Working with geo-spatial data in R

John Henderson

08 February 2014

Intro

- Rapid fire overview of geo-spatial miscellany in R
- If you're not familiar with R, don't worry about the code
- The code/data necessary to reproduce anything in this talk is all on github!
  - https://github.com/jwhendy/devFest-geo

Intro

- Rapid fire overview of geo-spatial miscellany in R
- If you're not familiar with R, don't worry about the code
- The code/data necessary to reproduce anything in this talk is all on github!
  - https://github.com/jwhendy/devFest-geo

My goal

Show you enough relatively cool things in 15min to entice you to give R a shot!

Using ggmap to get lat/lon coordinates

- Just use what you would type into Google Maps

```
# install.packages("ggmap")
library(ggmap)
geocode("St. Paul, MN")
```

```
       lon     lat
1 -93.08996 44.9537
```

```
geocode("2115 Summit Ave., St. Paul, MN")
```

```
       lon      lat
1 -93.18971 44.94412
```

```
geocode("University of St. Thomas, MN")
```

```
       lon      lat
1 -93.18975 44.94192
```

Grabbing maps

Methods

- Lat/lon + zoom level
- Bounding box

Sources

- Google
- Stamen
- Cloudmade
- OpenStreetMap

Using lat/lon + zoom

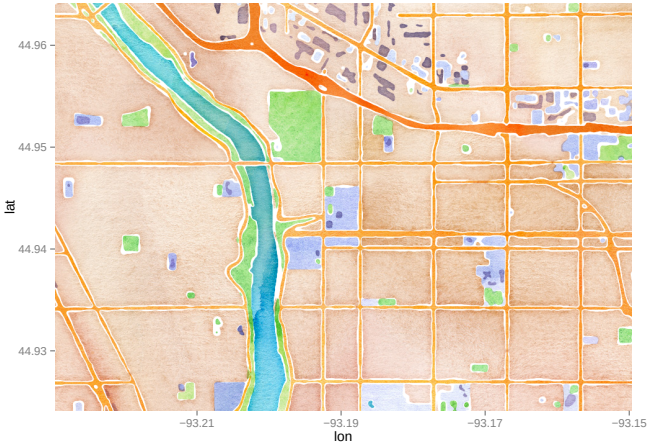
```
loc <- geocode("2115 Summit Ave, St. Paul, MN")
ust <- get_map(location = c(lon = loc$lon, lat = loc$lat),
               zoom = 15, source = "google",
               maptype = "hybrid", crop = T)

ggmap(ust)
```

Using lat/lon + zoom

Using lat/lon + bounding box

```
loc <- geocode("2115 Summit Ave, St. Paul, MN")

box <- c(left = loc$lon - 0.04, bottom = loc$lat - 0.02,
         right = loc$lon + 0.04, top = loc$lat + 0.02)

ust_box <- get_map(location = box, source = "stamen",
                   maptype = "watercolor", crop = T)

ggmap(ust_box)
```

