

Working with Image Data

Bootcamp Section 1

Joseph Adler, Drew Conway, Jake Hofman, Hilary Mason

February 1, 2011



Creative Commons Attribution-Share Alike 3.0

Acquiring image data

YAHOO! DEVELOPER NETWORK jake.hofman | Sign Out | Help

Home Documentation My Projects

Developer > APIs and Tools > Yahoo! Query Language > Console

YOUR YQL STATEMENT [permalink](#) [Create Query Alias](#)

```
select * from flickr.photos.search(100) where tags="kittens" and sort="interestingness-desc"
```

☐ XML ☒ JSON ☒ Diagnostics

FORMATTED TREE ☐ Wrap Text

flickr.photos.search

```
{
  "results": {
    "photo": [
      {
        "farm": "3",
        "id": "4220856234",
        "isfamily": "0",
        "isfriend": "0",
        "ispublic": "1",
        "owner": "36587311@N08",
        "secret": "029e5b8348",
        "server": "2570",
        "title": "Cuba Gallery: Cat / evil / kitten / pets / animal / cute / photography"
      },
      {
        "farm": "4"
      }
    ]
  }
}
```

THE REST QUERY [How do I use this?](#) [hide](#)

```
http://query.yahooapis.com/v1/public/yql?q=select%20*%20from%20flickr.photos.search(100)%20where%20tags%3D%22kitten
```

QUERY ALIASES

RECENT QUERIES

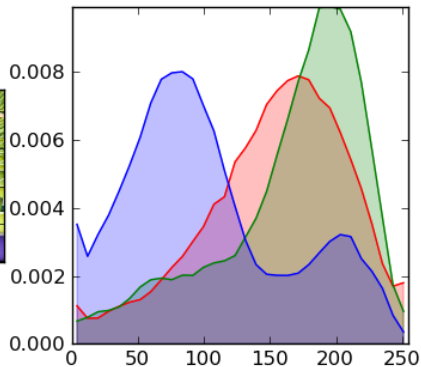
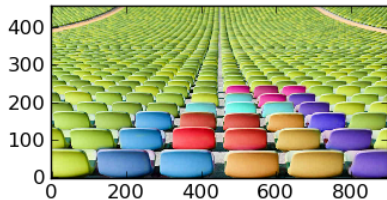
EXAMPLE QUERIES

get my friends sorted by nickname
get 10 flickr "cat" photos
get a flickr photo by photo ID
get the flickr image url by photo ID
get san francisco geo data
get san francisco woeld
get geo details about san francisco

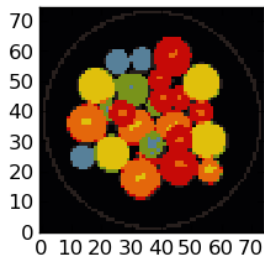
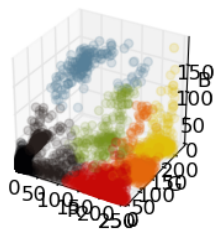
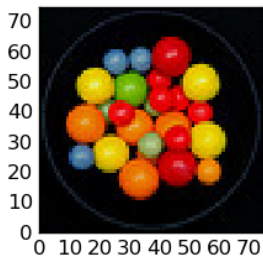
DATA TABLES (162)
[Show Community Tables](#) [What's this?](#)

[flickr.people.info2](#)
[flickr.people.publicphotos](#)
[flickr.photos.exif](#)
[flickr.photos.info](#)
[flickr.photos.interestingness](#)
[flickr.photos.recent](#)
[flickr.photos.search](#)
[flickr.photos.sizes](#)
[flickr.photosets.info](#)

