ggmap quickstart

For more functionality, see ggmap documentation and https://dl.dropboxusercontent.com/u/24648660/ggmap%20useR%202012.pdf

There are 2 basic steps to making a map using ggmap:

Part 1: Download map raster → Part 2: Plot raster and overlay data

Part 1: Downloading the map raster

Start by loading the package: **library(ggmap)**

1. Define location: 3 ways

- location/address
  myLocation <- "University of Washington"
- lat/long
  myLocation <- c(lon = -95.3632715, lat = 29.7632836)
- bounding box lowerleftlon, lowerleftlat, upperrightlon, upperrightlat (a little glitchy for google maps)
  myLocation <- c(-130, 30, -105, 50)

Convert location/address its lat/long coordinates:
geocode("University of Washington")

2. Define map source, type, and color

The get_map function provides a general approach for quickly obtaining maps from multiple sources. I like this option for exploring different styles of maps.

There are 4 map "sources" to obtain a map raster, and each of these sources has multiple "map types" (displayed on right).

- stamen: maptype = c("terrain", "toner", "watercolor")
- google: maptype = c("roadmap", "terrain", "satellite", "hybrid")
- osm: open street map
- cloudmade: 1000s of maps, but an api key must be obtained from http://cloudmade.com

```
myMap <- get_map(location=myLocation,
source="stamen", maptype="watercolor", crop=FALSE)
ggmap(myMap)
```

This will produce a map that looks something like this

NOTE: crop = FALSE because otherwise, with stamen plots, the map is slightly shifted when I overlay data.

3. Fine tune the scale of the map using zoom

The get_map function takes a guess at the zoom level, but you can alter it:

zoom = integer from 3-21

3 = continent, 10=city, 21=building

(openstreetmap limit of 18)

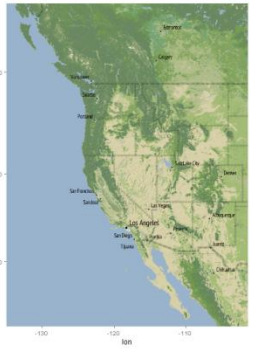
The following maps show different map source/type options (except cloudmade)

The appearance of these maps may be very different depending on zoom/scale

stamen: watercolor

stamen: toner

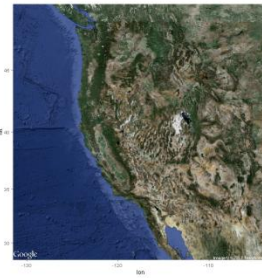
stamen: terrain



If you can't get the map you want by adjusting the location/zoom variables, the functions designed for the different map sources provide more options:
get_googlemap,
get_openstreetmap,
get_stamenmap,
get_cloudmademap

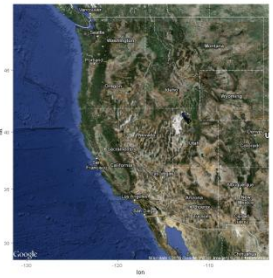
google: terrain

google: satellite

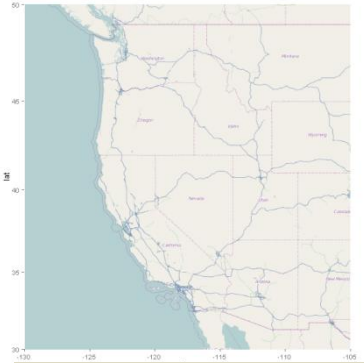


google: roadmap

google: hybrid



osm*



*Open street maps may return a '503 Service Unavailable' error. This means their server is unavailable, and you must wait for it to become available again.

