weather predictions suck

(using `python/R` to find out how badly)

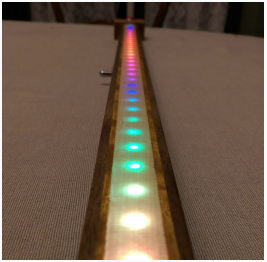
John Henderson

25 August 2016

`tcrug`

a bit about me

- mechanical engineer
- 3M, product development
- learned `R` ~5 years ago
- arduino ~3 years ago
- `python` ~6 mos ago
- interests: woodworking, data viz, home repair (?), piano, soccer, vintage tools, making



tl;dr

- I wanted to compare various weather provider accuracies
- [finally] ”learned” `python` via a Coursera class
- collected a mess of data via a Raspberry Pi and weather APIs
- took a look at initial results to validate approach using R
- looking for input/interested partners for a second round!

who cares about weather *that* much?

- t-shirt or long sleeves?
- should I bring an umbrella?
- maybe we *won't* be having that picnic

my motivation

unfortunately, sometimes the weather is a bigger deal!

when you have this...

my motivation

unfortunately, sometimes the weather is a bigger deal!

when you have this...

...rain means this

my observations

- rain predictions would swing wildly, even with sources claiming hyper accuracy like arcus (based on forecast.io)
- I probably had 4 different apps on my phone, and they'd disagree
- the *real* weather could swing either way... I put up the tarp for nothing, and scrambled to put it up when forecasts indicated I was safe

my observations

- rain predictions would swing wildly, even with sources claiming hyper accuracy like arcus (based on forecast.io)
- I probably had 4 different apps on my phone, and they'd disagree
- the *real* weather could swing either way... I put up the tarp for nothing, and scrambled to put it up when forecasts indicated I was safe
- I *hated* putting that tarp up

my observations

- rain predictions would swing wildly, even with sources claiming hyper accuracy like arcus (based on forecast.io)
- I probably had 4 different apps on my phone, and they'd disagree
- the *real* weather could swing either way... I put up the tarp for nothing, and scrambled to put it up when forecasts indicated I was safe
- I *hated* putting that tarp up
- like everything else in life, anger is an incredible motivator :)

I know kung-fu

- I've fizzled out with lpthw and dip at least a few times
- came across the python specialization on Coursera, which stuck
- nice intro, and definitely helped with the basics!
  - regex/parsing
  - `GET` / `POST` commands
  - databases
  - working with `json` and `.csv`
  - iterating through lists

what do I *really* want?

- for a given time, I want predictions from 0 - n hours/days ago
- as the actual day/time approaches, I want increasing granularity
- data: will it rain/snow and how much, temperature, , general conditions (sunny, cloudy, windy), humidity, pressure, wind speed/direction

what do I *really* want?

- for a given time, I want predictions from 0 - n hours/days ago
- as the actual day/time approaches, I want increasing granularity
- data: will it rain/snow and how much, temperature, , general conditions (sunny, cloudy, windy), humidity, pressure, wind speed/direction

Basically, for a given time, what did the forcasters *think* the weather would be 2 hrs ago, 4 days ago, a week ago?

data collection

- searched around, honed to three weather APIs based on apparent ease of use/being free: weather underground, aeris, and forecast.io
- `python` script for each source to call the API and output to `.csv`
- used a `cron` job on an RPi to collect hourly snapshots for ~3 weeks
- each snapshot contained hourly predictions for the next ~48hrs
- thus, after the first 48hrs, I had hourly predictions from 0-2 days ago for a given date/time

imports and defining the API URL

```
import urllib
import json
import time
import csv
#import antigravity

base_url = "https://api.forecast.io/forecast/"
api_key = "key"
options = "?exlude=flags,alerts,minutely"

# st. paul coordinates, per google
lat = "44.9442"
lon = "-93.0936"

url = base_url + api_key + "/" + lat + "," + lon + options
```

setting the outputs

```
now = time.strftime("%Y-%m-%d_%H:%M")

out_json = root + "forecast_" + now + ".json"
con_json = open(out_json, "w")
out_csv = open(root + "forecast_" + now + ".csv", "w")
con_csv = csv.writer(out_csv)

data = urllib.urlopen(url).read()
data_js = json.loads(data)
```

what fields to collect?

```
fields = [ "precipAccumulation",
           "precipType",
           "cloudCover",

           ...

           "precipProbability",
           "temperatureMin",
           "temperatureMax"
]
```

parsing the `json`

```
for item in data_js["hourly"]["data"] :
    snap_time = time.strftime("%Y-%m-%d %H:%M",
                              time.localtime(int(item["time"])))
    snapshot = {}
    for key, value in item.items() :
        if key in fields :
            snapshot[key] = str(value)

    snapshot["type"] = "hourly"
    snapshot["src"] = "forecast.io"
    the_data[snap_time] = snapshot
```

what I got!

