Applied Data Analysis (CS401)

ada

Lecture 2
Handling data

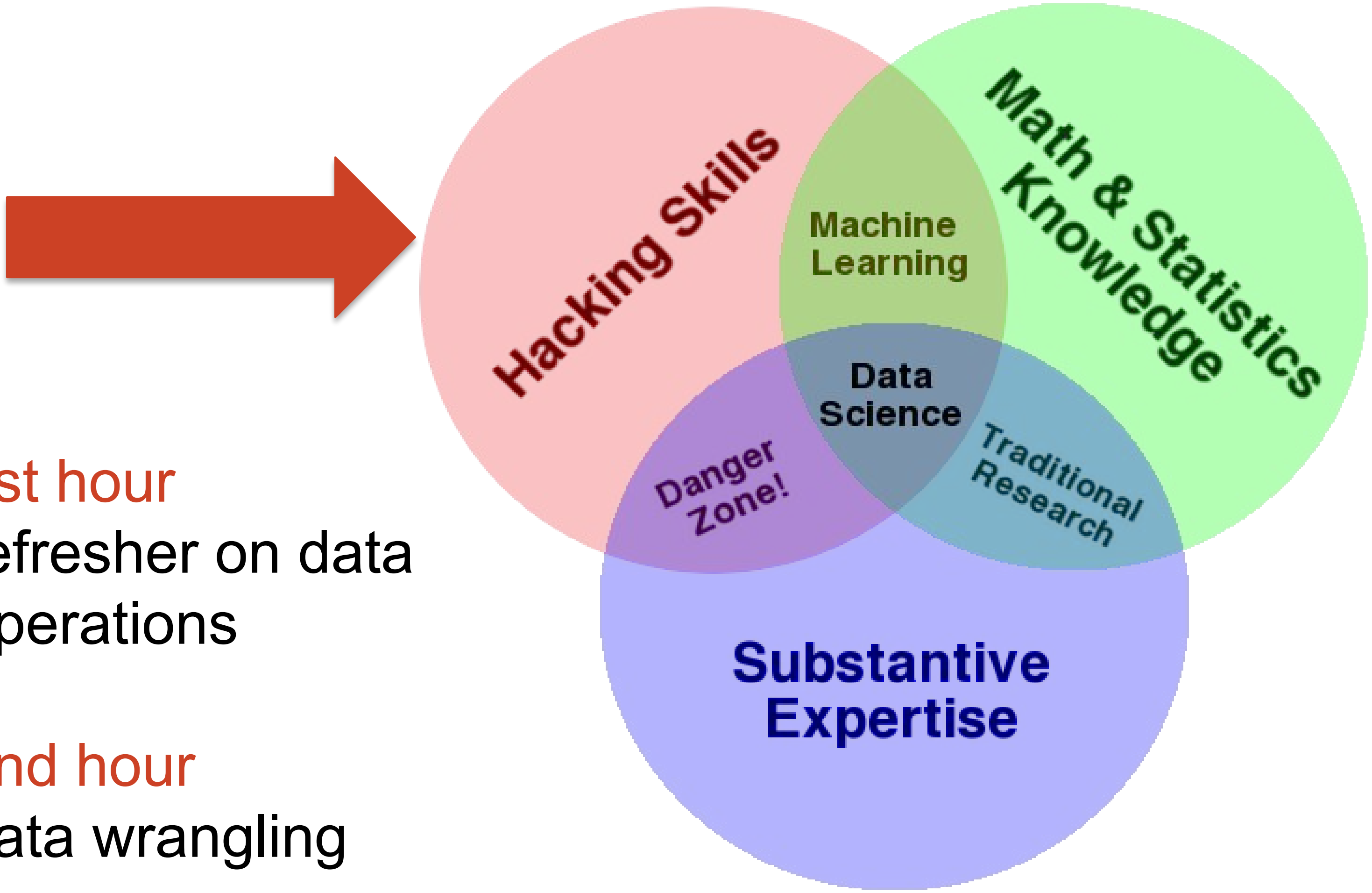
Robert West

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

dlab

Announcements

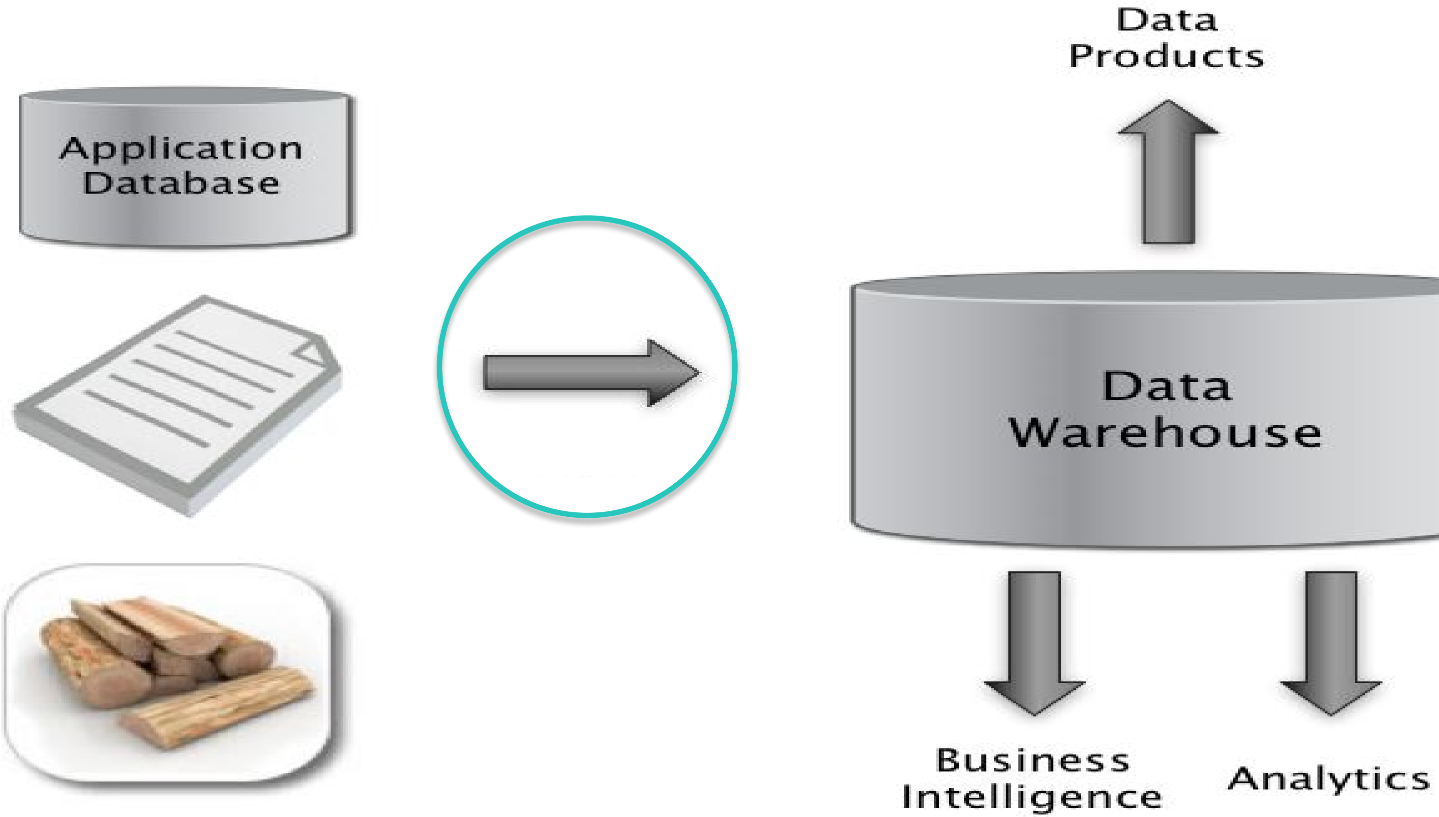
- Register your teams (3 people) here by tomorrow
  - May shuffle till after Homework 2, then fixed (incl. project)
- Homework 1 to be released in tomorrow’s lab session
  - Due October 11, 23:59 (i.e., in 2 weeks)
- Interested in preparing course notes in LaTeX?
  - Message @sharbat on Mattermost
  - Remuneration: extra credit, karma



1st hour
refresher on data operations

2nd hour
data wrangling

The big picture



Application Database

Data Warehouse

Data Products

Business Intelligence

Analytics

Key concept: structured data

A **data model** is a collection of concepts for describing data.

A **schema** is a description of a particular collection of data, using a given data model.

A toy model and schema

Meteorological measurements

- Concepts in data model:
  numbers, samples, vectors, matrices
- Samples are vectors of numbers; time series is matrix obtained by stacking vectors
- Schema: column 1 is integer and has time stamp; col 2 is float and contains temperature, etc.

Examples of data models

- Relational model
- Document model
- Network model
- …

The relational model

- The relational model is ubiquitous:
  - MySQL, PostgreSQL, Oracle, DB2, SQLite, …
  - You use it many times every day
- Data represented as tables (“relations”) describing
  - enitities,
  - relationships between entities
- Most of the data we will use can be “reduced” to the relation model

id	name
1	Bush
2	Trump
3	Obama

president	successor
1	3
3	2

What is a relation?

