Package 'ggmap'

July 2, 2014

Type Package

Title A package for spatial visualization with Google Maps and OpenStreetMap

Version 2.3

Date 2013-03-09

Author David Kahle <david.kahle@gmail.com>, Hadley Wickham <h.wickham@gmail.com>

Maintainer David Kahle <david.kahle@gmail.com>

Description ggmap allows for the easy visualization of spatial data and models on top of Google Maps, OpenStreetMaps, Stamen Maps,or CloudMade Maps using ggplot2.

Depends R (>= 2.14.0), ggplot2 (>= 0.9.2)

Imports proto, scales, RgoogleMaps, png, plyr, reshape2, grid, rjson,mapproj

Suggests MASS, stringr

License MIT

LazyData true

Collate 'crime.R' 'geocode.R' 'ggimage.R' 'ggmap.R' 'hadley.R' 'qmap.R' 'theme_nothing.R' 'zips.R' 'theme_inset.R' 'OSM_scale_lookup.R' 'gglocator.R' 'mapdist.R' 'revgeocode.R' 'route.R' 'help.R' 'get_googlemap.R' 'get_openstreetmap.R' 'get_stamenmap.R' 'LonLat2XY.R' 'XY2LonLat.R' 'get_cloudmademap.R' 'get_map.R' 'qmplot.R' 'make_bbox.R' 'wind.R' 'inset_raster.R' 'inset.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2013-04-13 06:29:27

R topics documented:

`crime` *Crime data*

Description

Lightly cleaned Houston crime from January 2010 to August 2010 geocoded with Google Maps

Author(s)

Houston Police Department, City of Houston

References

http://www.houstontx.gov/police/cs/stats2.htm

---

distQueryCheck *Check Google Maps Distance Matrix API query limit*

Description

Check Google Maps Distance Matrix API query limit

Usage

```
distQueryCheck()
```

Value

a data frame

Author(s)

David Kahle <david.kahle@gmail.com>

See Also

http://code.google.com/apis/maps/documentation/distancematrix/

Examples

```
distQueryCheck()
```

---

geocode *Geocode*

Description

geocodes a location using Google Maps. Note that in most cases by using this function you are agreeing to the Google Maps API Terms of Service at https://developers.google.com/maps/terms.

Usage

```
geocode(location,
  output = c("latlon", "latlona", "more", "all"),
  messaging = FALSE, sensor = FALSE,
  override_limit = FALSE)
```

Arguments

location	a character string specifying a location of interest (e.g. "Baylor University")
output	amount of output
messaging	turn messaging on/off
sensor	whether or not the geocoding request comes from a device with a location sensor
override_limit	override the current query count (.GoogleGeocodeQueryCount)

Details

note that the google maps api limits to 2500 queries a day.

Value

depends (at least a data.frame with variables lon and lat)

Author(s)

David Kahle <david.kahle@gmail.com>

See Also

http://code.google.com/apis/maps/documentation/geocoding/

Examples

```
## Not run:
geocode('Baylor University')
geocode('1600 Pennsylvania Avenue, Washington DC')
geocode('1600 Pennsylvania Avenue, Washington DC', messaging = TRUE)
geocode('the white house', messaging = TRUE)
geocode('the eiffel tower')
geocode(c('baylor university', 'salvation army waco'))
geocode(c('baylor university', 'the vatican'))
geocode(c('baylor university', 'the vatican'), output = 'latlona')
geocode(c('baylor university', 'the vatican'), output = 'more')
geocode('the eiffel tower', output = 'all')
geocodeQueryCheck()

# careful in running this...
library(stringr)
ads <- unique(crime$address)[1:120]
ads <- paste(ads, ', houston, texas', sep = '')
ads <- str_trim(ads)
gc <- geocode(ads)

## End(Not run)
```

geocodeQueryCheck *Check Google Geocoding API query limit*

Description

Check Google Geocoding API query limit

Usage

```
geocodeQueryCheck()
```

Value

a data frame

Author(s)

David Kahle <david.kahle@gmail.com>

See Also

https://developers.google.com/maps/documentation/geocoding/

Examples

```
geocodeQueryCheck()
```

geom_leg *Single line segments with rounded ends*

Description

Single line segments with rounded ends

Usage

```
geom_leg(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", arrow = NULL, ...)
```

Arguments

mapping	mapping
data	data
stat	stat
position	position
arrow	arrow
...	...

Details

only intended for use in ggmaps package. only designed for mercator projection.

See Also

geom_segment in ggplot2, inspired by http://spatialanalysis.co.uk/2012/02/great-maps-ggplot2/, route

Examples

```
## Not run:

(legs_df <- route(
"marrs mclean science, baylor university",
  "220 south 3rd street, waco, tx 76701", # ninfa's
  alternatives = TRUE))

options('device')$device(width = 11.65, height = 4.17)
qmap('424 clay avenue, waco, tx', zoom = 16, maprange = TRUE, maptype = 'satellite',
  base_layer = ggplot(aes(x = startLon, y = startLat), data = legs_df)) +
  geom_segment(
    aes(x = startLon, y = startLat, xend = endLon, yend = endLat, colour = route),
    alpha = 3/4, size = 3, data = legs_df
  ) +
  scale_x_continuous(breaks = pretty(c(-97.1325,-97.119),4), lim = c(-97.1325,-97.119)) +
  facet_wrap(~ route) + theme_bw() +
  labs(x = 'Longitude', y = 'Latitude', colour = 'Routes')

qmap('424 clay avenue, waco, tx', zoom = 16, maprange = TRUE, maptype = 'satellite',
  base_layer = ggplot(aes(x = startLon, y = startLat), data = legs_df)) +
  geom_leg(
    aes(x = startLon, y = startLat, xend = endLon, yend = endLat, colour = route),
    alpha = 3/4, size = 2, data = legs_df
  ) +
  scale_x_continuous(breaks = pretty(c(-97.1325,-97.119),4), lim = c(-97.1325,-97.119)) +
  facet_wrap(~ route) + theme_bw() +
  labs(x = 'Longitude', y = 'Latitude', colour = 'Routes')



## End(Not run)
```

---

get_cloudmademap *Get a CloudMade map*

Description

get_cloudmademap accesses a tile server for Stamen Maps and downloads/stiches map tiles/formats a map image.

Usage

```
get_cloudmademap(bbox = c(left = -95.80204, bottom = 29.38048, right = -94.92313, top = 30.14344),
  zoom = 10, api_key, maptype = 1, highres = TRUE,
  crop = TRUE, messaging = FALSE, urlonly = FALSE,
  filename = "ggmapTemp", color = c("color", "bw"), ...)
```

Arguments

bbox	a bounding box in the format c(lowerleftlon, lowerleftlat, upperrightlon, upperrightlat).
zoom	a zoom level
api_key	character string containing cloud made api key, see details
maptype	an integer of what cloud made calls style, see details
highres	double resolution
crop	crop raw map tiles to specified bounding box
messaging	turn messaging on/off
urlonly	return url only
filename	destination file for download (file extension added according to format)
color	color or black-and-white
...	...

Details

accesses cloud made maps. this function requires an api which can be obtained for free from http://cloudmade.com/user/show. thousands of maptypes ("styles"), including create-your-own options, are available from http://maps.cloudmade.com/editor

Value

a map image as a 2d-array of colors as hexadecimal strings representing pixel fill values.

Author(s)

David Kahle <david.kahle@gmail.com>

See Also

http://maps.cloudmade.com/, ggmap

Examples

```
## Not run:

# in what follows, enter your own api key
api_key <- '<your api key here>'

map <- get_cloudmademap(api_key = api_key)
```

```
ggmap(map)

map <- get_cloudmademap(maptype = 997, api_key = api_key)
ggmap(map)

map <- get_cloudmademap(maptype = 31643, api_key = api_key)
ggmap(map)

map <- get_cloudmademap(maptype = 31408, api_key = api_key)
ggmap(map)

map <- get_cloudmademap(maptype = 15434, api_key = api_key)
ggmap(map)

map <- get_cloudmademap(maptype = 9203, api_key = api_key)
ggmap(map)

map <- get_cloudmademap(maptype = 53428, api_key = api_key)
ggmap(map)

map <- get_cloudmademap(maptype = 15153, api_key = api_key)
ggmap(map)

map <- get_cloudmademap(maptype = 7877, api_key = api_key)
ggmap(map)




## End(Not run)
```

`get_googlemap` *Get a Google Map*

Description

get_googlemap accesses the Google Static Maps API version 2 to download a static map. Note that in most cases by using this function you are agreeing to the Google Maps API Terms of Service at https://developers.google.com/maps/terms.

