GENERAL ASSEMBLY

INTRO TO DATA SCIENCE
LECTURE 2: GETTING DATA

OCTOBER 1, 2014 // DAT10 SF

Francesco Mosconi, PhD

DATA SCIENCE IN THE NEWS

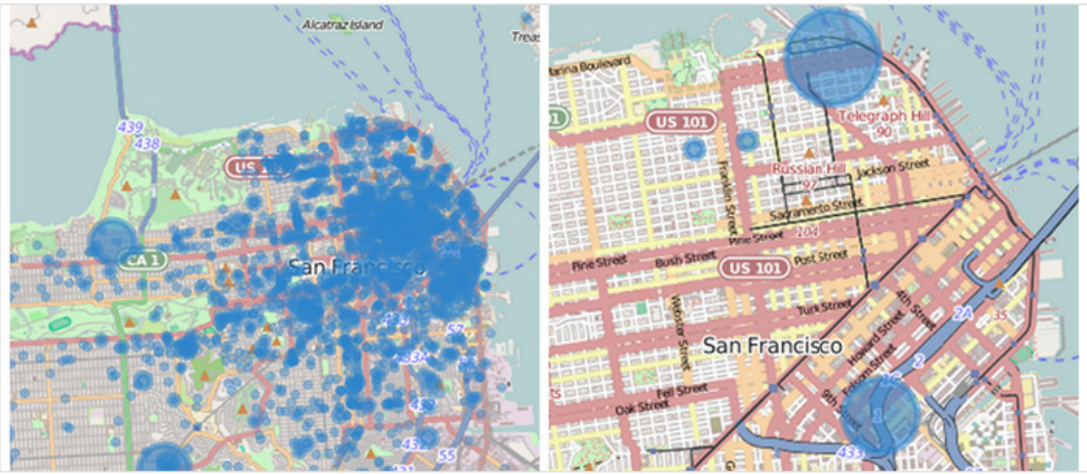
DATA SCIENCE IN THE NEWS

#UBERDATA

MAKING A BAYESIAN MODEL TO INFER UBER RIDER DESTINATIONS

SEPTEMBER 2, 2014

POSTED BY REN LU



Uber uses data science to predict where its riders want to go

Image Credit: Jason Tester Guerilla Futures/Flickr

Source: http://blog.uber.com/passenger-destinations

DATA SCIENCE IN THE NEWS

The Best Questions For A First Date

April 20th, 2011 by Christian Rudder

Tweet 3,371 Condividi 24mila

First dates are awkward. There is so much you want to know about the person across the table from you, and yet so little you can directly ask.

Source: http://blog.okcupid.com/index.php/page/3/

LAST TIME

- Data Science
- Data Scientist
- Data Mining Workflow
- Ipython
- Git

QUESTIONS?

GETTING DATA

GETTING DATA
WHERE

DATAIST (HILARY MASON & FRIENDS)

- 1. Obtain - pointing and clicking does not scale (APIs, Python, shell scripting)
- 2. Scrub - “Scrubbing data is the least sexy part of the analysis process, but often one that yields the greatest benefits” (Python, sed, awk, grep)
- 3. Explore - look at the data (visualizing, clustering, dimensionality reduction)
- 4. Model - “All models are wrong, but some are useful” / models are built to predict and interpret!
- 5. Interpret - “The purpose of computing is insight, not numbers”

