

# Package ‘prettymapr’

May 9, 2026

**Type** Package

**Title** Scale Bar, North Arrow, and Pretty Margins in R

**Version** 0.2.5

**Author** Dewey Dunnington <dewey@fishandwhistle.net>

**Maintainer** Dewey Dunnington <dewey@fishandwhistle.net>

**Description** Automates the process of creating a scale bar and north arrow in any package that uses base graphics to plot in R. Bounding box tools help find and manipulate extents. Finally, there is a function to automate the process of setting margins, plotting the map, scale bar, and north arrow, and resetting graphic parameters upon completion.

**License** GPL-2

**Imports** digest, rjson, httr, plyr

**Suggests** raster, rosm, sp

**URL** <https://github.com/paleolimbot/prettymapr>

**BugReports** <https://github.com/paleolimbot/prettymapr/issues>

**RoxygenNote** 7.2.3

**Encoding** UTF-8

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2024-02-23 20:20:02 UTC

## Contents

addnortharrow . . . . .	2
addscalebar . . . . .	3
clear_geocode_cache . . . . .	4
geocode . . . . .	4
get_default_geocoder . . . . .	6
makebbox . . . . .	7
mergebbox . . . . .	7

plotscalebar . . . . .	8
prettymap . . . . .	9
scalebarparams . . . . .	11
searchbbox . . . . .	12
set_cached . . . . .	13
zoombbox . . . . .	14

<b>Index</b>	<b>15</b>
--------------	-----------

---

addnortharrow	<i>Plot North Arrow</i>
---------------	-------------------------

---

## Description

Plot a north arrow (pointing directly "up") positioned based on current plot extents.

## Usage

```
addnortharrow(
  pos = "topright",
  padin = c(0.15, 0.15),
  scale = 1,
  lwd = 1,
  border = "black",
  cols = c("white", "black"),
  text.col = "black"
)
```

## Arguments

pos	Where to align the north arrow. One of "bottomleft", "bottomright", "topleft", or "topright".
padin	A vector of length 2 determining the distance in inches between the scalebar and the edge of the plottable area.
scale	Scale the default north arrow to make it bigger or smaller
lwd	The line width outlining the north arrow
border	The line color outlining the north arrow
cols	A vector of length 2 determining the two colors to be drawn for the north arrow
text.col	Color of the "N"

## Examples

```
plot(1:5, 1:5, asp=1)
addnortharrow()
```

---

addscalebar                      *Auto Plot Scalebar*

---

### Description

Automatically determines the geographical scale of the plot and draws a labelled scalebar.

### Usage

```
addscalebar(
  plotunit = NULL,
  plotepsg = NULL,
  widthhint = 0.25,
  unitcategory = "metric",
  htin = 0.1,
  padin = c(0.15, 0.15),
  style = "bar",
  bar.cols = c("black", "white"),
  lwd = 1,
  linecol = "black",
  tick.cex = 0.7,
  labelpadin = 0.08,
  label.cex = 0.8,
  label.col = "black",
  pos = "bottomleft"
)
```

### Arguments

plotunit	The unit which the current plot is plotted in, one of cm, m, km, in, ft, mi. or latlon. This parameter is optional if plotepsg is passed.
plotepsg	The projection of the current plot. If extents are valid lat/lons, the projection is assumed to be lat/lon (EPSG:4326), or Spherical Mercator otherwise (EPSG:3857). This is done to work seamlessly with OpenStreetMap packages.
widthhint	The fraction of the plottable width which the scale bar should (mostly) occupy.
unitcategory	One of "metric" or "imperial"
htin	Height (in inches) of the desired scale bar
padin	A vector of length 2 determining the distance in inches between the scalebar and the edge of the plottable area.
style	One of "bar" or "ticks".
bar.cols	If style=="bar", the colors to be repeated to make the bar.
lwd	The line width to use when drawing the scalebar
linecol	The line color to use when drawing the scalebar
tick.cex	If style=="ticks", the height of interior ticks.

labelpadin	The distance between the end of the scalebar and the label (inches)
label.cex	The font size of the label
label.col	The color of the label
pos	Where to align the scalebar. One of "bottomleft", "bottomright", "topleft", or "topright".