Usage

```
get_googlemap(center = c(lon = -95.3632715, lat = 29.7632836),
  zoom = 10, size = c(640, 640), scale = 2,
  format = c("png8", "gif", "jpg", "jpg-baseline", "png32"),
  maptype = c("terrain", "satellite", "roadmap", "hybrid"),
  language = "en-EN", region, markers, path, visible,
  style, sensor = FALSE, messaging = FALSE,
```

```
    urlonly = FALSE, filename = "ggmapTemp",
    color = c("color", "bw"), ...)
```

Arguments

center	the center of the map. this can either be 1. a longitude/latitude numeric vector or 2. a character string address (note that the latter uses a geocode)
zoom	map zoom, an integer from 3 (continent) to 21 (building), default value 10 (city)
size	rectangular dimensions of map in pixels - horizontal x vertical - with a max of c(640, 640). this parameter is affected in a multiplicative way by scale.
scale	multiplicative factor for the number of pixels returned possible values are 1, 2, or 4 (e.g. size = c(640,640) and scale = 2 returns an image with 1280x1280 pixels). 4 is reserved for google business users only. scale also affects the size of labels as well.
format	character string providing image format - png, jpeg, and gif formats available in various flavors
maptype	character string providing google map theme. options available are 'terrain', 'satellite', 'roadmap', and 'hybrid'
language	character string providing language of map labels (for themes with them) in the format 'en-EN'. not all languages are supported; for those which aren't the default language is used
region	borders to display as a region code specified as a two-character ccTLD ('top-level domain') value, see http://en.wikipedia.org/wiki/List_of_Internet_top-level_domains#Country_code_top-level_domains
markers	data.frame with first column longitude, second column latitude, for which google markers should be embedded in the map image, or character string to be passed directly to api
path	data.frame (or list of data.frames) with first column longitude, second column latitude, for which a single path should be embedded in the map image, or character string to be passed directly to api
visible	a location as a longitude/latitude numeric vector (or data frame with first column longitude, second latitude) or vector of character string addresses which should be visible in map extent
style	character string to be supplied directly to the api for the style argument . this is a powerful complex specification, see https://developers.google.com/maps/documentation/staticmaps/
sensor	specifies whether the application requesting the static map is using a sensor to determine the user's location
messaging	turn messaging on/off
urlonly	return url only
filename	destination file for download (file extension added according to format)
color	color or black-and-white
...	...

Value

a map image as a 2d-array of colors as hexadecimal strings representing pixel fill values.

Author(s)

David Kahle <david.kahle@gmail.com>

See Also

https://developers.google.com/maps/documentation/staticmaps/, ggmap

Examples

```
## Not run:

get_googlemap(urlonly = TRUE)

# get_googlemap has several argument checks
get_googlemap(zoom = 13.5)
get_googlemap(scale = 3)
get_googlemap(center = c(-30,-110))

# markers and paths are easy to access
d <- function(x=-95.36, y=29.76, n,r,a){
  round(data.frame(
    lon = jitter(rep(x,n), amount = a),
    lat = jitter(rep(y,n), amount = a)
  ), digits = r)
}
df <- d(n=50,r=3,a=.3)
map <- get_googlemap(markers = df, path = df,, scale = 2)
ggmap(map)
ggmap(map, fullpage = TRUE) +
  geom_point(aes(x = lon, y = lat), data = df, size = 3, colour = 'black') +
  geom_path(aes(x = lon, y = lat), data = df)

gc <- geocode('waco, texas')
center <- as.numeric(gc)
ggmap(get_googlemap(center = center, color = 'bw', scale = 2), fullpage = T)

# the scale argument can be seen in the following
# (make your graphics device as large as possible)
ggmap(get_googlemap(center, scale = 1), fullpage = TRUE) # pixelated
ggmap(get_googlemap(center, scale = 2), fullpage = TRUE) # fine



## End(Not run)
```

get_map *Grab a map.*

Description

get_map is a smart function which queries the Google Maps, OpenStreetMap, or Stamen Maps server for a map at a certain location at a certain spatial zoom. it is a wrapper for get_googlemap, get_openstreetmap, get_stamenmap, and get_cloudmademap functions. get_map was formerly (<2.0) called ggmap.

Usage

```
get_map(location = c(lon = -95.3632715, lat = 29.7632836),
  zoom = "auto", scale = "auto",
 maptype = c("terrain", "satellite", "roadmap", "hybrid", "toner", "watercolor"),
  messaging = FALSE, urlonly = FALSE,
  filename = "ggmapTemp", crop = TRUE,
  color = c("color", "bw"),
  source = c("google", "osm", "stamen", "cloudmade"),
  api_key)
```

Arguments

location	an address, longitude/latitude pair (in that order), or left/bottom/right/top bounding box
zoom	map zoom, an integer from 3 (continent) to 21 (building), default value 10 (city). openstreetmaps limits a zoom of 18, and the limit on stamen maps depends on the maptype. 'auto' automatically determines the zoom for bounding box specifications, and is defaulted to 10 with center/zoom specifications. maps of the whole world currently not supported.
scale	scale argument of get_googlemap or get_openstreetmap
maptype	character string providing map theme. options available are 'terrain', 'satellite', 'roadmap', and 'hybrid' (google maps), 'terrain', 'watercolor', and 'toner' (stamen maps), or a positive integer for cloudmade maps (see ?get_cloudmademap)
source	Google Maps ('google'), OpenStreetMap ('osm'), Stamen Maps ('stamen'), or CloudMade maps ('cloudmade')
messaging	turn messaging on/off
urlonly	return url only
filename	destination file for download (file extension added according to format)
crop	(stamen and cloudmade maps) crop tiles to bounding box
color	color ('color') or black-and-white ('bw')
api_key	an api key for cloudmade maps

Value

a data.frame with columns latitude, longitude, and fill

Author(s)

David Kahle <david.kahle@gmail.com>

See Also

ggmap, GetMap in package RgoogleMaps

Examples

```
## Not run:
map <- get_map()
ggmap(map, fullpage = TRUE)

map <- get_map(maptype = 'roadmap')
ggmap(map, fullpage = TRUE)

map <- get_map(maptype = 'hybrid')
ggmap(map, fullpage = TRUE)

map <- get_map(maptype = 'satellite')
ggmap(map, fullpage = TRUE)

map <- get_map(source = 'osm')
ggmap(map, fullpage = TRUE)

map <- get_map(source = 'stamen')
ggmap(map, fullpage = TRUE)

map <- get_map(source = 'stamen', maptype = 'watercolor')
ggmap(map, fullpage = TRUE)

map <- get_map(source = 'stamen', maptype = 'toner')
ggmap(map, fullpage = TRUE)



map <- get_map(location = 'texas', zoom = 6, source = 'stamen')
ggmap(map, fullpage = TRUE)

map <- get_map(location = 'united states', zoom = 4, source = 'stamen')
ggmap(map, fullpage = TRUE)

api_key <- '<your api key here>'
map <- get_map(location = 'baylor university', source = 'cloudmade',
  maptype = 53428, api_key = api_key, zoom = 14)
ggmap(map, fullpage = TRUE)
```

```
## End(Not run)
```

`get_openstreetmap` *Get an OpenStreetMap*

Description

get_openstreetmap accesses a tile server for OpenStreetMap and downloads/formats a map image.

Usage

```
get_openstreetmap(bbox = c(left = -95.80204, bottom = 29.38048, right = -94.92313, top = 30.14344),
  scale = 606250,
  format = c("png", "jpeg", "svg", "pdf", "ps"),
  messaging = FALSE, urlonly = FALSE,
  filename = "ggmapTemp", color = c("color", "bw"), ...)
```

Arguments

`bbox`	a bounding box in the format c(lowerleftlon, lowerleftlat, upperrightlon, upperrightlat)
`scale`	scale parameter, see http://wiki.openstreetmap.org/wiki/MinScaleDenominator. smaller scales provide a finer degree of detail, where larger scales produce more coarse detail
`format`	character string providing image format - png, jpeg, svg, pdf, and ps formats
`messaging`	turn messaging on/off
`urlonly`	return url only
`filename`	destination file for download (file extension added according to format)
`color`	color or black-and-white
`...`	...

