More SQL; Web APIs

Jen Helsby and Eric Potash

Computation for Public Policy

Lecture 12: February 10, 2016

computationforpolicy.github.io

COUNT DISTINCT

```
-- count number of distinct rows in a query

SELECT COUNT(DISTINCT community_id)

FROM community_tracts;
```

CREATE TABLE

There are basic ways to create a table:

- By first creating the schema and then INSERTing data.
- By defining a new table as the result of a query.

CREATE TABLE

```
CREATE TABLE tablename (
        column1 int,
        column2 text,
        column3 boolean
);
```

CREATE TABLE AS

```
CREATE TABLE tablename AS (

    SELECT ...

);
```

CREATE TEMP TABLE

```
CREATE TEMP TABLE tablename  (

        ...

);
```

CREATE TEMP TABLE AS

```
CREATE TEMP TABLE community_populations AS (
    SELECT community_id, sum(population)
    FROM community_tracts c
    JOIN block_population b
    ON substring(c.tract_id::text from 6 for 6) =
        substring(lpad(b.census_block::text, 10, '0') for 6)
    GROUP BY 1;
);
```

CREATE TEMP TABLE AS

```
CREATE TEMP TABLE community_crimes AS (
    SELECT community_area, count(*) as crime_count
    FROM crimes c
    GROUP BY 1
);
```

Querying temporary tables

```
SELECT community_area, crime_count,
    crime_count*1.0 / population as crime_rate
FROM community_population
JOIN community_crimes
    ON community_area = community_id;
```

DROP-ing tables

```
DROP TABLE community_crimes;
```

Homework 3 updates

- Technically multiple rows in cmecomp3 correspond to a single “evaluation” but for the purposes of this exercise you can treat each row as a distinct evaluation.

- I have reimported the data with the correct types (previously everything was text).
- Note that violation_flag is a text field, with values ‘Y’, ‘N’, ‘U’
    - can convert it to a boolean by writing (violation_flag = ‘Y’)

API

“An API (application program interface) is a set of routines, protocols, and tools for building software applications. The API specifies how software components should interact.”

Web API

Web APIs are APIs that work over the web. This covers a broad range of tasks including:

- Maps
- Social Media
- Business
- Transit
- Weather
- etc.

POST

APIs can have side-effects. For example:

- The Twilio API allows programs to send text messages.
- The Facebook API allows programs to post statuses.
- There are APIs for home automation that allow programs to control hardware.

GET

However, the APIs we'll use policy and social science research will be retrieving information, not modifying it. They are a data source not unlike a file.

So why use APIs for data?