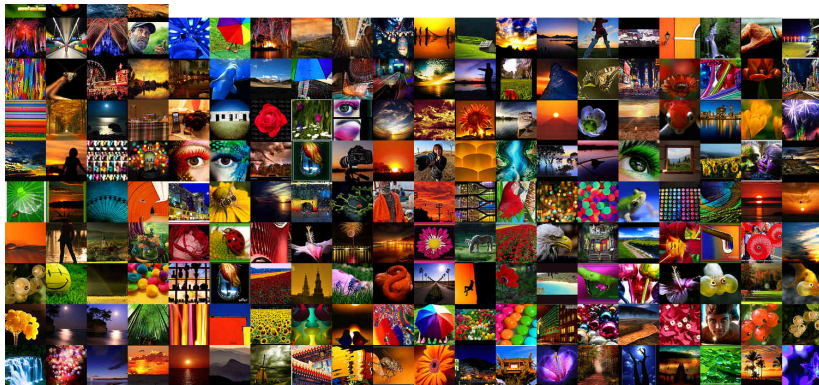
Image features



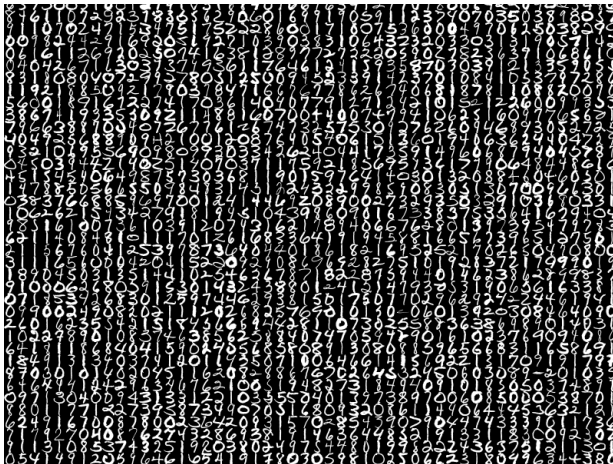
Clustering pixels



Clustering images



Classifying images



Classifying images



8

1 Acquiring image data

2 Image features

3 Clustering

4 Classification

One-liner to scrape images from a webpage

```
wget -O- http://bit.ly/gpCSQi |  
tr '\\"' '=' '\n' |  
egrep '^http.*(png|jpg|gif)' |  
xargs wget
```

One-liner to scrape images from a webpage

```
wget -O- http://bit.ly/gpCSQi |  
tr '\\"'=' '\n' |  
egrep '^http.*(png|jpg|gif)' |  
xargs wget
```

- ▶ get page source
- ▶ translate quotes and = to newlines
- ▶ match urls with image extensions
- ▶ download qualifying images

Simple screen scraping

One-liner to download ESL digit data

```
wget -Nr --level=1 --no-parent http://bit.ly/fsymq6
```

“cat flickr | xargs wget” ?

flickr®
from YAHOO!

You aren't signed in [Sign In](#) [Help](#)

[Home](#) [The Tour](#) [Sign Up](#) [Explore](#) [Upload](#)

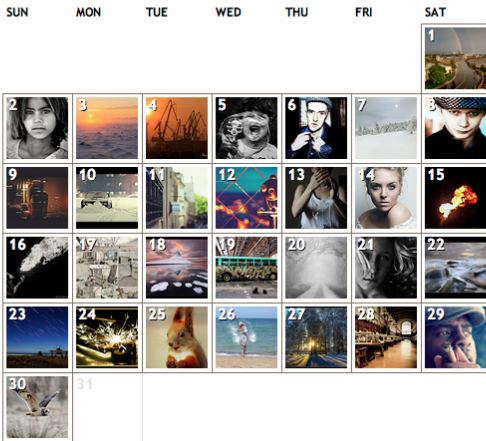
[Search](#)

Explore / Interestingness / January 2011

◀ [December 2010](#)

January 2011

February 2011 ▶





[Home](#) | [API](#) | [Community](#) | [Business](#) | [Attributions](#)

[Flickr API Changelog](#)

Getting Started

To begin using the Flickr API:


- 1 Request an [API key](#), to sign your API requests.
- 2 Read the [Community Guidelines](#) and the [API Terms of Use](#).
- 3 Build, build, build. Test, test, test.
- 4 Launch (and if it's an app of interest to the Flickr community, create a profile for your app in the [Flickr App Garden](#)).

What is YQL?

