ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

APPLIED DATA ANALYSIS

School Of Computer And Communication Sciences

Applied Data Analysis Summary

Prof. Catasta Michele
Distributed Information Systems Laboratory (LSIR)
michele.catasta@epfl.ch

June 10, 2016

Contents

1 Introduction

1.1 General information about the course

This course covers multiple topics in the data science field such as **Data Wrangling**, **Data Management**, **Data Mining**, **Machine Learning**, **Visualization**, **Statistics** and **Story telling**. It's about **breadth**, not depth. Indeed, Data science is evolving really quickly, hence learning in depth a specific tool won't pay off.

1.2 Data Science

When we talk about Data Science, we often use the term Big Data as the enormous amount of data that exist in the world. But Big Data is not only about collecting huge amount of data. It is challenging but not enough. The real value comes from the insights. The *internet* companies (Google, Facebook, etc.) understood this many years ago.

An accurate definition of Data Analysis is given by Wikipedia:

> **Analysis of data** *is a process of* **inspecting**, **cleaning**, **transforming**, *and* **modeling data** *with the goal of* **discovering useful information**, *suggesting conclusions, and supporting decision-making. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names,* **in different business**, *science, and social science* **domains**. Wikipedia - Data Analysis

Therefore, a Data Scientist has to master different kind of skills such as **Mathematics** (for the Statistics), **Programming** and the **Domain Expertise**. Drew Conway's Venn diagram, Figure 1, shows the different combination man can obtain with these three skills.

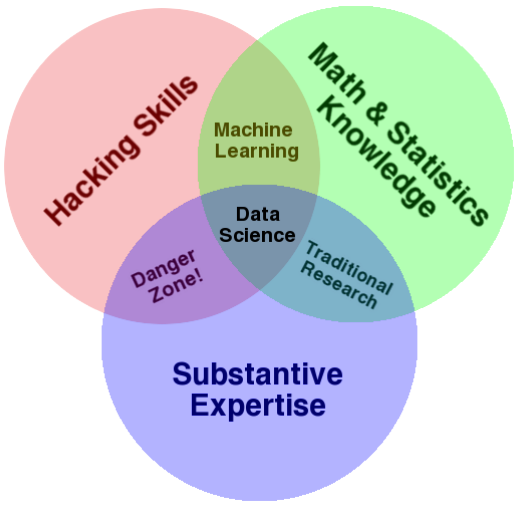


Figure 1: Venn Diagram describing the different combination of skills used by a Data Scientist (by Drew Conway)

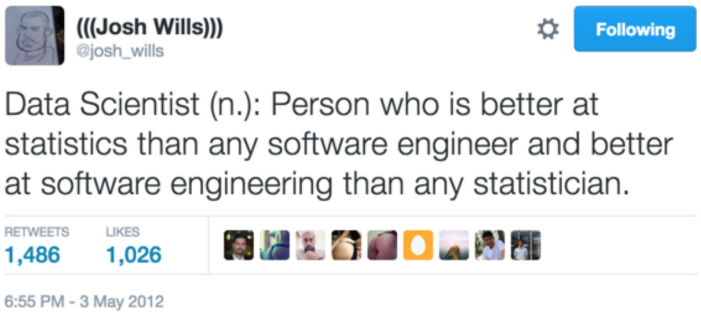
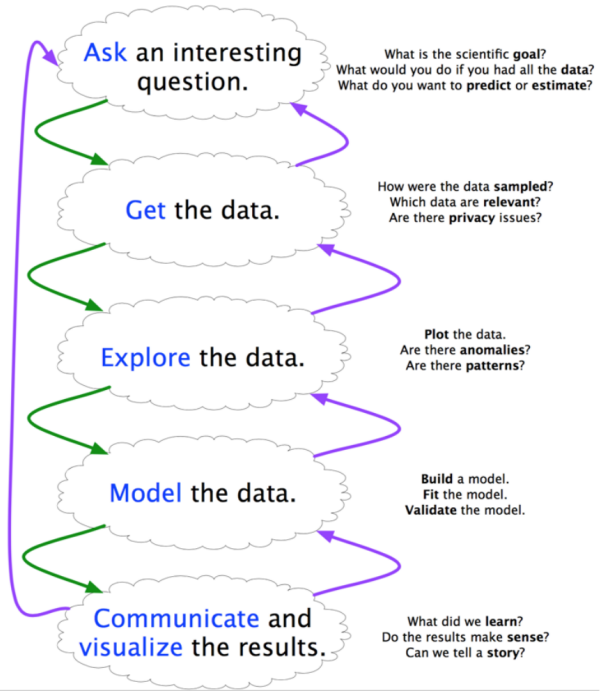


Figure 2: A tweet from Josh Wills, Data Scientist at Slack.