Relation: made up of 2 parts:

- *Schema:* specifies name of relation, plus name and type of each column

  Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real)

- *Instance:* the actual data at a given time
  - #rows = *cardinality*
  - #fields = *degree / arity*

Example: instance of students relation

sid	name	login	age	gpa
53666	Jones	jones@cs	18	5.4
53688	Smith	smith@eecs	18	5.2
53650	Smith	smith@math	19	5.8

Cardinality = 3, degree = 5 , all rows distinct

SQL ex.

```
SELECT   [DISTINCT]  target-list
FROM        relation-list
WHERE          qualification
```

```
SELECT        DISTINCT names
FROM          students
WHERE            age >= 19
```

relation-list: A list of relation names

target-list: A list of attributes of tables in *relation-list*

qualification: Comparisons combined using AND, OR and NOT.

- Comparisons are Attr *op* const or Attr1 *op* Attr2, where *op* is one of =≠<>≤≥

DISTINCT: optional keyword indicating that the answer should not contain duplicates.

- In SQL SELECT, the default is that duplicates are *not* eliminated! (Result is called a “multiset”)

Joins and inference

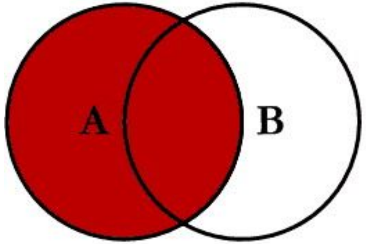
Chaining relations together is the basic inference method in relational DBs. It produces new relations (effectively new facts) from the data:

```
SELECT S.name, M.mortality
 FROM Students S, Mortality M
 WHERE S.Race=M.Race
```

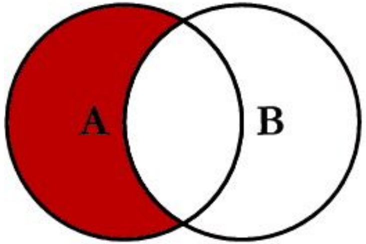
S **M**

Name	Name	Mortality	Mortality
Socrates	Socrates	Mortal	Mortal
Thor	Thor	Immortal	mmortal
Barney	Barney	Mortal	Mortal
Blarney stone	Blarney stone	Non-living	Non-living

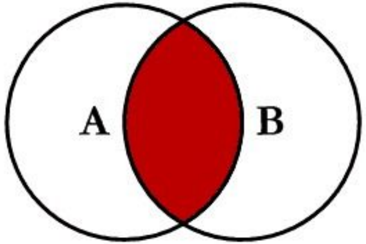
SQL JOINS



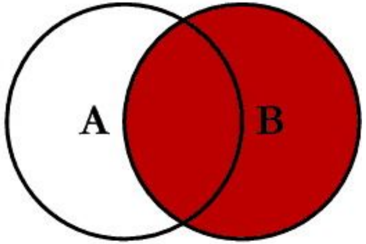
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



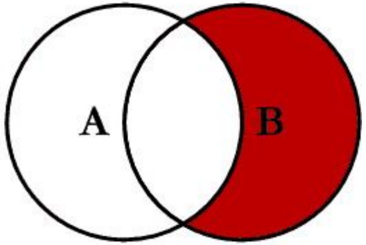
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



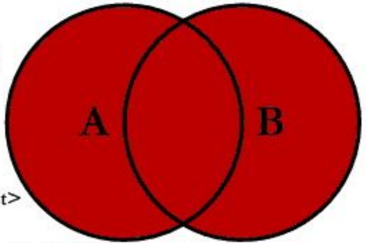
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



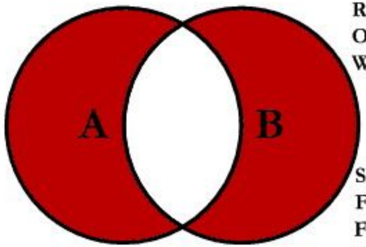
```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Aggregations and GroupBy

- One of the most common operations on data tables is aggregation (**count, sum, average, min, max,**…).
- They provide a means to see high-level patterns in the data, to make summaries of it, etc.
- You need ways of specifying which columns are being aggregated over, which is the role of a **GroupBy** operator.

Aggregations and GroupBy

sid	name	course	semester	grade	gpa
111	Jones	Stat 134	F13	A	4.0
111	Jones	CS 162	F13	B-	2.7
222	Smith	EE 141	S14	B+	3.3
222	Smith	CS162	F14	C+	2.3
222	Smith	CS189	F14	A-	3.7

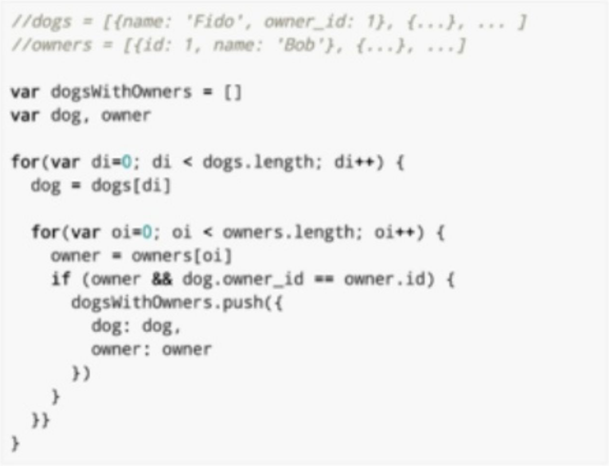
```
SELECT sid, name, AVG(gpa)
 FROM Students
 GROUP BY sid
```

sid	name	gpa
111	Jones	3.35
222	Smith	3.1

SQL is a declarative language

- SQL provides language for core data manipulations
- You think about what you want, not how to compute it

Imperative



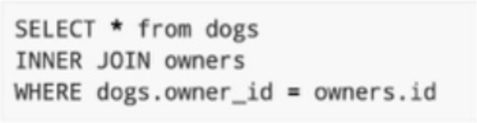
```
//dogs = [{name: 'Fido', owner_id: 1}, {...}, ... ]
//owners = [{id: 1, name: 'Bob'}, {...}, ...]

var dogsWithOwners = []
var dog, owner

for(var di=0; di < dogs.length; di++) {
  dog = dogs[di]

  for(var oi=0; oi < owners.length; oi++) {
    owner = owners[oi]
    if (owner && dog.owner_id == owner.id) {
      dogsWithOwners.push({
        dog: dog,
        owner: owner
      })
    }
  }}
}
```

Declarative

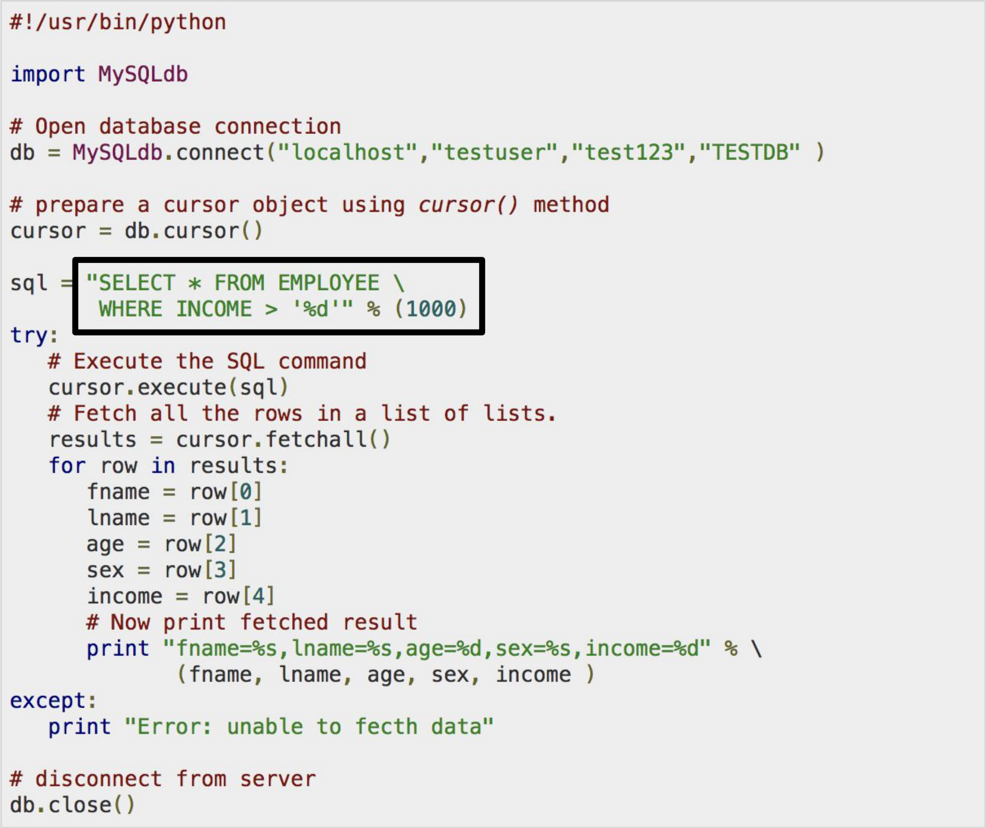


```
SELECT * from dogs
INNER JOIN owners
WHERE dogs.owner_id = owners.id
```

From: http://latentflip.com/imperative-vs-declarative/

SQL implementations

etc.



