_index
```

```
dd <- read_delim("02-data/developed_and_developing_countries.csv",
                                  delim = ";", escape_double = FALSE,
                                  trim_ws = TRUE)
```

```
#> Rows: 172 Columns: 2
#> -- Column specification --------------------------
#> Delimiter: ";"
#> chr (2): country, category
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

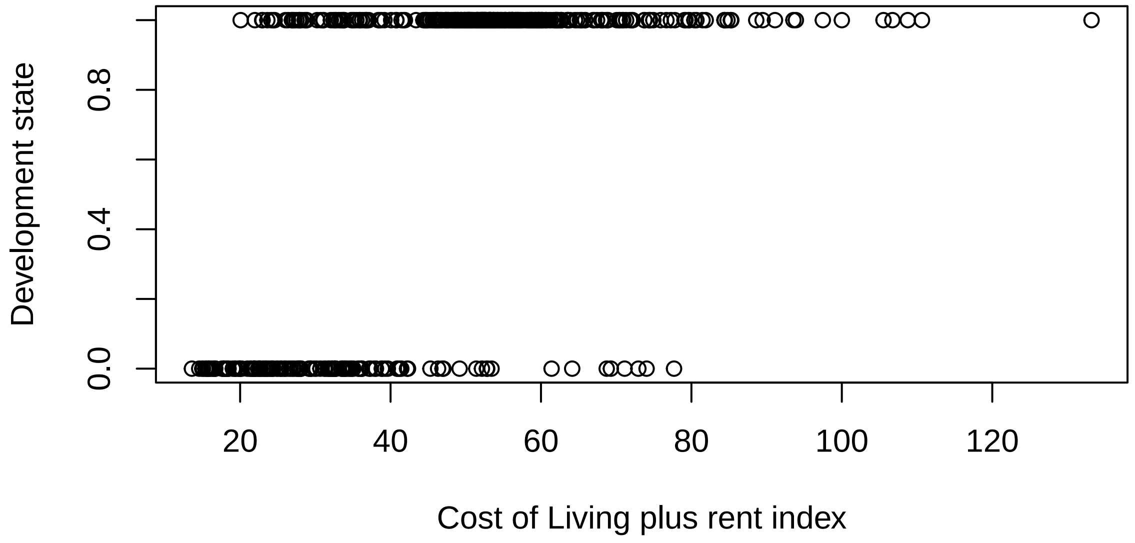
```
dd <- janitor::clean_names(dd)

colnames(dd)[2] <- "development"
dd$development[dd$development == "developed"] <- 1.0
dd$development[dd$development == "developing"] <- 0.0
dd$development <- as.double(dd$development)

dd$development[dd$development == "developing"] <- 0.0
dd$country[dd$country == "italy"] <- "Italy"
dd
```

```
#> # A tibble: 172 x 2
#>    country    development
#>    <chr>            <dbl>
#>  1 Austria              1
#>  2 Belgium                  1
#>  3 Denmark                  1
#>  4 Finland                  1
#>  5 France                   1
#>  6 Germany                  1
#>  7 Greece                   1
#>  8 Ireland                  1
#>  9 Italy                    1
#> 10 Luxembourg               1
#> # ... with 162 more rows
```

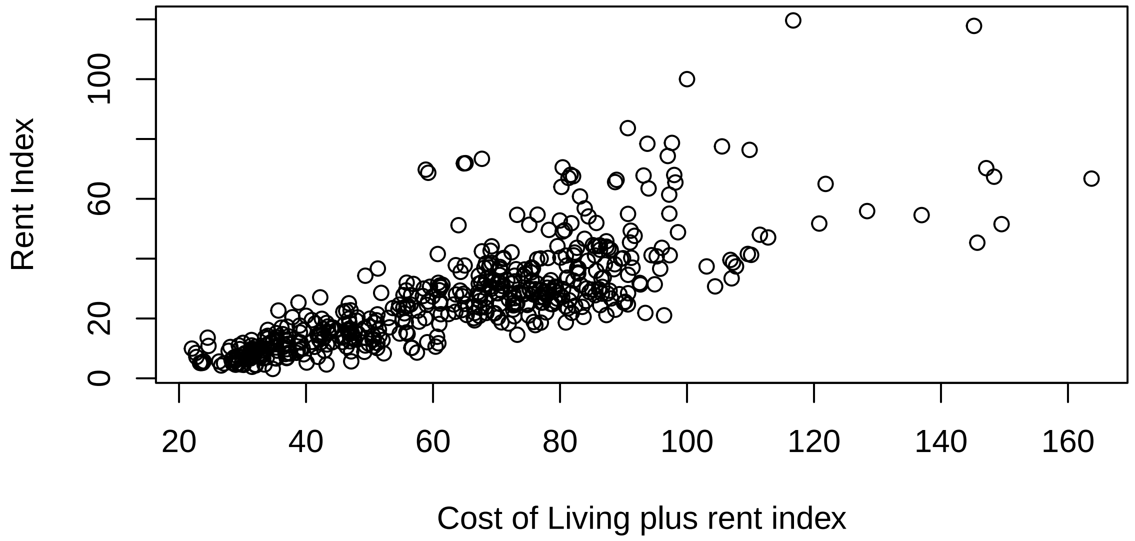
```
data <- left_join(dd, rawData, by="country")
data
```

```
#> # A tibble: 566 x 12
#>    country devel~1 city  state cost_~2   cli rent_~3 groce~4
#>    <chr>     <dbl> <chr> <chr>   <dbl> <dbl>   <dbl>   <dbl>
#>  1 Austria       1 Linz  <NA>     58.8  88.4    28.3    86.6
#>  2 Austria       1 Vien~ <NA>     54.2  76.0    31.7    72.4
#>  3 Austria       1 Graz  <NA>     49.4  75.0    23.0    72.6
#>  4 Belgium       1 Antw~ <NA>     56.9  86.2    26.6    79.6
#>  5 Belgium       1 Gent  <NA>     55.8  85.3    25.3    77.1
#>  6 Belgium       1 Leuv~ <NA>     56.5  84.4    27.8    85.5
#>  7 Belgium       1 Brus~ <NA>     58.8  82.7    34.2    74.2
#>  8 Denmark       1 Cope~ <NA>     73.7  97.1    49.6    78.2
#>  9 Denmark       1 Aalb~ <NA>     59.2  92.1    25.3    77.3
#> 10 Denmark       1 Arhus <NA>     59.8  87.1    31.6    67.2
#> # ... with 556 more rows, 4 more variables:
#> #   restaurant_price_index <dbl>,
#> #   local_purchasing_power_index <dbl>,
#> #   leverage_model_1 <dbl>, leverage_model_2 <dbl>, and
#> #   abbreviated variable names 1: development,
#> #   2: cost_of_living_plus_rent_index, 3: rent_index,
#> #   4: groceries_index
```

```
plot(data$cost_of_living_plus_rent_index,data$development
     ,xlab = "Cost of Living plus rent index", ylab = "Development state")
```