### Examples

```
plot(1:5, 1:5, asp=1)
addscalebar(plotunit="m")
```

---

clear\_geocode\_cache     *Clear cached results*

---

### Description

Clears the local cache of downloaded files (by default, an environment in the package namespace). Clearing a directory cache will result in all files with the extension ".cached" being deleted from that directory.

### Usage

```
clear_geocode_cache(cache = NA)
```

### Arguments

cache                    An environment, a directory name, or NA to clear the default internal cache

### Examples

```
clear_geocode_cache()
```

---

geocode                    *Geocode Locations*

---

### Description

Geocode locations using the Google Web API, the PickPoint.io API, or the Data Science Toolkit API. For large requests you should really use your own API key if you are using the default (pick-point). Note that the Google Terms seem to indicate that you cannot place locations obtained from their API on non-google maps. Locations are all geocoded with errors kept quiet, which may result in list output containing items with a \$status element describing the error message, or data frame output containing a non-OK status in the status column.

**Usage**

```

geocode(
  location,
  output = c("data.frame", "list"),
  source = "default",
  messaging = NULL,
  limit = 1,
  key = NULL,
  quiet = TRUE,
  cache = NA,
  progress = c("time", "text", "none"),
  ...
)

```

**Arguments**

location	A character vector (or an object that can be coerced to one) of locations to pass to the geocoding API.
output	One of <code>data.frame</code> or <code>list</code> . If <code>data.frame</code> , the results are distilled into columns: <code>query</code> , <code>source</code> , <code>status</code> , <code>rank</code> , <code>lon</code> , <code>lat</code> , <code>address</code> , <code>bbox_n</code> , <code>bbox_e</code> , <code>bbox_s</code> , and <code>bbox_w</code> . Other columns may also exist for certain API types. The data frame will have the same number of rows as the length of the input vector, and will always have the columns <code>query</code> , <code>source</code> , <code>status</code> , <code>lon</code> and <code>lat</code> . If <code>output='list'</code> , the raw JSON output from the geocoding API is returned as a <code>list</code> (containing lists). The list output of a failed geocode return varies by API type, but the length of the output list is guaranteed to be the same as the input vector.
source	One of "default", "google", "pickpoint", or "dsk". If "default", the function calls <code>getOption("prettymapr.geosource")</code> or chooses "pickpoint" if none is set. If using "pickpoint", please sign up for your own (free) API key to avoid using the default excessively.
messaging	TRUE if verbose messaging is desired (now deprecated, use <code>'quiet = FALSE'</code> instead).
limit	The number of results to return per query. This refers to individual locations, for which ambiguous queries may return multiple results (e.g. Halifax, Nova Scotia; Halifax, United Kingdom, etc.). The default is 1. Pass 0 if no limit on queries is desired.
key	API key if <code>source="pickpoint"</code> .
quiet	By default, error messages are suppressed, and are instead included in the output as objects with a <code>\$status</code> describing the error (list output) or the appropriate value in the <code>'status'</code> column (data frame output).
cache	The cache to use. Use NA for the internal cache (keeps first 1000 results), or a directory name (e.g. <code>'geo.cache'</code> ), which keeps an unlimited number of results. Use <a href="#">clear_geocode_cache</a> to clear the cache.
progress	A plyr status bar, one of "time", "text", or "none". Passing <code>quiet = FALSE</code> will also disable the progress bar.

... A number of key/value pairs to append to the URL, specifying further options specific to each API. Google users may wish to provide sensor, client and signature arguments for use with the enterprise version with the API, or to specify additional constraints on geocoding.

### Value

A list or data.frame; see documentation for output argument.