```
#!/usr/bin/python

import MySQLdb

# Open database connection
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

sql = "SELECT * FROM EMPLOYEE \
       WHERE INCOME > '%d'" % (1000)
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Fetch all the rows in a list of lists.
   results = cursor.fetchall()
   for row in results:
      fname = row[0]
      lname = row[1]
      age = row[2]
      sex = row[3]
      income = row[4]
      # Now print fetched result
      print "fname=%s,lname=%s,age=%d,sex=%s,income=%d" % \
             (fname, lname, age, sex, income )
except:
   print "Error: unable to fecth data"

# disconnect from server
db.close()
```

SQL and “SQL”

- The declarative-programming principles of SQL are widespread, even where it’s less obvious

“SQL”: Pandas/Python

- **Series**: a named, ordered dictionary
  - The keys of the dictionary are the **indexes**
  - Built on NumPy’s **ndarray**
  - Values can be any NumPy data type object
- **DataFrame**: a table with named columns (like relation in relational model)
  - Represented as a dict (col_name -> series)
  - Each Series object represents a column

Pandas operations (cf. Friday lab)

map() functions

filter (apply predicate to rows)

sort/group by

aggregate: sum, count, average, max, min

Pivot or reshape

Relational:

> union, intersection, difference, cartesian product (CROSS JOIN), select/filter, project, join: natural join (INNER JOIN), theta join, semi-join, etc.

Pandas vs. SQL

+ Pandas is lightweight and fast.

+ Natively Python, i.e., full SQL expressiveness plus the expressiveness of Python, especially for function evaluation.

+ Integration with plotting functions like Matplotlib.

- Tables must fit into memory.

- No post-load indexing functionality: indices are built when a table is created.

- No transactions, journaling, etc.

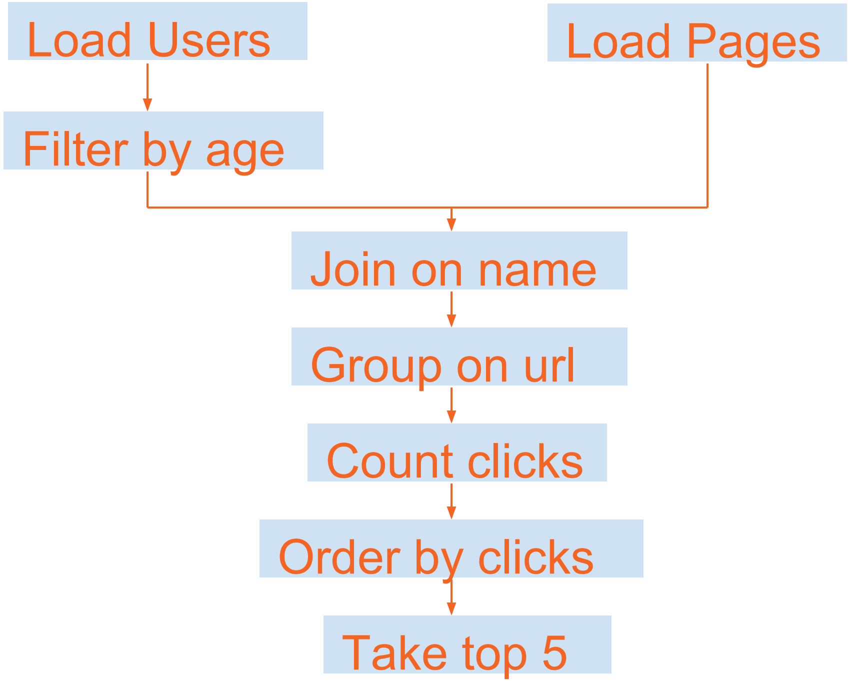
- Large, complex joins are slower.

“SQL”: Apache Pig

- Started at Yahoo! Research
- Features:
  - Expresses sequences of MapReduce jobs
  - Under the hood: entirely different from relational databases like MySQL
  - On surface: provides relational operators like SQL (JOIN, GROUP BY, etc.)

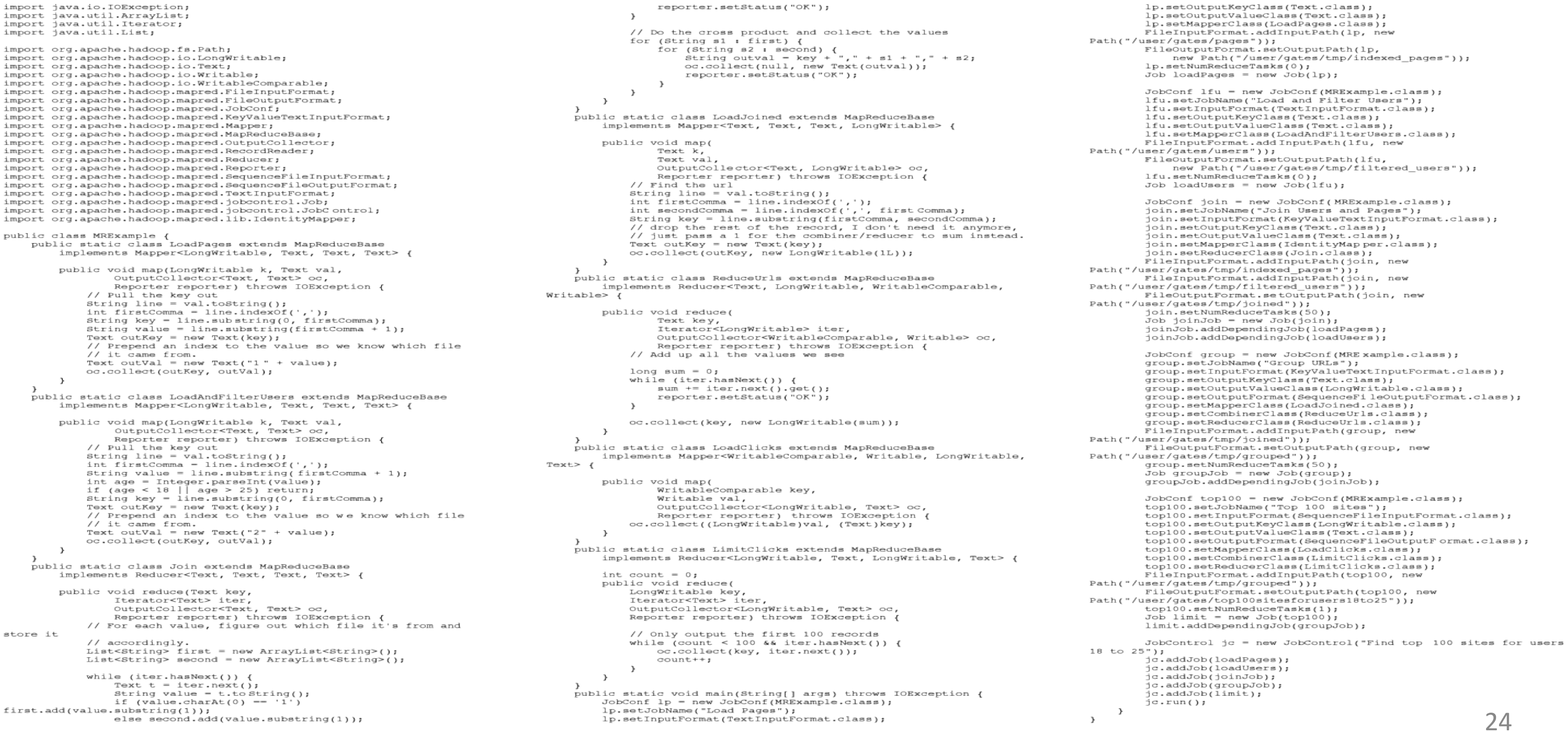
Pig example

Suppose you have user info in one file, website logs in another, and you need to find the top 5 pages most visited by users aged 18-25.



Example from http://wiki.apache.org/pig-data/attachments/PigTalksPapers/attachments/ApacheConEurope09.ppt

In MapReduce



