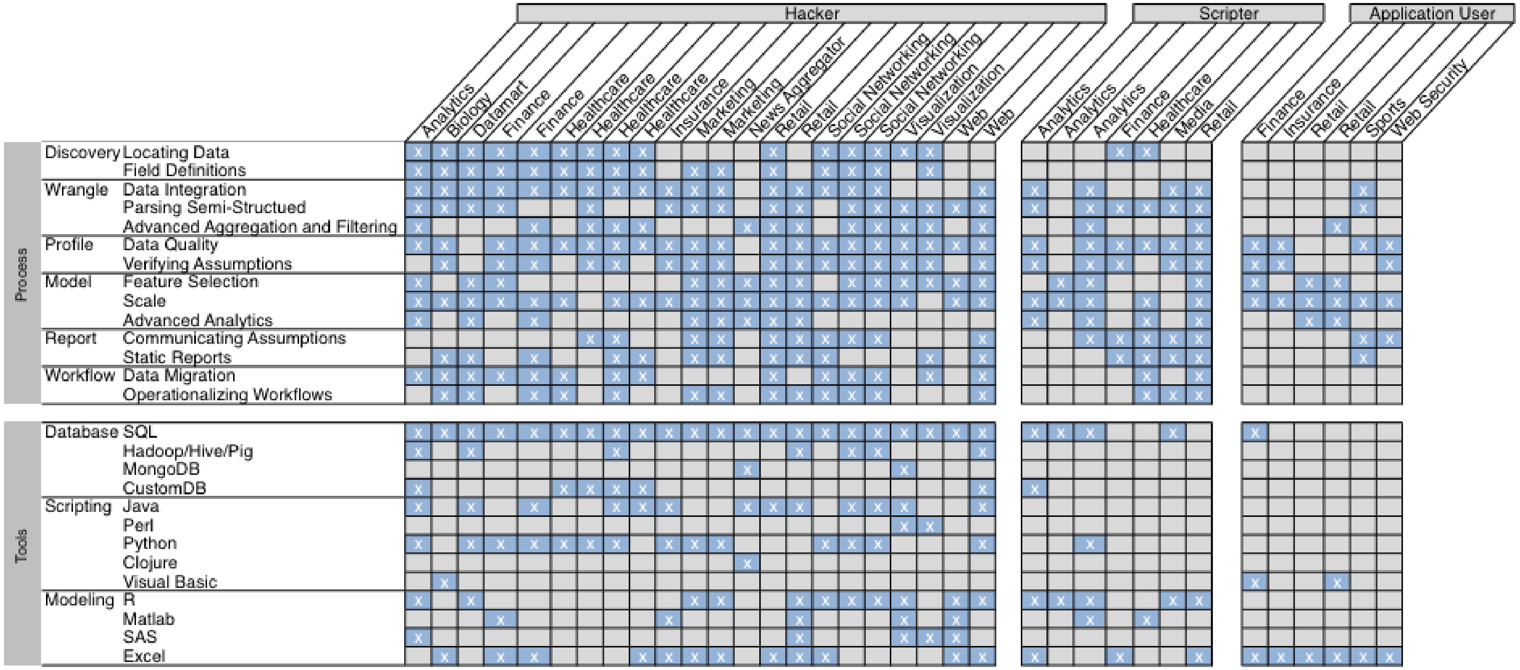
Introduction to Data Science
Lecture 2
Data Collection and Exploration

Yesi N. Kunang

Outline

- Data Types and Sources
- Data Preparation
- Exploration

Analyzing the Analysts

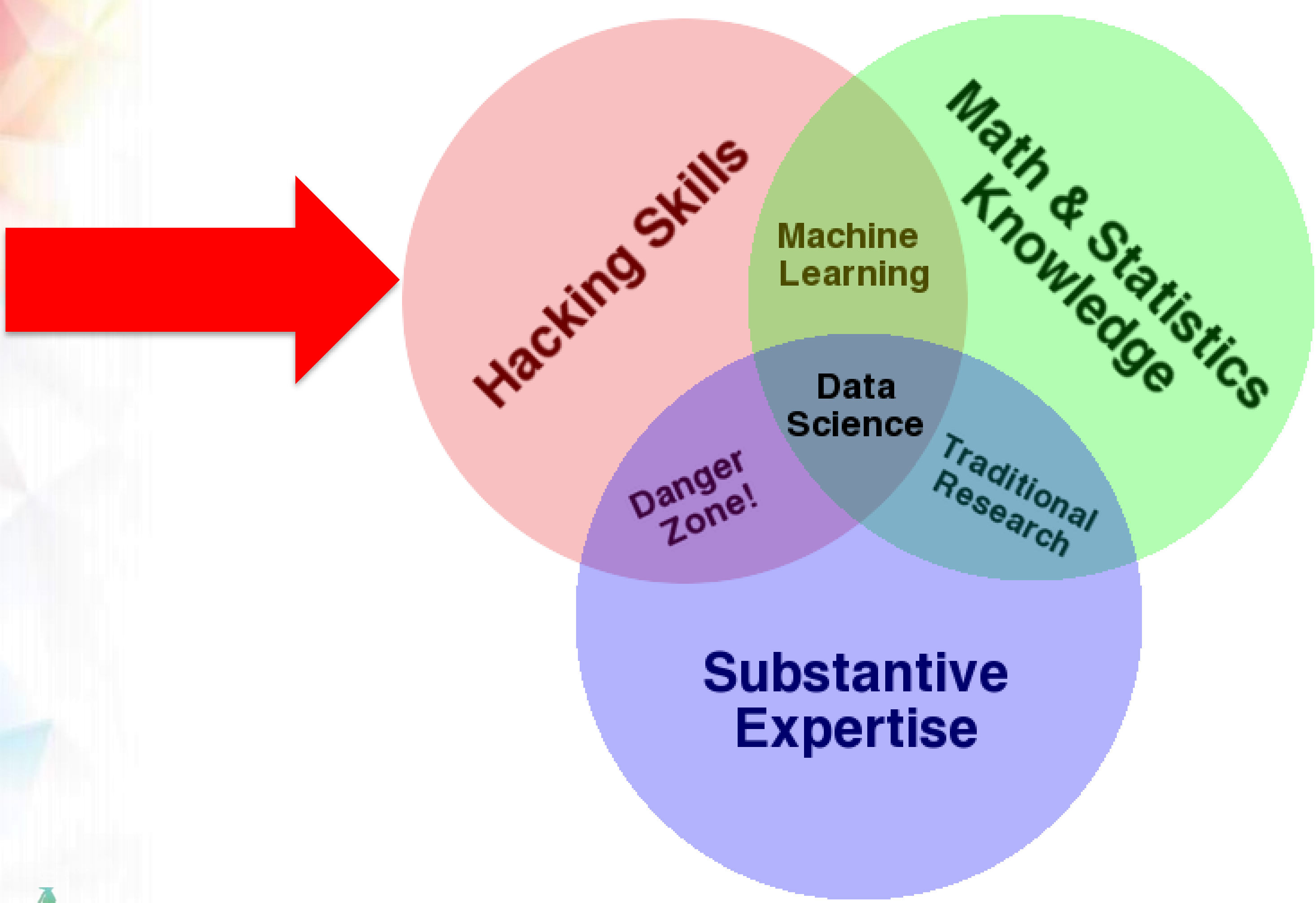


			Hacker																						Scripter							Application User					
			Analytics	Biology	Datamart	Finance	Finance	Healthcare	Healthcare	Healthcare	Healthcare	Insurance	Marketing	Marketing	News Aggregator	Retail	Retail	Social Networking	Social Networking	Social Networking	Visualization	Visualization	Web	Web	Analytics	Analytics	Analytics	Finance	Healthcare	Media	Retail	Finance	Insurance	Retail	Retail	Sports	Web Security
Process	Discovery	Locating Data	X	X	X	X	X	X	X	X	X					X		X	X	X	X	X						X	X								
		Field Definitions	X	X	X	X	X	X	X	X	X		X	X		X		X	X	X		X															
	Wrangle	Data Integration	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X				X	X		X			X	X					X	
		Parsing Semi-Structued	X	X	X	X			X			X	X	X		X	X		X	X	X	X	X	X	X		X	X	X	X	X					X	
		Advanced Aggregation and Filtering	X				X		X	X	X				X	X	X		X	X	X	X		X			X				X				X		
	Profile	Data Quality	X	X		X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X			X	X
		Verifying Assumptions		X		X	X		X	X		X	X	X		X	X	X	X	X	X	X		X	X		X	X		X	X	X	X				X
	Model	Feature Selection	X		X	X							X	X	X	X	X		X	X	X	X	X	X		X	X				X	X		X	X		
		Scale	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X		X		X	X	X	X	X	X	X
		Advanced Analytics	X		X		X						X	X	X	X	X								X		X		X		X			X	X		
	Report	Communicating Assumptions							X	X			X	X		X	X	X	X	X				X			X	X	X	X	X					X	X
		Static Reports		X	X		X			X	X		X	X		X	X	X				X		X				X	X	X	X					X	
	Workflow	Data Migration	X	X	X	X	X	X		X	X					X		X	X	X		X		X					X		X						
		Operationalizing Workflows		X	X		X	X		X			X	X		X	X	X	X	X				X					X	X	X						
Tools	Database	SQL	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			X		X					
		Hadoop/Hive/Pig	X		X					X							X		X	X				X													
		MongoDB													X						X																
		CustomDB	X					X	X	X	X													X	X												
	Scripting	Java	X		X		X			X	X	X			X	X	X		X	X	X			X													
		Perl																			X	X															
		Python	X		X	X	X	X	X	X		X	X	X				X	X	X				X			X										
		Clojure													X																						
		Visual Basic		X																												X			X		
	Modeling	R	X		X								X	X			X	X	X	X	X		X	X	X	X	X			X	X						
		Matlab				X						X					X				X		X				X		X								
		SAS	X														X				X	X	X														
		Excel		X		X	X				X	X	X	X		X	X	X					X	X	X			X			X	X	X	X	X	X	X

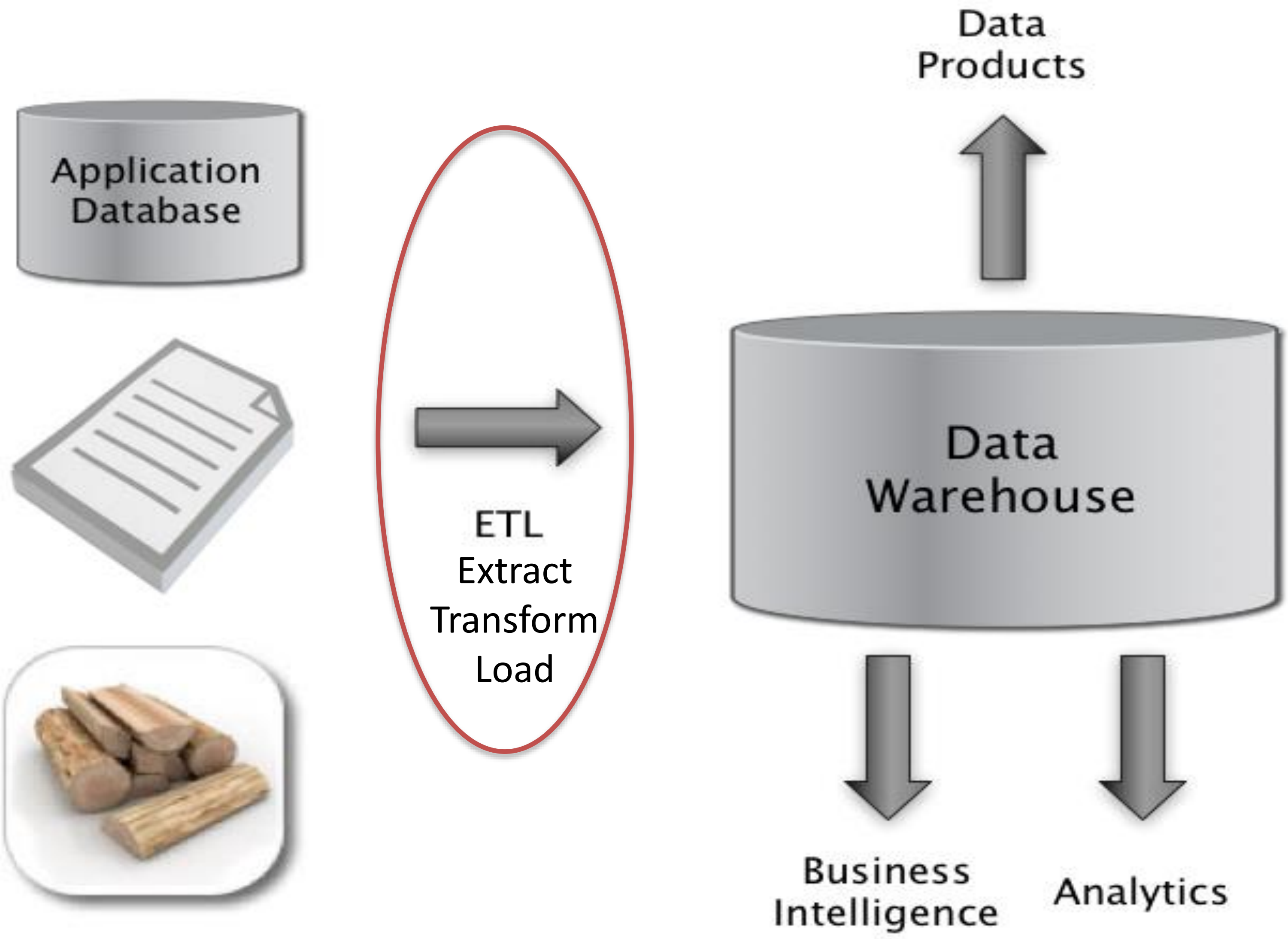
Fig. 1. Respondents, Challenges and Tools. The matrix displays interviewees (grouped by archetype and sector) and their corresponding challenges and tools. *Hackers* faced the most diverse set of challenges, corresponding to the diversity of their workflows and toolset. *Application users* and *scripters* typically relied on the IT team to perform certain tasks and therefore did not perceive them as challenges.

