Class 5 Data Wrangling with R

Dr. Wei Miao

UCL School of Management

October 16, 2024

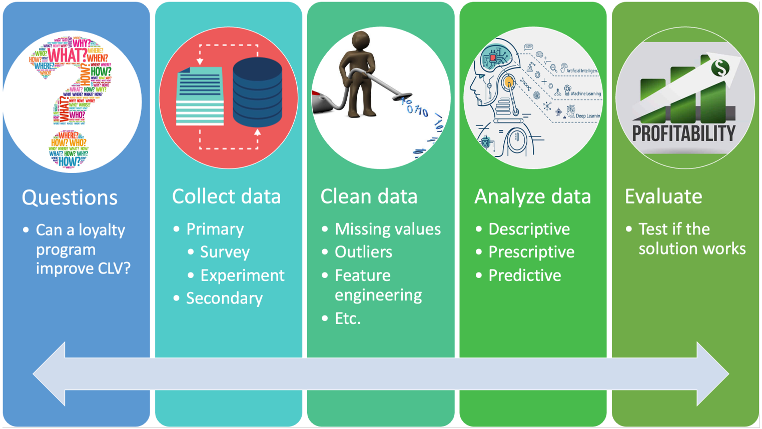
Section 1

Data Analytics Workflow

Class Objectives

- Understand the major steps to conduct data analytics. We will use improve the marketing efficiency for the M&S case study as an example.
- **Data collection:** Learn how to collect first-hand survey data and how to load second-hand data into R
- **Data cleaning:** Learn how to use the `dplyr` package to clean data
- **Data analysis:** Learn how to conduct descriptive analytics for the M&S case study

Overview of a Data Analytics Project



Questions

- Can a loyalty program improve CLV?

Collect data

- Primary
  - Survey
  - Experiment
- Secondary

Clean data

- Missing values
- Outliers
- Feature engineering
- Etc.

Analyze data

- Descriptive
- Prescriptive
- Predictive

Evaluate

- Test if the solution works

Business Objective: Our Business Question in Weeks 3 - 5

Our project for M&S in Weeks 3-5: Help M&S to **improve marketing efficiency** by improving its **ROI** on its targeted marketing offers. The project will involve data collection, data cleaning, and data analysis, including both **descriptive** and **prescriptive** analytics, to identify the most profitable customers and develop a personalized marketing targeting strategy.

Business Objective: Example Dissertation Projects in Term 3

- ***Burberry*** provides relevant **product recommendations** on Burberry.com to facilitate in-session **product exploration** and to create a more personalised user experience. This project is to develop a new **product recommendation system** that tailors suggestions to individual users based on their product selections and preferences.
- The ***AXA*** project will explore **fraud detection approaches** using **unsupervised ML** including models such as isolation forests. The candidate will develop an understanding of the business problem and our data, formulating hypotheses and testing them. They will build, evaluate, and interpret their ML models.
- At ***Waitrose***, it's crucial to balance product availability with minimizing waste by understanding sales rates. Factors like product shelf life, varying sales velocities, promotions, and unexpected trends make it challenging to find a one-size-fits-all solution. Current manual forecasting introduces inaccuracies and process delays. We aim to develop a **machine learning algorithm** to generate daily **product forecasts**, integrate with our stock management system, and enable automated, accurate forecasting.

Section 2

Data Collection

Types of Data by Source

- **Primary Data:** Data that are generated by the data analyst through surveys, interviews, experiments, specially designed for understanding and solving the research problem at hand.
- **Secondary Data:** Existing data generated by the company's or consumer's past activities, as part of organizational record keeping.

Basis for Comparison	Primary Data	Secondary Data
Meaning	Primary data refers to the first-hand data gathered by the analyst.	Secondary data means data collected by someone else earlier (usually by the company).
Data	New data	Historical data in the past
Source	Surveys, observations, experiments, questionnaire, personal interviews, etc.	Company databases, government publications, websites, books, journal articles, internal records, etc.
Cost	Expensive; Very involved and costly	Economical; Quick and easy
Collection time	Long	Short
Specific	Always specific to the researcher's needs.	May or may not be specific to the researcher's needs.

Types of Data by Structure

- We often consider 2 dimensions of a dataset: unit and time.
  - **Cross-sectional data**: data collected at a single point in time.
  - **Longitudinal data**: data collected over time but may not contain the same individuals.
  - **Panel data**: data collected over time and contain the same individuals.

Primary Data: Marketing Surveys

- A marketing survey is often the easiest and most cost-effective way to collect primary data. We often collect the following variables:
    - purchase intention: how likely a customer will buy a product, helps to predict sales
    - willingness to pay: how much a customer is willing to pay for a product, helps to set the optimal price
    - shopping basket: what products a customer usually buys, helps to cross-sell
    - share of wallet: how much a customer spends on a product category, helps to identify the high-potential customers for market penetration
    - demographics: gender, age, income, education, etc., helps to segment customers
- Let's see an example in Mentimeter of how to design a marketing survey!
- You can design surveys to collect data for your term 1 projects or term 3 dissertation.
    - The quick start guide on how to conduct market research surveys

Limitation of Marketing Surveys

- **Hawthorne Effect** and **Response Bias**: Participants may answer in ways they think are socially desirable or expected, rather than their true feelings or behavior.
- **Sampling Bias**: The sample may not be representative of the customer population.
- **Fatigue**: Long surveys may lead to respondent fatigue, causing rushed or careless answers toward the end. Do not ask too many questions!

Section 3

Data Wrangling with R

Data Frame Basics

- Data Frame is the R object that we will deal with most of the time in the MSc program. You can think of `data.frame` as an Excel spreadsheet.
    - Each row stands for an **observation**; usually a record for a customer
    - Each column stands for a **variable**; each column should have a unique name.
    - Each column must contain the same data type, but the different columns can store different data types.

