Tutorial_7_own_solutions_markdown

true

Tutorial 7 Business Analytics - own solutions

R Markdown - The definitive guide (online book): https://bookdown.org/yihui/rmarkdown/ Tidymodels Documentation: https://tidymodels.org

```
#install.packages("tidymodels")

## checking for updates of the tidyverse package
#tidyverse_update()

#install.packages("GGally")
```

Loading Packages

```
library(tidyverse) # collection of packages for data analysis including: ggplot2 (data visualization pa
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## Warning: package 'ggplot2' was built under R version 3.6.2
```

```
## Warning: package 'tibble' was built under R version 3.6.2
```

```
## Warning: package 'tidyr' was built under R version 3.6.2
```

```
## Warning: package 'readr' was built under R version 3.6.2
```

```
## Warning: package 'purrr' was built under R version 3.6.2
```

```
## Warning: package 'dplyr' was built under R version 3.6.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate) # tools to parse and manipuulate dates
```

```
## Warning: package 'lubridate' was built under R version 3.6.2
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(GGally) # ggplot2 extension (for a heat map plot)
```

```
## Warning: package 'GGally' was built under R version 3.6.2
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
library(graphics) # base graphics (e.g. histogram)

library(stats) # computing distribution functions, distribution quantiles; random number generation

library(car) # companion to applied regression package
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```r
library(lmtest) # testing linear regression models package
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(aod) # analysis of overdispersed data package

library(tidymodels) # collection of packages for modeling and machine learning
```

```
## Warning: package 'tidymodels' was built under R version 3.6.2
```

```
## -- Attaching packages ------------------------------------- tidymodels 0.1.2 --
```

```
## v broom     0.7.2      v recipes   0.1.15
## v dials     0.0.9      v rsample   0.0.8
## v infer     0.5.3      v tune      0.1.2
## v modeldata 0.1.0      v workflows 0.2.1
## v parsnip   0.1.4      v yardstick 0.0.7
```

```
## Warning: package 'broom' was built under R version 3.6.2
```

```
## Warning: package 'dials' was built under R version 3.6.2
```

```
## Warning: package 'infer' was built under R version 3.6.2
```

```
## Warning: package 'modeldata' was built under R version 3.6.2
```

```
## Warning: package 'parsnip' was built under R version 3.6.2
```

```
## Warning: package 'recipes' was built under R version 3.6.2
```

```
## Warning: package 'rsample' was built under R version 3.6.2

## Warning: package 'tune' was built under R version 3.6.2

## Warning: package 'workflows' was built under R version 3.6.2

## Warning: package 'yardstick' was built under R version 3.6.2

## -- Conflicts ---------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x car::recode()     masks dplyr::recode()
## x car::some()       masks purrr::some()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
```

```
# getting the current WD
getwd()
```

```
## [1] "/Users/Manu/Desktop/TUM_Master_Mgt_Technology/TUM_WS_20/Business Analytics/Tutorials/Tutorial 7
```

```
# changing the WD
##setwd("/Users/Manu/Desktop/TUM Master Mgt & Technology/TUM WS 20/Business Analytics/Tutorials/Tutoria
```

Controlling the size of plots/images

```
# The Markdown syntax is: ![caption](path/to/image)
```

The size of plots made in R can be controlled by the chunk option **fig.width** and **fig.height** (in inches). Equivalently, you can use the **fig.dim** option to specify the width and height in a numeric vector of length 2, e.g., fig.dim = c(8, 6) means fig.width = 8 and fig.height = 6. These options set the physical size of plots, and you can choose to display a different size in the output using chunk options **out.width** and **out.height**, e.g., out.width = "50%".

