[redacted] Report

Course:

Data Engineering for MADS

(EBM213A05.2022-2023.1)

● Redacted for privacy reasons

Introduction

[redacted] is an expert in the area of [redacted], selling large international brands, regional brands, niche brands and exclusive brands. [redacted]'s goal is simple: A shopping experience that makes families happy.
During the first assignment we translated [redacted]'s management dilemma into a research question as follows:

Management dilemma:
[redacted] *has discovered volatility in customer behaviour like sales, visits and conversion in the category beachwear. Of course, this is partially due to seasonal influences. One interesting relationship is still uninvestigated: the influence of the daily temperature. Of course, the behaviour of customers is also affected by other factors like customer background variables (age, gender), their search behaviour, their browsing behaviour, and so on.*

Management question:
How can [redacted] *increase the conversion rate for the beachwear category?*
The aggregation in our analysis has been done on the **session level**.

Research question:
What are the drivers for the conversion rate?

Based on the research question, we formed sub-questions which led us to hypotheses as follows:

1. **Who has a high conversion rate?**
   H1: Mobile website customers have a lower conversion rate than regular websites.
   H2: Women between the ages of 18 to 25 have a higher conversion rate than other customer segments.

   For the second hypothesis, we do not have access to the age of the customers, therefore it could not be tested with distinction for age, only gender.

2. **What effect do product characteristics have on conversion rate?**
   H3: Customers with a bigger price range have a higher conversion rate.

3. **When are conversion rates the highest?**
   H4: Weekend customers have a higher conversion rate (*source*: Bhargava et al., 2011).

4. **What effect do external factors have on conversion rates?**
   H5: The COVID-19 pandemic had a positive impact on the conversion rate.

In this paper, we use several statistical techniques such as the Kolmogorov-Smirnov test, Wilcoxon rank sum test, Kruskal-Wallis test, and simple linear regression, in order to obtain initial answers to our hypothesis.

Data description

In this section, we mainly discuss the data set used in our analysis and provide some initial insights into its structure. The data set was created by combining the data provided by [redacted] with weather data and covid data and consists of 168893 observations with 30 variables.
The weather data has been acquired from the following website:
https://www.knmi.nl/home
and the Covid data from:
https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_time_series

Description of the variables in the data set

For our analysis, we have incorporated a total of # variables from the [redacted] data and # variables from external sources, such as A, B and C. In the following section, we will break down the meaning and method of creating our variables.

Firstly, from the *customer table,* we have created:

- **Age** (*age*), which is the customer's age. We calculated this by subtracting the existing *birth_year* variable from the current year. (note: [redacted] source data does not include age, therefore this variable is '122' for all customers)
- **Relationship length** (*relationship_length*), which is a measure of the duration of a customer's relationship with [redacted]. We calculated this by subtracting the existing *start_year* variable from the current year.
- Two **gender** dummy variables (*male_u* & *female*), the first showcasing if the customer is a male or other, while the second showcasing if the customer is a female. Both utilise the existing *sex* variable, with *male_u* being '1', when *sex* is equal to 'M' or 'U', otherwise '0'. While *female* is '1' when *sex* is 'F', otherwise '0'. Thereby using the CASE WHEN statement.

Next, from the *article events table* we have created **conversion rate per customer**, in the following way:

- First, we have created the **number of sessions** variable (*n_sessions*) by counting the unique *session_id*'s and saving the number as float in the *n_sessions* variable, thereby using the CAST() and COUNT() functions.
- Next, we have created the **number of sales per session** variable (*n_sale_sessions*) by counting the unique *session_id*'s, where the *article_event_type* is '40', which is the number code for sale. We did this using WHERE and specifying *article_event_type* should equal forty.
- We have then used the above-described variables to create **conversion rate per customer** (*conversion_rate*). To do this we have divided the number of sales per session by the number of sessions and saved this per customer.

Next, to create the **favourite channel** variable (*fav_channel*) per customer from the *order table*, we have executed the following steps:

- First, we have created the **number of uses** variable (*n_uses*), which describes how many times a channel has been used by a customer. We calculated this by running a COUNT() command on the existing variable *channel*.
- Next, we have created the **maximum number of uses** variable (*max_uses*), which showcases the number of uses that is highest for that customer. To calculate it we ran a MAX() command on *n_uses*.
- Then, we create a new variable *fav_channel* by filtering *n_uses* by *max_uses*. The new variable signifies the channel that has been used most by that customer, their favourite channel. Thereby, making use of INNER JOIN and especially indicating *n_uses* should equal *max_uses*.
- Finally, if the *fav_channel* is null, we assign "no_orders". Thereby using a CASE WHEN statement.

Next, to create the **favourite article group** variable (*fav_art_group*) from the *order table*, we execute the following steps:

- First, we created the total **number of purchases** per subcategory (*n_purchases*). We calculated this using the existing variable *items,* running a SUM() command on *items*. Also while LEFT JOIN-ing the *article table*.
- Next, we create *max_purchases* by running MAX on *n_purchases* in a subquery. However, to avoid ties we identify the subcategory with the lowest *n_purchases*, using MIN(). Finally, with the subquery, we also make sure to keep only the *n_purchases* equal to *max_purchases*.
- Finally, to create *fav_art_group* we use a CASE WHEN, where we make sure NULLs are recorded as 'no_purchases', otherwise it's equal to the *sub_category* name associated with max_purchases.

Next, we created the lowest and highest **price** variables from the *orders table*, we executed the following steps, by using MIN() and MAX() on the existing *sales_amount* variable to create *min_price* and *max_price* respectively.

So far we have been creating variables at customer-level aggregation, however, for our final table, we work on the session level. This is why we need to add all of the above variables to a new table with the key variable *session_id*. Furthermore, we rename all of the above variables by adding "*cust_*" to the name, as to distinguish them from the variables we will create next on the session level, such as conversion rate, views, and sale dummy per session, which we describe below.

Firstly, we create the **conversion rate per session** variable, through the following steps:

- Initially, we get the base information needed with a series of CASE WHEN statements combined with SUM functions at *article_event_type* from the *article events table* to create dummy variables for each article event type - *type_10*, *type_20*, *type_30*, *type_40*, *type_50*.

- Next, we do the conversion rate calculation with the help of a multilevel CASE WHEN statement, accounting for the possibility of *type_10* and *type_40* being 0 together, one being 0 or neither being 0, thereby adjusting the way *conversion_rate_session* is calculated. All this is then multiplied by 100 to create the conversion rate in a session.
- Finally, we noticed that the above calculations sometimes return a value for *conversion_rate_session* higher than 100. We have addressed this with a CASE WHEN statement assigning 100 to any *conversion_rate_session* value higher than 100 while keeping the original value if it is lower than or equal to 100.

Next, we create a **dummy variable that indicates if the session had a sale** or not (*session_sale_dummy*). We do this by checking if *article_event_type* being '40' (sale) is true, in which case we assign *session_sale_dummy* a value of '1', otherwise '0', thereby using a CASE WHEN statement.

Next, we create *n_session_views*, which indicates the **number of product views in a session**. We do this by summing up all of the times *article_event_type* is '10' (view). This is achieved with SUM and a CASE WHEN statement assigning '1' when *article_event_type* is '10'. We employ the same set of functions to create the **number of sales in a session** variable (*n_session_sales*), except the CASE WHEN statement now assigns '1' when *article_event_type* is '40' (sale).