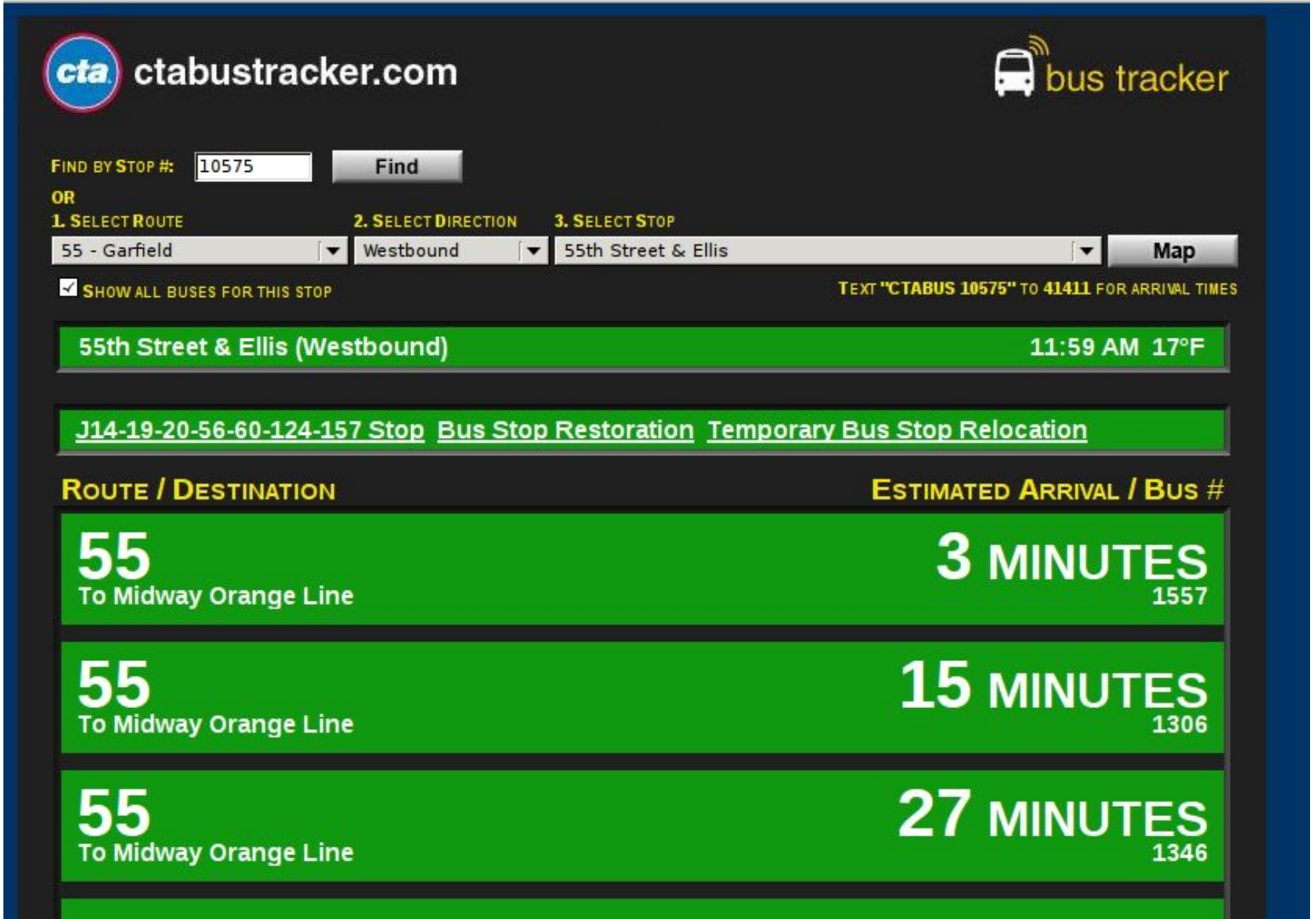
Unlike files, APIs are ideal for:

- querying small pieces of data
- data that has a complex structure
- data which is changing over time

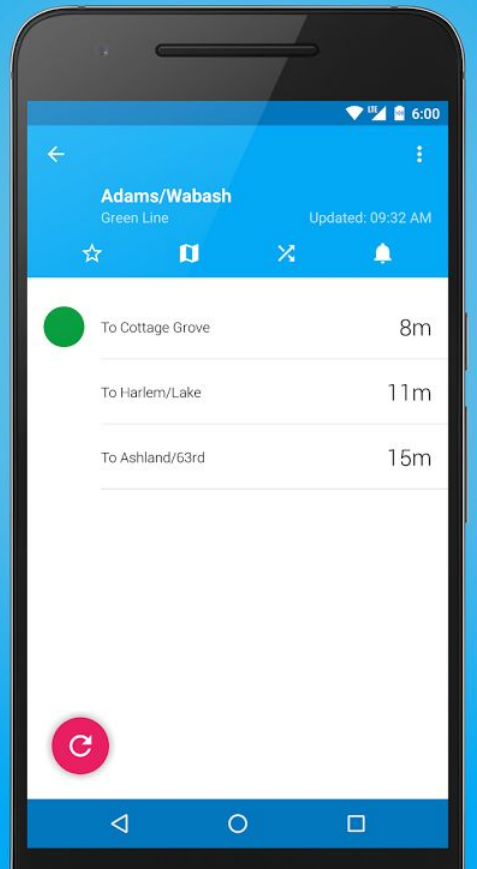
API Workflow

- Request
- Response
- Analysis

Sort of a public, user-friendly database.