Exercise 7.1

The data set (raw_data.csv) contains data from an online shop. Table 1 describes the attributes and values.

a) Load raw_data.csv and rename all attributes to match the "description" column in Table 1. Hint: read_delim(), rename()

```
# read_delim() is amore general version of read_csv
raw_data <- read_delim("raw_data.csv",delim = ";")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   ID = col_double(),
##   od = col_date(format = ""),
##   dd = col_character(),
##   size = col_character(),
##   price = col_double(),
##   tax = col_double(),
##   a6 = col_double(),
##   a7 = col_character(),
##   a8 = col_double(),
##   a9 = col_double()
```

Table 1: Attributes

Attribute	Description	Comment
ID	ID	
od	order_date	
dd	delivery_date	
size	size	ordinal: "S"<"M"<"L"<"XL"<"XXL"<"XXXL"
price	price	
tax	tax	
a6	salutation	nominal 2: "Company" 3: "Mr." 4: "Mrs."
a7	date_of_birth	
a8	state	nominal: 1: "BW" 2: "BY" 3: "BE" 4: "BB" 5: "HB" 6: "HH" 7: "HE" 8: "MV" 9: "NI" 10: "NW" 11: "RP" 12: "SL" 13: "SN" 14: "ST" 15: "SH" 16: "TH"
a9	return_shipment	0: “no” 1: “yes”

Figure 1: Attributes table

```
## )
```

```
head(raw_data)
```

```
## # A tibble: 6 x 10
##       ID od         dd         size  price   tax    a6 a7            a8    a9
##    <dbl> <date>     <chr>      <chr> <dbl> <dbl> <dbl> <chr>      <dbl> <dbl>
## 1 103939 2012-06-26 ?          l      29.9  5.68     4 1900-11-19     8     0
## 2  11788 2012-04-10 2012-04-11 l      49.9  9.48     4 1953-05-24     7     1
## 3  96553 2012-06-20 2012-06-21 m      59.9 11.4      4 1954-04-10    11     0
## 4 100950 2012-06-24 2012-06-26 l      69.9 13.3      4 1965-01-11     7     0
## 5 223454 2012-09-22 1990-12-31 xxl    89.9 17.1      4 1962-05-11    10     0
## 6  63989 2012-05-22 2012-05-23 xxl    39.9  7.58     4 1955-05-23    15     1
```

```
# getting all the feature names
names(raw_data)
```

```
##  [1] "ID"    "od"    "dd"    "size"  "price" "tax"   "a6"    "a7"    "a8"
## [10] "a9"
```

```
# renaming the column names:

colnames(raw_data) <-  c("ID","order_date","delivery_date", "size", "price","tax","salutation","date_of

names(raw_data)
```

```
##  [1] "ID"              "order_date"      "delivery_date"   "size"
##  [5] "price"           "tax"             "salutation"      "date_of_birth"
##  [9] "state"           "return_shipment"
```

b) Correct the data types for all nominal attributes and assign the corresponding labels from the "comment" column in Table 1. Hint: mutate(), factor()

```
# inspecting dimensionality, features names, data types and the first few values
str(raw_data)
```

```
## tibble [300 x 10] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ID             : num [1:300] 103939 11788 96553 100950 223454 ...
##  $ order_date     : Date[1:300], format: "2012-06-26" "2012-04-10" ...
##  $ delivery_date  : chr [1:300] "?" "2012-04-11" "2012-06-21" "2012-06-26" ...
##  $ size           : chr [1:300] "l" "l" "m" "l" ...
##  $ price          : num [1:300] 29.9 49.9 59.9 69.9 89.9 39.9 39.9 59.9 69.9 39.9 ...
##  $ tax            : num [1:300] 5.68 9.48 11.38 13.28 17.08 ...
##  $ salutation     : num [1:300] 4 4 4 4 4 4 4 2 4 4 ...
##  $ date_of_birth  : chr [1:300] "1900-11-19" "1953-05-24" "1954-04-10" "1965-01-11" ...
##  $ state          : num [1:300] 8 7 11 7 10 15 10 1 1 11 ...
##  $ return_shipment: num [1:300] 0 1 0 0 0 1 1 0 1 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ID = col_double(),
##   ..   od = col_date(format = ""),
##   ..   dd = col_character(),
##   ..   size = col_character(),
##   ..   price = col_double(),
##   ..   tax = col_double(),
##   ..   a6 = col_double(),
##   ..   a7 = col_character(),
```

```
##   ..   a8 = col_double(),
##   ..   a9 = col_double()
##   .. )
```