Finally, we create two **dummy variables indicating customer loyalty**. The first variable (*loyal_sale_dummy*) measures loyalty through sales, namely a customer is considered loyal if they have made a purchase in May, June and July. The second variable (*loyal_session_dummy*) measures loyalty through sessions, whereby we label a customer as loyal if they have had a session in May, June and July. This is achieved through the following steps:

- Initially, we need to create variables indicating whether a customer had a session and sale during each of the six months we have data (January through July) from *article events table*. For each month with the help of a SUM() and CASE WHEN statements targeting *article_event_type* '40', we calculate the total sales a given customer has in that month. This creates *n_jan_sale*, *n_feb_sale* and so on, until July. With a COUNT function, we create a variable indicating the number of sessions the customer had in that month, creating *n_jan_custsessions, n_feb_custsessions* and so on, until July. We run these two queries for each month separately and target that specific month with WHERE *article_event_date* is BETWEEN the dates of that month.
- Finally, we use the above-created variables for sales in a CASE WHEN statement assigning a '1' to *loyal_sale_dummy,* if *n_may_sale*, *n_jun_sale* and *n_jul_sale* are more than zero. We employ the same method to create *loyal_session_dummy* but instead assign it a '1' if *n_may_custsessions*, *n_jun_custsessions* and *n_jul_custsessions* are more than zero.

The above-described variable creation efforts resulted in the following variables for the final [redacted] table:

Motivation of the manner in which we combined the data sources

All of the above-described variables were combined in one session-level aggregated table and imported into R. There we combined it with weather and COVID data, which we have accomplished through the following steps:

- First, we loaded the COVID data, which we acquired from a Github repository. Since this data includes other countries, we make sure to only get data applicable to our [redacted] dataset - The Netherlands. Next, we transposed the data and excluded variables not needed for our analysis, such as *province*, *country*, *latitude* and *longitude*, keeping *covidcases* (number of covid cases on a certain date).
    - We created a *date* variable by extracting the date from the row names with RegEx and used as.Date() to convert the string into a date object.
    - We created the *NewCovidCases* variable with the diff() function, showcasing the difference in the number of cases between a given day and the day before.
      Note on Outlier: Here we found an extreme outlier. We dealt with it by assigning any value above '150000', the maximum *NewCovidCases* value that is less than '150000', essentially capping this variable to that value.
    - Next, we created the *Week* variable indicating the amount of seven-day periods that have occurred since the beginning of the year, in other words, it shows which week it is. To calculate this we use the command week(), which we run on *date*.
    - Next, we create *WeekAverageCovid,* which showcases the average number of COVID cases in a certain week.
    - Next, we create the *wday* variable, which indicates the day of the week (1=monday…7=sunday). We achieve this by using the function wday().
    - Next, using the *wday* variable in an *ifelse()* short statement, we create *weekend_dummy*, which has a value of '1' if *wday* is equal to 6 or 7.
    - Finally, using the *weekend_dummy* variable in an *ifelse()* short statement, we create *week_dummy*, which is assigned a value of '0', when *weekend_dummy* is 1.

- Second, we loaded our weather data, which we acquired from KNMI. We have adjusted the data to our needs by aggregating the data to date. This will later be useful to assign weather parameters to each session based on its date. Besides *date* we keep the following variables:
    - *windspd:* daily mean windspeed (in 0.1 m/s)
    - *temp:* daily mean temperature (in 0.1 degrees Celsius)
    - *sun_duration:* sunshine duration calculated from global radiation (in 0.1 hours)
    - *perc_max_sunduration:* percentage of max. potential sun duration
    - *rain_duration:* precipitation duration (in 0.1 hours)
    - *rain_amount:* daily precipitation amount (in 0.1 mm)
    - *cloud_cover:* mean daily cloud cover (in octants; 9=sky invisible)

Next, we combined this data with our COVID data described earlier using merge() on *date*, creating a new data frame containing all of our external data aggregated on the *date* level, resulting in the following variables for the final external data table:

Finally, we combined our [redacted] data and newly created external data df., merging on *session_date* and *date* to create our final data frame, ready for analysis.

Statistical descriptives of variables for which this is relevant

Once we had our combined data frame ready, we explored some key descriptive statistics:

- Mean of Week/Weekend: 71.2% of sessions during the week / 28.8% during the weekend
- Mean of Session Sale Dummy: 10.6% of sessions have a sale
- Mean of Session Conv. Rate: 5.36% average session conversion rate
- Mean of Relationship Length: 11 years average relationship length
- Min Customer Last Order Date: the oldest order in our dataset happened on 2007-07-25
- Mean Customer Min. Price: the average lowest price customers would pay is €26.65
- Mean Customer Max. Price: the average highest price customer would pay is €48.74
- Mean Number of Views per Session: on average customers view 4-5 product pages
- The standard deviation of Customer Max. Price: ~23.11, relatively high. Suggests max. price is spread out and customers differ on their max. price preference.
- The standard deviation of Customer Min. Price: ~15.40, relatively high. Suggests min. price is spread out and customers differ on their min. price preference.

Description of how we identified and remedied missing observations and outliers

Afterwards, we made sure to check for outliers. We did this by generating boxplots for our variables and did not find any significant outliers that required intervention.

To check for missing observations we ran md.pattern(). Inspecting the patterns with missings, we see that 51.386 observations have two missing values, namely for minimum/maximum price values, this is noteworthy because these customers have purchased, therefore should have a minimum and maximum price. After consultation with our instructor team, we learned this is due to an irregularity in the data. Next, 470 observations have 3 missing values, namely minimum/maximum price and last order date. This is a logical occurrence, as these are the customers who have not placed an order. We will not impute these missings, as these customers are not of interest to our analyses, because they have not converted within our data's timeframe. Next, we have 37.544 observations with 10 missing values. We have found that this is explained by the anomaly of customer [redacted], who exists in Wehkamp's customer table, but not in the session table. As many key values are missing for these observations we will not impute them or include them in the analyses. Finally, we can see that 79.493 observations have no missing values, which is a sufficient sample size for the analyses.

Analyses & Results

To answer our subquestions, we ran tests investigating the effect of multiple key characteristics as driver variables and session conversion rate as our KPI.

We divide the analyses and result reporting per subquestion. After each report we also indicate whether we have confirmed or rejected the corresponding hypotheses:

1. **Who has a high conversion rate?**

We started with **gender**, namely whether females or males have a higher *session conversion rate*. We investigated through the following process:

- We first performed the **one-sample Kolmogorov-Smirnov test**, on our continuous KPI variable, *session conversion rate*. Our test resulted in a test statistic for the variable *session conversion rate* equal to 0.5, $p<.001$. Since the p-value is less than .05, we can reject the null hypothesis. We have sufficient evidence to say that the data does not come from a normal distribution.

- In the next step, as our KPI is numerical and our driver is categorical ($k=2$), we used the nonparametric test, the **Wilcoxon rank sum test**, for comparing two independent groups of samples. Performing the test returned the following results: $W = 704422978$, $p<.001$. Since the p-value is less than .05, we can reject the null hypothesis, concluding that the difference between females and males in their conversion rate is significant.

