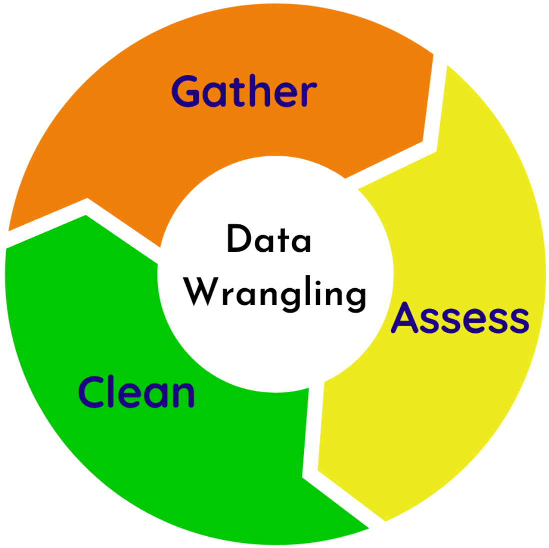

Lecture 5

Data Wrangling

What is Data Wrangling?

- Data wrangling involves getting source data ready for effective and precise analysis.
- This encompasses tasks such as eliminating incomplete entries, standardizing feature formats, and integrating relevant information from external sources.
- Data wrangling is also referred to as data munging or data preparation in some contexts.
- Data wrangling has six steps.

Steps of Data Wrangling

Step	Description
Step 1: Discovering	Discovery, also called data exploration, familiarizes the data scientist with source data in preparation for subsequent steps.
Step 2: Structuring	Structuring data transforms features to uniform formats, units, and scales.
Step 3: Cleaning	Cleaning data removes or replaces missing and outlier data.
Step 4: Enriching	Enriching data derives new features from existing features and appends new data from external sources.
Step 5: Validating	Validating data verifies that the dataset is internally consistent and accurate.
Step 6: Publishing	Publishing data makes the dataset available to other data scientists by storing data in a database, uploading data to the cloud, or distributing data files.

Extract, Transform, Load (ETL)

- ETL: Extracts, transforms, and loads data from transactional to analytic databases. ELT loads raw data and transforms in-place.

- ETL vs. Data Wrangling: Both structure, clean, enrich data. ETL automates for new data, larger volumes. Wrangling manual, static datasets.

- ETL Tools: Extract and merge data from databases. Data wrangling uses manual methods like spreadsheets, Python, R, SQL.

Data Wrangling with R

- `dplyr`, an R package, aids data wrangling on `tidy` dataframes. Tidy data has one instance per row, each column representing a feature with a specific data type (e.g., character, numeric).

- The `tidyverse` is a set of R packages for data wrangling, data visualization, and modeling with a common structure and syntax. `tidyverse`-style packages prioritize readability over conciseness, and use a different syntax than base R.

Data Wrangling with Python

- pandas, a Python package, facilitates data wrangling. A pandas DataFrame class stores and handles datasets. Here, "dataframe" in lowercase denotes a DataFrame object.

- Dataframes comprise rows and columns, representing instances and features. Columns have data types. They are labeled with integers or strings.

- Row labels are the index and column labels are columns. Typically, row labels are auto-generated integers, while column labels are specified strings.

`dplyr` in R

- `dplyr` offers six key functions for data wrangling, known as verbs, each representing a specific action. These functions can be combined using the pipe operator, `|>`, for complex operations. For more information on dplyr data wrangling, refer to its documentation.

- Use `glimpse()` to check dataframes. `tidyverse` functions accept both American and British English spellings, like "`summarize`" = "`summarise`" and "`color`" = "`colour`".

dplyr in R

Function	Description
mutate()	Calculate new features using functions of existing features
select()	Chooses features based on column names
filter()	Chooses instances based on values of one or more features
summarize()	Calculate one or more descriptive statistics from features
arrange()	Sort rows in a dataframe based on one or more features
group_by()	Group instances into subsets based on values of one or more features (typically combined with another data wrangling verb)

Manipulating Data

- Data wrangling starts with data discovery, where patterns are explored. This can be visual through plots or numerical with stats.