```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.KeyValueTextInputFormat;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.RecordReader;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapred.SequenceFileInputFormat;
import org.apache.hadoop.mapred.SequenceFileOutputFormat;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.jobcontrol.Job;
import org.apache.hadoop.mapred.jobcontrol.JobC ontrol;
import org.apache.hadoop.mapred.lib.IdentityMapper;

public class MRExample {
    public static class LoadPages extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, Text> {

        public void map(LongWritable k, Text val,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // Pull the key out
            String line = val.toString();
            int firstComma = line.indexOf(',');
            String key = line.sub string(0, firstComma);
            String value = line.substring(firstComma + 1);
            Text outKey = new Text(key);
            // Prepend an index to the value so we know which file
            // it came from.
            Text outVal = new Text("1 " + value);
            oc.collect(outKey, outVal);
        }
    }
    public static class LoadAndFilterUsers extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, Text> {

        public void map(LongWritable k, Text val,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // Pull the key out
            String line = val.toString();
            int firstComma = line.indexOf(',');
            String value = line.substring( firstComma + 1);
            int age = Integer.parseInt(value);
            if (age < 18 || age > 25) return;
            String key = line.substring(0, firstComma);
            Text outKey = new Text(key);
            // Prepend an index to the value so w e know which file
            // it came from.
            Text outVal = new Text("2" + value);
            oc.collect(outKey, outVal);
        }
    }
    public static class Join extends MapReduceBase
        implements Reducer<Text, Text, Text, Text> {

        public void reduce(Text key,
                Iterator<Text> iter,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // For each value, figure out which file it's from and
store it
            // accordingly.
            List<String> first = new ArrayList<String>();
            List<String> second = new ArrayList<String>();

            while (iter.hasNext()) {
                Text t = iter.next();
                String value = t.to String();
                if (value.charAt(0) == '1')
first.add(value.substring(1));
                else second.add(value.substring(1));
                reporter.setStatus("OK");
            }

            // Do the cross product and collect the values
            for (String s1 : first) {
                for (String s2 : second) {
                    String outval = key + "," + s1 + "," + s2;
                    oc.collect(null, new Text(outval));
                    reporter.setStatus("OK");
                }
            }
        }
    }
    public static class LoadJoined extends MapReduceBase
        implements Mapper<Text, Text, Text, LongWritable> {

        public void map(
                Text k,
                Text val,
                OutputCollector<Text, LongWritable> oc,
                Reporter reporter) throws IOException {
            // Find the url
            String line = val.toString();
            int firstComma = line.indexOf(',');
            int secondComma = line.indexOf(',', first Comma);
            String key = line.substring(firstComma, secondComma);
            // drop the rest of the record, I don't need it anymore,
            // just pass a 1 for the combiner/reducer to sum instead.
            Text outKey = new Text(key);
            oc.collect(outKey, new LongWritable(1L));
        }
    }
    public static class ReduceUrls extends MapReduceBase
        implements Reducer<Text, LongWritable, WritableComparable,
Writable> {

        public void reduce(
                Text key,
                Iterator<LongWritable> iter,
                OutputCollector<WritableComparable, Writable> oc,
                Reporter reporter) throws IOException {
            // Add up all the values we see

            long sum = 0;
            while (iter.hasNext()) {
                sum += iter.next().get();
                reporter.setStatus("OK");
            }

            oc.collect(key, new LongWritable(sum));
        }
    }
    public static class LoadClicks extends MapReduceBase
        implements Mapper<WritableComparable, Writable, LongWritable,
Text> {

        public void map(
                WritableComparable key,
                Writable val,
                OutputCollector<LongWritable, Text> oc,
                Reporter reporter) throws IOException {
            oc.collect((LongWritable)val, (Text)key);
        }
    }
    public static class LimitClicks extends MapReduceBase
        implements Reducer<LongWritable, Text, LongWritable, Text> {

        int count = 0;
        public void reduce(
            LongWritable key,
            Iterator<Text> iter,
            OutputCollector<LongWritable, Text> oc,
            Reporter reporter) throws IOException {

            // Only output the first 100 records
            while (count < 100 && iter.hasNext()) {
                oc.collect(key, iter.next());
                count++;
            }
        }
    }
    public static void main(String[] args) throws IOException {
        JobConf lp = new JobConf(MRExample.class);
        lp.setJobName("Load Pages");
        lp.setInputFormat(TextInputFormat.class);
        lp.setOutputKeyClass(Text.class);
        lp.setOutputValueClass(Text.class);
        lp.setMapperClass(LoadPages.class);
        FileInputFormat.addInputPath(lp, new
Path("/user/gates/pages"));
        FileOutputFormat.setOutputPath(lp,
            new Path("/user/gates/tmp/indexed_pages"));
        lp.setNumReduceTasks(0);
        Job loadPages = new Job(lp);

        JobConf lfu = new JobConf(MRExample.class);
        lfu.setJobName("Load and Filter Users");
        lfu.setInputFormat(TextInputFormat.class);
        lfu.setOutputKeyClass(Text.class);
        lfu.setOutputValueClass(Text.class);
        lfu.setMapperClass(LoadAndFilterUsers.class);
        FileInputFormat.add InputPath(lfu, new
Path("/user/gates/users"));
        FileOutputFormat.setOutputPath(lfu,
            new Path("/user/gates/tmp/filtered_users"));
        lfu.setNumReduceTasks(0);
        Job loadUsers = new Job(lfu);

        JobConf join = new JobConf( MRExample.class);
        join.setJobName("Join Users and Pages");
        join.setInputFormat(KeyValueTextInputFormat.class);
        join.setOutputKeyClass(Text.class);
        join.setOutputValueClass(Text.class);
        join.setMapperClass(IdentityMap per.class);
        join.setReducerClass(Join.class);
        FileInputFormat.addInputPath(join, new
Path("/user/gates/tmp/indexed_pages"));
        FileInputFormat.addInputPath(join, new
Path("/user/gates/tmp/filtered_users"));
        FileOutputFormat.se tOutputPath(join, new
Path("/user/gates/tmp/joined"));
        join.setNumReduceTasks(50);
        Job joinJob = new Job(join);
        joinJob.addDependingJob(loadPages);
        joinJob.addDependingJob(loadUsers);

        JobConf group = new JobConf(MRE xample.class);
        group.setJobName("Group URLs");
        group.setInputFormat(KeyValueTextInputFormat.class);
        group.setOutputKeyClass(Text.class);
        group.setOutputValueClass(LongWritable.class);
        group.setOutputFormat(SequenceFi leOutputFormat.class);
        group.setMapperClass(LoadJoined.class);
        group.setCombinerClass(ReduceUrls.class);
        group.setReducerClass(ReduceUrls.class);
        FileInputFormat.addInputPath(group, new
Path("/user/gates/tmp/joined"));
        FileOutputFormat.setOutputPath(group, new
Path("/user/gates/tmp/grouped"));
        group.setNumReduceTasks(50);
        Job groupJob = new Job(group);
        groupJob.addDependingJob(joinJob);

        JobConf top100 = new JobConf(MRExample.class);
        top100.setJobName("Top 100 sites");
        top100.setInputFormat(SequenceFileInputFormat.class);
        top100.setOutputKeyClass(LongWritable.class);
        top100.setOutputValueClass(Text.class);
        top100.setOutputFormat(SequenceFileOutputF ormat.class);
        top100.setMapperClass(LoadClicks.class);
        top100.setCombinerClass(LimitClicks.class);
        top100.setReducerClass(LimitClicks.class);
        FileInputFormat.addInputPath(top100, new
Path("/user/gates/tmp/grouped"));
        FileOutputFormat.setOutputPath(top100, new
Path("/user/gates/top100sitesforusers18to25"));
        top100.setNumReduceTasks(1);
        Job limit = new Job(top100);
        limit.addDependingJob(groupJob);

        JobControl jc = new JobControl("Find top 100 sites for users
18 to 25");
        jc.addJob(loadPages);
        jc.addJob(loadUsers);
        jc.addJob(joinJob);
        jc.addJob(groupJob);
        jc.addJob(limit);
        jc.run();
    }
}
```

In Pig

```
Users    = load ‘users’ as (name, age);
Filtered = filter Users by
                  age >= 18 and age <= 25;
Pages    = load ‘pages’ as (user, url);
Joined   = join Filtered by name, Pages by user;
Grouped  = group Joined by url;
Summed   = foreach Grouped generate group,
                   count(Joined) as clicks;
Sorted   = order Summed by clicks desc;
Top5     = limit Sorted 5;

