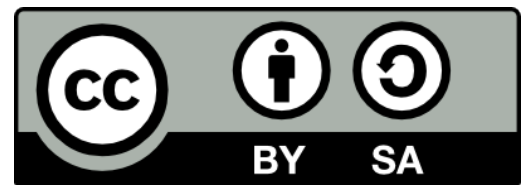
# Big Data I:

# Ingeniería de datos

Felipe Ortega
Dpto. de Estadística e Investigación Operativa
Universidad Rey Juan Carlos

March 10, 2015

Universidad
Rey Juan Carlos



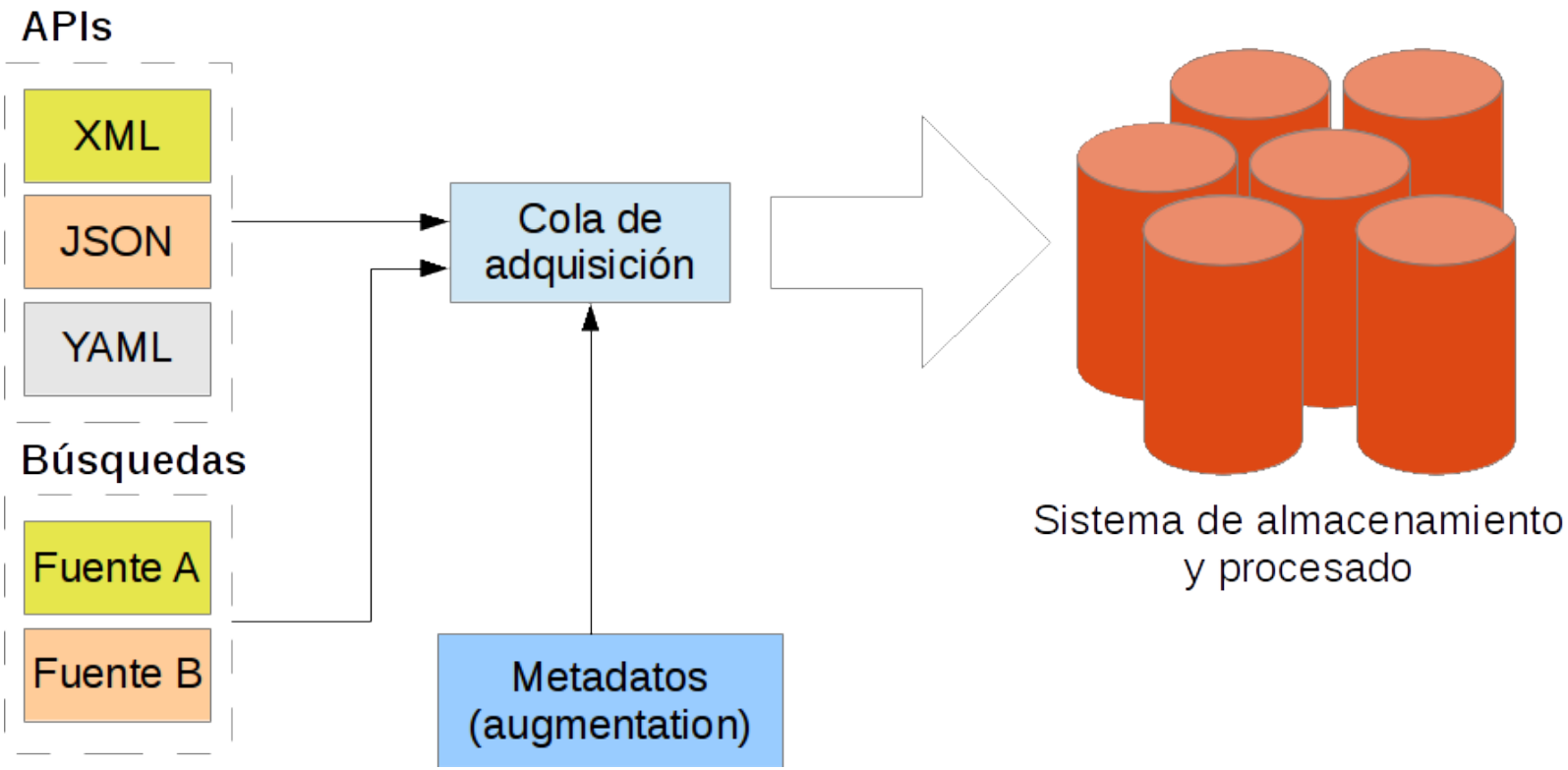
# Obtención de datos

# Obtención de datos

- Etapa crucial y con frecuencia infravalorada.
- Con frecuencia, la obtención y preparación de datos consume cerca del **85% del tiempo total** del proyecto de análisis de datos.
- Diferentes retos.
  - Multiplicidad de fuentes.
  - Métodos de obtención de datos (*scrapping*, *streaming*, APIs...).
  - Diferentes formatos de representación.
  - Consolidación de datos obtenidos.