### Examples

```
# don't test to speed up checking time

geocode("wolfville, ns")
geocode("wolfville, ns", output="list")
geocode("halifax", limit=0)
geocode("Paddy's Pub Wolfville NS", source="google")
geocode(c("Houston, TX", "San Antonio TX", "Cleveland OH"), source="google")

#fails quietly
geocode("don't even think about geocoding this")
geocode("don't even think about geocoding this", output="list")
```

---

get\_default\_geocoder *Get/Set the default geocoder*

---

### Description

The [geocode](#) function can use google, pickpoint, or data science toolkit to turn human-readable names into coordinates. Use these methods to get/set the default source. These will need to be called once per namespace load.

### Usage

```
get_default_geocoder()

set_default_geocoder(geocoder)
```

### Arguments

geocoder The new source to use. One of "pickpoint", "google", or "dsk".

### Examples

```
get_default_geocoder()
set_default_geocoder("google")
(set_default_geocoder(NULL))
```

---

makebbox	<i>Create a Bounding Box</i>
----------	------------------------------

---

**Description**

Convenience method to create a bounding box like that returned by `sp::bbox()`. To generate a bounding box from lists of lat/lon values use `sp::bbox(cbind(lons, lats))`.

**Usage**

```
makebbox(n, e, s, w)
```

**Arguments**

n	North bounding latitude
e	East bounding longitude
s	South bounding latitude
w	West bounding longitude

**Value**

A 2x2 matrix describing a bounding box like that returned by `sp::bbox()`

**See Also**

`sp::bbox`

**Examples**

```
makebbox(45.125, -64.25, 44.875, -64.75)
```

---

mergebbox	<i>Combine bounding boxes</i>
-----------	-------------------------------

---

**Description**

Create a single bounding box that encloses all of the bounding boxes.

**Usage**

```
mergebbox(...)
```

**Arguments**

... An arbitrary number of bounding boxes as generated by `sp::bbox`, [makebbox](#) or [searchbbox](#)

**Value**

A single bounding box that contains all of its arguments.

**Examples**

```
box1 <- makebbox(45, -64, 44, -65)
box2 <- makebbox(45.5, -64.5, 44.5, -65.6)
mergebbox(box1, box2)
```

---

plotscalebar

*Raw Plot Scale Bar*


---

**Description**

Just in case anybody is hoping to draw a custom scalebar, this is the method used to plot it. If you don't know what this is, you should probably be using [addscalebar](#).

**Usage**

```
plotscalebar(
  x,
  y,
  ht,
  params,
  style = "bar",
  adj = c(0, 0),
  tick.cex = 0.7,
  bar.cols = c("black", "white"),
  lwd = 1,
  linecol = "black"
)
```

**Arguments**

x	The position (user) to draw the scale bar
y	The position (user) to draw the scale bar
ht	The height(in user coordinates) to draw the scale bar
params	Scalebar parameters as generated by <a href="#">scalebarparams</a>
style	One of bar or ticks
adj	Where to align the scale bar relative to x and y

tick.cex	If style=="ticks", the height of interior ticks.
bar.cols	A vector of color names to be repeated for a bar style scalebar.
lwd	Passed when drawing lines associated with the scalebar
linecol	Passed when drawing lines associated with the scalebar

**See Also**

[addscalebar](#)

---

```
prettymap
```

```
Plot A Pretty Map
```

---

**Description**

This function executes everything in `plotexpression`, then draws north arrow and scale bar using [addnortharrow](#) and [addscalebar](#). Specify that plot is in a non lat/lon projection by passing `scale.plotepsg=...` or `plotunit="m"`.

**Usage**

```
prettymap(
  plotexpression,
  oma = c(0, 0, 0, 0),
  mai = c(0, 0, 0, 0),
  drawbox = FALSE,
  box.lwd = 1,
  drawscale = TRUE,
  scale.pos = "bottomleft",
  scale.htin = 0.1,
  scale.widthhint = 0.25,
  scale.unitcategory = "metric",
  scale.style = "bar",
  scale.bar.cols = c("black", "white"),
  scale.lwd = 1,
  scale.linecol = "black",
  scale.padin = c(0.15, 0.15),
  scale.labelpadin = 0.08,
  scale.label.cex = 0.8,
  scale.label.col = "black",
  scale.plotunit = NULL,
  scale.plotepsg = NULL,
  scale.tick.cex = 0.8,
  drawarrow = FALSE,
  arrow.pos = "topright",
  arrow.scale = 1,
  arrow.padin = c(0.15, 0.15),
```

```

    arrow.lwd = 1,
    arrow.cols = c("white", "black"),
    arrow.border = "black",
    arrow.text.col = "black",
    title = NULL,
    ...
)