The Yahoo! Query Language is an expressive SQL-like language that lets you query, filter, and join data across Web services. With YQL, *apps run faster with fewer lines of code and a smaller network footprint.*

<http://developer.yahoo.com/yql>

¹<http://oreillynet.com/pub/e/1369>

 YOUR YQL STATEMENT [permalink](#) [Create Query Alias](#)

```
select * from flickr.photos.search(100) where tags="kittens" and sort="interestingness-desc"
```

☐ XML ☒ JSON ☒ Diagnostics

FORMATTED ☒ TREE ☐ Wrap Text

flickr.photos.search

```
{
  "results": {
    "photo": [
      {
        "farm": "3",
        "id": "4220856234",
        "isfamily": "0",
        "isfriend": "0",
        "ispublic": "1",
        "owner": "36587311@N08",
        "secret": "029e5b8348",
        "server": "2570",
        "title": "Cuba Gallery: Cat / evil / kitten / pets / animal / cute / photography"
      },
      {
        "farm": "4",

```

THE REST QUERY [How do I use this?](#) [hide](#)

```
http://query.yahooapis.com/v1/public/yql?q=select%20*%20from%20flickr.photos.search(100)%20where%20tags%3D%22kitt
```

▶ QUERY ALIASES

▶ RECENT QUERIES

▼ EXAMPLE QUERIES

get my friends sorted by nickname
get 10 flickr "cat" photos
get a flickr photo by photo ID
get the flickr image url by photo ID
get san francisco geo data
get san francisco world
get geo details about san francisco

DATA TABLES (162)

Show Community Tables

[What's this?](#)

[Filter Tables](#)

flickr.people.info2
flickr.people.publicphotos
flickr.photos.exif
flickr.photos.info
flickr.photos.interestingness
flickr.photos.recent
flickr.photos.search
flickr.photos.sizes
flickr.photosets.info

<http://developer.yahoo.com/yql/console>

Python function for public YQL queries

```
YQL_PUBLIC = 'http://query.yahooapis.com/v1/public/yql'

def yql_public(query):
    # escape query
    query_str = urlencode({'q': query, 'format': 'json'})

    # fetch results
    url = '%s%s' % (YQL_PUBLIC, query_str)
    result = urlopen(url)

    # parse json and return
    return json.load(result)['query']['results']
```

²See <http://python-yql.org/> for a more robust client

Fetch info for “interestingness” photos

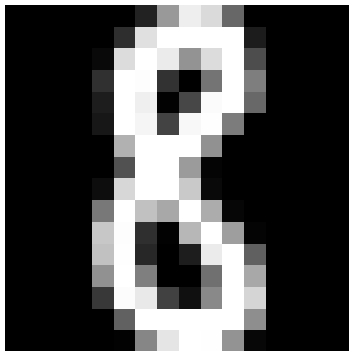
```
./simpleyql.py ‘‘select * from  
    flickr.photos.interestingness(100)’’
```

Download thumbnails for photos tagged with “vivid”

```
./download_flickr.py vivid 500
```

- 1 Acquiring image data
- 2 Image features
- 3 Clustering
- 4 Classification

Grayscale images \leftrightarrow 2-d arrays of $M \times N$ pixel intensities

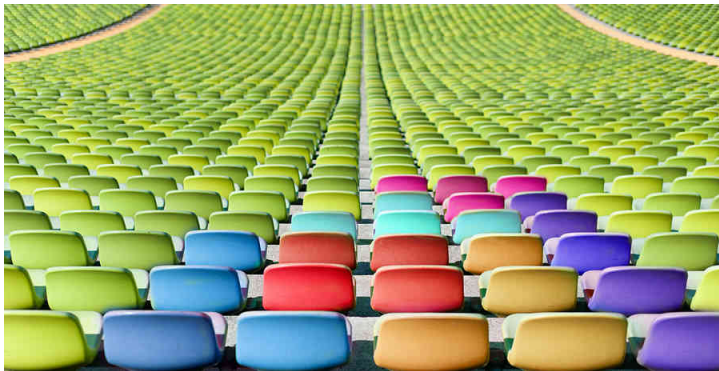


Images as arrays

Grayscale images \leftrightarrow 2-d arrays of $M \times N$ pixel intensities

[illegible]

Color images \leftrightarrow 3-d arrays of $M \times N \times 3$ RGB pixel intensities



```
import matplotlib.image as mpimg  
I = mpimg.imread('chairs.jpg')
```