From Kandel, Paepcke, Hellerstein and Heer, “Enterprise Data Analysts and Visualization: An Interview Study”, IEEE VAST 2012

Data Science – One Definition



The Big Picture



Application Database

ETL
Extract
Transform
Load

Data Products

Data Warehouse

Business Intelligence

Analytics

Data Preparation overview

- ETL
  - We need to **extract** data from the **source(s)**
  - We need to **load** data into the **sink**
  - We need to **transform** data at the source, sink, or in a **staging area**

  - Sources: file, database, event log, web site, HDFS…
  - Sinks: Python, R, SQLite, RDBMS, NoSQL store, files, HDFS…

Data Sources at Web Companies

- Examples from Facebook
  - Application databases
  - Web server logs
  - Event logs
  - API server logs
  - Ad server logs
  - Search server logs
  - Advertisement landing page content
  - Wikipedia
  - Images and video

Structured Data

Semi-structured Data

Unstructured Data

The (changing) role of Schema

Schema specify the **structure** and **types** of a data repository, e.g. the types of each column in a table.

They may also specify constraints **within** or **between** data fields.

Traditional databases are **schema-on-write**. You cannot load data into a table without a schema.

Newer (noSQL) data stores are **schema-on-read** or **schemaless**: You can defer applying a schema until you read the data, or avoid schema altogether.

Schema-on-Write

SQL:

```
CREATE SCHEMA Sprockets
  CREATE TABLE NineProngs (source int, cost int, partnumber int)
GO


INSERT INTO NineProngs (source, cost, partnumber)
VALUES (5, 100, 45312453)
```

Schema-on-Read Data Types

XML: Generalizes HTML and specifies data **structure**. XML schema can be applied later to interpret XML data and specify **data types**. Here is some XML-encoded **data**:

```
<location>
  <latitude>37.78333</latitude>
  <longitude>122.4167</longitude>
</location>
```

When stored without a schema, the numerical data are stored as **strings**.

XML Schema

```
<location>
  <latitude>37.78333</latitude>
  <longitude>122.4167</longitude>
</location>
```

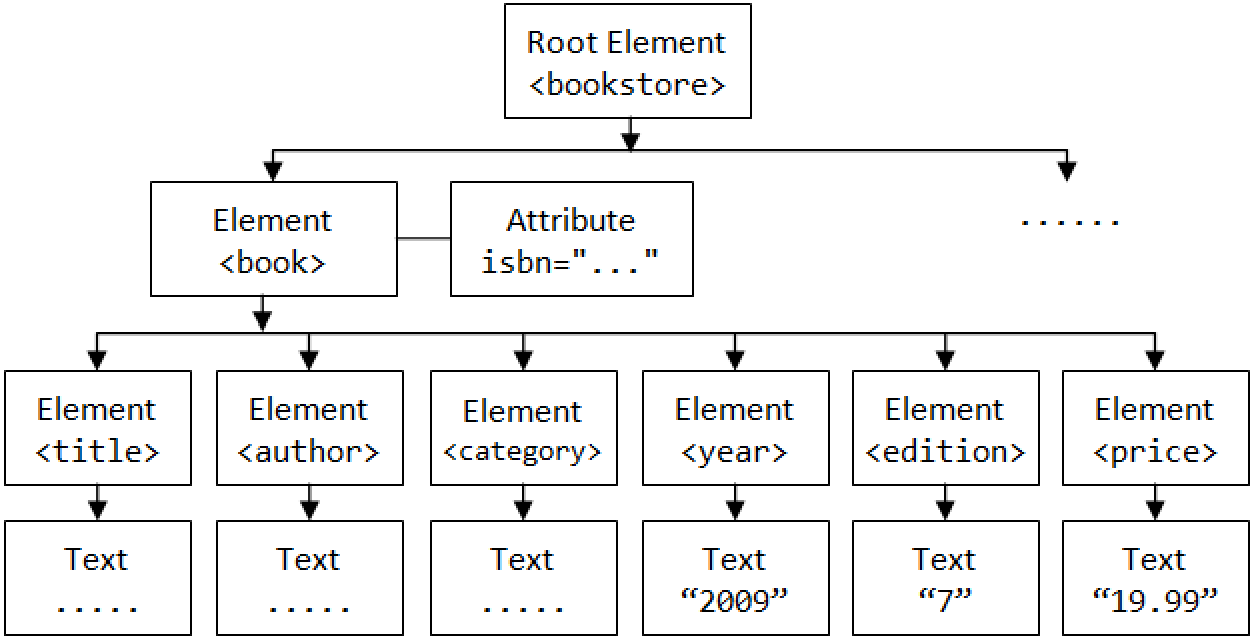
An XML schema for this element:

```
…
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="unqualified">
<xsd:complexType name="location">
  <xsd:sequence>
    <xsd:element name="latitude" type="xsd:decimal"/>
    <xsd:element name="longitude" type="xsd:decimal"/>
  </xsd:sequence>
</xsd:complexType name=“location">
```

XML and DOM

XML is a text format that encodes DOM (Document-Object Models) which is a data structure e.g. for Web pages.

The DOM is tree-structured:



XML Queries

XML schema allow a DB to interpret the data when running queries, e.g. to do **arithmetic** or **range queries** on numerical values.

XQuery is a standard for querying XML data with or without schema:

```
<places>{
for $city in /map/city
  let $latlong := $city/location
  where ((xs:float($latlong/latitude) < 39) and
         (xs:float($latlong/latitude) > 38))
  return
    <place name="{$city/name}“/>
}</places>
```

JSON

JSON (Javascript Object Notation) by contrast is a schemaless data description language (Schema support was added later):

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
       "streetAddress": "21 2nd Street",
       "city": "New York",
       "state": "NY",
       "postalCode": "10021-3100" },
  "phoneNumbers": [
      { "type": "home",
        "number": "212 555-1234" },
      { "type": "office",
        "number": "646 555-4567" } ],
  "children": [],
  "spouse": null
}
```

JSON

JSON is typically used to represent hierarchical data structures directly in the target language (Javascript or Java).

Transformations on the data are **procedural** in the target language (not declarative in a language as in Xquery).

Easier for some tasks, but painful for e.g. schema changes.

Data Tools

XML:

- Separation between schema and data.
- Data can be represented and stored without schema (as strings).
- More verbose (but not true after compression or in DB).
- Standard Query/Transformation languages XSLT and Xquery.

JSON:

- Types inferred inline. Schema rarely used but can be.
- Data without schema use type inference (string, int, float,…).
- More succinct in ASCII form.
- Transformation/ingestion rely on code (Java or Javascript).

Data Tools

XML:

- Mark Logic Server
  - XQuery-based, semi-structured data, late/early Schema use
  - Also many traditional DB features: transactions, journaling, fine-grained access control,…

JSON:

- MongoDB
  - JSON native, “schemaless”
  - Based on Open-source code

Log Files – Example Apache Web Log

Processes, usually daemons, create logs
e.g., `httpd, mysqld, syslogd`

-
  ```
  66.249.65.107 - - [08/Oct/2007:04:54:20 -0400] "GET /support.html HTTP/1.1" 200 11179 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"
  ```

-
  ```
  111.111.111.111 - - [08/Oct/2007:11:17:55 -0400] "GET / HTTP/1.1" 200 10801 "http://www.google.com/search?q=log+analyzer&ie=utf-8&oe=utf-8 &aq=t&rls=org.mozilla:en-US:official&client=firefox-a" "Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.8.1.7) Gecko/20070914 Firefox/2.0.0.7"
  ```