```

## Arguments

plotexpression	An expression to plot the map, can be in brackets. e.g. <code>plot(stuff); text(places, "readme!")</code> or <code>{plot(stuff); text(places, "readme!")}</code>
oma	A vector of length 4 describing the outer margin area. See documentation for <code>graphics::par</code> .
mai	A vector of length 4 describing the margin area in inches. See documentation for <code>graphics::par</code> .
drawbox	TRUE if box should be drawn around map, FALSE otherwise.
box.lwd	The line width of the box
drawscale	TRUE if scalebar should be drawn, FALSE otherwise.
scale.pos	Where to align the scalebar. One of "bottomleft", "bottomright", "topleft", or "topright".
scale.htin	Height (in inches) of the desired scale bar
scale.widthhint	The fraction of the plottable width which the scale bar should (mostly) occupy.
scale.unitcategory	One of "metric" or "imperial"
scale.style	One of "bar" or "ticks".
scale.bar.cols	If <code>style=="bar"</code> , the colors to be repeated to make the bar.
scale.lwd	The line width to use when drawing the scalebar
scale.linecol	The line color to use when drawing the scalebar
scale.padin	A vector of length 2 determining the distance in inches between the scalebar and the edge of the plottable area.
scale.labelpadin	The distance between the end of the scalebar and the label (inches)
scale.label.cex	The font size of the label
scale.label.col	The color of the label
scale.plotunit	The unit which the current plot is plotted in, one of cm, m, km, in, ft, mi. or latlon. This parameter is optional if <code>plotepsg</code> is passed.
scale.plotepsg	The projection of the current plot. If extents are valid lat/lons, the projection is assumed to be lat/lon (EPSG:4326), or Spherical Mercator otherwise (EPSG:3857). This is done to work seamlessly with OpenStreetMap packages.

<code>scale.tick.cex</code>	If <code>style=="ticks"</code> , the height of interior ticks.
<code>drawarrow</code>	TRUE if north arrow should be drawn, FALSE otherwise
<code>arrow.pos</code>	Where to align the north arrow. One of "bottomleft", "bottomright", "topleft", or "topright".
<code>arrow.scale</code>	Scale the default north arrow to make it bigger or smaller
<code>arrow.padin</code>	A vector of length 2 determining the distance in inches between the scalebar and the edge of the plottable area.
<code>arrow.lwd</code>	The line width outlining the north arrow
<code>arrow.cols</code>	A vector of length 2 determining the two colors to be drawn for the north arrow
<code>arrow.border</code>	The line color outlining the north arrow
<code>arrow.text.col</code>	Color of the "N"
<code>title</code>	Plot title, or NULL if none is desired.
<code>...</code>	Further graphical parameters to set while executing plotting code

### Examples

```
prettymap(plot(1:5, 1:5, asp=1), scale.plotunit="cm", drawarrow=FALSE)
#add a title
prettymap(plot(1:5, 1:5, asp=1), title="My Plot")
```

---

scalebarparams

*Get Scale Bar Parameters*

---

### Description

Get default scale bar parameters based on the current plot (i.e. `par("usr")`). The algorithm attempts to detect the best equally divisible distance to use for the scale bar, and returns a list object with attributes that allow any type of scale bar to be drawn. The only way to manipulate the values chosen by the algorithm is to change the `widthhint` argument. For generic XY plots, pass `plotunit`.

