Package ‘basf’

December 9, 2020

Version 0.2.0
Title Plot Simple Features with ‘base’ Sensibilities
Description Resurrects the standard plot for shapes established by the ‘base’ and ‘graphics’ packages. This is suited to workflows that require plotting using the established and traditional idioms of plotting spatially coincident data where it belongs. This package depends on ‘sf’ and only replaces the plot method.
License GPL-3
Encoding UTF-8
LazyData true
ByteCompile true
RoxygenNote 7.1.1
Depends sf
URL https://github.com/mdsumner/basf
BugReports https://github.com/mdsumner/basf/issues
Suggests spelling, testthat (>= 2.1.0), vdiffr
Language en-US
Imports tibble, raster
NeedsCompilation no
Author Michael Sumner [aut, cre]
Maintainer Michael Sumner <mdsumner@gmail.com>
Repository CRAN
Date/Publication 2020-12-09 08:30:18 UTC

R topics documented:

plot.sf ................................................................. 2
read_ext ................................................................. 2

Index 4
plot.sf

Plot simple features simply

Description

Overrides `sf::plot.sf` and wraps the call to `plot(st_geometry(x))`. When working with spatial data it’s often useful to create maps where we overlay spatially coincident data in a plot.

Usage

```r
## S3 method for class 'sf'
plot(x, ...)
```

Arguments

- `x`: sf object (the data.frame one)
- `...`: arguments passed to `[sf::plot_sfc]`

Details

This is used so that plots aren’t always faceted by all attributes, coloured by them, or leaving the plot unusable for subsequent additional drawing. The key feature is that we don’t have to change our behaviour and good habits depending on the format in use.

Value

used for its side effects of creating a plot

Examples

```r
x <- read_sf(system.file("shape/nc.shp", package="sf"))
## all we've changed is the plot command
plot(x)
## we can overplot without format-specific acrobatics
plot(x[sample(1:nrow(x), 10), ], col = rainbow(10), add = TRUE)
abline(v = 34); axis(2)
```

---

read_ext

Read GDAL vector data

Description

Read vector shape data with optional extent filter.

Usage

```r
read_ext(x, ext = NULL, ...)
```
**read_ext**

**Arguments**

- `x` used as `sf::st_read()` dsn argument
- `ext` optional extent (as per raster package)
- `...` arguments passed to `sf::st_read()`

**Details**

The extent coordinates must be in the same projection as the source, or the result could be wrong.

Uses the `sf` package to pass the extent down to GDAL’s ExecuteSQL as WKT, use anything with an extent that the `raster` package understands.

**Value**

`sf` object, see `sf::st_read()`

**Examples**

```r
## Not run:
\donttest{
  # read_ext("myshapefile", raster::extent(100, 120, -40, -30))
}
## End(Not run)
```
Index

plot.sf, 2
read_ext, 2