toaster Map examples

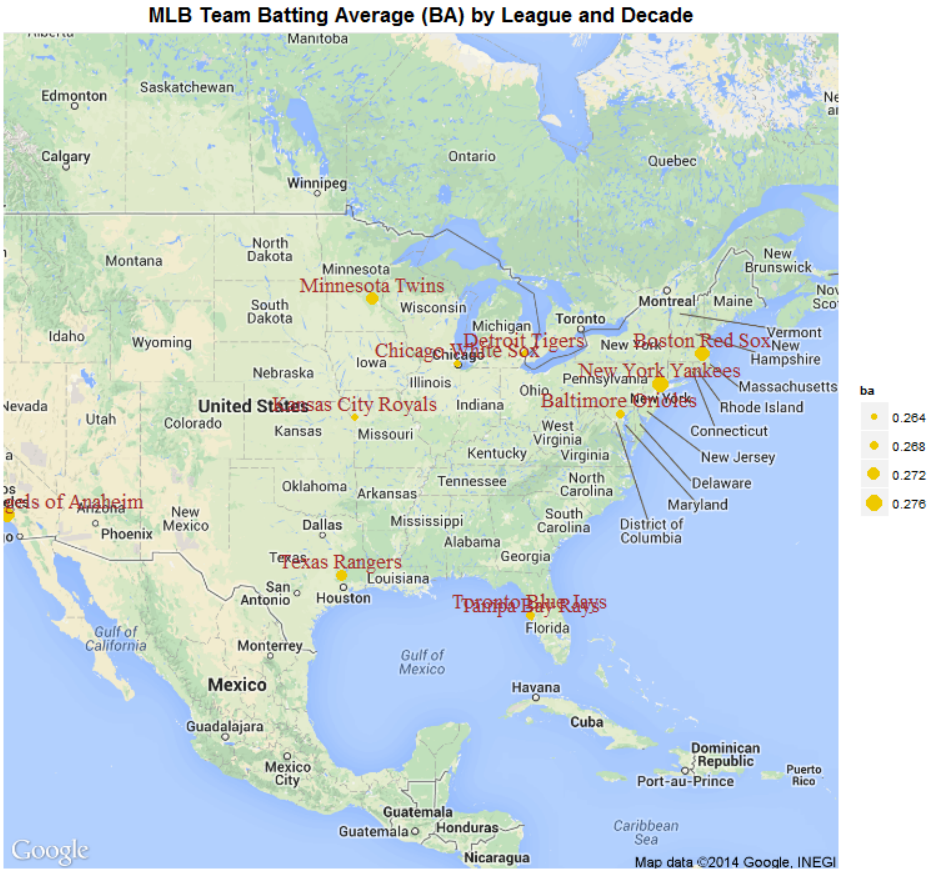
Simple Examples

Map of American League team batting averages in 2000s

Simple Map Example

```
# compute and retrieve team BAs
teamsAL2000 = compute(conn, "teams_enh",
                      by=c("teamid", "franchname name"),
                      aggregates=c("AVG(ba) ba", "AVG(slg) slg", "SUM(w) w", "SUM(l)
l"),
                      where="lgid = 'AL' and decadeid = 2000", stringsAsFactors=FALSE)

# display map with teams and their BAs
createMap(teamsAL2000, zoom=4, maptype="roadmap", source='google',
          locator='center', locationName = 'name',
          metricName='ba', labelName='name',
          textColour='brown', textFamily='serif', textFace='bold', textSize=6,
          title = "MLB Team Batting Average (BA) by League and Decade")
```



Several problems with this example are worth mentioning:

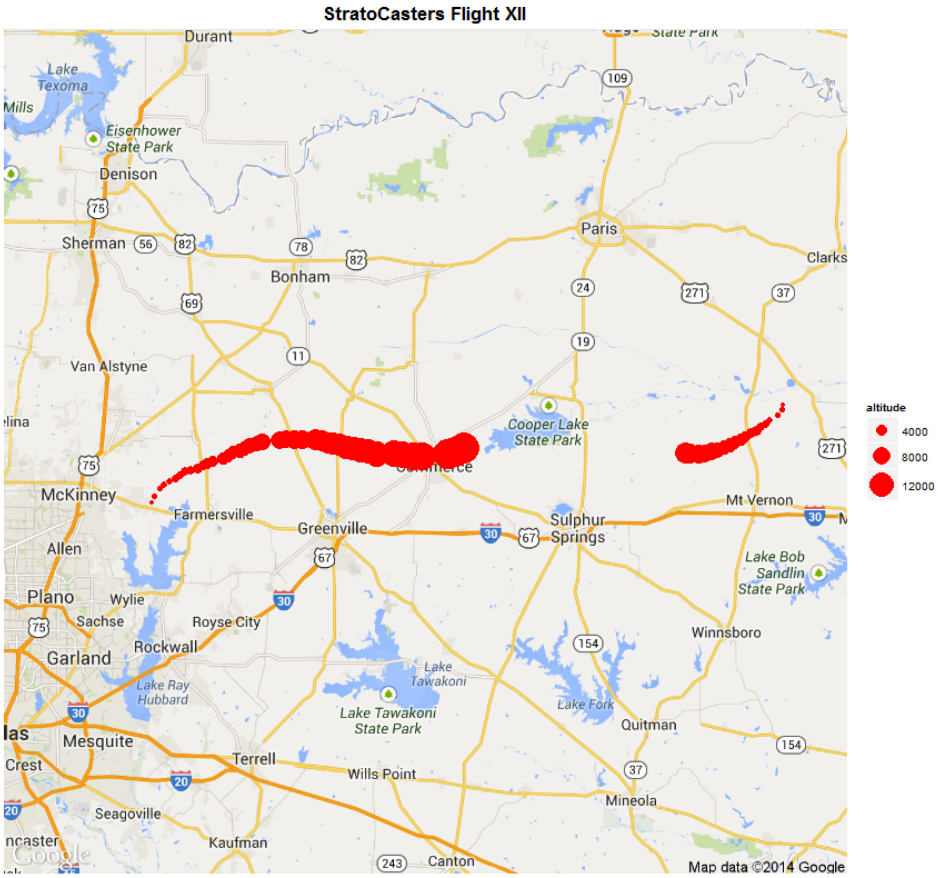
- although there are 15 AL teams map displays only 11 of them
- some team names are not fully visible
- _Toronto Blue Jays_ is placed in Tampa, Florida (probably their spring camp location)
- map is not ideally positioned missing part of western continental US

