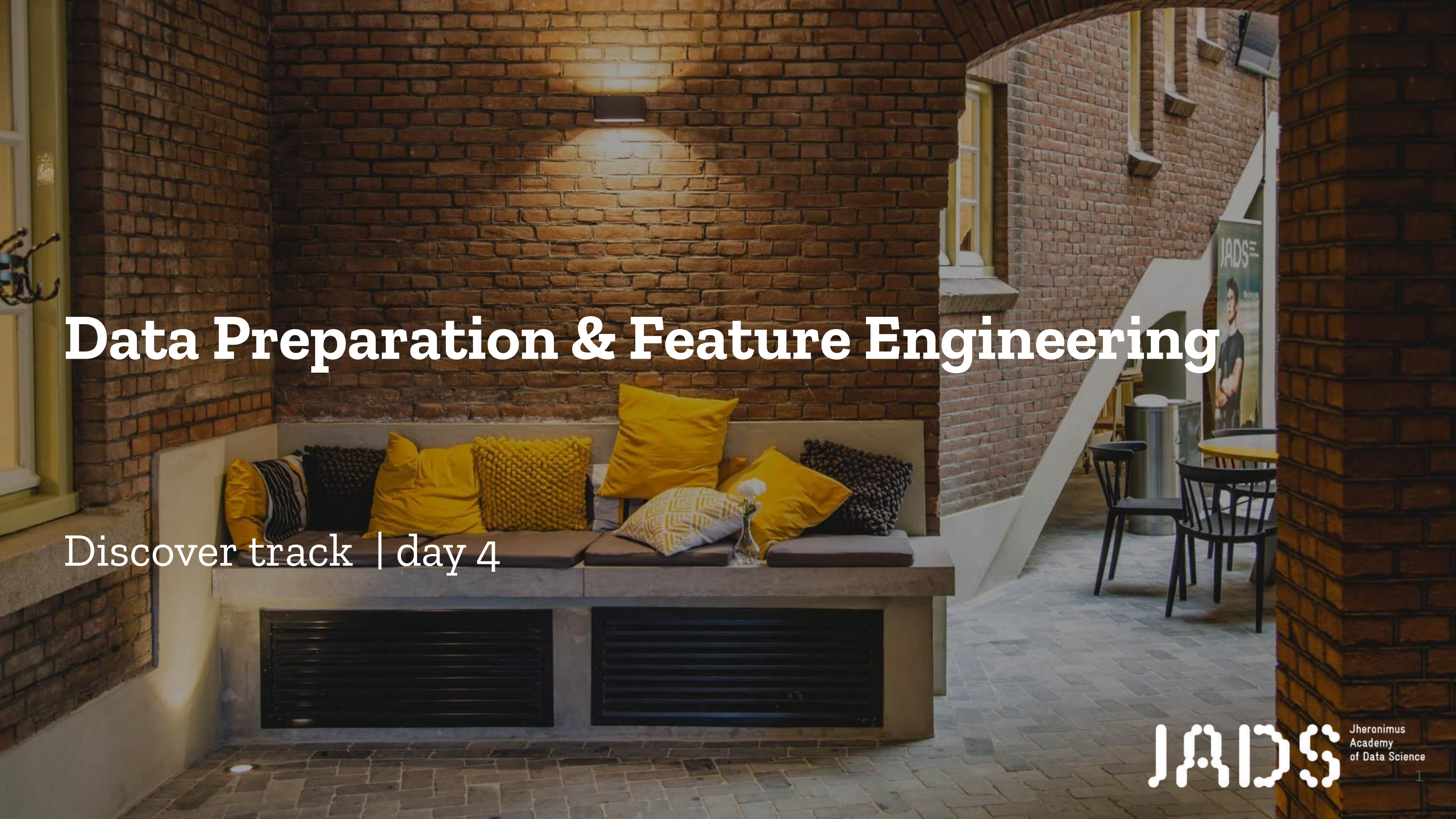
