Instacart Grocery Basket Analysis

Case study by Julia Fortuny Wollny

2021

Context

Instacart is an online grocery store that operates through an app. They already have very good sales but they want to **uncover more information about sales patterns.**

The Instacart stakeholders are most interested in the variety of customers in their database along with their purchasing behaviours. They assume they can't target everyone using the same methods, and they're considering a targeted marketing strategy. They want to **target different customers with applicable marketing campaigns** to see whether they have an effect on the sale of their products.

Objective

Perform an **initial data and exploratory analysis** of some of their data in order to derive insights and suggest strategies for better segmentation based on the provided criteria.

My analysis would inform what possible strategies could look like to ensure Instacart targets the right customer profiles with the appropriate products.

Some key questions were:

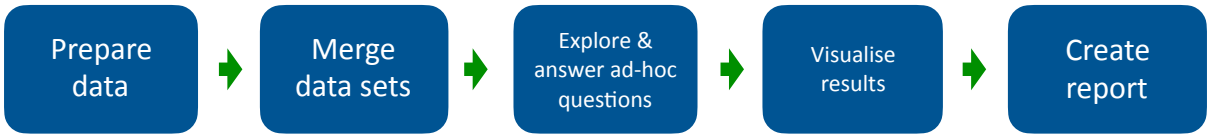
- What are the busiest days of the week and hours of the day?
- At what time do people spend the most money?
- Are there differences in ordering habits based on a customer's loyalty status?
- What purchasing differences are there between different customer profiles?

Role:	**Project duration:**
Data Analyst	1 month, delivered on time
Data used:	**Tools used:**
Open-source data sets from Instacart containing details on products, transactions and customers. **Download** project brief here.	• Anaconda navigator • Python for data analysis • NumPy, Pandas, Matplotlib, Scipy and Seaborn libraries • Jupyter notebook • Excel

Skills

✦ Jupyter fundamentals	✦ Combining & exporting data
✦ Checking, adjusting data types	✦ Deriving new variables, columns & flags
✦ Pandas dataframes	✦ Grouping data & aggregating variables
✦ Data wrangling & subsetting with Python	✦ Data visualisation with Python
✦ Data consistency checks with Python	✦ Coding etiquette & Excel reporting

Approach & Methodology



I started by importing two data sets for orders and products to my Jupyter notebook.

I then wrangled the data to suit the needs of my upcoming analysis. This included:

- Dropping columns
- Renaming columns
- Changing data types
- Transposing data

Here I adjusted some data types:

```
[63]: # Check data types df_ords
      df_ords.dtypes

t[63]: order_id                  int64
       user_id                   int64
       order_number              int64
       order_dow                 int64
       order_hour_of_day         int64
       days_since_prior_order  float64
       dtype: object

[64]: # Change order_id, user_id, order_number to string because not numeric
      df_ords['order_id'] = df_ords['order_id'].astype('str')
      df_ords['user_id'] = df_ords['user_id'].astype('str')
      df_ords['order_number'] = df_ords['order_number'].astype('str')

[65]: # Change order_dow, order_hour_of_day, days_since... to save on RAM
      df_ords['order_dow'] = df_ords['order_dow'].astype('int8')
      df_ords['order_hour_of_day'] = df_ords['order_hour_of_day'].astype('int8')
      df_ords['days_since_prior_order'] = df_ords['days_since_prior_order'].astype('float16')

[66]: df_ords.dtypes

t[66]: order_id                  object
       user_id                   object
       order_number              object
       order_dow                   int8
       order_hour_of_day           int8
       days_since_prior_order   float16
       dtype: object
```

Checking and adjusting data types is very important as we can save on memory usage. Python automatically assigns 64-bit versions to numeric variables. Oftentimes this is not necessary.