ctabustracker.com
bus tracker
FIND BY STOP #: 10575
Find
OR
1. SELECT ROUTE
55 - Garfield
2. SELECT DIRECTION
Westbound
3. SELECT STOP
55th Street & Ellis
Map
SHOW ALL BUSES FOR THIS STOP
TEXT "CTABUS 10575" TO 41411 FOR ARRIVAL TIMES
55th Street & Ellis (Westbound)
11:59 AM 17°F
J14-19-20-56-60-124-157 Stop
Bus Stop Restoration
Temporary Bus Stop Relocation
ROUTE / DESTINATION
ESTIMATED ARRIVAL / BUS #
55
To Midway Orange Line
3 MINUTES
1557
55
To Midway Orange Line
15 MINUTES
1306
55
To Midway Orange Line
27 MINUTES
1346

Get quick arrival times right from the CTA



CTA Bus API

CTA Bus Time Endpoint

Base URL: http://www.ctabustracker.com/bustime/api/v1/gettime

Parameters:

Name	Value	Description
Key	string (required)	25-digit Bus Tracker API access key.

Response

A well-formed XML document containing the current system time will be returned as a response to **gettime**.

Response Fields:

Name	Description
bustime-response	Root element of the response document.
Error	Child element of the root element. Contains a message if the processing of the request resulted in an error.
Tm	Child element of the root element containing the current system date and (local) time. Date and time is represented in the following format: YYYYMMDD HH:MM:SS. Month is represented as two digits where January is equal to "01" and December is equal to "12". Time is represented using a 24-hour clock.

CTA Bus Prediction Endpoint

Predictions

Base URL: http://www.ctabustracker.com/bustime/api/v1/getpredictions

Parameters:

Name	Value	Description
key	string (required)	25-digit Bus Tracker API access key.
stpid	comma-delimited list of stop IDs (not available with **vid** parameter)	Set of one or more stop IDs whose predictions are to be returned. For example: 5029,1392,2019,4367 will return predictions for the four stops. A maximum of 10 identifiers can be specified.
rt	comma-delimited list of route designators (optional, available with **stpid** parameter)	Set of one or more route designators for which matching predictions are to be returned.
vid	comma-delimited list of vehicle IDs (not available with **stpid** parameter)	Set of one or more vehicle IDs whose predictions should be returned. For example: 509,392,201,4367 will return predictions for four vehicles. A maximum of 10 identifiers can be specified.
top	number (optional)	Maximum number of predictions to be returned.

CTA Bus Prediction Endpoint

Response Fields:

Name	Description
bustime-response	Root element of the response document.
error	Child element of the root element. Message if the processing of the request resulted in an error.
prd	Child element of the root element. Encapsulates a predicted arrival or

	departure time for the specified set of stops or vehicles.
tmstmp	Child element of the **prd** element. Date and time (local) the prediction was generated. Date and time is represented in the following format: YYYYMMDD HH:MM. Month is represented as two digits where January is equal to "01" and December is equal to "12". Time is represented using a 24-hour clock.
typ	Child element of the **prd** element. Type of prediction. 'A' for an arrival prediction (prediction of when the vehicle will arrive at this stop). 'D' for a departure prediction (prediction of when the vehicle will depart this stop, if applicable). Predictions made for first stops of a route or layovers are examples of departure predictions.
stpid	Child element of the **prd** element. Unique identifier representing the stop for which this prediction was generated.
stpnm	Child element of the **prd** element. Display name of the stop for which this prediction was generated.
vid	Child element of the **prd** element. Unique ID of the vehicle for which this prediction was generated.

dstp	Child element of the **prd** element. Linear distance (feet) left to be traveled by the vehicle before it reaches the stop associated with this prediction.
rt	Child element of the **prd** element. Alphanumeric designator of the route (ex. "20" or "X20") for which this prediction was generated.
rtdir	Child element of the **prd** element. Direction of travel of the route associated with this prediction (ex. "East Bound").
des	Child element of the **prd** element. Final destination of the vehicle associated with this prediction.
prdtm	Child element of the **prd** element. Predicted date and time (local) of a vehicle's arrival or departure to the stop associated with this prediction. Date and time is represented in the following format: YYYYMMDD HH:MM. Month is represented as two digits where January is equal to "01" and December is equal to "12". Time is represented using a 24-hour clock.
dly	Child element of the **prd** element. "true" if the vehicle is delayed. The **dly** element is only present if the vehicle that generated this prediction is delayed.