Details

this is simply a wrapper for the gui web-based version at http://www.openstreetmap.org/. if you don't know how to get the map you want, go there, navigate to the map extent that you want, click the export tab at the top of the page, and copy the information into this function. the scale argument is a tricky number to properly specify. in most cases, if you get an error when downloading an openstreetmap the error is attributable to an improper scale specification. OSM_scale_lookup can help; but the best way to get in the correct range is to go to http://www.openstreetmap.org/, navigate to the map of interest, click export at the top of the page, click 'map image' and then copy down the scale listed. in some cases the OSM server is unavailable. in these cases you will receive an error message for download.file with the message HTTP status was '503 Service Unavailable'. you can confirm this by setting urlonly = TRUE, and then entering the URL in a web browser. the solution is either (1) change sources or (2) wait for the OSM servers to come back up.

Value

a map image as a 2d-array of colors as hexadecimal strings representing pixel fill values.

Author(s)

David Kahle <david.kahle@gmail.com>

See Also

http://www.openstreetmap.org/, ggmap

Examples

```
## Not run:
osm <- get_openstreetmap()
ggmap(osm)



## End(Not run)
```

---

get_stamenmap *Get a Stamen Map*

Description

get_stamenmap accesses a tile server for Stamen Maps and downloads/stiches map tiles/formats a map image.

Usage

```
get_stamenmap(bbox = c(left = -95.80204, bottom = 29.38048, right = -94.92313, top = 30.14344),
  zoom = 10,
  maptype = c("terrain", "watercolor", "toner"),
  crop = TRUE, messaging = FALSE, urlonly = FALSE,
  filename = "ggmapTemp", color = c("color", "bw"), ...)
```

Arguments

bbox	a bounding box in the format c(lowerleftlon, lowerleftlat, upperrightlon, upperrightlat).
zoom	a zoom level
maptype	terrain, watercolor, or toner
crop	crop raw map tiles to specified bounding box
messaging	turn messaging on/off
urlonly	return url only

filename	destination file for download (file extension added according to format)
color	color or black-and-white
...	...

Details

accesses stamen maps.

Value

a map image as a 2d-array of colors as hexadecimal strings representing pixel fill values.

Author(s)

David Kahle <david.kahle@gmail.com>

See Also

http://maps.stamen.com/#watercolor, ggmap

Examples

```
## Not run:

gc <- geocode('duncan hall, rice university')
google <- get_googlemap('rice university', zoom = 15)
ggmap(google) +
  geom_point(aes(x = lon, y = lat), data = gc, colour = 'red', size = 2)

bbox <- as.numeric(attr(google, 'bb'))[c(2,1,4,3)]
names(bbox) <- c('left','bottom','right','top')
stamen <- get_stamenmap(bbox, zoom = 15)
ggmap(stamen) +
  geom_point(aes(x = lon, y = lat), data = gc, colour = 'red', size = 2)

osm <- get_openstreetmap(bbox, scale = OSM_scale_lookup(15))
ggmap(osm) +
  geom_point(aes(x = lon, y = lat), data = gc, colour = 'red', size = 2)


ggmap(get_stamenmap(bbox, zoom = 15, maptype = 'watercolor'))+
  geom_point(aes(x = lon, y = lat), data = gc, colour = 'red', size = 2)

ggmap(get_stamenmap(bbox, zoom = 15, maptype = 'toner'))+
  geom_point(aes(x = lon, y = lat), data = gc, colour = 'red', size = 2)


## End(Not run)
```

ggimage *Plot an image using ggplot2*

Description

ggimage is the near ggplot2 equivalent of image.

Usage

```
ggimage(mat, fullpage = TRUE, coord_equal = TRUE,
  scale_axes = FALSE)
```

Arguments

mat	a matrix, imagematrix, array, or raster (something that can be coerced by as.raster)
fullpage	should the image take up the entire viewport?
coord_equal	should the axes units be equal?
scale_axes	should the axes be [0,ncol(mat)-1]x[0,nrow(mat)-1] (F) or [0,1]x[0,1] (T)

Value

a ggplot object

Author(s)

David Kahle <david.kahle@gmail.com>

Examples

```
## Not run:
img <- matrix(1:16, 4, 4)
image(img)
ggimage(t(img[,4:1]), fullpage = FALSE, scale_axes = TRUE)
ggimage(t(img[,4:1]), fullpage = FALSE)



data(hadley)
ggimage(hadley)
ggimage(hadley, coord_equal = FALSE)

x <- seq(1, 438, 15); n <- length(x)
df <- data.frame(x = x, y = -(120*(scale((x - 219)^3 - 25000*x) + rnorm(n)/2 - 3)))
qplot(x, y, data = df, geom = c('smooth','point'))
ggimage(hadley, fullpage = FALSE) +
  geom_smooth(aes(x = x, y = y), fill = I('gray60'), data = df,
    colour = I('green'), size = I(1)) +
  geom_point(aes(x = x, y = y), data = df,
```

```
    colour = I('green'), size = I(3), fill = NA)

## End(Not run)
```

---

gglocator *Locator for ggplots.*

Description

Locator for ggplots. (Note : only accurate when extent = "normal" when using ggmap.)

Usage

```
gglocator(n = 1, message = FALSE, xexpand = c(0.05, 0),
  yexpand = c(0.05, 0))
```

Arguments

n	number of points to locate.
message	turn messaging from grid.ls on/off
xexpand	expand argument in scale_x_continuous
yexpand	expand argument in scale_y_continuous

Value

a data frame with columns according to the x and y aesthetics

Author(s)

Tyler Rinker with help from Baptiste Auguie and StackOverflow user DWin with additions and canning by David Kahle <david.kahle@gmail.com>.

Examples

```
## Not run:
df <- expand.grid(x = 0:-5, y = 0:-5)
(p <- qplot(x, y, data = df) +
  annotate(geom = 'point', x = -2, y = -2, colour = 'red'))
gglocator()

p +
  scale_x_continuous(expand = c(0,0)) +
  scale_y_continuous(expand = c(0,0))
gglocator(1, xexpand = c(0,0), yexpand = c(0,0))


## End(Not run)
```

ggmap *Plot a ggmap object*

Description

ggmap plots the raster object produced by get_map.

Usage

```
ggmap(ggmap, extent = "panel", base_layer,
  maprange = FALSE, legend = "right", padding = 0.02,
  darken = c(0, "black"), ...)
```

Arguments

ggmap	an object of class ggmap (from function get_map)
extent	how much of the plot should the map take up? 'normal', 'panel', or 'device' (default)
base_layer	a ggplot(aes(...), ...) call; see examples
maprange	logical for use with base_layer; should the map define the x and y limits?
legend	'left', 'right' (default), 'bottom', 'top', 'bottomleft', 'bottomright', 'topleft', 'topright', 'none' (used with extent = 'device')
padding	distance from legend to corner of the plot (used with legend, formerly b)
darken	vector of the form c(number, color), where number is in [0, 1] and color is a character string indicating the color of the darken. 0 indicates no darkening, 1 indicates a black-out.
...	...

