Working with Image Data
Bootcamp Section 1

Joseph Adler, Drew Conway, Jake Hofman, Hilary Mason

February 1, 2011

Acquiring image data

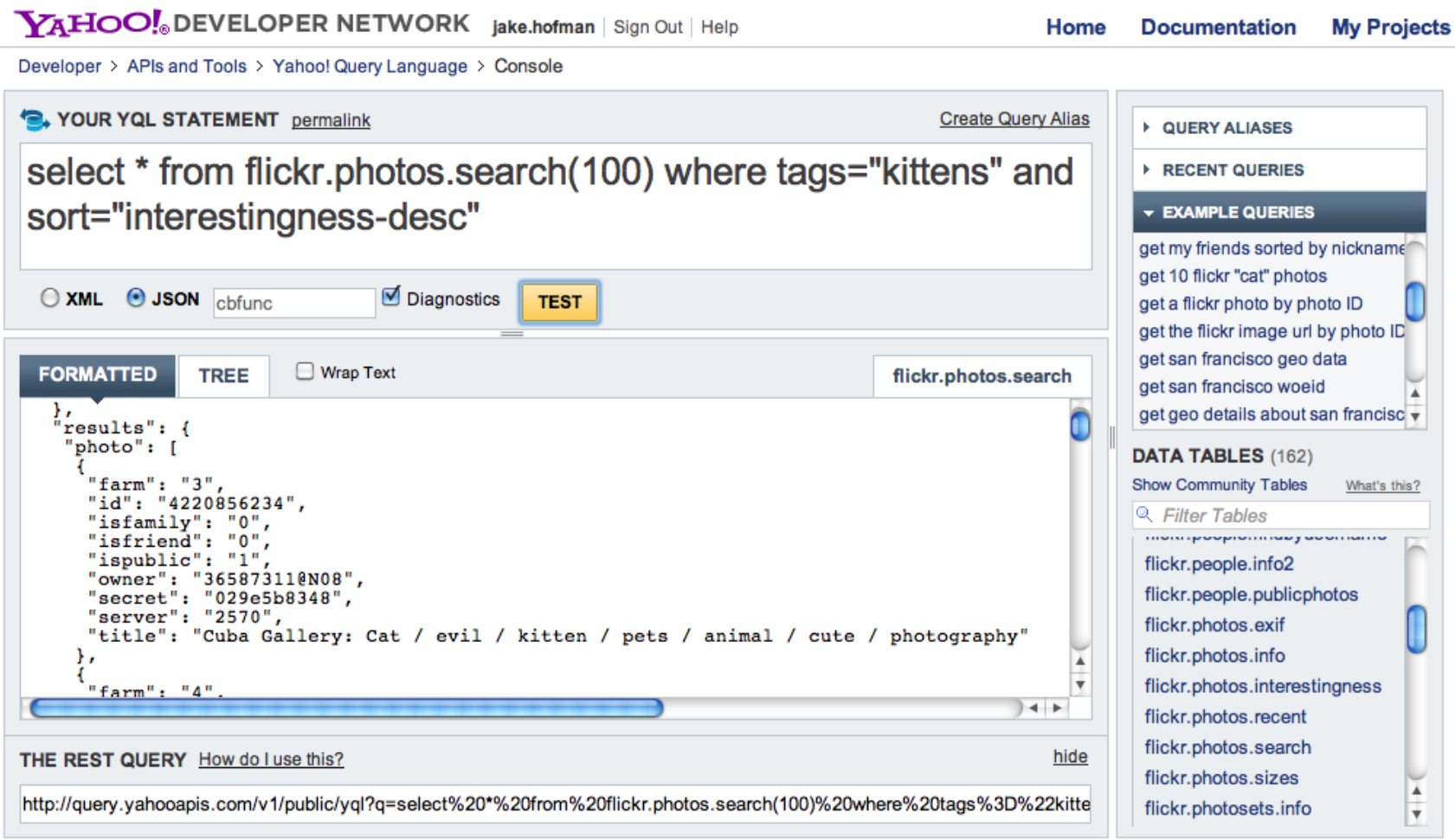
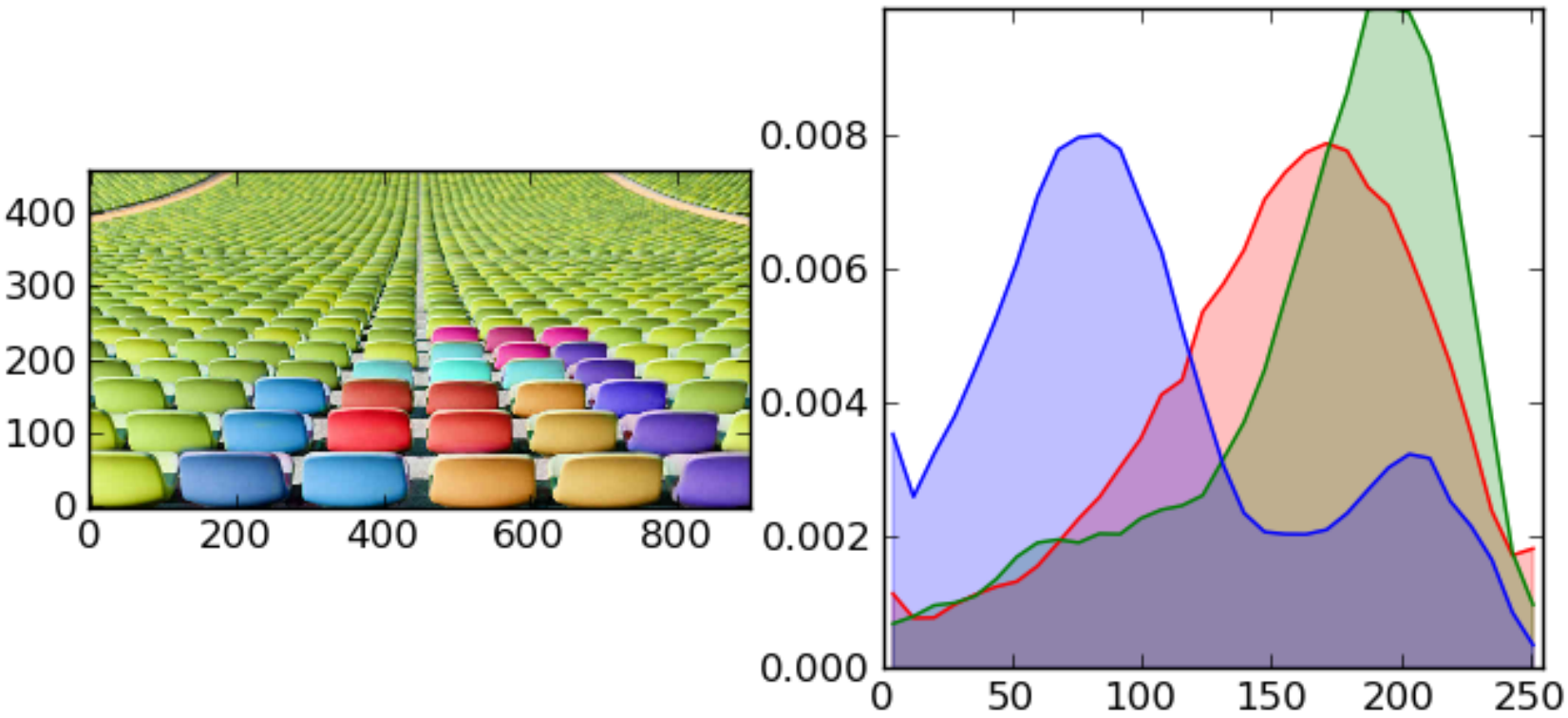
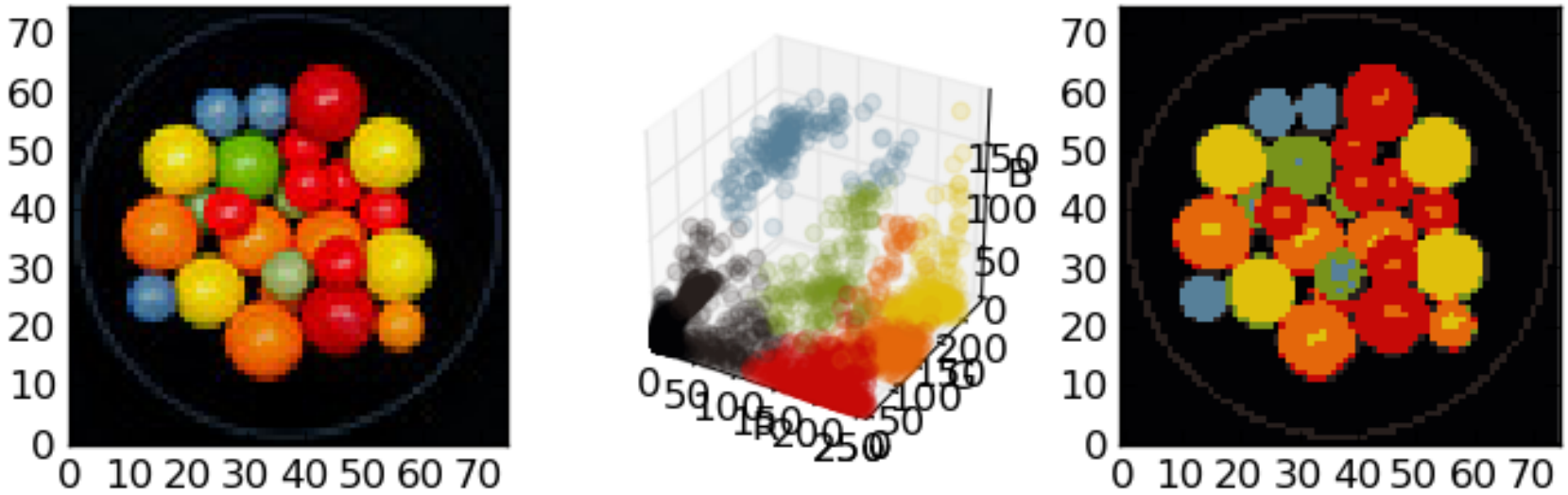


