

COMP 499 Introduction to Data Analytics

Lecture 7 — Review of Lectures 1 – 6

Greg Butler

Data Science Research Centre

and

Centre for Structural and Functional Genomics

and

Computer Science and Software Engineering

Concordia University, Montreal, Canada

`gregb@cs.concordia.ca`

Review

What is important in the course?

Concepts, terminology, tool, techniques, skills

What did you learn in the labs?

What is the aim of the course?

What is the aim of data analytics?

Processes

Data Analytics

Data Wrangling

Data Cleaning

Exploratory Data Analytics

Descriptive Data Analysis

Predictive Data Analysis (not yet)

Prescriptive Data Analysis (not yet)

Data Analysis

Descriptive Data Analysis

is describing your data from past activities

Predictive Data Analysis

provides results for unseen data for future activities

Prescriptive Data Analysis

models viable solutions to a problem
and the impact of considering a solution

Descriptive vs Predictive vs Prescriptive

Descriptive Analytics, which use data aggregation and data mining to provide insight into the past and answer: *"What has happened?"*

Predictive Analytics, which use statistical models and forecasts techniques to understand the future and answer: *"What could happen?"*

Prescriptive Analytics, which use optimization and simulation algorithms to advice on possible outcomes and answer: *"What should we do?"*

<https://halobi.com/blog/descriptive-predictive-and-prescriptive-analytics-explained/>

Data Analytics Process

Steps in Data Analytics

- ▶ Setting Questions
- ▶ Data Wrangling
- ▶ Exploratory Data Analysis
- ▶ Modeling
- ▶ Story telling

Iterative process

The process is rarely linear.

Each step can push data scientist to revisit methods, techniques
... or reconsider whether the original question was the right one?
And the final answer simply sparks more questions!

Data Wrangling Process

Steps in Data Wrangling

- ▶ Discover
- ▶ Structure
- ▶ Cleanse
- ▶ Enrich
- ▶ Validate
- ▶ Publish

Process: Data Cleaning

Steps in Data Cleaning

- ▶ Data Cleaning
- ▶ Data Compatibility
- ▶ Missing Data
- ▶ Outliers
- ▶ Remove Duplicates
- ▶ Consistency of Data

Process: Exploratory Data Analysis

Exploratory Data Analysis

Learn about the properties of the data

Steps for Exploratory Data Analysis

- ▶ Descriptive statistics: mean/median and variance, quantiles, outliers
- ▶ Correlation
- ▶ Fitting curves and distributions
- ▶ Dimension reduction
- ▶ Clustering

Tools

Jupyter

Python

pandas

OpenRefine (some)

Data

Data scales (Measurement scales)

Accuracy, precision, significant digits

Levels of interpretation

Normalization

Metadata

Self-descriptive data

Common data formats

Tidy data schema (from R) for pandas

Statistics

population, sample
central tendency, variation
robust statistics
descriptive statistics/analytics

Scientific Method

Similar but different to Data Analytics

Planned — data generation, and data analysis

Hypothesis-driven

Data Warehouses and Business Intelligence

Operational/transactional databases

Data warehouse

Data Mart

Data Warehouse does not use 3NF

Data cube

Extract-Transform-Load

OLAP vs OLTP

Lecture 1 Big Data

1. Big Data

- ▶ Actionable Data
- ▶ History
- ▶ Five V's
- ▶ Types of Jobs
- ▶ Privacy and Security

Big Data (<http://dsrc.encs.concordia.ca/what-is-bigdata.html>)

Big Data

Definition of “*Big*” has changed as we have become more advanced

History

Hollerith Cards 1890 (US population census)

Economic Data 1952 (GDP etc)

Computers 1959 — The First Digital Data Tsunami

World Wide Web 1990's — The Second Digital Data Tsunami

Social Media 1985 — The Third Digital Data Tsunami

Internet of Things 2000 — The Fourth Digital Data Tsunami

Big Science — 1960's onwards

Deep Knowledge — 2011 onwards

A key notion is **actionable data** that is useful in supporting decisions, determining actions, and adding value to an endeavour.

Big Data

The 5 V's

Volume: amount of data

Variety: different types of data

Velocity: rate at which data is generated

Veracity: trustworthiness, level of noise

Value: usefulness of data to a business

Drivers

Transactions

Mobile

Social Media

Internet of Things

MGI Report

McKinsey Global Institute, *Big data: The next frontier for innovation, competition, and productivity*, May 2011.

Lecture 1 Data Analytics and Context

1. Context

- ▶ Data to Application Data
- ▶ Data Levels of Interpretation
- ▶ Lexical to Pragmatics
- ▶ Approaches to Data Analytics

Context — Data to Knowledge

Data

raw, calibrated, normalized, validated

derived, aggregated, interpreted

Metadata describes source and properties of the data

Information

newsworthy

actionable

Claude Shannon's information theory

Knowledge

applicable wisdom, organized information

concepts, relations, constraints, taxonomy/ontology

axioms, rules, plans

Application — aka Knowledge Translation

Context — Data Level of Interpretation

Raw Data

raw values obtained directly from the measurement device

Calibrated Data

raw physical values, corrected with calibration operators

Validated Data

calibrated data that has been filtered through quality assurance procedures

(most commonly used data for scientific purposes)

Derived Data

frequently aggregated data, such as gridded or averaged data

Interpreted Data

derived data that is related to other data sets, or to the literature of the field

Context — Syntax to Pragmatics

Lexical = atomic units

Defined by regular expressions

Represented as enum's

Syntax = structure

Defined by grammars

Represented as Abstract Syntax Trees (AST)

Semantics = meaning

Defined by interpretation mappings

Represented as actions (procedural) in compiling

Pragmatics = goals

Context — Approaches to Data Analysis

Scripting

Unix tools, eg
text files, csv files for inputs, outputs, intermediate steps
stepwise development of analysis
script captures steps, parameters
easy to replay

Notebooks

Jupyter, eg
interactive scripting with “literate programming”
keep track of thought processes during analysis
work with files to replay analysis