store Top5 into ‘top5sites’;
```

“SQL”: Unix command line

```
cat users.txt \
| awk ‘$2 >= 18 && $2 <= 25’ \
| join -1 1 -2 1 pages.txt - \
| cut -f 4 \
| sort \
| uniq -c \
| sort -k 1,1 -n -r \
| head -n 5
```

Other data models: document model

- Document model

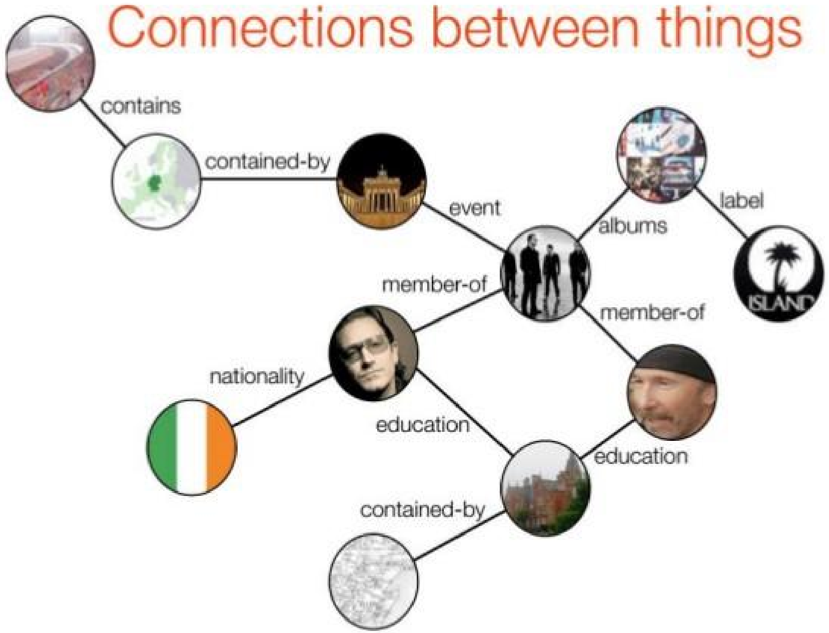
```xml
<contact>
  <id>656</id>
  <firstname>Chuck</firstname>
  <lastname>Smith</lastname>
  <phone>(123) 555-0178</phone>
  <phone>(890) 555-0133</phone>
  <address>
    <street1>Rue de l’Ale 8</street1>
    <city>Lausanne</city>
    <zip>1007</zip>
    <country>CH</country>
  </address>
</contact>
```

- Same in relational model

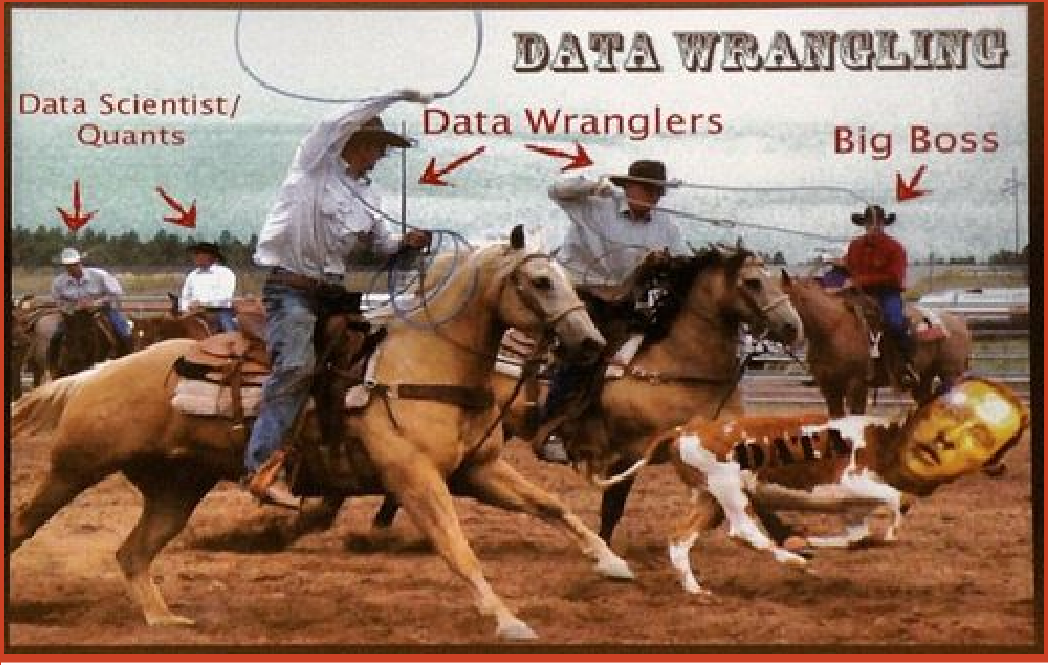
id	first name	...
656	Chuck	...
...	...	...

id	phone
656	(123) 555-0178
656	(890) 555-0133
...	...

Other data models: network model



Data Wrangling



Working with raw data sucks

Data comes in all shapes and sizes

– CSV files, PDFs, SQL dumps, .jpg, …

Different files have different formatting

– Spaces instead of NULLs, extra rows

“Dirty” data: Unwanted anomalies, duplicates

Raw data without thinking

==

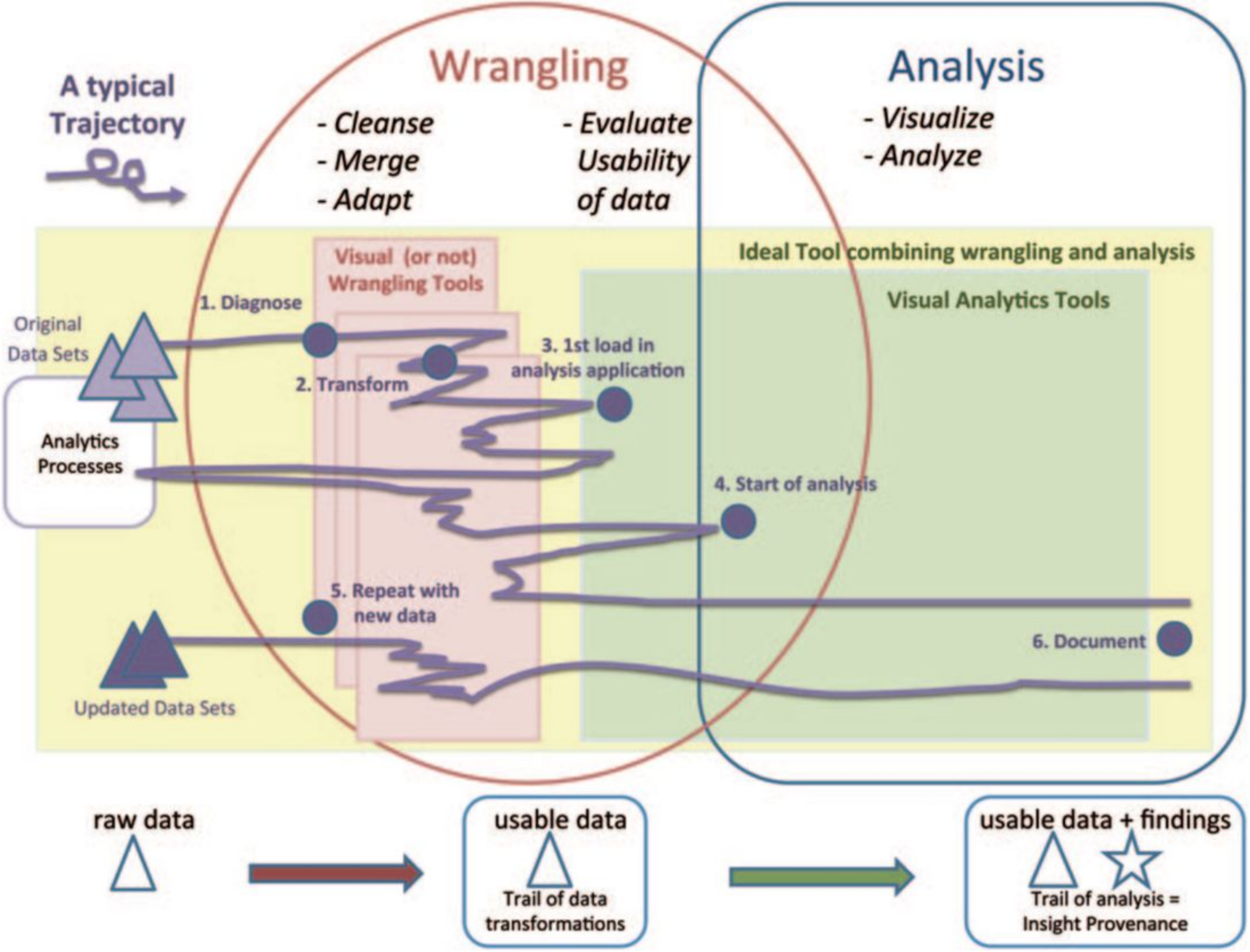
Recipe for disaster

What is data wrangling?

- a.k.a. **data munging**
- **Goal**: extract and standardize the raw data
  - Combine multiple data sources
  - Clean data anomalies
- **Strategy**: Combine automation with interactive visualizations to aid in cleaning
- **Outcome**: Improve efficiency and scale of data importing

Wrangling takes **between 50% and 80% of your time**…

[Source]

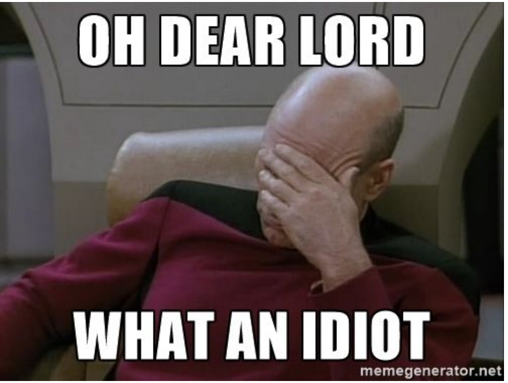


Types of data problems

- Missing data
- Incorrect data
- Inconsistent representations of the same data
- About 75% of data problems require human intervention (e.g., crowdsourcing, experts, etc.)
- Tradeoff between cleaning data vs. over-sanitizing data

link

“Dirty Data” horror stories



“Dear Idiot” letter

17,000 men are pregnant