-
  ```
  111.111.111.111 - - [08/Oct/2007:11:17:55 -0400] "GET /style.css HTTP/1.1" 200 3225 “"http://www.loganalyzer.net/" "Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.8.1.7) Gecko/20070914 Firefox/2.0.0.7"
  ```

Tabular Data

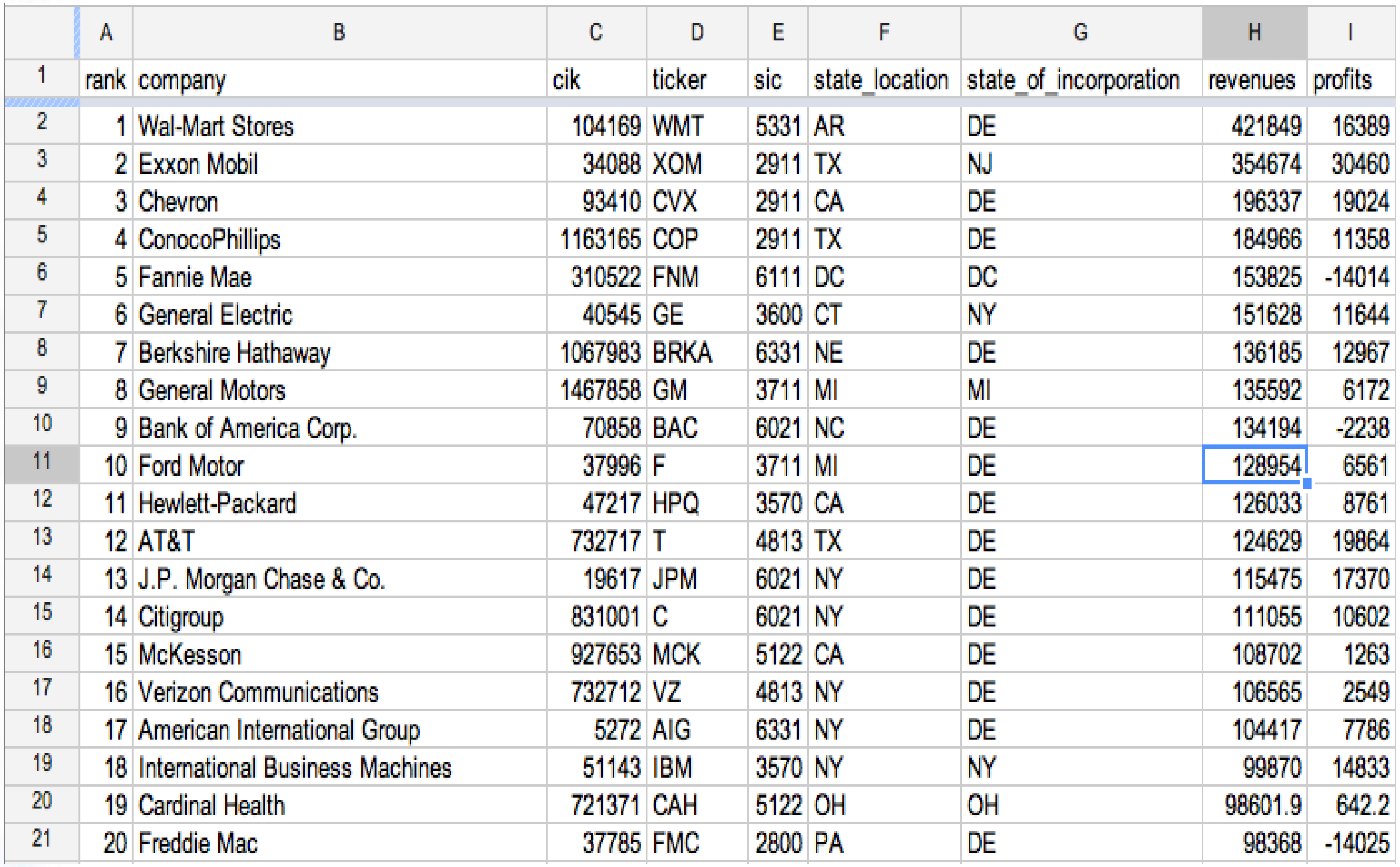
- What is a table?
  - A **table** is a collection of **rows** and **columns**
  - Each row has an **index**
  - Each column has a **name**
  - A **cell** is specified by an (index, name) pair
  - A cell may or may not have a **value**

- Schema = (minimally) column types.

- Often stored as text files in CSV or TSV format.

Tabular Data

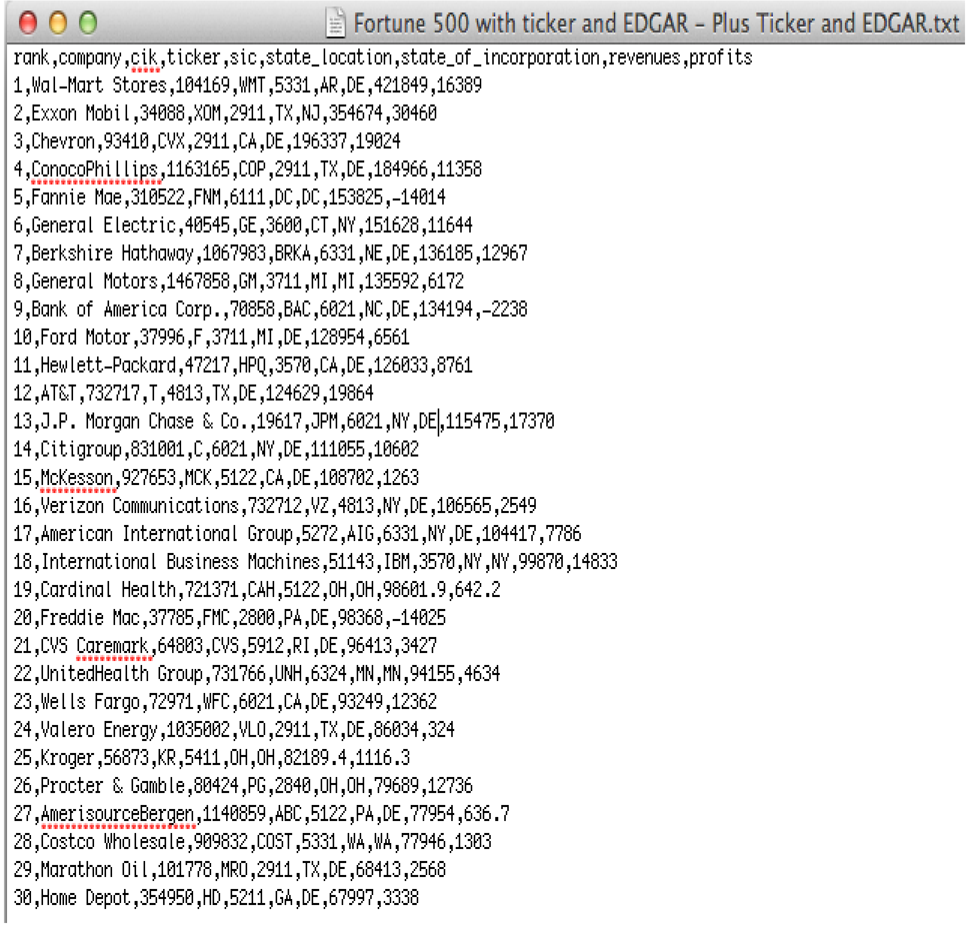
- Fortune 500



	A	B	C	D	E	F	G	H	I
1	rank	company	cik	ticker	sic	state_location	state_of_incorporation	revenues	profits
2	1	Wal-Mart Stores	104169	WMT	5331	AR	DE	421849	16389
3	2	Exxon Mobil	34088	XOM	2911	TX	NJ	354674	30460
4	3	Chevron	93410	CVX	2911	CA	DE	196337	19024
5	4	ConocoPhillips	1163165	COP	2911	TX	DE	184966	11358
6	5	Fannie Mae	310522	FNM	6111	DC	DC	153825	-14014
7	6	General Electric	40545	GE	3600	CT	NY	151628	11644
8	7	Berkshire Hathaway	1067983	BRKA	6331	NE	DE	136185	12967
9	8	General Motors	1467858	GM	3711	MI	MI	135592	6172
10	9	Bank of America Corp.	70858	BAC	6021	NC	DE	134194	-2238
11	10	Ford Motor	37996	F	3711	MI	DE	128954	6561
12	11	Hewlett-Packard	47217	HPQ	3570	CA	DE	126033	8761
13	12	AT&T	732717	T	4813	TX	DE	124629	19864
14	13	J.P. Morgan Chase & Co.	19617	JPM	6021	NY	DE	115475	17370
15	14	Citigroup	831001	C	6021	NY	DE	111055	10602
16	15	McKesson	927653	MCK	5122	CA	DE	108702	1263
17	16	Verizon Communications	732712	VZ	4813	NY	DE	106565	2549
18	17	American International Group	5272	AIG	6331	NY	DE	104417	7786
19	18	International Business Machines	51143	IBM	3570	NY	NY	99870	14833
20	19	Cardinal Health	721371	CAH	5122	OH	OH	98601.9	642.2
21	20	Freddie Mac	37785	FMC	2800	PA	DE	98368	-14025

Tabular Data (csv)

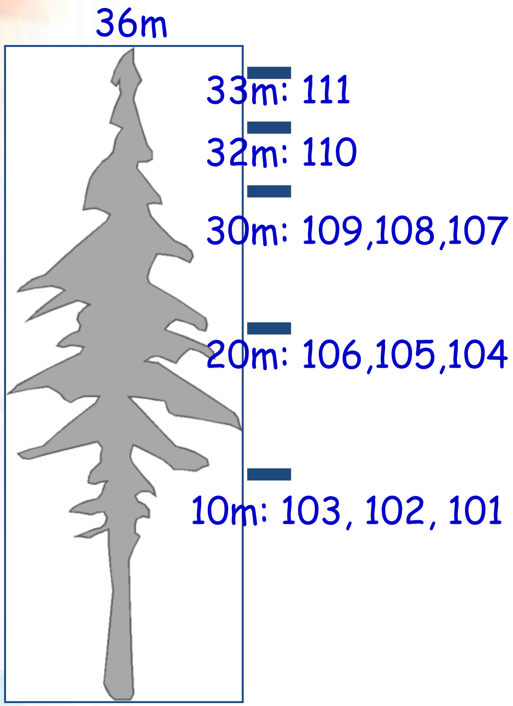
- Fortune 500

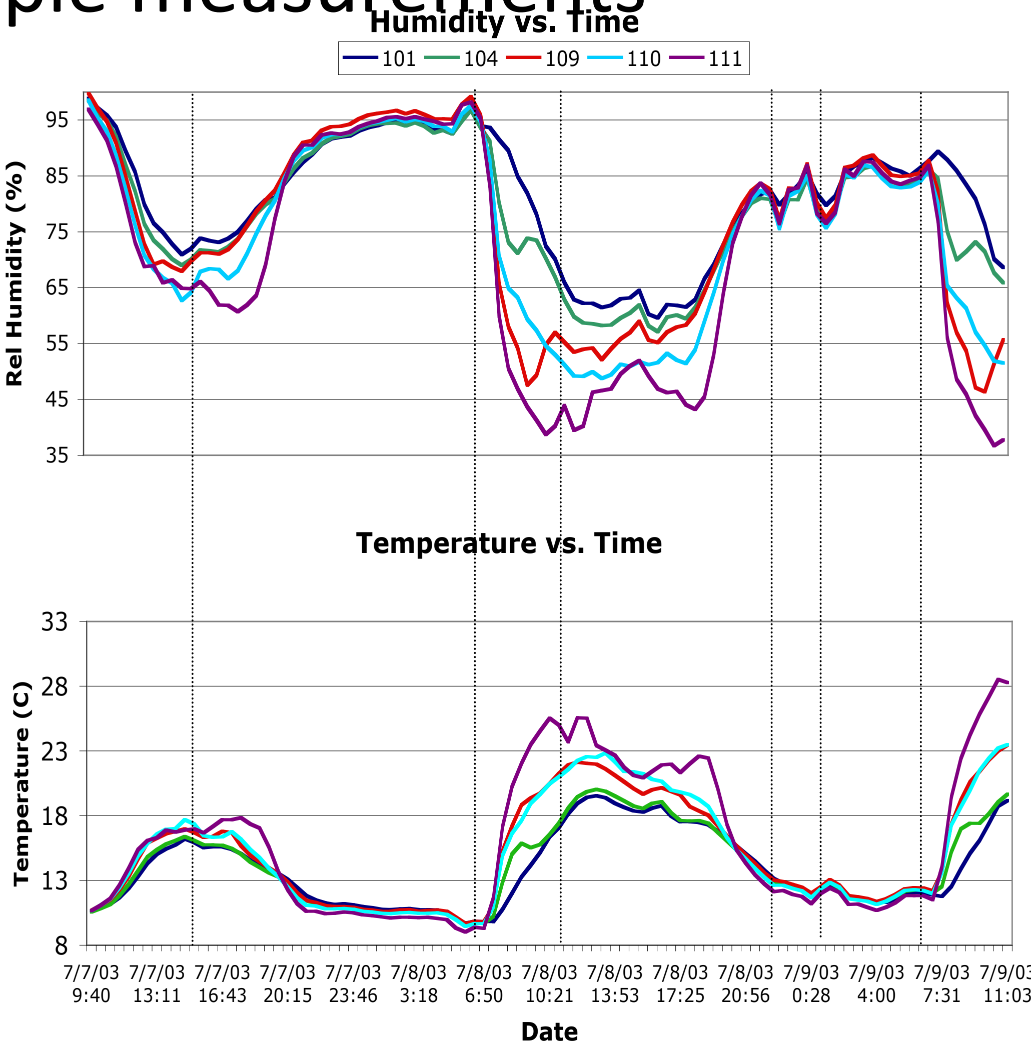


Fortune 500 with ticker and EDGAR – Plus Ticker and EDGAR.txt

```
rank,company,cik,ticker,sic,state_location,state_of_incorporation,revenues,profits
1,Wal-Mart Stores,104169,WMT,5331,AR,DE,421849,16389
2,Exxon Mobil,34088,XOM,2911,TX,NJ,354674,30460
3,Chevron,93410,CVX,2911,CA,DE,196337,19024
4,ConocoPhillips,1163165,COP,2911,TX,DE,184966,11358
5,Fannie Mae,310522,FNM,6111,DC,DC,153825,-14014
6,General Electric,40545,GE,3600,CT,NY,151628,11644
7,Berkshire Hathaway,1067983,BRKA,6331,NE,DE,136185,12967
8,General Motors,1467858,GM,3711,MI,MI,135592,6172
9,Bank of America Corp.,70858,BAC,6021,NC,DE,134194,-2238
10,Ford Motor,37996,F,3711,MI,DE,128954,6561
11,Hewlett-Packard,47217,HPQ,3570,CA,DE,126033,8761
12,AT&T,732717,T,4813,TX,DE,124629,19864
13,J.P. Morgan Chase & Co.,19617,JPM,6021,NY,DE,115475,17370
14,Citigroup,831001,C,6021,NY,DE,111055,10602
15,McKesson,927653,MCK,5122,CA,DE,108702,1263
16,Verizon Communications,732712,VZ,4813,NY,DE,106565,2549
17,American International Group,5272,AIG,6331,NY,DE,104417,7786
18,International Business Machines,51143,IBM,3570,NY,NY,99870,14833
19,Cardinal Health,721371,CAH,5122,OH,OH,98601.9,642.2
20,Freddie Mac,37785,FMC,2800,PA,DE,98368,-14025
21,CVS Caremark,64803,CVS,5912,RI,DE,96413,3427
22,UnitedHealth Group,731766,UNH,6324,MN,MN,94155,4634
23,Wells Fargo,72971,WFC,6021,CA,DE,93249,12362
24,Valero Energy,1035002,VLO,2911,TX,DE,86034,324
25,Kroger,56873,KR,5411,OH,OH,82189.4,1116.3
26,Procter & Gamble,80424,PG,2840,OH,OH,79689,12736
27,AmerisourceBergen,1140859,ABC,5122,PA,DE,77954,636.7
28,Costco Wholesale,909832,COST,5331,WA,WA,77946,1303
29,Marathon Oil,101778,MRO,2911,TX,DE,68413,2568
30,Home Depot,354950,HD,5211,GA,DE,67997,3338
```

Internet of Things: Example measurements





Tabular Data from Sensors

Goals

- Want to support both long-term (trend) and short-term (real-time) queries.
- Want low latency but also efficient, real-time indexing for longer-term queries.
- Want triggers (alerts) for a variety of conditions.

Tabular Data from Sensors

Tools:

- Microsoft SQL server, Oracle

Analysis:

- Matlab still widely used for analysis in Financial services, Python tools.

Syslog – A Standard for System Messages

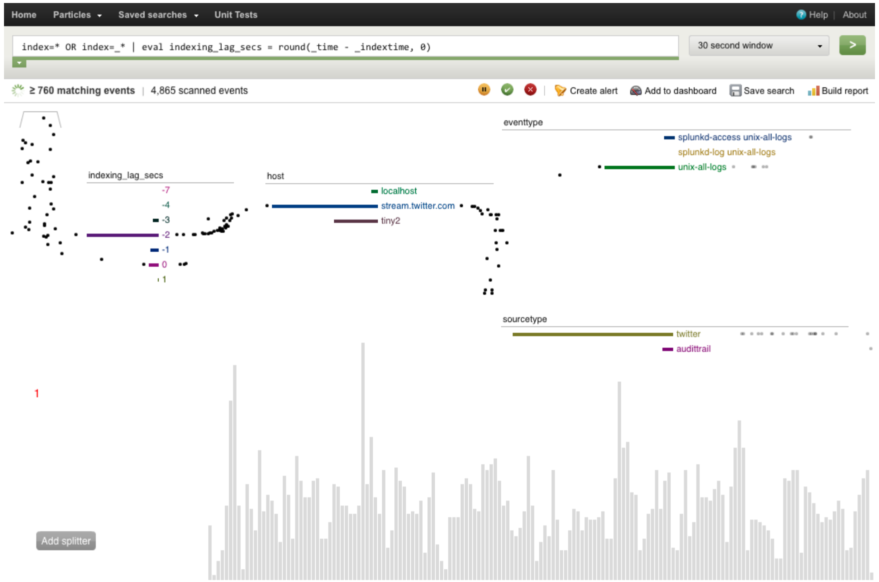
- Developed by Eric Allman (at Berkeley) as part of the Sendmail project
- Standardized by the IETF in RFC 3164 and RFC 5424
- Listens on port 514 using UDP
- Puts data in /var/log/messages by default
- Enables rich analysis: splunk>

Syslog

```
dhcp-47-129:DataScienceF15> syslog -w 10
Feb  3 15:18:11 dhcp-47-129 Evernote[1140] <Warning>: -[EDAMAccounting read:]: unexpected field ID 23 with type 8.  Skipping.
Feb  3 15:18:11 dhcp-47-129 Evernote[1140] <Warning>: -[EDAMUser read:]: unexpected field ID 17 with type 12.  Skipping.
Feb  3 15:18:11 dhcp-47-129 Evernote[1140] <Warning>: -[EDAMAuthenticationResult read:]: unexpected field ID 6 with type 11.  Skipping.
Feb  3 15:18:11 dhcp-47-129 Evernote[1140] <Warning>: -[EDAMAuthenticationResult read:]: unexpected field ID 7 with type 11.  Skipping.
Feb  3 15:18:11 dhcp-47-129 Evernote[1140] <Warning>: -[EDAMAccounting read:]: unexpected field ID 19 with type 8.  Skipping.
Feb  3 15:18:11 dhcp-47-129 Evernote[1140] <Warning>: -[EDAMAccounting read:]: unexpected field ID 23 with type 8.  Skipping.
Feb  3 15:18:11 dhcp-47-129 Evernote[1140] <Warning>: -[EDAMUser read:]: unexpected field ID 17 with type 12.  Skipping.
Feb  3 15:18:11 dhcp-47-129 Evernote[1140] <Warning>: -[EDAMSyncState read:]: unexpected field ID 5 with type 10.  Skipping.
Feb  3 15:18:49 dhcp-47-129 com.apple.mtmd[47] <Notice>: low priority thinning needed for volume Macintosh HD (/) with 18.9 <= 20.0 pct free space
```

“Splunking”

- Grab data from many machines
- Index it
- Check for unusual events:
  - Disk problems
  - Network congestion
  - Security attacks
- Monitor Resources
  - Network
  - Memory usage
  - Disk use, latency
  - Threads
- Dashboard for cloud administration.