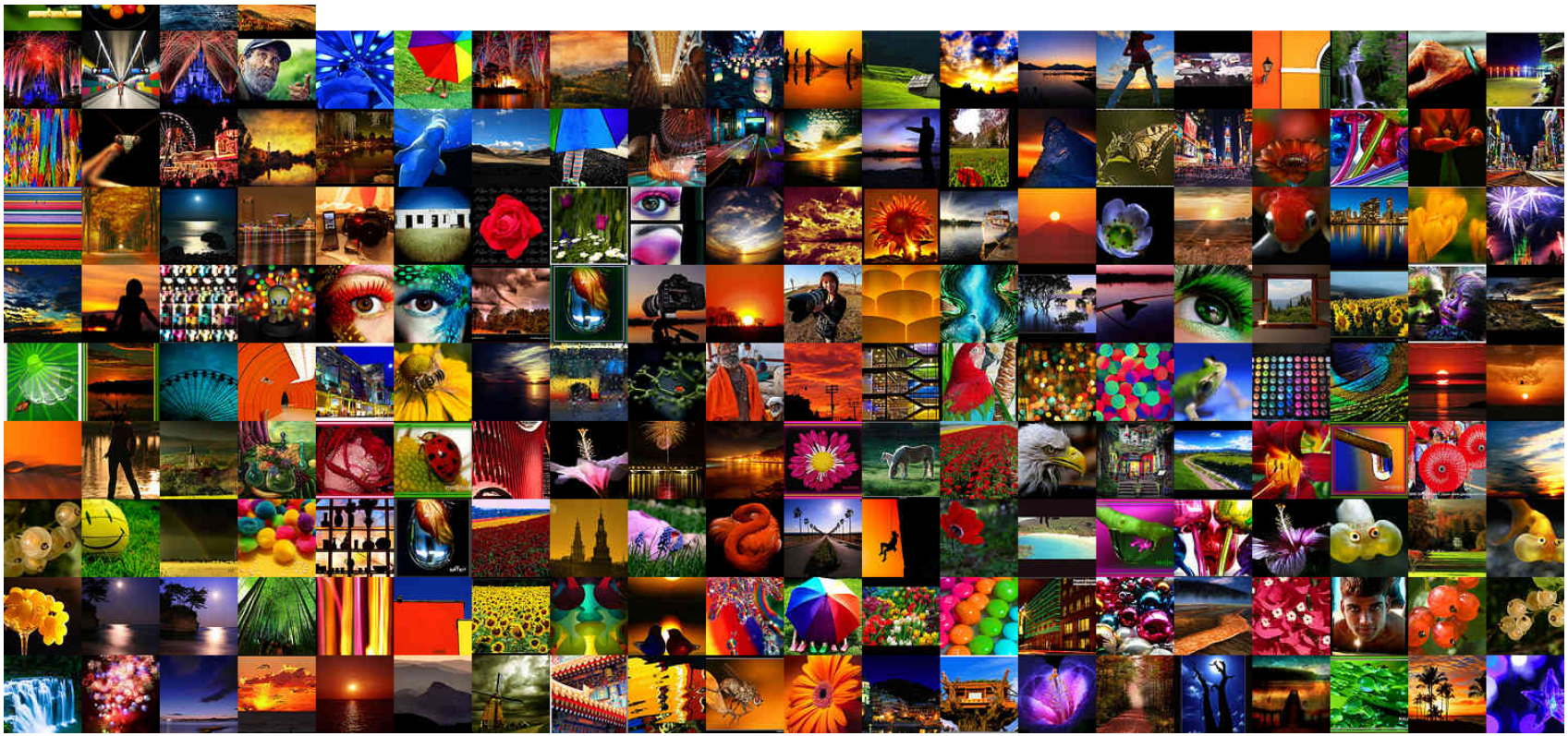
Image features



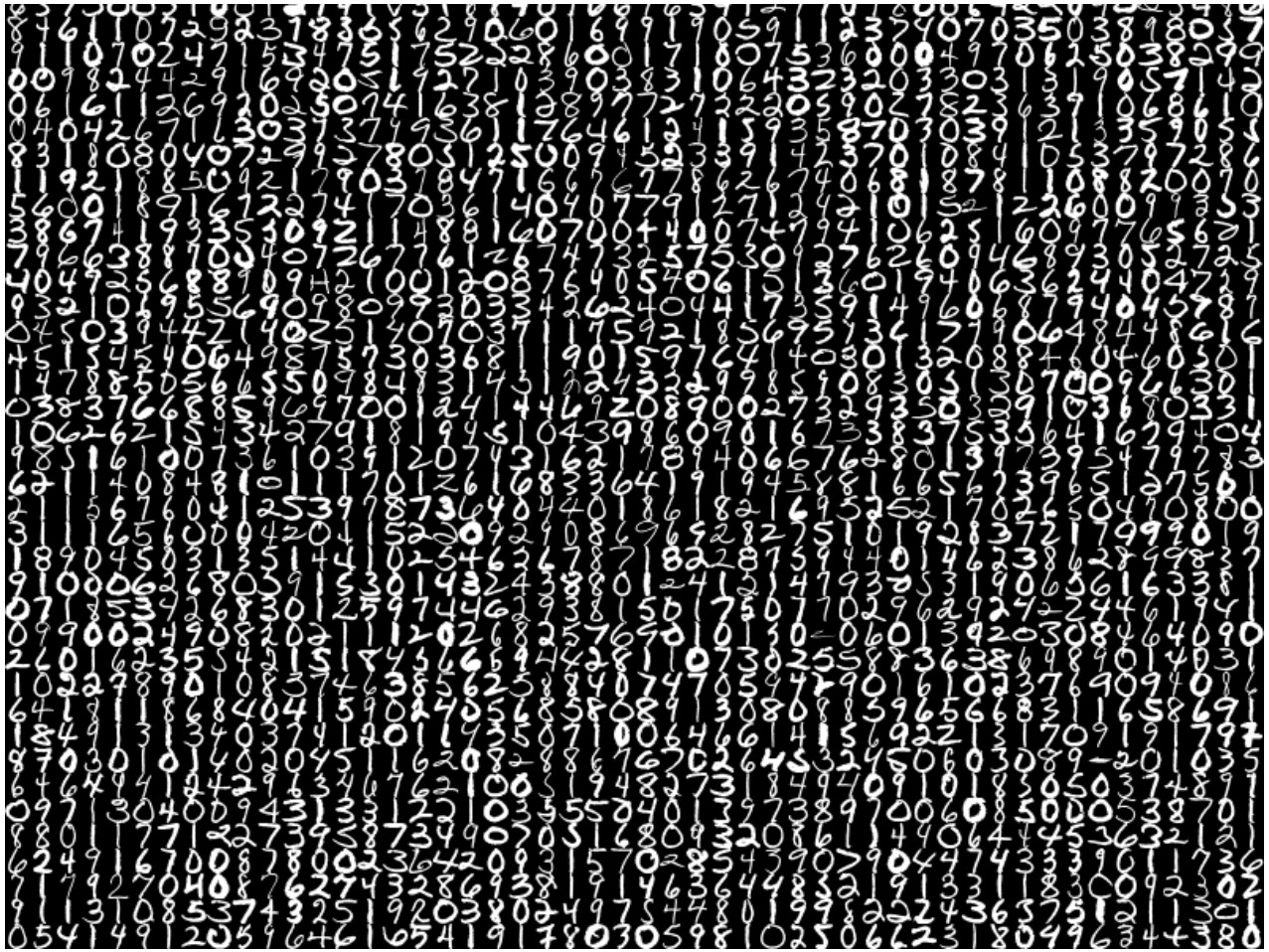
Clustering pixels



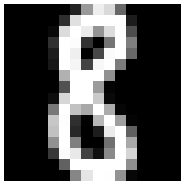
Clustering images



Classifying images



Classifying images



↓

8

Outline

Simple screen scraping