A practical definition of Data Science

Data Science is about the whole processing pipeline to extract information out of data. As such, a Data Scientist **understands and cares about the whole data pipeline**.



A **data pipeline** consists of 3 steps:

1. Preparing to run a model.
   Gathering, cleaning, integrating, restructuring, transforming, loading, filtering, deleting, combining, merging, verifying, extracting, shaping
2. Running the model
3. Communicating the results

A "good" Data Scientist will always go back and forth between the steps. The diagram on the left shows exactly what can happen.

In this course, you will develop the following skills:

data muning/scraping/sampling/cleaning in order to get an informative, manageable data setlength

data storage and management in order to be able to access data quickly and reliably during subsequent analysis

exploratory data analysis to generate hypotheses and intuition about the data

prediction based on statistical tools such as regression, classification, and clustering

communication of results through visualization, stories and interpretable summaries

2 Basic concepts

A data science student is attended to understand the **Grammar of Data Science**. Having some backgrounds in SQL concepts is also a good thing because, as it is very common, people loves to make example with it. Here is a brief refresh of some definitions and concepts about data science.

- **Structured data** requires two key concepts:
  - **Data model** is a collection of concepts for describing data.
  - **Schema** is a description of a particular collection of data, using a given data model.
- The **Relational model** is on of the most common approach to manage data (SQL like) and can handle most of the data. A counter example is the facebook-like data which requires **graph model**. This model is made of 2 parts:
  - The **Schema**.
    For example, `Students(sid: string, name:string, age:integer)`
  - The **Instance**, *i.e.* the data at a given time.
    Definitons:
    - **Cardinality** is the number of rows. (Number of items)
    - **Degree** or **Arity** is the number of fields. (Number of attributes)
- Definitions of some "Database" terms:
  - A **JOIN** is a mean to combine tables based on shared attributes (most of the time some **IDs**). Despite its apparent simplicity beware of the many ways to compute a JOIN and check what is the default JOIN of a language before using it. The FIG 3 summarises these possibilities.
  - **Aggregation**, **reduction**, and **groupby** are the action of reducing data with a common opperation (**sum**, **count**, **average**, ...) to summarise them.
- Definitions of some "Pandas" terms:
  - **Series** are a name, ordered dictionary
    - keys are indexes
    - built on **numpy.ndarray** (so values can be any Numpy data type)
  - **DataFrame** is a table with named column
    - the columns are series
    - it is indeed a dictionary with (columnName $\rightarrow$ series)

