Package ‘plotdap’

May 9, 2026

Title Easily Visualize Data from 'ERDDAP™' Servers via the 'rerddap' Package

Version 1.2.0

Date 2026-05-04

Description Easily visualize and animate 'tabledap' and 'griddap' objects obtained via the 'rerddap' package in a simple one-line command, using either base graphics or 'ggplot2' graphics. 'plotdap' handles extracting and reshaping the data, map projections and continental outlines. Optionally the data can be animated through time using the 'gganmiate' package.

License MIT + file LICENSE

URL https://github.com/rmendels/plotdap

BugReports https://github.com/rmendels/plotdap/issues

Depends R (>= 4.3.0)

Imports cmocean, dplyr, gganimate, ggnewscale, ggplot2 (>= 3.1.0), isoband, lazyeval, lubridate, magrittr, mapdata, maps, raster, rerddap (>= 1.0.0), rlang, scales, sf, tidyr, viridis

Suggests Cairo, knitr, plotly, rmarkdown, testthat

LazyData true

Encoding UTF-8

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Carson Sievert [aut], Roy Mendelssohn [aut, ctb, cre]

Maintainer Roy Mendelssohn <roy.mendelssohn@noaa.gov>

Repository CRAN

Date/Publication 2026-05-09 14:41:33 UTC

Contents

add_ggplot . . . 2
add_griddap . . . 3
add_griddap_contours . . . 4
add_tabledap . . . 6
bbox_set . . . 7
murSST . . . 8
plotdap . . . 9
print.ggplotdap . . . 10
print.plotdap . . . 11
QMwind . . . 11
sardines . . . 12

Index **13**

`add_ggplot` *Add ggplot2 elements to a plotdap object*

Description

`add_ggplot` allows for plotdap ggplot maps to be modified by further ggplot2 settings

Usage

```
add_ggplot(plot, ...)
```

Arguments

`plot`	a plotdap object.
`...`	arguments passed along to `geom_sf()` (if `method='ggplot2'`, otherwise ignored).

Value

A plotdap object

Examples

```
p <- plotdap(
  crs = "+proj=laea +y_0=0 +lon_0=155 +lat_0=-90 +ellps=WGS84 +no_defs")
p <- add_ggplot(
 p,
 ggplot2::theme_bw()
)
```

add_griddap *Add rerddap::griddap() data to a plotdap map*

Description

add_griddap adds the data from an 'rerddap::griddap() call to a 'plotdap' map

Usage

```
add_griddap(
  plot,
  grid,
  var,
  fill = "viridis",
  maxpixels = 10000,
  time = mean,
  animate = FALSE,
  cumulative = FALSE,
  ...
)
```

Arguments

plot	a plotdap object.
grid	a griddap object.
var	a formula defining a variable, or function of variables to visualize.
fill	either a character string of length 1 matching a name in the package cmocean or a vector of color codes. This defines the colorscale used to encode values of var.
maxpixels	integer > 0. Maximum number of cells to use for the plot. If maxpixels < ncell(x), sampleRegular is used before plotting. If gridded=TRUE maxpixels may be ignored to get a larger sample
time	how to resolve multiple time frames. Choose one of the following: • A function to apply to each observation at a particular location (mean is the default). • A character string (of length 1) matching a time value.
animate	whether to animate over the time variable (if it exists). Currently only implemented for method='ggplot2' and requires the gganimate package.
cumulative	- if animation should be cumulative -default FALSE
...	arguments passed along to geom_sf() (if method='ggplot2', otherwise ignored).

Value

A plotdap object

Examples

```
# base plotting tends to be faster,
# but is less extensible plotdap("base")

# actual datasets in data folder to meet execution timings


 # murSST <- rerddap::griddap(
 #  ' jplMURSST41', latitude = c(35, 40), longitude = c(-125, -120.5),
 #   time = c('last', 'last'), fields = 'analysed_sst'
 # )

 # QMwind <- rerddap::griddap(
 #  'erdQMwindmday', time = c('2016-11-16', '2017-01-16'),
 #  latitude = c(30, 50), longitude = c(210, 240),
 #  fields = 'x_wind'
 #  )

p <- plotdap(crs = "+proj=robin")
p <- add_griddap(p, murSST, ~analysed_sst)

 # p <- plotdap(mapTitle = "Average wind over time")
 # p <- add_griddap(p, QMwind, ~x_wind)

# p <- plotdap("base", crs = "+proj=robin")
# p <- add_griddap(p, murSST, ~analysed_sst)

# layer tables on top of grids
require(magrittr)
p <- plotdap("base") %>%
  add_griddap(murSST, ~analysed_sst) %>%
  add_tabledap(sardines, ~subsample_count)

# multiple time periods
p <- plotdap("base", mapTitle = "Average wind over time")
p <- add_griddap(p, QMwind, ~x_wind)
```

`add_griddap_contours` *Add contour lines to a plotdap object*

Description

Computes isolines from a griddap data frame and appends them as a `geom_sf()` layer to the **plotdap object's internal ggplot**. Call this *before* `add_ggplot()` or `print()`.