Value

a ggplot object

Author(s)

David Kahle <david.kahle@gmail.com>

See Also

get_map, qmap

Examples

```
## Not run:

## extents and legends
###################################################
hdf <- get_map()
ggmap(hdf, extent = 'normal')
ggmap(hdf) # extent = 'panel', note qmap defaults to extent = 'device'
ggmap(hdf, extent = 'device')

require(MASS)
mu <- c(-95.3632715, 29.7632836); nDataSets <- sample(4:10,1)
chkpts <- NULL
for(k in 1:nDataSets){
  a <- rnorm(2); b <- rnorm(2); si <- 1/3000 * (outer(a,a) + outer(b,b))
  chkpts <- rbind(chkpts, cbind(mvrnorm(rpois(1,50), jitter(mu, .01), si), k))
}
chkpts <- data.frame(chkpts)
names(chkpts) <- c('lon', 'lat','class')
chkpts$class <- factor(chkpts$class)
qplot(lon, lat, data = chkpts, colour = class)

ggmap(hdf, extent = 'normal') +
  geom_point(aes(x = lon, y = lat, colour = class), data = chkpts, alpha = .5)

ggmap(hdf) +
  geom_point(aes(x = lon, y = lat, colour = class), data = chkpts, alpha = .5)

ggmap(hdf, extent = 'device') +
  geom_point(aes(x = lon, y = lat, colour = class), data = chkpts, alpha = .5)

theme_set(theme_bw())
ggmap(hdf, extent = 'device') +
  geom_point(aes(x = lon, y = lat, colour = class), data = chkpts, alpha = .5)

ggmap(hdf, extent = 'device', legend = 'topleft') +
  geom_point(aes(x = lon, y = lat, colour = class), data = chkpts, alpha = .5)


## maprange
###################################################

hdf <- get_map()
mu <- c(-95.3632715, 29.7632836)
points <- data.frame(mvrnorm(1000, mu = mu, diag(c(.1, .1))))
names(points) <- c('lon', 'lat')
points$class <- sample(c('a','b'), 1000, replace = TRUE)

ggmap(hdf) + geom_point(data = points) # maprange built into extent = panel, device
ggmap(hdf) + geom_point(aes(colour = class), data = points)

ggmap(hdf, extent = 'normal') + geom_point(data = points)
```

```
# note that the following is not the same as extent = panel
ggmap(hdf, extent = 'normal', maprange = TRUE) + geom_point(data = points)

# and if you need your data to run off on a extent = device (legend included)
ggmap(hdf, extent = 'normal', maprange = TRUE) +
  geom_point(aes(colour = class), data = points) +
  theme_nothing() + theme(legend.position = 'right')




## cool examples
#################################################

# contour overlay
ggmap(get_map(maptype = 'satellite'), extent = 'device') +
  stat_density2d(aes(x = lon, y = lat, colour = class), data = chkpts, bins = 5)


# adding additional content
library(grid)
baylor <- get_map('baylor university', zoom = 15, maptype = 'satellite')
ggmap(baylor)

# use gglocator to find lon/lat's of interest
(clicks <- clicks <- gglocator(2) )
expand.grid(lon = clicks$lon, lat = clicks$lat)

ggmap(baylor) + theme_bw() +
  annotate('rect', xmin=-97.11920, ymin=31.5439, xmax=-97.101, ymax=31.5452,
    fill = I('black'), alpha = I(3/4)) +
  annotate('segment', x=-97.110, xend=-97.11920, y=31.5450, yend=31.5482,
    colour=I('red'), arrow = arrow(length=unit(0.3,"cm")), size = 1.5) +
  annotate('text', x=-97.110, y=31.5445, label = 'Department of Statistical Science',
    colour = I('red'), size = 6) +
  labs(x = 'Longitude', y = 'Latitude') + ggtitle('Baylor University')



baylor <- get_map('baylor university', zoom = 16, maptype = 'satellite')

ggmap(baylor, extent = 'device') +
  annotate('rect', xmin=-97.1164, ymin=31.5441, xmax=-97.1087, ymax=31.5449,
    fill = I('black'), alpha = I(3/4)) +
  annotate('segment', x=-97.1125, xend=-97.11920, y=31.5449, yend=31.5482,
    colour=I('red'), arrow = arrow(length=unit(0.4,"cm")), size = 1.5) +
  annotate('text', x=-97.1125, y=31.5445, label = 'Department of Statistical Science',
    colour = I('red'), size = 6)



# a shapefile like layer
data(zips)
```

```
ggmap(get_map(maptype = 'satellite', zoom = 8), extent = 'device') +
  geom_polygon(aes(x = lon, y = lat, group = plotOrder),
    data = zips, colour = NA, fill = 'red', alpha = .2) +
  geom_path(aes(x = lon, y = lat, group = plotOrder),
    data = zips, colour = 'white', alpha = .4, size = .4)

library(plyr)
zipsLabels <- ddply(zips, .(zip), function(df){
  df[1,c("area", "perimeter", "zip", "lonCent", "latCent")]
})
ggmap(get_map(maptype = 'satellite', zoom = 9),
    extent = 'device', legend = 'none', darken = .5) +
  geom_text(aes(x = lonCent, y = latCent, label = zip, size = area),
    data = zipsLabels, colour = I('red')) +
  scale_size(range = c(1.5,6))




## crime data example
#####################################################

# only violent crimes
violent_crimes <- subset(crime,
  offense != 'auto theft' &
  offense != 'theft' &
  offense != 'burglary'
)

# rank violent crimes
violent_crimes$offense <-
  factor(violent_crimes$offense,
    levels = c('robbery', 'aggravated assault',
      'rape', 'murder')
  )

# restrict to downtown
violent_crimes <- subset(violent_crimes,
  -95.39681 <= lon & lon <= -95.34188 &
   29.73631 <= lat & lat <=  29.78400
)


# get map and bounding box
theme_set(theme_bw(16))
HoustonMap <- qmap('houston', zoom = 14, color = 'bw',
  extent = 'device', legend = 'topleft')

# the bubble chart
library(grid)
HoustonMap +
   geom_point(aes(x = lon, y = lat, colour = offense, size = offense), data = violent_crimes) +
   scale_colour_discrete('Offense', labels = c('Robery','Aggravated Assault','Rape','Murder')) +
```

```
  scale_size_discrete('Offense', labels = c('Robery','Aggravated Assault','Rape','Murder'),
    range = c(1.75,6)) +
  guides(size = guide_legend(override.aes = list(size = 6))) +
  theme(
    legend.key.size = unit(1.8,'lines'),
    legend.title = element_text(size = 16, face = 'bold'),
    legend.text = element_text(size = 14)
  ) +
  labs(colour = 'Offense', size = 'Offense')


# a contour plot
HoustonMap +
  stat_density2d(aes(x = lon, y = lat, colour = offense),
    size = 3, bins = 2, alpha = 3/4, data = violent_crimes) +
  scale_colour_discrete('Offense', labels = c('Robery','Aggravated Assault','Rape','Murder')) +
  theme(
    legend.text = element_text(size = 15, vjust = .5),
    legend.title = element_text(size = 15,face='bold'),
    legend.key.size = unit(1.8,'lines')
  )



# a filled contour plot...
HoustonMap +
  stat_bin2d(aes(x = lon, y = lat, colour = offense, fill = offense),
    size = .5, bins = 30, alpha = 2/4, data = violent_crimes) +
  scale_colour_discrete('Offense',
    labels = c('Robery','Aggravated Assault','Rape','Murder'),
    guide = FALSE) +
  scale_fill_discrete('Offense', labels = c('Robery','Aggravated Assault','Rape','Murder')) +
  theme(
    legend.text = element_text(size = 15, vjust = .5),
    legend.title = element_text(size = 15,face='bold'),
    legend.key.size = unit(1.8,'lines')
  )

# ... with hexagonal bins
HoustonMap +
  stat_binhex(aes(x = lon, y = lat, colour = offense, fill = offense),
    size = .5, binwidth = c(.00225,.00225), alpha = 2/4, data = violent_crimes) +
  scale_colour_discrete('Offense',
    labels = c('Robery','Aggravated Assault','Rape','Murder'),
    guide = FALSE) +
  scale_fill_discrete('Offense', labels = c('Robery','Aggravated Assault','Rape','Murder')) +
  theme(
    legend.text = element_text(size = 15, vjust = .5),
    legend.title = element_text(size = 15,face='bold'),
    legend.key.size = unit(1.8,'lines')
  )
```

```
# changing gears (get a color map)
houston <- get_map('houston', zoom = 14)