I also created subsets, such as for breakfast items in the products set, to further analyse this specific group:

```
In [88]: # Create subset for breakfast items
         df_breakfast =  df_prods[df_prods['department_id']==14]

In [89]: df_breakfast.head()
```

Out[89]:

	product_id	product_name	department_id	prices
27	28	Wheat Chex Cereal	14	10.1
33	34	NaN	14	12.2
67	68	Pancake Mix, Buttermilk	14	13.7
89	90	Smorz Cereal	14	3.9
210	211	Gluten Free Organic Cereal Coconut Maple Vanilla	14	3.6

Then I conducted basic consistency checks of each dataframe, including:

- Find and address mixed type variables
- Find and address missing values
- Find and address duplicate values

```
In [113]: # Df_ords: Check for missing values
          df_ords.isnull().sum()

Out[113]: order_id                      0
          user_id                       0
          order_number                  0
          orders_day_of_week            0
          order_hour_of_day             0
          days_since_prior_order   206209
          dtype: int64
```

There are 206209 missing values in the column days since prior order. In fact, these should be zeroes because it simply means the customer was a new customer at this point, so they didn't have any prior orders.

These were not real missing values, so I inputed 0 (zero).

In the following example there were 2 values with unrealistically high prices. I replaced those with NaN (= missing):

```
[140]: # Check summary statistics
       df_prods_clean_no_dups.describe()
```

t[140]:

	product_id	department_id	prices
count	49672.000000	49672.000000	49672.000000
mean	24850.349775	11.728942	9.993282
std	14340.705287	5.850779	453.615536
min	1.000000	1.000000	1.000000
25%	12432.750000	7.000000	4.100000
50%	24850.500000	13.000000	7.100000
75%	37268.250000	17.000000	11.100000
max	49688.000000	21.000000	99999.000000

We have the unusually high price of 99999.

```
[141]: # Check for all outliers price > 100
       df_prods_clean_no_dups.loc[df_prods_clean_no_dups['prices'] > 100]
```

t[141]:

	product_id	product_name	department_id	prices
21554	21553	Lowfat 2% Milkfat Cottage Cheese	16	14900.0
33666	33664	2 % Reduced Fat Milk	16	99999.0

There are 2 products where the price is obviously wrong. This must be corrupted somehow.

I then exported the data set with the changes.

Next, I combined the orders and product data sets by **merging** them. This was necessary to conduct further analyses on the entire data.

I chose to merge, because the shape of both data sets wasn't the same, and I chose an **inner join** because I was only interested in data which matched completely.

Because of its size, this new data set was exported as a **pickle**.

I then **derived new variables** using .loc and if-statements. I created the following columns:

- Price range: to determine whether a product is low, mid or high range,
- Busiest day: to determine if a transaction happened on the least busy, regular or busiest weekday,
- Busiest period of the day: to determine if a transaction happened when fewest, average or most orders happened.

I used **group and aggregation** functions to **derive new columns**, such as:

A **loyalty flag** for customers based on the maximum orders they placed.

```
In [9]: ords_prods_merge.loc[ords_prods_merge['max_order'] > 40, 'loyalty_flag'] = 'Loyal customer'

In [12]: ords_prods_merge.loc[(ords_prods_merge['max_order'] <= 40) & (ords_prods_merge['max_order'] >

In [13]: ords_prods_merge.loc[ords_prods_merge['max_order'] <= 10, 'loyalty_flag'] = 'New customer'

In [14]: ords_prods_merge.head()
```

Out[14]:

ed	product_name	department_id	prices	_merge	price_range_loc	busiest_day	busiest_period_of_day	max_order	loyalty_flag
0	Soda	7	9.0	both	Mid-range product	Regularly busy	Average orders	15	Regular customer
1	Soda	7	9.0	both	Mid-range product	Least busy	Average orders	15	Regular customer
1	Soda	7	9.0	both	Mid-range product	Least busy	Average orders	15	Regular customer
1	Soda	7	9.0	both	Mid-range product	Least busy	Average orders	15	Regular customer
1	Soda	7	9.0	both	Mid-range product	Least busy	Most orders	15	Regular customer

In this way, I could start drawing the first insights and answering the first ad-hoc questions, such as:

- **What type of customer has the highest spending?**

```
In [16]: # Check basic statistics of prices grouped by loyalty flag
ords_prods_merge.groupby('loyalty_flag').agg({'prices': ['mean', 'min', 'max']})
```

Out[16]:

	prices		
	mean	min	max
loyalty_flag			
Loyal customer	7.778902	1.0	25.0
New customer	7.798587	1.0	25.0
Regular customer	7.800433	1.0	25.0

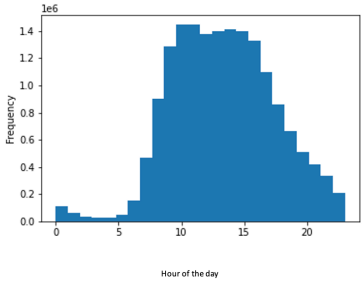
The price mean of loyal customers is the lowest, while the price mean of regular customers is the highest.

The most loyal customers have the lowest spending.

I then proceeded to create **spending** and **order frequency flags**.

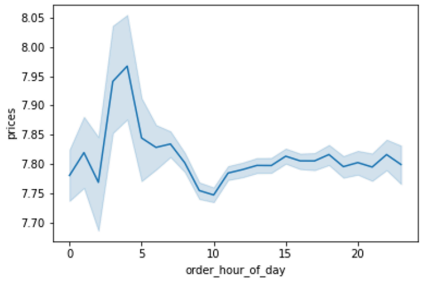
Following that I went on to answer more **ad-hoc business questions** using visualisations.

- **Which is the busiest hour of the day?**



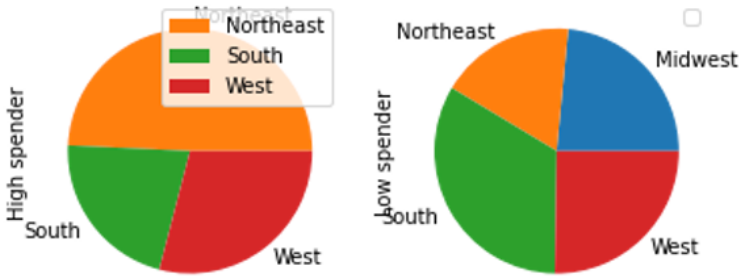
The busiest hours of the day are between 9 and 11 am and the least busy are between 11 and 12 pm.

- **What time of the day are the most expensive items bought?**



The hour of the day at which the most expensive items are bought is at 5 am.

- **Are there differences between regions?**

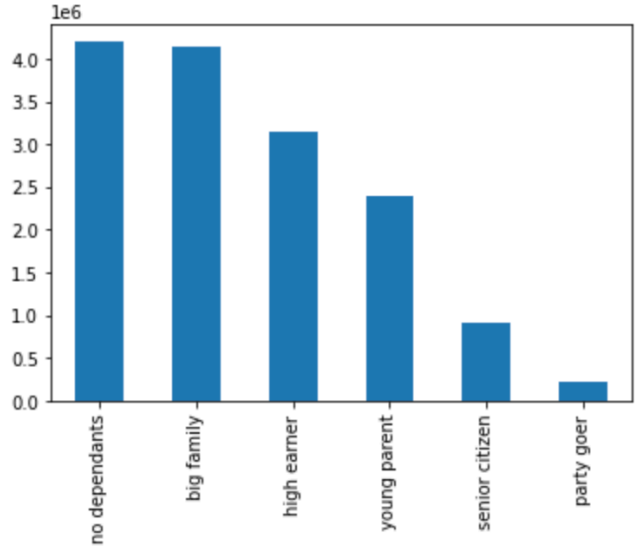


Most high spenders are in the Northeast and most low spenders in the South.