Using lat/lon + bounding box

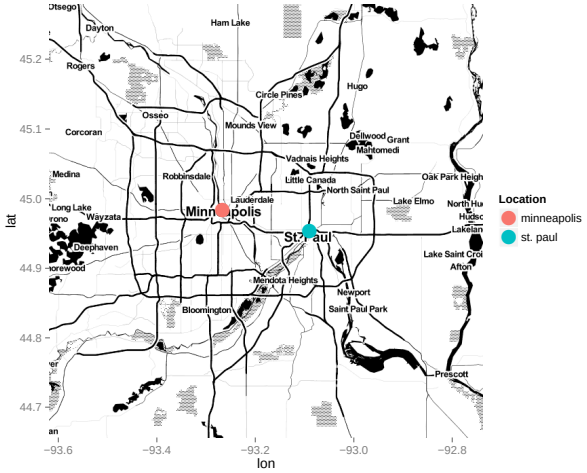


Overplotting maps

- `ggmap()` adds onto existing `ggplot2` functionality
- Thus, layering standard `ggplot2` graphics on top of maps is easy!

```
locs <- data.frame(names = c("st. paul", "minneapolis"))                  # grab locations
locs <- cbind(locs, geocode(as.character(locs$names)))

mid <- get_map(location = c(lon = mean(locs$lon), lat = mean(locs$lat)),  # get a map
               zoom = 10, source = "stamen", maptype = "toner", crop = T)

p <- ggmap(mid)                                                           # plot map
p <- p + geom_point(aes(x = lon, y = lat, colour = factor(names)),        # overplot w/ points
                    dat = locs, size = 6)
p <- p + scale_colour_discrete("Location")
```

Overplotting ggmaps



Working with world/country maps

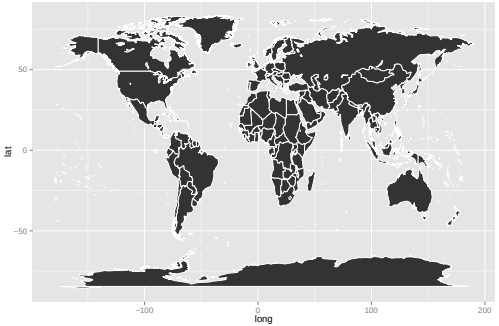
- `get_map()` limited by allowable zoom levels (e.g. `0 < zoom < 21`)
- For larger scale data, the `maps` package is necessary
- Various maps available; see the documentation

```
library(maps)
world <- map_data("world")
head(map, 5)
```

```
       long      lat group order region subregion
1 -133.3664 58.42416     1     1 Canada      <NA>
2 -132.2681 57.16308     1     2 Canada      <NA>
3 -132.0498 56.98610     1     3 Canada      <NA>
4 -131.8797 56.74001     1     4 Canada      <NA>
5 -130.2492 56.09945     1     5 Canada      <NA>
```

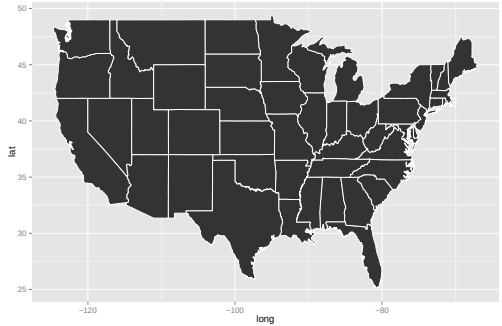
World map example

```
# pay attention to column names (long vs. lon!)
p <- ggplot(world, aes(x = long, y = lat, group = group))
p <- p + geom_polygon(colour = "white")
p
```



United States example

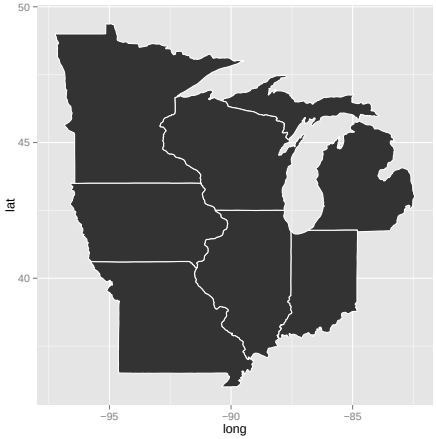
```
usa <- map_data("state")
p <- ggplot(usa, aes(x = long, y = lat, group = group))
p <- p + geom_polygon(colour = "white")
p
```



Subsetting areas

```
states <- c("minnesota", "wisconsin", "illinois", "indiana",
            "iowa", "missouri", "michigan")

states_map <- map_data("state")

states_map <- states_map[states_map$region %in% states, ]

p <- ggplot(states_map, aes(x = long, y = lat, group = group))
p <- p + geom_polygon(colour = "white")
p
```

Subsetting areas



A hobby project

- Internal talks at 3M typically given live to US audience; recorded for int'l
- Organized two "reverse talks" to reach a wider global audience
- Attendance records shattered
- Wanted to visualize impact/reach!