```
index=* OR index=_* | eval indexing_lag_secs = round(_time - _indextime, 0)
```

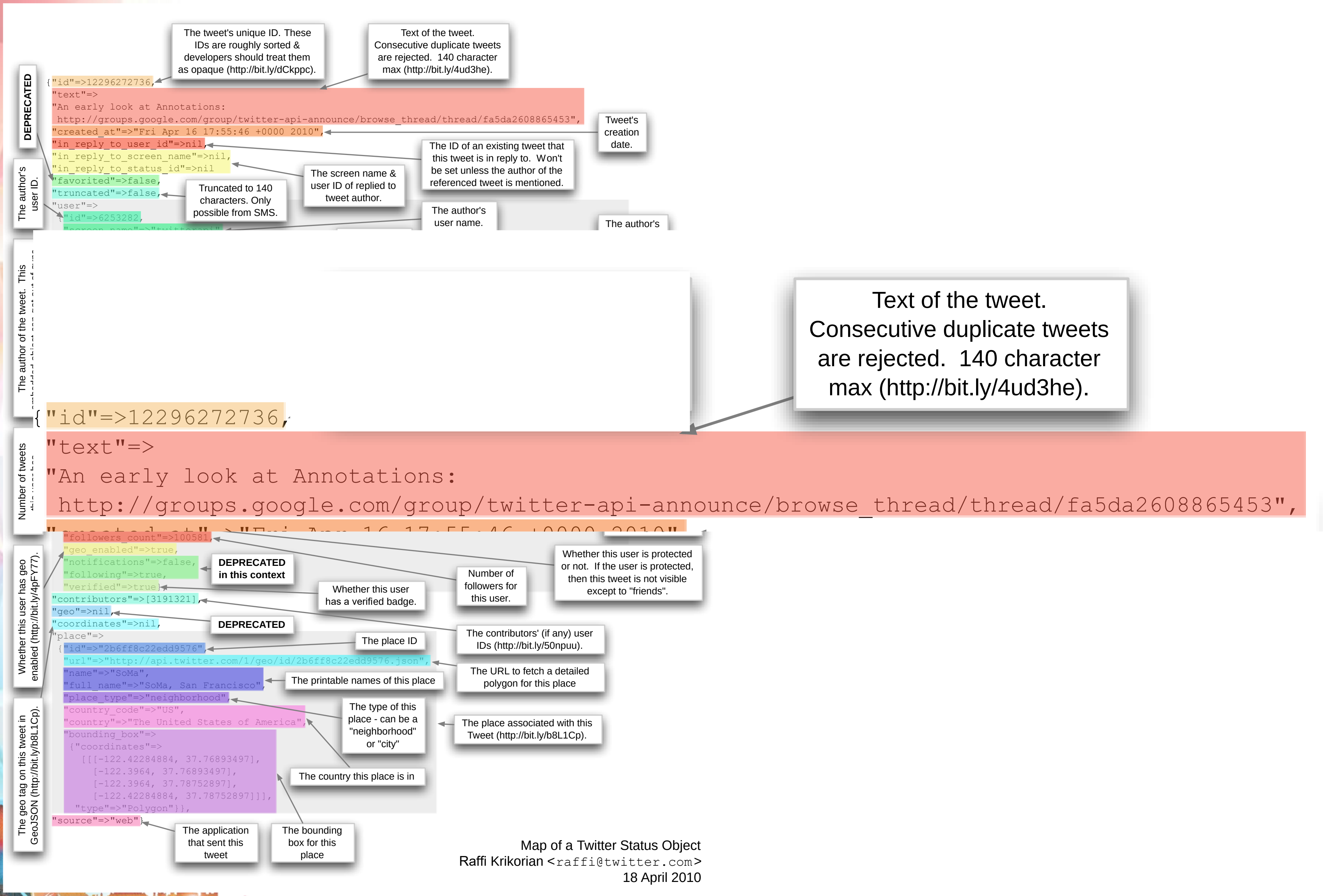
≥ 760 matching events | 4,865 scanned events

Some Questions

1) How Many Characters are there in a Tweet?

2) How Many Bytes are there in the API record for a Tweet?

Tweet JSON Format



```
{"id"=>12296272736,
"text"=>
"An early look at Annotations:
 http://groups.google.com/group/twitter-api-announce/browse_thread/thread/fa5da2608865453",
"created_at"=>"Fri Apr 16 17:55:46 +0000 2010",
"in_reply_to_user_id"=>nil,
"in_reply_to_screen_name"=>nil,
"in_reply_to_status_id"=>nil
"favorited"=>false,
"truncated"=>false,
"user"=>
 {"id"=>6253282,
```

```
  "followers_count"=>100581,
  "geo_enabled"=>true,
  "notifications"=>false,
  "following"=>true,
  "verified"=>true},
"contributors"=>[3191321],
"geo"=>nil,
"coordinates"=>nil,
"place"=>
 {"id"=>"2b6ff8c22edd9576",
  "url"=>"http://api.twitter.com/1/geo/id/2b6ff8c22edd9576.json",
  "name"=>"SoMa",
  "full_name"=>"SoMa, San Francisco",
  "place_type"=>"neighborhood",
  "country_code"=>"US",
  "country"=>"The United States of America",
  "bounding_box"=>
   {"coordinates"=>
     [[[-122.42284884, 37.76893497],
       [-122.3964, 37.76893497],
       [-122.3964, 37.78752897],
       [-122.42284884, 37.78752897]]],
    "type"=>"Polygon"}},
"source"=>"web"}
```

The tweet's unique ID. These IDs are roughly sorted & developers should treat them as opaque (http://bit.ly/dCkppc).

Text of the tweet. Consecutive duplicate tweets are rejected. 140 character max (http://bit.ly/4ud3he).

Tweet's creation date.

DEPRECATED

The ID of an existing tweet that this tweet is in reply to. Won't be set unless the author of the referenced tweet is mentioned.

The screen name & user ID of replied to tweet author.

Truncated to 140 characters. Only possible from SMS.

The author's user ID.

The author's user name.

The author's

The author of the tweet. This

Number of tweets

Whether this user is protected or not. If the user is protected, then this tweet is not visible except to "friends".

Number of followers for this user.

DEPRECATED in this context

Whether this user has a verified badge.

Whether this user has geo enabled (http://bit.ly/4pFY77).

The contributors' (if any) user IDs (http://bit.ly/50npuu).

DEPRECATED

The place ID

The URL to fetch a detailed polygon for this place

The printable names of this place

The type of this place - can be a "neighborhood" or "city"

The place associated with this Tweet (http://bit.ly/b8L1Cp).

The country this place is in

The geo tag on this tweet in GeoJSON (http://bit.ly/b8L1Cp).

The application that sent this tweet

The bounding box for this place

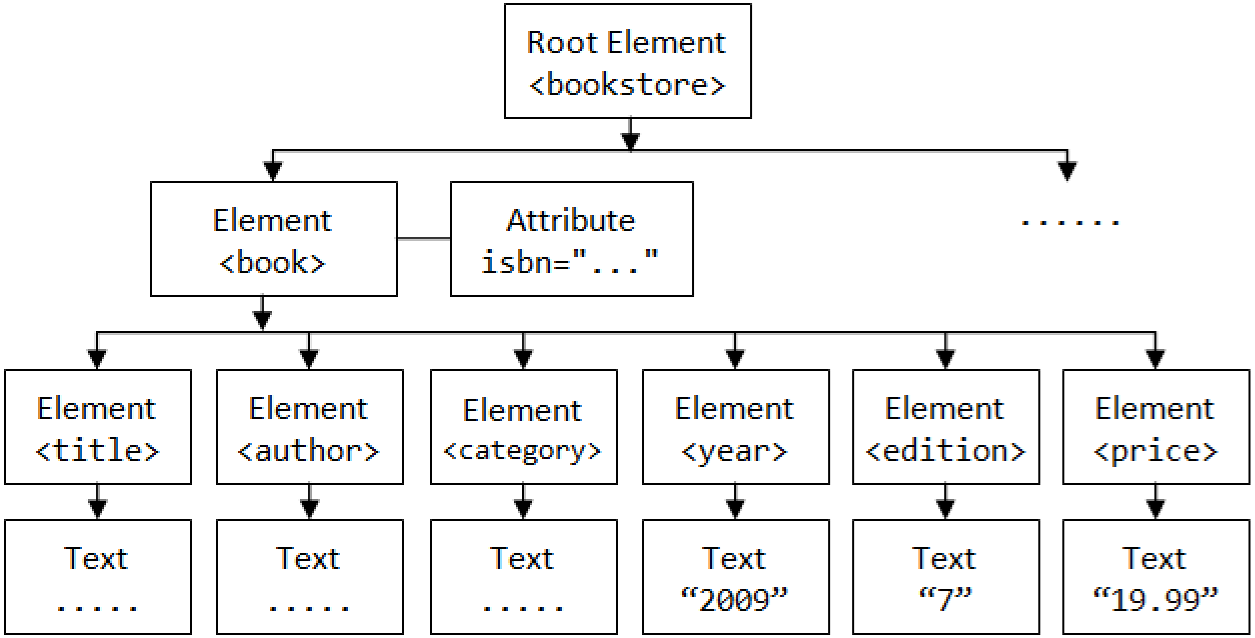
Text of the tweet. Consecutive duplicate tweets are rejected. 140 character max (http://bit.ly/4ud3he).

```
{"id"=>12296272736,
"text"=>
"An early look at Annotations:
 http://groups.google.com/group/twitter-api-announce/browse_thread/thread/fa5da2608865453",
```

Map of a Twitter Status Object
Raffi Krikorian <raffi@twitter.com>
18 April 2010

Processing XML and JSON

- The DOM is an easy object to work with: all the data in the object is accessible by links.
- The problem is that I might not care about most of the data, and I might not be able to fit the DOM for a large object in RAM.



Event-Driven Parsing: SAX

```
<?xml version="1.0" encoding="UTF-8"?> → Document Header
<!-- bookstore.xml --> → Comment
<bookstore> → Start-element “bookstore”
  <book ISBN="0123456001"> → Start-element “book”
     <title>Java For Dummies</title> → Start-element “title”
                                       End-element “title”
     <author>Tan Ah Teck</author>
     <category>Programming</category>
     <year>2009</year>
     <edition>7</edition>
     <price>19.99</price>
  </book> → End-element “book”
```

Event-Driven Parsing: SAX

A SAX parser finds all the open-close-tag events in an XML documents, and does callbacks to user code.

- User code can respond to only a subset of events corresponding to the tags it is interested in.
- User code can correctly compute aggregates from the data rather than create a record for each tag.
- User code must implement a state machine to keep track of “where it is” in the DOM tree.
- User code can implement flexible error recover strategies for ill-formed XML.

What about JSON?

Most JSON parsers construct the “DOM” directly.

But there are a few SAX-style parsers:

- Jackson
- JSON-simple

What about HTML?

- Common Crawl, about 5 billion web pages, between 0.2-0.5% of Google’s web crawl.
- 60 TB, hosted on Amazon S3, also available for download.
- Includes link data, page rank.
- In ARC (Internet Archive) File format.

- So there’s plenty of data, and there are many crawlers for targeted exploration…
  - HTTrack, …

HTML Tag Soup

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><!-- types/widgets/pages/common/page.tmpl home/index_v3.html generated by index_v3 on Wed 29
Feb 2012 11:04:41 PM PST -->
<title>San Francisco Bay Area &mdash; News, Sports, Business, Entertainment, Classifieds: SFGate</title>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
<meta name="description" content="Find local news &amp; information, updated weather, traffic, classifieds,
sports scores, real estate, jobs, cars, food &amp; wine, travel, entertainment, events and more on SFGate.com.
Connect to the Bay Area community." />
<meta name="keywords" content="San Francisco, San Francisco Bay Area, news, local events, breaking news,
world news, San Francisco Chronicle, SFGate" />
<meta property="fb:page_id" content="105702905593" />
<meta property="fb:admins" content="653226748,658759748" />
<!-- /widgets/sitewide/css/all/inc.html widgets/pages/common/post_write_mtime/css_inc.tmpl -->
<!-- generated by sitewidecss on Thu 16 Feb 2012 10:41:53 AM PST -->
<link rel="stylesheet" type="text/css" title="SFGate" media="all"
href="http://imgs.sfgate.com/css1329417713/sitewide/css/sitewide.css" />
<!-- sitewide/css/all/inc.html end css_inc.tmpl -->
```

HTML Tools - Parsing

- “Beautiful Soup”
  http://www.crummy.com/software/BeautifulSoup/
  a Python API for handling real HTML. DOM or SAX interfaces.

- “TagSoup”
  http://ccil.org/~cowan/XML/tagsoup/
  provides a Sax interface, i.e. a streaming parse, to Java applications. Can transform to a format you want using XSLT.

- Taggle, part of the Arabica toolset
  http://www.jezuk.co.uk/cgi-bin/view/arabica/code
  is a version of TagSoup written in C++. You may want to use this if you have a lot of data.

Web Services

Most large web sites today actively discourage screen-scraping to get their content, and provide Web Service APIs instead.

This is the “right” way to get data from online sources.

Web Services

W3C definition:

a "Web service" as "a software system designed to support interoperable machine-to-machine interaction over a network".

Two kinds:

- XML-based RPC-style messages: SOAP
- REST-style stateless interactions, URLs encode state

Can run over different transports, but usually HTTP

Examples

Twitter: REST API and streaming API with JSON content. Provides sampling, searching and filtering capabilities.

Amazon: has a “product advertising API” in XML with a WSDL spec. Includes product search, reviews etc.

Livejournal: RSS/Atom + custom XML/RPC. Search by keyword, topic, follow friend links.

Netflix: Javascript, Atom and REST interfaces.