The attributes/features "salutation", "state" and "return_shipment" are nominal attributes (simple distinction and no order).

```
# getting the different values of the salutation attribute
unique(raw_data$salutation);
```

```
## [1] 4 2 3
```

```
unique(raw_data$state);
```

```
##  [1]  8  7 11 10 15  1  3  2  6 13  5 12  9 14  4 16
```

```
unique(raw_data$return_shipment)
```

```
## [1] 0 1
```

Nominal attributes: Values of Nominal attributes represent some category or state (simple distinction) and that's why nominal attributes also referred as categorical attributes and there is no order (rank, position) among values of a nominal attribute. Example : Binary Attributes : Binary data have only 2 values/states.

```
# summary stats for all attributes
summary(raw_data)
```

```
##        ID           order_date         delivery_date          size
##  Min.   :  2292   Min.   :2012-04-02   Length:300         Length:300
##  1st Qu.:108790   1st Qu.:2012-06-27   Class :character   Class :character
##  Median :252054   Median :2012-10-16   Mode  :character   Mode  :character
##  Mean   :247748   Mean   :2012-10-17
##  3rd Qu.:379603   3rd Qu.:2013-02-06
##  Max.   :480356   Max.   :2013-03-31
##
##      price             tax          salutation   date_of_birth
##  Min.   :  9.90   Min.   : 1.88   Min.   :2.00   Length:300
##  1st Qu.: 29.90   1st Qu.: 5.68   1st Qu.:4.00   Class :character
##  Median : 49.90   Median : 9.48   Median :4.00   Mode  :character
##  Mean   : 56.33   Mean   :10.70   Mean   :3.98
##  3rd Qu.: 69.90   3rd Qu.:13.28   3rd Qu.:4.00
##  Max.   :359.00   Max.   :68.21   Max.   :4.00
##  NA's   :6        NA's   :6
##      state       return_shipment
##  Min.   : 1.00   Min.   :0.0000
##  1st Qu.: 2.00   1st Qu.:0.0000
##  Median : 8.00   Median :0.0000
##  Mean   : 7.22   Mean   :0.4567
##  3rd Qu.:10.00   3rd Qu.:1.0000
##  Max.   :16.00   Max.   :1.0000
##
```

```
# the mutate() function creates, modifies or deletes columns

# transforming the salutation variable so that we have 3 different classes/categories using the factor

# levels are the catagory/class values that the data vector takes on now, labels contains the new label
raw_data <- mutate(raw_data, salutation = factor(raw_data$salutation, levels =c(2,3,4), labels = c("Comp
```

```
head(raw_data$salutation)
```

```
## [1] Mrs. Mrs. Mrs. Mrs. Mrs. Mrs.
## Levels: Company Mr. Mrs.
```

```
# transforming the other two nominal attributes:
raw_data <- mutate(raw_data, return_shipment = factor(raw_data$return_shipment, levels =c(0,1), labels =

head(raw_data$return_shipment);
```

```
## [1] No  Yes No  No  No  Yes
## Levels: No Yes
```

```
raw_data <- mutate(raw_data, state = factor(raw_data$state, labels = c("BW", "BY", "BE", "BB", "HB", "H

head(raw_data$state)
```

```
## [1] MV HE RP HE NW SH
## Levels: BW BY BE BB HB HH HE MV NI NW RP SL SN ST SH TH
```

```
summary(raw_data);
```

```
##        ID           order_date         delivery_date          size
##  Min.   :  2292   Min.   :2012-04-02   Length:300         Length:300
##  1st Qu.:108790   1st Qu.:2012-06-27   Class :character   Class :character
##  Median :252054   Median :2012-10-16   Mode  :character   Mode  :character
##  Mean   :247748   Mean   :2012-10-17
##  3rd Qu.:379603   3rd Qu.:2013-02-06
##  Max.   :480356   Max.   :2013-03-31
##
##      price             tax          salutation  date_of_birth          state
##  Min.   :  9.90   Min.   : 1.88   Company:  1   Length:300         NW     :83
##  1st Qu.: 29.90   1st Qu.: 5.68   Mr.    :  4   Class :character   BW     :40
##  Median : 49.90   Median : 9.48   Mrs.   :295   Mode  :character   BY     :38
##  Mean   : 56.33   Mean   :10.70                                    MV     :30
##  3rd Qu.: 69.90   3rd Qu.:13.28                                    RP     :22
##  Max.   :359.00   Max.   :68.21                                    HE     :20
##  NA's   :6        NA's   :6                                        (Other):67
##  return_shipment
##  No :163
##  Yes:137
##
##
##
##
##
```

```
head(raw_data)
```

```
## # A tibble: 6 x 10
##       ID order_date delivery_date size  price   tax salutation date_of_birth
##    <dbl> <date>     <chr>         <chr> <dbl> <dbl> <fct>      <chr>
## 1 103939 2012-06-26 ?             l      29.9  5.68 Mrs.       1900-11-19
## 2  11788 2012-04-10 2012-04-11    l      49.9  9.48 Mrs.       1953-05-24
```

```
## 3  96553 2012-06-20 2012-06-21    m      59.9 11.4  Mrs.       1954-04-10
## 4 100950 2012-06-24 2012-06-26    l      69.9 13.3  Mrs.       1965-01-11
## 5 223454 2012-09-22 1990-12-31    xxl    89.9 17.1  Mrs.       1962-05-11
## 6  63989 2012-05-22 2012-05-23    xxl    39.9  7.58 Mrs.       1955-05-23
## # ... with 2 more variables: state <fct>, return_shipment <fct>
```