Next, we wanted to test if more **loyal customers** have a higher *session conversion rate*. As mentioned earlier we have two indicators for loyalty - one based on sales and the other on sessions, therefore we performed two sets of analyses. As we have already checked the normality of *session conversion rate* and found it to be non-normally distributed, we perform the **Wilcoxon rank sum test** for each loyalty metric:

- Performing the test for *loyalty based on sales* returned the following results: $W=115210767$, $p<.001$. Since the p-value is less than .05, we can reject the null hypothesis, concluding that there is a significant difference in the conversion rates of loyal customers based on sales and non-loyal customers.
- Performing the test for *loyalty based on sessions* returned the following results: $W=1722959233$, $p<.001$. Since the p-value is less than .05, we can reject the null hypothesis, concluding that there is a significant difference in the conversion rates of loyal customers based on sessions and non-loyal customers.

Next, we investigate whether **relationship length** influences *session conversion rate*. As *relationship length* is a continuous variable, we have opted for **simple linear regression**. After performing the analysis we found that:

- The overall regression model was statistically significant (Adjusted $R^2=0.0001585$, $p<.001$)
- *Relationship length* is significant ($p<.001$) with a positive estimate value, concluding that *relationship length* has a significant positive influence on *conversion rate*.

We were also interested if **favourite channel** has an impact on *session conversion rate*. As we have already checked the normality of the *session conversion rate* and found it to be

non-normally distributed, as well as *favourite channel* being a categorical driver (k>2), we perform the **Kruskal-Wallis test**. The test returned the following results:

- KW chi-squared=22408, p<.001, thereby indicating a significant difference in *conversion rate* among *favorite channels*.

Additionally, we investigated specifically which **channel** has a higher *mean session conversion rate*. To do this we used a proprietary function (*seg.summ*), which splits the data by reported groups. In our case, we calculated the *mean conversion rate per channel*. We found the *mean conversion rate* to be highest for website sessions, compared to mobile and offline.

Next, we also investigated if **favourite article group** has an impact on *session conversion rate*. Here we performed the **Kruskal-Wallis test** based on the same reasoning as for *favourite channel*. The test returned the following results:

- KW chi-squared=22814, p<.001, thereby indicating a significant difference in *conversion rate* among *favorite article groups*.

Additionally, we investigated specifically which **article group** has a higher *mean session conversion rate*. To do this we employed the same function as for our investigation on *channels*. Based on our results we composed the following top 5 ranking of *mean conversion rate per article groups*: (1) Ladies Fashion Brands - Tops, (2) Ladies Fashion Brands - Other, (3) Telstar, (4) Boys Fashion Brands, (5) Desigual. Based on insufficient data we excluded "Girls Fashion Brands", which had a disproportionately higher score than all other *article groups*.

After considering these findings we **confirmed H1**: *Mobile website customers have a lower conversion rate than on regular websites*, as our analysis indicated website customers have a higher *mean session conversion rate* than mobile customers.

Due to the limitations of the Wilcoxon rank sum test we **partially confirm our H2**: *Women between the ages 18 to 25 have a higher conversion rate than other customer segments*, as our results show women and men have different *session conversion rates*, however, we cannot infer if one is higher than the other.

2. What effect do product characteristics have on conversion rate?

First, we investigated the influence of **price** on *session conversion rate*. In our data, we have two measures for *price* - *min. price* and *max. price*. As both of our *price* measures are continuous variables, we have opted for **simple linear regression**. After performing the analysis we found that:

- The overall regression model was statistically significant (Adjusted R^2=0.000571, p<.001)
- *Minimum price* is significant (p<.001) with a negative estimate value, indicating a significant negative effect of *min. price* on *conversion rate*. Therefore, the lower the minimum price of the product, the higher the conversion rate.
- *Maximum price* is insignificant (p=0.978).

Next, we wanted to see whether the ***number of sales*** or ***views*** has a more significant effect on *session conversion rate*. To investigate this we employed a **simple linear regression**, as both our KPI and two drivers are continuous variables. After performing the analysis we found that:

- The overall regression model was statistically significant (Adjusted R^2=0.3669, p<.001)
- *Number of views* is significant (p<.001) with a negative estimate value, indicating a significant negative effect of *number of views* on *conversion rate*. Therefore, a customer viewing more pages on the site has a negative effect on *conversion rate*. We interpret this as 'window shopping' and perhaps that people who are viewing a lot of product pages are there to look around and not to deliberately buy. Furthermore, this might also be a case of "hedonic browsing" (Zheng et al., 2019)[1].
- *Number of sales* is significant (p<.001) with a positive estimate value, indicating a significant positive direction of the effect of *numbers of sales* on *conversion rate*. This is a logical phenomenon, as *number of sales* is part of the calculation of *conversion rate* in the numerator.

We have been able to **confirm H3**: *Customers with a bigger price range have a higher conversion rate*, as our regression results have shown that as the minimum price goes down the conversion rate increases and vice versa. It is noteworthy that we were unable to confirm if the same holds when the maximum price changes, therefore we can only confirm our hypothesis for the lower part of the price range.

3. When are conversion rates the highest?

Here, we investigated whether *session conversion rate* is influenced by the **time of the week** - distinguishing between week-weekend through the variable *weekend dummy*. As we have already checked the normality of *session conversion rate* and found it to be non-normally distributed, we performed the **Wilcoxon rank sum test** and got the following results:

- Performing the test returned the following results: W = 2981356267, p<.001, therefore we can reject the null hypothesis, concluding that there is a significant difference in the *session conversion rates* during the *week* and *weekend*.

After considering these findings, due to limitations of the test, we **partially confirm H4**: *Weekend customers have a higher conversion rate*, as our results show a significant difference between week and weekend conversion rates, however, we cannot infer if it is higher.

4. What effect do external factors have on conversion rates?

First, we investigated whether *session conversion rate* is influenced by the average **amount of new COVID cases** in the week of the session. As both *session conversion rate* and *new COVID cases* are continuous variables, we have opted for **simple linear regression**. After performing the analysis we found that:

- The overall regression model was statistically significant (Adjusted R^2=0.0.0002187, p<.001)

[1] Zheng, X., Men, J., Yang, F., & Gong, X. (2019). Understanding impulse buying in mobile commerce: An investigation into hedonic and utilitarian browsing. International Journal of Information Management, 48, 151-160.

- *Average new weekly COVID cases* is significant ($p<.001$) with a negative estimate value, indicating a significant negative direction of the effect of *new COVID cases* on *session conversion rate*. Therefore, the more *COVID cases* the lower *session conversion rate* is.

Finally, we investigated how **weather** variables affect *session conversion rate*. We focus on *temperature*, *wind speed*, *sun duration* and *rain duration*. As all of these are continuous variables, we opted for **simple linear regression**. After performing the analysis we found that:

- The overall regression model was statistically significant (Adjusted $R^2=0.00008954$, $p<.001$)
- *Temperature* is marginally significant ($p=0.0783$) with a positive estimate value, indicating a possible positive direction of the effect of *temperature* on *session conversion rate*. Therefore, it is possible that when the *temperature* goes up, *session conversion rate* also goes up.
- *Wind speed*, *sun duration* and *rain duration* were all insignificant (respectively: $p=0.2159$; $p=0.1507$; $p=0.7066$).

After considering these findings we **rejected H5**: *The COVID-19 pandemic had a positive impact on the conversion rate*, as our regression results have shown a significant negative impact of new covid cases on the session conversion rate.

Additionally, although not one of our hypotheses, we found that weather factors do not have a statistically significant effect on session conversion rate, besides temperature, which is marginally significant ($p=0.0783$).