One-liner to scrape images from a webpage

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

Simple screen scraping

One-liner to scrape images from a webpage

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

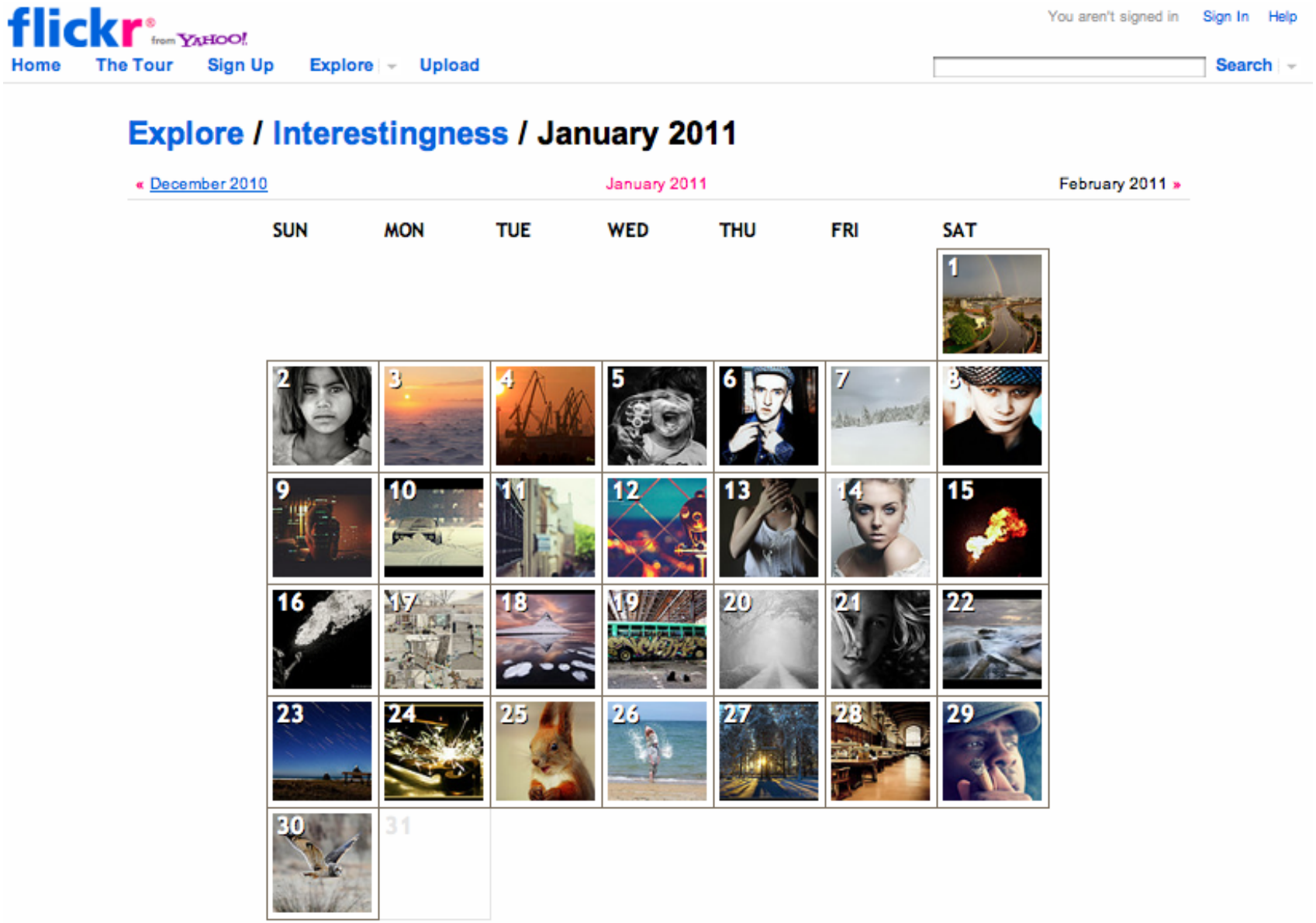
- get page source
- translate quotes and = to newlines