As the crow flies

CHF 10,000 compute-cluster bill

[Source]

Diagnosing data problems

Visualizations and basic stats can convey issues in “raw” data

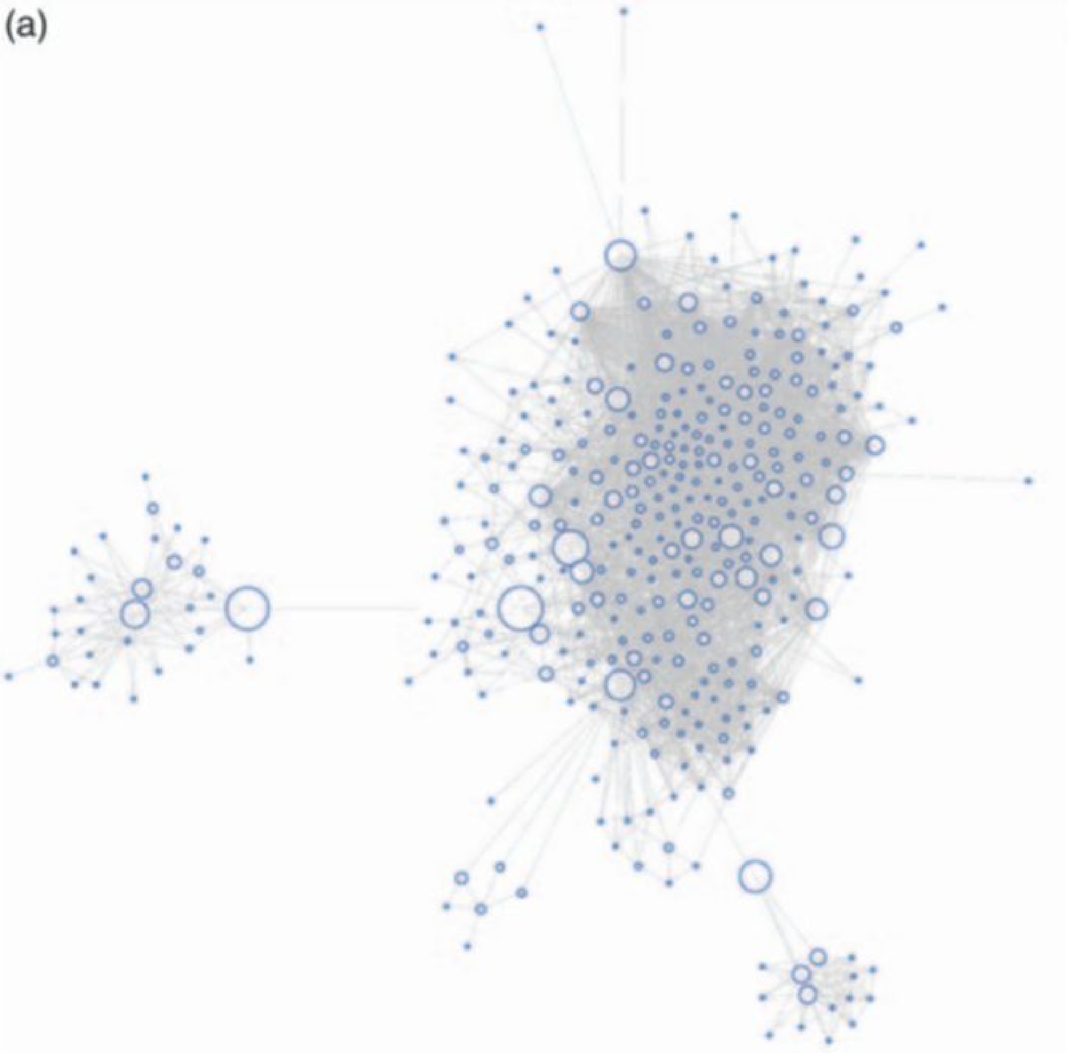
Different representations highlight different types of issues:

– Outliers often stand out in a plot

– Missing data will cause gaps or zero values

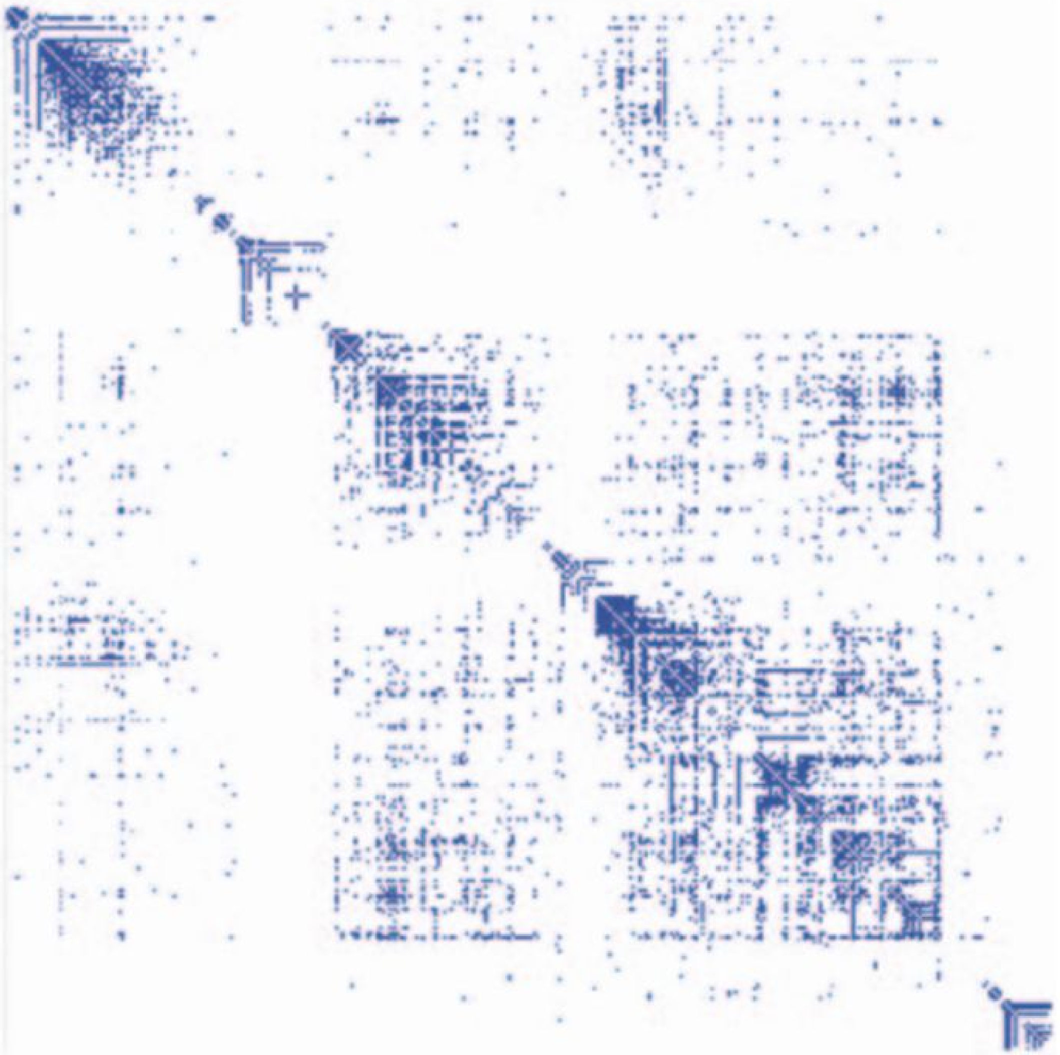
Becomes increasingly difficult as data gets larger
(sampling to the rescue!)

Facebook graph



Matrix view (1)

automatic permutation of rows and columns to highlight patterns of connectivity

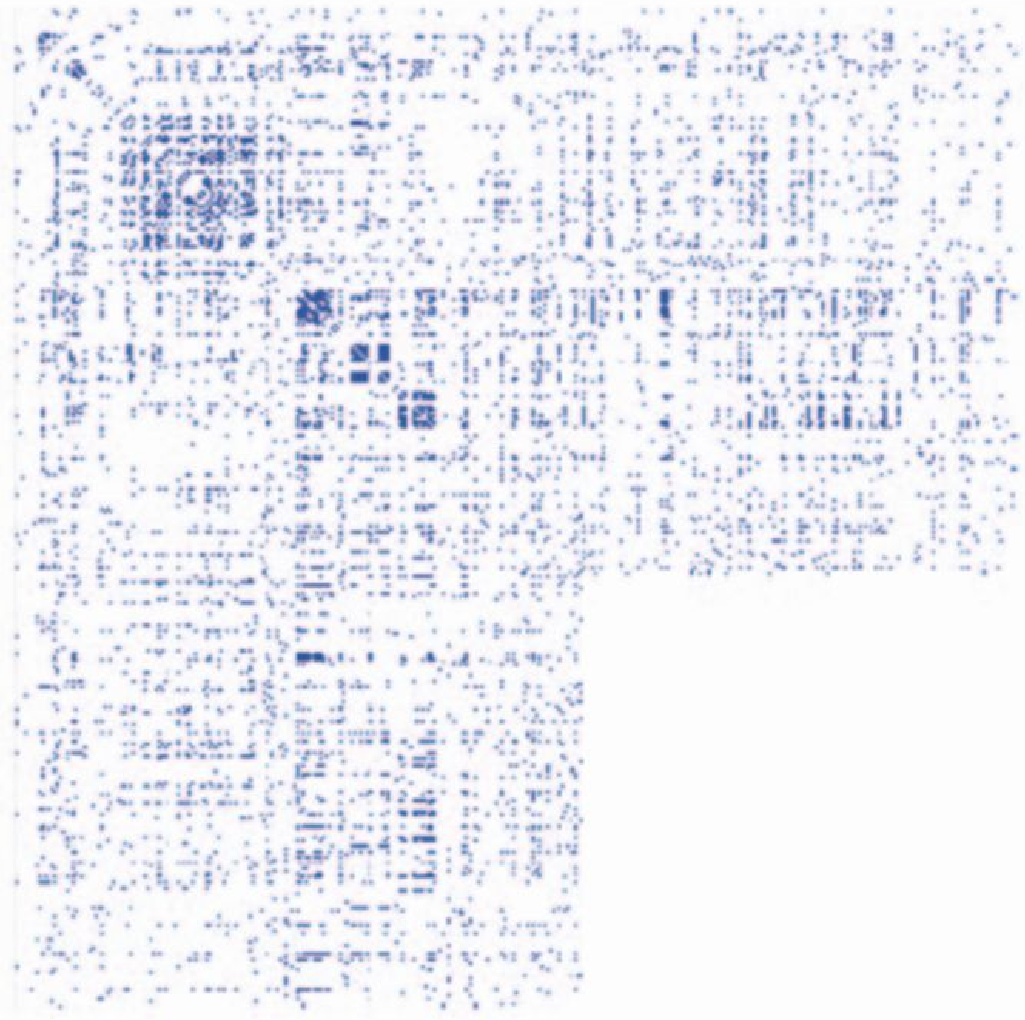


Matrix view (2)

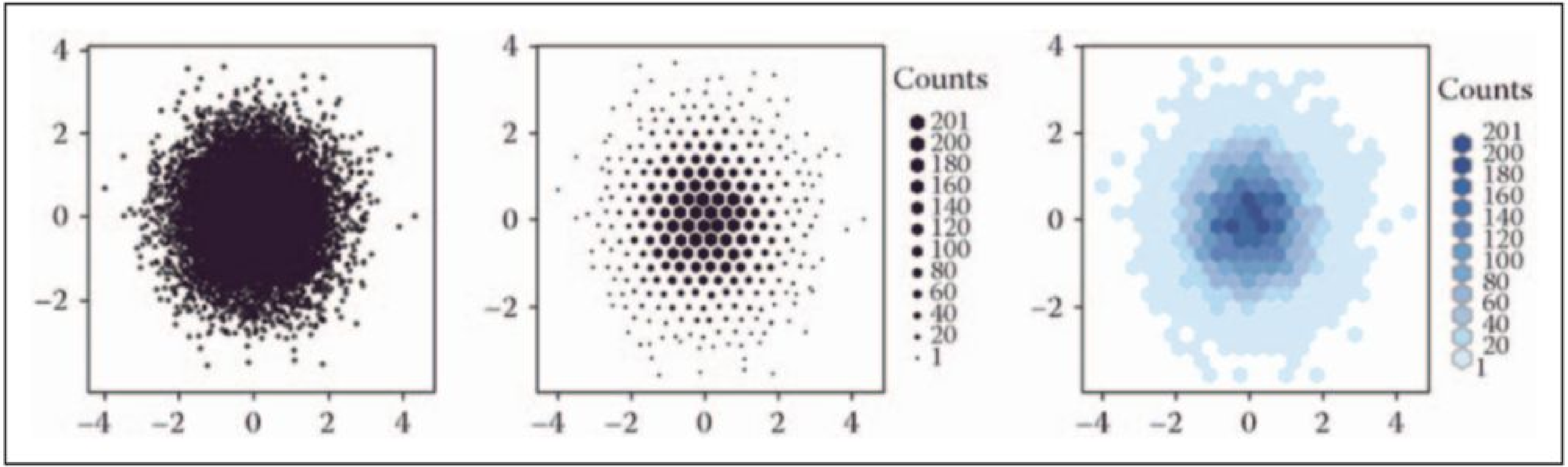
rows and columns sorted in the order provided by the Facebook API

Can you guess what’s going on?

[Source]



Viz at scale? Careful!

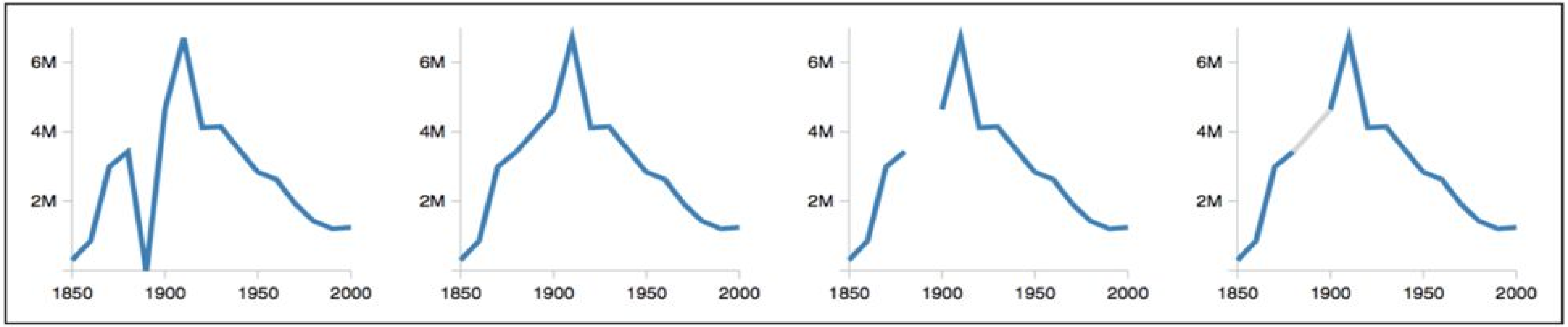


Dealing with missing data

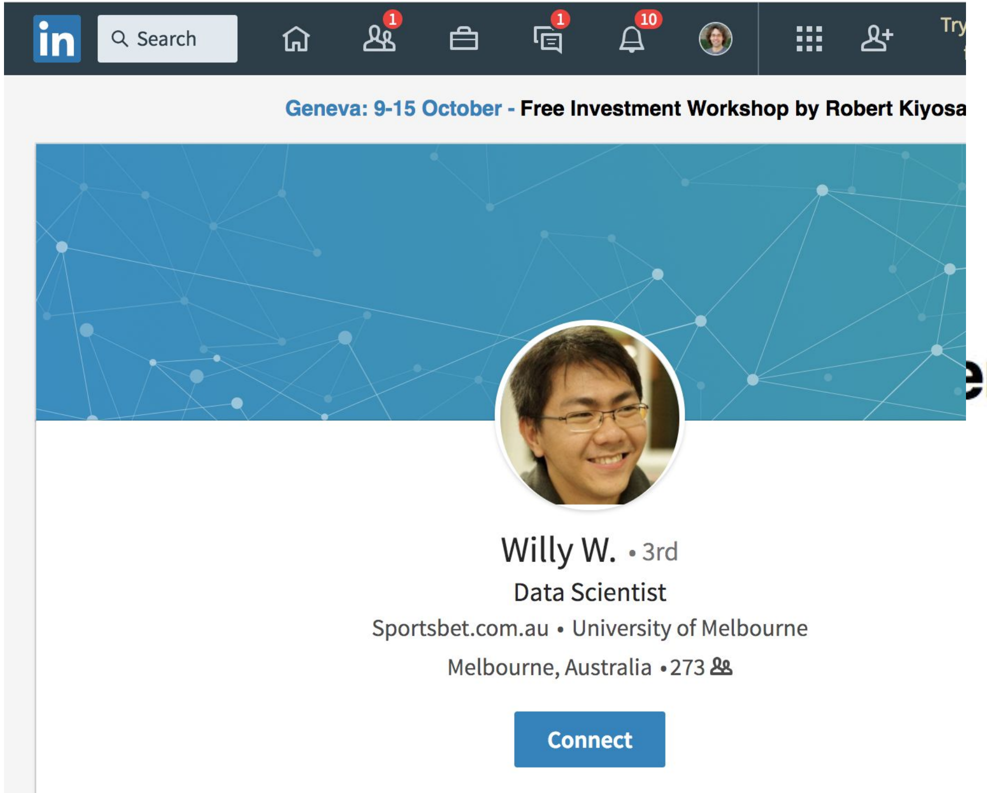
Knowledge about domain and data collection should drive your choice!

- Set values to zero?
- Interpolate based on existing data?

U.S. census counts of people working as ‘‘Farm Laborers’’; values from 1890 are **missing due to records being burned in a fire**



“My name is Willy”



Geneva: 9-15 October - Free Investment Workshop by Robert Kiyosa

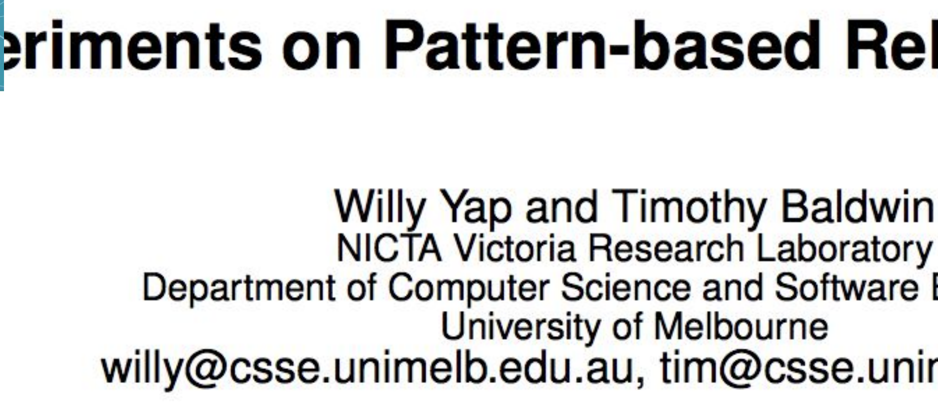
Willy W. • 3rd

Data Scientist

Sportsbet.com.au • University of Melbourne

Melbourne, Australia • 273

First name	Last name
Willy	NULL
…	…



eriments on Pattern-based Rel

Willy Yap and Timothy Baldwin
NICTA Victoria Research Laboratory
Department of Computer Science and Software E
University of Melbourne
willy@csse.unimelb.edu.au, tim@csse.unir

Data preparation

What to do before analysis

Deal with **uncertain data** (can arise from measurement errors, wrong sampling strategies, etc.)

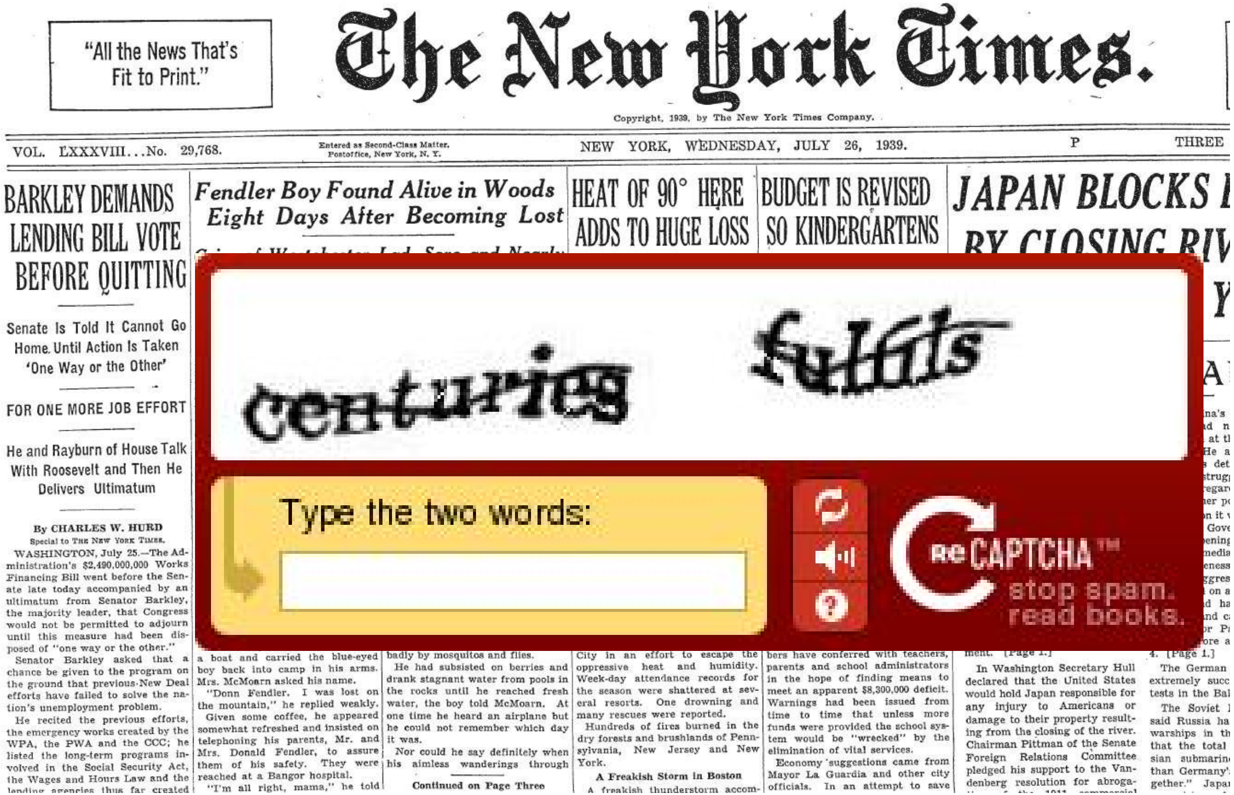
Parse/transform data (with the techniques we saw during the first hour) to obtain meaningful records