Source: http://www.dataists.com/2010/09/a-taxonomy-of-data-science/

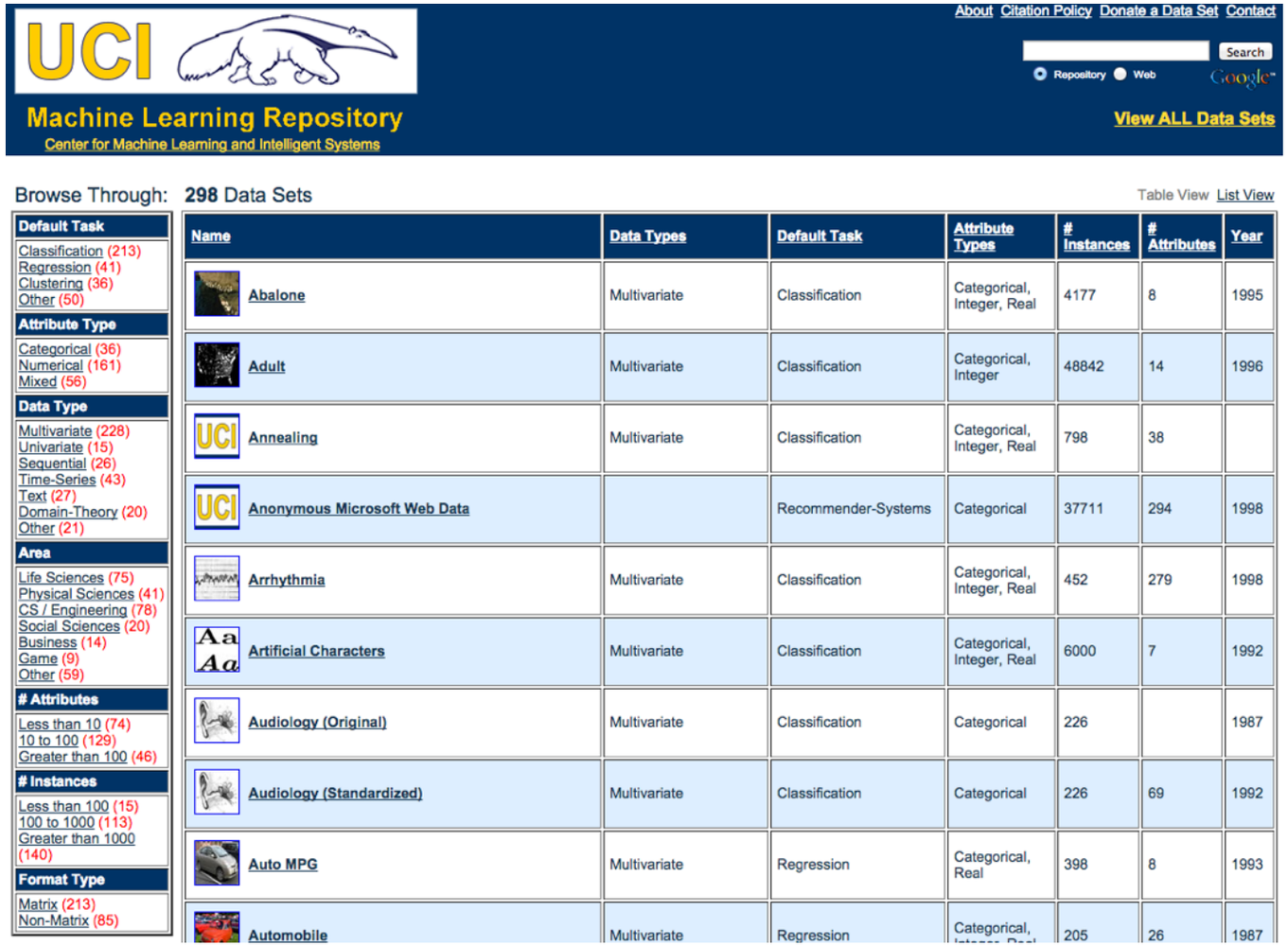
DATAIST (HILARY MASON & FRIENDS)

- 1. Obtain - pointing and clicking does not scale (APIs, Python, shell scripting)
- 2. Scrub - “Scrubbing data is the least sexy part of the analysis process, but often one that yields the greatest benefits” (Python, sed, awk, grep)
- 3. Explore - look at the data (visualizing, clustering, dimensionality reduction)
- 4. Model - “All models are wrong, but some are useful” / models are built to predict and interpret!
- 5. Interpret - “The purpose of computing is insight, not numbers”

Source: http://www.dataists.com/2010/09/a-taxonomy-of-data-science/

Where can we get data from?