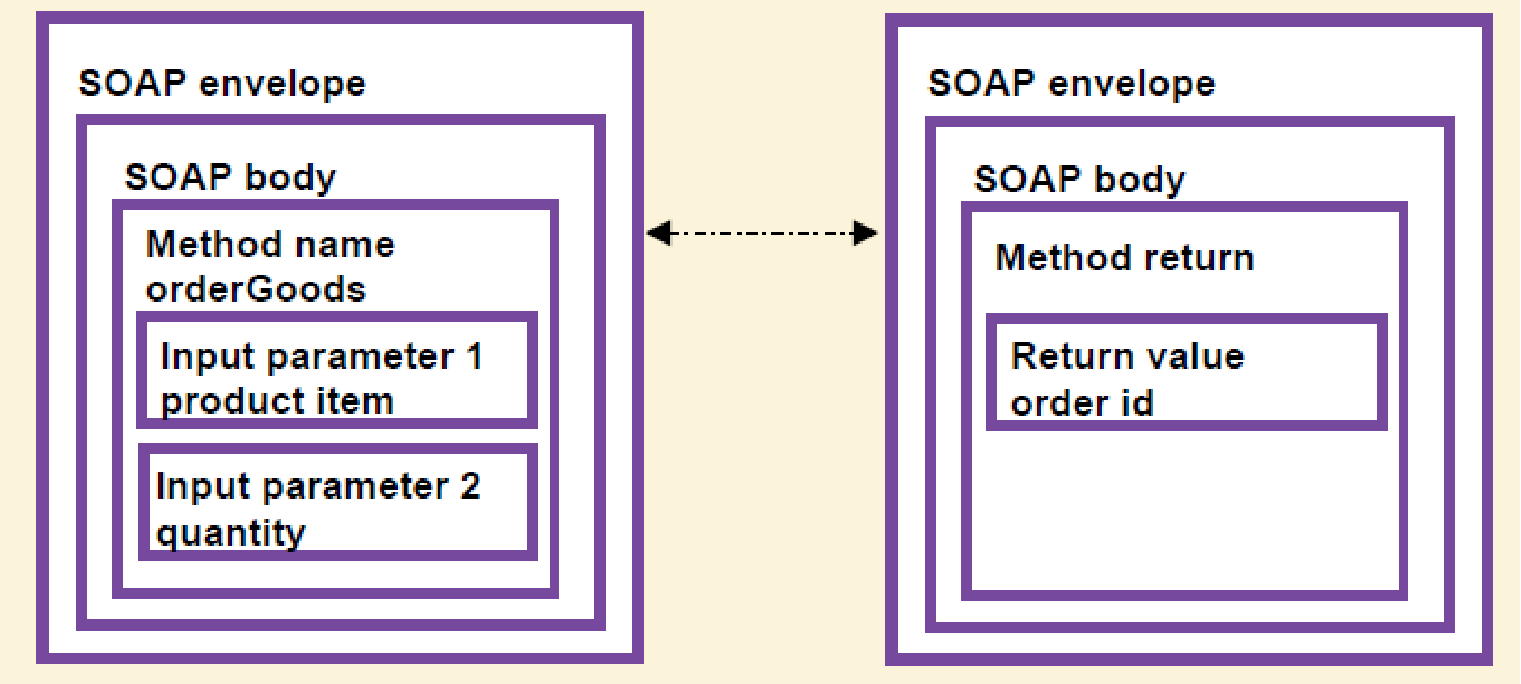
Ebay: Many APIs for searching, buying and posting. WSDL descriptions, client code in Java and .NET

Flickr: Comprehensive API set, free for non-commercial use. REST, XML-RPC, SOAP, with client code in many languages.

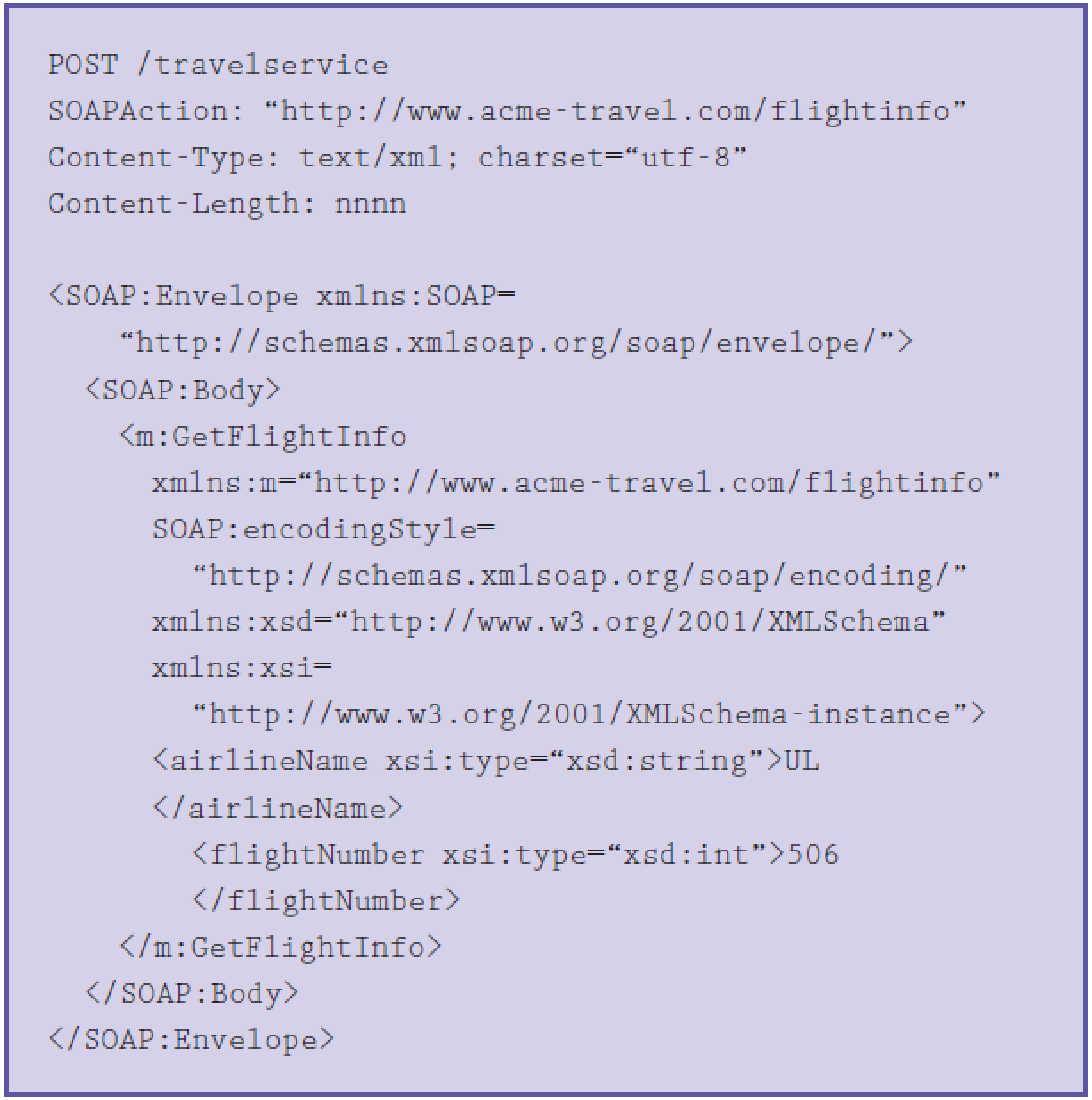
vBulletin: REST interface, most actions supported

SOAP RPC

SOAP RPC messages typically encode arguments that are presented to the calling program as parameters and return values. HTTP POST/GET are used to communicate:



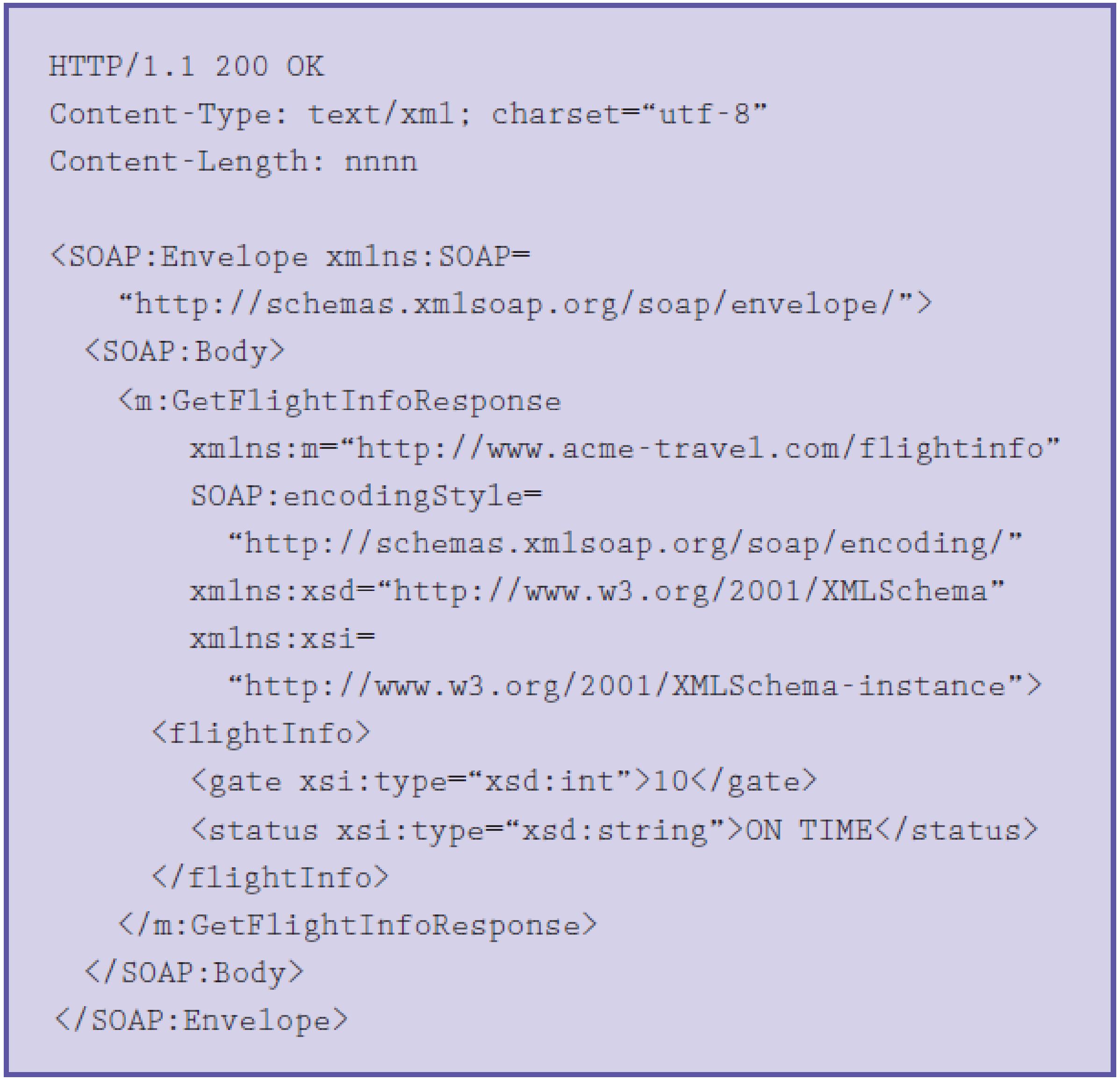
Soap RPC



```
POST /travelservice
SOAPAction: “http://www.acme-travel.com/flightinfo”
Content-Type: text/xml; charset=“utf-8”
Content-Length: nnnn

<SOAP:Envelope xmlns:SOAP=
    “http://schemas.xmlsoap.org/soap/envelope/”>
  <SOAP:Body>
    <m:GetFlightInfo
      xmlns:m=“http://www.acme-travel.com/flightinfo”
      SOAP:encodingStyle=
        “http://schemas.xmlsoap.org/soap/encoding/”
      xmlns:xsd=“http://www.w3.org/2001/XMLSchema”
      xmlns:xsi=
        “http://www.w3.org/2001/XMLSchema-instance”>
      <airlineName xsi:type=“xsd:string”>UL
      </airlineName>
        <flightNumber xsi:type=“xsd:int”>506
        </flightNumber>
    </m:GetFlightInfo>
  </SOAP:Body>
</SOAP:Envelope>
```

Soap Response

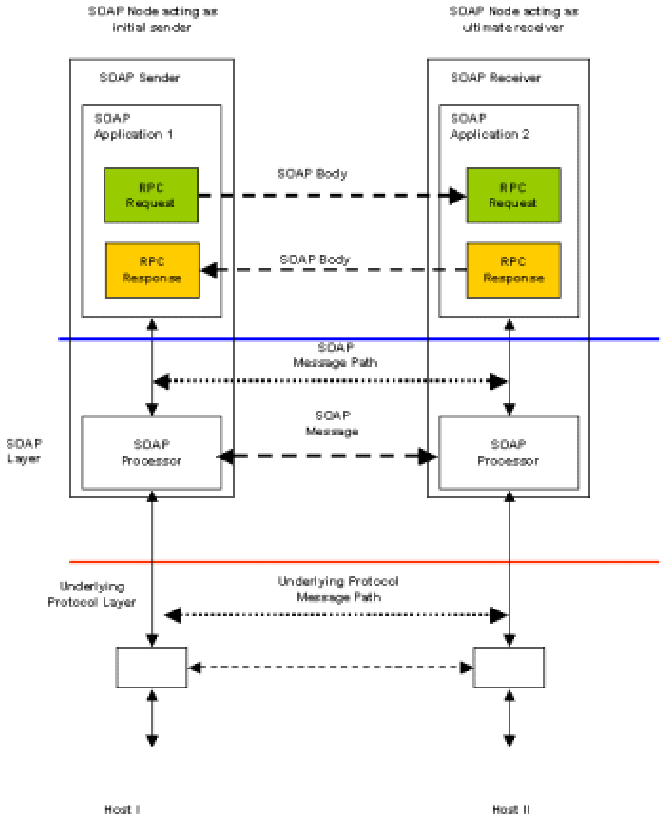


```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=“utf-8”
Content-Length: nnnn

<SOAP:Envelope xmlns:SOAP=
    “http://schemas.xmlsoap.org/soap/envelope/”>
  <SOAP:Body>
    <m:GetFlightInfoResponse
        xmlns:m=“http://www.acme-travel.com/flightinfo”
        SOAP:encodingStyle=
          “http://schemas.xmlsoap.org/soap/encoding/”
        xmlns:xsd=“http://www.w3.org/2001/XMLSchema”
        xmlns:xsi=
          “http://www.w3.org/2001/XMLSchema-instance”>
      <flightInfo>
        <gate xsi:type=“xsd:int”>10</gate>
        <status xsi:type=“xsd:string”>ON TIME</status>
      </flightInfo>
    </m:GetFlightInfoResponse>
  </SOAP:Body>
</SOAP:Envelope>
```

Web Services

XML-RPC, requires a request-response cycle. Often longer “conversations.” i.e. it’s a stateful protocol, and both endpoints need to agree on the state.



