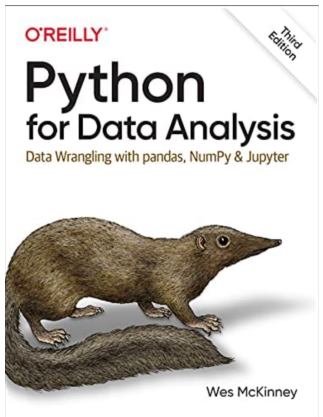
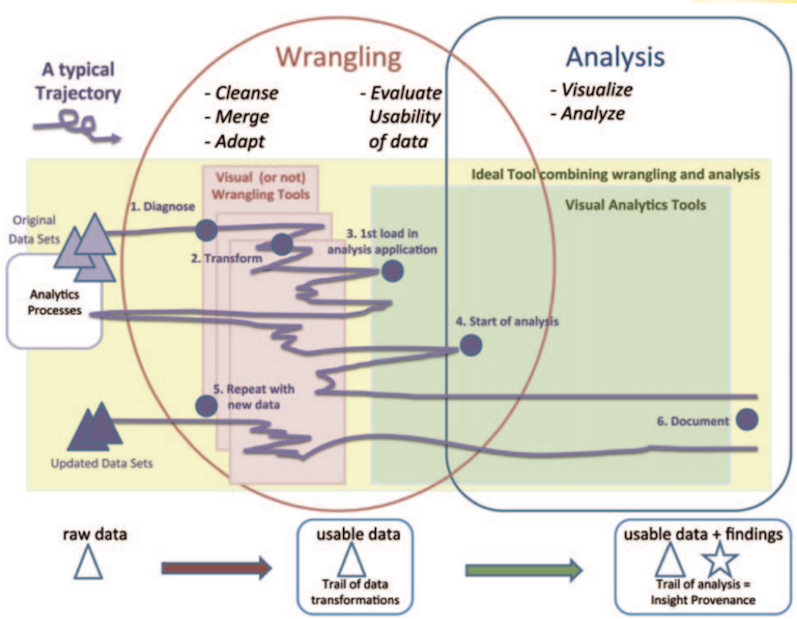
7.2: Data Wrangling and Cleaning

- **Instructor**: Dr. GP Saggese, gsaggese@umd.edu
- **Reference**
  - Pandas tutorial
  - Web
    - Many free resources (e.g., official docs)
  - Mastery
    - https://wesmckinney.com/book
    - Read cover-to-cover and try the examples 2–3 times to really *master* them



Data Wrangling and Data Cleaning

- **Data wrangling and cleaning**
  - Turn *messy, inconsistent, incomplete* data into a *coherent* dataset
- **Most effort** in data science is spent here, not on modeling
  - It is iterative work
  - Clean → Model → Clean → ...



- **Data wrangling**
  - Transform data into a useful structure
  - Reshape tables, merge sources, create new variables
- **Data cleaning**
  - Correct or remove incorrect, corrupt, or irrelevant data
  - Handle missing values, outliers, inconsistent formats
- **Often done together**

Typical Data Wrangling Workflow

- **Inspect data**
  - Look at column names, data types
  - Look at a few rows
  - Basic summaries
- **Diagnose problems**
  - Missing values, unusual codes, inconsistent units, strange distributions
- **Decide strategy**, based on project
  - What to keep
  - What to fix
  - What to drop

- **Apply transformations**
  - Filter, join, reshape, recode, aggregate
- **Validate results**
  - Check that changes make sense and that key statistics are reasonable

Data Extraction

- **Web scraping**
  - Use APIs or scrape data explicitly
  - Scraping is challenging
    - Fragile (page changes can break your parser)
    - Throttling/rate limits
    - Cat-and-mouse game with websites
  - Set up pipelines for periodic scraping
  - Tools for automated scraping
    - E.g., `import.io`, `portia`, ..
- **Various sources of data**
  - Files (e.g., CSV, JSON, XML)
  - Databases
  - Spreadsheets
  - AWS S3 buckets
  - Web APIs
  - Logs
  - ...
- Raw data reflects **collection needs, not analysis needs**

Tidy Data

- **Definition**: Tidy data is a structured format for datasets that makes them easier to manipulate, analyze, and visualize
  - Tidy data, Wickham, 2014
- **Core Principles**
  - Each variable forms a *column*
  - Each observation forms a *row*
  - Each type of observational unit forms a *table*
  - Key tools in pandas: `pandas.melt()`, `pandas.pivot()`