UCI



Name	Data Types	Default Task	Attribute Types	# Instances	# Attributes	Year
Abalone	Multivariate	Classification	Categorical, Integer, Real	4177	8	1995
Adult	Multivariate	Classification	Categorical, Integer	48842	14	1996
Annealing	Multivariate	Classification	Categorical, Integer, Real	798	38	
Anonymous Microsoft Web Data		Recommender-Systems	Categorical	37711	294	1998
Arrhythmia	Multivariate	Classification	Categorical, Integer, Real	452	279	1998
Artificial Characters	Multivariate	Classification	Categorical, Integer, Real	6000	7	1992
Audiology (Original)	Multivariate	Classification	Categorical	226		1987
Audiology (Standardized)	Multivariate	Classification	Categorical	226	69	1992
Auto MPG	Multivariate	Regression	Categorical, Real	398	8	1993
Automobile	Multivariate	Regression	Categorical,	205	26	1987

Source: http://archive.ics.uci.edu/ml/datasets.html

USA.GOV

USA.gov Government Made Easy

Search the Government... SEARCH

Español

Follow Us:

1-800-FED-INFO (333-4636)

Services and Information | Government Agencies and Elected Officials | Blog

- Benefits, Grants, and Loans
- Businesses and Nonprofits
- Consumer Complaints and Protection
- Consumer Publications
- Disasters, Public Safety, and Laws
- Environment, Energy, and Agriculture
- Government Sales and Auctions
- Health Insurance, Nutrition, and Food Safety
- History, Genealogy, and Culture
- Immigration, Citizenship, and International
- Jobs, Training, and Education
- Mortgages, Housing, and Family
- Passports and Travel
- Public Service and Volunteerism
- Reference and General Government
- Register to Vote and Elections
- Science and Technology
- Unclaimed Money, Taxes, and Credit Reports

More for Developers

- Other USA.gov Resources
- USA.gov GitHub Account

From Other Federal Agencies

- Other Federal Government Developer Resources
- Other Federal Government GitHub Accounts

About The Data

1.USA.gov URLs are created whenever anyone shortens a .gov or .mil URL using bitly.

We provide a raw pub/sub feed of data created any time anyone clicks on a 1.USA.gov URL. The pub/sub endpoint responds to http requests for any 1.USA.gov URL and returns a stream of JSON entries, one per line, that represent real-time clicks.

If you are using the 1.USA.gov data and have questions, feedback, or want to tell us about your product, please e-mail us.

How to Access The Data

Source: http://www.usa.gov/About/developer-resources/1usagov.shtml

KAGGLE

Source: http://www.kaggle.com/

LISTS OF DATASETS CURATED BY FAMOUS DATA SCIENTISTS

- 1) Pete Skomoroch (LinkedIn) https://delicious.com/pskomoroch/dataset
- 2) Hilary Mason (Accel Partners, Bitly) https://bitly.com/bundles/hmason/1
- 3) Kevin Chai (U. of New South Wales, Sydney) http://kevinchai.net/datasets
- 4) Jeff Hammerbacher (Cloudera) http://www.quora.com/Jeff-Hammerbacher/Introduction-to-Data-Science-Data-Sets
- 5) Jerry Smith (3i-MIND) http://datascientistinsights.com/2013/10/07/data-repositories-mothers-milk-for-data-scientists/
- 6) Gregory Piatetsky-Shapiro (KDD) http://www.kdnuggets.com/datasets/index.html

- Bonus: http://www.quora.com/Data/Where-can-I-find-large-datasets-open-to-the-public

Source: http://blog.mortardata.com/post/67652898761/6-dataset-lists-curated-by-data-scientists

Pair exercise:

choose a data source and look at what data you can get
discuss how you would use the data

QUESTIONS?

GETTING DATA

WHAT

DATA FORMATS?

- Text
- CSV
- Json
- Xml
- dat
- images, binaries, etc. etc. etc.

Source: http://www.dataists.com/2010/09/a-taxonomy-of-data-science/

JSON

- JSON (JavaScript Object Notation) is a borrowed JavaScript form turned into a string that can be passed between applications.

JSON

- JSON (JavaScript Object Notation) is a borrowed JavaScript form turned into a string that can be passed between applications.