All maps can be displayed in black and white
color = "bw"

myMap <- get_map(location=myLocation, source="osm", color="bw"))

- If you use Rstudio: Sometimes a plot will not display. Increasing the size of the plot window may help. dev.off() prior to plotting may also help.
- The urlonly = TRUE will return a url that will display your map in a web browser. Which is pretty cool and may be handy!
- legend="topleft" will inset the legend on the top left of the map is data is overlayed (page 2).

ggmap quickstart

Part 2: Plotting the maps and data

1 . Plot the raster:

```
ggmap(myMap)
```

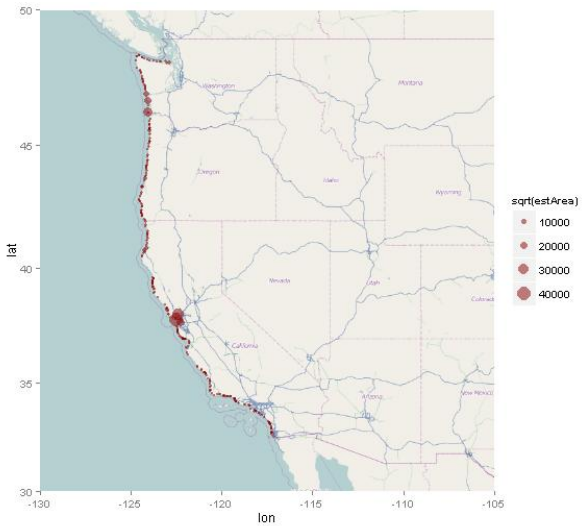
2 . Add points with latitude/longitude coordinates:

```
ggmap(myMap)+
geom_point(aes(x = Longitude, y = Latitude), data = data,
        alpha = .5, color="darkred", size = 3)
```

alpha = transparency
color = color
size = size of points

The size, color, alpha, and shape of the points can be scaled relative to another variable (in this case estArea) within the aes function:

```
ggmap(myMap)+
geom_point(aes(x = Longitude, y = Latitude, size=sqrt(estArea)),
data = data, alpha = .5, color="darkred")
```



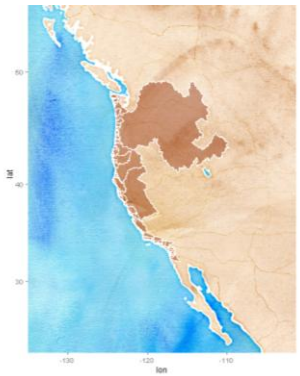
Additional functions can be added to control scaling, e.g.:

```
ggmap(myMap)+
 geom_point(aes(x = Longitude, y = Latitude, size=sqrt(estArea)),
 data = data, alpha = .5, color="darkred")+
 scale_size(range=c(3,20))
```

3. Add polygons from shp file

The shp file is imported into R using the rgdal package, and must be transformed to geographic coordinates (latitude/longitude) on the World Geodetic System of 1984 (WGS84) datum using the rgdal package:

```
library(rgdal)
shpData <- readOGR(dsn="C:\\Documents and Settings\\Watershed", layer="WS")
proj4string(shpData)     # describes data's current coordinate reference system
# to change to correct projection:
shpData <- spTransform(shpData,
CRS("+proj=longlat +datum=WGS84"))
```



To plot the data:

```
geom_polygon(aes(x = long, y = lat, group=id),
data = shpData, color ="white", fill ="orangered4",
alpha = .4, size = .2)
```

color= outline color
fill = polygon color
alpha = transparency
size = outline size

4. Annotate figure

```
baylor <- get_map('baylor university', zoom = 15, maptype = 'satellite')
ggmap(baylor) +
 annotate('rect', xmin=-97.11, ymin=31.54, xmax=-97.12, ymax=31.55, col="red", fill="white")+
 annotate('text', x=-97.12, y=31.54, label = 'Statistical Sciences', colour = I('red'), size = 8)+
 annotate('segment', x=-97.12, xend=-97.12, y=31.55, yend=31.55,
       colour=I('red'), arrow = arrow(length=unit(0.3,"cm")), size = 1.5) +
 labs(x = 'Longitude', y = 'Latitude') + ggtitle('Baylor University')
```

Controlling size and color

size

```
scale_size(range = c(3, 20))
```

But, the following is better for display because it is based on area (rather than radius)

```
scale_area(range=c(3,20))
```

color

Continuous variables: color gradient between n colors, e.g.:

```
scale_colour_gradientn(colours =
rainbow_hcl(7))
```

Discrete variables, e.g.:

```
scale_colour_manual(values=rainbow_hcl(7))
scale_colour_manual(values=c("8" = "red",
"4" = "blue","6" = "green")
```

*Use colorspace and RColorBrewer to choose color combinations