Request:

http://www.ctabustracker.com/bustime/api/v1/getpredictions?key=89dj2he89d8j3j3ksjhdue93j&rt=20&stpid=456

Response:

```
<?xml version="1.0"?>
<bustime-response>
<tm></tm>
      <prd>
            <tmstmp>20090611 14:34</tmstmp>
            <typ>A</typ>
            <stpid>456</stpid>
            <stpnm>Madison & Jefferson</stpnm>
            <vid>2013</vid>
            <dstp>891</dstp>
            <rt>20</rt>
            <rtdir>West Bound</rtdir>
            <rtdst>Austin</rtdst>
            <prdtm>20090611 14:40</prdtm>
      </prd>
      <prd>
            <tmstmp>20090611 14:34</tmstmp>
            <typ>A</typ>
            <stpid>456</stpid>
            <stpnm>Madison & Jefferson</stpnm>
            <vid>6435</vid>
            <dstp>1587</dstp>
            <rt>20</rt>
            <rtdir>West Bound</rtdir>
            <rtdst>Austin</rtdst>
            <prdtm>20090611 14:48</prdtm>
      </prd>
</bustime-response>
```

Chicago, IL

U.S. Cellular Field/Bridgeport | Report | Change Station

Forecast | History | Calendar | Rain / Snow | Health

Elev **594** ft **41.83** °N, **87.64** °W | Updated **1** sec ago

19.8 °F

Clear

Feels Like **20** °F

Wind from **WNW**
Gusts **9** mph

Today is forecast to be ***NEARLY THE SAME*** temperature as yesterday.

Today
High **21** | Low **16** °F
0% Chance of Precip.

Yesterday
High **19.9** | Low **11.7** °F
Precip. **0** in

Pressure	**30.31** in
Visibility	**10.0** miles
Clouds	**Few 3500** ft
Windchill	**19** °F
Dew Point	**-2** °F
Humidity	**40**%
Rainfall	**0.00** in
Snow Depth	*Not available.*

Sun & Moon

6:51 am
5:18 pm

Waxing Crescent,
13% visible

```
METAR KMDW 111753Z 35007KT 10SM
FEW035 M08/M18 A3032 RMK A02
SLP288 4/001 T10781183 11078 21128
50005
```

Weatherunderground API

needs.

STRATUS PLAN	CUMULUS PLAN	ANVIL PLAN
Geolookup Autocomplete Current conditions 3-day forecast summary Astronomy Almanac for today	Geolookup Autocomplete Current conditions 3-day forecast summary Astronomy Almanac for today	Geolookup Autocomplete Current conditions 3-day forecast summary Astronomy Almanac for today
	10-day forecast summary Hourly 1-day forecast Satellite thumbnail Dynamic Radar image Severe alerts Tides and Currents Tides and Currents Raw Severe alerts	10-day forecast summary Hourly 1-day forecast Satellite thumbnail Dynamic Radar image Severe alerts Tides and Currents Tides and Currents Raw Severe alerts
		Hourly 10-day forecast Yesterday's weather summary Travel Planner Webcams thumbnails Dynamic animated Radar image Dynamic animated Satellite image Current Tropical Storms

Endpoints

Feature: forecast

Returns a summary of the weather for the next 3 days. This includes high and low temperatures, a string text forecast and the conditions.

Response Fields

txt_forecast: forecastday

period
icon
icon_url
title
fctext
fcttext_metric

Forecast for San Francisco, California

http://api.wunderground.com/api/*10b150b0989de03c*/**forecast**
/q/CA/San_Francisco.json

Hide Response


```
{
  "response": {
  "version": "0.1",
  "termsofService": "http://www.wunderground.com/weather/api/d/terms.html",
  "features": {
  "forecast": 1
  }
  },
  "forecast": {
  "txt_forecast": {
  "date": "2:00 PM PDT",
  "forecastday": [{
  "period": 0,
  "icon": "partlycloudy",
  "icon_url": "http://icons-ak.wxug.com/i/c/k/partlycloudy.gif",
  "title": "Tuesday",
  "fcttext": "Partly cloudy in the morning, then clear. High of 68F. Breezy. Winds from the West
  "fcttext_metric": "Partly cloudy in the morning, then clear. High of 20C. Windy. Winds from th
  "pop": "0"
  }, {
```