REST

REpresentation State Transfer

Stateless Client/Server Protocol: Principles

1. Each message in the protocol contains all the information needed by the receiver to understand and/or process it. This constraint attempts to "keep things simple" and avoid needless complexity

2. Set of Uniquely Addressable Resources
   - "Everything is a Resource" in a RESTful system
   - Requires universal syntax for resource identification (e.g. URI)

REST

3. Set of Well-Defined Operations that can be applied to all resources
   - In context of HTTP, the primary methods are
   - POST, GET, PUT, DELETE
   - these are similar (but not exactly) to the database notion of
   - CRUD (Create, Read, Update, Delete)

4. The use of Hypermedia both for Application Information and State Transitions
   - Resources are typically stored in a structured data format that supports hypermedia links, such as XHTML or XML

REST example

```
<user>
   <name>Jane</name>
   <gender>female</gender>
   <location href="http://www.example.org/us/ny/new_york">
      New York City, NY, USA</location>
</user>
```

This documentation is a representation used for the User resource

It might live at http://www.example.org/users/jane/

- If a user needs information about Jane, they GET this resource
- If they need to modify it, they GET it, modify it, and PUT it back
- The href to the Location resource allows savvy clients to gain access to its information with another simple GET request

Implication: Clients cannot be too “thin”; need to understand resource formats

REST vs. RPC

In RPC systems, the design emphasis is on **verbs**

- What operations can I invoke on a system?
- getUser(), addUser(), removeUser(), updateUser(), getLocation(), updateLocation(), listUsers(), listLocations(), etc.

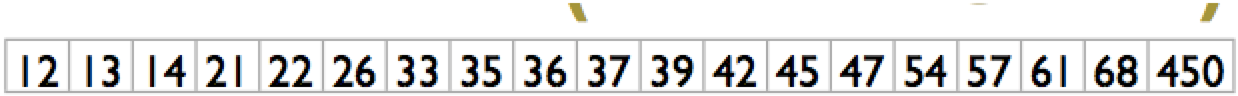
In REST systems, the design emphasis is on **nouns**

- User, Location
- In REST, you would define XML representations for these resources and then apply the standard methods to them

Dirty Data

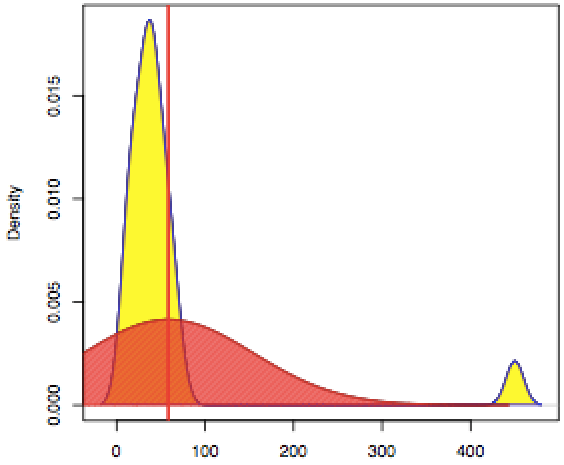
- The Statistics View:
  - There is a process that produces data
  - We want to model ideal samples of that process, but in practice we have non-ideal samples:
    - **Distortion** – some samples are corrupted by a process
    - **Selection Bias** - likelihood of a sample depends on its value
    - **Left and right censorship** - users come and go from our scrutiny
    - **Dependence** – samples are supposed to be independent, but are not (e.g. social networks)
  - – You can add new models for each type of imperfection, but you can't model everything.
  - – What's the best trade-off between accuracy and simplicity?

Numeric Outliers



12	13	14	21	22	26	33	35	36	37	39	42	45	47	54	57	61	68	450

ages of employees (US)

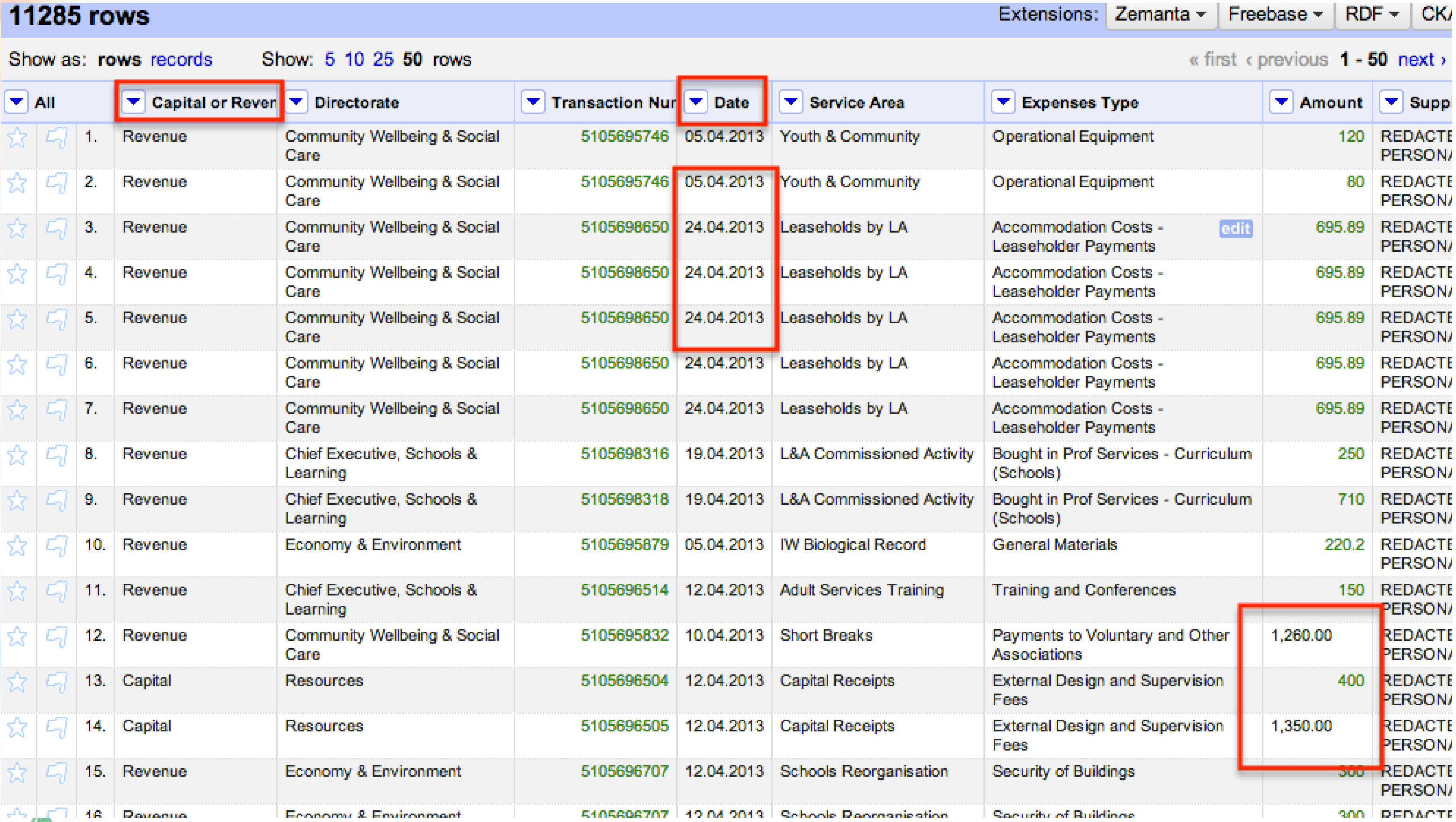


- median 37
- mean 58.52632
- variance 9252.041

Adapted from Joe Hellerstein's 2012 CS 194 Guest Lecture

Data Cleaning Tools: OpenRefine

- Spreadsheet-like tool allowing data quality checking: reformatting, substitution, constraint checking etc.



11285 rows
Extensions: Zemanta ▾ Freebase ▾ RDF ▾ CK/
Show as: rows records
Show: 5 10 25 50 rows
« first ‹ previous 1 - 50 next ›
edit

All	Capital or Reven	Directorate	Transaction Nur	Date	Service Area	Expenses Type	Amount	Supp
1.	Revenue	Community Wellbeing & Social Care	5105695746	05.04.2013	Youth & Community	Operational Equipment	120	REDACTE PERSON/
2.	Revenue	Community Wellbeing & Social Care	5105695746	05.04.2013	Youth & Community	Operational Equipment	80	REDACTE PERSON/
3.	Revenue	Community Wellbeing & Social Care	5105698650	24.04.2013	Leaseholds by LA	Accommodation Costs - Leaseholder Payments	695.89	REDACTE PERSON/
4.	Revenue	Community Wellbeing & Social Care	5105698650	24.04.2013	Leaseholds by LA	Accommodation Costs - Leaseholder Payments	695.89	REDACTE PERSON/
5.	Revenue	Community Wellbeing & Social Care	5105698650	24.04.2013	Leaseholds by LA	Accommodation Costs - Leaseholder Payments	695.89	REDACTE PERSON/
6.	Revenue	Community Wellbeing & Social Care	5105698650	24.04.2013	Leaseholds by LA	Accommodation Costs - Leaseholder Payments	695.89	REDACTE PERSON/
7.	Revenue	Community Wellbeing & Social Care	5105698650	24.04.2013	Leaseholds by LA	Accommodation Costs - Leaseholder Payments	695.89	REDACTE PERSON/
8.	Revenue	Chief Executive, Schools & Learning	5105698316	19.04.2013	L&A Commissioned Activity	Bought in Prof Services - Curriculum (Schools)	250	REDACTE PERSON/
9.	Revenue	Chief Executive, Schools & Learning	5105698318	19.04.2013	L&A Commissioned Activity	Bought in Prof Services - Curriculum (Schools)	710	REDACTE PERSON/
10.	Revenue	Economy & Environment	5105695879	05.04.2013	IW Biological Record	General Materials	220.2	REDACTE PERSON/
11.	Revenue	Chief Executive, Schools & Learning	5105696514	12.04.2013	Adult Services Training	Training and Conferences	150	REDACTE PERSON/
12.	Revenue	Community Wellbeing & Social Care	5105695832	10.04.2013	Short Breaks	Payments to Voluntary and Other Associations	1,260.00	REDACTE PERSON/
13.	Capital	Resources	5105696504	12.04.2013	Capital Receipts	External Design and Supervision Fees	400	REDACTE PERSON/
14.	Capital	Resources	5105696505	12.04.2013	Capital Receipts	External Design and Supervision Fees	1,350.00	REDACTE PERSON/
15.	Revenue	Economy & Environment	5105696707	12.04.2013	Schools Reorganisation	Security of Buildings	300	REDACTE PERSON/
16.	Revenue	Economy & Environment	5105696707	12.04.2013	Schools Reorganisation	Security of Buildings	300	REDACTE

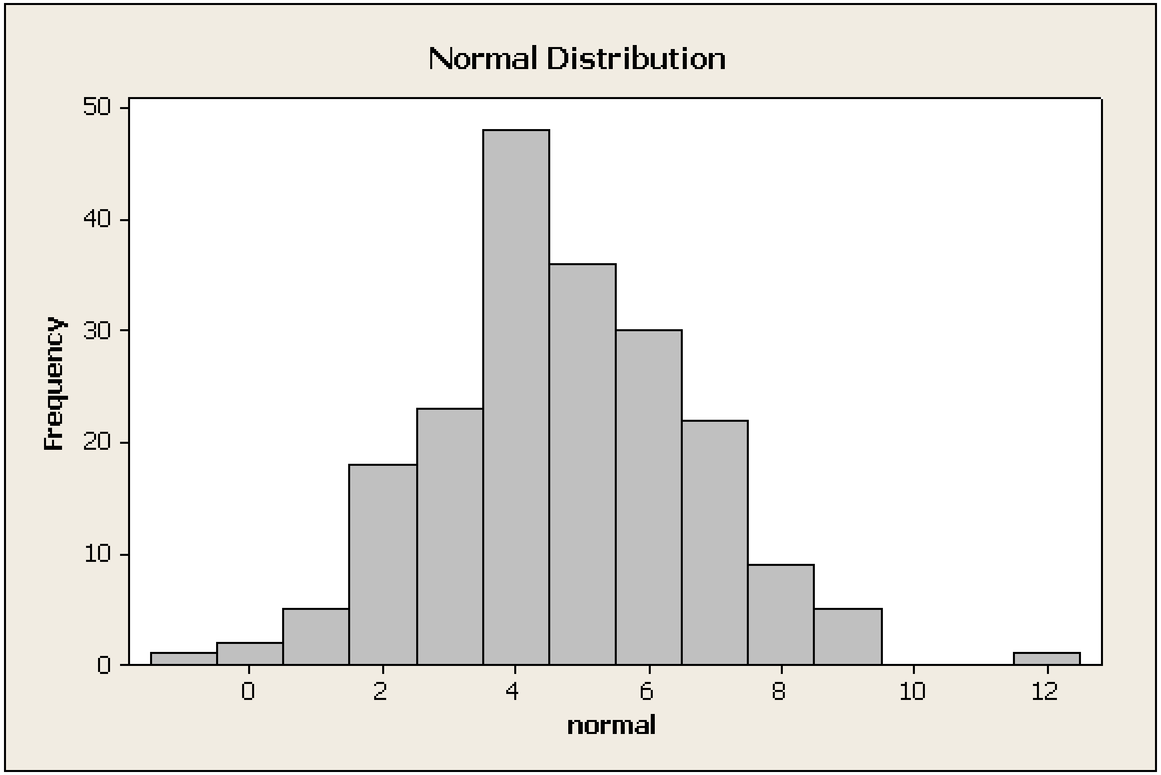
Exploring

- Get familiar with your favorite graphing package:
  - Matplotlib is widely used in Python
  - Ggplot is good for more advanced plots (similar to R)
  - D3.js popular for interactive graphics, but low-level:
    - Bokeh provides high-level primitives
    - Vega/Vincent same goals, developed by Trifacta

- Get fluent with plotting:
  - Histograms
  - Scatter plots
  - Line and bar plots