JSON

- JSON (JavaScript Object Notation) is a borrowed JavaScript form turned into a string that can be passed between applications.
- JSON is a lightweight data-interchange format.

JSON

- JSON (JavaScript Object Notation) is a borrowed JavaScript form turned into a string that can be passed between applications.
- JSON is a lightweight data-interchange format.
- JSON is easy for humans to read and write.

JSON

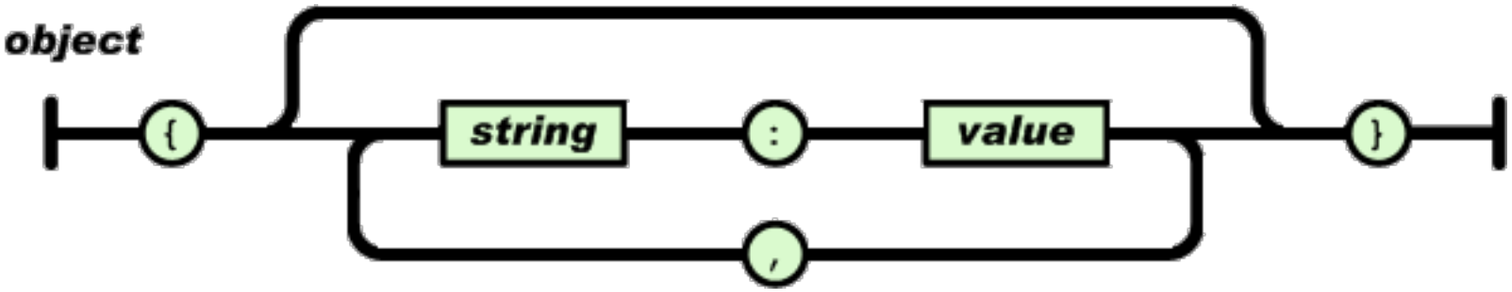
- JSON (JavaScript Object Notation) is a borrowed JavaScript form turned into a string that can be passed between applications.
- JSON is a lightweight data-interchange format.
- JSON is easy for humans to read and write.
- JSON is easy for machines to parse and generate.

JSON

- JSON (JavaScript Object Notation) is a borrowed JavaScript form turned into a string that can be passed between applications.
- JSON is a lightweight data-interchange format.
- JSON is easy for humans to read and write.
- JSON is easy for machines to parse and generate.
- JSON are passed through applications as strings, and converted into native objects per language.

JSON

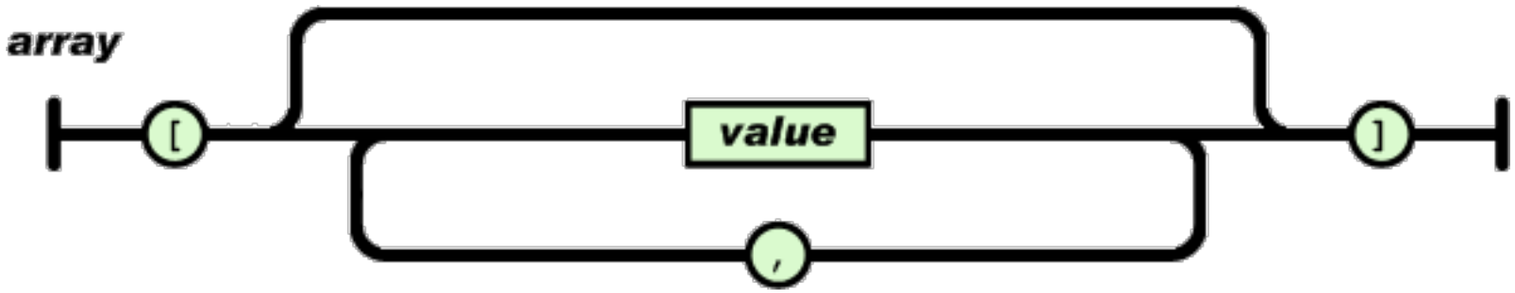
An object is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right brace). Each name is followed by : (colon) and the name/value pairs are separated by , (comma).