Images as arrays

Color images \leftrightarrow 3-d arrays of $M \times N \times 3$ RGB pixel intensities

```
array([[106, 114, 63],
       [119, 129, 69],
       [135, 146, 77],
       ...,
       [ 12,  12,  12],
       [ 12,  12,  12],
       [ 12,  12,  12]],

      [[126, 134, 87],
       [129, 138, 83],
       [132, 143, 77],
       ...,
       [ 19,  19, 19],
       [ 19,  19, 19],
       [ 19,  19, 19]],

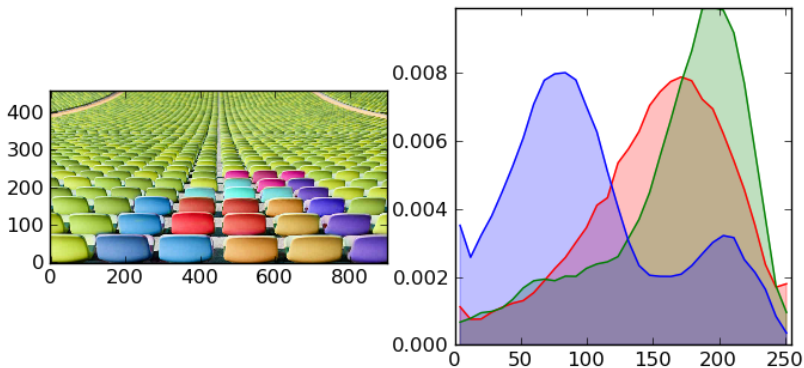
      [[124, 131, 87],
       [121, 129, 78],
       [116, 126, 63],
       ...,
       [ 31,  31, 31],
       [ 31,  31, 31],
       [ 31,  31, 31]],

      [[122, 156, 69],
       [128, 162, 75],
       [144, 178, 91],
       ...,
       [157, 187, 127],
       [160, 190, 128],
       [156, 187, 120]])
```

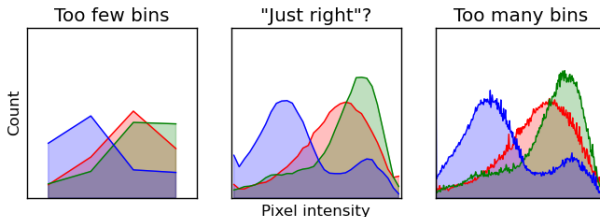
```
import matplotlib.image as mpimg
I = mpimg.imread('chairs.jpg')
```

Intensity histograms

Disregard all spatial information, simply count pixels by intensities
(e.g. lots of bright green and dark blue pixels)



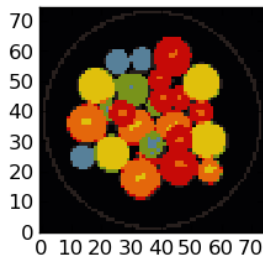
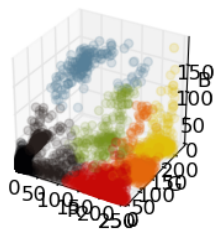
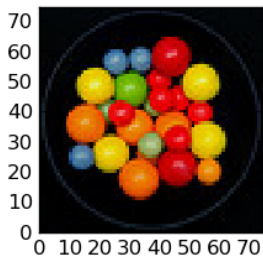
How many bins for pixel intensities?



Too many bins gives a noisy, **overly complex** representation of the data, while using **too few** bins results in an **overly simple** one

- 1 Acquiring image data
- 2 Image features
- 3 Clustering
- 4 Classification

Clustering pixels

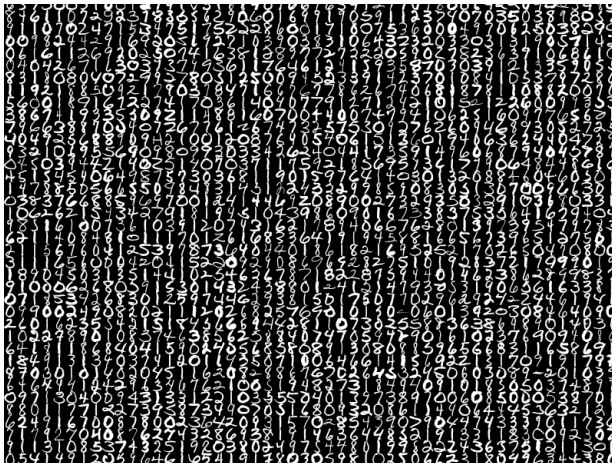


Clustering images



- 1 Acquiring image data
- 2 Image features
- 3 Clustering
- 4 Classification

Classifying images



Classifying images



8