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

	John Smith	Jane Doe	Mary Johnson
treatmenta	—	16	3
treatmentb	2	11	1

"Messy" data

name	trt	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

Tidy data

Reshaping data

- **Wide to long**
  - Turn similar columns into key-value pairs
- **Long to wide**
  - Turn key-value pairs into separate columns
- **What is the best shape?**
  - Choose the shape that simplifies analysis and visualization

type	date	clicks	conversions	impressions
0	2020-01-01	1.0	NaN	18.0
1	2020-01-02	2.0	NaN	19.0
2	2020-01-03	1.0	1.0	14.0
3	2020-01-04	NaN	NaN	5.0
4	2020-01-05	1.0	NaN	8.0
5	2020-01-06	1.0	1.0	15.0
6	2020-01-07	2.0	NaN	8.0

Wide format

	date	type	count
0	2020-01-01	impressions	18.0
1	2020-01-02	impressions	19.0
2	2020-01-03	impressions	14.0
3	2020-01-04	impressions	5.0
4	2020-01-05	impressions	8.0
...	...	...	...
91	2020-01-28	conversions	NaN
92	2020-01-29	conversions	NaN
93	2020-01-30	conversions	NaN
94	2020-01-31	conversions	NaN
95	2020-02-01	conversions	NaN

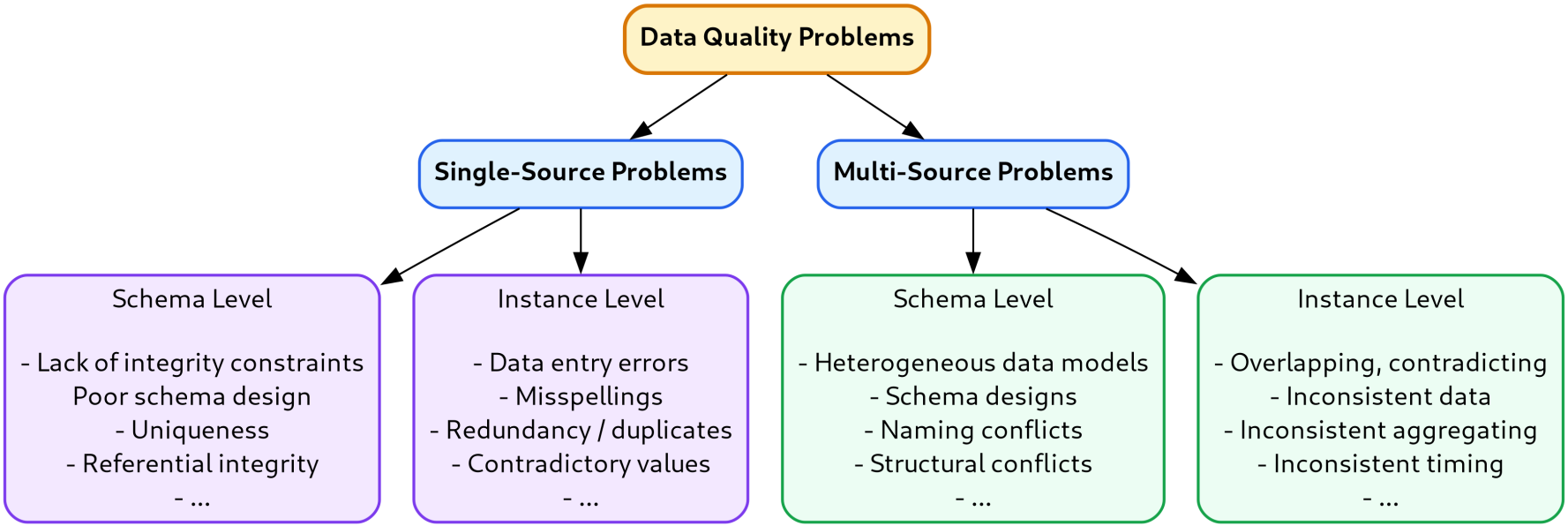
Long format

Split-Apply-Combine

- **Common data analysis pattern**
  - *Split*: break data into smaller pieces
  - *Apply*: operate on each piece independently
  - *Combine*: combine pieces back together
  - The Split-Apply-Combine Strategy for Data Analysis, Wickam, 2011
- **Examples**
  - Group-wise ranking
  - Group statistics (sums, means, counts)
  - Create separate models per group
- **Pros**
  - Code is compact
  - Easy to parallelize
- **Supported by many systems**
  - Pandas
  - `SQL GROUP BY` operator
  - Map-Reduce

```
In [94]: animals.groupby("kind").height.agg(
   ....:     min_height="min",
   ....:     max_height="max",
   ....: )
   ....:
Out[94]:
      min_height  max_height
kind
cat          9.1         9.5
dog          6.0        34.0
```

Classification of Data Problems



- Several problems **reduce trust and usability of data**
  - *Single-Source*: Occur within a single dataset
  - *Multi-Source*: Arise when integrating data from multiple sources
  - *Schema-Level*: Related to the structure or design of data
  - *Instance-Level*: Related to actual data values stored