Managerial Insights

The purpose of this report is to find out what are the drivers of the conversion rate on the session level and obtain initial answers to our research question.
Based on our analysis, we were able to find that the session conversion rate is driven by multiple indicators such as the gender of the customer, his/her loyalty, and favorite channel. One important factor to mention is that the longer the relationship length of a customer, the higher will be the session conversion rate. We also saw that the big majority of people prefer using the website channel when shopping from [redacted].
Other indicators worth mentioning are the number of views and sales. Notice that the more time customers spend on viewing the articles, the less likely they are to actually buy them. In contrast, when the number of sales goes up, the conversion rate will follow as well.
We also took into consideration how external factors might affect the session conversion rate. Therefore, we conclude that the COVID-19 pandemic had a significant impact on the conversion rate. More specifically, when the number of COVID cases is increasing, the conversion rate is decreasing.
Lastly, we saw that temperature is marginally significant, meaning that when it goes up, there is a possibility that also the conversion rate will go up.

To conclude, understanding the drivers of our KPI - the session conversion rate is vital in order to be able to assess customer behaviour and based on it, make decisions on how to increase the number of sales and provide the best shopping experience.

APPENDIX A

SQL Script

SQL Script

```sql
/* see what customer table looks like */
SELECT *
FROM customer
ORDER BY customer_id
LIMIT 10;


/* last order date is wrong in the customer table and
/or articeevents/order table! PUT THIS IN THE REPORT
 */
SELECT last_order_date
FROM customer
WHERE customer_id = '[redacted]
';
SELECT article_event_date, article_event_type
FROM articleevents
WHERE customer_id = '[redacted]
' AND article_event_date > '2022-07-01';
SELECT *
FROM public.order
WHERE customer_id = '[redacted]
';
/* THIS CUSTOMER DOES NOT EXIST IN THE CUSTOMER TABLE
, BUT DOES IN THE SESSION TABLE, ALSO PUT THIS IN
REPORT */
SELECT *
FROM customer
WHERE customer_id = '[redacted]
';
SELECT *
FROM public.session
WHERE customer_id = '[redacted]
'
LIMIT 10;


/* the first table we can work from, no age class
since all ages are 122 */
CREATE TEMP TABLE tablebase AS
SELECT customer_id,
        2022 - start_year AS relationship_length,
```

SQL Script

```
32          2022 - birth_year AS age,
33          last_order_date,
34          CASE WHEN sex = 'M' THEN 1 WHEN sex = 'U'
   THEN 1 ELSE 0 END AS male_u,
35          CASE WHEN sex = 'F' THEN 1 ELSE 0 END AS
   female
36 FROM customer;
37
38 SELECT *
39 FROM tablebase
40 LIMIT 10;
41
42 /* check if there are the same number of session in
   both tables, can safely use article event table*/
43 SELECT COUNT(DISTINCT session_id)
44 FROM public.session;
45 SELECT COUNT(DISTINCT session_id)
46 FROM articleevents;
47
48 /* begin making the conversion column*/
49 CREATE TEMP TABLE n_sessions_table AS
50 SELECT customer_id, CAST(COUNT(DISTINCT session_id)
   AS float) AS n_sessions
51 FROM articleevents
52 GROUP BY customer_id;
53
54 CREATE TEMP TABLE n_sale_sessions_table AS
55 SELECT customer_id, CAST(COUNT(DISTINCT session_id)
   AS float) AS n_sale_sessions
56 FROM articleevents
57 WHERE article_event_type = 40
58 GROUP BY customer_id;
59
60 /*not all customers have bought something in sessions
   */
61 SELECT COUNT(DISTINCT customer_id)
62 FROM n_sessions_table;
63 SELECT COUNT(DISTINCT customer_id)
64 FROM n_sale_sessions_table;
65
66 /* make conversion and join it with main table*/
```

SQL Script

```sql
CREATE TEMP TABLE conversion_table AS
SELECT nst.customer_id, nsst.n_sale_sessions, nst.
n_sessions
FROM n_sessions_table nst
LEFT JOIN n_sale_sessions_table nsst
ON nst.customer_id = nsst.customer_id;

CREATE TEMP TABLE table1 AS
SELECT tb.*, ct.n_sale_sessions/ct.n_sessions AS
conversion_rate
FROM tablebase tb
LEFT JOIN conversion_table ct
ON tb.customer_id = ct.customer_id;

/* The main table with conversion rate and no NULL
 */
CREATE TEMP TABLE table2 AS
SELECT customer_id, age, relationship_length, male_u
, female,
        CASE WHEN conversion_rate IS NULL THEN 0
ELSE conversion_rate END AS conversion_rate,
        last_order_date
FROM table1;

SELECT *
FROM table2
LIMIT 20;

/* count number of people who have bought something
from [redacted] since januari*/
SELECT COUNT(*)
FROM table2
WHERE conversion_rate <> 0;

/* begin making the fav channel column */
CREATE TEMP TABLE channel_count_table AS
SELECT customer_id, channel, COUNT(channel) AS
n_uses
FROM public.order
GROUP BY customer_id, channel;
/* select for the most choses channel by inner
```

SQL Script

```
100 joining on itself*/
101 CREATE TEMP TABLE fav_channel_table AS
102 SELECT t1.customer_id, MAX(t1.channel) AS
    fav_channel, MAX(t1.n_uses) AS n_uses
103 FROM channel_count_table t1
104 INNER JOIN (SELECT customer_id, MAX(n_uses) AS
    max_uses
105             FROM channel_count_table
106             GROUP BY customer_id) t2
107 ON t1.customer_id = t2.customer_id AND t1.n_uses =
    t2.max_uses
108 GROUP BY t1.customer_id;
109
110 /* merge with the main table*/
111 CREATE TEMP TABLE table3 AS
112 SELECT t1.*,
113        CASE WHEN t2.fav_channel IS NULL THEN '
    no_orders' ELSE t2.fav_channel END AS fav_channel
114 FROM table2 t1
115 LEFT JOIN fav_channel_table t2
116 ON t1.customer_id = t2.customer_id;
117
118 /*check the table, seems to all be fine, having a
    fav channel implies a positive conversion rate*/
119 SELECT *
120 FROM table3
121 WHERE fav_channel <> 'no_orders'
122 LIMIT 10;
123
124 /* easy to make a fav article group since we did
    that in lecture 3*/
125 /*get number of purchases from the subcategories*/
126 CREATE TEMP TABLE art_subcat1 AS
127 SELECT ord.customer_id , sub_category, SUM(ord.items
    ) AS n_purchases
128 FROM public.order ord
129 LEFT JOIN article art ON ord.article_id = art.
    article_id
130 GROUP BY ord.customer_id, art.sub_category;
131
132 SELECT *
```