Desiderata

It’s always ideal if you can put your hands on the **code/documentation about the dataset** you are analyzing (provenance)

It’s always ideal if the provided **data format is nicely parsable** (otherwise you need regexes, or maybe even pay humans)

Highly non-parseable data



"All the News That's Fit to Print."

The New York Times.

Copyright, 1939, by The New York Times Company.

VOL. LXXXVIII...No. 29,768. Entered as Second-Class Matter, Postoffice, New York, N. Y. NEW YORK, WEDNESDAY, JULY 26, 1939. P THREE

BARKLEY DEMANDS LENDING BILL VOTE BEFORE QUITTING

Senate Is Told It Cannot Go Home Until Action Is Taken 'One Way or the Other'

FOR ONE MORE JOB EFFORT

He and Rayburn of House Talk With Roosevelt and Then He Delivers Ultimatum

By CHARLES W. HURD

Special to THE NEW YORK TIMES.

WASHINGTON, July 25.—The Administration's $2,490,000,000 Works Financing Bill went before the Senate late today accompanied by an ultimatum from Senator Barkley, the majority leader, that Congress would not be permitted to adjourn until this measure had been disposed of "one way or the other."

Senator Barkley asked that a chance be given to the program on the ground that previous New Deal efforts have failed to solve the nation's unemployment problem.

He recited the previous efforts, the emergency works created by the WPA, the PWA and the CCC; he listed the long-term programs involved in the Social Security Act, the Wages and Hours Law and the

Fendler Boy Found Alive in Woods Eight Days After Becoming Lost

a boat and carried the blue-eyed boy back into camp in his arms. Mrs. McMoarn asked his name.

"Donn Fendler. I was lost on the mountain," he replied weakly.

Given some coffee, he appeared somewhat refreshed and insisted on telephoning his parents, Mr. and Mrs. Donald Fendler, to assure them of his safety. They were reached at a Bangor hospital.