“Spreadsheet” Environments

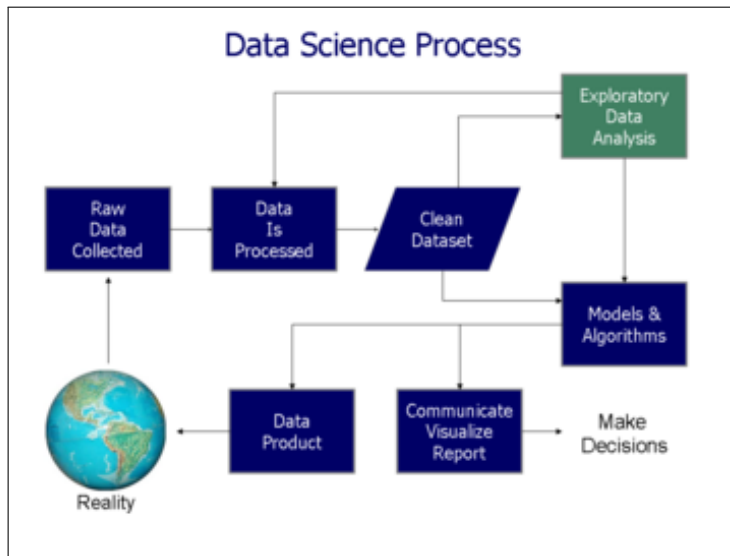
OpenRefine, eg
lots of tools, little guidance
need macros, histories, to capture/replay work
often proprietary

Lecture 1 Data Analytics

1. Data Analytics

- ▶ Data Wrangling
- ▶ Exploratory Data Analysis
- ▶ Modeling
- ▶ Story telling

Data Analytics



Data Analytics — Example from wikipedia

Find the variables which best predict the tip given to the waiter.

The variables available in the data collected:

- ▶ the tip amount,
- ▶ total bill,
- ▶ payer gender,
- ▶ smoking/non-smoking section,
- ▶ time of day,
- ▶ day of the week, and
- ▶ size of the party

The approach is to fit a regression model to predict the tip rate.

The fitted model is

$$\text{▶ } \textit{tip_rate} = 0.18 - 0.01 \times \textit{party_size}$$

if size of the dining party increases by one (leading to a higher bill), the tip rate will decrease by 1%.

Data Analytics Process

Steps in Data Analytics

- ▶ Setting Questions
- ▶ Data Wrangling
- ▶ Exploratory Data Analysis
- ▶ Modeling
- ▶ Story telling

Iterative process

The process is rarely linear.

Each step can push data scientist to revisit methods, techniques
... or reconsider whether the original question was the right one?
And the final answer simply sparks more questions!

Data Analytics: Setting Questions

Ask an Interesting Question

Steps

- ▶ Is there a business goal to achieve?
- ▶ Some object of scientific interest that would be helpful to discover?
- ▶ What parameters would the ideal answer fulfill?

Data Analytics: Data Wrangling

Design a Data Collection Program

- ▶ Establish whether or not the data exists in the real world and is relevant to the question
- ▶ Devise a collection scheme to acquire it
Logistical considerations? Cost? Privacy issues?
- ▶ Coordinate with departments or agencies needed for collection program liaison

Collect and Review the Data

- ▶ Store the incoming data to allow modeling and reporting
- ▶ Join data from multiple sources in relevant & logical manner
- ▶ Check for anomalies or unusual patterns
 - ▶ Caused by the collection process?
 - ▶ Inherent to topic of investigation?
 - ▶ Correct them, or develop new collection scheme?

Data Analytics: Data Wrangling

Data Wrangling or Data Munging

Bring skills and intuition to bear ...
to take messy, incoherent information ...
and shuffle it into clean, accessible sets

“Munging” the Data

- ▶ Select your tools to comb through raw
- ▶ Store the munged data as a fresh data set, **or**
- ▶ use programmatic pre-processing for each subsequent query

Data Analytics: Exploratory Data Analysis

Exploratory Data Analysis

Learn about the properties of the data

Steps

- ▶ Descriptive statistics: mean/median and variance, quantiles, outliers
- ▶ Correlation
- ▶ Fitting curves and distributions
- ▶ Dimension reduction
- ▶ Clustering

Data Analytics: Modeling

Modeling

the fun stuff of getting “*meaning*” from a clean data set

Steps

- ▶ Build a data model to fit the question
- ▶ Validate the model against the actual collected data
- ▶ Perform the necessary statistical analyses
- ▶ Machine-learning or recursive analysis
- ▶ Regression testing and other classical statistical analysis techniques
- ▶ Compare results against other techniques or sources

Data Analytics: Story telling

Visualize and Communicate the Results

The most challenging part of the data scientist's job is taking the results of the investigation and presenting them to the public or internal consumers of information in a way that makes sense and can be easily communicated.

Steps

- ▶ Graph or chart the information
- ▶ Tell a story to fit the results: Interpret the data to describe the real-world sources in a plausible manner
- ▶ Assist decision-makers in using the results to drive their decisions

Lecture 2 Numbers and Data

1. Measurement Scales
2. Normalization
3. Accuracy & Precision
4. Significant Digits
5. Data Formats
6. Data Schemas
7. Metadata
8. Self-Descriptive Data

Data Scales

Categorical

Nominal

Values have *names* as in enum or scalar type
equality testing allowed
mode is measure of central tendency

Ordinal

Ranked values, such as *good, better, best*
equality and comparison allowed
median is measure of central tendency
mean and deviation do not make sense

Continuous

Interval

Difference between values can be determined, eg integers
has no absolute zero
equality, comparison, $+$, $-$ allowed
mean is measure of central tendency; deviation makes sense

Ratio

Value is a ratio of continuous values, eg real number
has absolute zero
also \times , $/$ allowed
geometric mean is measure of central tendency

Data Scales

See video from UoVirginia: <https://www.youtube.com/watch?v=zHcQPKP6NpM>

Robust Statistics

median and Inter-Quartile Range (IQR) are robust to outliers

Outliers — John Tukey's Definition

Outlier is more than 1.5 times IQR from Q1 or Q3

Extreme value is more than 3.0 times IQR from Q1 or Q3

Plots — Categorical Data

Bar chart shows frequency, so shows modes (one or more)

Plots — Continuous Data

Histogram shows frequency, so shows modes (one or more)

Box plot shows median, Q1, Q3 box and whiskers to min and max
if outliers then shows fences at $Q1 - 1.5IQR$ and $Q3 + 1.5IQR$

Both show central tendency, variability, and skewness; not modes

Contingency Tables and Scatter Plots

Normalization

A normal form ...