- match urls with image extensions
- download qualifying images

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 API

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).

YQL: SELECT * FROM Internet[1]

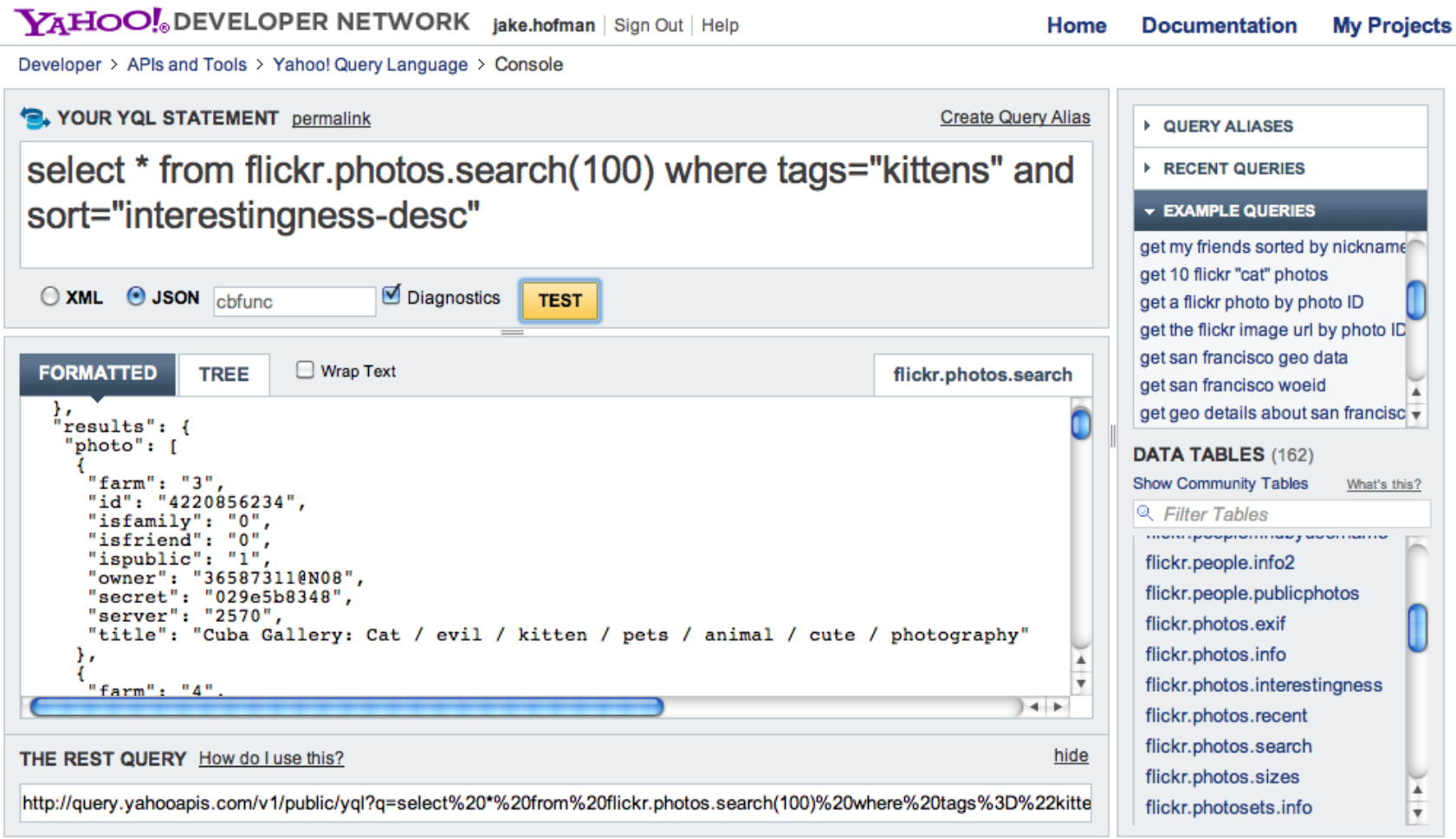
What is YQL?