Usage

```
add_griddap_contours(
```

```
  p,
  data,
  var,
  n_breaks = 10L,
  breaks = NULL,
  lon_col = "longitude",
  lat_col = "latitude",
  color = "black",
  linewidth = 0.3,
  alpha = 1,
  crs = 4326L
)
```

Arguments

p	a plotdap object (result of `plotdap()` piped through `add_griddap()`). **Not** the result of `add_ggplot()`.
data	data frame typically `dat$data` from a `griddap()` call.
var	character; name of the variable column to contour.
n_breaks	integer; number of evenly-spaced interior contour levels (ignored when `breaks` is supplied). Default 10.
breaks	numeric vector of explicit contour levels, or `NULL`.
lon_col	name of longitude column (default `"longitude"`).
lat_col	name of latitude column (default `"latitude"`).
color	contour line color (default `"black"`).
linewidth	contour line width (default `0.3`).
alpha	opacity 01 (default `1`).
crs	integer EPSG code for the data (default `4326`).

Value

the plotdap object with contour lines added to its internal ggplot. Pipe the result to `add_ggplot()` or `print()` as usual.

Examples

```
## Not run:
  library(rerddap)
  library(plotdap)

  myURL <- "https://coastwatch.pfeg.noaa.gov/erddap/"
  info <- rerddap::info("jplMURSST41", url = myURL)
  dat  <- griddap(info,
            latitude  = c(30, 50),
            longitude = c(-140, -110),
            time      = c("2020-06-15", "2020-06-15"),
            fields    = "analysed_sst")
```

```
  p <- plotdap() |>
         add_griddap(dat, ~analysed_sst)

  ## Default: 10 evenly-spaced contour levels, then render
  p |> add_griddap_contours(dat$data, "analysed_sst") |> add_ggplot()

  ## Explicit break values
  p |> add_griddap_contours(dat$data, "analysed_sst",
                            breaks = c(10, 12, 14, 16, 18, 20)) |> add_ggplot()

  ## More levels, thinner grey lines
  p |> add_griddap_contours(dat$data, "analysed_sst",
                            n_breaks = 20, color = "grey40", linewidth = 0.2) |>
       add_ggplot()

## End(Not run)
```

---

add_tabledap *Add rerddap::tabledap data to a plotdap map*

Description

add_tabledap adds the data from an 'rerddap::tabledap()' call to a 'plotdap' map

Usage

```
add_tabledap(
  plot,
  table,
  var,
  color = c("#132B43", "#56B1F7"),
  size = 1.5,
  shape = 19,
  animate = FALSE,
  cumulative = FALSE,
  ...
)
```

Arguments

plot	a plotdap object.
table	a tabledap object.
var	a formula defining a variable, or function of variables to visualize.
color	either a character string of length 1 matching a name in cmocean or a vector of color codes. This defines the colorscale used to encode values of var.
size	the size of the symbol.

shape	the shape of the symbol. For valid options, see the 'pch' values section on points. `plot(0:25, 0:25, pch = 0:25)` also gives a quick visual of the majority of possibilities.
animate	whether to animate over the `time` variable (if it exists). Currently only implemented for `method='ggplot2'` and requires the gganimate package.
cumulative	- if animation should be cumulative -default FALSE
...	arguments passed along to `geom_sf()` (if `method='ggplot2'`, otherwise ignored).

Value

A plotdap object

Examples

```
# base plotting tends to be faster,
# but is less extensible plotdap("base")

# test datasets in data folder to meet execution timings
# code given to extract the data


sardines <- rerddap::tabledap(
 'FRDCPSTrawlLHHaulCatch',
 fields = c('latitude',  'longitude', 'time', 'scientific_name', 'subsample_count'),
  'time>=2010-01-01', 'time<=2012-01-01',
  'scientific_name="Sardinops sagax"'
  )


p <- plotdap()
p1 <- add_tabledap(p, sardines, ~subsample_count)
p2 <- add_tabledap(p, sardines, ~log2(subsample_count))

# using base R plotting
p <- plotdap("base")
p <- add_tabledap(p, sardines, ~subsample_count)

# robinson projection
p <- plotdap(crs = "+proj=robin")
p <- add_tabledap(p, sardines, ~subsample_count)
```

`bbox_set` *change bounding box in plotdap object*

Description

bbox_setchanges the bounding box in an plotdap object. Particularly needed if using gganimate::animate()

Usage

```
bbox_set(
  plotobj,
  landmask = TRUE,
  xlim = NULL,
  ylim = NULL,
  interactive = FALSE
)
```

Arguments

plotobj	valid plotdap object
landmask	optional: if true land is plotted over the data
xlim	optional: new x-values of the bounding box otherwise taken from the plotdap object
ylim	optional: new y-values of the bounding box otherwise taken from the plotdap object
interactive	optional: if true creates an interactive version of the graphic

Value

a ggplot object with reset limits. If interactive is true, then returns a ggplot object that will work with the ‘plotly‘ package.

