UMD DATA605 - Big Data Systems

Orchestration with Airflow Data wrangling Deployment

Instructor: Dr. GP Saggese - gsaggese@umd.edu**

v1.1

UMD DATA605 - Big Data Systems Orchestration with Airflow **Data wrangling (Pandas)** Deployment

Dr. GP Saggese gsaggese@umd.edu

Resources

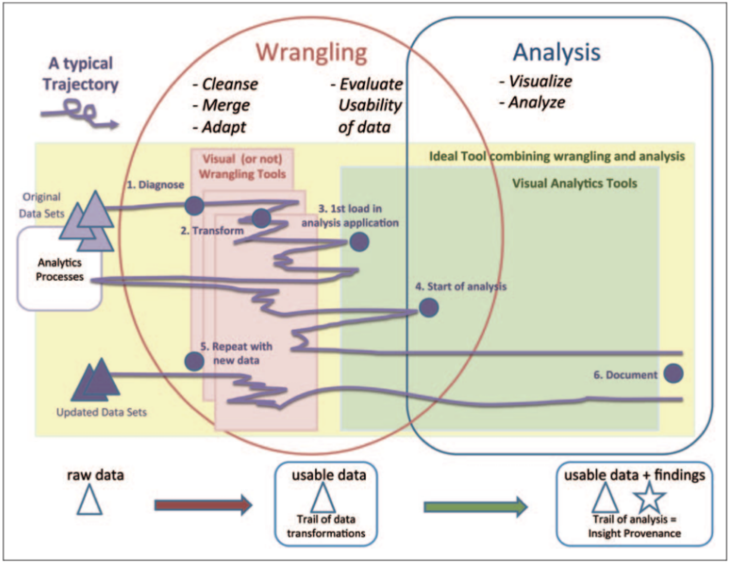
- Pandas tutorial
- Class project
- Web
  - https://pandas.pydata.org
  - Onslaught of free resources
- Mastery
  - https://wesmckinney.com/book
  - Read cover-to-cover and execute all examples 2-3x time to really *master*

Overview

- **Data wrangling**
  - Aka “data preparation”, “data munging”, “data curation”
  - Get data into a structured form suitable for analysis
  - Often it is the step where majority of time (80-90%) is spent
- **Key steps**
  - **Scraping**: extract information from sources (e.g., webpages)
  - **Data cleaning**: remove inconsistencies / errors
  - **Data transformation**: get data into the right structure
  - **Data integration**: combine data from multiple sources
  - **Information extraction**: extract structured information from unstructured / text sources

SCIENCE ACADEMY

Overview



igure 1. The iterative process of wrangling and analysis. One or more initial data sets may be used and new versions nay come later. The wrangling and analysis phases overlap. While wrangling tools tend to be separated from the visual inalysis tools, the ideal system would provide integrated tools (light yellow). The purple line illustrates a typical iterative rocess with multiple back and forth steps. Much wrangling may need to take place before the data can be loaded within

Overview

- Many of the data wrangling problems are not easy to formalize, and have seen little research work, e.g.,
  - Data cleaning: mainly statistics, outlier detection, imputation
  - Data transformation, i.e., put the data in the “right” structure (e.g., tidy data)
  - Information extraction: feature computation, highly domain specific
- Others aspects have been studied in depth, e.g.,
  - Schema mapping
  - Data integration
- In an ETL process
  - Data extraction is the E step
  - Data wrangling is the T step

Overview

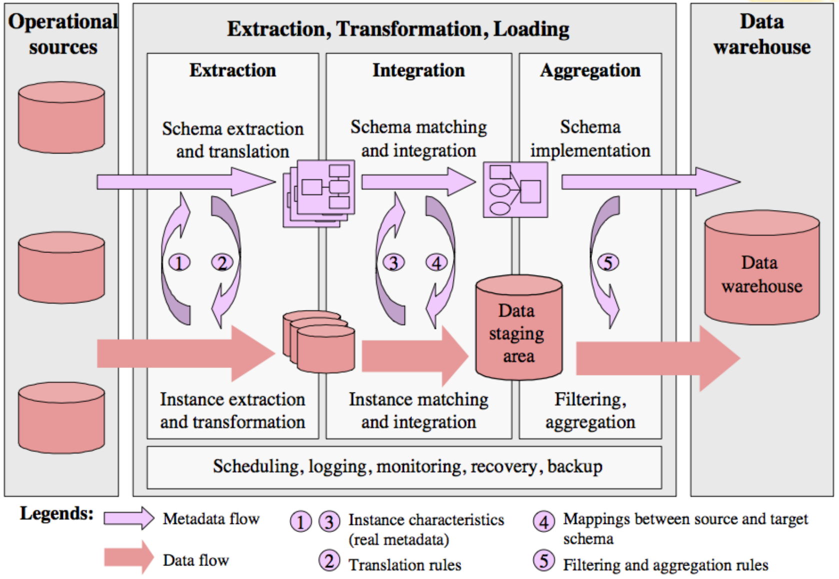

Operational sources
Extraction, Transformation, Loading
Extraction
Integration
Aggregation
Data warehouse
Schema extraction and translation
Schema matching and integration
Schema implementation
1
2
3
4
5
Data staging area
Data warehouse
Instance extraction and transformation
Instance matching and integration
Filtering, aggregation
Scheduling, logging, monitoring, recovery, backup
Legends:
Metadata flow
Data flow
1 3 Instance characteristics (real metadata)
2 Translation rules
4 Mappings between source and target schema
5 Filtering and aggregation rules