Single-Source Problems

- **Depends largely on source**
  - *Databases* often enforce schema / instance constraints
  - *Spreadsheet data* is often "clean"
    - Schema exists
  - *Logs* are messy
  - *Web-page data* is messier
- **Types of problems**
  - Ill-formatted data
  - Missing/illegal values, misspellings, wrong fields, extraction issues
  - Duplicated records, contradicting information, referential integrity violations
  - Unclear default/missing values
  - Evolving schemas/classification schemes (categorical attributes)
  - Outliers

Single-Source Problems: Example

- **Sources of errors in data**
  - Data entry errors: arbitrary values entered
  - Measurement errors: sensor data inaccuracies
  - Distillation errors: processing and summarization issues
  - Data integration errors: inconsistencies across combined sources

Scope/Problem		Dirty Data	Reasons/Remarks
Attribute	Missing values	phone=9999-999999	unavailable values during data entry (dummy values or null)
	Misspellings	city="Liipzig"	usually typos, phonetic errors
	Cryptic values, Abbreviations	experience="B"; occupation="DB Prog."	
	Embedded values	name="J. Smith 12.02.70 New York"	multiple values entered in one attribute (e.g. in a free-form field)
	Misfielded values	city="Germany"	
Record	Violated attribute dependencies	city="Redmond", zip=77777	city and zip code should correspond
Record type	Word transpositions	$name_1$= "J. Smith", $name_2$="Miller P."	usually in a free-form field
	Duplicated records	emp_1=(name="John Smith",...); emp_2=(name="J. Smith",...)	same employee represented twice due to some data entry errors
	Contradicting records	emp_1=(name="John Smith", bdate=12.02.70); emp_2=(name="John Smith", bdate=12.12.70)	the same real world entity is described by different values
Source	Wrong references	emp=(name="John Smith", deptno=17)	referenced department (17) is defined but wrong

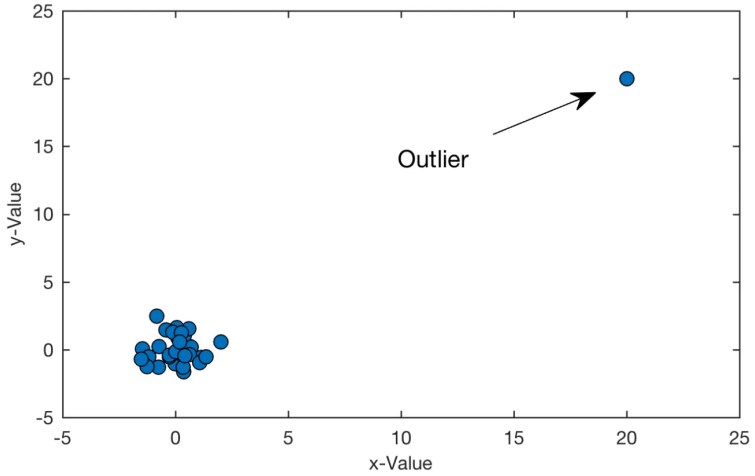
Table 2. Examples for single-source problems at instance level

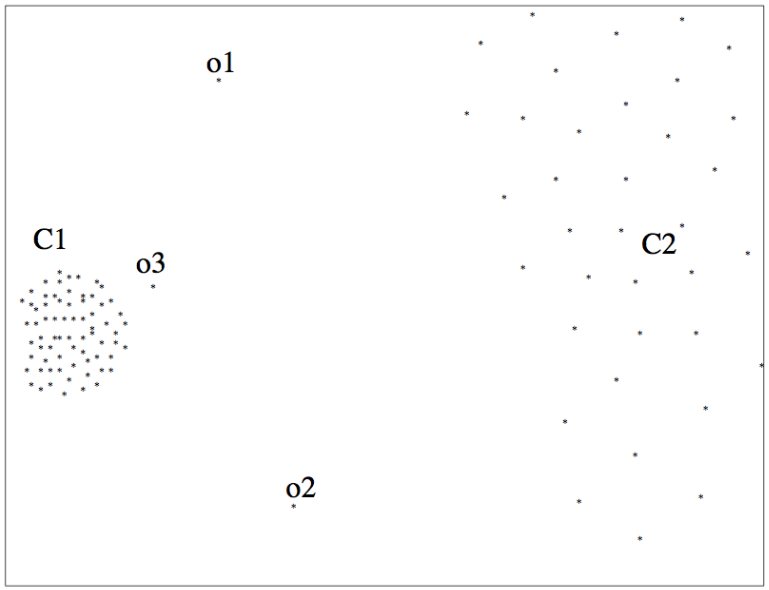
Multi-Source Problems

- **Different data sources**
  - Developed separately
  - Maintained by different people
  - Stored in different systems
- **Schema mapping / transformation**
  - Map information across sources
  - Naming conflicts
    - Same name for different objects
    - Different names for same objects
  - Different representations across sources
- **Entity resolution**
  - Match entities across sources
- **Data quality issues**
  - Contradicting information
  - Mismatched information