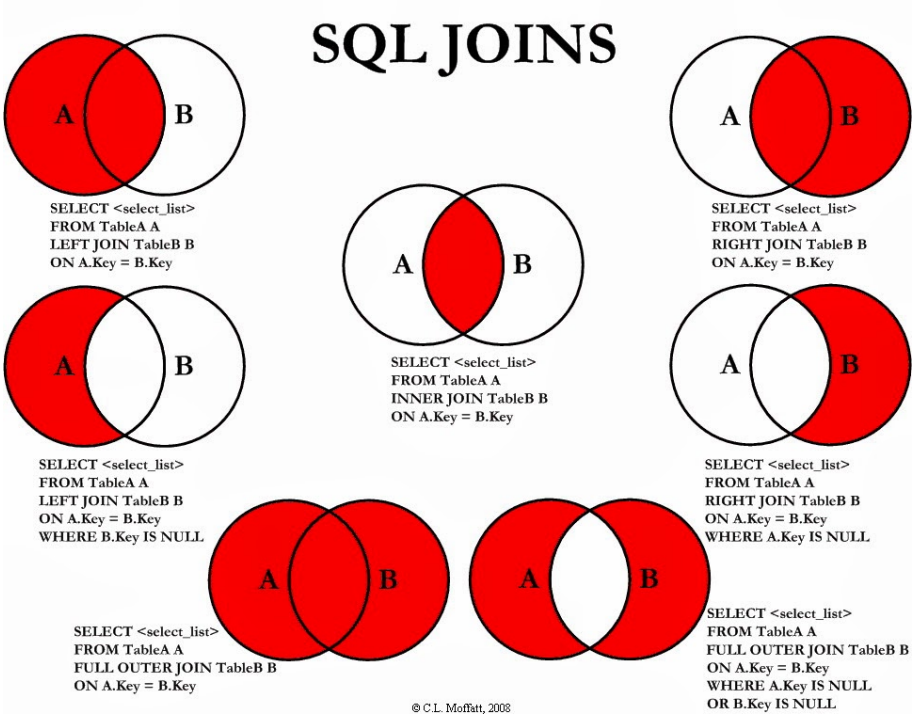


Figure 3: Different ways to join two tables and the related SQL command.

2.1 Panda vs SQL

Panda is built to allow easy and fast **data exploration** and not to be a database manager, as SQL is. Thus there are benefits and drawbacks of using it.

Pros	**Cons**
Lightweight & fast	Tables stored directly in memory
Great expressivness (combine SQL + Python)	No post-load indexing functionality
Easy plot for data visualization (eg Matplotlib)	No transactions, journalings
	Large, complex joins are slower

2.2 OnLine Analytical Processing (OLAP cubes)

OLAP tools enable users to analyze multidimensional data interactively from multiple perspectives. Conceptually, it is like an n-dimensional spreadsheet (a cube) on which we can apply various opperations to take decisions.

OLAP cubes are an other way to see data table and are contructed based on them, as shows FIG 4.

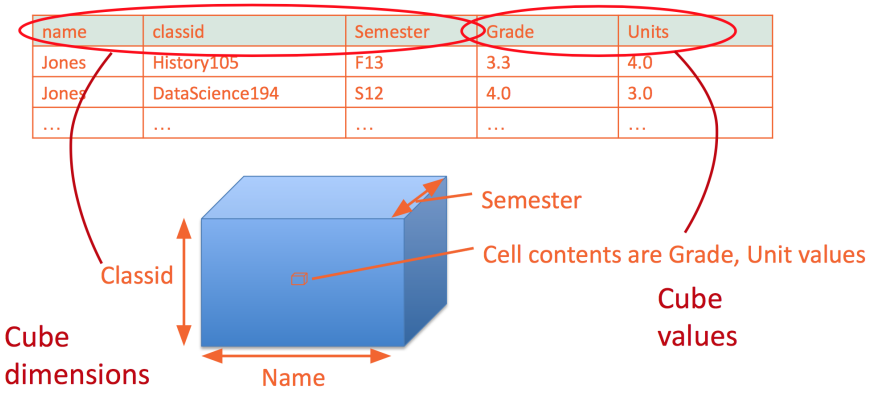
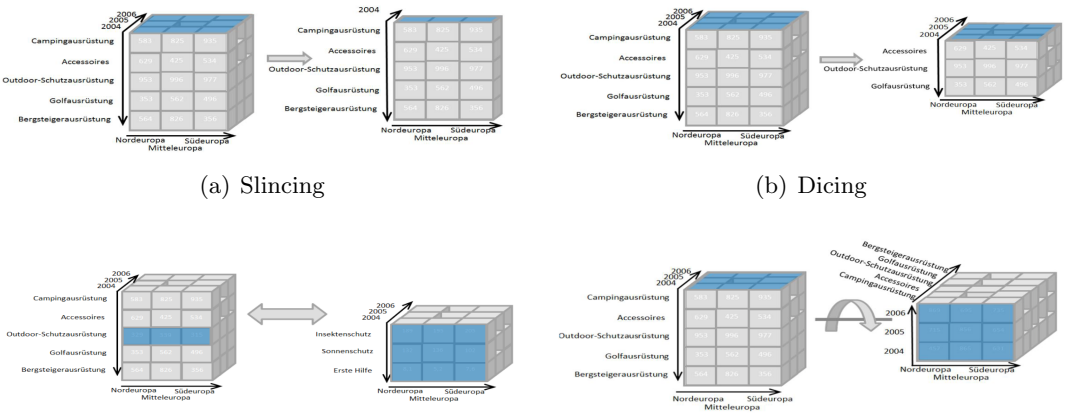


Figure 4: Construction of an OLAP cube from a table.