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`

YQL: Console



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

YQL + Python[2]

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']
```

[2] See `http://python-yql.org/` for a more robust client

YQL + Python + Flickr

Fetch info for "interestingness" photos

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

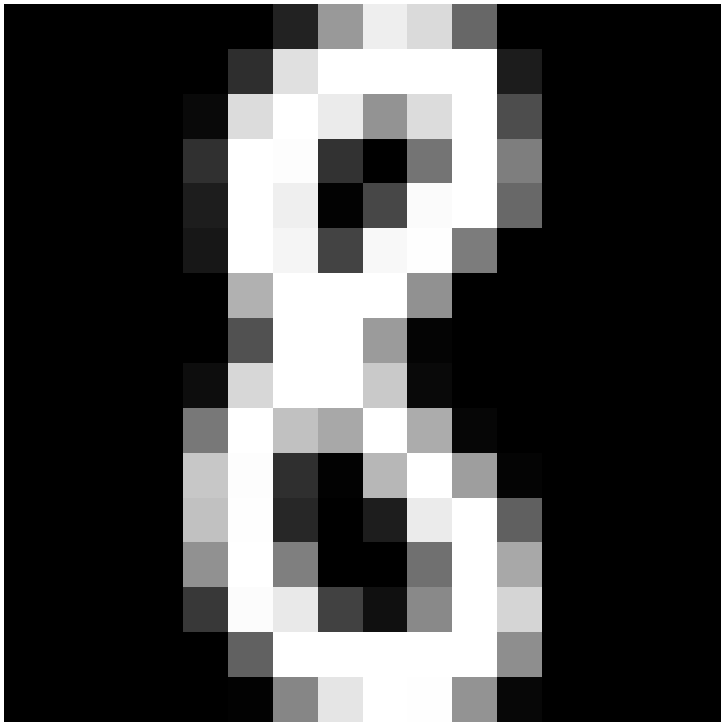
Download thumbnails for photos tagged with "vivid"

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

Outline

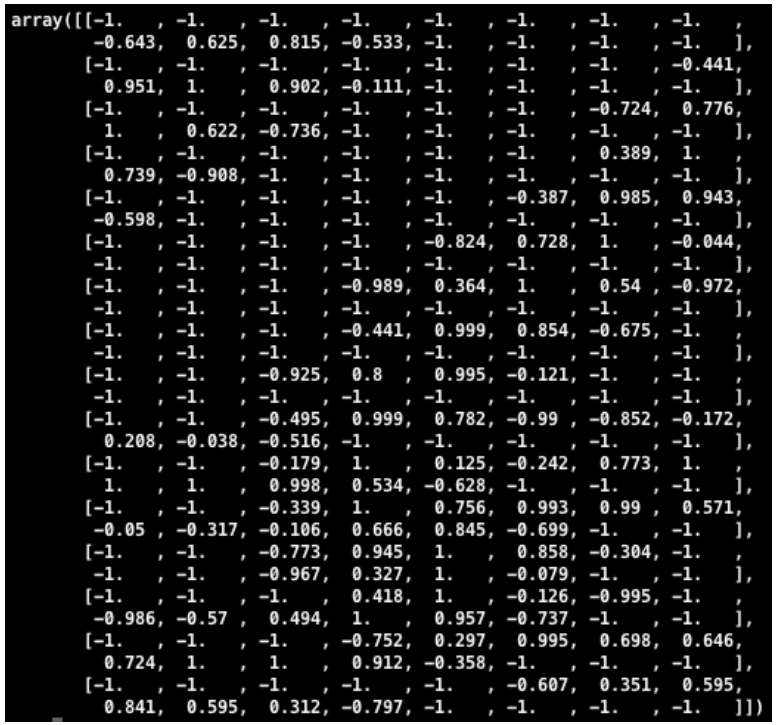
Images as arrays

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



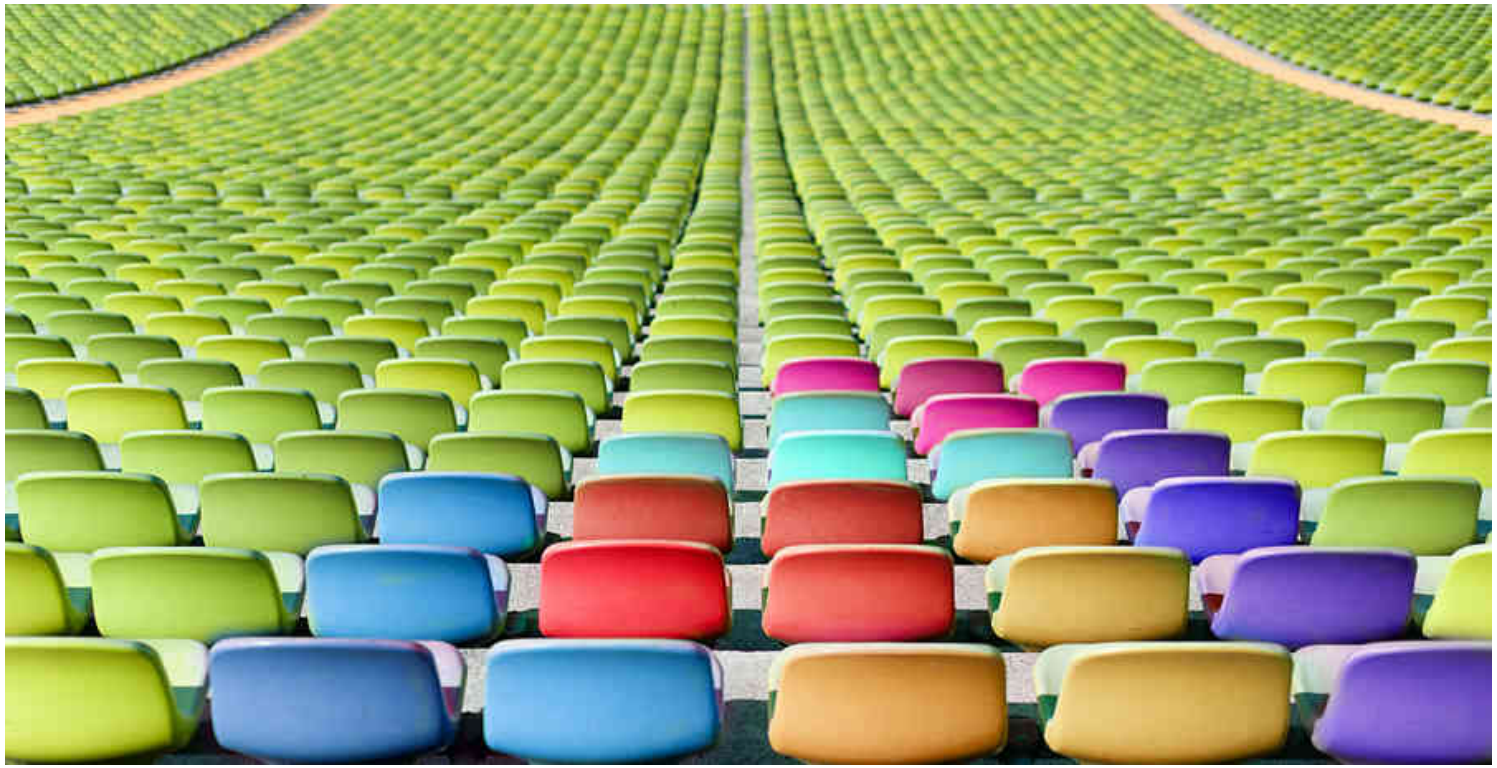
Images as arrays

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



Images as arrays

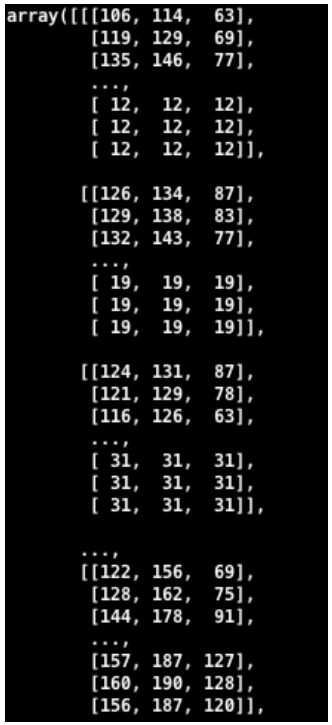
Color images $\leftrightarrow$ 3-d arrays of $M \times N \times 3$ RGB pixel intensities