# Obtención de datos

- Multiplicidad de fuentes



# Obtención de datos: aspectos de diseño

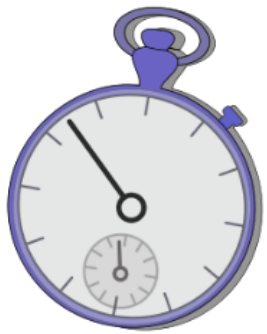
- Construir módulos intercambiables para manejar cada tipo de fuente.
- Misma interfaz de uso, ocultando peculiaridades del manejo de cada fuente o tipo de datos.
- Considerar diseños basados en colas de elementos (datos o bloques de datos de entrada) que permitan gestionar:
  - Distintas velocidades de adquisición.
  - Datos heterogéneos.
  - Mantenimiento estricto del orden de llegada.

# Obtención de datos: aspectos de diseño

- Nunca debemos asumir que las fuentes nos van a enviar los datos correctamente representados o íntegros.
- Ejemplo: Beautiful Soup.
  - Biblioteca Python para adquisición de datos HTML (y XML).
  - Soporta fallos en sintaxis HTML (o XML) de los documentos de origen.

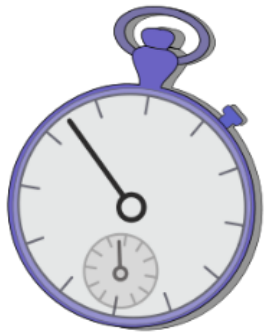
# Obtención de datos: aspectos de diseño

- Aquí importa (y mucho) la velocidad de ejecución.
  - En flujos de datos en tiempo real podemos perder datos si no los recuperamos a tiempo.
  - Los tiempos de espera para tratamiento de fuentes de gran volumen se pueden alargar demasiado (días, semanas).
- Ejemplos: lxml, UJSON (Python).



# Obtención de datos: aspectos de diseño

- Pero también hay que respetar los límites impuestos por determiandos sistemas fuente.
  - En APIs públicas, se suele limitar el número de consultas que pueden realizarse en un cierto intervalo, también la cantidad de datos devueltos por cada consulta o el rango temporal que podemos abarcar.
- Ejemplos: Twitter REST API 1.1, Facebook.



# Representación de datos

- Formatos relacionados con tecnologías web.
  - HTML, XML, JSON, YAML, etc.
- Procesamiento.
  - CSV, HDF5, ff, otros formatos específicos.
- Metadatos.
  - RDF (datos enlazados).

# Representación de datos

- Ejemplos [3]: JSON

**JSON sample** [edit]

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

# Representación de datos

- Ejemplos [3]: YAML

**YAML sample** [edit]

The above JSON code is also 100% valid YAML; however indents.[40]

```yaml
---
  firstName:  John
  lastName:  Smith
  age: 25
  address:
        streetAddress: 21 2nd Street
        city: New York
        state: NY
        postalCode: 10021

  phoneNumber:
        -
            type: home
            number: 212 555-1234
        -
            type: fax
            number: 646 555-4567
```