Operations on OLAP cubes are the following and are illustrated on FIG 5

- **Slicing** fixes one or more variable
- **Dicing** selects a range of one or more variable
- **Driling up/down** changes levels of a hierarchically-indexed variable, ie "zoom" on a variable and see the sub-categories it contains.
- **Pivoting** change the point of view of the cube. Swap an aggregated variable an a detailed one.



(a) Slincing (b) Dicing

(c) Driling up/down (d) Pivoting

Figure 5: Operations on OLAP cubes

Pros	Cons
The main adventage of OLAP cubes is that their are **conceptualy simpler** to understand by a non-scientist person, eg a business man who have to take day-to-day decisions based on company's data. Aggregations are limited but cover the main common cases that we can encounter.	Because of the "on-line" behaviour of this approach, all type of aggregation must be pre-calculated amoung all combination of axis which is very **expensive in memory and in time** (when updating the data)

3 Data Wrangling

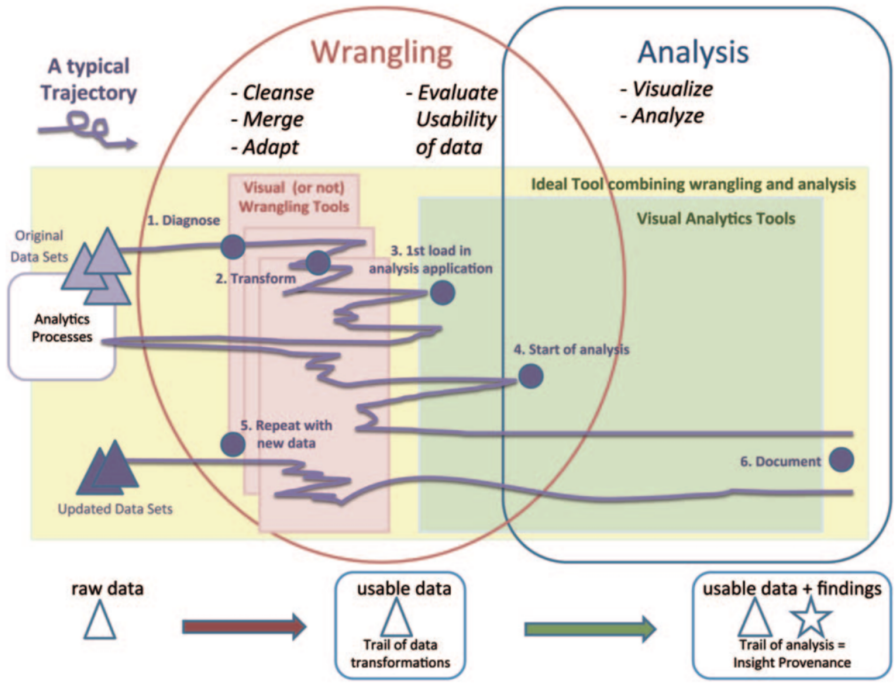


Figure 6: Things do not always happen as expected...

Before any analysis, data need to be transformed from "dirty" to clean and processable data.
Data comes from different sources (excel or SQL?), sometime collected through different methods over time, with different conventions (space or NaN?), etc ... Data wrangling's goal is to **extract and standardize these raw data**. The best way to do it is to **combine automation with visualizations** in order to find outliers.
Data's problem can come from (non-exhaustive):

- Missing data
- Incorrect data
- Inconsistent representations of the same data

- Non-standardized data (centimeter or inches? farenheit or celsuis ?)
- Duplicated data