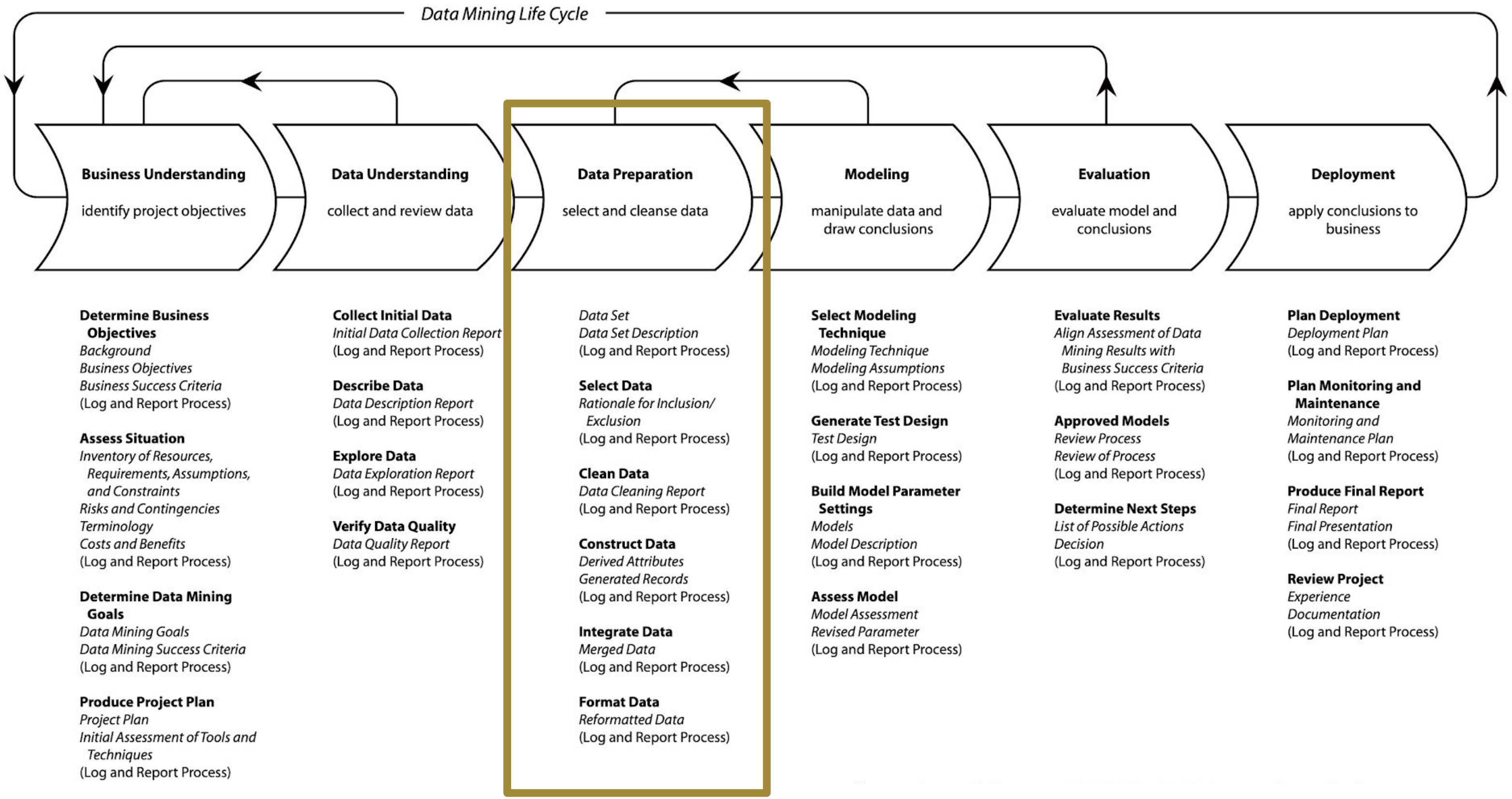