c) Correct the data type for the ordinal attribute size and assign the corresponding labels from the "comment" column in Table 1. Hint: toupper(), table()

```
# tabulate the absolute frequencies of the attribute size
table(raw_data$size);
```

```
##
##    l    L    m    M    s    S   xl   XL  xxl  XXL xxxl
##   44   37   42   26   11   10   38   35   33   22    2
```

```
# display unique values of the attribute size
unique(raw_data$size)
```

```
##  [1] "l"    "m"    "xxl"  "xl"   "S"    "XL"   "L"    "s"    "XXL"  "M"
## [11] "xxxl"
```

```
# size attribute values are converted form lower case to upper case with the toupper() function
raw_data <- mutate(raw_data,size = toupper(size))

head(raw_data$size)
```

```
## [1] "L"   "L"   "M"   "L"   "XXL" "XXL"
```

d) Correct the data types for all date attributes. Create separate attributes for weekday, year, month, day, and quarter of "order date". Hint: mutate(), across(), as_date() from package "lubridate"

order date, delivery_date and date_of_birth should be dates but only order date was parsed correctly, because the others contain '?'. lubridate::as_date provides better date parsing than R's standard as.date. mutate can be used with across function to perform similar mutations across multiple columns.

```
# transforming the three data attributes
##raw_data <- mutate(raw_data,across(.cols = c(order_date, delivery_date, date_of_birth), .fns = as_dat

raw_data <- raw_data %>% mutate(across(.cols = c(order_date, delivery_date, date_of_birth), .fns = as_da
```

```
## Warning: Problem with `mutate()` input `..1`.
## i  19 failed to parse.
## i Input `..1` is `across(.cols = c(order_date, delivery_date, date_of_birth), .fns = as_date)`.
```

```
## Warning: 19 failed to parse.
```

```
## Warning: Problem with `mutate()` input `..1`.
## i  24 failed to parse.
## i Input `..1` is `across(.cols = c(order_date, delivery_date, date_of_birth), .fns = as_date)`.
```

```
## Warning: 24 failed to parse.
```

e) Find missing values (only NA), fill missing prices/tax with averages or remove the instances. Hint: mutate(), summarize(), across(), if_else(), na.omit()

```
# boolean test for missing values of each value in the data set
##is.na(raw_data)

# sum and the percentage of missing values in the data set:
sum(is.na(raw_data)); # 12 missing values
```

```
## [1] 55
```

```
mean(is.na(raw_data)) # 4 percent of the data set are missing
```

```
## [1] 0.01833333
```