is a unique representation for an entity

Examples

a string “ *the Happiest day of My Life* ”

to all lower case

and without leading or trailing blanks

and only one blank between words

“the happiest day of my life”

Normalization creates a normal form

allows simple test for equality

More Examples

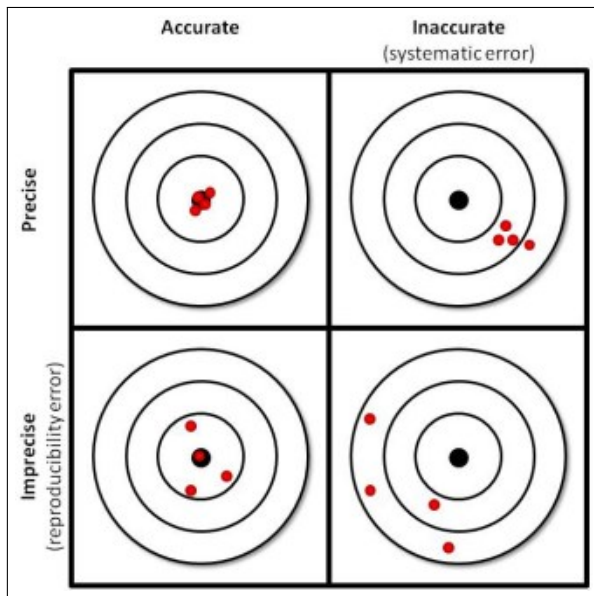
Names

Dates

Currency

Metric vs Imperial measurements

Accuracy and Precision



Significant Digits

Problem

Showing more digits in a number than are meaningful
Especially in decimal component

Examples

0.046 has two significant digits

4009 kg has four significant digits

7.90 has three significant digits

8200 has 2, 3, or 4 significant digits (**unclear**)

8.200×10^3 has four significant digits

8.20×10^3 has three significant digits

8.2×10^3 has two significant digits

Problem

Need to know significant digits for input data
Need to keep track of sig. digits in arithmetic
Be careful formatting output

Reference

https://www.physics.uoguelph.ca/tutorials/sig_fig/SIG_dig.htm

Significant Digits

Decimal Point Convention

8200. means that zero's are significant, so 4 significant digits

8200 means that zero's are not significant, so 2 significant digits

Calculating Number of Significant Digits

Basically, never more than smallest number of significant digits amongst the inputs

See https://www.saddleback.edu/faculty/jzoval/worksheets_tutorials/ch1worksheets/sig_figs_in_calc_rules_7_1_09.pdf

Data Formats

comma-separated values (csv)

Tab-separated values (tsv)

Attribute-Relation File Format (ARFF)

XML

RDF

Binary files (BLOBs)

HDF5 (Hierarchical Data Format version 5)

Data Formats — ARFF — Weka

ARFF files

ASCII files: Header followed by Data

Header

- ▶ the name of the relation,
- ▶ a list of the attributes (columns in data),
- ▶ their types

```
% 1. Title: Iris Plants Database
%
% 2. Sources:
%   (a) Creator: R.A. Fisher
%   (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
%   (c) Date: July, 1988
%
@RELATION iris

@ATTRIBUTE sepallength NUMERIC
@ATTRIBUTE sepalwidth NUMERIC
@ATTRIBUTE petallength NUMERIC
@ATTRIBUTE petalwidth NUMERIC
@ATTRIBUTE class {Iris-setosa,Iris-versicolor,Iris-virginica}
```


Data Formats — ARFF

Data looks like

@DATA

```
5.1,3.5,1.4,0.2,Iris-setosa  
4.9,3.0,1.4,0.2,Iris-setosa  
4.7,3.2,1.3,0.2,Iris-setosa  
4.6,3.1,1.5,0.2,Iris-setosa  
5.0,3.6,1.4,0.2,Iris-setosa  
5.4,3.9,1.7,0.4,Iris-setosa  
4.6,3.4,1.4,0.3,Iris-setosa  
5.0,3.4,1.5,0.2,Iris-setosa  
4.4,2.9,1.4,0.2,Iris-setosa  
4.9,3.1,1.5,0.1,Iris-setosa
```

Data Schemas

Tidy Data Schema in R

Tabular format with properties

1. Each variable is saved in its own column
2. Each observation is saved in its own row
3. Each type of observation is stored in its own (single) table

See video

<https://www.youtube.com/watch?v=1ELALQ10-yM&list=PL9HYL-VRX0oQOWAFoKHFQAsWAI3ImbNPk&index=2>

Metadata

Metadata

is data that provides information about other data

For example

Means of creation of the data

Purpose of the data

Time and date of creation

Creator or author of the data

Location on a computer network where the data was created

Standards used

File size

Data quality

Source of the data

Process used to create the data

Provenance of Data

is the origin and/or history of an object (that is, data, in our case).

Self-Descriptive Data

You can make sense of the file as a stand-alone.

therefore human-readable

ARFF

XML

HDF

Lecture 3 Data Warehouses

Matteo Golfarelli and Stefano Rizzi, **Data Warehouse Design: Modern Principles and Methodologies**, *Chapter 1:*

Introduction to Data Warehousing, McGraw-Hilli Companies, Italy, 2009.

1. Databases
2. Data Warehouses
3. Data Marts
4. Extract-Transform-Load
5. SQL: Select-From-Where
6. SQL: GroupBy-Aggregation
7. Data Cubes
8. Dashboards
9. Business Intelligence

Lecture 4 Python

1. Python array
2. Python numpy ndarray
3. Python pandas DataFrame

Python Types

| Built-in type | Operator | Mutable | Example | Description |
|---------------|----------|---------|------------------------|---------------------------------------|
| bool | | No | True | Boolean |
| bytearray | | Yes | bytearray(b'\x01\x04') | Array of bytes |
| bytes | b'' | No | b'\x00\x17\x02' | |
| complex | | No | (1+4j) | Complex number |
| dict | {:} | Yes | {'a': True, 45: 'b'} | Dictionary, indexed by, e.g., strings |
| float | | No | 3.1 | Floating point number |
| frozenset | | No | frozenset({1, 3, 4}) | Immutable set |
| int | | No | 17 | Integer |
| list | [] | Yes | [1, 3, 'a'] | List |
| set | {} | Yes | {1, 2} | Set with unique elements |
| slice | : | No | slice(1, 10, 2) | Slice indices |
| str | "" or '' | No | "Hello" | String |
| tuple | (,) | No | (1, 'Hello') | Tuple |