Figure 1. Steps of building a data warehouse: the ETL process

Data Extraction

- Data may reside in a wide variety of different sources
  - Files (e.g., CSV, JSON, XML)
  - Many databases
  - Spreadsheets
  - AWS S3 buckets
  - ...
  - Most analytical tools support importing data from such sources through adapters
- Web scraping
  - In some cases there may be APIs, in other cases data may have to be explicitly scraped
  - Scraping data from web sources is tough
    - Can be fragile
    - Throttling
    - It's cat-and-mouse game between scrapers and website
  - Often pipelines are set up to do this on a periodic basis
  - Several tools out there to do this (somewhat) automatically
    - E.g., import.io, portia, ...

Tidy Data

- Tidy data, Wickham, 2014
  - Each variable forms a column
  - Each observation forms a row
- Wide vs long format

	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

Data Quality Problems

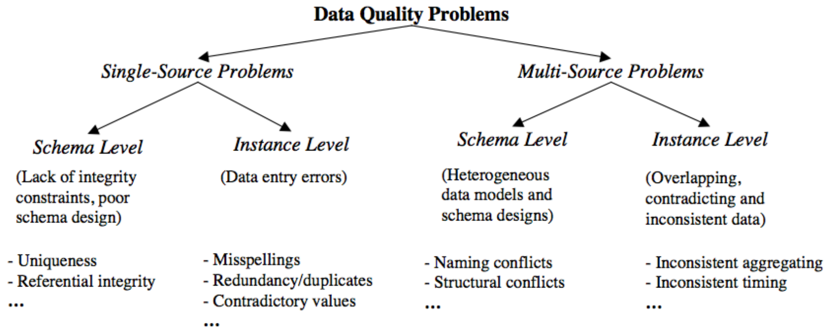


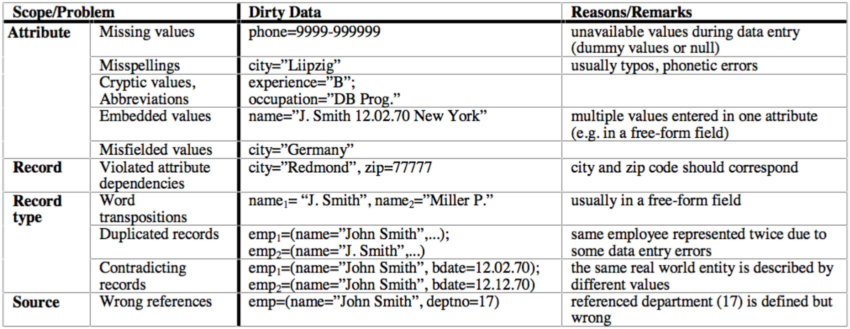
Figure 2. Classification of data quality problems in data sources

Single-Source Problems

- Depends largely on the source
- Databases can enforce constraints
- Data extracted from spreadsheets is often “clean”
  - At least there is a schema
- Logs are messy
- Data scraped from web-pages is much more messy
- Types of problems:
  - Ill-formatted data
  - Missing or illegal values, misspellings, use of wrong fields, extraction issues (e.g., not easy to separate out different fields)
  - Duplicated records, contradicting information, referential integrity violations
  - Unclear default/missing values
  - Evolving schemas or classification schemes (for categorical attributes)
  - Outliers

Data Quality Problems

Data Quality Problems



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

SCIENCE ACADEMY

# Multi-Source Problems

## Multi-Source Problems

- Different data sources are:
  - Developed separately
  - Maintained by different people
  - Stored in different systems
  - ...
- Schema mapping / transformation
  - Mapping information across sources
  - Naming conflicts: same name used for different objects, different names for same objects
  - Structural conflicts: different representations across sources
- Entity resolution
  - Matching entities across sources
- Data quality issues
  - Contradicting information
  - Mismatched information
  - ...