rewrite count NA cells by attribute function without pipe operator - unresolved! Purpose of the pipe operator: pipe operators can make code look cleaner and they make it easier to keep track of the flow of operations.

```
# Find missing values (only NA)
#summarize_all(sum(mutate_all(is.na(.)))) # rewritten version

raw_data %>% mutate_all(is.na) %>% summarize_all(sum) #original version
```

```
## # A tibble: 1 x 10
##      ID order_date delivery_date  size price   tax salutation date_of_birth
##   <int>      <int>         <int> <int> <int> <int>      <int>         <int>
## 1     0          0            19     0     6     6          0            24
## # ... with 2 more variables: state <int>, return_shipment <int>
```

The syntax of the "if_else()" function is: if_else(condition, true, false, missing = NULL)

```
# Filling the NA values ofthe attributes "price" and "tax" with the respective mean value of the respec

raw_data <- mutate(raw_data, tax = if_else(is.na(tax), mean(tax, na.rm=TRUE), tax), price = if_else(is.
```

```
# checking for missing values again:
sum(is.na(raw_data))
```

```
## [1] 43
```

```
# Removing the instances/data points with NA values
raw_data <- na.omit(raw_data)

#raw_data %>% mutate_all(is.na) %>% summarize_all(sum)

summarize(raw_data, across(.cols = everything(), .fns = ~sum(is.na(.))))
```

```
## # A tibble: 1 x 10
##      ID order_date delivery_date  size price   tax salutation date_of_birth
##   <int>      <int>         <int> <int> <int> <int>      <int>         <int>
## 1     0          0             0     0     0     0          0             0
## # ... with 2 more variables: state <int>, return_shipment <int>
```