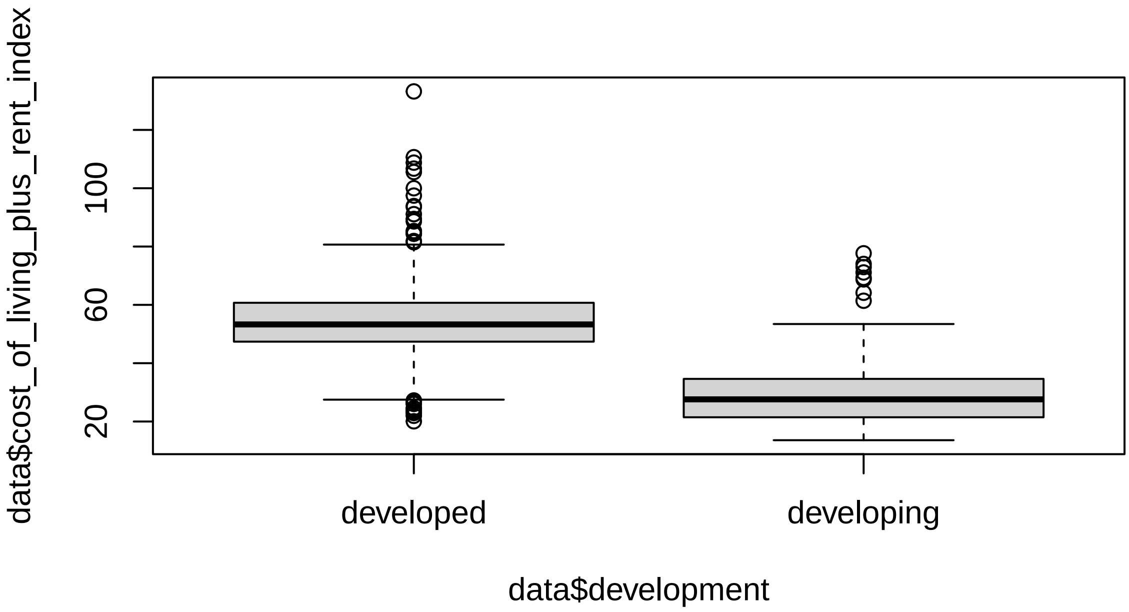
```
plot(data$groceries_index,data$rent_index
          ,xlab = "Cost of Living plus rent index", ylab = "Rent Index")
```



```
data$development[data$development == 1.0] <- "developed"
data$development[data$development == 0.0] <- "developing"
data
```

```
#> # A tibble: 566 x 12
#>    country devel~1 city  state cost_~2   cli rent_~3 groce~4
#>    <chr>   <chr>   <chr> <chr>   <dbl> <dbl>   <dbl>   <dbl>
#>  1 Austria develo~ Linz  <NA>     58.8  88.4    28.3    86.6
#>  2 Austria develo~ Vien~ <NA>     54.2  76.0    31.7    72.4
#>  3 Austria develo~ Graz  <NA>     49.4  75.0    23.0    72.6
#>  4 Belgium develo~ Antw~ <NA>     56.9  86.2    26.6    79.6
#>  5 Belgium develo~ Gent  <NA>     55.8  85.3    25.3    77.1
#>  6 Belgium develo~ Leuv~ <NA>     56.5  84.4    27.8    85.5
#>  7 Belgium develo~ Brus~ <NA>     58.8  82.7    34.2    74.2
#>  8 Denmark develo~ Cope~ <NA>     73.7  97.1    49.6    78.2
#>  9 Denmark develo~ Aalb~ <NA>     59.2  92.1    25.3    77.3
#> 10 Denmark develo~ Arhus <NA>     59.8  87.1    31.6    67.2
#> # ... with 556 more rows, 4 more variables:
#> #   restaurant_price_index <dbl>,
#> #   local_purchasing_power_index <dbl>,
#> #   leverage_model_1 <dbl>, leverage_model_2 <dbl>, and
#> #   abbreviated variable names 1: development,
#> #   2: cost_of_living_plus_rent_index, 3: rent_index,
#> #   4: groceries_index
```

```
boxplot(data$cost_of_living_plus_rent_index~data$development)
```


data$cost_of_living_plus_rent_index
developed
developing
data$development