The inspiration

Calculating great circle arcs

- `gcIntermediate()` function from `geosphere` package

```
library(geosphere)
arc <- gcIntermediate(c(lon_1, lat_1), c(lon_2, lat_2),
                      n = steps, addStartEnd = T)
```

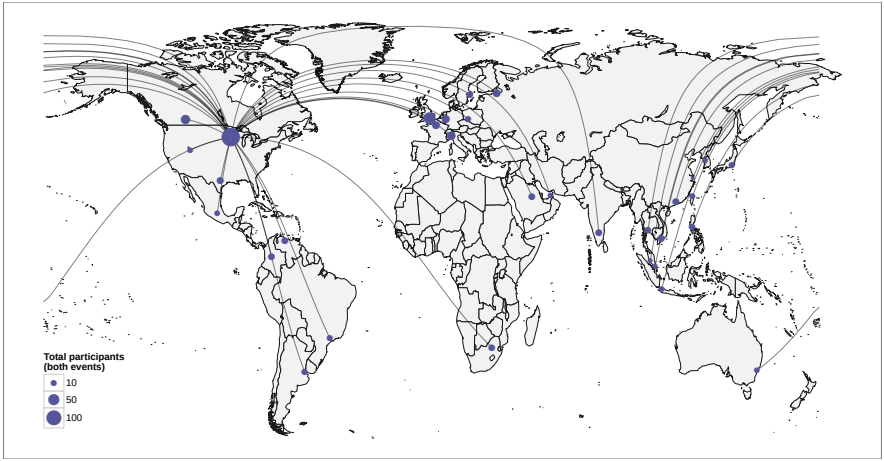
Crossing the Date Line

```
# draw great circles from St. Paul to everywhere else
gcircles <- lapply(1:nrow(end), function(i) {                  # arc for each set
  temp <- gcIntermediate(start[, c("lon", "lat")],
                         end[i, c("lon", "lat")],
                         n = 50, addStartEnd = T,
                         breakAtDateLine = T)

  if(is.list(temp) == T) {                                     # were two half-paths returned?
    ids <- c(rep(paste0("i", i), nrow(temp[[1]])),             # if so, make id's for both halves
             rep(paste0("j", i), nrow(temp[[2]])))
    temp <- as.data.frame(rbind(temp[[1]], temp[[2]]))         # combine two sets of points
    temp$id <- ids                                             # assign ids for plotting
  }
  ...
```

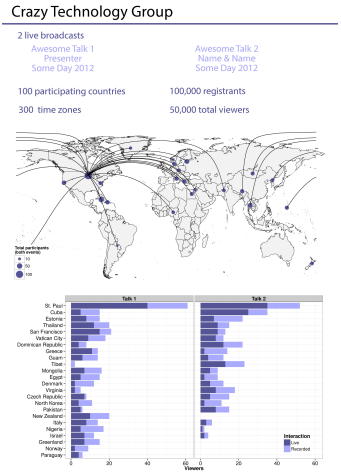
The plot

```
p <- ggplot()
p <- p + geom_polygon(aes(x = long, y = lat, group = group),              # world map
                      data = world, colour = "gray10", fill = "gray95")
p <- p + geom_line(aes(x = lon, y = lat, group = id),                     # great circles
                   dat = gcircles, lwd = 0.4, alpha = 0.5)
p <- p + geom_point(aes(x = lon, y = lat, size = sqrt(total/pi)),         # points
                    dat = talks_agg, colour = "#555599")
p <- p + scale_size("Total participants\n(both events)",                  # adjust legend
                    limits = c(0, max(sqrt(talks_agg$total / pi)) + 1),
                    breaks = sqrt(c(10, 50, 100) / pi),
                    labels = c(10, 50, 100), range = c(1, 10))
p <- p + theme_bw()                                                       # bw theme
p <- p + theme(axis.text = element_blank(), axis.title = element_blank(), # tweak
               axis.ticks = element_blank(), panel.grid = element_blank(),
               legend.position = c(0.092, 0.15))
```