SQL Script

```
FROM art_subcat1
ORDER BY customer_id
LIMIT 10;

/*get the max n_purchases, using a subquery makes the most sense, keep the rows with the highest purchases, but we get ties! use min*/
CREATE TEMP TABLE maxorder_cat AS
SELECT t1.customer_id, MIN(t1.sub_category) AS sub_category, MIN(t1.n_purchases) AS n_purchases
FROM art_subcat1 t1
INNER JOIN (SELECT customer_id, MAX(n_purchases) AS max_purchases
            FROM art_subcat1
            GROUP BY customer_id) t2
ON t1.customer_id = t2.customer_id AND t1.n_purchases = t2.max_purchases
GROUP BY t1.customer_id;

SELECT *
FROM maxorder_cat
ORDER BY customer_id
LIMIT 10;

/* merge with the main table to, can make them into dummy variables in R */
CREATE TEMP TABLE table4 AS
SELECT t1.*, CASE WHEN t2.sub_category IS NULL THEN 'no_purchases' ELSE t2.sub_category END AS fav_art_group
FROM table3 t1
LEFT JOIN maxorder_cat t2
ON t1.customer_id = t2.customer_id;
/*looks fine*/
SELECT *
FROM table4
WHERE fav_art_group <> 'no_purchases'
LIMIT 10;

/* start work on the acceptable price range column, can get everything from orders, since articles are
```

SQL Script

```
ordered 1 by 1*/
/*sales amount is current price of the article that
    is ordered, because of the way order table is
    structered*/
CREATE TEMP TABLE price_range_table AS
SELECT customer_id, MIN(sales_amount) AS min_price,
    MAX(sales_amount) AS max_price
FROM public.order
GROUP BY customer_id;

SELECT *
FROM price_range_table
LIMIT 10;

/* merge it with the main table, can make ranges in
    R, probably easier, Letting them be NULLs for 0
    purchases makes sense*/
CREATE TEMP TABLE table5 AS
SELECT t1.*, t2.min_price AS min_price_art, t2.
    max_price AS max_price_art
FROM table4 t1
LEFT JOIN  price_range_table t2
ON t1.customer_id = t2.customer_id;
/*looks fine*/
SELECT *
FROM table5
WHERE min_price_art IS NOT NULL
LIMIT 10;

/* change the main table to a session focused table
    , 168893 distinct sessions */
CREATE TEMP TABLE session_table1 AS
SELECT t1.session_id, t1.customer_id, t1.
    session_date, age AS cust_age , relationship_length
    AS cust_relationship_length, male_u, female,
        conversion_rate AS cust_conversion_rate,
    last_order_date AS cust_last_order_date, fav_channel
     AS cust_fav_channel,
        fav_art_group AS cust_fav_art_group,
    min_price_art AS cust_min_price, max_price_art AS
    cust_max_price
```

SQL Script

```
192 FROM public.session t1
193 LEFT JOIN table5 t2
194 ON t1.customer_id = t2.customer_id;
195
196 SELECT *
197 FROM session_table1
198 LIMIT 10;
199
200 /* making a conversion rate per session, using
    type10's and type 40's to calculate the conversion
    rate, here we get the base info we need*/
201 CREATE TEMP TABLE session_sumtypes_table AS
202 SELECT session_id,
203         CAST(SUM(CASE WHEN article_event_type = '10'
     THEN 1 ELSE 0 END) AS float) AS type_10,
204         SUM(CASE WHEN article_event_type = '20' THEN
     1 ELSE 0 END) AS type_20,
205         SUM(CASE WHEN article_event_type = '30' THEN
     1 ELSE 0 END) AS type_30,
206         CAST(SUM(CASE WHEN article_event_type = '40'
     THEN 1 ELSE 0 END) AS float) AS type_40,
207         SUM(CASE WHEN article_event_type = '50' THEN
     1 ELSE 0 END) AS type_50
208 FROM articleevents
209 GROUP BY session_id;
210
211 /* calculating the actual session conversion rate,
    some are gonna be over 100 percent, fix that in the
    next table */
212 CREATE TEMP TABLE session_conversion_rate_table1 AS
213 SELECT session_id,
214         100.0 * (CASE WHEN type_10 > 0 AND type_40
     > 0 THEN type_40/type_10
215         WHEN type_10 = 0 AND type_40 > 0 THEN 1.0
216         WHEN type_10 > 0 AND type_40 = 0 THEN 0.0
217         WHEN type_10 = 0 AND type_40 = 0 THEN 0.0
    END) AS conversion_rate_session
218 FROM session_sumtypes_table;
219
220 /* sanity check: in total 658 have conversion rate
    above 100%, might affect our result, need to fix it
```

SQL Script

```
220  */
221
222 SELECT MIN(conversion_rate_session), MAX(
    conversion_rate_session)
223 FROM session_conversion_rate_table1;
224
225 SELECT *
226 FROM session_conversion_rate_table1
227 WHERE conversion_rate_session = 1800.0;
228
229 SELECT *
230 FROM session_sumtypes_table
231 WHERE session_id = 'C59521C4E70438327E154DB72F7258D9
    ' OR session_id = '3D5DA9F9A656255C9CB9982D78B9DB95'
     OR session_id = '04B1C313D259FCD316A943246E177519';
232
233 SELECT COUNT(*)
234 FROM session_conversion_rate_table1
235 WHERE conversion_rate_session > 100.0;
236
237 /* sanity check ends */
238
239 CREATE TEMP TABLE session_conversion_rate_table2 AS
240 SELECT session_id,
241         CAST((CASE WHEN conversion_rate_session >
    100 THEN 100.00 ELSE conversion_rate_session END) AS
     DECIMAL(5,2)) AS conversion_rate_session
242 FROM session_conversion_rate_table1;
243
244 SELECT *
245 FROM session_conversion_rate_table2
246 ORDER BY conversion_rate_session DESC
247 LIMIT 100;
248
249 /*merging it with the main table to be used */
250 CREATE TEMP TABLE session_table2 AS
251 SELECT t1.*, t2.conversion_rate_session AS
    session_conversion_rate
252 FROM session_table1 t1
253 LEFT JOIN session_conversion_rate_table2 t2
254 ON t1.session_id = t2.session_id;
```

SQL Script

```
255
256 SELECT *
257 FROM session_table2
258 LIMIT 10;
259
260 /* make a dummy variables that tells us if the
    session has had a sale or not*/
261 CREATE TEMP TABLE session_sale_dummy_table AS
262 SELECT session_id,
263         CASE WHEN SUM(CASE WHEN article_event_type
     = '40' THEN 1 ELSE 0 END) > 0 THEN 1 ELSE 0 END AS
    session_sale_dummy,
264         SUM(CASE WHEN article_event_type = '10' THEN
     1 ELSE 0 END) AS n_session_views,
265         SUM(CASE WHEN article_event_type = '40' THEN
     1 ELSE 0 END) AS n_session_sales
266 FROM articleevents
267 GROUP BY session_id;
268
269 /* add it to the main table to be used*/
270 CREATE TEMP TABLE session_table3 AS
271 SELECT t1.*, t2.session_sale_dummy, t2.
    n_session_views, t2.n_session_sales
272 FROM session_table2 t1
273 LEFT JOIN session_sale_dummy_table t2
274 ON t1.session_id = t2.session_id;
275
276 SELECT *
277 FROM session_table3
278 LIMIT 10;
279
280 /* check if we still have all the sessions, we do
     !*/
281 SELECT COUNT(DISTINCT session_id)
282 FROM session_table3;
283
284 /* NO CUSTOMER LOYALTY, WEEK-WEEKEND SESSION, RAINY/
    CLOUDY/SUNNY SESSION OR COIVD SESSION DATA IN TABLE
    YET, ALSO FIX CUST CONVERSION RATE*/
285 /* customer loyalty variable make */
286 CREATE TEMP TABLE jan_table AS
```