Data Preparation & Feature Engineering

Discover track | day 4

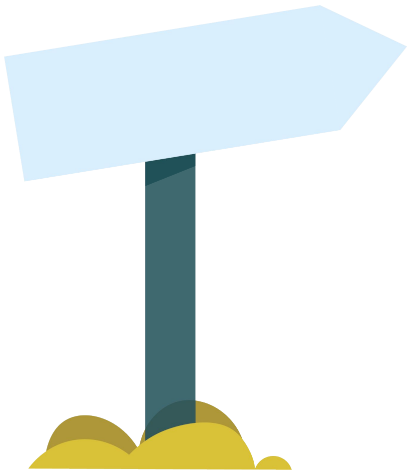
Jheronimus
Academy
of Data Science

Data preparation



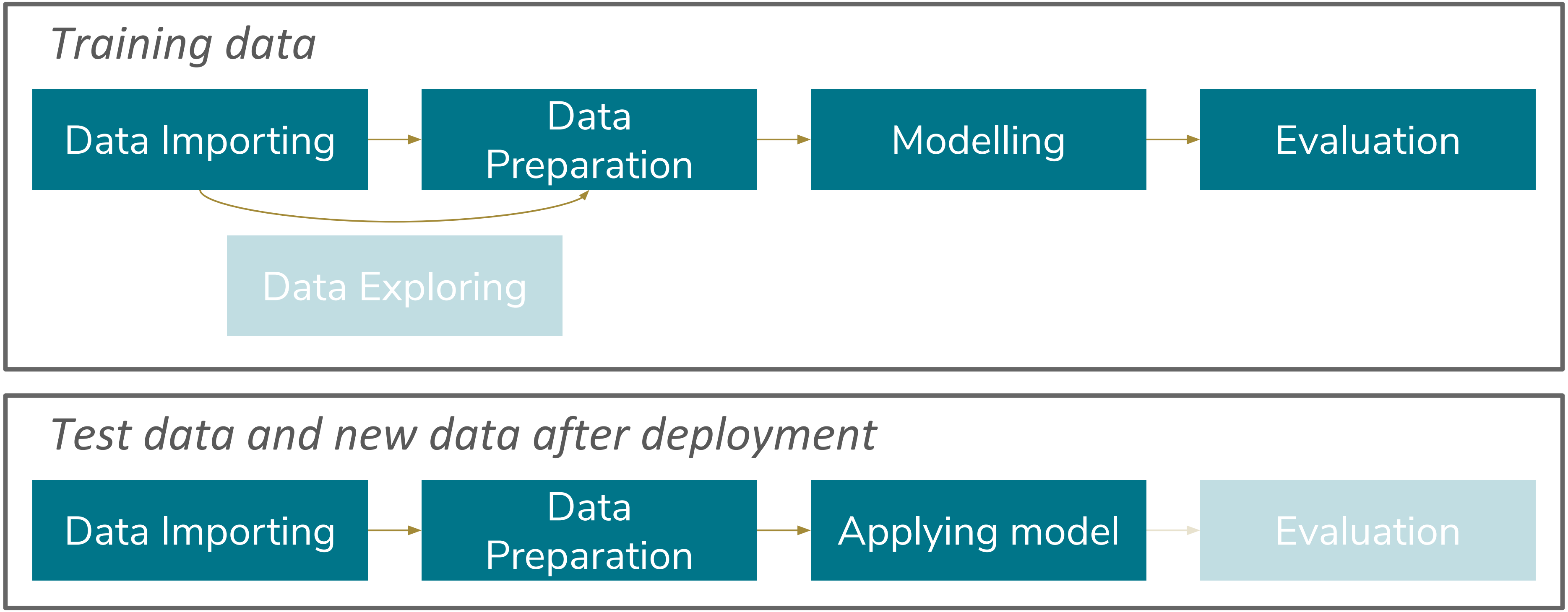
Learning objectives for today

- Know best practices in cleaning and working with data
- Understand different types of data processing technologies and patterns
- How to transform raw data into:
  - technically correct,
  - real-world consistent, and
  - ‘prediction-appropriate’ data
- The importance of feature engineering
- Basic working knowledge of SQL for data preparation



Partly based on: [*An introduction to data cleaning with R*]