HoustonMap <- ggmap(houston, extent = 'device', legend = 'topleft')

# a filled contour plot...
HoustonMap +
  stat_density2d(aes(x = lon, y = lat, fill = ..level.., alpha = ..level..),
    size = 2, bins = 4, data = violent_crimes, geom = 'polygon') +
  scale_fill_gradient('Violent\nCrime\nDensity') +
  scale_alpha(range = c(.4, .75), guide = FALSE) +
  guides(fill = guide_colorbar(barwidth = 1.5, barheight = 10))

# ... with an insert

overlay <- stat_density2d(aes(x = lon, y = lat, fill = ..level.., alpha = ..level..),
    bins = 4, geom = 'polygon', data = violent_crimes)


HoustonMap +
  stat_density2d(aes(x = lon, y = lat, fill = ..level.., alpha = ..level..),
    bins = 4, geom = 'polygon', data = violent_crimes) +
  scale_fill_gradient('Violent\nCrime\nDensity') +
  scale_alpha(range = c(.4, .75), guide = FALSE) +
  guides(fill = guide_colorbar(barwidth = 1.5, barheight = 10)) +
  inset(
    grob = ggplotGrob(ggplot() + overlay +
      scale_fill_gradient('Violent\nCrime\nDensity') +
      scale_alpha(range = c(.4, .75), guide = FALSE) +
      theme_inset()
    ),
    xmin = attr(houston,'bb')$ll.lon +
      (7/10) * (attr(houston,'bb')$ur.lon - attr(houston,'bb')$ll.lon),
    xmax = Inf,
    ymin = -Inf,
    ymax = attr(houston,'bb')$ll.lat +
      (3/10) * (attr(houston,'bb')$ur.lat - attr(houston,'bb')$ll.lat)
  )









## more examples
#################################################

# you can layer anything on top of the maps (even meaningless stuff)
df <- data.frame(
  lon = rep(seq(-95.39, -95.35, length.out = 8), each = 20),
```

```
  lat = sapply(
    rep(seq(29.74, 29.78, length.out = 8), each = 20),
    function(x) rnorm(1, x, .002)
  ),
  class = rep(letters[1:8], each = 20)
)

qplot(lon, lat, data = df, geom = 'boxplot', fill = class)

HoustonMap +
  geom_boxplot(aes(x = lon, y = lat, fill = class), data = df)



## the base_layer argument - faceting
####################################################

df <- data.frame(
  x = rnorm(1000, -95.36258, .2),
  y = rnorm(1000,  29.76196, .2)
)

# no apparent change because ggmap sets maprange = TRUE with extent = 'panel'
ggmap(get_map(), base_layer = ggplot(aes(x = x, y = y), data = df)) +
  geom_point(colour = 'red')

# ... but there is a difference
ggmap(get_map(), base_layer = ggplot(aes(x = x, y = y), data = df), extent = 'normal') +
  geom_point(colour = 'red')

# maprange can fix it (so can extent = 'panel')
ggmap(get_map(), maprange = TRUE, extent = 'normal',
  base_layer = ggplot(aes(x = x, y = y), data = df)) +
  geom_point(colour = 'red')



# base_layer makes faceting possible
df <- data.frame(
  x = rnorm(10*100, -95.36258, .075),
  y = rnorm(10*100,  29.76196, .075),
  year = rep(paste('year',format(1:10)), each = 100)
)
ggmap(get_map(), base_layer = ggplot(aes(x = x, y = y), data = df)) +
  geom_point() +  facet_wrap(~ year)

ggmap(get_map(), base_layer = ggplot(aes(x = x, y = y), data = df), extent = 'device') +
  geom_point() +  facet_wrap(~ year)


## neat faceting examples
####################################################
```

```
# simulated example
df <- data.frame(
  x = rnorm(10*100, -95.36258, .05),
  y = rnorm(10*100,  29.76196, .05),
  year = rep(paste('year',format(1:10)), each = 100)
)
for(k in 0:9){
  df$x[1:100 + 100*k] <- df$x[1:100 + 100*k] + sqrt(.05)*cos(2*pi*k/10)
  df$y[1:100 + 100*k] <- df$y[1:100 + 100*k] + sqrt(.05)*sin(2*pi*k/10)
}

options('device')$device(width = 10.93, height = 7.47)
ggmap(get_map(),
  base_layer = ggplot(aes(x = x, y = y), data = df)) +
  stat_density2d(aes(fill = ..level.., alpha = ..level..),
    bins = 4, geom = 'polygon') +
  scale_fill_gradient2(low = 'white', mid = 'orange', high = 'red', midpoint = 10) +
  scale_alpha(range = c(.2, .75), guide = FALSE) +
  facet_wrap(~ year)



# crime example by month
levels(violent_crimes$month) <- paste(
  toupper(substr(levels(violent_crimes$month),1,1)),
  substr(levels(violent_crimes$month),2,20), sep = ''
)
houston <- get_map(location = 'houston', zoom = 14, source = 'osm', color = 'bw')
HoustonMap <- ggmap(houston,
  base_layer = ggplot(aes(x = lon, y = lat), data = violent_crimes)
  )

HoustonMap +
  stat_density2d(aes(x = lon, y = lat, fill = ..level.., alpha = ..level..),
    bins = I(5), geom = 'polygon', data = violent_crimes) +
  scale_fill_gradient2('Violent\nCrime\nDensity',
    low = 'white', mid = 'orange', high = 'red', midpoint = 500) +
  labs(x = 'Longitude', y = 'Latitude') + facet_wrap(~ month) +
  scale_alpha(range = c(.2, .55), guide = FALSE) +
  ggtitle('Violent Crime Contour Map of Downtown Houston by Month') +
  guides(fill = guide_colorbar(barwidth = 1.5, barheight = 10))



## distances example
##################################################

origin <- 'marrs mclean science, baylor university'
gc_origin <- geocode(origin)
destinations <- data.frame(
  place = c("Administration", "Baseball Stadium", "Basketball Arena",
```

```
    "Salvation Army", "HEB Grocery", "Cafe Cappuccino", "Ninfa's Mexican",
    "Dr Pepper Museum", "Buzzard Billy's", "Mayborn Museum","Flea Market"
  ),
  address = c("pat neff hall, baylor university", "baylor ballpark",
    "ferrell center", "1225 interstate 35 s, waco, tx",
    "1102 speight avenue, waco, tx", "100 n 6th st # 100, waco, tx",
    "220 south 3rd street, waco, tx", "300 south 5th street, waco, tx",
    "100 north jack kultgen expressway, waco, tx",
    "1300 south university parks drive, waco, tx",
    "2112 state loop 491, waco, tx"
  ),
  stringsAsFactors = FALSE
)
gc_dests <- geocode(destinations$address)
(dist <- mapdist(origin, destinations$address, mode = 'bicycling'))

dist <- within(dist, {
  place = destinations$place
  fromlon = gc_origin$lon
  fromlat = gc_origin$lat
  tolon = gc_dests$lon
  tolat = gc_dests$lat
})
dist$minutes <- cut(dist$minutes, c(0,3,5,7,10,Inf),
  labels = c('0-3','3-5', '5-7', '7-10', '10+'))

library(scales)
qmap('baylor university', zoom = 14, legend = 'bottomright',
    base_layer = ggplot(aes(x = lon, y = lat), data = gc_origin)) +
  geom_rect(aes(
    x = tolon, y = tolat,
    xmin = tolon-.00028*nchar(place), xmax = tolon+.00028*nchar(place),
    ymin = tolat-.0005, ymax = tolat+.0005, fill = minutes, colour = 'black'
  ), alpha = .7, data = dist) +
 geom_text(aes(x = tolon, y = tolat, label = place, colour = 'white'), size = 3, data = dist) +
  geom_rect(aes(
    xmin = lon-.004, xmax = lon+.004,
    ymin = lat-.00075, ymax = lat+.00075, colour = 'black'
  ), alpha = .5, fill = I('green'), data = gc_origin) +
  geom_text(aes(x = lon, y = lat, label = 'My Office', colour = 'black'), size = 5) +
  scale_fill_manual('Minutes\nAway\nby Bike',
    values = colorRampPalette(c(muted('green'), 'blue', 'red'))(5)) +
  scale_colour_identity(guide = 'none') +
  theme(
    legend.direction = 'horizontal',
    legend.key.size = unit(2, 'lines')
  ) +
  guides(
    fill = guide_legend(
      title.theme = element_text(size = 16, face = 'bold', colour = 'black'),
      label.theme = element_text(size = 14, colour = 'black'),
      label.position = 'bottom',
      override.aes = list(alpha = 1)
```

```
  )
)
```

```
## darken argument
################################################
ggmap(get_map())
ggmap(get_map(), darken = .5)
ggmap(get_map(), darken = c(.5,'white'))
ggmap(get_map(), darken = c(.5,'red')) # why?


## End(Not run)
```

