dkahle / **ggmap**

A package for plotting maps in R with ggplot2

394 commits | **4** branches | **0** packages | **1** release | **13** contributors

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

dkahle re-rox; allow make_bbox() to accept different f for longitude and lat... … ✓ Latest commit 8365275 on 4 Mar

.github	update issue template & pr template & rbuildignore	2 years ago
R	re-rox; allow make_bbox() to accept different f for longitude and lat...	2 months ago
data	switch class order in crime data	2 years ago
inst	more minor updates; clean check --as-cran (minus orphan listing)	16 months ago
man	re-rox; allow make_bbox() to accept different f for longitude and lat...	2 months ago
revdep	revdepcheck::revdep_check()	8 months ago
tests	add fake key to tests for travis	14 months ago
tools	readme re-compile	12 months ago
.Rbuildignore	use appveyor	14 months ago
.gitignore	initial commit of revdepcheck::revdep_check() and revdepcheck::revdep...	16 months ago
.travis.yml	travis: run tests on r-devel	13 months ago
DESCRIPTION	re-rox; allow make_bbox() to accept different f for longitude and lat...	2 months ago
NAMESPACE	first attempt removing bootstrap requirement (#264)	15 months ago
NEWS	final modifications before merging back onto master. passes all cran ...	16 months ago
NEWS.md	Properly encode #'s in Google URLs; Closes #272	14 months ago
README.Rmd	readme re-compile	12 months ago
README.md	readme re-compile	12 months ago
appveyor.yml	use appveyor	14 months ago
cran-comments.md	fix revdep check url	5 years ago
ggmap.Rproj	project overhead	5 years ago

README.md

CRAN 3.0.0 | build passing | build passing

Attention!

Google has recently changed its API requirements, and **ggmap** users are now required to register with Google. From a user's perspective, there are essentially three ramifications of this:

1. Users must register with Google. You can do this at https://cloud.google.com/maps-platform/. While it will require a valid credit card (sorry!), there seems to be a fair bit of free use before you incur charges, and even then the charges are modest for light use.

2. Users must enable the APIs they intend to use. What may appear to **ggmap** users as one overarching "Google Maps" product, Google in fact has several services that it provides as geo-related solutions. For example, the Maps Static API provides map images, while the Geocoding API provides geocoding and reverse geocoding services. Apart from the relevant Terms of Service, generally **ggmap** users don't need to think about the different services. For example, you just need to remember that `get_googlemap()` gets maps, `geocode()` geocodes (with Google, DSK is done), etc., and **ggmap** handles the queries for you. *However*, you do need to enable the APIs before you use them. You'll only need to do that once, and then they'll be ready for you to use. Enabling the APIs just means clicking a few radio buttons on the Google Maps Platform web interface listed above, so it's easy.

3. Inside R, after loading the new version of **ggmap**, you'll need provide **ggmap** with your API key, a hash value (think string of jibberish) that authenticates you to Google's servers. This can be done on a temporary basis with `register_google(key = "[your key]")` or permanently using `register_google(key = "[your key]", write = TRUE)` (note: this will overwrite your `~/.Renviron` file by replacing/adding the relevant line). If you use the former, know that you'll need to re-do it every time you reset R.

Your API key is *private* and unique to you, so be careful not to share it online, for example in a GitHub issue or saving it in a shared R script file. If you share it inadvertantly, just get on Google's website and regenerate your key - this will retire the old one. Keeping your key private is made a bit easier by **ggmap** scrubbing the key out of queries by default, so when URLs are shown in your console, they'll look something like `key=xxx`. (Read the details section of the `register_google()` documentation for a bit more info on this point.)