Python array

Module array in Python 3.3

```
class array.array(typecode[, initializer])
```

array

an object type for an array of basic values:

characters, integers, floating point numbers

Arrays behave very much like lists

except the type of objects is constrained

The type is specified a type code, eg

'l' C signed long (int)

'u' unicode character

'd' C double (float)

Example

```
array('l')
```

```
array('l', [1, 2, 3, 4, 5])
```

```
array('d', [1.0, 2.0, 3.14])
```


Python Array Methods

| Method | Description |
|----------------------------------|--|
| <u>append()</u> | Adds an element at the end of the list |
| <u>clear()</u> | Removes all the elements from the list |
| <u>copy()</u> | Returns a copy of the list |
| <u>count()</u> | Returns the number of elements with the specified value |
| <u>extend()</u> | Add the elements of a list (or any iterable), to the end of the current list |
| <u>index()</u> | Returns the index of the first element with the specified value |
| <u>insert()</u> | Adds an element at the specified position |
| <u>pop()</u> | Removes the element at the specified position |
| <u>remove()</u> | Removes the first item with the specified value |
| <u>reverse()</u> | Reverses the order of the list |
| <u>sort()</u> | Sorts the list |

Python numpy Types

| Numpy type | Char | Mutable | Example | |
|------------|---------|---------|-------------------------------------|--|
| array | | Yes | <code>np.array([1, 2])</code> | One-, two, or many-dimensional |
| matrix | | Yes | <code>np.matrix([[1, 2]])</code> | Two-dimensional matrix |
| bool_ | | — | <code>np.array([1], 'bool_')</code> | Boolean, one byte long |
| int_ | | — | <code>np.array([1])</code> | Default integer, same as C's long |
| int8 | b | — | <code>np.array([1], 'b')</code> | 8-bit signed integer |
| int16 | h | — | <code>np.array([1], 'h')</code> | 16-bit signed integer |
| int32 | i | — | <code>np.array([1], 'i')</code> | 32-bit signed integer |
| int64 | l, p, q | — | <code>np.array([1], 'l')</code> | 64-bit signed integer |
| uint8 | B | — | <code>np.array([1], 'B')</code> | 8-bit unsigned integer |
| float_ | | — | <code>np.array([1.])</code> | Default float |
| float16 | e | — | <code>np.array([1], 'e')</code> | 16-bit half precision floating point |
| float32 | f | — | <code>np.array([1], 'f')</code> | 32-bit precision floating point |
| float64 | d | — | | 64-bit double precision floating point |
| float128 | g | — | <code>np.array([1], 'g')</code> | 128-bit floating point |
| complex_ | | — | | Same as <code>complex128</code> |
| complex64 | | — | | Single precision complex number |
| complex128 | | — | <code>np.array([1+1j])</code> | Double precision complex number |
| complex256 | | — | | 2 128-bit precision complex number |

Python numpy ndarray

numpy.array(object, dtype=None, copy=True, order='K', subok=False, ndmin=0)

Create an array.

Parameters: **object** : *array_like*

An array, any object exposing the array interface, an object whose `__array__` method returns an array, or any (nested) sequence.

dtype : *data-type, optional*

The desired data-type for the array. If not given, then the type will be determined as the minimum type required to hold the objects in the sequence. This argument can only be used to 'upcast' the array. For downcasting, use the `.astype(t)` method.

copy : *bool, optional*

If true (default), then the object is copied. Otherwise, a copy will only be made if `__array__` returns a copy, if obj is a nested sequence, or if a copy is needed to satisfy any of the other requirements (**dtype**, **order**, etc.).

Python numpy ndarray

Examples

```
>>> np.array([1, 2, 3])
array([1, 2, 3])
```

Upcasting:

```
>>> np.array([1, 2, 3.0])
array([ 1.,  2.,  3.])
```

More than one dimension:

```
>>> np.array([[1, 2], [3, 4]])
array([[1, 2],
       [3, 4]])
```

Minimum dimensions 2:

```
>>> np.array([1, 2, 3], ndmin=2)
array([[1, 2, 3]])
```

Type provided:

```
>>> np.array([1, 2, 3], dtype=complex)
array([ 1.+0.j,  2.+0.j,  3.+0.j])
```

Python pandas Types

| Pandas type | Mutable | Example | Description |
|------------------|---------|-------------------------------------|---------------------------------|
| Series | Yes | <code>pd.Series([2, 3, 6])</code> | One-dimension (vector-like) |
| DataFrame | Yes | <code>pd.DataFrame([[1, 2]])</code> | Two-dimensional (matrix-like) |
| Panel | Yes | <code>pd.Panel([[[1, 2]]])</code> | Three-dimensional (tensor-like) |
| Panel4D | Yes | <code>pd.Panel4D([[[[1]]]])</code> | Four-dimensional |

Python pandas Types

Panel and Panel4D

are deprecated, and replaced by xarray

xarray

xarray: N-D labeled arrays and datasets in Python

xarray (formerly **xray**) is an open source project and Python package that makes working with labelled multi-dimensional arrays simple, efficient, and fun!

Xarray introduces labels in the form of dimensions, coordinates and attributes on top of raw [NumPy](#)-like arrays, which allows for a more intuitive, more concise, and less error-prone developer experience. The package includes a large and growing library of domain-agnostic functions for advanced analytics and visualization with these data structures.

Xarray was inspired by and borrows heavily from [pandas](#), the popular data analysis package focused on labelled tabular data. It is particularly tailored to working with [netCDF](#) files, which were the source of xarray's data model, and integrates tightly with [dask](#) for parallel computing.

Python pandas DataFrame

pandas.DataFrame

`class pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)`

[\[source\]](#)

Two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). Arithmetic operations align on both row and column labels. Can be thought of as a dict-like container for Series objects. The primary pandas data structure.

Parameters:

data : *ndarray (structured or homogeneous), Iterable, dict, or DataFrame*

Dict can contain Series, arrays, constants, or list-like objects

Changed in version 0.23.0: If data is a dict, argument order is maintained for Python 3.6 and later.

index : *Index or array-like*

Index to use for resulting frame. Will default to RangeIndex if no indexing information part of input data and no index provided

columns : *Index or array-like*

Column labels to use for resulting frame. Will default to RangeIndex (0, 1, 2, ..., n) if no column labels are provided

dtype : *dtype, default None*

Data type to force. Only a single dtype is allowed. If None, infer

copy : *boolean, default False*

Copy data from inputs. Only affects DataFrame / 2d ndarray input

Python pandas DataFrame

See also:

`DataFrame.from_records`

Constructor from tuples, also record arrays.