## Data Cleaning: Outlier Detection

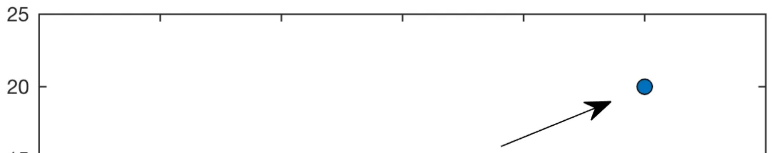
## Data Cleaning: Outlier Detection

- Quantitative Data Cleaning for Large Databases, Hellerstein, 2008
  - Focuses on numerical data (i.e., integers/floats that measure some quantities of interest)
- Sources of errors in data
  - Data entry errors: users putting in arbitrary values to satisfy the form
  - Measurement errors: especially sensor data
  - Distillation errors: errors that pop up during processing and summarization
  - Data integration errors: inconsistencies across sources that are combined together

## Univariate Outlier Detection

## Univariate Outlier Detection

- A set of values can be characterized by metrics such as
  - Center (e.g., mean)
  - Dispersion (e.g., standard deviation)
  - Higher momenta (e.g., skew, kurtosis)
- Use statistics to identify outliers
  - Must watch out for “masking”: one extreme outlier may alter the metrics sufficiently to mask other outliers
  - Robust statistics: minimize effect of corrupted data
  - Robust center metrics:
    - Median
    - k%-trimmed mean (i.e., discard lowest and highest k% values)
  - Robust dispersion:
    - Median absolute deviation (MAD)
    - Median distance of values from the median value



# Outlier Detection

## Outlier Detection

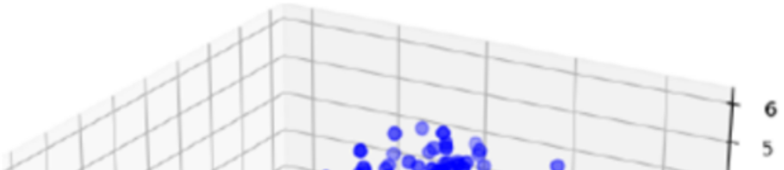
- For Gaussian data
  - Any data points 1.4826x MAD away from median
  - May need to eyeball the data (e.g., plot a histogram) to decide if this is true
- For non-Gaussian data
  - Estimate generating distribution (parametric approach)
  - Distance-based methods: look for data points that do not have many neighbors
  - Density-based methods:
    - Define *density* to be average distance to *k* nearest neighbors
    - *Relative density* = density of node/average density of its neighbors
    - Use relative density to decide if a node is an outlier
- Most of these techniques start breaking down as the dimensionality of the data increases
  - *Curse of dimensionality*
    - You need an O(eˆn) points with n dimensions to estimate
    - "In high dimensional spaces, data is always is sparse"
  - Can project data into lower-dimensional space and look for outliers there
    - Not as straightforward
- Wikipedia article on Outliers

SCIENCE ACADEMY

## Multivariate Outliers

## Multivariate Outliers

- One set of techniques *multivariate Gaussian distribution* data
  - Defined by a *mean* μ and a *covariance matrix* Σ
- Mean / covariance are not *robust* (sensitive to outliers)
- Robust statistics analogous to univariate case
- Iterative approach
  - Mahalanobis distance of a point is the square root of $(x-\mu)'\Sigma^{-1}(x-\mu)$
  - Measures how far the point x is from a multivariate normal distribution
  - Outliers are points that are too far away according to Mahalanobis distance
  - Remove outlier points
  - Recompute the mean and covariance
- Often volume of data is too much
  - Approximation techniques often used
- Need to try different techniques based on the data



## Time Series Outliers

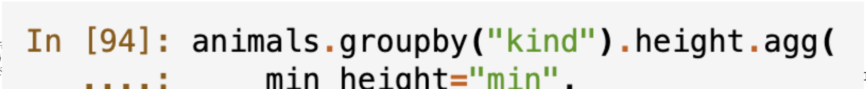
## Time Series Outliers

- Often data is in the form of a time series
- A **time series** is a sequence of data points recorded at regular time intervals tracking a variable over time
  - Stock prices
  - Sales revenue
  - Website traffic
  - Inventory levels
  - Energy consumption
  - Market demand
  - Social media engagement
  - Hourly energy usage
  - Customer satisfaction ratings over time
  - Weekly retail foot traffic
  - ...
- Rich literature on *forecasting* in time series data
- Can use the historical patterns in the data to flag outliers
  - Rolling MAD (median absolute variation)

## Split-Apply-Combine

## Split-Apply-Combine

- The Split-Apply-Combine Strategy for Data Analysis, Wickam, 2011
- Common data analysis pattern
  - Split: break data into smaller pieces
  - Apply: operate on each piece independently
  - Combine: combine the pieces back together
- Pros
  - Code is compact
  - Easy to parallelize
- E.g.,
  - group-wise ranking
  - group vars (e.g., sums, means, counts)
  - create new models per group
- Supported by many languages
  - Pandas
  - SQL GROUP BY operator
  - Map-Reduce



```
In [94]: animals.groupby("kind").height.agg(
   ....:         min_height="min",
```