f) Calculate a new attribute "delivery time" as the difference of order and delivery date in days. Inspect the values for errors and set the value to "NA" for corresponding instances. Hint: Negative delivery time is impossible.

```
raw_data[c("order_date","delivery_date")]
```

```
## # A tibble: 259 x 2
##    order_date delivery_date
```

```
##    <date>     <date>
##  1 2012-04-10 2012-04-11
##  2 2012-06-20 2012-06-21
##  3 2012-06-24 2012-06-26
##  4 2012-09-22 1990-12-31
##  5 2012-05-22 2012-05-23
##  6 2012-09-06 2012-09-07
##  7 2012-08-25 2012-08-29
##  8 2013-01-13 2013-01-21
##  9 2013-03-29 2013-04-11
## 10 2012-05-17 2012-05-22
## # ... with 249 more rows
```

```
# creating a new attribute "delivery time" and format it as a numeric object
raw_data$delivery_time <- as.numeric(raw_data$delivery_date - raw_data$order_date, unit="days")

raw_data$delivery_time;
```

```
##   [1]     1     1     2 -7936     1     1     4     8    13     5     4     1
##  [13]     2    50    58    33    16     2     5     1    43     1     6     2
##  [25]    39     4     3    67     1     3     2     3    76    35     2     2
##  [37]     1     3     3     7     3     5     1     2     5     2     1     4
##  [49]     5     1     2     4 -7855     4     3     8     1     1     4     2
##  [61]    82     2     5     2     3     1     9     2     4    51     2     2
##  [73]     1     1     2    48     3 -7843    14     1    17    13     1     5
##  [85]    41     3    10     2     3     1     3    40     2     1     3    29
##  [97]     3     3     2     1     1    80     5    43     1     4     2     4
## [109]    10    47     3    46     1     3     3     1     1     4    18     4
## [121] -7843    38     3     1    32     2    55    41    26    33    41     9
## [133]    19     3     4     2     4    35     2     3    29     2    16     2
## [145]     2     4     1     4     4     4     4     5    91     1     2     1
## [157]     6     3     4     1     2     2     1    21     4     3     4     5
## [169]     2     3     1     4     1     3     3     3     4     9     2     4
## [181]    27     3    37     5    98     2     4     2     3     3     6     3
## [193]     1     3     1     1     3     2     1     3    96    38     2     1
## [205]     4     2     4     4    22    26    42     3     2 -8052     4     5
## [217]     2     1     3     2    36     5     3    12     3     6     2     5
## [229]    33     2     5     5     4     2     2     3     2     2     4     2
## [241]    23     4     3     2     3     2     8     9     1     8    33     1
## [253]     2     2     1   111     2     4     2
```

```
class(raw_data$delivery_time)
```

```
## [1] "numeric"
```

```
# setting negative values for delivery time to NA
raw_data$delivery_time <- ifelse(raw_data$delivery_time >= 0, raw_data$delivery_time,"NA")

raw_data$delivery_time;
```

```
##   [1] "1"   "1"   "2"   "NA"  "1"   "1"   "4"   "8"   "13"  "5"   "4"   "1"
##  [13] "2"   "50"  "58"  "33"  "16"  "2"   "5"   "1"   "43"  "1"   "6"   "2"
##  [25] "39"  "4"   "3"   "67"  "1"   "3"   "2"   "3"   "76"  "35"  "2"   "2"
##  [37] "1"   "3"   "3"   "7"   "3"   "5"   "1"   "2"   "5"   "2"   "1"   "4"
##  [49] "5"   "1"   "2"   "4"   "NA"  "4"   "3"   "8"   "1"   "1"   "4"   "2"
##  [61] "82"  "2"   "5"   "2"   "3"   "1"   "9"   "2"   "4"   "51"  "2"   "2"
```

```
##  [73] "1"   "1"   "2"   "48"  "3"   "NA"  "14"  "1"   "17"  "13"  "1"   "5"
##  [85] "41"  "3"   "10"  "2"   "3"   "1"   "3"   "40"  "2"   "1"   "3"   "29"
##  [97] "3"   "3"   "2"   "1"   "1"   "80"  "5"   "43"  "1"   "4"   "2"   "4"
## [109] "10"  "47"  "3"   "46"  "1"   "3"   "3"   "1"   "1"   "4"   "18"  "4"
## [121] "NA"  "38"  "3"   "1"   "32"  "2"   "55"  "41"  "26"  "33"  "41"  "9"
## [133] "19"  "3"   "4"   "2"   "4"   "35"  "2"   "3"   "29"  "2"   "16"  "2"
## [145] "2"   "4"   "1"   "4"   "4"   "4"   "4"   "5"   "91"  "1"   "2"   "1"
## [157] "6"   "3"   "4"   "1"   "2"   "2"   "1"   "21"  "4"   "3"   "4"   "5"
## [169] "2"   "3"   "1"   "4"   "1"   "3"   "3"   "3"   "4"   "9"   "2"   "4"
## [181] "27"  "3"   "37"  "5"   "98"  "2"   "4"   "2"   "3"   "3"   "6"   "3"
## [193] "1"   "3"   "1"   "1"   "3"   "2"   "1"   "3"   "96"  "38"  "2"   "1"
## [205] "4"   "2"   "4"   "4"   "22"  "26"  "42"  "3"   "2"   "NA"  "4"   "5"
## [217] "2"   "1"   "3"   "2"   "36"  "5"   "3"   "12"  "3"   "6"   "2"   "5"
## [229] "33"  "2"   "5"   "5"   "4"   "2"   "2"   "3"   "2"   "2"   "4"   "2"
## [241] "23"  "4"   "3"   "2"   "3"   "2"   "8"   "9"   "1"   "8"   "33"  "1"
## [253] "2"   "2"   "1"   "111" "2"   "4"   "2"
```