"I'm all right, mama," he told

badly by mosquitos and flies.

He had subsisted on berries and drank stagnant water from pools in the rocks until he reached fresh water, the boy told McMoarn. At one time he heard an airplane but he could not remember which day it was.

Nor could he say definitely when his aimless wanderings through

Continued on Page Three

HEAT OF 90° HERE ADDS TO HUGE LOSS

City in an effort to escape the oppressive heat and humidity. Week-day attendance records for the season were shattered at several resorts. One drowning and many rescues were reported.

Hundreds of fires burned in the dry forests and brushlands of Pennsylvania, New Jersey and New York.

A Freakish Storm in Boston

BUDGET IS REVISED SO KINDERGARTENS

bers have conferred with teachers, parents and school administrators in the hope of finding means to meet an apparent $8,300,000 deficit. Warnings had been issued from time to time that unless more funds were provided the school system would be "wrecked" by the elimination of vital services.

Economy suggestions came from Mayor La Guardia and other city officials. In an attempt to save

JAPAN BLOCKS

ment. [Page 1.]

In Washington Secretary Hull declared that the United States would hold Japan responsible for any injury to Americans or damage to their property resulting from the closing of the river. Chairman Pittman of the Senate Foreign Relations Committee pledged his support to the Vandenberg resolution for abroga-

4. [Page 1.]

- Entire NY Times archive (since 1851) digitized as of 2015

What’s next?

What we have seen today is definitely **not an exhaustive list** (when you get stuck, Google is your friend!)

E.g., when we move to machine learning, we will learn **how to prepare features** (i.e., attributes) with normalization, rescaling, etc.

Don’t be surprised when multiple iterations are required!

Credits

- Last year's version of these slides