SQL Script

```
287 SELECT customer_id, COUNT(DISTINCT session_id) AS
    n_jan_custsessions, SUM(CASE WHEN article_event_type
     = '40' THEN 1 ELSE 0 END) AS n_jan_sale
288 FROM public.articleevents
289 WHERE article_event_date BETWEEN '2022-01-01' AND '
    2022-01-31'
290 GROUP BY customer_id;
291
292 CREATE TEMP TABLE feb_table AS
293 SELECT customer_id, COUNT(DISTINCT session_id) AS
    n_feb_custsessions, SUM(CASE WHEN article_event_type
     = '40' THEN 1 ELSE 0 END) AS n_feb_sale
294 FROM public.articleevents
295 WHERE article_event_date BETWEEN '2022-02-01' AND '
    2022-02-28'
296 GROUP BY customer_id;
297
298 CREATE TEMP TABLE mar_table AS
299 SELECT customer_id, COUNT(DISTINCT session_id) AS
    n_mar_custsessions, SUM(CASE WHEN article_event_type
     = '40' THEN 1 ELSE 0 END) AS n_mar_sale
300 FROM public.articleevents
301 WHERE article_event_date BETWEEN '2022-03-01' AND '
    2022-03-31'
302 GROUP BY customer_id;
303
304 CREATE TEMP TABLE apr_table AS
305 SELECT customer_id, COUNT(DISTINCT session_id) AS
    n_apr_custsessions, SUM(CASE WHEN article_event_type
     = '40' THEN 1 ELSE 0 END) AS n_apr_sale
306 FROM public.articleevents
307 WHERE article_event_date BETWEEN '2022-04-01' AND '
    2022-04-30'
308 GROUP BY customer_id;
309
310 CREATE TEMP TABLE may_table AS
311 SELECT customer_id, COUNT(DISTINCT session_id) AS
    n_may_custsessions, SUM(CASE WHEN article_event_type
     = '40' THEN 1 ELSE 0 END) AS n_may_sale
312 FROM public.articleevents
313 WHERE article_event_date BETWEEN '2022-05-01' AND '
```

SQL Script

```
313 2022-05-31'
314 GROUP BY customer_id;
315
316 CREATE TEMP TABLE jun_table AS
317 SELECT customer_id, COUNT(DISTINCT session_id) AS
    n_jun_custsessions, SUM(CASE WHEN article_event_type
     = '40' THEN 1 ELSE 0 END) AS n_jun_sale
318 FROM public.articleevents
319 WHERE article_event_date BETWEEN '2022-06-01' AND '
    2022-06-30'
320 GROUP BY customer_id;
321
322 CREATE TEMP TABLE jul_table AS
323 SELECT customer_id, COUNT(DISTINCT session_id) AS
    n_jul_custsessions, SUM(CASE WHEN article_event_type
     = '40' THEN 1 ELSE 0 END) AS n_jul_sale
324 FROM public.articleevents
325 WHERE article_event_date BETWEEN '2022-07-01' AND '
    2022-07-31'
326 GROUP BY customer_id;
327
328 CREATE TEMP TABLE monthly_table AS
329 SELECT  t0.customer_id,
330         t1.n_jan_custsessions, t1.n_jan_sale,
331         t2.n_feb_custsessions, t2.n_feb_sale,
332         t3.n_mar_custsessions, t3.n_mar_sale,
333         t4.n_apr_custsessions, t4.n_apr_sale,
334         t5.n_may_custsessions, t5.n_may_sale,
335         t6.n_jun_custsessions, t6.n_jun_sale,
336         t7.n_jul_custsessions, t7.n_jul_sale
337 FROM customer t0
338 LEFT JOIN jan_table t1
339 ON t0.customer_id = t1.customer_id
340 LEFT JOIN feb_table t2
341 ON t0.customer_id = t2.customer_id
342 LEFT JOIN mar_table t3
343 ON t0.customer_id = t3.customer_id
344 LEFT JOIN apr_table t4
345 ON t0.customer_id = t4.customer_id
346 LEFT JOIN may_table t5
347 ON t0.customer_id = t5.customer_id
```

SQL Script

```sql
LEFT JOIN jun_table t6
ON t0.customer_id = t6.customer_id
LEFT JOIN jul_table t7
ON t0.customer_id = t7.customer_id;

SELECT *
FROM monthly_table
LIMIT 10;

CREATE TEMP TABLE loyal_table AS
SELECT customer_id,
        CASE WHEN n_may_sale > 0 AND n_jun_sale > 0
    AND n_jul_sale > 0 THEN 1 ELSE 0 END AS
    loyal_sale_dummy,
        CASE WHEN n_may_custsessions > 0 AND
    n_jun_custsessions > 0 AND n_jul_custsessions > 0
    THEN 1 ELSE 0 END AS loyal_session_dummy
FROM monthly_table
ORDER BY loyal_session_dummy DESC;


CREATE TEMP TABLE session_table4 AS
SELECT t1.*, t2.loyal_sale_dummy AS loyal_sale_cust
    , t2.loyal_session_dummy AS loyal_session_cust
FROM session_table3 t1
LEFT JOIN loyal_table t2
On t1.customer_id = t2.customer_id;

SELECT *
FROM session_table4
LIMIT 10;

SELECT *
FROM session_table4;

SELECT COUNT(DISTINCT customer_id)
FROM session_table4;

/* SANITY CHECKS */

/* same session number: 168893 */
```

SQL Script

```sql
SELECT COUNT(DISTINCT session_id)
FROM session_table4;

/* GOOD! we get a result of 5.36, adjusted compared to 5.75 (result of Disaster check - conversion-% (sales/views)) */
SELECT AVG(session_conversion_rate) AS avg_cr_session
FROM session_table4;

SELECT MIN(session_date), MAX(session_date)
FROM session_table4;

SELECT MIN(cust_relationship_length), MAX(cust_relationship_length), AVG(cust_relationship_length)
FROM session_table4;

/* total number of loyal session: 2518 */
SELECT COUNT(DISTINCT customer_id)
FROM session_table4
WHERE loyal_session_cust = 1;

/* total number of loyal sale: 87 */
SELECT COUNT(DISTINCT customer_id)
FROM session_table4
WHERE loyal_sale_cust = 1;

/* One missing value in both loyal_sale and loyal_session:
"E79..." THIS CUSTOMER DOES NOT EXIST */
SELECT *
FROM session_table4
WHERE loyal_sale_cust IS NULL OR loyal_session_cust IS NULL;

/* SANITY CHECKS ENDS */
```

APPENDIX B

R Script

R Script

```
# PREPARATION ----

# clean workspace
rm(list = ls())

# load libraries
library(librarian)
librarian::shelf(stringr,lubridate,dplyr,mice,car,
ggplot2,  cran_repo = 'https://cran.r-project.org')

# set working directory
setwd("")


# DATA WORK ----

# read in knmi weather data and [redacted] data
weatherdataknmi_detailed <- read.csv("
weatherdataknmi_detailed_csv.txt")

[redacted]_data <- data.frame(read.csv("[redacted]data1.csv
"))
# Download COVID data from the web and adjust it to
our needs ----
Covid_cases <- read.csv("https://github.com/
CSSEGISandData/COVID-19/raw/master/csse_covid_19_data
/csse_covid_19_time_series/
time_series_covid19_confirmed_global.csv")

Covid_cases_reduced <- Covid_cases[
Covid_cases$Country.Region == "Netherlands" &
Covid_cases$Province.State == "",]

# Transpose the dataframe and exclude the first four
entries (province, country, lat, lon)
Neth_covidcases <- data.frame(t(Covid_cases_reduced
[,-c(1:4)]))

# Extract a date variable from the row.names and set
names
Neth_covidcases$date <- as.Date(row.names(
```