Map of StratoCasters Flight XII path with altitude

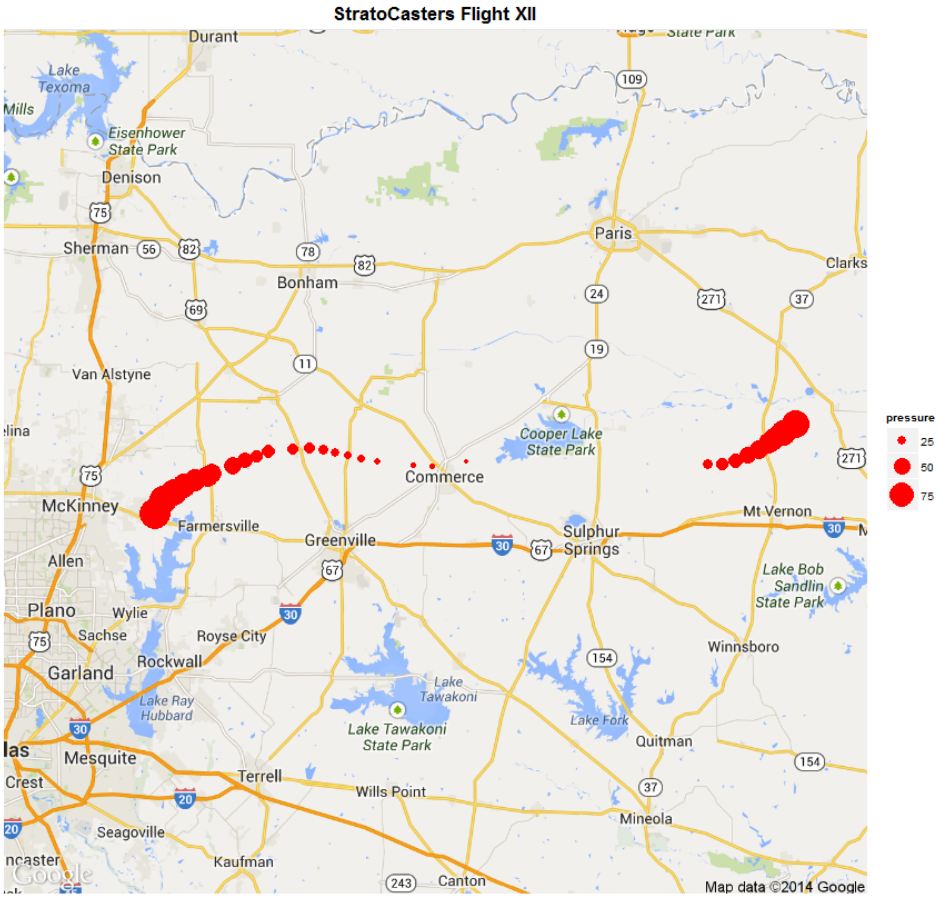
StratoCasters Flight XII Path with Altitude

```
stratPositions = compute(conn, "flights", by=c('flightid', 'measurementtime'),
                         aggregates=c('AVG(altitude) altitude', 'AVG(pressure)
pressure', 'AVG(lat) lat', 'AVG(lon) lon'),
                         where="altitude IS NOT NULL")

createMap(stratPositions, zoom=9,
          maptype="roadmap", source='google',
          locator='center', lonName='lon', latName='lat',
          shapeColour = 'red',
          metricName='altitude', scaleRange=c(2,13),
          geocodeFun=geocodeMem, getmapFun=getmapMem,
          title = "StratoCasters Flight XII")
```



Note, that in `createMap` just changing parameter from `metricName='altitude'` to `metricName='pressure'` creates new map with air pressure measurements instead:



Despite noted shortcomings, these examples demonstrated main features of map functions in **toaster**:

- calls function *compute* and retrieves data from Aster database;
- geocodes each row (data point) using values stored in the column specified in parameter `locationName`;
- calculates map location from geocoded values and uses latitude and longitude averages to center it (parameter `locator`);
- downloads and formats map image based on calculated location and parameter *zoom* from source specified (parameter `source`) and parameter `maptype`;
- display map with data artifacts (parameters `metricName` and `labelName`);

Next, problems above will be addressed step-by-step.

> **Map APIs**
> Map functions rely on several map APIs (e.g. Google API) that require internet access. If you run R on computer without internet access then **toaster** map functions won't work.
>
> These APIs use public key access that restrict user access via delays and limiting number of calls. For example, Geocoding API has a number of request limitations in place to prevent abuse: an unspecied short-term rate limit is in place as well as a 24-hour limit of 2,500 requests.To minimize access to map APIs we suggest using memoisation mechanism, in particular toaster supports it with both internally and with special parameters such as geo

Geocoding

Geocoding is optional but important part of map visualization. Map coordinates commonly used are longitude and latitude. To place or locate a position on any map we will need to know these coordinates. If data already contain longitude and latitude assigned then geocoding is not necessary. But in many cases data may contain geographic information that needs converting to map coordinates.

> **Geocoding** is the process of finding associated geographic coordinates (often expressed as latitude and longitude) from other geographic data, such as street addresses, or ZIP codes (postal codes).

toaster map functions support both geocoded data and general geographical data. In latter case **toaster** attempts geocoding with using Google

first.