- Data manipulation organizes datasets for research. It's used to group, compare, and calculate stats.

Grouping Data

- Grouping splits a dataframe into subsets based on a categorical feature.
- Groups can have unique analyses/models or aid calculations like group sizes or means.
- A frequency table shows categorical feature group sizes.

Pivot Tables

- A pivot table computes stats by grouping two categorical features.
- Rows and columns represent the features.
- A contingency table is a pivot table, counting instances in categorical feature combinations.

Data Manipulation in R

- `dplyr` offers three functions for frequency, contingency, and pivot tables. These aid the group-calculate-combine approach for descriptive stats.
  - Group the dataset: `group_by()` groups the dataset based on one or two categorical features.
  - Calculate descriptive statistics: `summarize()` calculates descriptive statistics within each group.
  - Combine means: For a single grouping feature, the table is in the right format. With two features, `spread()` restructures it.
- `group_by()`, `summarize()`, and `spread()` can be nested within one another. But, many R users prefer to use the pipe, `|>`, for better readability.

Data Manipulation in Python

- The `pandas` package offers two data manipulation methods. `df.groupby()` divides a dataframe `df` into subsets, while `df.pivot_table()` creates pivot tables using two categorical features. Both can be used alongside pandas' descriptive statistics functions.

- `df.groupby()` uses the 'by' parameter to set the grouping feature. Missing values can be eliminated with `dropna=True` or placed in a distinct category with `dropna=False`. More parameters are detailed in the group by documentation.

- `df.pivot_table()` has various parameters. '`value`' designates pivot table elements, '`index`' specifies rows, and '`columns`' specifies columns. '`aggfunc`' sets a function for row/column values, defaulting to `np.mean`.

Feature Scaling

- Numeric features in datasets vary in scales, sometimes drastically. Algorithms perform better with similar scales. Scaling normalizes features to consistent ranges. Common methods: standardization and normalization.
- **Standardization** converts features to a range centered at 0, with 1 representing a standard deviation:

$$x_{standardized} = \frac{x_{original} - \mu_x}{\sigma_x}$$

- The standardized value is called a ***z-score***. Since each unit represents one standard deviation, most z-scores fall between -2 and 2.

- **Normalization** converts features to the range [0,1]:

$$x_{normalized} = \frac{x_{original} - \min x}{\max x - \min x}$$

Feature Scaling

- Standardization is preferred due to positioning values around mean and standard deviation. Normalization suits algorithms needing uniform scales.

- Standardization is best when outliers are present. Standardized values are not skewed by outliers, but most normalized values are compressed into a small range.

- Feature scaling terminology varies. Standardization is sometimes called z-score normalization. Normalization is sometimes called min-max scaling.

Uncleaned/Dirty Data

- Dirty datasets often contain missing, outlier, and duplicate data.
  - **Missing data** is value unknown or inapplicable. In databases, it's `NULL`. In R, it's `NA`. Other software uses NaN or NaT.
  - **Outliers** are numeric values far from others in the same feature. Typically, they're 2-3 standard deviations from the mean.
  - **Duplicates** are identical instances in data. They're often errors and should be deleted.
- Missing, outlier, and duplicate data are collectively called **dirty data**.
- Dirty data introduces bias and inefficiency in analysis. Missing data poses interpretation challenges. Erroneous duplicates overly influence with frequent appearance. Outliers distort results due to potential errors.

Handling Dirty Data

- **Dirty Data Removal:**
  - Discard instances if small random part is dirty.
  - Remove features with high missing values, not outliers/duplicates.
  - Pairwise discard complex, less common due to varying instance counts.

- **Data Imputation:**
  - Replace missing/outliers with new values.
  - Use hot-/cold-deck from same/different instances.
  - Mean imputation: replace with mean (exclude missing/outliers).
  - Regression imputation: value from regression, useful for high correlation.