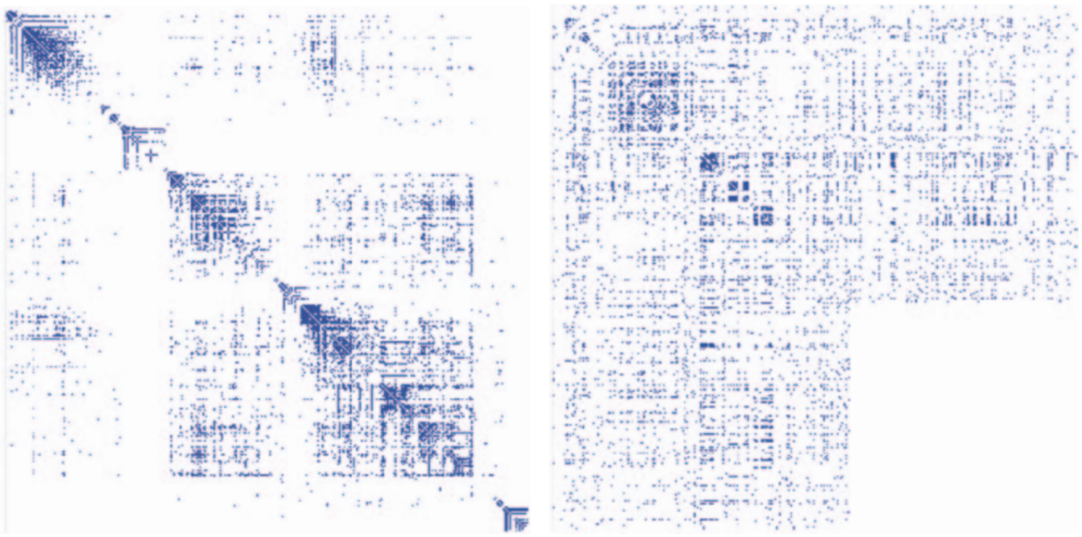
About 75% of theses problem will need **human intervention** to be corrected (by the data-scientist or by crowdsourcing).
Even if it seems really dirty, **beware not to over-sanitize the data!**. Applying what we can call "defensive programming" is not a good idea because we risk to lose any interesting data, keeping only the ones that fit perfectly in our model.

3.1 Diagnosis of the data

One of the most important aspect of Data Wrangling is to **understand** the data and to **find possible problems**. In order to "diagnose" the data, two tools can be used:

- **Visualization** (A *toughtful* visualization will always help)
- **Basic Statistics**

Matrix visualizations of the facebook graph is shown in Figure 7. The Relational visualization, Figure 7(a), does not show any particular problem in the data. But the Time dependant visualization, Figure 7(b), shows that the Facebook API reached its limit while collecting data.



(a) Relational visualization (b) Time dependant visualization

Figure 7: Matrix visualization of the facebook graph.

3.2 Dealing with missing values

Values can often miss from the data we have, because of various events (war, fire, ...). We must detect and correct these values with different method according with the domain we are working in.

Whatever the method used, it's good to keep track of these changes to know which are original data and which are modified ones.

- Set values to zero FIG 8(a)
- Interpolate based on existing data FIG 8(b)
- Omit missing data FIG 8(c)
- Interpolation with track kept 8(d)

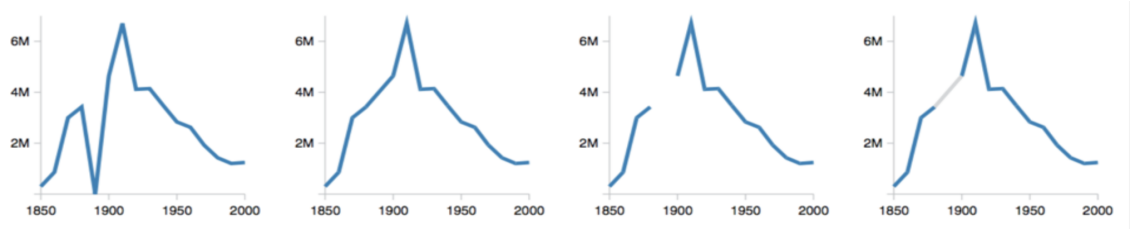


Figure 8: To deal with missing values.

3.3 General procedure

Once the data are well wrangled and before trying to analyse them we must take care of two more steps:

1. **Deal with uncertain data** (can arise from measurement errors, wrong sampling strategies, etc.)
2. **Parse/trasform data** (with aggregation and reduction techniques) to obtain meaningful records

It's always ideal to have the code and/or the documentation about the dataset you are analyzing (provenance).