The Geocoding API has a number of request limitations (see info box above). To alleviate these restrictions **toaster** uses caching version of *geoc ode* function by default. This cache will live only inside of each call to map function without persisting across consecutive calls. More effective approach suggests creating caching version of *geocode* function and pass it to map functions on all subsequent calls. The same consideration applies to APIs used to retrieve map images. Example illustrates this pattern using **memoise** package:

Memoise geocode function before using map functions

```
require(toaster)
require(memoise)

geocodeMem = memoise(geocode)
getmapMem = memoise(get_map)
map1 = createMap(...., geocodeFun = geocodeMem, getmapFun = getmapMem)
map2 = createMap(...., geocodeFun = geocodeMem, getmapFun = getmapMem)
```

Code below is non-trivial example of geocoding existing MLB teams using web scraping of official MLB page with addresses of AL and NL teams:

Geocoding MLB teams using web scraping and Google API

```
require(toaster)
require(memoise)
require(XML)

url = "http://mlb.mlb.com/team/index.jsp"
pagetree = htmlTreeParse(url, useInternalNodes = T)
teams_names = xpathApply(pagetree, "//ul[@id = 'team_by_team']//ul[@class = 'al team'
or @class = 'nl team']/li/h5/a/text()", xmlValue)
teams_addr1 = xpathApply(pagetree, "//ul[@id = 'team_by_team']//ul[@class = 'al team'
or @class = 'nl team']/li[3]/text()", xmlValue)
teams_addr2 = xpathApply(pagetree, "//ul[@id = 'team_by_team']//ul[@class = 'al team'
or @class = 'nl team']/li[4]/text()", xmlValue)
webteams = data.frame(name=unlist(teams_names), address=paste(unlist(teams_addr1),
",", unlist(teams_addr2)),
                      stringsAsFactors=FALSE)

# geocode
geocodeMem = memoise(geocode)
geoteams = adply(webteams, 1, function(x) geocodeMem(x$address, output="latlon"))
```

The page from MLB.com site may change or even move. In either case code above will likely to break as it depends both on page URL (http://mlb.mlb.com/team/index.jsp) and on certain page format.

If you run the code you get a data frame with team's longitude and latitude originating from addresses specified on the web page. In later examples we will use map functions in both ways - by providing geocodes and by giving addresses and letting **toaster** geocode teams automatically.

Map Locating

Maps are static images downloaded from one of online providers: Google, OpenStreetMap, Stamen, or Cloudemade.

Map functions locate maps in the same way
All toaster map functions will use parameters *location* and *mapLocator* described below in the same way.

toaster obtains map by its location which is specified with either 2- or 4- valued vector in *location* parameter:

1. Parameter `location` is numeric vector of length 2: a longitude/latitude pair specifying the center of the map and accompanied by a *zoom* parameter.
2. parameter `location` is numeric vector of length 4: a bounding box specification left/bottom/right/top, specifying corresponding bounds of the map.

toaster will obtain and format map as described above when *location* parameter is present. When it is not **toaster** will use elaborate methods to derive map location from data. Based on another parameter `locator` it will one of the following:

1. Parameter `locator = 'center'` (default): it will scan all data longitudes and latitudes to compute their average values and use them for map center location.
2. Parameter `Locator = 'box'`: it will scan all data longitudes and latitudes to compute their max and min values and use them for map bounding box (plus 10% margin around).

Of course, these operations are performed only after all data is geocoded first (as necessary).

Google Maps and bounding box
When bounding box option is used with Google API then following warning may appear:

```
Warning: bounding box given to google - spatial extent only approximate.
converting bounding box to center/zoom specification. (experimental)
```

Troubleshooting

Function `createMap` fails with error 'Error in unit(x, default.units) : 'x' and 'units' must have length > 0'

This is due invalid value of zoom parameter when it is too high (too detailed level of the map) and data artifacts didn't fit onto map image. Adjust value of zoom parameter to lower value (e.g. zoom=4 fits entire continental US).

Examples

References

- Latitude and Longitude