R Data Cleaning Functions

Function	Package	Description
`is.na(vector)`	`Base`	Logical operator, returns `TRUE` for missing values and `FALSE` otherwise.
`complete.cases(df)`	`Base`	Logical operator, returns a list with `TRUE` for rows in a dataframe `df` with no missing values and `FALSE` for rows with at least one missing value.
`na.omit(df)`	`Base`	Omits all rows with at least one missing value.
`distinct(df)`	`dplyr`	Select only unique rows from a dataframe.
`drop_na(df)`	`dplyr`	Drop rows with missing values.
`replace_na(df)`	`dplyr`	Replace missing values from a given list of values.
`select(df)`	`dplyr`	Keep a list of features, or drop features by adding a negative sign. Ex: `-feature` removes `feature` from the dataframe.

Python Data Cleaning Functions

Method	Parameters	Description

`df.drop()`	`labels=None`
	`axis=0`
	`inplace=False`
		Removes rows (**`axis=0`**) or columns (**`axis=1`**) from dataframe **`df`**. **`labels`** specifies the labels of rows or columns to drop.

`df.drop_duplicates()`	`subset=None` `inplace=False`	Removes duplicate rows from **`df`**. **`subset`** specifies the labels of columns used to identify duplicates. If **`subset=None`**, all columns are used.

`df.dropna()`	`axis=0`
	`how='any'`
	`subset=None`
	`inplace=False`
		Removes rows (**`axis=0`**) or columns (**`axis=1`**) containing missing values from **`df`**. **`subset`** specifies labels on the opposite axis to consider for missing values. **`how`** indicates whether to drop the row or column if **`any`** or if **`all`** values are missing.

`df.duplicated()`	`subset=None`	Returns a Boolean series that identifies duplicate rows in **`df`**. **`true`** indicates a duplicate row. **`subset`** specifies the labels of columns used to identify duplicates. If **`subset=None`**, all columns are used.
`df.fillna()`	`value=None` `inplace=False`	Replaces NA and NaN values in **`df`** with **`value`**, which may be a scalar, dict, Series, or DataFrame.
`df.isnull()` `df.isna()`	*none*	Returns a dataframe of Boolean values. **`True`** in the returned dataframe indicates the corresponding value of the input **`df`** is None, NaT or NaN.

`df.mean()`	`axis=0`
	`skip_na=True`
	`numeric_only=None`
		Returns the mean values of rows (**`axis=0`**) or columns (**`axis=1`**) of **`df`**. **`skipna`** indicates whether to exclude unknown values in the calculation. **`numeric_only`** indicates whether to exclude non-numeric rows or columns.

`df.replace()`	`to_replace=None`
	`value=NoDefault.no_default`
	`inplace=False`
		Replaces **`to_replace`** values in **`df`** with **`value`**. **`to_replace`** and **`value`** may be str, dict, list, regex, or other data types.

Appending Data

- Datasets can be enriched by appending new instances or features from external datasets. Leading sources of public datasets are described in the table below:

Name	Link	Description
Kaggle	kaggle.com	Over 50,000 datasets on a broad range of subjects. Also provides Jupyter notebooks that analyze the datasets.
FiveThirtyEight	data.fivethirtyeight.com	Datasets on politics, sports, science, economics, health, and culture, initially developed to support FiveThirtyEight publications.
University of California Irvine Machine Learning Repository	archive.ics.uci.edu	622 datasets, primarily in science, engineering, and business.
Data.gov	data.gov	U.S. government datasets on agriculture, climate, energy, maritime, oceans, and health.
World Bank Open Data	data.worldbank.org	Global datasets on subjects such as health, education, agriculture, and economics.
Nasdaq Data Link	data.nasdaq.com	Financial and economic datasets.

Case Study

Diamond Prices

Next Lecture

Data Exploration