Data analysis pipeline



Starting point

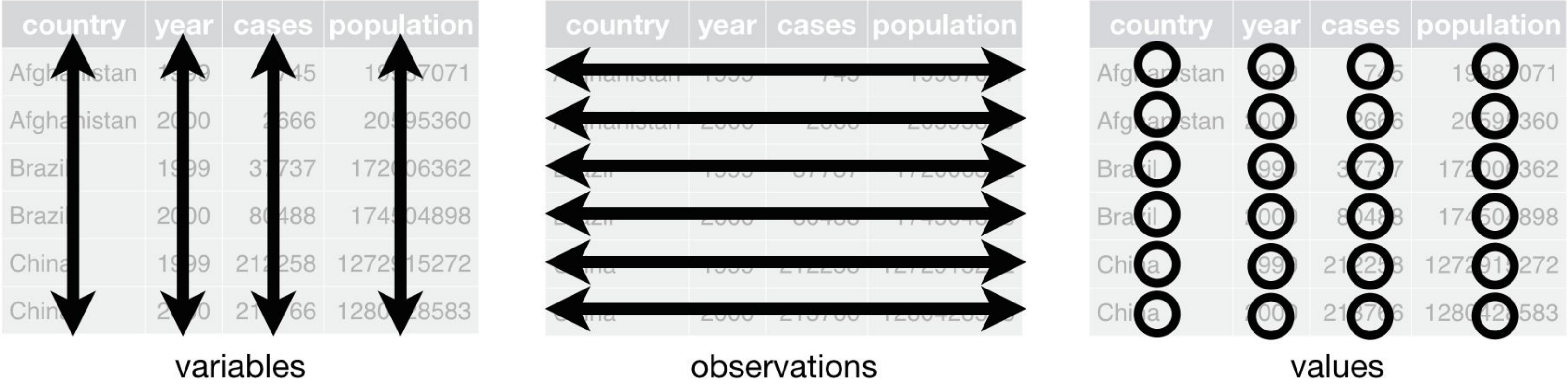
1. Start with the data in exactly the same shape as a newly collected dataset would have. Check whether your dataset does not already contain some non-automated pre-processing steps:
   - Selections
   - Aggregations
   - Outlier handling etc.

2. Always work from a script (SQL, R, Python)
   - never make manual adjustments to the data

Data prep steps

1. Importing and possibly combining datasets
2. Making data technically correct
   - Correct headers
   - Correct data classes (integer, string, factor, date, etc)
3. Making data real-world consistent
   - Detecting inconsistencies or errors
   - Selecting the cause of the inconsistency or error
   - Correcting the inconsistency or error
4. Making data ‘prediction-appropriate’
   - Excluding variables
   - Defining Y and redefining X-variables
   - Handling correlated predictors
   - Adding domain knowledge

Work with tidy (structured) data



- Each dataset much have its own **table** (dataframe)
- Each variable must have its own **column** (features)
- Each observation must have its own **row** (records, unit of analysis, grain)
- Each value must have its own **cell**

https://r4ds.had.co.nz/tidy-data.html

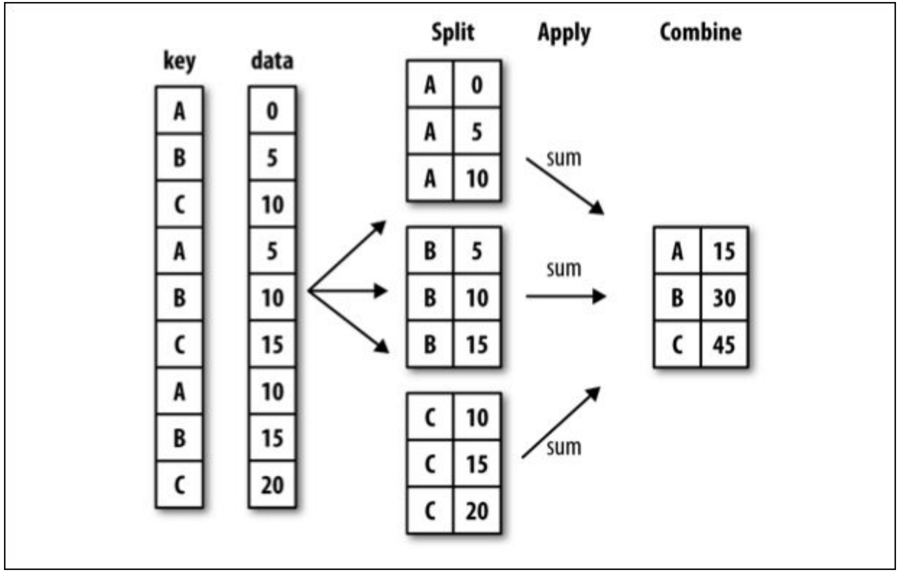
Pivoting: from wide to long format (and back)

- Use **long format** throughout your data processing pipeline
- Use **wide format** for presenting and communicating results
- Daily frustration: sources in wide format

https://r4ds.had.co.nz/tidy-data.html

Split-apply-combine strategy for data processing

1. Extract a subset of the data for which it is easy to solve the problem.
2. Solve the problem by hand, checking results as you go.
3. Write a function that encapsulates the solution.
4. Use the appropriate plyr function to split up the original data, apply the function to each piece and join the pieces back together.