The new version of **ggmap** is now on CRAN soon, but you can install the latest version, including an important bug fix in `mapdist()`, here with:

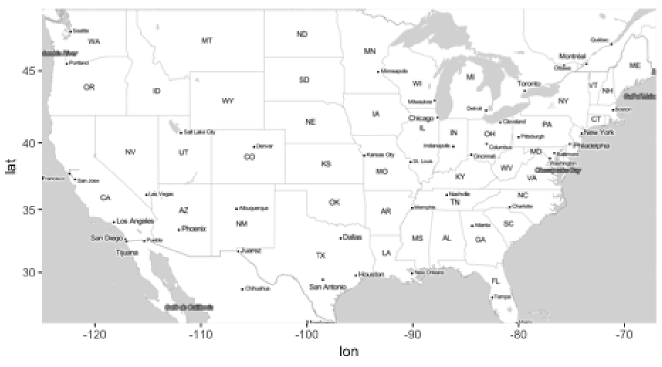
```
if(!requireNamespace("devtools")) install.packages("devtools")
devtools::install_github("dkahle/ggmap")
```

ggmap

ggmap is an R package that makes it easy to retrieve raster map tiles from popular online mapping services like Google Maps and Stamen Maps and plot them using the **ggplot2** framework:

```
library("ggmap")
#  Loading required package: ggplot2
#  Registered S3 methods overwritten by 'ggplot2':
#    method         from 
#    [.quosures     rlang
#    c.quosures     rlang
#    print.quosures rlang
#  Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
#  Please cite ggmap if you use it! See citation("ggmap") for details.

us <- c(left = -125, bottom = 25.75, right = -67, top = 49)
get_stamenmap(us, zoom = 5, maptype = "toner-lite") %>% ggmap()
#  Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.
```



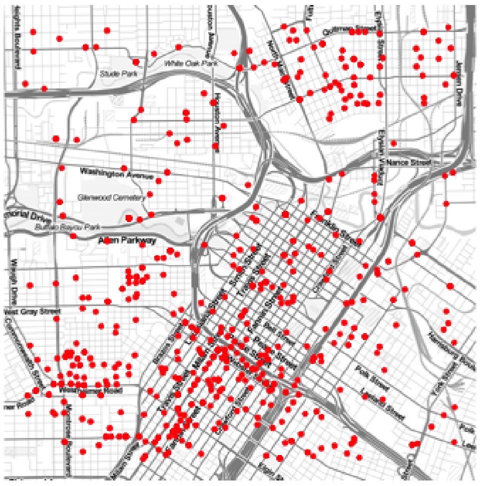
Use `qmplot()` in the same way you'd use `qplot()`, but with a map automatically added in the background:

```
library("dplyr")
#
#  Attaching package: 'dplyr'
#  The following objects are masked from 'package:stats':
#
#      filter, lag
#  The following objects are masked from 'package:base':
#
#      intersect, setdiff, setequal, union
library("forcats")

# define helper
`%notin%` <- function(lhs, rhs) !(lhs %in% rhs)

# reduce crime to violent crimes in downtown houston
violent_crimes <- crime %>%
  filter(
    offense %notin% c("auto theft", "theft", "burglary"),
    -95.39681 <= lon & lon <= -95.34188,
     29.73631 <= lat & lat <=  29.78400
  ) %>%
  mutate(
    offense = fct_drop(offense),
    offense = fct_relevel(offense, c("robbery", "aggravated assault", "rape", "murder"))
  )

# use qmplot to make a scatterplot on a map
qmplot(lon, lat, data = violent_crimes, maptype = "toner-lite", color = I("red"))
#  Using zoom = 14...
#  Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.
```



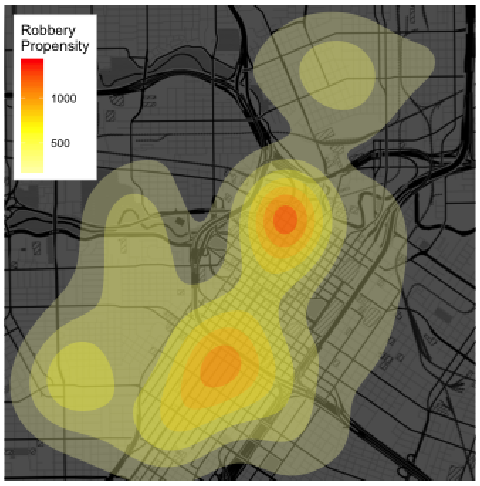
All the **ggplot2** geom's are available. For example, you can make a contour plot with `geom = "density2d"` :

```
qmplot(lon, lat, data = violent_crimes, maptype = "toner-lite", geom = "density2d", color = I("red"))
```

In fact, since **ggmap**'s built on top of **ggplot2**, all your usual **ggplot2** stuff (geoms, polishing, etc.) will work, and there are some unique graphing perks **ggmap** brings to the table, too.