R Script

```
29 Neth_covidcases),"X%m.%d.%y")
30 colnames(Neth_covidcases) <- c("covidcases", "date")
31 rownames(Neth_covidcases) <- c()
32
33 # make difference column
34 Neth_covidcases$NewCovidCases <- c(NA, diff(
   Neth_covidcases$covidcases,1))
35 plot(Neth_covidcases$date,
   Neth_covidcases$NewCovidCases)
36 boxplot(Neth_covidcases$NewCovidCases)
37
38 # adjust the 1 extreme outlier
39 Neth_covidcases$NewCovidCases[
   Neth_covidcases$NewCovidCases > 150000] <- max(
   Neth_covidcases$NewCovidCases[
   Neth_covidcases$NewCovidCases < 150000],na.rm = TRUE)
40 plot(Neth_covidcases$date,
   Neth_covidcases$NewCovidCases)
41 boxplot(Neth_covidcases$NewCovidCases)
42
43 # add a week + weekly average column for 2022
44 Neth_covidcases <- Neth_covidcases[
   Neth_covidcases$date > "2021-12-31", ]
45 Neth_covidcases$Week <- week(Neth_covidcases$date)
46
47 Weeks <- unique(Neth_covidcases$Week)
48 for (iWeek in Weeks) {
49   Neth_covidcases$WeekAverageCovid[
   Neth_covidcases$Week == iWeek] <- mean(
   Neth_covidcases$NewCovidCases[Neth_covidcases$Week
    == iWeek],na.rm = TRUE)
50 }
51
52 # check if makes sense
53 plot(Neth_covidcases$date,
   Neth_covidcases$NewCovidCases)
54 lines(Neth_covidcases$date,
   Neth_covidcases$WeekAverage, type = "l", col = "red")
55
56 # make a weekend+week dummy, (1 = monday, 7 = sunday
   ) ----
```

R Script

```r
Neth_covidcases$wday <- wday(Neth_covidcases$date, week_start = getOption("lubridate.week.start", 1))
Neth_covidcases$weekend_dummy <- ifelse(Neth_covidcases$wday == 6 | Neth_covidcases$wday == 7, 1, 0)
Neth_covidcases$week_dummy <- ifelse(Neth_covidcases$weekend_dummy == 1, 0, 1)

# select for columns to be used
Neth_covidcases <- Neth_covidcases[, c("date", "weekend_dummy", "week_dummy", "NewCovidCases", "WeekAverageCovid")]


# adjust weather data to needs ----
# got a few different sources for weather data, using the detailed weather data here
weatherdataknmi_detailed$date <- as.Date(as.character(weatherdataknmi_detailed$YYYYMMDD), "%Y%m%d")

# function to get aggregate summaries, remove the NA's from missing data weather stations
seg.summ <- function(data , groups) {aggregate(data , list(groups), function(x) mean(as.numeric(x), na.rm = TRUE))}

# take average from all measurement stations and select desired columns
weather_data <- seg.summ(weatherdataknmi_detailed, weatherdataknmi_detailed$date)
weather_data$date <- as.Date(as.character(weather_data$YYYYMMDD), "%Y%m%d")

# FG         : Daily mean windspeed (in 0.1 m/s)
# TG         : Daily mean temperature in (0.1 degrees Celsius)
# SQ         : Sunshine duration (in 0.1 hour) calculated from global radiation (-1 for <0.05 hour)
# SP         : Percentage of maximum potential sunshine duration
# DR         : Precipitation duration (in 0.1 hour)
```

R Script

```
# RH        : Daily precipitation amount (in 0.1 mm
) (-1 for <0.05 mm)
# NG        : Mean daily cloud cover (in octants; 9=
sky invisible)
weather_data_selected <- data.frame(weather_data[, c
("date", "FG", "TG", "SQ", "SP", "DR", "RH", "NG")])
colnames(weather_data_selected) <- c("date", "
windspd", "temp", "sun_duration", "
perc_max_sunduration", "rain_duration", "rain_amount
", "cloud_cover")

# merge the data into 1 external dataset ----
external_data <- data.frame(merge(Neth_covidcases,
weather_data_selected))
str(external_data); summary(external_data)

# merging [redacted] and the external dataset ----
# preparing [redacted] data
[redacted]_data$session_date <- as.Date(
[redacted]_data$session_date)
[redacted]_data$cust_last_order_date <- as.Date(
[redacted]_data$cust_last_order_date)
[redacted]_data <- [redacted]_data %>% mutate_at(c("
cust_age", "cust_relationship_length", "male_u", "
female", "cust_conversion_rate", "cust_min_price", "
cust_max_price",
                                                   "
session_conversion_rate", "session_sale_dummy", "
n_session_views", "n_session_sales", "
loyal_sale_cust", "loyal_session_cust"), as.numeric)
str([redacted]_data); summary([redacted]_data)

#the actual merge
session_data <- data.frame(merge([redacted]_data,
external_data, by.x = "session_date", by.y = "date"
))

# session data descriptives ----
#structure, summary, standard deviation of session
data
str(session_data); summary(session_data)
```

R Script

```
104 options(scipen=999)
105 sapply(session_data, sd, na.rm = TRUE)
106
107 #tests to see if merge lost nothing
108 min(wehkamp_data$session_date) == min(
    session_data$session_date)
109 max(wehkamp_data$session_date) == max(
    session_data$session_date)
110 nrow(wehkamp_data) == nrow(session_data)
111
112 # the default customer
113 default_cust <- "[redacted]"
114 summary(session_data[session_data$customer_id ==
    default_cust, ])
115
116 # last row of md pattern is about the default
    customer probably
117 md.pattern(session_data)
118
119 # to do : look at missing observations and see if
    can be remedied, do the tests and visualizations
120
121 #write function for generating boxplots repeatedly
    to save time
122 session_data_num <- Filter(is.numeric, session_data)
123 boxplot_all <- function(data) {
124   for (i in 1:ncol(data)) {
125     boxplot(data[,i], xlab=names(data)[i])
126   }
127 }
128 # test for function: boxplot(
    session_data_num$cust_max_price,xlab=names(
    session_data_num)[7])
129
130 # generate boxplots for all numeric variables to
    find outliers
131 boxplot_all(session_data_num)
132
133 session_data$session_conversion_rate <-
    session_data$session_conversion_rate/100
134
```

R Script

```
135
136
137 # INFLUENCES ON CONVERSION RATE ----
138 attach(session_data)
139
140
141
142 ## INTERNAL DRIVERS ----
143
144 ### GENDER ----
145 # male has slightly higher conversion rate and
    siginificant
146 # seem to be not normal, maybe wilcoxon rank test is
     better, also gives that difference is significant
147 ks.test(session_conversion_rate, "pnorm")
148
149 #Wilcoxon Rank Test - Gender - significant
    difference!
150 wilcox.test(session_conversion_rate~female)
151
152
153 ### LOYALTY ----
154 # when using loyalty by session views over 3 months
    , result shows those view less have higher CR;
155
156 #Wilcoxon Rank Test - Two loyalty measures -
    significant diference!
157 wilcox.test(session_conversion_rate~
    loyal_session_cust)
158 wilcox.test(session_conversion_rate~loyal_sale_cust)
159
160
161 ### RELATIONSHIP LENGTH ----
162 # significant and model is better than base model,
    positive coefficent too, longer relationship =
    better
163 rl_cr_cus <- lm(cust_conversion_rate~
    cust_relationship_length, data=session_data)
164 summary(rl_cr_cus)
165 rl_cr_se <- lm(session_conversion_rate~
    cust_relationship_length, data=session_data)
```