The result



Infographic



Creating `.kml` files for Google Earth

- Pretty easy! (Follow along with this, this, this, and this)

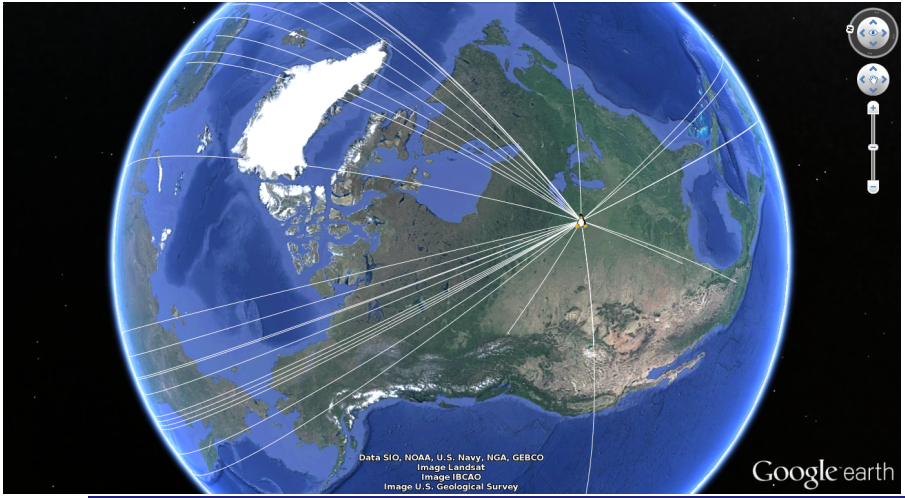
```
talks_agg$size <- 100 * talks_agg$total


talks_sp <- talks_agg
coordinates(talks_sp) <- c("lon", "lat")                    # converts to spatial object
proj4string(talks_sp) <- CRS("+init=epsg:4238")             # G Earth coordinate system
talks_ll <- spTransform(talks_sp,
                        CRS("+proj=longlat +datum=WGS84"))  # Something else G Earth wants


# open a file connection and write the data with custom icon/size/label
kml_open("./data/talks-w-gcircles.kml")
kml_layer.SpatialPoints(talks_ll, colour = "white", labels = city, size = total,
                        shape="http://upload.wikimedia.org/wikipedia/commons/a/af/Tux.png")
kml_close("./data/talks-w-gcircles.kml")
```

Adding great circles

- A bit hackish, but manually writing the kml code works, too!

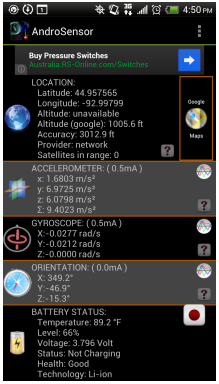
```
# for each point set, generate the following kml syntax
gcircs <- lapply(1:nrow(end), function(i) {
  paste0("<Placemark><LineString><tesselate>1</tesselate><coordinates>",
         start$lon, ",", start$lat, " ", end[i, "lon"], ",", end[i, "lat"],
         "</coordinates></LineString></Placemark>") })

# combine the results into a single table
gcircs <- do.call(rbind, gcircs)

# print out each line of the above table
# retult can simply be pasted after last <Placemark> in .kml file
write.table(gcircs, row.names = F)
```