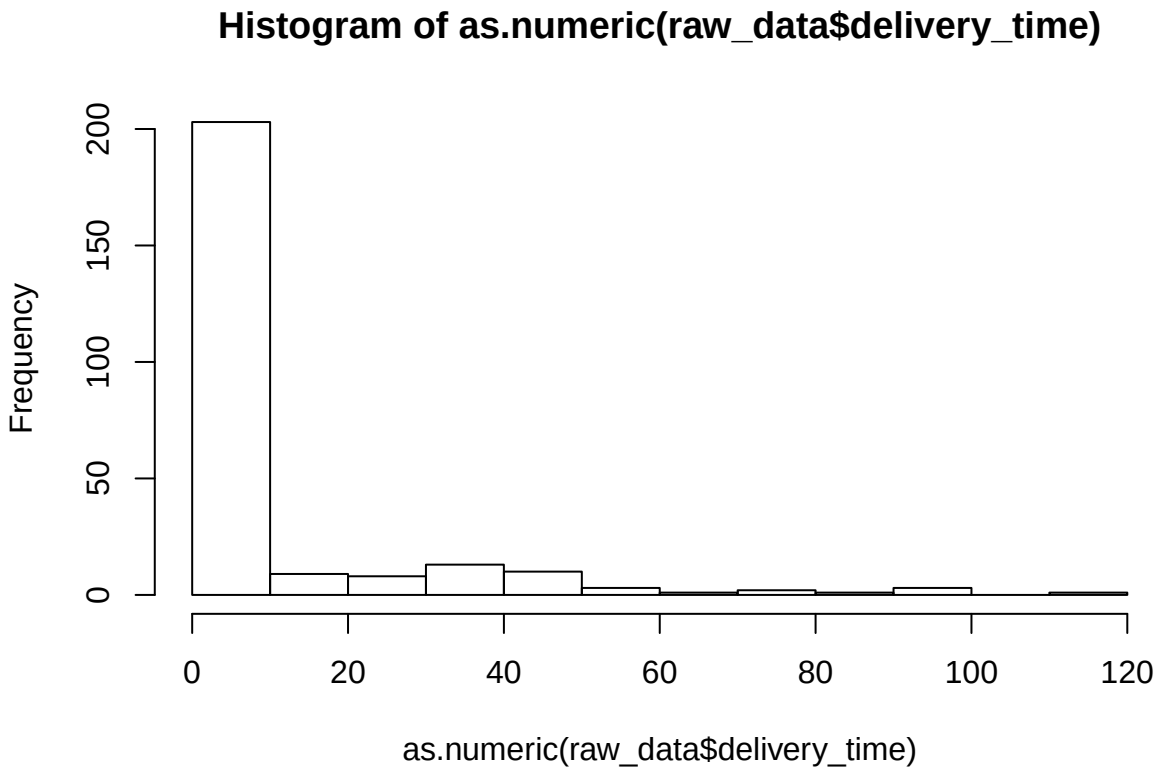
```r
# tabulates the values and the frequencies of delivery time
# there are 5 negative values (now NA cells)
count(raw_data, delivery_time, sort = TRUE)
```

```
## # A tibble: 50 x 2
##    delivery_time     n
##    <chr>         <int>
##  1 2                53
##  2 1                44
##  3 3                41
##  4 4                34
##  5 5                16
##  6 NA                5
##  7 33                4
##  8 6                 4
##  9 8                 4
## 10 9                 4
## # ... with 40 more rows
```