- about 1800 files from 2016-03-10 to 2016-03-28
- ~300 empty files so something went awry at times
- overall, pretty happy with the test run!

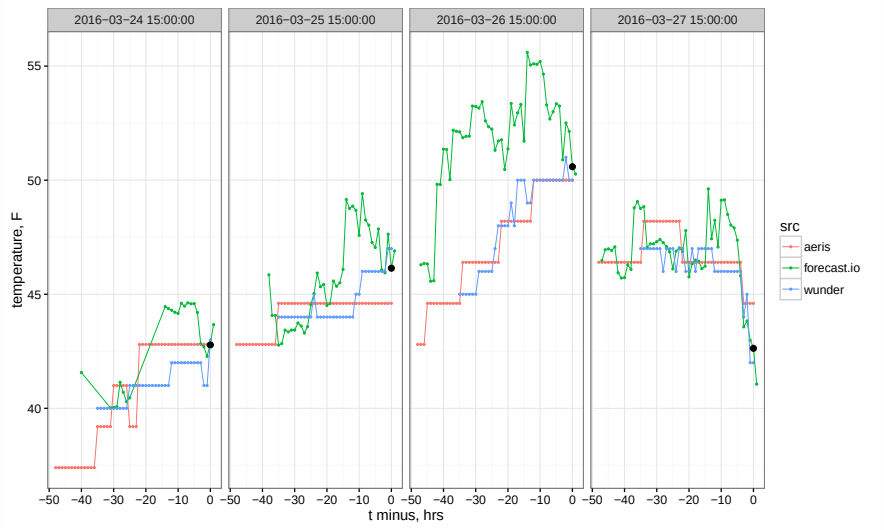
the data

```
> head(data)

    src   type                 key temp humidity summary wind_spd wind_dir pop
1 aeris hourly 2016-03-10 11:00:00 44.6        7      65        0        0   0
2 aeris hourly 2016-03-10 23:00:00 39.2        4      79        0        0   0
3 aeris hourly 2016-03-11 06:00:00 33.8        1      85        0        0   0
4 aeris hourly 2016-03-09 22:00:00 37.4        3      88        0        0   2
5 aeris hourly 2016-03-11 01:00:00 35.6        2      82        0        0   0
6 aeris hourly 2016-03-10 22:00:00 41.0        5      73        0        0   0

  precip pressure  snap_date snap_time                snap
1      7       41 2016-03-09     17:00 2016-03-09 17:00:00
2      6        3 2016-03-09     17:00 2016-03-09 17:00:00
3     10        7 2016-03-09     17:00 2016-03-09 17:00:00
4      4       96 2016-03-09     17:00 2016-03-09 17:00:00
5      6        3 2016-03-09     17:00 2016-03-09 17:00:00
6      6        3 2016-03-09     17:00 2016-03-09 17:00:00

                      grp
1 aeris2016-03-10 11:00:00
2 aeris2016-03-10 23:00:00
3 aeris2016-03-11 06:00:00
4 aeris2016-03-09 22:00:00
5 aeris2016-03-11 01:00:00
6 aeris2016-03-10 22:00:00
```