```
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

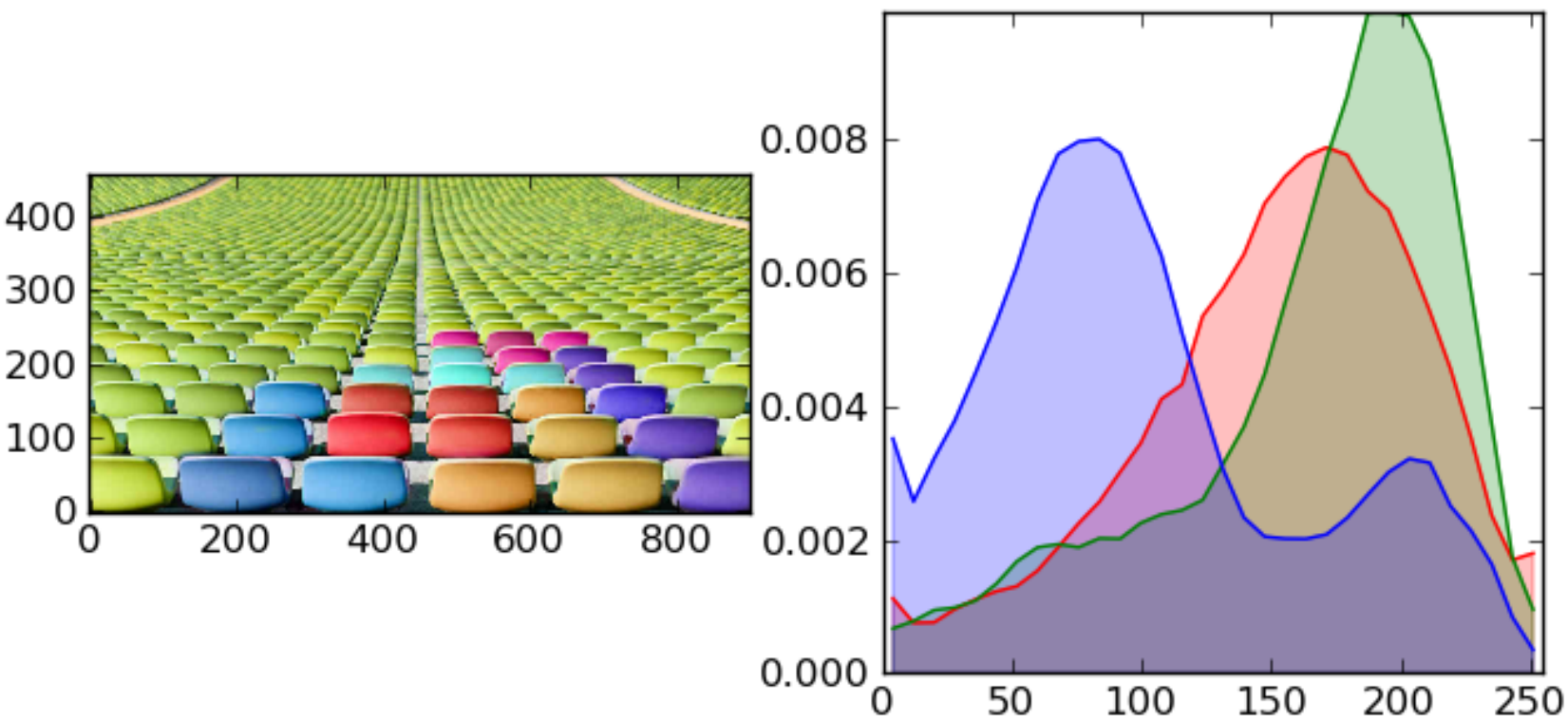


```
array([[[106, 114,  63],
        [119, 129,  69],
        [135, 146,  77],
        ...,
        [ 12,  12,  12],
        [ 12,  12,  12],
        [ 12,  12,  12]],

       [[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)



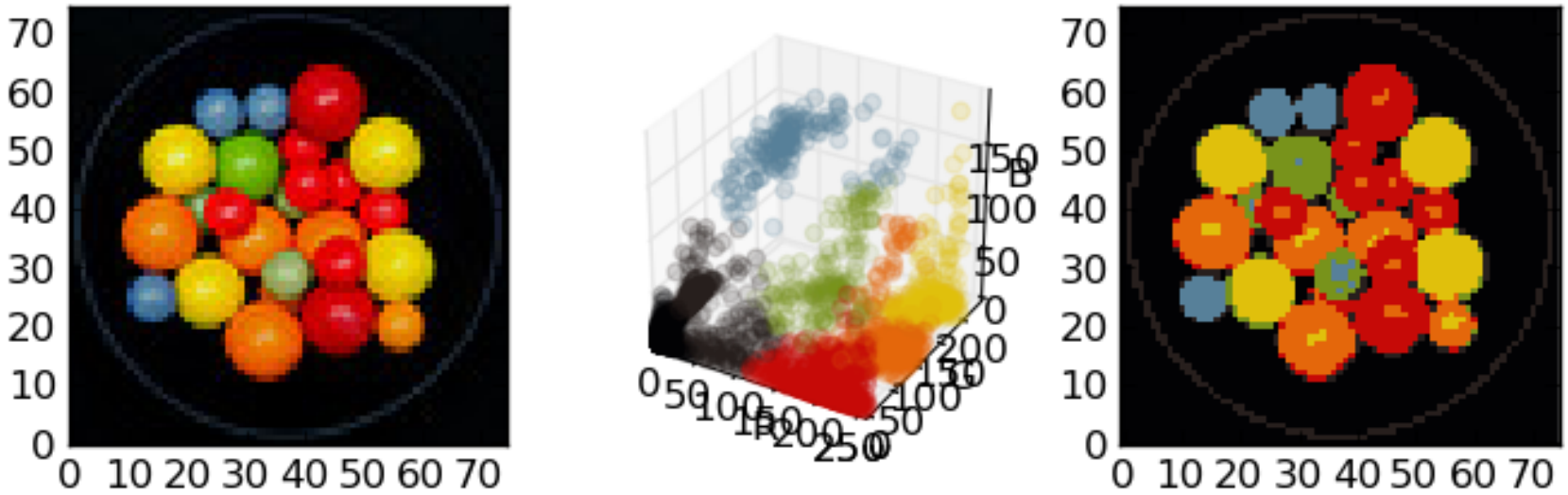
Intensity histograms