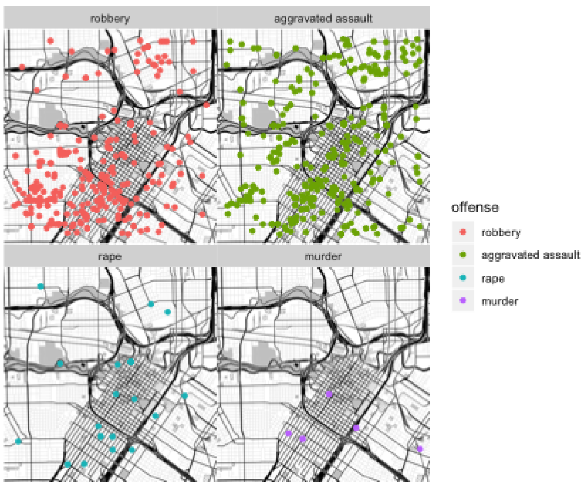
```
robberies <- violent_crimes %>% filter(offense == "robbery")

qmplot(lon, lat, data = violent_crimes, geom = "blank",
  zoom = 14, maptype = "toner-background", darken = .7, legend = "topleft"
) +
  stat_density_2d(aes(fill = ..level..), geom = "polygon", alpha = .3, color = NA) +
  scale_fill_gradient2("Robbery\nPropensity", low = "white", mid = "yellow", high = "red", midpoint = 650)
#  Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.
```



Faceting works, too:

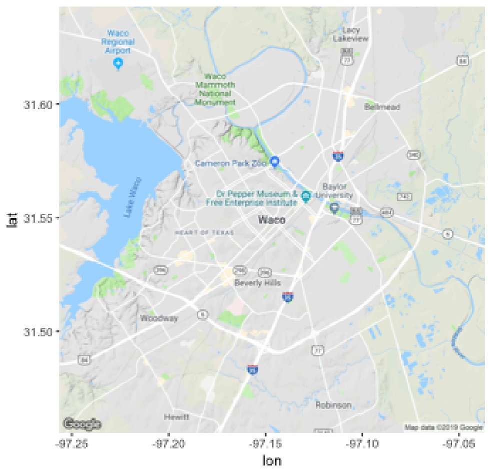
```
qmplot(lon, lat, data = violent_crimes, maptype = "toner-background", color = offense) +
  facet_wrap(~ offense)
#  Using zoom = 14...
#  Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.
```



Google Maps and Credentials

Google Maps can be used just as easily. However, since Google Maps use a center/zoom specification, their input is a bit different:

```
get_googlemap("waco texas", zoom = 12) %>% ggmap()
#  Source : https://maps.googleapis.com/maps/api/staticmap?center=waco%20texas&zoom=12&size=640x640&scale=2&maptype=terr
#  Source : https://maps.googleapis.com/maps/api/geocode/json?address=waco+texas&key=xxx
```



Moreover, you can get various different styles of Google Maps with **ggmap** (just like Stamen Maps):

```
get_googlemap("waco texas", zoom = 12, maptype = "satellite") %>% ggmap()
get_googlemap("waco texas", zoom = 12, maptype = "hybrid") %>% ggmap()
get_googlemap("waco texas", zoom = 12, maptype = "roadmap") %>% ggmap()
```

Google's geocoding and reverse geocoding API's are available through `geocode()` and `revgeocode()` , respectively:

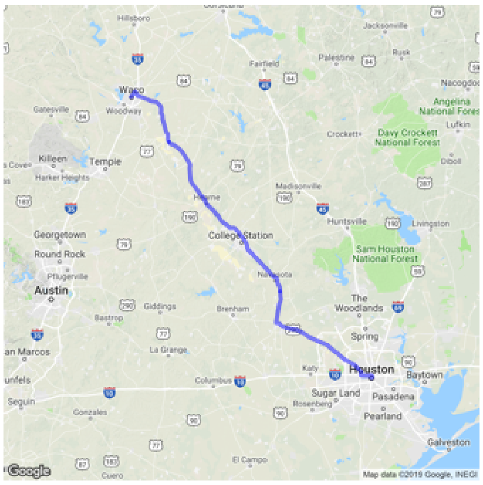
```
geocode("1301 S University Parks Dr, Waco, TX 76798")
#  Source : https://maps.googleapis.com/maps/api/geocode/json?address=1301+S+University+Parks+Dr,+Waco,+TX+76798&key=xxx
#  # A tibble: 1 x 2
#      lon   lat
#    <dbl> <dbl>
#  1 -97.1  31.6
revgeocode(c(lon = -97.1161, lat = 31.55098))
#  Source : https://maps.googleapis.com/maps/api/geocode/json?latlng=31.55098,-97.1161&key=xxx
#  Multiple addresses found, the first will be returned:
#    1301 S University Parks Dr, Waco, TX 76706, USA
#    55 Baylor Ave, Waco, TX 76706, USA
#    1437 FM434, Waco, TX 76706, USA
#    Bear Trail, Waco, TX 76706, USA
#    Robinson, TX 76706, USA
#    Waco, TX, USA
#    McLennan County, TX, USA
#    Texas, USA
#    United States
#  [1] "1301 S University Parks Dr, Waco, TX 76706, USA"
```

There is also a `mutate_geocode()` that works similarly to **dplyr**'s `mutate()` function:

```
tibble(address = c("white house", "", "waco texas")) %>%
  mutate_geocode(address)
#  Source : https://maps.googleapis.com/maps/api/geocode/json?address=white+house&key=xxx
#  "white house" not uniquely geocoded, using "1600 pennsylvania ave nw, washington, dc 20500, usa"
#  Source : https://maps.googleapis.com/maps/api/geocode/json?address=waco+texas&key=xxx
#  # A tibble: 3 x 3
#    address       lon   lat
#    <chr>       <dbl> <dbl>
#  1 white house -77.0  38.9
#  2 ""           NA    NA
#  3 waco texas  -97.1  31.5
```

Treks use Google's routing API to give you routes (`route()` and `trek()` give slightly different results; the latter hugs roads):

```
trek_df <- trek("houson, texas", "waco, texas", structure = "route")
#  Source : https://maps.googleapis.com/maps/api/directions/json?origin=houson,+texas&destination=waco,+texas&key=xxx&mo
qmap("college station, texas", zoom = 8) +
  geom_path(
    aes(x = lon, y = lat),  colour = "blue",
    size = 1.5, alpha = .5,
    data = trek_df, lineend = "round"
  )
#  Source : https://maps.googleapis.com/maps/api/staticmap?center=college%20station,%20texas&zoom=8&size=640x640&scale=2
#  Source : https://maps.googleapis.com/maps/api/geocode/json?address=college+station,+texas&key=xxx
```



(They also provide information on how long it takes to get from point A to point B.)

Map distances, in both length and anticipated time, can be computed with `mapdist()`). Moreover the function is vectorized:

```
mapdist(c("houston, texas", "dallas"), "waco, texas")
#  Source : https://maps.googleapis.com/maps/api/distancematrix/json?origins=dallas&destinations=waco,+texas&key=xxx&mod
#  Source : https://maps.googleapis.com/maps/api/distancematrix/json?origins=houston,+texas&destinations=waco,+texas&key
#  # A tibble: 2 x 9
#    from          to                m    km miles seconds minutes hours mode
#    <chr>         <chr>         <int> <dbl> <dbl>   <int>   <dbl> <dbl> <chr>
#  1 houston, tex… waco, texas 298227  298. 185.    10257   171.   2.85 drivi…
#  2 dallas        waco, texas 152480  152.  94.8    5356    89.3  1.49 drivi…
```

Installation

- From CRAN: `install.packages("ggmap")`
- From Github:

```
if (!requireNamespace("devtools")) install.packages("devtools")
devtools::install_github("dkahle/ggmap")
```