`DataFrame.from_dict`

From dicts of Series, arrays, or dicts.

`DataFrame.from_items`

From sequence of (key, value) pairs `pandas.read_csv`, `pandas.read_table`, `pandas.read_clipboard`.

Python pandas DataFrame

Attributes

| | |
|----------------|--|
| T | Transpose index and columns. |
| at | Access a single value for a row/column label pair. |
| axes | Return a list representing the axes of the DataFrame. |
| blocks | (DEPRECATED) Internal property, property synonym for <code>as_blocks()</code> . |
| columns | The column labels of the DataFrame. |
| dtypes | Return the dtypes in the DataFrame. |
| empty | Indicator whether DataFrame is empty. |
| ftypes | Return the ftypes (indication of sparse/dense and dtype) in DataFrame. |
| iat | Access a single value for a row/column pair by integer position. |
| iloc | Purely integer-location based indexing for selection by position. |
| index | The index (row labels) of the DataFrame. |
| is_copy | Return the copy. |
| ix | A primarily label-location based indexer, with integer position fallback. |
| loc | Access a group of rows and columns by label(s) or a boolean array. |
| ndim | Return an int representing the number of axes / array dimensions. |
| shape | Return a tuple representing the dimensionality of the DataFrame. |
| size | Return an int representing the number of elements in this object. |
| style | Property returning a Styler object containing methods for building a styled HTML representation for the DataFrame. |
| values | Return a Numpy representation of the DataFrame. |

Python pandas DataFrame

Syntax – Creating DataFrames

| | a | b | c |
|---|---|---|----|
| 1 | 4 | 7 | 10 |
| 2 | 5 | 8 | 11 |
| 3 | 6 | 9 | 12 |

```
df = pd.DataFrame(  
    {"a" : [4 ,5, 6],  
     "b" : [7, 8, 9],  
     "c" : [10, 11, 12]},  
    index = [1, 2, 3])
```

Specify values for each column.

```
df = pd.DataFrame(  
    [[4, 7, 10],  
     [5, 8, 11],  
     [6, 9, 12]],  
    index=[1, 2, 3],  
    columns=['a', 'b', 'c'])
```

Specify values for each row.

| | | a | b | c |
|---|---|---|---|----|
| n | v | | | |
| d | 1 | 4 | 7 | 10 |
| | 2 | 5 | 8 | 11 |
| e | 2 | 6 | 9 | 12 |

```
df = pd.DataFrame(  
    {"a" : [4 ,5, 6],  
     "b" : [7, 8, 9],  
     "c" : [10, 11, 12]},  
    index = pd.MultiIndex.from_tuples(  
        [('d',1),('d',2),('e',2)],  
        names=['n','v']))
```

Create DataFrame with a MultiIndex

Python pandas DataFrame

Tidy Data – A foundation for wrangling in pandas

In a tidy data set:



Each **variable** is saved
in its own **column**

&

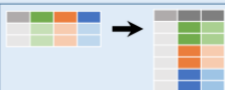


Each **observation** is
saved in its own **row**

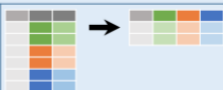
Tidy data complements pandas's **vectorized operations**. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.

Python pandas DataFrame

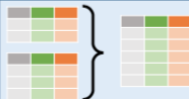
Reshaping Data – Change the layout of a data set



`pd.melt(df)`
Gather columns into rows.



`df.pivot(columns='var', values='val')`
Spread rows into columns.



`pd.concat([df1,df2])`
Append rows of DataFrames



`pd.concat([df1,df2], axis=1)`
Append columns of DataFrames

`df.sort_values('mpg')`

Order rows by values of a column (low to high).

`df.sort_values('mpg',asending=False)`

Order rows by values of a column (high to low).

`df.rename(columns = {'y':'year'})`

Rename the columns of a DataFrame

`df.sort_index()`

Sort the index of a DataFrame

`df.reset_index()`

Reset index of DataFrame to row numbers, moving index to columns.

`df.drop(columns= ['Length', 'Height'])`

Drop columns from DataFrame

Python pandas DataFrame

Summarize Data

df['w'].value_counts()

Count number of rows with each unique value of variable

len(df)

of rows in DataFrame.

df['w'].nunique()

of distinct values in a column.

df.describe()

Basic descriptive statistics for each column (or GroupBy)



pandas provides a large set of **summary functions** that operate on different kinds of pandas objects (DataFrame columns, Series, GroupBy, Expanding and Rolling (see below)) and produce single values for each of the groups. When applied to a DataFrame, the result is returned as a pandas Series for each column. Examples:

sum()

Sum values of each object.

count()

Count non-NA/null values of each object.

median()

Median value of each object.

quantile([0.25,0.75])

Quantiles of each object.

apply(function)

Apply function to each object.

min()

Minimum value in each object.

max()

Maximum value in each object.

mean()

Mean value of each object.

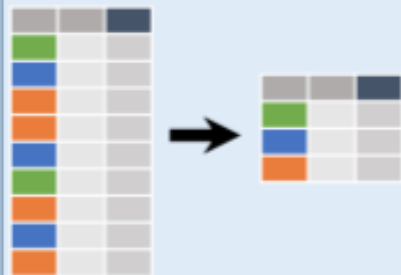
var()

Variance of each object.

std()

Standard deviation of each object.

Group Data



`df.groupby(by="col")`

Return a GroupBy object, grouped by values in column named "col".

`df.groupby(level="ind")`

Return a GroupBy object, grouped by values in index level named "ind".

All of the summary functions listed above can be applied to a group.
Additional GroupBy functions:

`size()`

Size of each group.

`agg(function)`

Aggregate group using function.

Handling Missing Data

`df.dropna()`

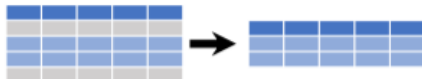
Drop rows with any column having NA/null data.

`df.fillna(value)`

Replace all NA/null data with value.

Python pandas DataFrame

Subset Observations (Rows)



`df[df.Length > 7]`

Extract rows that meet logical criteria.

`df.drop_duplicates()`

Remove duplicate rows (only considers columns).