Finally, I created 5 customer profiles to segment the Instacart customer base:

- Senior citizen
- High earner
- Big family
- Party goer
- No dependents

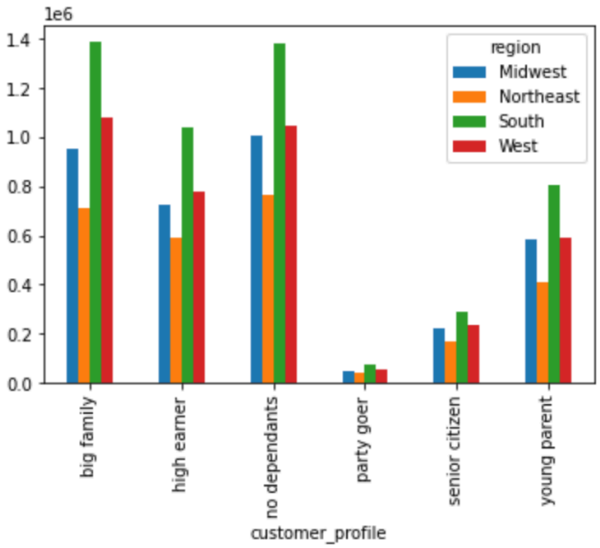
I could now show how big each group is and how they compare across regions.



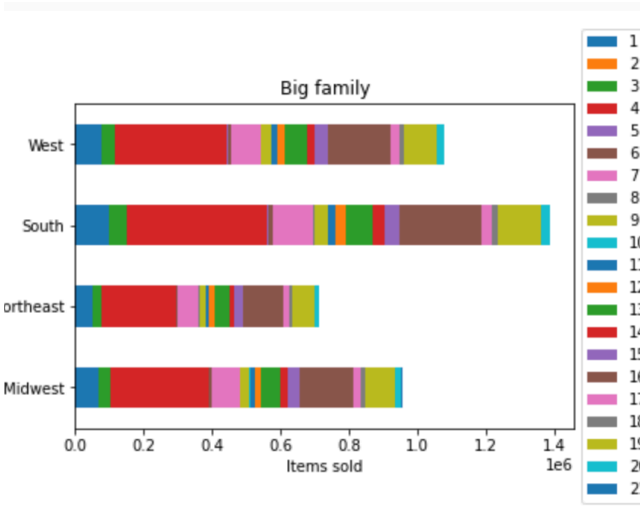
Customers with no dependants are the most numerous group of customers of Instacart across all regions.

Party goers are the smallest group.

The regional differences within each group are similar.



I finished by visualising the **number of items from each department** bought **by customer profile and region**. Here is the chart for the big family customer profile.



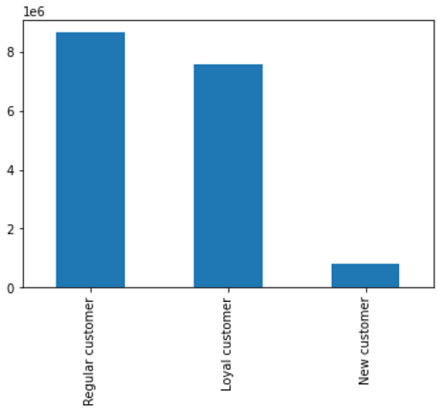
Customers with big families bought most items **in the South** and the **best-selling department was 4, "produce".**

I compiled all my findings, including the population flow and wrangling steps taken, into a report to stakeholders.

The **final report** can be downloaded here.

My **code** for this project can be found here.

Recommendations



Most customers are regular customers, with a maximum of orders of more than 10 but less than 40. The smallest group are the new customers. **I would recommend to focus on incentives for new customers to return.**

More recommendations can be found in the final report.

Future steps

Implement one of the recommendations, such as the one mentioned above, in a marketing campaign.

Then, **track and measure results** and use these to **conduct a new analysis** to **assess the impact of the marketing campaign**.

Deliverables

- Answers to ad-hoc business questions,
- Final report for stakeholders, download here,
- Scripts and code, here.

Find me on:

Tableau: https://public.tableau.com/app/profile/julia.fortuny

LinkedIn: https://www.linkedin.com/in/julia-f-18144718/

Github: https://github.com/juliafor/