Source: http://json.org/

JSON

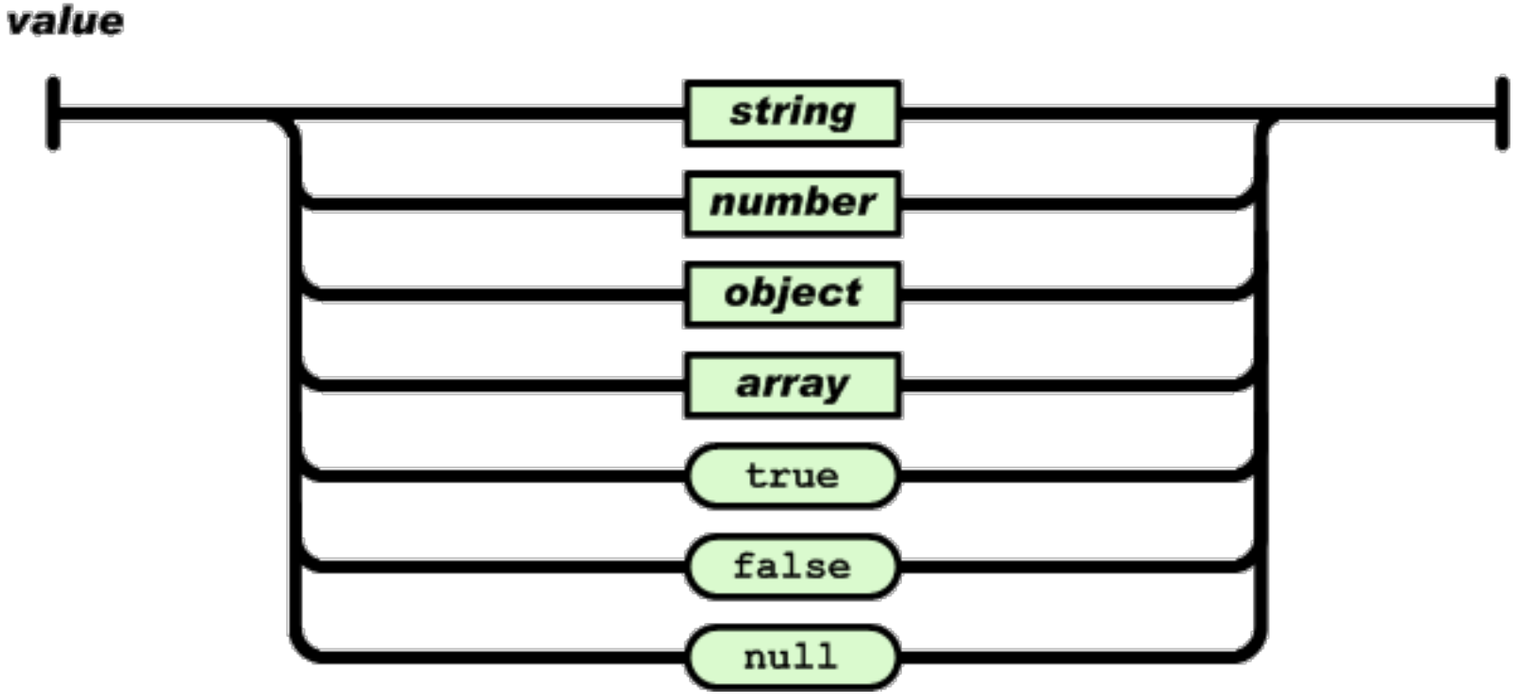
An array is an ordered collection of values. An array begins with [(left bracket) and ends with] (right bracket). Values are separated by , (comma).