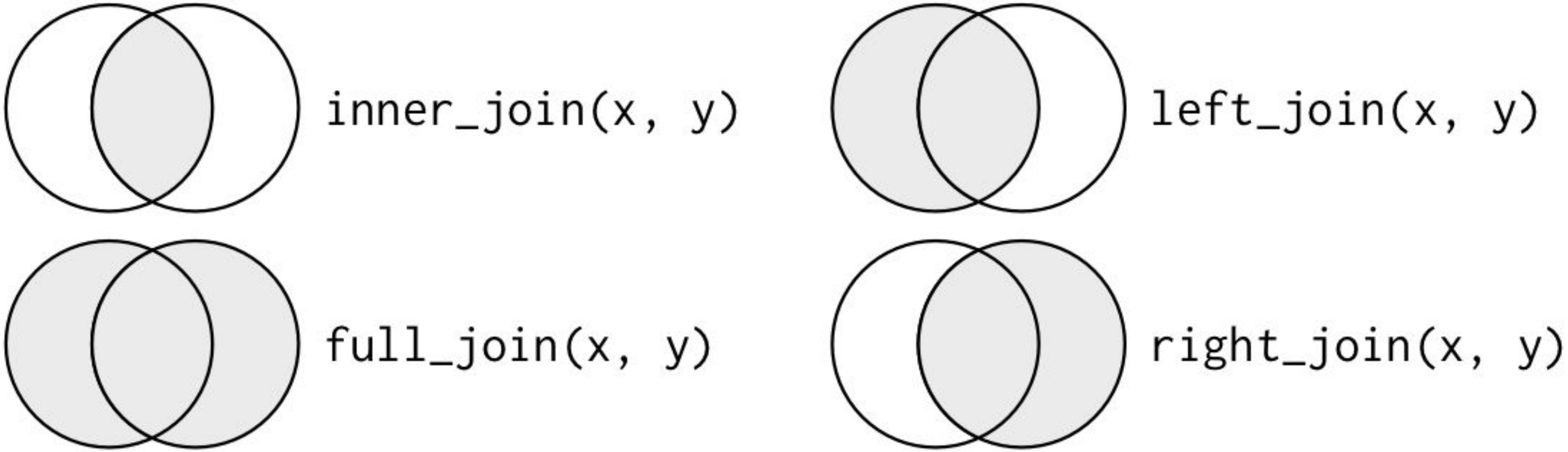


Hadley Wickham, the Split-Apply-Combine Strategy for Data Analysis

Same data processing concepts, different tools

concept	Spreadsheet	SQL	R tidyverse	python pandas
Sorting	SORT()	ORDER BY	arrange()	sort_values()
Filtering	FILTER()	WHERE	filter()	filter()
Split	n/a	GROUP BY	group_by()	group_by
Apply	many functions	many functions	many functions	many functions
Combine	VLOOKUP	JOIN	join(), inner_join(), left_join etc.	join(), concat(), merge() etc.