---

ggmapplot *Don't use this function, use ggmap.*

Description

ggmap plots the raster object produced by get_map.

Usage

```
ggmapplot(ggmap, fullpage = FALSE, base_layer,
  maprange = FALSE, expand = FALSE, ...)
```

Arguments

ggmap	an object of class ggmap (from function get_map)
fullpage	logical; should the map take up the entire viewport?
base_layer	a ggplot(aes(...), ...) call; see examples
maprange	logical for use with base_layer; should the map define the x and y limits?
expand	should the map extend to the edge of the panel? used with base_layer and maprange=TRUE.
...	...

Value

a ggplot object

Author(s)

David Kahle <david.kahle@gmail.com>

See Also

get_map, qmap

Examples

```
## Not run:
this is a deprecated function, use ggmap.

## End(Not run)
```

---

hadley *Highly unofficial ggplot2 image*

Description

Highly unofficial ggplot2 image

Author(s)

Garrett Grolemund <grolemund@gmail.com>

---

inset *Add ggplot2 insets to a map*

Description

This is identical to ggplot2::annotation_custom for use with ggmap

Usage

```
inset(grob, xmin = -Inf, xmax = Inf, ymin = -Inf,
  ymax = Inf)
```

Arguments

grob	grob to display
xmin,xmax	x location (in data coordinates) giving horizontal location of raster
ymin,ymax	y location (in data coordinates) giving vertical location of raster

Details

Most useful for adding tables, inset plots, and other grid-based decorations

Note

annotation_custom expects the grob to fill the entire viewport defined by xmin, xmax, ymin, ymax. Grobs with a different (absolute) size will be center-justified in that region. Inf values can be used to fill the full plot panel

---

inset_raster *Add a raster annotation to a map*

Description

This is a special version of ggplot2::annotation_raster for use with ggmap

Usage

```
inset_raster(raster, xmin, xmax, ymin, ymax)
```

Arguments

raster	raster object to display
xmin,xmax	x location (in data coordinates) giving horizontal location of raster
ymin,ymax	y location (in data coordinates) giving vertical location of raster

Details

Most useful for adding bitmap images

---

legs2route *Convert a leg-structured route to a route-structured route*

Description

Convert a leg-structured route to a route-structured route

Usage

```
legs2route(legsdf)
```

Arguments

legsdf	a legs-structured route, see route

See Also

geom_path in ggplot2

Examples

```
## Not run:

(legs_df <- route('houston','galveston'))
legs2route(legs_df)
(legs_df <- route(
  "marrs mclean science, baylor university",
  "220 south 3rd street, waco, tx 76701", # ninfa's
  alternatives = TRUE))

legs2route(legs_df)




from <- 'houson, texas'
to <- 'waco, texas'
legs_df <- route(from, to)


qmap('college station, texas', zoom = 8) +
  geom_segment(
    aes(x = startLon, y = startLat, xend = endLon, yend = endLat),
    colour = 'red', size = 1.5, data = legs_df
  )
# notice boxy ends

qmap('college station, texas', zoom = 8) +
  geom_leg(
    aes(x = startLon, y = startLat, xend = endLon, yend = endLat),
    colour = 'red', size = 1.5, data = legs_df
  )
# notice overshooting ends

route_df <- legs2route(legs_df)
qmap('college station, texas', zoom = 8) +
  geom_path(
    aes(x = lon, y = lat),
    colour = 'red', size = 1.5, data = route_df, lineend = "round"
  )




## End(Not run)
```

LonLat2XY *Convert a lon/lat coordinate to a tile coordinate*

Description

Convert a lon/lat coordinate to a tile coordinate for a given zoom. Decimal tile coordinates (x, y) are reported.

Usage

```
LonLat2XY(lon_deg, lat_deg, zoom, xpix = 256, ypix = 256)
```

Arguments

lon_deg	longitude in degrees
lat_deg	latitude in degrees
zoom	zoom
xpix	width of tile in pixels
ypix	length of tile in pixels

Value

a data frame with columns X, Y, x, y

Author(s)

David Kahle <david.kahle@gmail.com>, based on function LatLon2XY by Markus Loecher, Sense Networks <markus@sensenetworks.com> in package RgoogleMaps

See Also

http://wiki.openstreetmap.org/wiki/Slippy_map_tilenames

Examples

```
## Not run:
gc <- geocode('baylor university')
LonLat2XY(gc$lon, gc$lat, 10)


## End(Not run)
```

make_bbox *Compute a bounding box*

Description

Compute a bounding box for a given longitude / latitude collection.

Usage

```
make_bbox(lon, lat, data, f = 0.05)
```

Arguments

lon	longitude
lat	latitude
data	(optional) a data frame containing lon and lat as variables
f	number specifying the fraction by which the range should be extended

Examples

```
make_bbox(lon, lat, data = crime)

(lon <- sample(crime$lon, 10))
(lat <- sample(crime$lat, 10))
make_bbox(lon, lat)
make_bbox(lon, lat, f = .10) # bigger box
```

mapdist *Compute map distances using Google*

Description

Compute map distances using Google Maps. Note that in most cases by using this function you are agreeing to the Google Maps API Terms of Service at https://developers.google.com/maps/terms.

Usage

```
mapdist(from, to,
  mode = c("driving", "walking", "bicycling"),
  output = c("simple", "all"), messaging = FALSE,
  sensor = FALSE, language = "en-EN",
  override_limit = FALSE)
```

Arguments

from	name of origin addresses in a data frame (vector accepted)
to	name of destination addresses in a data frame (vector accepted)
output	amount of output
mode	driving, bicycling, or walking
messaging	turn messaging on/off
sensor	whether or not the geocoding request comes from a device with a location sensor
language	language
override_limit	override the current query count (.GoogleDistQueryCount)

Details

if parameters from and to are specified as geographic coordinates, they are reverse geocoded with revgeocode. note that the google maps api limits to 2500 element queries a day.

Value

a data frame (output='simple') or all of the geocoded information (output='all')

Author(s)

David Kahle <david.kahle@gmail.com>

See Also

http://code.google.com/apis/maps/documentation/distancematrix/

Examples

```
## Not run:

from <- c('houston, texas', 'dallas')
to <- 'waco, texas'
mapdist(from, to)
mapdist(from, to, mode = 'bicycling')
mapdist(from, to, mode = 'walking')

from <- c('houston', 'houston', 'dallas')
to <- c('waco, texas', 'san antonio', 'houston')
mapdist(from, to)

mapdist('the white house', 'washington monument', mode = 'walking')

# geographic coordinates are accepted as well
(wh <- as.numeric(geocode('the white house')))
(wm <- as.numeric(geocode('washington monument')))
mapdist(wh, wm, mode = 'walking')
mapdist('the white house', wm, mode = 'walking')
```

```
distQueryCheck()


## End(Not run)
```

`OSM_scale_lookup` *Look up OpenStreetMap scale for a given zoom level.*

Description

Look up OpenStreetMap scale for a given zoom level.

Usage

```
OSM_scale_lookup(zoom = 10)
```

Arguments

`zoom`	google zoom

Details

The calculation of an appropriate OSM scale value for a given zoom level is a complicated task. For details, see http://wiki.openstreetmap.org/wiki/FAQ or http://almien.co.uk/OSM/Tools/Scale/.

Value

scale

Author(s)

David Kahle <david.kahle@gmail.com>

Examples

```
OSM_scale_lookup(zoom = 3)
OSM_scale_lookup(zoom = 10)

## Not run:

ggmapplot(ggmap(zoom =  3, source = 'osm', scale = 47500000), fullpage = TRUE)
ggmapplot(ggmap(zoom =  4, source = 'osm', scale = 32500000), fullpage = TRUE)
ggmapplot(ggmap(zoom =  5, source = 'osm', scale = 15000000), fullpage = TRUE)
ggmapplot(ggmap(zoom =  6, source = 'osm', scale = 10000000), fullpage = TRUE)
ggmapplot(ggmap(zoom =  7, source = 'osm', scale =  6000000), fullpage = TRUE)
ggmapplot(ggmap(zoom =  8, source = 'osm', scale =  2800000), fullpage = TRUE)
ggmapplot(ggmap(zoom =  9, source = 'osm', scale =  1200000), fullpage = TRUE)
ggmapplot(ggmap(zoom = 10, source = 'osm', scale =   575000), fullpage = TRUE)
```

```
ggmapplot(ggmap(zoom = 11, source = 'osm', scale =   220000), fullpage = TRUE)
ggmapplot(ggmap(zoom = 12, source = 'osm', scale =   110000), fullpage = TRUE)
ggmapplot(ggmap(zoom = 13, source = 'osm', scale =    70000), fullpage = TRUE)
ggmapplot(ggmap(zoom = 14, source = 'osm', scale =    31000), fullpage = TRUE)
ggmapplot(ggmap(zoom = 15, source = 'osm', scale =    15000), fullpage = TRUE)
ggmapplot(ggmap(zoom = 16, source = 'osm', scale =     7500), fullpage = TRUE)
ggmapplot(ggmap(zoom = 17, source = 'osm', scale =     4000), fullpage = TRUE)
ggmapplot(ggmap(zoom = 18, source = 'osm', scale =     2500), fullpage = TRUE)
ggmapplot(ggmap(zoom = 19, source = 'osm', scale =     1750), fullpage = TRUE)
ggmapplot(ggmap(zoom = 20, source = 'osm', scale =     1000), fullpage = TRUE)

# the USA
lonR <- c(1.01,.99)*c(-124.73,-66.95)
latR <- c(.99,1.01)*c(24.52, 49.38)
qmap(lonR = lonR, latR = latR, source = 'osm', scale = 325E5)


## End(Not run)
```

---

qmap *Quick map plot*

Description

qmap is a wrapper for ggmap and get_map.