R Script

```
summary(rl_cr_se)


### FAVORITE CHANNEL ----
#As session conversion rate is not normal, ANOVA cannot be used here, therefore we apply Kruskal-Wallis test.
kruskal.test(session_conversion_rate~factor(cust_fav_channel, ordered = FALSE))
#Kruskal-Wallis showed fav channels have significantly different session/customer conversion rates.

#### MEAN CONVERSION RATE PER CHANNEL ----
# Website has highest conversion
seg.summ(session_data[, c("cust_conversion_rate", "session_conversion_rate")], session_data$cust_fav_channel)


### FAVORITE ARTICLE GROUPS ----
# have significantly different session/customer conversion rates
kruskal.test(cust_conversion_rate~factor(cust_fav_art_group, ordered = FALSE))
kruskal.test(session_conversion_rate~factor(cust_fav_art_group, ordered = FALSE))

#### MEAN CONVERSION RATE PER ARTICLE GROUP ----
# girls fashion brands probably has insufficient data, below it is Telstar and Lds Fashion Brands Tops
seg.summ(session_data[, c("cust_conversion_rate", "session_conversion_rate")], session_data$cust_fav_art_group)


### PRICE ----
# price is significant and the lower min price, the higher conversion rate, the higher max price, the higher conversion rate as well.
```

R Script

```
price_cr_cus <- lm(cust_conversion_rate~
cust_min_price + cust_max_price)
summary(price_cr_cus)
# same is not true for session conversion rate, max
price is insignificant, min price is the same
price_cr_se <- lm(session_conversion_rate~
cust_min_price + cust_max_price)
summary(price_cr_se)


### NUMBER OF VIEWS - NUMBER OF SALES ----
# both significant, but as views go up, cr goes down
, as sales go up so does the conversion rate as
expected.
views_sales_cr_se <- lm(session_conversion_rate~
n_session_views + n_session_sales)
summary(views_sales_cr_se)
# again we see it is sales not views that help
conversion rates, we want customers to buy not
linger


### WEEKDAY/WEEKEND ----
# we see that during the week the session conversion
 rate is significantly higher
wilcox.test(session_conversion_rate~weekend_dummy)

library(bannerCommenter)



## EXTERNAL DRIVERS ----

### COVID CASES ----
# the higher new covid cases, the lower session
conversion rate was, significantly
covid_cr_se <- lm(session_conversion_rate~
WeekAverageCovid)
summary(covid_cr_se)

### WEATHER FACTORS ----
```

R Script

```
# only temperature was somewhat significant with a positive coeficient, the model was better than base overall
weather_cr_se <- lm(session_conversion_rate~windspd + temp + sun_duration + rain_duration)
summary(weather_cr_se)


# VISUALIZATIONS ----


## GENDER ----
session_data1 <- session_data %>%
  filter(female %in% c(0,1))
ggplot(data=session_data1,
       aes(x=as.factor(female), y=cust_conversion_rate)) +
  geom_boxplot()

session_data2 <- session_data %>%
  filter(female %in% c(0,1))
ggplot(data=session_data2,
       aes(x=as.factor(female), y=session_conversion_rate)) +
  geom_boxplot()

## LOYALTY ----
session_data3 <- session_data %>%
  filter(loyal_session_cust %in% c(0,1))
ggplot(data=session_data3,
       aes(x=as.factor(loyal_session_cust), y=cust_conversion_rate)) +
  geom_boxplot()

session_data4 <- session_data %>%
  filter(loyal_session_cust %in% c(0,1))
ggplot(data=session_data4,
       aes(x=as.factor(loyal_session_cust), y=session_conversion_rate)) +
  geom_boxplot()

```

R Script

```
255 session_data5 <- session_data %>%
256   filter(loyal_sale_cust %in% c(0,1))
257 ggplot(data=session_data5,
258        aes(x=as.factor(loyal_sale_cust), y=
    cust_conversion_rate)) +
259   geom_boxplot()
260
261 session_data6 <- session_data %>%
262   filter(loyal_sale_cust %in% c(0,1))
263 ggplot(data=session_data6,
264        aes(x=as.factor(loyal_sale_cust), y=
    session_conversion_rate)) +
265   geom_boxplot()
266
267 ## WEEK/WEEKEND ----
268 ggplot(data=session_data,
269        aes(x=as.factor(weekend_dummy), y=
    session_conversion_rate)) +
270   geom_boxplot()
271
272 ggplot(data=session_data,
273        aes(x=as.factor(weekend_dummy), y=
    cust_conversion_rate)) +
274   geom_boxplot()
275
276 ## LINEAR MODELs ----
277 session_data %>%
278   select(cust_conversion_rate,
    cust_relationship_length, cust_min_price,
    cust_max_price) %>%
279   pairs
280 #Careful! figure below takes long time to generate
281 session_data %>%
282   select(session_conversion_rate,
    cust_relationship_length, cust_min_price,
    cust_max_price,
283          n_session_views, n_session_sales,
    WeekAverageCovid,
284          windspd, temp, sun_duration, rain_duration
    ) %>%
285   pairs
```

R Script

```
## BAR ----
channel_avg_cr <- seg.summ(session_data[, c("cust_conversion_rate", "session_conversion_rate")], session_data$cust_fav_channel)
channel_avg_cr <- channel_avg_cr[-2,]
channel_avg_cr <- channel_avg_cr[-1,]
colnames(channel_avg_cr)[1] <- 'Channels'

detach(session_data)
attach(channel_avg_cr)
ggplot(data=channel_avg_cr)+
  geom_bar(aes(y=session_conversion_rate, x=reorder(Channels,session_conversion_rate)),
           fill=4,
           stat="identity", show.legend=FALSE)+
  ggtitle("Mean session conversion rate by channels")+
  theme(plot.title=element_text(hjust = 0.5))+
  theme(axis.text = element_text(size=14))+
  theme(axis.title = element_text(size=14))+
  theme(plot.title = element_text(size=14))+
  xlab("")+ylab("Session Conversion Rate")+
  coord_flip()

article_avg_cr <- seg.summ(session_data[, c("cust_conversion_rate", "session_conversion_rate")], session_data$cust_fav_art_group)
article_avg_cr <- article_avg_cr[-c(14,15,8,5),]
colnames(article_avg_cr)[1] <- "article_groups"
detach(channel_avg_cr)
attach(article_avg_cr)
ggplot(data=article_avg_cr)+
  geom_bar(aes(y=session_conversion_rate, x=reorder(article_groups,session_conversion_rate)),
           fill='#f15c80',
           stat="identity", show.legend=FALSE)+
  ggtitle("Mean session conversion rate by article groups")+
  theme(plot.title=element_text(hjust = 0.5))+
  theme(axis.text = element_text(size=10))+
```

R Script

```
319   theme(axis.title = element_text(size=14))+
320   theme(plot.title = element_text(size=14))+
321   xlab("")+ylab("Session Conversion Rate")+
322   coord_flip()
323
324
```