Accessing APIs Programmatically

Desirable to make requests:

- that depend on other information
- at regular or adaptive time intervals
- on a large scale

Reading a URL

In [6]:
```
import urllib2

url = 'http://api.wunderground.com/api/10b150b0989de03c/forecast/q/CA/San_Francisco.json'
page = urllib2.urlopen(url).read()
```

In [7]:
```
print page
```


```
{
  "response": {
  "version":"0.1",
  "termsofService":"http://www.wunderground.com/weather/api/d/terms.html",
  "features": {
  "forecast": 1
  }
        }
                ,
        "forecast":{
                "txt_forecast": {
                "date":"8:56 AM PST",
                "forecastday": [
                {
                "period":0,
                "icon":"partlycloudy",
                "icon_url":"http://icons.wxug.com/i/c/k/partlycloudy.gif",
                "title":"Thursday",
                "fcttext":"Cloudy skies this morning will become partly cloudy this afternoo
```

Parsing JSON

```
In [8]: import json

In [13]: p = json.loads(page)

In [16]: p['forecast']['simpleforecast']['forecastday'][0]
```

```
Out[16]: {u'avehumidity': 62,
 u'avewind': {u'degrees': 320, u'dir': u'NW', u'kph': 10, u'mph': 6},
 u'conditions': u'Partly Cloudy',
 u'date': {u'ampm': u'PM',
  u'day': 11,
  u'epoch': u'1455246000',
  u'hour': 19,
  u'isdst': u'0',
  u'min': u'00',
  u'month': 2,
  u'monthname': u'February',
  u'monthname_short': u'Feb',
  u'pretty': u'7:00 PM PST on February 11, 2016',
  u'sec': 0,
  u'tz_long': u'America/Los_Angeles',
  u'tz_short': u'PST'
```

```
In [17]: p['forecast']['simpleforecast']['forecastday'][0]['high']
Out[17]: {u'celsius': u'18', u'fahrenheit': u'65'}
```

CTA API: store key in file and construct URL

In [34]:
```
with open('cta_api_key') as f:
    key=f.readline()
```

In [38]:
```
url = 'http://www.ctabustracker.com/bustime/api/v1/getpredictions?key=' + key[:-1] + '&rt=20&s
```

Read URL into BeautifulSoup object

In [40]:
```
from BeautifulSoup import BeautifulSoup
page = urllib2.urlopen(url)
soup = BeautifulSoup(page)
```

In [45]:
```
soup
```

Out[45]:
```
<?xml version='1.0' encoding='utf-8'?>
<bustime-response>
<prd>
<tmstmp>20160211 12:32</tmstmp>
<typ>A</typ>
<stpnm>Madison &amp;amp; Jefferson</stpnm>
<stpid>456</stpid>
<vid>1024</vid>
<dstp>6369</dstp>
<rt>20</rt>
<rtdir>Westbound</rtdir>
<des>Austin</des>
<prdtm>20160211 12:44</prdtm>
<tablockid>20 -813</tablockid>
<tatripid>1040695</tatripid>
<zone></zone>
</prd>
<prd>
<tmstmp>20160211 12:32</tmstmp>
<typ>A</typ>
<stpnm>Madison &amp;amp; Jefferson</stpnm>
```

Find an element

```
In [44]: soup.find('rt')
Out[44]: <rt>20</rt>
```

Get elements contents

```
In [52]: soup.find('rt').contents[0]
Out[52]: u'20'
```

```
In [55]: soup.find('prdtm')
Out[55]: <prdtm>20160211 12:44</prdtm>

In [57]: soup.find('prdtm').contents[0]
Out[57]: u'20160211 12:44'
```

Get multiple elements

```
In [60]: soup.findAll('prdtm')
Out[60]: [<prdtm>20160211 12:44</prdtm>, <prdtm>20160211 12:55</prdtm>]
```

Parse timestamps

In [68]:
```
from pandas import Timestamp
now = Timestamp(soup.find('tmstmp').contents[0])
arrival = Timestamp(soup.find('prdtm').contents[0])
```

In [69]:
```
arrival-now
```

Out[69]:
```
Timedelta('0 days 00:12:00')
```