### Usage

```
scalebarparams(
  plotunit = NULL,
  plotepsg = NULL,
  widthhint = 0.25,
  unitcategory = "metric",
  extents = graphics::par("usr")
)
```

**Arguments**

plotunit	The unit which the current plot is plotted in, one of cm, m, km, in, ft, mi. or latlon. This parameter is optional if plotepsg is passed.
plotepsg	The projection of the current plot. If extents are valid lat/lons, the projection is assumed to be lat/lon (EPSG:4326), or Spherical Mercator otherwise (EPSG:3857). This is done to work seamlessly with OpenStreetMap packages.
widthhint	The fraction of the plottable width which the scale bar should (mostly) occupy.
unitcategory	One of "metric" or "imperial"
extents	The plot extents

**Value**

a list of parameters: \$widthu (width of the scalebar in human readable units); \$unit (the human readable unit); \$majordivu (the size of the divisions in human readable units); \$majordivs (the number of divisions); \$widthplotunit (width of the scalebar in plotting units); \$majordivplotunit (the width of divisions in plotting units); \$labeltext (label text); and extents the user extents (par('usr')) that were used to calculate the parameters.

**See Also**

[addscalebar](#)

**Examples**

```
plot(1:5, 1:5, asp=1)
scalebarparams(plotunit="m")
```

---

searchbbox

*Query The Interwebs For A Bounding Box*

---

**Description**

Use the PickPoint.io API or Google API to retrieve a bounding box for the given query. Note that if you would like to use google as a source, you must agree to the Google API terms and conditions.

**Usage**

```
searchbbox(querystring, ...)
```

**Arguments**

querystring	The search query. Passing a vector in will find the bounding box that contains all bounding boxes returned.
...	Additional paramters to be passed on to <a href="#">geocode</a> . Passing source="google" may be useful if google is desired as a source. Use options(prettymapr.geosource="google") to permanently use google as a source.

**Value**

A 2x2 matrix describing a bounding box like that returned by `sp::bbox()`

**Examples**

```
#don't test to speed up checking time

searchbbox("kings county, NS")
searchbbox("University Ave. Wolfville NS", source="google")
searchbbox("Wolfville ns", source="google")
searchbbox(c("Vermont", "Nova Scotia"))
```

---

set\_cached

*Internal cache methods*

---

**Description**

For internal use only.

**Usage**

```
set_cached(cache, url, data, ...)
```

```
get_cached(cache, url, ...)
```

```
clear_cache(cache, ...)
```

```
cache_size(cache, ...)
```

**Arguments**

cache, url, data, ...

For internal use only

**Value**

Values for internal use

zoombbox

*Zoom the extents of a bounding box*

---

**Description**

Manipulate the extents of a bounding box by zooming and moving an existing bbox. This is helpful when manipulating the extents of a plot created by `canvec.qplot()`

**Usage**

```
zoombbox(bbox, factor = 1, offset = c(0, 0))
```

**Arguments**

<code>bbox</code>	An existing bbox
<code>factor</code>	A factor to zoom by. $>1$ will zoom in, $<1$ will zoom out. If a vector is passed, the first element will zoom the X extent, the second element will zoom the Y extent.
<code>offset</code>	A vector describing the X and Y offset that should be applied.

**Value**

A zoomed bounding box.

**Examples**

```
box1 <- makebbox(45, -64, 44, -65)
zoombbox(box1, c(.2, .5))
```

# Index

addnortharrow, [2](#), [9](#)  
addscalebar, [3](#), [8](#), [9](#), [12](#)

cache\_size (set\_cached), [13](#)  
clear\_cache (set\_cached), [13](#)  
clear\_geocode\_cache, [4](#), [5](#)

geocode, [4](#), [6](#), [12](#)  
get\_cached (set\_cached), [13](#)  
get\_default\_geocoder, [6](#)

makebbox, [7](#), [8](#)  
mergebbox, [7](#)

plotscalebar, [8](#)  
prettymap, [9](#)

scalebarparams, [8](#), [11](#)  
searchbbox, [8](#), [12](#)  
set\_cached, [13](#)  
set\_default\_geocoder  
    (get\_default\_geocoder), [6](#)

zoombbox, [14](#)