`df.head(n)`

Select first n rows.

`df.tail(n)`

Select last n rows.

`df.sample(frac=0.5)`

Randomly select fraction of rows.

`df.sample(n=10)`

Randomly select n rows.

`df.iloc[10:20]`

Select rows by position.

`df.nlargest(n, 'value')`

Select and order top n entries.

`df.nsmallest(n, 'value')`

Select and order bottom n entries.

Logic in Python (and pandas)

| | | | |
|----|------------------------|---|-------------------------------------|
| < | Less than | != | Not equal to |
| > | Greater than | <code>df.column.isin(values)</code> | Group membership |
| == | Equals | <code>pd.isnull(obj)</code> | Is NaN |
| <= | Less than or equals | <code>pd.notnull(obj)</code> | Is not NaN |
| >= | Greater than or equals | <code>&, , ~, ^, df.any(), df.all()</code> | Logical and, or, not, xor, any, all |

Python pandas DataFrame

Combine Data Sets

| adf | | bdf | |
|-----|----|-----|----|
| x1 | x2 | x1 | x3 |
| A | 1 | A | T |
| B | 2 | B | F |
| C | 3 | D | T |



Standard Joins

| x1 | x2 | x3 |
|----|----|-----|
| A | 1 | T |
| B | 2 | F |
| C | 3 | NaN |

```
pd.merge(adf, bdf,  
        how='left', on='x1')
```

Join matching rows from bdf to adf.

| x1 | x2 | x3 |
|----|-----|----|
| A | 1.0 | T |
| B | 2.0 | F |
| D | NaN | T |

```
pd.merge(adf, bdf,  
        how='right', on='x1')
```

Join matching rows from adf to bdf.

| x1 | x2 | x3 |
|----|----|----|
| A | 1 | T |
| B | 2 | F |

```
pd.merge(adf, bdf,  
        how='inner', on='x1')
```

Join data. Retain only rows in both sets.

| x1 | x2 | x3 |
|----|-----|-----|
| A | 1 | T |
| B | 2 | F |
| C | 3 | NaN |
| D | NaN | T |

```
pd.merge(adf, bdf,  
        how='outer', on='x1')
```

Join data. Retain all values, all rows.

Filtering Joins

| x1 | x2 |
|----|----|
| A | 1 |
| B | 2 |

```
adf[adf.x1.isin(bdf.x1)]
```

All rows in adf that have a match in bdf.

| x1 | x2 |
|----|----|
| C | 3 |

```
adf[~adf.x1.isin(bdf.x1)]
```

All rows in adf that do not have a match in bdf.

Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
      .rename(columns={
                  'variable' : 'var',
                  'value' : 'val'})
      .query('val >= 200')
      )
```

Lecture 5 Descriptive Data Analytics

Descriptive Analytics is describing your data;
that is, data from past activities

1. Five Numbers
2. Python pandas describe()
3. Plots: Bar Chart, Histogram, Box Plot
4. Pareto Diagrams
5. Violin Plot
6. Normalization and Z-scores
7. Comparing Two Attributes
8. Correlation is not Causality

Describing Data

Four Features to Describe Data Sets

Center: the point where about half of the observations are on either side.

Spread: the variability of the data.

Shape: described by symmetry, skewness, number of peaks, etc.

Unusual features: gaps where there are no observations and outliers.

Five Numbers of Robust Statistical Descriptors

Five Number Summary

- ▶ maximum
- ▶ third quartile Q_3
- ▶ median
- ▶ first quartile Q_1
- ▶ minimum

Descriptors

What Else to Describe?

- ▶ number of observations
- ▶ number of entries
- ▶ number of unique entries
- ▶ number of missing entries
- ▶ number of outliers
- ▶ number of extreme values

Python pandas describe

Describing a numeric series.

```
>>> s = pd.Series([1, 2, 3])
>>> s.describe()
count      3.0
mean       2.0
std        1.0
min        1.0
25%        1.5
50%        2.0
75%        2.5
max        3.0
dtype: float64
```

Describing a categorical series.

```
>>> s = pd.Series(['a', 'a', 'b', 'c'])
>>> s.describe()
count      4
unique      3
top         a
freq        2
dtype: object
```


Python pandas describe

```
>>> df = pd.DataFrame({'categorical': pd.Categorical(['d','e','f']),  
...                    'numeric': [1, 2, 3],  
...                    'object': ['a', 'b', 'c']  
...                    })
```

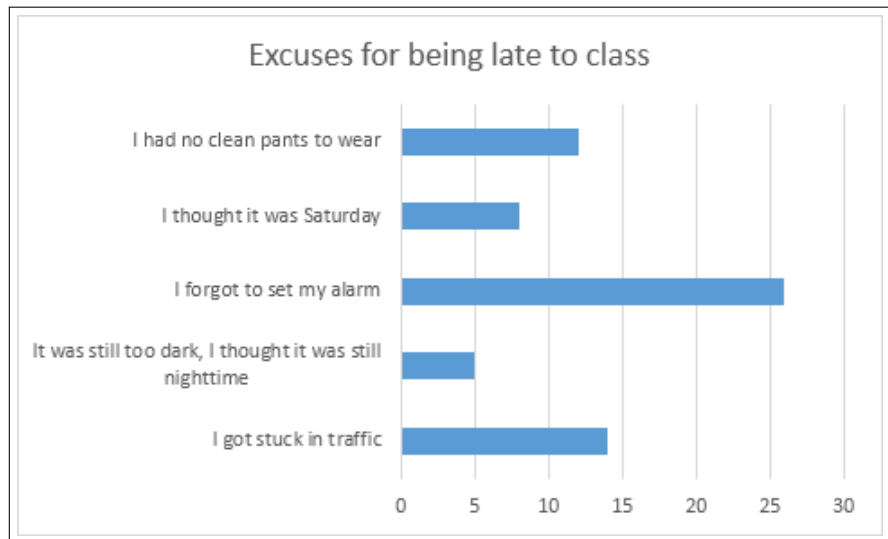
Describing all columns of a DataFrame regardless of data type.