Source: http://json.org/

JSON

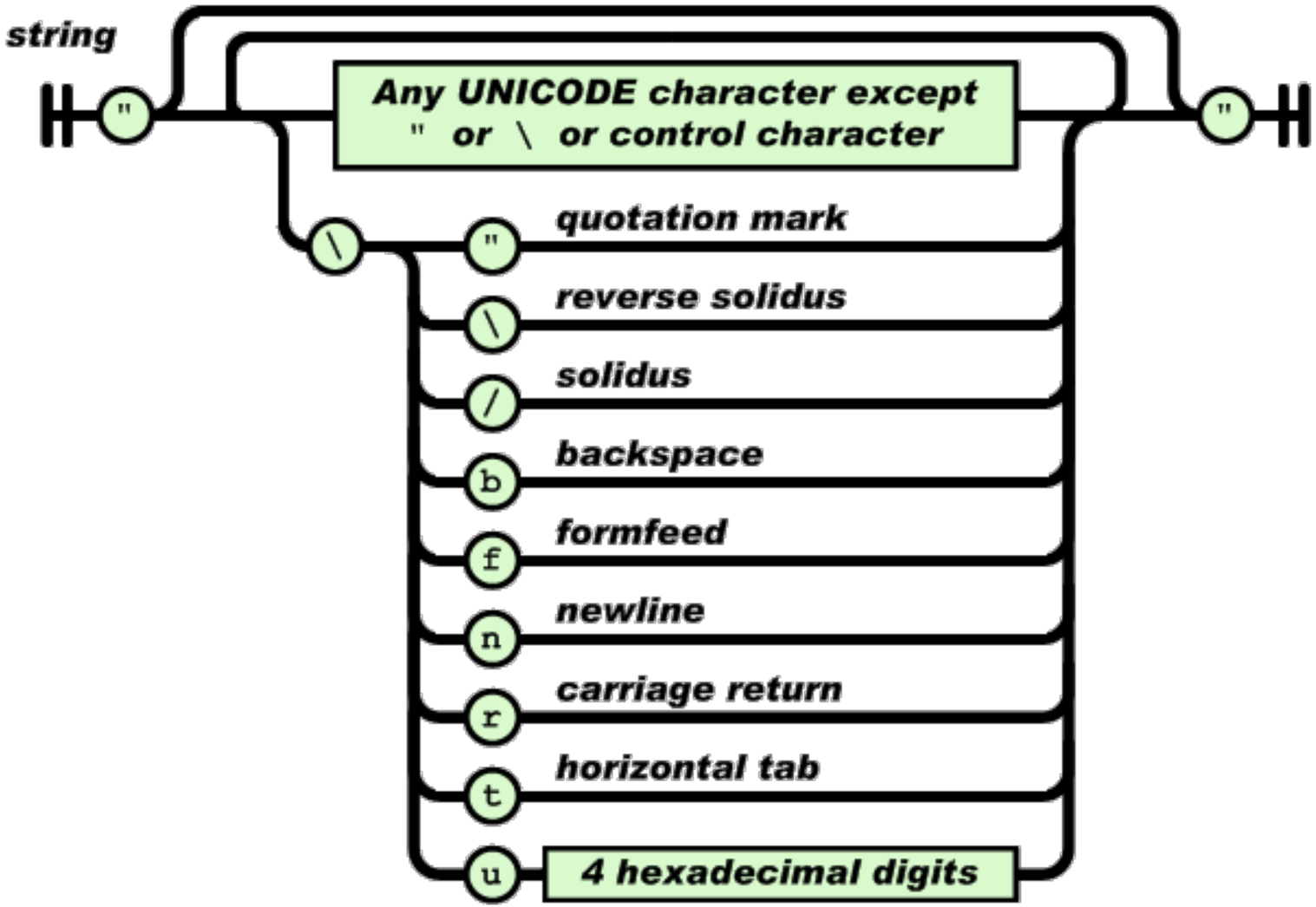
A value can be a string in double quotes, or a number, or true or false or null, or an object or an array. These structures can be nested.