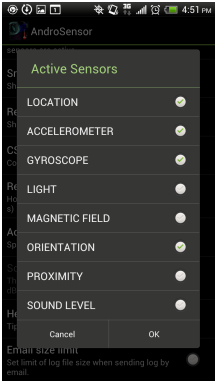
`.kml` file in Google Earth



Getting some GPS data

AndroSensor





Reading/cleaning the data

```
gps <- read.csv("./data/gps-data.csv", sep = ";")

# subset to columns we're interested in
gps <- gps[, c(10, 11, 13, 18)]

# give the columns sensical names
names(gps) <- c("lat", "lon", "speed", "time")
head(gps, 5)
```

```
       lat       lon speed time
1 44.92633 -93.09771    NA    5
2 44.92633 -93.09771    NA  505
3 44.92633 -93.09771    NA 1011
4 44.92633 -93.09771    NA 1532
5 44.92626 -93.09747     0 2032
```

Plot path and speed over a map

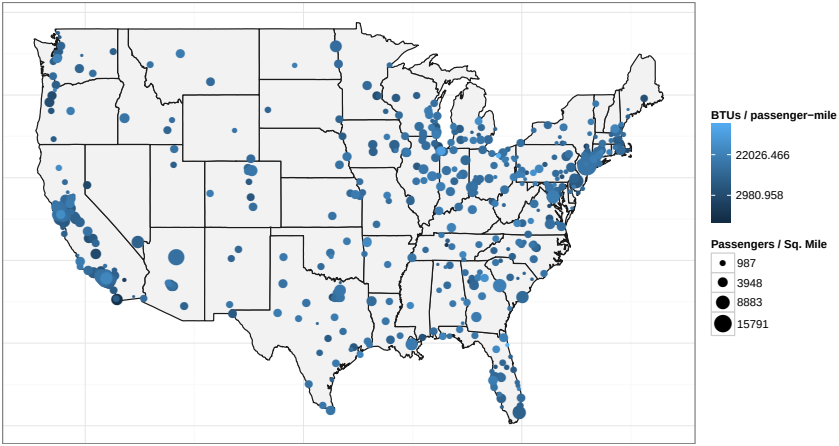
```
p <- ggmap(gps_map)
p <- p + geom_point(aes(x = lon, y = lat, colour = speed), data = gps, size = 3)
p <- p + scale_colour_continuous(low = "black", high = "red", na.value = NA)
```

Public transportation efficiency

- Background information can be found here

```
transpo <- read.csv("./data/public-transpo.csv")

transpo_agg <- ddply(transpo, .(city, state), summarize,                    # stats by state
                     btus_pmile_ave = mean(btus_pmile),
                     density = mean(population) / mean(service_area_sq_mi))
transpo_agg$lookup <- paste0(transpo_agg$city, ", ", transpo_agg$state)

coords <- geocode(transpo_agg$lookup)                                       # city/state string

transpo_agg <- cbind(transpo_agg, coords)                                   # get lat/lon

# let's not run that again...
write.table(transpo_agg, file = "./data/transpo-agg-geocoded.csv", row.names = F, sep = ",")
```