Understanding joins



https://r4ds.had.co.nz/relational-data.html#mutating-joins

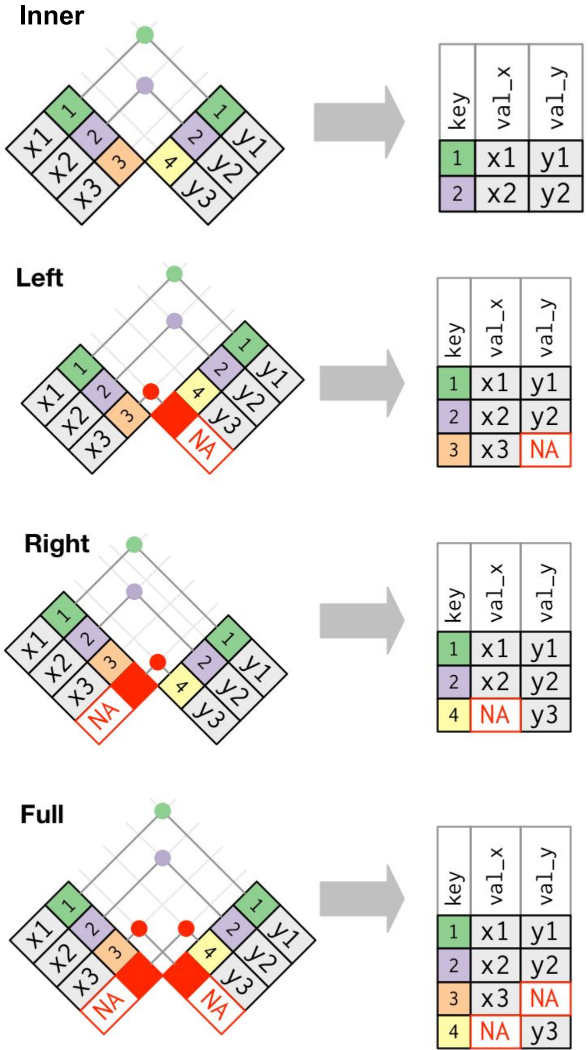
Understanding joins

x

1	x1
2	x2
3	x3

y

1	y1
2	y2
4	y3



https://r4ds.had.co.nz/relational-data.html#mutating-joins

Making data ‘prediction-appropriate’

1. Select usable variables
2. Defining the outcome Y
3. Redefining the X-variables
4. Handling correlated predictors
5. Adding domain knowledge

1. Select usable variables

Delete variables that:

- will not be known at the moment of modelling
- are (practically) constant
- are (practically) unique (unless they can be clustered meaningfully)

2. Defining the outcome Y

This is a crucial step in your project. Not working with the 'right' definition of Y is likely to result in either,

- a project that is of little use (as the outcome is not what users want), or
- additional project iterations (going back in a later stage to adjust the definition because model outcomes are not satisfactory), or
- never achieving good enough performance due to Y being of too low quality / too noisy

1. Be as sure as you can be regarding the definition and quality of Y
2. Define the outcome Y, and
3. Describe / visualize the newly defined Y

3. Re-defining the X-variables

It is not trivial how to approach this. Take into account that:

- Some machine learning algorithms automatically look for relevant interactions / non-linear relationships
- Some machine learning algorithms don’t, and will only consider such relationships when explicitly being told to do so
- Even machine learning algorithms that do automatically look for interactions / non-linear relationships, could be helped when not having to discover those themselves

→ Finding out whether adjusting the input variables boosts model performance is a process of trial and error (model selection)

4. Handling correlated predictors

Highly correlated variables can destabilize the process of model building and tend to decrease model performance (bias-variance trade-off)

In healthcare or psychology, this typically occurs with questionnaire items

If a cluster of correlated features contains relevant information, explore different options for handling correlation:

- Principal Component Analysis (data-driven)
- Use sub scales (at least partly based on domain expertise)
- Use sub scales plus a few individual relevant items (at least partly based on domain expertise)
- Automatic feature selection using LASSO or Random Forest (data-driven)

5. Adding domain knowledge

Principles:

- Adding domain knowledge to your dataset is one of the most important aspects in a traditional machine learning project
- Your raw data are practically never structured / defined in such a way that they contain the most value to the problem you want to solve
- A simple model based on relevant data will perform much better than the most sophisticated model based on the raw data
- Feature engineering is partly data science and partly dark art
- Feature engineering depends largely on the availability of domain experts

Feature engineering

What variables can we create out of the variables that we have in order to increase the relevant information for modelling the outcome Y?

=> *Learning by example(s)*

Feature engineering - examples

- Converting absolute values into relative changes within the unit of analysis (e.g. change in a patient’s medication use)
- Adding information about a higher unit of analysis (adding information about a clinic or therapist to a patient record)
- Categorizing numeric variables based on relevant cut-off values (e.g. converting Beck’s Depression Inventory scores into mild, moderate and severe depression, or converting #minutes of daily movement into less than versus at least 30 minutes per day, changing ‘last year’s maternity care cost’ into a yes/no dummy, or dividing age into <18, 18-45, 46-65, and 65+)

Feature engineering - examples

- Aggregating questionnaire items into relevant subscales and/or selecting specific items (e.g. converting the 45 items of the Outcome Questionnaire into subscales on Symptom Distress, Interpersonal relations and Social Role, in addition to selecting separate 'risk-items' on suicidality and aggression)
- Aggregating factor variables to decrease the number of different categories
- Describing 'agenda data' (e.g. scheduled appointments) into meaningful features (e.g. treatment intensity, continuity, etc.)
- Combining different inputs into one meaningful variable (e.g. construct 'treatment intensity' out of number of sessions and time in treatment)