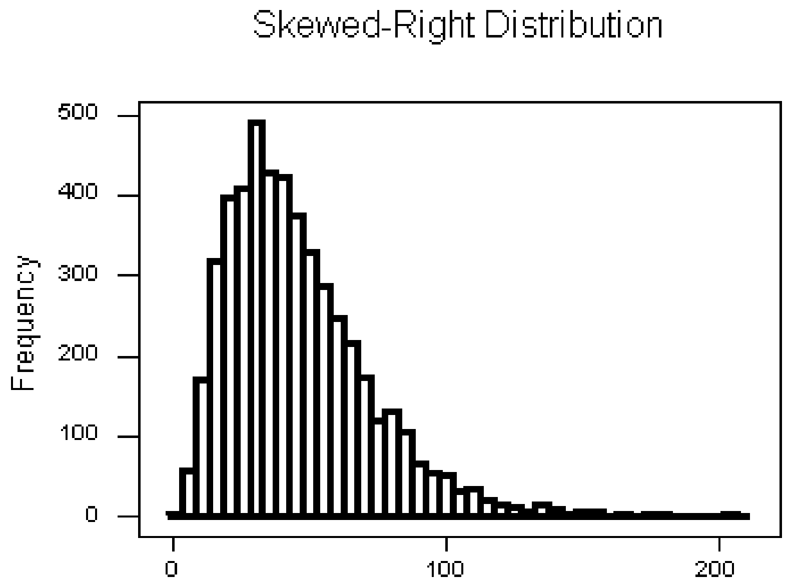
Looking at Data

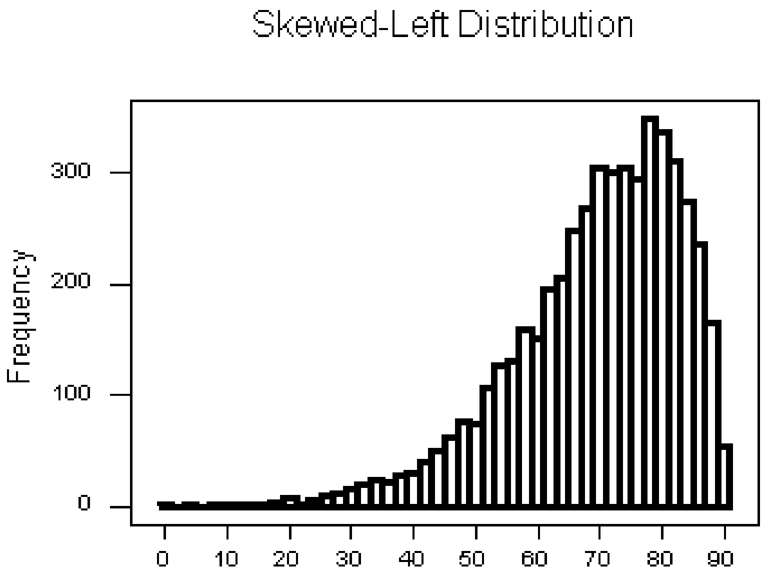
- Histograms can tell you a lot about a single variable, discrete or continuous:



Looking at Data

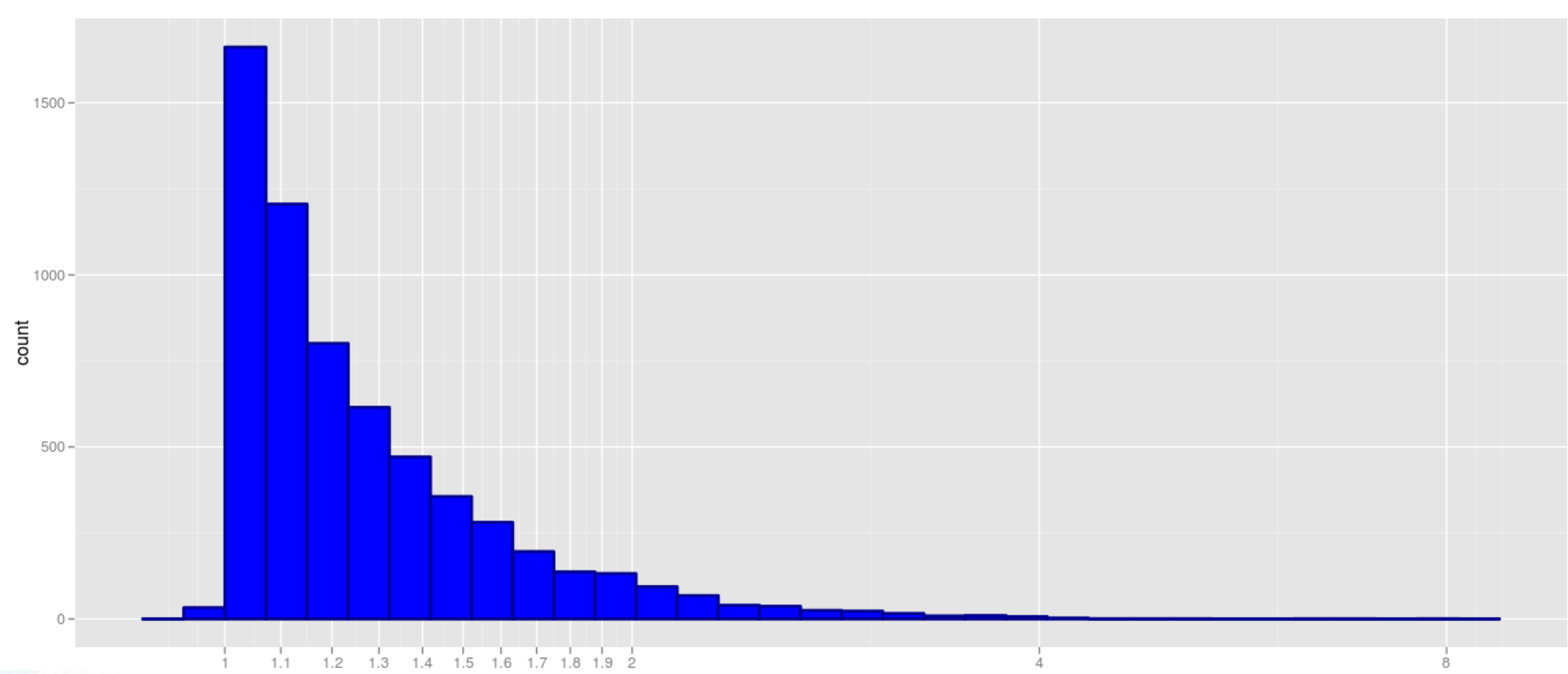
- Skewed distributions:





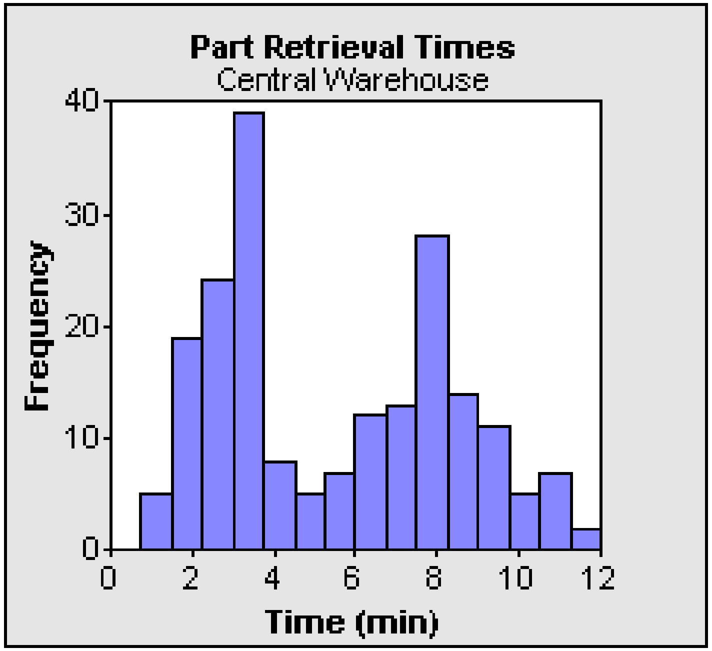
Long-tailed data

- Long tailed data



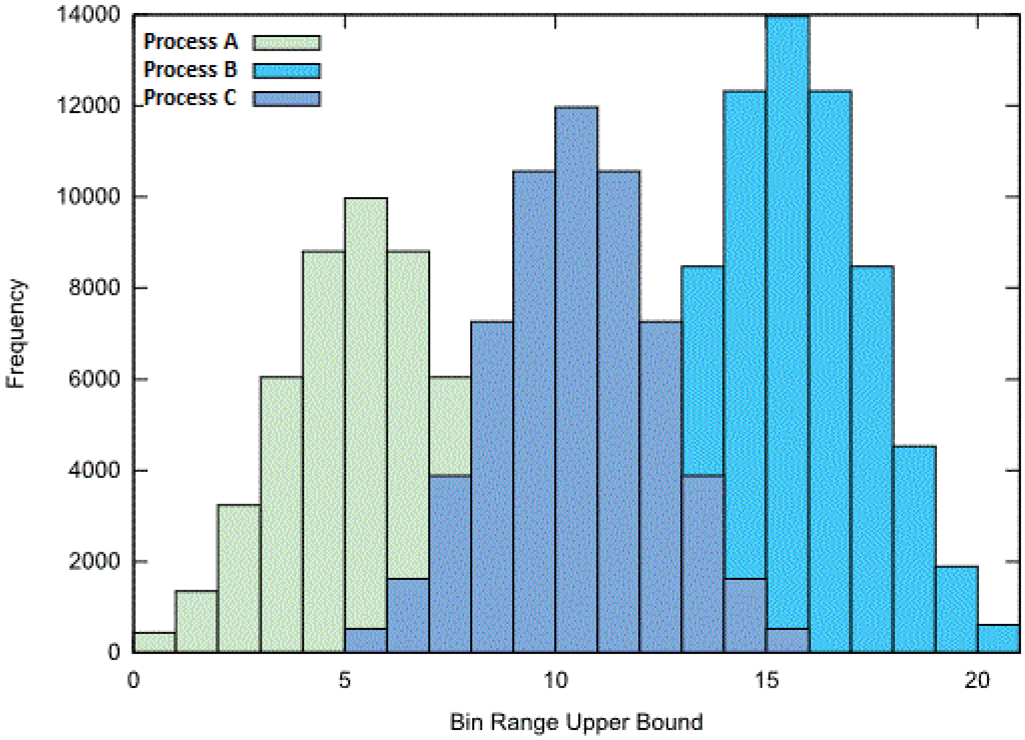
Multimodal data

- Two or more distinct peaks in a histogram.
- Suggests two or more distinct populations of samples.
- Often arise from gender/political views, other binary factors.
- But don’t guess!! Explore further by using, e.g. color and a histogram of multiple populations.



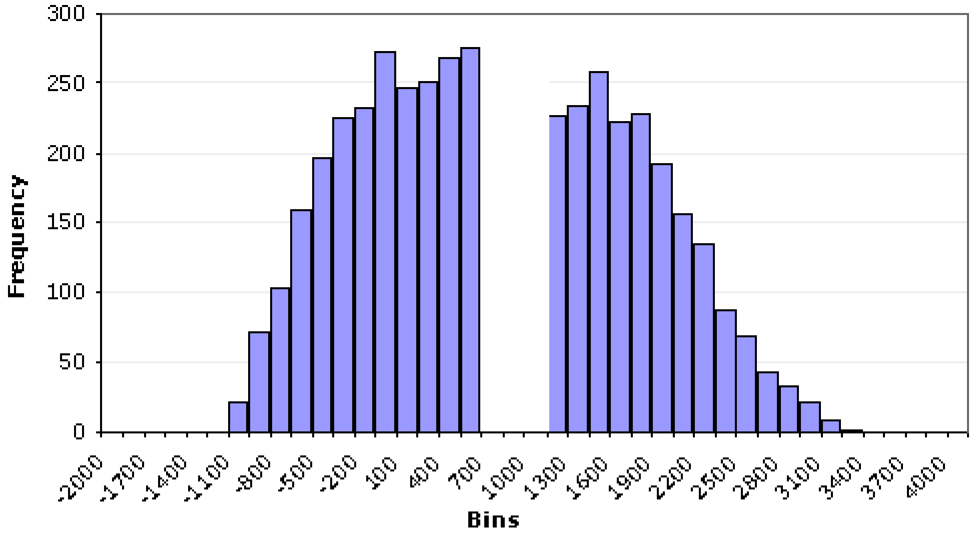
Multimodal data

- Explore further by using, e.g. color and a histogram of multiple populations.



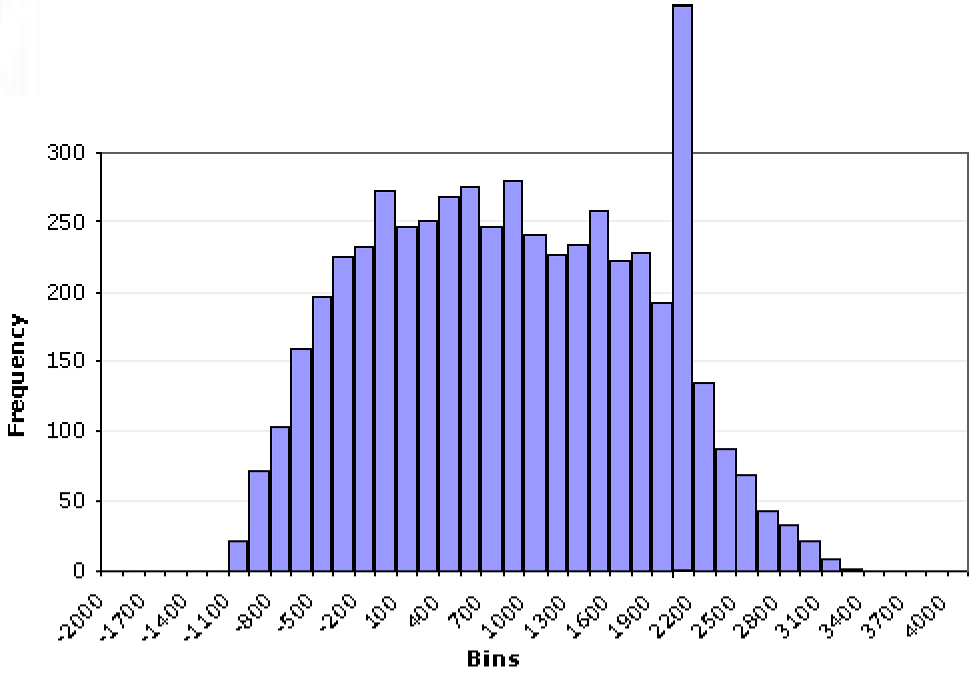
Weird data

- Some data are very hard to explain.
- Don’t try. Trace through the data pipeline to find where the strangeness comes from. Usually it’s a processing bug.