```
>>> df.describe(include='all')
```

| | categorical | numeric | object |
|--------|-------------|---------|--------|
| count | 3 | 3.0 | 3 |
| unique | 3 | NaN | 3 |
| top | f | NaN | c |
| freq | 1 | NaN | 1 |
| mean | NaN | 2.0 | NaN |
| std | NaN | 1.0 | NaN |
| min | NaN | 1.0 | NaN |
| 25% | NaN | 1.5 | NaN |
| 50% | NaN | 2.0 | NaN |
| 75% | NaN | 2.5 | NaN |
| max | NaN | 3.0 | NaN |

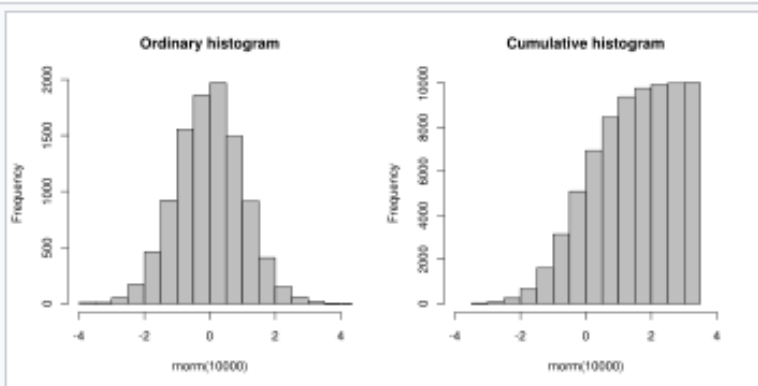
Bar Chart


Bar Chart



Histogram

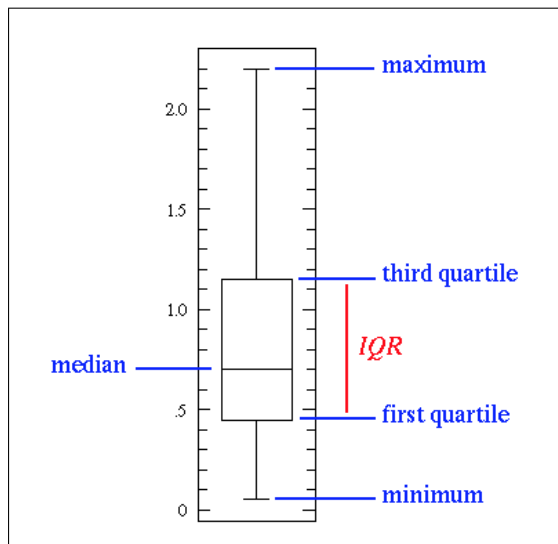
Histogram



An ordinary and a cumulative histogram of the same data. 
The data shown is a random sample of 10,000 points from a normal distribution with a mean of 0 and a standard deviation of 1.

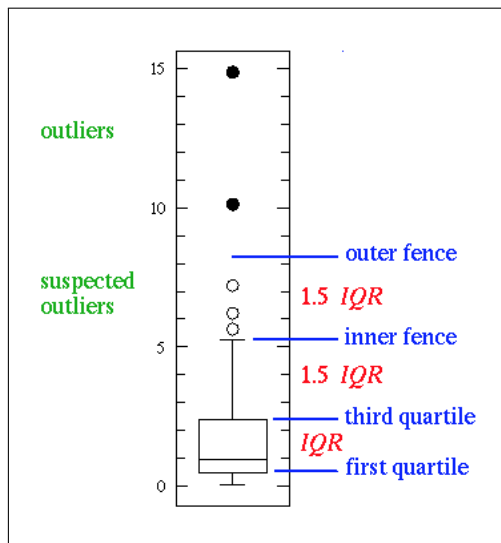
Box Plot

Box Plot



Box Plot

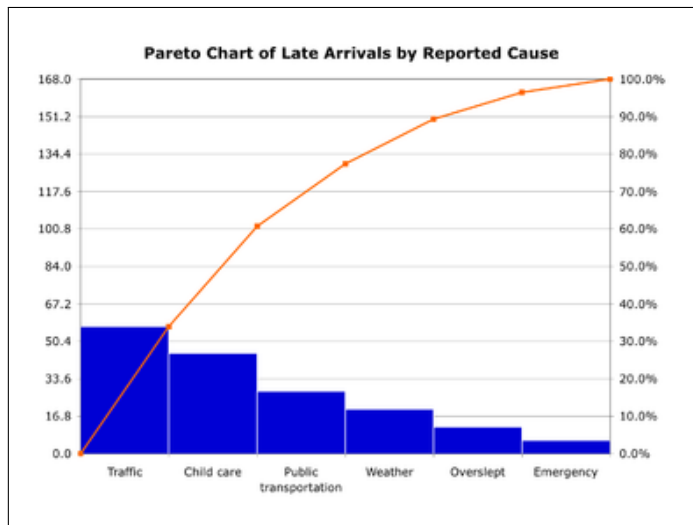
Box Plot



Pareto Diagram

Pareto Diagram

Order by decreasing frequency

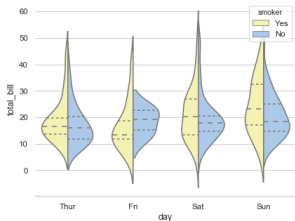


Violin Plot

Violin Plot

shows frequency too

Grouped violinplots with split violins



Python source code: [\[download source: grouped_violinplots.py\]](#)

```
import seaborn as sns
sns.set(style="whitegrid", palette="pastel", color_codes=True)