4 Data Variety

The "3 Vs" of Big Data: *Volume*, *Velocity* and **Variety**. A course on Database will cover the *Volume* and *Velocity* parts. In this course, we try to figure out how to address the **Variety** part.

ETL:

- **Extract** from the *source(s)*.
- **Load** data into the *sink*.

- **Transform** data at the source, sink, or in a *staging area.*

Semi-structured data: no strict structure, but some info are given. =¿ Data range comes from very structured data in DB to totally unstructered data such as web pages.

Traditional DB are **schema-on-write**. (You can't load data into a table without a schema!) But NoSQL (Next generation) DB are **schema-on-read** or **schemaless**. You can either avoid having a schema or have a schema only when you read data. Examples of schemaless: Youtube, Google Cache.

Schema-on-write data type: SQL Schema-rn-read data type: XML (when stored without a schema, the numerical data is stored as **strings**) Using a parser, we can return a typed data. (Faster because number not string + can make checks. =¿ Type is good)

XML and **DOM** (Document-Object Models). Every webpage on internet becomes a DOM. Tree-structured. XML QUeries: allows a DB to interpret data when running queries, *i.e.* to do **arithmetic** or **range queries** on the numerical values. (With or without a schema)

JSON (Javascript Object Notation) Totally schemaless. But we can also use with schema. It's used to represent **hierachical data structures** directly in the languages. (Lot of libraries to do that). Transformation on the data are **procedural** in the target language. 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 uses type inference (string, int, float, ...).
  - More succinct in ASCII form.
  - Transformation/ingestion rely on code (Java or Javascript).

Tabular Data -¿ CSV or TSV Description of 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 (index, name) pair - A cell may or may not have a **value** Schema = (minimal) column types)

Sensors output data in the form of time series -¿ tabular data. Systems dealing with sensor data should

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

Complexity of data format does not determine complexity of the system required to handle it. -¿ Stock market

Log Files Processes, usually daemons, create logs. (https, mysqld, syslogd, ...) Syslog - Standard for System messages. Enables rich analysis. "Spelunking" for bugs -¿ Splunk: Monitor ressources many machines.

Processing XML and JSON DOM is an easy object to work with: all the data is accessible by links once it's in the memory. The problem is that I might not care about most of the data =¿ we might **not be able to fit the DOM for a large object in RAM**.

SAX: Event-Driven Parsing. Helps to deal with Big Data. Exists with pretty much any format that exists. It finds all the **open-close-tag events** in an XML documents, and **does callbacks to user code**.
Pros: + User code can respond to only a subset of events corresponding to the tags. + User code cna correctly computer aggregates from the data rathen than create a record for each tag. + User code can implement felxible error recovery strategies for ill-formed XML - User code must implement a state machine to keep track of "where it is" in the DOM tree.

Most JSON parsers construct the "DOM" directly. Sometimes SAY-style is the only way to handle ill-formed dataset (endless array of objects)

Binary formats Often the key to performance, **avoiding expensive parsing**! Modern formats support nested structures, various levels of schema enforcement, **compression**, etc. **Consider converting to one of those formats at the beginning of your processing pipeline (especially on a project with Big Data!)**

- Protocol Buffers (Goggle)
- Avro (supports schema evolution)
- Parquet (column oriented, first-class citizen in Spark)
- etc.

HTML Common Crawl has 0.1% of Google's web crawl. Because Google can scrape data which are behind the form (hidden link, authentication, forms, etc.) using their technology. Common Crawl just uses a simple "algorithm" to extract links.
List and describe in one line the different HTML stuff. WikiData, schema.org, etc.

Web Services Large web sites discourage to do screen-scraping. User Web Service APIs. This is the **right** was to get data from online sources.
W3C: A "Web service" is "a software system designed to support interoperable machine-to-machine interaction over a network". Two kinds: - XML-base RPC-style messages: SOAP - REST-stly stateless interactions, URLs encode state.

REST: REpresentation State Transfer DEFINITION HERE! slide 47-48

REST vs RPC