Usage

```
qmap(location = "houston", ...)
```

Arguments

location	character; location of interest
...	stuff to pass to ggmap and get_map.

Value

a ggplot object

Author(s)

David Kahle <david.kahle@gmail.com>

See Also

ggmap and get_map.

Examples

```
## Not run:
qmap(location = 'baylor university')
qmap(location = 'baylor university', zoom = 14)
qmap(location = 'baylor university', zoom = 14, source = 'osm')
qmap(location = 'baylor university', zoom = 14, source = 'osm', scale = 20000)
qmap(location = 'baylor university', zoom = 14, maptype = 'satellite')
qmap(location = 'baylor university', zoom = 14, maptype = 'hybrid')
qmap(location = 'baylor university', zoom = 14, maptype = 'toner', source = 'stamen')
qmap(location = 'baylor university', zoom = 14, maptype = 'watercolor', source = 'stamen')

api_key <- '<your api key here>'
qmap(location = 'baylor university', zoom = 14, maptype = 15434,
  source = 'cloudmade', api_key = api_key)

wh <- geocode('the white house')
qmap('the white house', maprange = TRUE,
  base_layer = ggplot(aes(x=lon, y=lat), data = wh)) +
  geom_point()




## End(Not run)
```

---

qmplot *Quick map plot*

Description

qmplot is the ggmap equivalent to the ggplot2 function qplot and allows for the quick plotting of maps with data/models/etc. qmplot is still experimental.

Usage

```
qmplot(x, y, ..., data, zoom, source = "stamen",
  extent = "device", legend = "right", padding = 0.02,
  darken = c(0, "black"), mapcolor = "color",
  facets = NULL, margins = FALSE, geom = "auto",
  stat = list(NULL), position = list(NULL),
  xlim = c(NA, NA), ylim = c(NA, NA), main = NULL,
  f = 0.05, xlab = deparse(substitute(x)),
  ylab = deparse(substitute(y)))
```

Arguments

x	longitude values
y	latitude values

...	other aesthetics passed for each layer
data	data frame to use (optional). If not specified, will create one, extracting vectors from the current environment.
zoom	map zoom, see get_map
source	map source, see get_map
extent	how much of the plot should the map take up? 'normal', 'panel', or 'device' (default)
legend	'left', 'right' (default), 'bottom', 'top', 'bottomleft', 'bottomright', 'topleft', 'topright', 'none' (used with extent = 'device')
padding	distance from legend to corner of the plot (used with extent = 'device')
darken	vector of the form c(number, color), where number is in [0, 1] and color is a character string indicating the color of the darken. 0 indicates no darkening, 1 indicates a black-out.
mapcolor	color ('color') or black-and-white ('bw')
facets	faceting formula to use. Picks facet_wrap or facet_grid depending on whether the formula is one sided or two-sided
margins	whether or not margins will be displayed
geom	character vector specifying geom to use. defaults to "point"
stat	character vector specifying statistics to use
position	character vector giving position adjustment to use
xlim	limits for x axis
ylim	limits for y axis
main	character vector or expression for plot title
f	number specifying the fraction by which the range should be extended
xlab	character vector or expression for x axis label
ylab	character vector or expression for y axis label

Examples

```
qmplot(lon, lat, data = crime)


# only violent crimes
violent_crimes <- subset(crime,
  offense != 'auto theft' &
  offense != 'theft' &
  offense != 'burglary'
)

# rank violent crimes
violent_crimes$offense <-
  factor(violent_crimes$offense,
    levels = c('robbery', 'aggravated assault',
```

```
      'rape', 'murder')
  )

# restrict to downtown
violent_crimes <- subset(violent_crimes,
  -95.39681 <= lon & lon <= -95.34188 &
   29.73631 <= lat & lat <=  29.78400
)

theme_set(theme_bw())

qmplot(lon, lat, data = violent_crimes, colour = offense, darken = .5,
  size = I(3.5), alpha = I(.6), legend = 'topleft')

qmplot(lon, lat, data = violent_crimes, geom = c('point','density2d'))
qmplot(lon, lat, data = violent_crimes) + facet_wrap(~ offense)
qmplot(lon, lat, data = violent_crimes, extent = 'panel') + facet_wrap(~ offense)
qmplot(lon, lat, data = violent_crimes, extent = 'panel', colour = offense) +
  facet_wrap(~ month)




# doesn't quite work yet....
qmplot(long, lat, xend = long + delta_long,
  yend = lat + delta_lat, data = seals, geom = 'segment')


library(scales)
library(grid)
options('device')$device(width = 4.98, height = 5.97)
qmplot(lon, lat, data = wind, size = I(.5), alpha = I(.5)) +
  ggtitle('NOAA Wind Report Sites')

# thin down data set...
s <- seq(1, 227, 8)
thinwind <- subset(wind,
  lon %in% unique(wind$lon)[s] &
  lat %in% unique(wind$lat)[s]
)

# for some reason adding arrows to the following plot bugs
theme_set(theme_bw(18))
options('device')$device(width = 6.13, height = 7.04)
qmplot(lon, lat, data = thinwind, geom = 'tile', fill = spd, alpha = spd,
    legend = 'bottomleft') +
  geom_leg(aes(xend = lon + delta_lon, yend = lat + delta_lat)) +
  scale_fill_gradient2('Wind Speed\nand\nDirection',
    low = 'green', mid = muted('green'), high = 'red') +
  scale_alpha('Wind Speed\nand\nDirection', range = c(.1, .75)) +
  guides(fill = guide_legend(), alpha = guide_legend())
```

revgeocode *Reverse geocode*

Description

reverse geocodes a longitude/latitude location using Google Maps. Note that in most cases by using this function you are agreeing to the Google Maps API Terms of Service at https://developers.google.com/maps/terms.

Usage

```
revgeocode(location,
  output = c("address", "more", "all"),
  messaging = FALSE, sensor = FALSE,
  override_limit = FALSE)
```

Arguments

location	a location in longitude/latitude format
output	amount of output
messaging	turn messaging on/off
sensor	whether or not the geocoding request comes from a device with a location sensor
override_limit	override the current query count (.GoogleGeocodeQueryCount)

Details

note that the google maps api limits to 2500 queries a day.

Value

depends (at least an address)

Author(s)

David Kahle <david.kahle@gmail.com>

See Also

http://code.google.com/apis/maps/documentation/geocoding/

Examples

```
## Not run:

( gc <- as.numeric(geocode('Baylor University')) )
revgeocode(gc)
revgeocode(gc, output = 'more')
revgeocode(gc, output = 'all')
geocodeQueryCheck()


## End(Not run)
```

`route` *Grab a route from Google*

Description

Grab a route from Google. Note that in most cases by using this function you are agreeing to the Google Maps API Terms of Service at https://developers.google.com/maps/terms.