Efficiency visualized



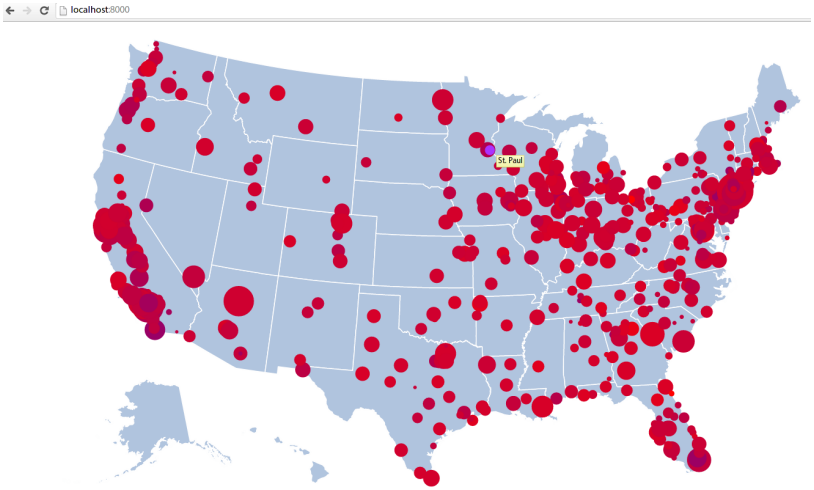
Converting data to `.json` for d3

- R is an excellent environment for working with data
  - Reshape (wide $\leftrightarrow$ long)
  - Calculate columns
  - Summarize, e.g. with `ddply`
- Manipulate in R, output to `.json` with this or `jsonlite` package

```
library(jsonlite)
plot_json <- toJSON(plot)               # convert with jsonlite
file_con <- file("./d3/transpo.json")  # create file
writeLines(plot_json, file_con)         # write results to file
close(file_con)
```

```
[ { "city" : "Abilene", "state" : "TX", "btus_pmile_ave" : 13574.1, "density" : 2547.39,
    "lookup" : "Abilene, TX", "lon" : -99.73, "lat" : 32.45 }, ...
```

And voilà, a d3 version!

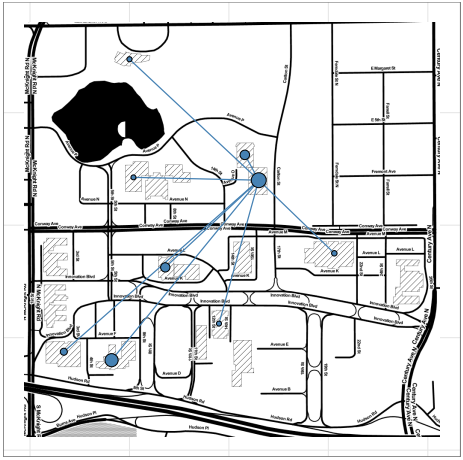


Visualizing work activity

- Tracked work projects with Timesheet (projects, tags, location)

```
time <- read.csv("./data/timesheet.csv")              # read data
time <- ddply(time, .(Location), summarize,           # total time per loc
              total = sum(Duration))

# figure out building coords and merge with timesheet data
bldgs <- data.frame(loc = time$Location))
bldgs$lat <-  c(44.9515, 44.9576, 44.9525, 44.9545, 44.9577, 44.955,  44.9512, 44.9619, 44.9585)
bldgs$lon <- -c(93.0033, 92.9935, 92.9955, 92.9982, 92.9998, 92.9897, 93.0009, 93.00,   92.9942)
time <- merge(time, bldgs, by.x = "Location", by.y = "loc")

# line from my building to each of the other ones
lines <- data.frame(lat = c(rep(bldgs[2, "lat"], 8), bldgs[c(1, 3:9), "lat"]),
                    lon = c(rep(bldgs[2, "lon"], 8), bldgs[c(1, 3:9), "lon"]), grp = rep(1:8, 2))
```

Visualizing work activity



R packages used

- Getting coordinates/working with Google: `ggmap`
- Great circle paths: `geosphere`
- Larger scale maps: `maps`
- Projection conversion/KML output: `rgdal`, `maptools`, `sp`, and `plotKML`
- Summary computations on data: `ddply()` function from `plyr`
- Reshaping data (wide $\leftrightarrow$ long): `reshape2`
- Plotting: `ggplot2`

References

- ggmap: Spatial Visualization with ggplot2, Kahle & Wickham
- How to draw good looking maps in R, uchicagoconsulting
- Great circles on a recentered worldmap, AnthroSpace
- How to map connections with great circles, FlowingData
- SO question on creating `.kml` files
- For any given package, find the documentation on CRAN

Code and files from this presentation are on github!

Tools

Presentation made entirely with open source software!



- Emacs and Org-mode for reproducible code environment
- LaTeX / Beamer for typesetting
- Torino Beamer theme so that it wasn't obvious I was using Beamer

Questions?

Come say "Hi" at the TC R Users Group to learn more :)