How many bins for pixel intensities?

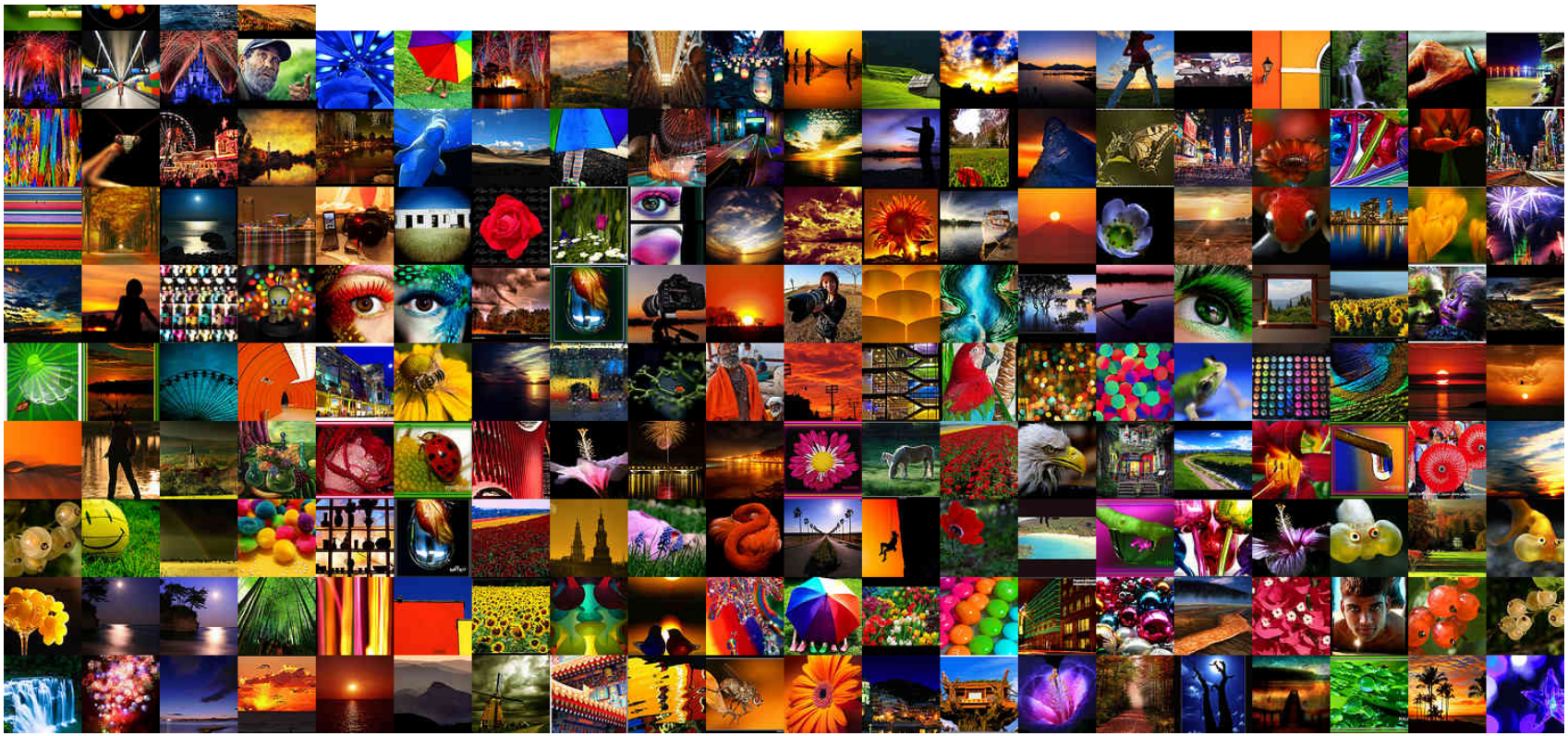
Too many bins gives a noisy, **overly complex** representation of the data, while using **too few** bins results in an **overly simple** one

Outline

Clustering pixels

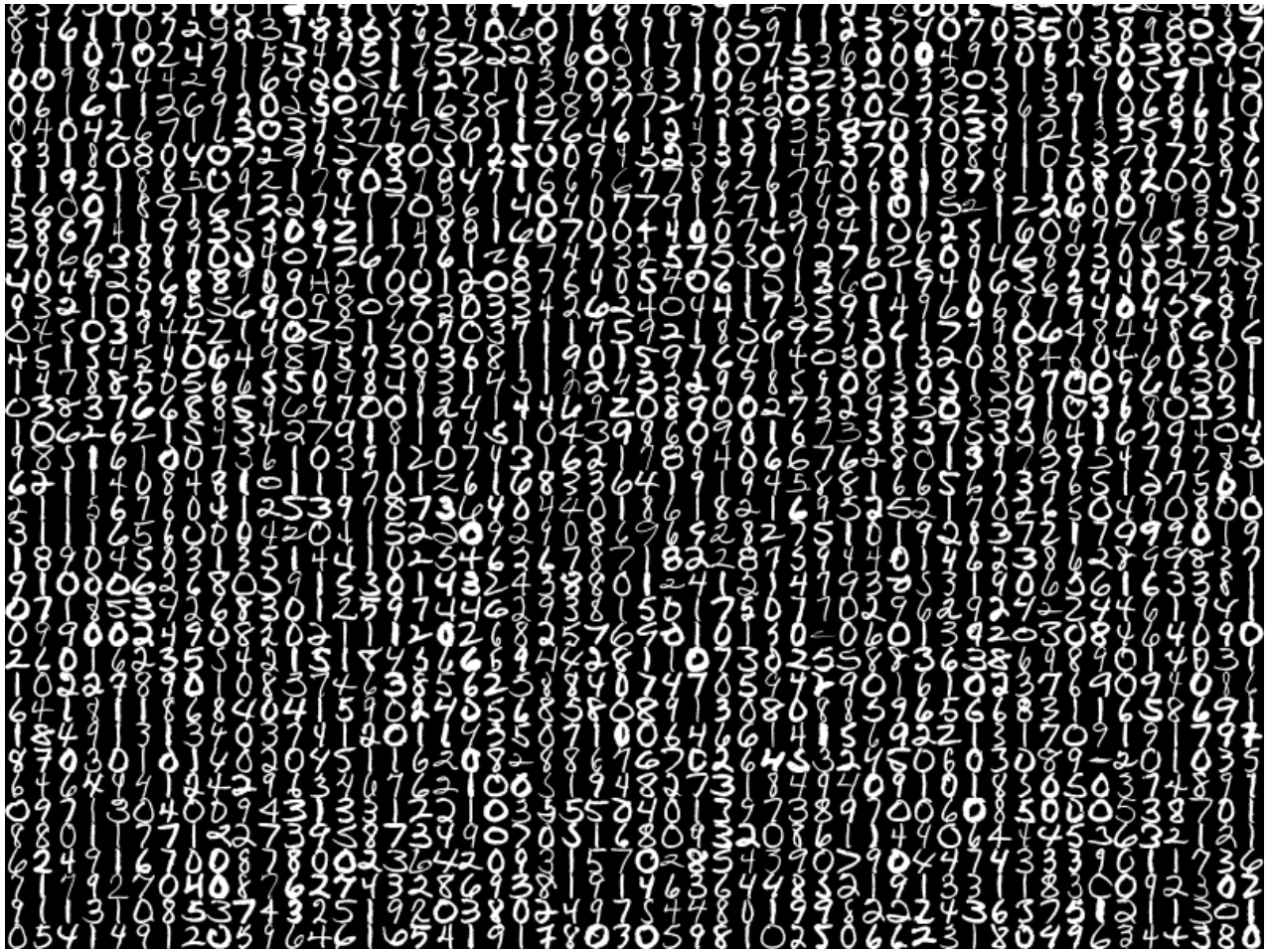


Clustering images

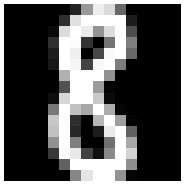


Outline

Classifying images



Classifying images



↓

8