# Representación de datos

- Ejemplos [3]: XML

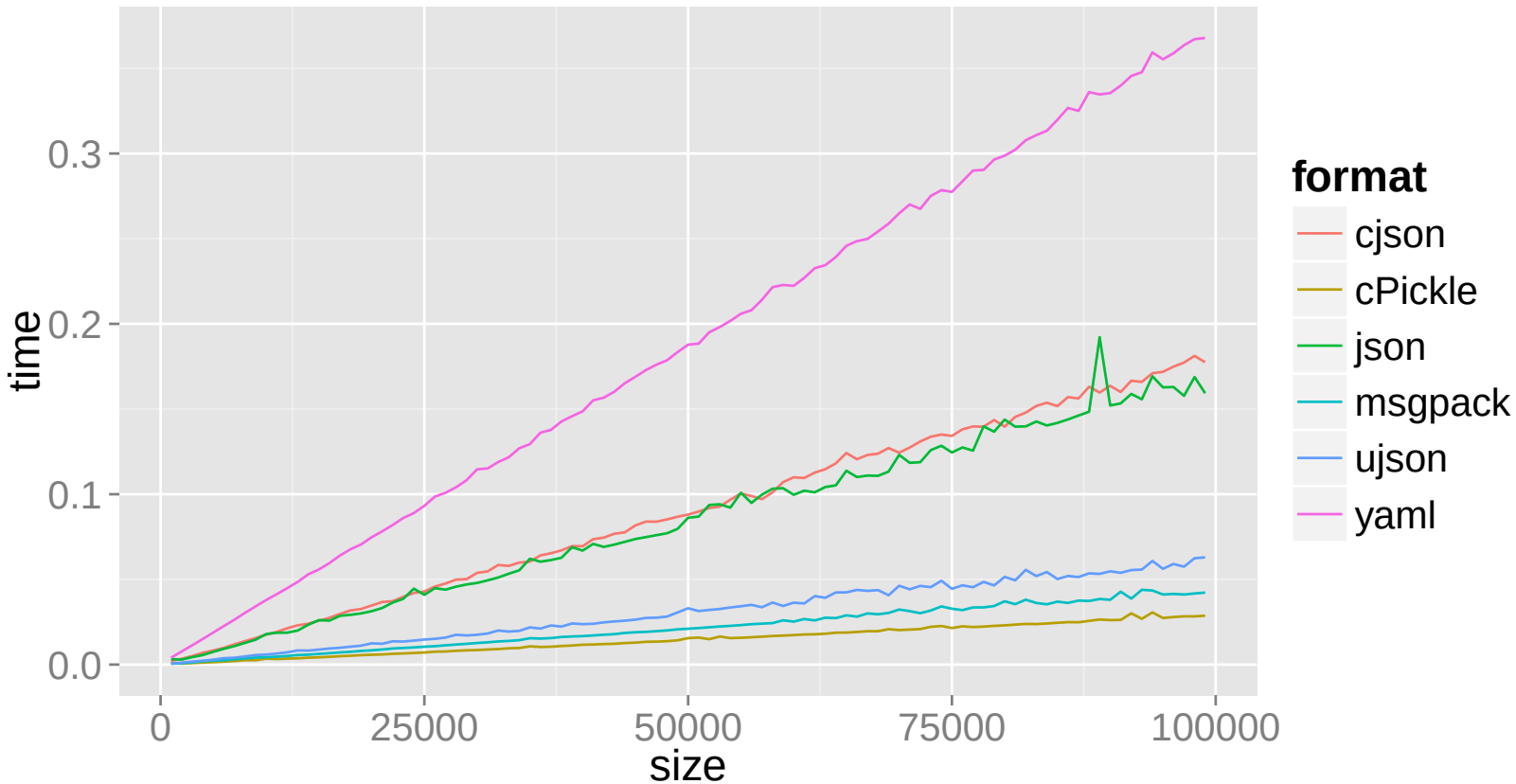
**XML samples** [edit]

```xml
<person>
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <age>25</age>
  <address>
    <streetAddress>21 2nd Street</streetAddress>
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021</postalCode>
  </address>
  <phoneNumbers>
    <phoneNumber type="home">212 555-1234</phoneNumber>
    <phoneNumber type="fax">646 555-4567</phoneNumber>
  </phoneNumbers>
</person>
```

```xml
<person firstName="John" lastName="Smith" age="25">
  <address streetAddress="21 2nd Street" city="New York" state="NY" postalCode="10021" />
  <phoneNumbers>
     <phoneNumber type="home" number="212 555-1234"/>
     <phoneNumber type="fax"  number="646 555-4567"/>
  </phoneNumbers>
</person>
```

# Representación de datos

- Benchmark bibliotecas serialización (Python) [4].



# Representación de datos: procesamiento

- Almacenamiento de estructuras de datos de gran tamaño en disco.
- Estándares
  - Hyerarchical Data Format version 5 (HDF5).
- Otros formatos específicos.
  - Paquetes R `ff`, `ffbase` o `bigmemory`.

# Representación de datos: HDF5

- Conjunto de datos jerárquicos, estructurados y autodescriptivos (metadatos).
- Capaz de escalar con facilidad al nivel de Exabyte (~1000 TB), compresión transparente, ubicación en múltiples dispositivos.
- Capacidad de indexación y E/S parcial.
  - Evitamos cargar grandes volúmenes de datos en memoria o búsquedas secuenciales.
- Bibliotecas disponibles en C, C++, Python, MATLAB, etc.

# Representación de datos: HDF5

- Recomendable cuando los datos sean [5]:
  - Grandes arrays numéricos.
  - De tipo homogéneo.
  - Que se puedan organizar jerárquicamente.
  - Con metadatos de tipo arbitrario.
- Para gestión de relaciones entre datos mejor usar bases de datos.
- Se puede usar también formatos más sencillos (e.g. CSV) para casos simples.

# Representación de datos: otros formatos

- Proyecto `ff` para el lenguaje R.
- Permite manejar grandes volúmenes de datos en R, sin necesidad de recurrir a clusters o cloud computing.
- Implementación de estructuras de datos comunes en R (ej. data frames).
- Implementación en C y C++ a bajo nivel, transparente para el usuario.
- Soporte para aplicación paralela de operaciones sobre datos en disco.

# Representación de datos: RDF

- Resource Description Framework.
- Familia de estándares de representación de metadatos promovida por W3C.
- Tripletas (sujeto-predicado-objeto) definen grafos dirigidos.
- Ofrecen información sobre ubicación y relaciones entre los datos almacenados (recursos web enlazados).
- Es posible consultar el grafo mediante el lenguaje `SPARQL`.

# Representación de datos

- Ejemplo grafo RDF.