Install and Load the `dplyr` package

- In R, we use the `dplyr` package for data cleaning and manipulation.[1]

```
pacman::p_load(dplyr)
```

- Load a csv format dataset called `data_full` using `read.csv()`

```
data_full <- read.csv("https://www.dropbox.com/scl/fi/2q7ppqtyca0pd3j48
```

- To browse the whole dataset, we can simply click the dataset in the environment

[1] pacman is a package management tool that makes our lives easier by loading multiple packages at once.

First Look at the Dataset

- What variables do the data have? The data types of each variable?

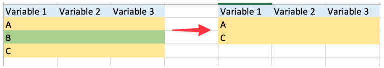
```r
str(data_full)
```

Common Data Wrangling Operations

- **Filter rows** (`filter`)
- **Sort rows** (`arrange`)
- **Select columns** (`select`)
- **Generate new columns** (`mutate`)
- **Group aggregation** (`group_by`)
- Merge datasets (`join`)

Subset Rows Based on Conditions: `filter`

- We can use `filter()` to select rows that meet certain logical criteria.



- **Important**: To store the generated new subset of data in RStudio, we need to assign it to a new object.

Subset Rows Based on Conditions: `filter`

Example: From data_full, find customers who are single

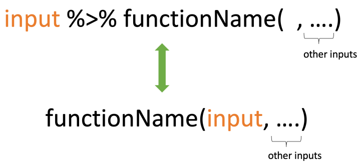
```
# keep only single customers
filter(data_full, Marital_Status == "Single" )
```

The Pipe Operator

Pipe Operator

%>% passes the **object in front** as the **first argument** of the **subsequent function**.[a]

[a] Starting from R 4.0 version, base R introduces the native pipe operator |>



Example of the Pipe Operator

```r
# without using pipe
filter(data_full, Marital_Status == 'Single')

# with pipe
data_full %>% filter(Marital_Status == 'Single')
```

Why Do We Need Pipe Operator for Data Wrangling?

- **Exercise**: find out **single** customers who have a **PhD** without using pipe.

```
# based on data_full, find out customers who are single
data_full_single <-

# based on data_full_single,
# further find out single customers who have a PhD
data_full_single_PhD <-
```

Why Do We Need Pipe Operator for Data Wrangling?

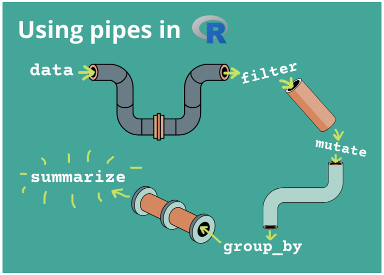
- **Exercise**: find out **single** customers who have a **PhD** using pipe.

```
data_full_single_PhD <- data_full %>%
  filter(Marital_Status == 'Single') %>%
  filter(Education == 'PhD')

  ## You can even continue with more filter steps with more pipe operat
```

Why Do We Need Pipe Operator for Data Wrangling?

- The pipe works like a conveyor belt in a factory, passing the intermediate outputs from the previous data wrangling step to the next step for further processing until you finish your data wrangling task.



Sort Rows: `arrange`

- `arrange()` orders the rows by the values of selected columns.
    - ascending order by default; for descending order, put a minus sign in front of the variable.
    - allows multiple sorting variables separated by comma.
- **Example**: sort customers based on income in descending order.

```
data_full %>%
  arrange(-Income)
```

- **Exercise**: sort customers based on income in descending order and age in ascending order.

Generate New Variables: `mutate`

- `mutate()` generates new variables in the dataset while preserving existing variables
- **Example**: create a new variable named `Age` from `Year_Birth`.

```
data_full %>%
  mutate(Age = 2023 - Year_Birth)
```

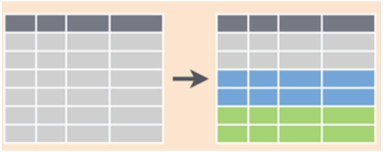
- **Exercise**: create a new variable named `totalkids`, which is the sum of `Kidhome` and `Teenhome`.

Aggregation by Groups: `group_by`

- `group_by()` allows us to aggregate data by group and compute statistics for each group

```
# group by marital status
data_full %>%
    group_by(Marital_Status)
```

- Internally, the dataset is already grouped based on the specified variable(s).



Aggregation by Groups: `group_by() %>% summarise()`

- After aggregating data, we can use `summarise()` to compute group-specific statistics for us.
  - Similar to `mutate()` in generating new variables
  - Different from `mutate()` in that the new variable is computed based on groups.

```r
# compute the average income for each marital status group
data_full %>%
  group_by(Marital_Status) %>%
  summarise(avg_income = mean(Income,na.rm = T)) %>%
  ungroup()
```

- What if you replace `summarise()` with `mutate()`?

Aggregation by Groups: `group_by()` Multiple Groups

- We can have multiple group variables for `group_by` , such as computing average income for each marital status, education combination

```
# compute the average income for each marital, education group
data_full %>%
  group_by(Marital_Status,Education) %>%
  summarise(avg_income = mean(Income,na.rm = T)) %>%
  ungroup() %>%
  head(5)
```

After-Class

- (essential) Cheatsheet for `dplyr`. This cheatsheet provides a quick reference for the most commonly used functions in the `dplyr` package. It's very important to familiarize yourself with these functions as you will use them a lot in your future projects.
- (optional) Complete the after-class exercise for Week 3. If you still have time, you can also complete the data camp exercise on the dplyr package. The link is here.