Usage

```
route(from, to,
  mode = c("driving", "walking", "bicycling"),
  structure = c("legs", "route"),
  output = c("simple", "all"), alternatives = FALSE,
  messaging = FALSE, sensor = FALSE,
  override_limit = FALSE)
```

Arguments

`from`	name of origin addresses in a data frame (vector accepted)
`to`	name of destination addresses in a data frame (vector accepted)
`output`	amount of output
`structure`	structure of output, see examples
`mode`	driving, bicycling, or walking
`alternatives`	should more than one route be provided?
`messaging`	turn messaging on/off
`sensor`	whether or not the geocoding request comes from a device with a location sensor
`override_limit`	override the current query count (.GoogleRouteQueryCount)

Value

a data frame (output='simple') or all of the geocoded information (output='all')

Author(s)

David Kahle <david.kahle@gmail.com>

See Also

https://developers.google.com/maps/documentation/directions/, legs2route, routeQueryCheck, geom_leg

Examples

```
## Not run:

from <- 'houson, texas'
to <- 'waco, texas'
route_df <- route(from, to, structure = 'route')
qmap('college station, texas', zoom = 8) +
  geom_path(
    aes(x = lon, y = lat),  colour = 'red', size = 1.5,
    data = route_df, lineend = 'round'
  )

qmap('college station, texas', zoom = 6) +
  geom_path(
    aes(x = lon, y = lat), colour = 'red', size = 1.5,
    data = route_df, lineend = 'round'
  )

theme_set(theme_bw())
legs_df <- route(from, to, alternatives = TRUE)
p <- qplot(route, minutes, data = legs_df, geom = 'bar',
    stat = 'identity', fill = factor(leg)) +
  scale_fill_discrete(guide = 'none') +
  labs(x = 'Route', y = 'Time (Minutes)', fill = 'Leg') +
  opts(
    title = 'Route Time by Leg',
    plot.background = theme_rect(fill = 'green'),
    axis.text.x = theme_text(colour = 'white'),
    axis.title.x = theme_text(colour = 'white'),
    axis.text.y = theme_text(colour = 'white'),
    axis.title.y = theme_text(colour = 'white', angle = 90),
    plot.title = theme_text(colour = 'white')
  )

route_df <- legs2route(legs_df)
options('device')$device(width = 7.56, height = 6.84)
qmap('college station, texas', zoom = 8, maptype = 'hybrid', fullpage = FALSE) +
  geom_path(
    aes(x = lon, y = lat, colour = route),
    alpha = 3/4, size = 1.75, data = route_df, lineend='round'
  ) +
  labs(x = 'Longitude', y = 'Latitude', colour = 'Routes') +
  opts(title = 'Approximate Routes from Houston to Waco') +
```

```
  ggmap:::annotation_custom(ggplotGrob(p),
    xmin = -96.5, xmax = -94.5, ymin = 30.35, ymax = 32.2)

routeQueryCheck()



(legs_df <- route(
  'marrs mclean science, baylor university',
  '220 south 3rd street, waco, tx 76701', # ninfa's
  alternatives = TRUE))

options('device')$device(width = 11.65, height = 4.17)
qmap('424 clay avenue, waco, tx', zoom = 16, maprange = TRUE, maptype = 'hybrid',
    base_layer = ggplot(aes(x = startLon, y = startLat), data = legs_df)) +
  geom_leg(
    aes(x = startLon, y = startLat, xend = endLon, yend = endLat, colour = route),
    alpha = 3/4, size = 2, data = legs_df
  ) +
  scale_x_continuous(breaks = pretty(c(-97.1325,-97.119),4), lim = c(-97.1325,-97.119)) +
  facet_wrap(~ route) + theme_bw() +
  labs(x = 'Longitude', y = 'Latitude', colour = 'Routes')

## End(Not run)
```

`routeQueryCheck` *Check Google Maps Directions API query limit*

Description

Check Google Maps Directions API query limit

Usage

```
routeQueryCheck()
```

Value

a data frame

Author(s)

David Kahle <david.kahle@gmail.com>

See Also

https://developers.google.com/maps/documentation/directions/

Examples

```
## Not run:
routeQueryCheck()

## End(Not run)
```

---

theme_inset *Make a ggplot2 inset theme.*

Description

theme_inset is a ggplot2 theme geared towards making inset plots.

Usage

```
theme_inset(base_size = 12)
```

Arguments

base_size	base size, not used.

Value

a ggplot2 theme (i.e., a list of class options).

Author(s)

David Kahle <david.kahle@gmail.com>

Examples

```
library(ggplot2)
## Not run:


n <- 50
df <- expand.grid(x = 1:n,y = 1:n)[sample(n^2,.5*n^2),]
qplot(x, y, data = df, geom = 'tile')
qplot(x, y, data = df, geom = 'tile') + theme_nothing()

qplot(1:10, 1:10) +
  annotation_custom(
    grob = ggplotGrob(qplot(1:10,1:10)),
    8, Inf, -Inf, 2
  )

qplot(1:10, 1:10) +
  annotation_custom(
    grob = ggplotGrob(qplot(1:10,1:10) + theme_nothing()),
```

```
    8, Inf, -Inf, 2
  )

qplot(1:10, 1:10) +
  annotation_custom(
    grob = ggplotGrob(qplot(1:10,1:10) + theme_inset()),
    8, Inf, -Inf, 2
  )


## End(Not run)
```

---

theme_nothing *Make a blank ggplot2 theme.*

Description

theme_nothing simply strips all thematic element in ggplot2.

Usage

```
theme_nothing(base_size = 12, legend = FALSE)
```

Arguments

base_size	base size, not used.
legend	should the legend be included?

Value

a ggplot2 theme (i.e., a list of class options).

Author(s)

David Kahle <david.kahle@gmail.com>

Examples

```
library(ggplot2)
## Not run:

n <- 50
df <- expand.grid(x = 1:n,y = 1:n)[sample(n^2,.5*n^2),]
p <- qplot(x, y, data = df, geom = 'tile')
p
p + theme_nothing()
p + theme_nothing(legend = TRUE) # no difference
p +
  scale_x_continuous(expand = c(0,0)) +
```

```
  scale_y_continuous(expand = c(0,0)) +
  theme_nothing()

qplot(1:10,1:10) +
  theme_nothing()  +
  theme(panel.background = element_rect(fill = "black"))


df$class <- factor(sample(0:1, .5*n^2,  replace = TRUE))
p <- qplot(x, y, data = df, geom = "tile", fill = class)
p
p + theme_nothing()
p + theme_nothing(legend = TRUE)

p <- p +
  scale_x_continuous(expand = c(0,0)) +
  scale_y_continuous(expand = c(0,0))
p
p + theme_nothing()
p + theme_nothing(legend = TRUE)


## End(Not run)
```

`wind` *Wind data from Hurricane Ike*

Description

Wind data from Hurricane Ike

Details

Powell, M. D., S. H. Houston, L. R. Amat, and N Morisseau-Leroy, 1998: The HRD real-time hurricane wind analysis system. J. Wind Engineer. and Indust. Aerodyn. 77&78, 53-64

Author(s)

Atlantic Oceanographic and Meteorological Laboratory (AOML), a division of the National Oceanic and Atmospheric Administration (NOAA)

References

http://www.aoml.noaa.gov/hrd/Storm_pages/ike2008/wind.html

XY2LonLat *Convert a tile coordinate to a lon/lat coordinate*

Description

Convert a tile coordinate to a lon/lat coordinate for a given zoom. Decimal tile coordinates are accepted.

Usage

```
XY2LonLat(X, Y, zoom, x = 0, y = 0, xpix = 256,
  ypix = 256)
```

Arguments

X	horizontal map-tile coordinate (0 is map-left)
Y	vertical map-tile coordinate (0 is map-top)
zoom	zoom
x	within tile x (0 is tile-left)
y	within tile y (0 it tile-top)
xpix	width of tile in pixels
ypix	length of tile in pixels

Value

a data frame with columns lon and lat (in degrees)

Author(s)

David Kahle <david.kahle@gmail.com>, based on function XY2LatLon by Markus Loecher, Sense Networks <markus@sensenetworks.com> in package RgoogleMaps

See Also

http://wiki.openstreetmap.org/wiki/Slippy_map_tilenames

Examples

```
## Not run:
XY2LonLat(480, 845, zoom = 11)


## End(Not run)
```

zips — *Zip code data for the Greater Houston Metropolitan Area from the 2000 census*

Description

Zip code data for the Greater Houston Metropolitan Area from the 2000 census

Author(s)

U.S. Census Bureau, Geography Division, Cartographic Products Management Branch

References

http://www.census.gov/geo/www/cob/z52000.html

Index