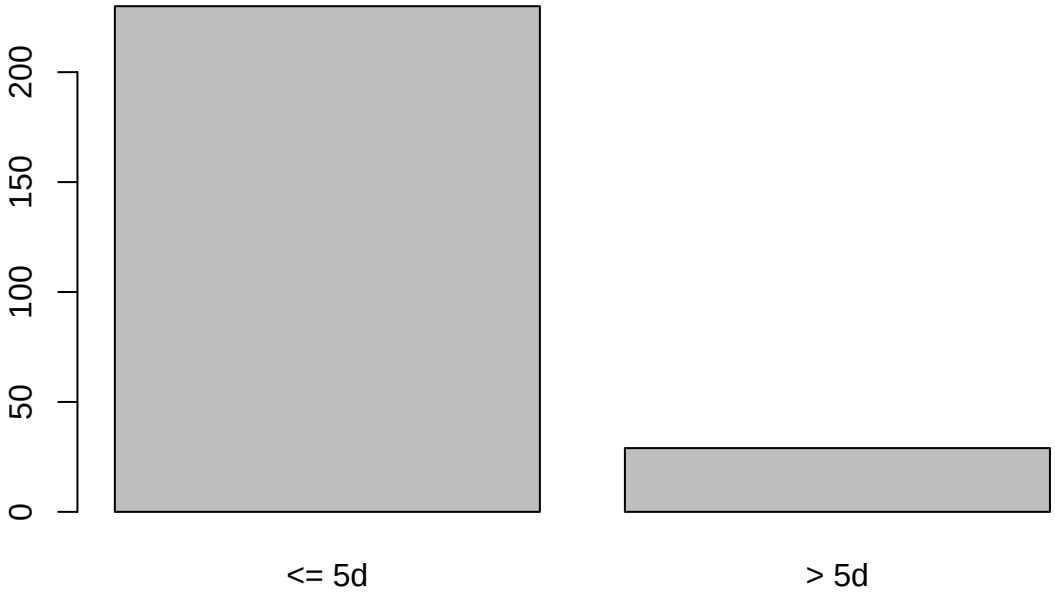
g) Plot a histogram for the new "delivery time" column. Then discretize ("bin") it to levels "NA", "<= 5d", and "> 5d" in a new attribute "delivery_time_discrete" and plot a bar chart for it. Hint: hist(), barplot()

```r
# plotting a histogram of the delivery time attribute
hist(as.numeric(raw_data$delivery_time))
```

```
## Warning in hist(as.numeric(raw_data$delivery_time)): NAs introduced by coercion
```




```r
# binning the data as described in the question using the __case_when__ method.
raw_data <- mutate(raw_data, dt_binned = case_when(
    is.na(delivery_time) ~ "NA",
    delivery_time <= 5 ~ "<= 5d",
    TRUE ~ "> 5d")
)
barplot(table(raw_data$dt_binned))
```



h) Compute the correlation matrix for the numerical attributes only. Plot the matrix of the scatterplots. Plot the heat map of the correlation matrix. Hint: cor(), pairs(), ggcorr() from package "GGally"

```
# To calculate the correlation of all numeric attributes, the numeric attributes are first selected and

# ggcorr() creates a correlation matrix which is plotted as a heat map:
ggcorr(select(raw_data, where(is.numeric)));
```


```
# regular correlation matrix of all (four) numeric attributes
cor(select(raw_data, where(is.numeric)))
```

```
##                ID      price        tax
## ID    1.00000000 0.04074815 0.04074797
## price 0.04074815 1.00000000 1.00000000
## tax   0.04074797 1.00000000 1.00000000
```

i) Standardize all numerical values and again compute their correlation matrix. Hint: scale()

Standardization subtracts from each value the mean of the distribution and divides by the standard deviation. Thus the resulting values have zero mean and unit standard deviation. This can be achieved in R by using the scale method.

```
#raw_data <- scale(select(raw_data, where(is.numeric)))

#raw_data <-  mutate(select(raw_data, where(is.numeric)), (across(.cols = everything(), .fns = scale)))

raw_data <- raw_data %>% select(where(is.numeric)) %>% mutate(across(.cols = everything(), .fns = scale

summarize(raw_data, across(everything(), .fns = c(~mean(., na.rm=T), ~sd(., na.rm=T))))
```

```
## # A tibble: 1 x 6
##        ID_1  ID_2   price_1 price_2    tax_1 tax_2
##       <dbl> <dbl>     <dbl>   <dbl>    <dbl> <dbl>
```

```
## 1 -8.82e-17     1 -8.43e-17       1 4.59e-19     1
```

```
head(raw_data)
```

```
## # A tibble: 6 x 3
##   ID[,1] price[,1] tax[,1]
##    <dbl>     <dbl>   <dbl>
## 1 -1.57    -0.174  -0.174
## 2 -0.987    0.0989  0.0989
## 3 -0.957    0.372   0.372
## 4 -0.112    0.919   0.919
## 5 -1.21    -0.448  -0.448
## 6 -0.308   -0.448  -0.448
```

As can be observed the resulting columns have zero mean and unit standard deviation. The correlation can be calculated as before.

```
corr <- raw_data %>% select(where(is.numeric)) %>% cor(use="pairwise.complete.obs")

ggcorr(corr)
```