Examples

```
p <- plotdap()
p <- add_tabledap(p, sardines, ~subsample_count)
xlim = c(-125, -115)
ylim <- c(30., 50.)
p <- bbox_set(p, xlim = xlim, ylim = ylim)
```

---

murSST *murSST Data*

Description

pre-Download of murSST in ‘add_griddap()‘ example so that example can run within CRAN Time limits

Usage

```
murSST
```

Format

An object of class `griddap_nc` (inherits from `nc`, `data.frame`) with 0 rows and 2 columns.

Details

obtained using the ‘rerddap‘ command murSST <- griddap(’jplMURSST41’, latitude = c(22, 51), longitude = c(-140, -105), time = c(’last’, ’last’), fields = ’analysed_sst’)

---

plotdap *Visualize rerddap data*

Description

Visualize data returned from rerddap servers. Use `plotdap()` to initialize a plot, specify the plotting method (specifically, ’base’ or ’ggplot2’), and set some global options/parameters. Then use `add_tabledap()` and/or `add_griddap()` to add "layers" of actual data to be visualized.

Usage

```
plotdap(
  method = c("ggplot2", "base"),
  mapData = maps::map("world", plot = FALSE, fill = TRUE),
  crs = NULL,
  datum = sf::st_crs(4326),
  mapTitle = NULL,
  mapFill = "gray80",
  mapColor = "gray90",
  ...
)
```

Arguments

`method`	the plotting method. Currently ggplot2 and base plotting are supported.
`mapData`	an object coercable to an sf object via `st_as_sf()`.
`crs`	a coordinate reference system: integer with the epsg code, or character with proj4string.
`datum`	crs that provides datum to use when generating graticules. Set to `NULL` to hide the graticule.
`mapTitle`	a title for the map.
`mapFill`	fill used for the map.
`mapColor`	color used to draw boundaries of the map.
`...`	arguments passed along to `geom_sf()` (if `method='ggplot2'`, otherwise ignored).

Details

The "ggplot2" method is slower than "base" (especially for high-res grids/rasters), but is more flexible/extensible. Additional ggplot2 layers, as well as scale defaults, labels, theming, etc. may be modified via the `add_ggplot()` function. See the mapping vignette for an introduction and overview of rerddap's visualization methods – `browseVignettes(package = "rerddap")`.

Value

A plotdap object

Author(s)

Carson Sievert

See Also

`tabledap()`, `griddap()`

Examples

```
# base plotting tends to be faster (especially for grids),
# but is less extensible plotdap("base")

 plotdap()
 plotdap("base")
```

---

print.ggplotdap *Print a ggplot plotdap object*

Description

Print a ggplot plotdap object

Usage

```
## S3 method for class 'ggplotdap'
print(x, ...)
```

Arguments

`x`	a ggplotdap object
`...`	currently unused

print.plotdap — *Print a plotdap object*

Description

Print a plotdap object

Usage

```
## S3 method for class 'plotdap'
print(x, ...)
```

Arguments

x	a plotdap object
...	currently unused

QMwind — *QMwind Data*

Description

pre-Download of QMwind in ‘add_griddap()‘ example so that example can run within CRAN Time limits

Usage

```
QMwind
```

Format

An object of class `griddap_nc` (inherits from `nc`, `data.frame`) with 0 rows and 2 columns.

Details

obtained using the ‘rerddap‘ command wind <- griddap(’erdQMwindmday’, time = c(’2016-11-16’, ’2017-01-16’), latitude = c(30, 50), longitude = c(210, 240), fields = ’x_wind’))

`sardines` *sardine Data*

Description

pre-Download of sardine data in ‘add_tabledap()‘ example so that example can run within CRAN Time limits

Usage

```
sardines
```

Format

An object of class `tabledap` (inherits from `data.frame`) with 56 rows and 5 columns.

Details

obtained using the ‘rerddap‘ command sardines <- tabledap(’FRDCPSTrawlLHHaulCatch’, fields = c(’latitude’, ’longitude’, ’time’, ’ scientific_name’, ’subsample_count’), ’time>=2010-01-01’, ’time<=2012-01-01’, ’scientific_name="Sardinops sagax"’))

Index

∗ **datasets**
    murSST, 8
    QMwind, 11
    sardines, 12

add_ggplot, 2
add_griddap, 3
add_griddap_contours, 4
add_tabledap, 6

bbox_set, 7

griddap, *3*

mean, *3*
murSST, 8

plotdap, *3*, *6*, 9
points, *7*
print.ggplotdap, 10
print.plotdap, 11

QMwind, 11

sardines, 12

tabledap, *6*