# Load the example tips dataset
tips = sns.load_dataset("tips")

# Draw a nested violinplot and split the violins for easier comparison
sns.violinplot(x="day", y="total_bill", hue="smoker",
               split=True, inner="quart",
               palette={"Yes": "y", "No": "b"},
               data=tips)
sns.despine(left=True)
```

Normalization and Z-scores

Normalization of Numbers

means getting them on the same scale

so they can be compared *apples* to *apples*

eg use frequency rather than count

eg use Z-scores of a normal distribution
to allow for different mean and variance

Comparing Two Attributes

Adapted from Frank E. Harrell Jr. on graphics:

<http://biostat.mc.vanderbilt.edu/twiki/pub/Main/StatGraphCourse/graphscourse.pdf>

Two categorical variables

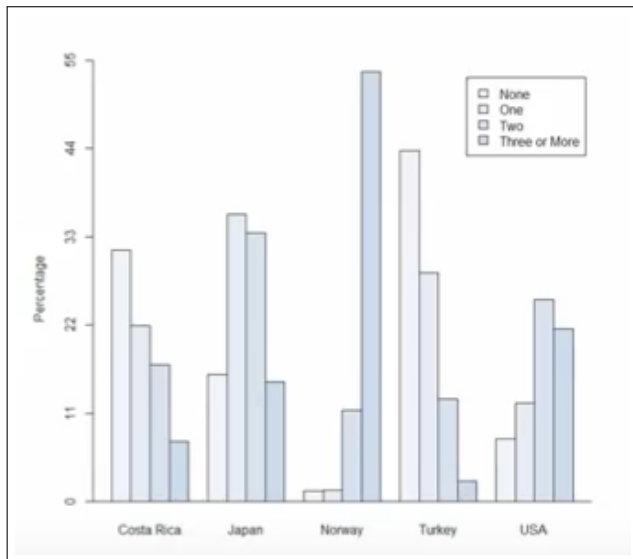
- Use frequency table
 - One categorical variable and other continuous variable
- Box plots of continuous variable values for each category of categorical variable
- Side-by-side dot plots (means + measure of uncertainty, SE or confidence interval)
 - Do not link means across categories!

Two continuous variables

- Scatter plot of raw data if sample size is not too large
- Prediction with confidence bands

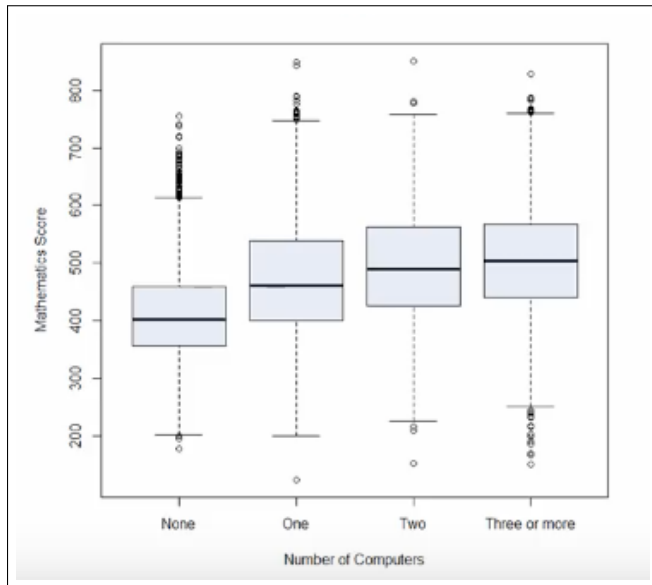
Comparing Two Attributes

Compare categorical and categorical



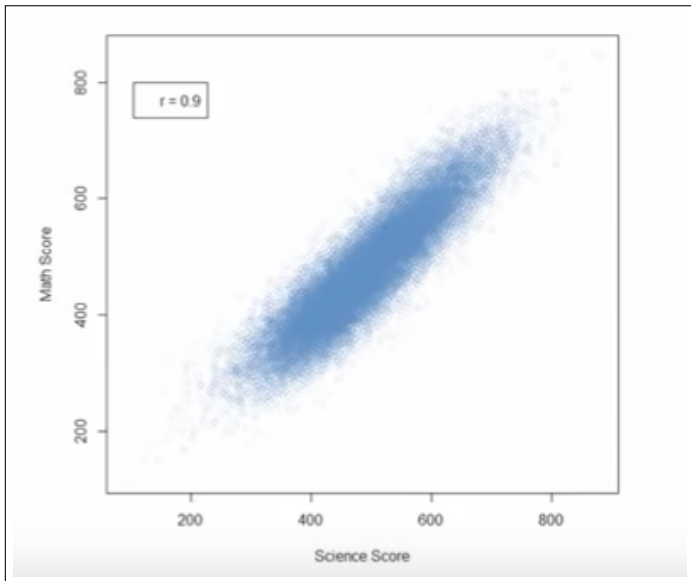
Comparing Two Attributes

Compare categorical and continuous



Comparing Two Attributes

Compare continuous and continuous



Correlation is not Causality

These are different concepts
and
correlation does not imply causality

Lecture 5 Data Wrangling

1. Data Wrangling Overview

- ▶ Discover
- ▶ Structure
- ▶ Cleanse
- ▶ Enrich
- ▶ Validate
- ▶ Publish

Data Wrangling — Discovery

Discover what data is available

Extract step of ETL

Data Wrangling — Structure

Organize data into suitable format

Transform step of ETL

Data Wrangling — Cleanse

Clean the data

Iterative step with basic data analysis

Data Wrangling — Enrich

Discover and include related data

Integrate new data sets and data types
add more data fields

Data Wrangling — Validate

Check data is consistent and complete

Consistency

Does your data fit into expected values for it?

Do field values match the data type for the column?

Are values within acceptable ranges?

Are rows unique? Duplicated?

Completeness

Are all expected values included in your data?

Are some fields missing values?

Are there expected values that are not present in the dataset?

Test routines for your data wrangling process

Data Wrangling — Publish

Make available for analysis

Load step of ETL

into data warehouse in traditional business intelligence setting

Lecture 6 Data Cleaning

1. Data Cleaning
2. Data Compatibility
3. Missing Data
4. Outliers
5. Remove Duplicates
6. Consistency of Data

Data Cleaning

Data Cleaning

detecting and correcting corrupt or inaccurate records

Error

is information lost in data acquisition

Artifact

systematic problem arising from data processing

Sniff Test

look closely to see if something may be wrong

Data Cleaning Process

Iterative

Keep raw data from data acquisition

You need a reference data set
you will need to re-process it many times

Script your cleaning steps

to refine and re-run your process

Data Compatibility

Comparing apples to apples

- ▶ unit conversions on numbers
- ▶ character code representations
- ▶ name unification
- ▶ time unification
- ▶ date unification
- ▶ financial unification

... And normalize to same scale

Z-scores

Missing Data

Delete observations or columns
that have missing values

Imputation — assign a value by inference

- ▶ fixed value, eg zero
- ▶ mean value (of column values)
- ▶ random value
from random observations
- ▶ by interpolation
from similar observations
by linear regression
or machine learning

Outliers

Duplicates

Remove duplicates

Duplicates arise due to merging multiple data sets

Consistency of Data

Cluster/sort data values

To bring together
duplicate and similar data values
to make it easy to see differences/errors
(See OpenRefine video 1 of 3)

Cluster observations

To bring together
duplicate and similar observations
to make it easy to see differences/errors

Check for consistency

Differences need to be investigated