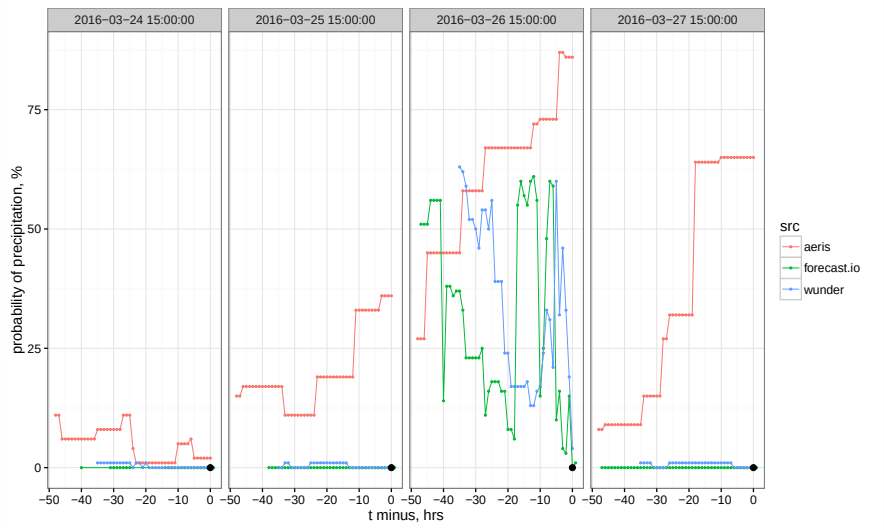
post-processing

- with the data grabbed, I switched to `R`
- basic process:
  - massage: string → numeric, decimal → %, etc.
  - set a target date range (*waaay* too much data for one plot)
  - group the data based on the `key` (target date/time of prediction)
  - create a "t-minus" column (snapshot time - pred target)
  - overplot the actual conditions on top of snapshots
- for the examples, I used four days' predictions for 3p

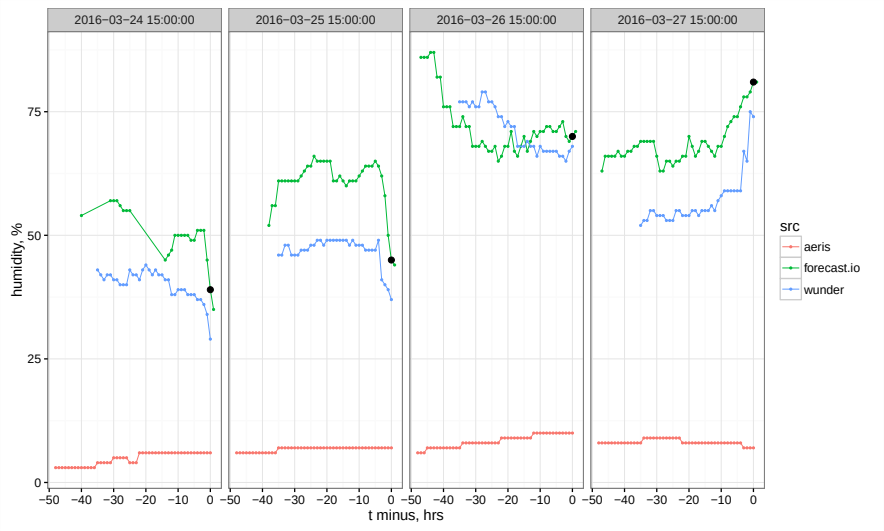
temperature



pop


2016−03−24 15:00:00
2016−03−25 15:00:00
2016−03−26 15:00:00
2016−03−27 15:00:00
probability of precipitation, %
75
50
25
0
−50 −40 −30 −20 −10 0
t minus, hrs
src
aeris
forecast.io
wunder

humidity


2016−03−24 15:00:00
2016−03−25 15:00:00
2016−03−26 15:00:00
2016−03−27 15:00:00
humidity, %
75
50
25
0
−50 −40 −30 −20 −10 0
t minus, hrs
src
aeris
forecast.io
wunder

issues/questions

- overlooked units (mainly ^{o}C/F), which required some fixing
- missing snapshots
- is `.csv` the best way to roll?
- how the heck to visualize so much data (`d3`, `shiny`?)
- what's the best source of the actual weather?
- is `R` as good/better than `python` for the data grab?

issues/questions

- better way to run through data returned from API call?
- picked the handy ”hourly” predictions provided by APIs; ideally I’d like to test the limits of each provider (some will predict out 10-15 days!)
- scrape/analyze embedded radar images?
- cleaner `python/R` in general...

the pitch!

Anyone want to partner up?

my vision

- re-write/tweak code
- optimize data storage/processing
- multiple climates / ~1 week
- figure out a better viz engine
- write up a comparison

perks

- internet fame, obviously
- noob teaches `python` to a noob
- I learn some better `R`
- educate the world on the horrible predictions they're trusting their garage roofs to

thanks for having me!

thanks for having me!

questions?