Univariate Outlier Detection

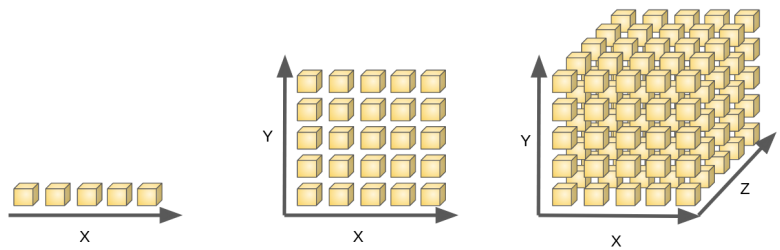
- **Problems with outliers**
  - Extreme outliers may alter metrics, masking others
- Use statistics to **identify outliers**
  - Robust statistics: minimize effect of corrupted data
  - Robust center metrics
    - Median
    - k%-trimmed mean (discard lowest and highest k% values)
  - Robust dispersion
    - Median absolute deviation (MAD)
    - Median distance from median value





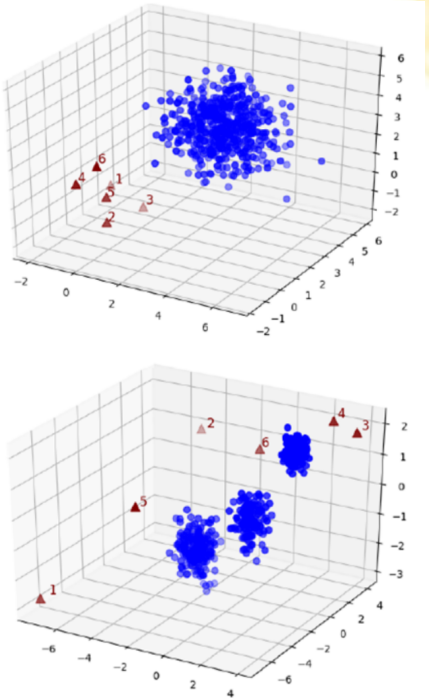
Outlier Detection

- **For Gaussian data**
  - Data points 1.5x MAD from median
  - Plot histogram to confirm
- **For non-Gaussian data**
  - Estimate generating distribution (parametric or not)
  - Distance-based: find points with few neighbors
  - Density-based:
    - Define *density* as the average distance to k nearest neighbors
    - Use relative density to identify outliers
- **Curse of dimensionality**
  - *"In high-dimensional spaces, data is sparse"*
  - Techniques break down as data dimensionality increases
  - Need $O(e^n)$ points with n dimensions to estimate
  - Project data into lower-dimensional space to find outliers
    - Not straightforward



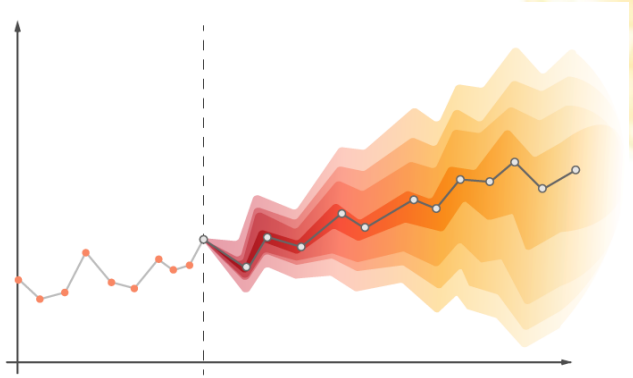
Multivariate Outliers

- **Mean/covariance not robust**
  - Sensitive to outliers
  - Use robust statistics
- Assume **multivariate Gaussian data**
  - Defined by *mean* $\underline{\mu}$ and *covariance matrix* $\underline{\underline{\Sigma}}$
  - *Iterative approach*
    - Mahalanobis distance: $\sqrt{(\underline{x}-\underline{\mu})^T\underline{\underline{\Sigma}}^{-1}(\underline{x}-\underline{\mu})}$
    - Measures distance of point x from distribution
    - Identify outliers: points too far by Mahalanobis distance
    - Remove outliers
    - Recompute mean and covariance
- Data volume often large
  - Use approximation techniques
- Try different techniques based on data



Time Series Outliers

- Often data is in the form of a **time series**
- A **time series** is a sequence of data points recorded at regular intervals
  - Stock prices
  - Sales revenue
  - Website traffic
  - Inventory levels
  - Energy consumption
  - Market demand
  - Social media engagement
  - Hourly energy usage
  - Weekly retail foot traffic
  - ...



- Rich literature on **forecasting** in time series data
- Use historical patterns to flag **outliers**
  - Rolling MAD (median absolute variation)