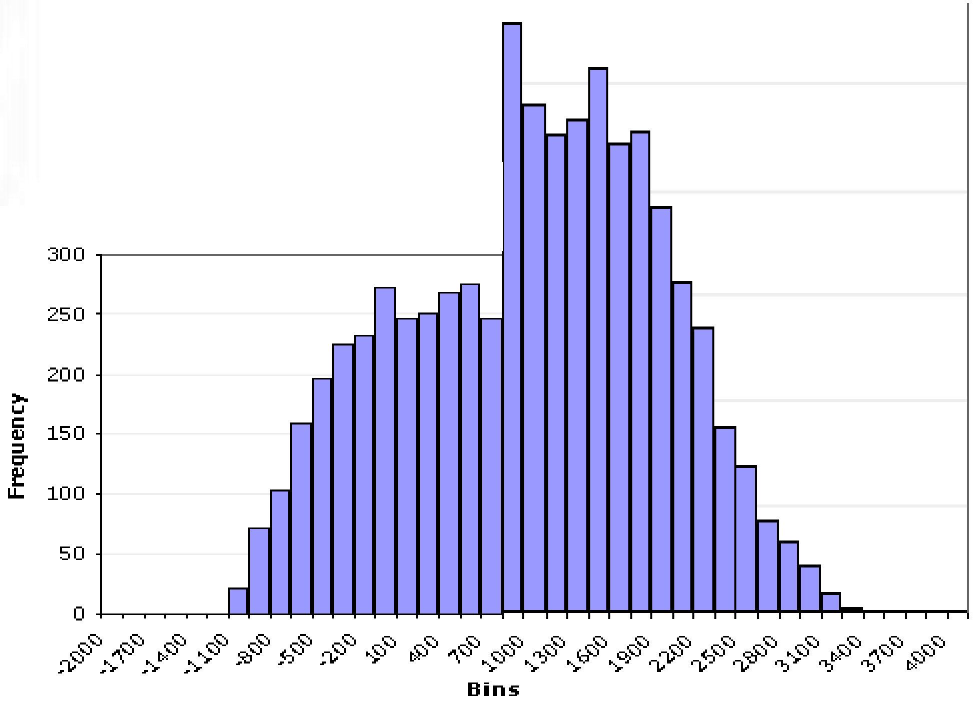
Weird data

- Some data are very hard to explain.
- Don’t try. Trace through the data pipeline to find where the strangeness comes from. Usually it’s a processing bug.



Weird data

- Some data are very hard to explain.
- Don’t try. Trace through the data pipeline to find where the strangeness comes from. Usually it’s a processing bug.

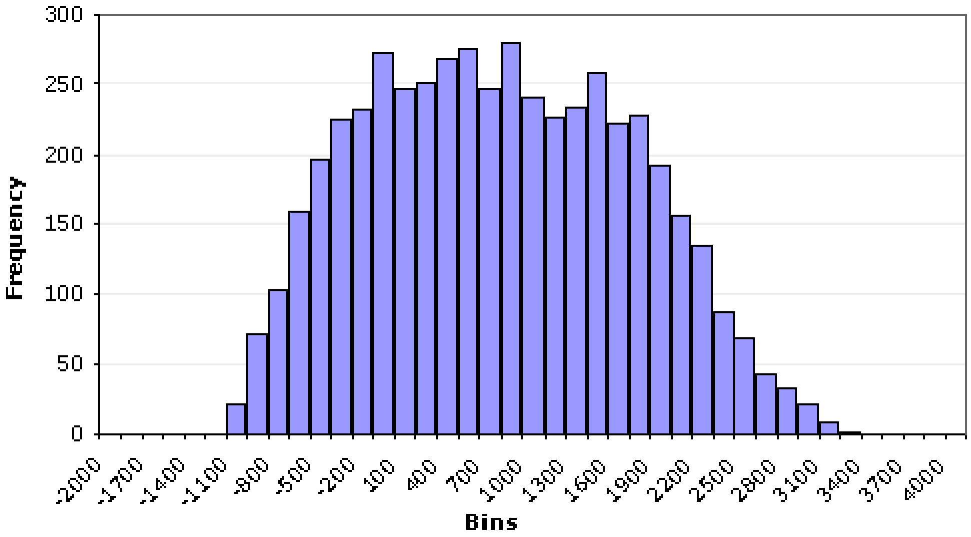


Proactive Weird data Detection

- If data look normal, take a picture and save it for later…

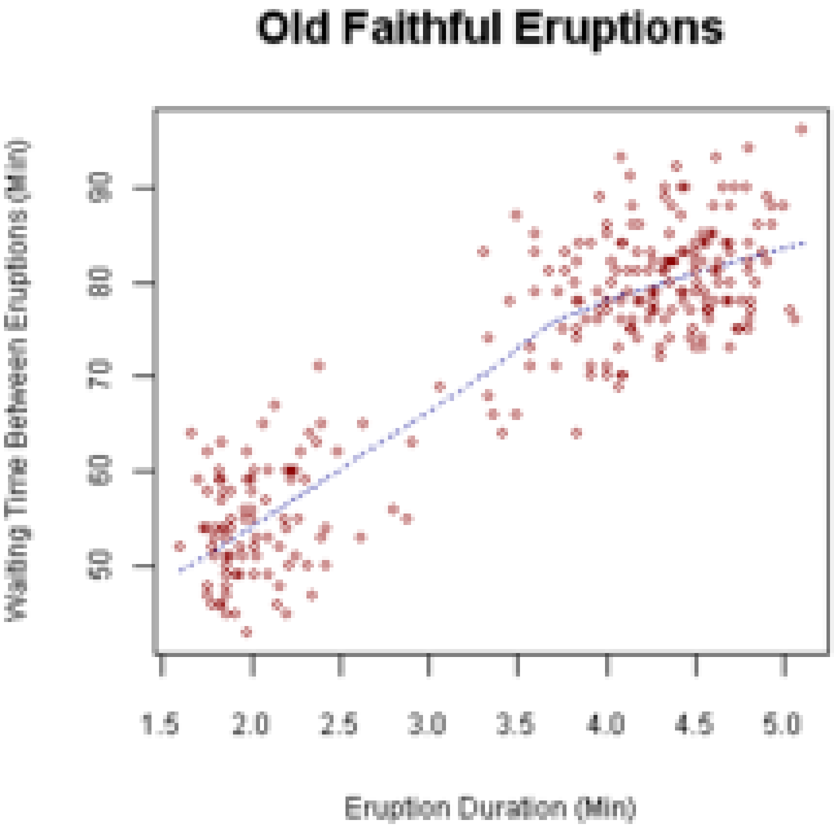
- Then periodically compare new data with old whenever there is a pipeline update.

- Always try to have a theory of what the data should look like.



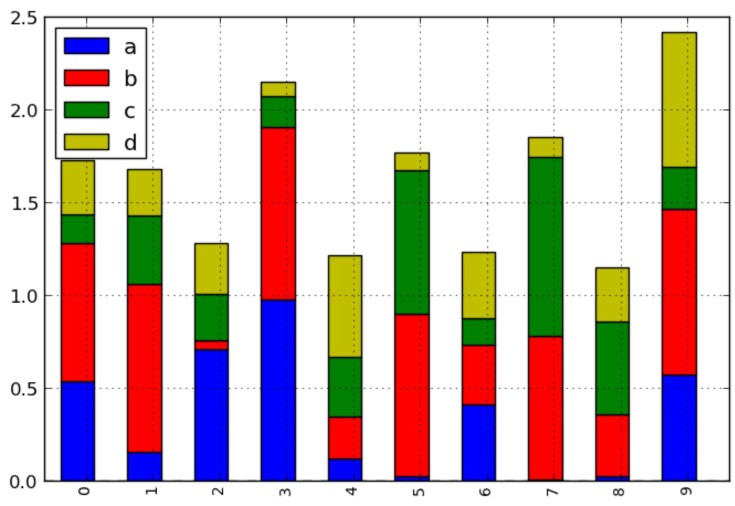
Two variables – Scatter plots

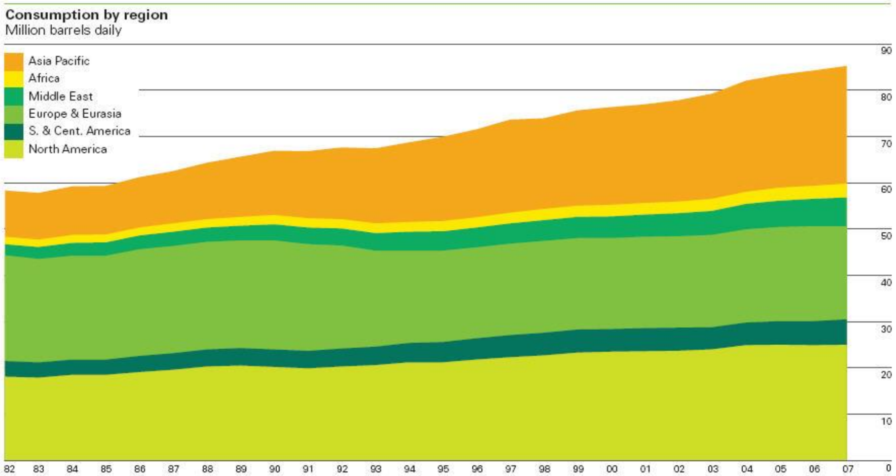
- Scatter plots quickly expose the relationships between two variables



More than two variables

- **Stacked plot:** stack variable is discrete:

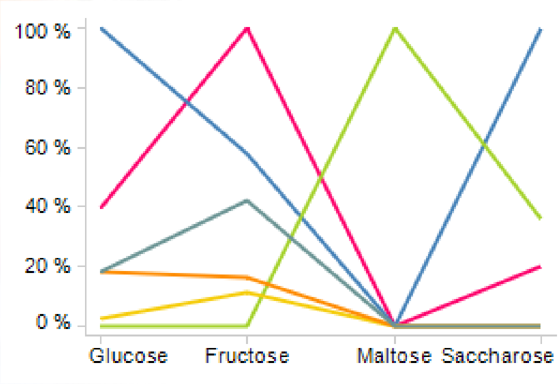




World oil consumption rose by about 1mmb/d in 2007, just below the 10-year average. OECD consumption declined nearly 400,000b/d. China accounted for the largest increment to consumption even though the growth rate was below average. Consumption in oil exporting regions was robust.

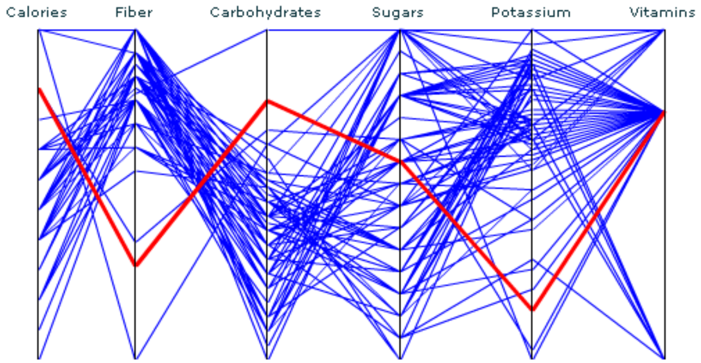
More than two variables

- **Parallel coordinate plot:** one discrete variable, an arbitrary number of other variables:



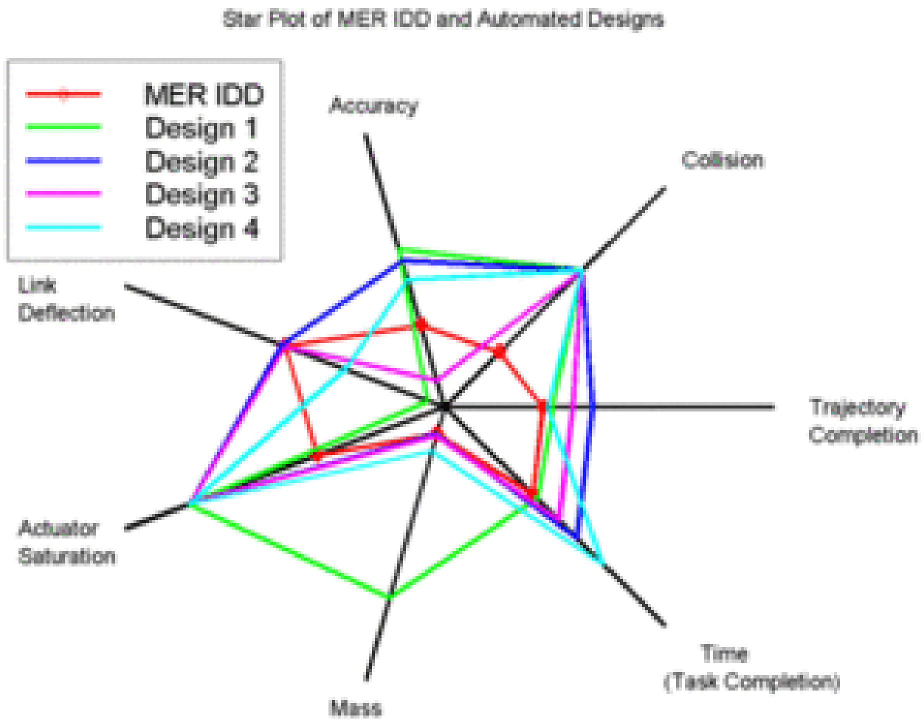
Line by: (Row Number) +

Color by: Food +



More than two variables

- **Radar Chart:** Similar: one discrete variable (design here), an arbitrary number of other variables:



Closing Remarks

- We argued for analysts to **form expectations** of what the data should look like. This helps guard against pipeline errors and to identify interesting patterns.
- An observer should also be atune to patterns that we not part of their theory. In other words to "expect the unexpected".