Source: http://json.org/

JSON

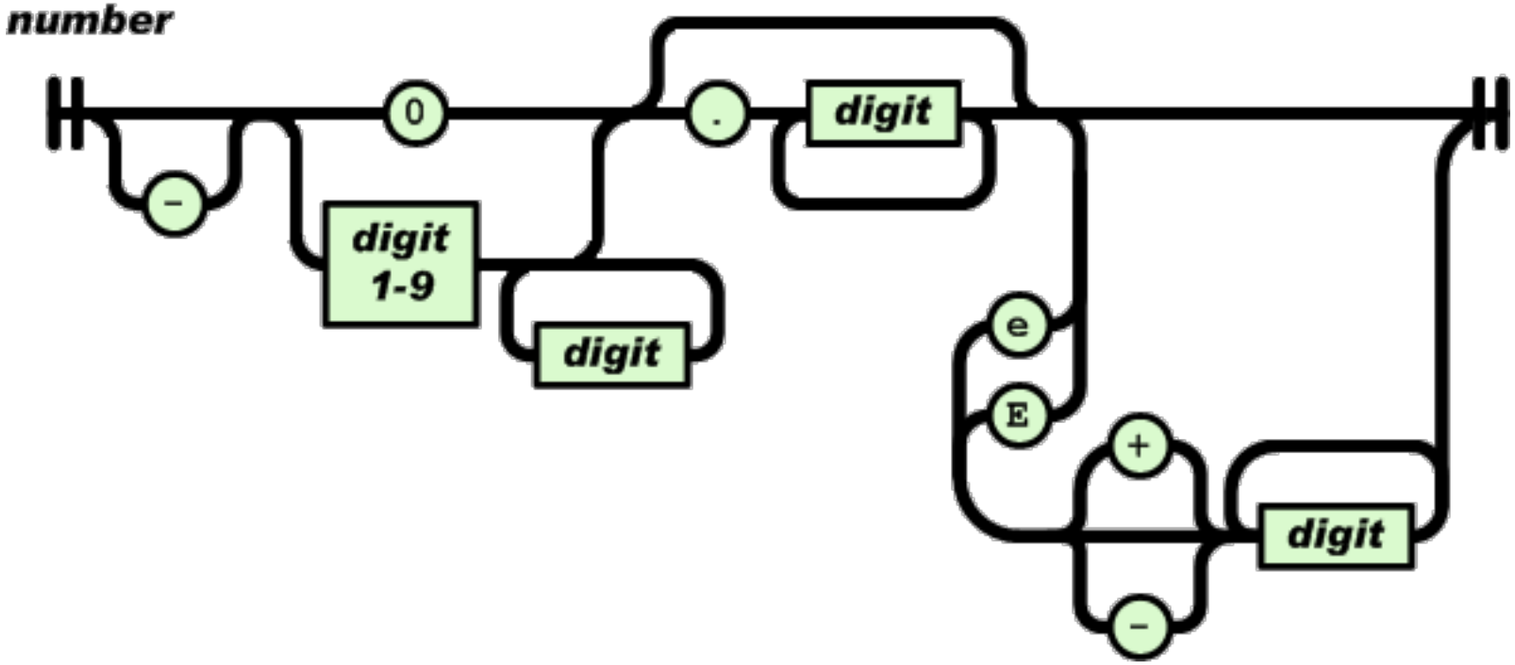
A string is a sequence of zero or more Unicode characters, wrapped in double quotes, using backslash escapes. A character is represented as a single character string. A string is very much like a C



Source: http://json.org/

JSON

A number is very much like a C or Java number, except that the octal and hexadecimal formats are not used.



Source: http://json.org/

```
>>> someFile = open('/Users/epodojil/GA_Data_Science/a.json').read()
>>> print json.dumps(someFile)
"{\n    \"glossary\": {\n        \"title\": \"example glossary\",\n    \"GlossDiv\": {\n            \"title\": \"S\",\n        \"
"SGML\",\n            \"SortAs\": \"SGML\",\n            \"GlossTerm\": \"Standard Generalized Markup Language\",\n            \"Acr
ef\": {\n                \"para\": \"A meta-markup language, used to create markup languages such as DocBook.\",\n
"GlossSee\": \"markup\"\n                }\n            }\n        }\n    }\n}"
>>> print someFile
{
    "glossary": {
        "title": "example glossary",
    "GlossDiv": {
            "title": "S",
      "GlossList": {
                "GlossEntry": {
                    "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard Generalized Markup Language",
          "Acronym": "SGML",
          "Abbrev": "ISO 8879:1986",
          "GlossDef": {
                        "para": "A meta-markup language, used to create markup languages such as DocBook.",
            "GlossSeeAlso": ["GML", "XML"]
                    },
          "GlossSee": "markup"
                }
            }
        }
    }
}
>>> print json.loads(someFile)
```

```
>>> someFile = open('/Users/epodojil/GA_Data_Science/a.json').read()
>>> print json.dumps(someFile)
"{\n    \"glossary\": {\n        \"title\": \"example glossary\",\n    \"GlossDiv\": {\n            \"title\": \"S\",\n      \"
\"SGML\",\n          \"SortAs\": \"SGML\",\n          \"GlossTerm\": \"Standard Generalized Markup Language\",\n          \"Acr
ef\": {\n                        \"para\": \"A meta-markup language, used to create markup languages such as DocBook.\",\n
\"GlossSee\": \"markup\"\n                }\n            }\n        }\n    }\n}"
>>> print someFile
{
    "glossary": {
        "title": "example glossary",
    "GlossDiv": {
            "title": "S",
      "GlossList": {
            "GlossEntry": {
                "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard Generalized Markup Language",
          "Acronym": "SGML",
          "Abbrev": "ISO 8879:1986",
          "GlossDef": {
                        "para": "A meta-markup language, used to create markup languages such as DocBook.",
            "GlossSeeAlso": ["GML", "XML"]
                    },
          "GlossSee": "markup"
                }
            }
        }
    }
}
>>> print json.loads(someFile)
{u'glossary': {u'GlossDiv': {u'GlossList': {u'GlossEntry': {u'GlossDef': {u'GlossSeeAlso': [u'GML', u'XML'], u'para': u'A meta-
': u'markup', u'Acronym': u'SGML', u'GlossTerm': u'Standard Generalized Markup Language', u'Abbrev': u'ISO 8879:1986', u'SortAs
```

String

Object

Python Dict

INTRO TO DATA SCIENCE, GETTING DATA

Json

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
  },
  "phoneNumber": [
    {
      "type": "home",
      "number": "212 555-1239"
    },
    {
      "type": "fax",
      "number": "646 555-4567"
    }
  ],
  "gender": {
    "type": "male"
  }
}
```

Source: http://en.wikipedia.org/wiki/JSON

INTRO TO DATA SCIENCE, GETTING DATA

Json

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
  },
  "phoneNumber": [
    {
      "type": "home",
      "number": "212 555-1239"
    },
    {
      "type": "fax",
      "number": "646 555-4567"
    }
  ],
  "gender": {
    "type": "male"
  }
}
```

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
  gender:
        type: male
```

Source: http://en.wikipedia.org/wiki/JSON

INTRO TO DATA SCIENCE, GETTING DATA

Json

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
  },
  "phoneNumber": [
    {
      "type": "home",
      "number": "212 555-1239"
    },
    {
      "type": "fax",
      "number": "646 555-4567"
    }
  ],
  "gender": {
    "type": "male"
  }
}
```

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>
  <gender>
    <type>male</type>
  </gender>
</person>
```

Source: http://en.wikipedia.org/wiki/JSON

QUESTIONS?

GETTING DATA
HOW

APIS

- APIs (Application Programming Interface) allow people to interact with the structures of an application to get, put, delete, or update data.

APIS

- APIs (Application Programming Interface) allow people to interact with the structures of an application to get, put, delete, or update data.

- Best practices for APIs are to use RESTful principles.

RESTFUL APIS

- Base URL and collection.
- Interactive media type (usually JSON)
- Operations (GET, PUT, POST, DELETE)
- Driven by Hypertext (http requests)

Collection

GET https://api.instagram.com/v1/users/10

Operation

GET https://api.instagram.com/v1/users/search/?q=andy

Querystring

HEADER 2

- RESTful APIs can always be accessed using cURL requests: hence why hypertext access is a requirement
- Most have language libraries to make it easier to access through the language of your choice.
- http://www.pythonapi.com/

Source:

Pair exercise:

- http://www.pythonapi.com/

choose an API and look at what data you can get
install python module to extract data
discuss how you could leverage the data from that API

QUESTIONS?