Package 'mapedit'

February 2, 2020

Title Interactive Editing of Spatial Data in R

Description Suite of interactive functions and helpers for selecting and editing geospatial data.

Version 0.6.0

Date 2020-02-01

URL https://github.com/r-spatial/mapedit

BugReports https://github.com/r-spatial/mapedit/issues

License MIT + file LICENSE

Depends R (>= 3.1.0)

Imports dplyr, htmltools (>= 0.3), htmlwidgets, jsonlite, leafem, leaflet (>= 2.0.1), leaflet.extras (>= 1.0), leafpm, mapview, methods, miniUI, raster, scales, sf (>= 0.5-2), shiny, sp

Suggests crayon

Enhances geojsonio

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

NeedsCompilation no

Author Tim Appelhans [aut, cre],
Kenton Russell [aut],
Lorenzo Busetto [aut],
Josh O'Brien [ctb],
Jakob Gutschlhofer [ctb]

Maintainer Tim Appelhans <tim.appelhans@gmail.com>

Repository CRAN

Date/Publication 2020-02-02 17:20:02 UTC

R topics documented:

`mapedit-package` *mapedit: interactive editing and selection for geospatial data*

Description

mapedit, a RConsortium funded project, provides interactive tools to incorporate in geospatial workflows that require editing or selection of spatial data.

Edit

- `editMap`
- `editFeatures`
- Shiny edit module `editModUI`, `editMod`

#' @section Edit:

- `selectMap`
- `selectFeatures`
- Shiny edit module `selectModUI`, `selectMod`

Author(s)

Maintainer: Tim Appelhans <tim.appelhans@gmail.com>

Authors:

- Kenton Russell
- Lorenzo Busetto

Other contributors:

- Josh O'Brien [contributor]
- Jakob Gutschlhofer [contributor]

See Also

Useful links:

- https://github.com/r-spatial/mapedit
- Report bugs at https://github.com/r-spatial/mapedit/issues

`addToolbar` *Add a (possibly customized) toolbar to a leaflet map*

Description

Add a (possibly customized) toolbar to a leaflet map

Usage

```
addToolbar(leafmap, editorOptions, editor, targetLayerId)
```

Arguments

`leafmap`	leaflet map to use for Selection
`editorOptions`	A list of options to be passed on to either `leaflet.extras::addDrawToolbar` or `leafpm::addPmToolbar`.
`editor`	Character string giving editor to be used for the current map. Either `"leafpm"` or `"leaflet.extras"`.
`targetLayerId`	`string` name of the map layer group to use with edit

Value

The leaflet map supplied to `leafmap`, now with an added toolbar.

`drawFeatures` *Draw (simple) features on a map*

Description

Draw (simple) features on a map

Usage

```
drawFeatures(
  map = NULL,
  sf = TRUE,
  record = FALSE,
  viewer = shiny::paneViewer(),
  title = "Draw Features",
  editor = c("leaflet.extras", "leafpm"),
  editorOptions = list(),
  ...
)
```

Arguments

map	a background leaflet or mapview map to be used for editing. If NULL a blank mapview canvas will be provided.
sf	logical return simple features. The default is TRUE. If sf = FALSE, GeoJSON will be returned.
record	logical to record all edits for future playback.
viewer	function for the viewer. See Shiny viewer. NOTE: when using browserViewer(browser = getOption("browser")) to open the app in the default browser, the browser window will automatically close when closing the app (by pressing "done" or "cancel") in most browsers. Firefox is an exception. See Details for instructions on how to enable this behaviour in Firefox.
title	string to customize the title of the UI window.
editor	character either "leaflet.extras" or "leafpm"
editorOptions	list of options suitable for passing to either leaflet.extras::addDrawToolbar or leafpm::addPmToolbar.
...	additional arguments passed on to editMap.

Details

When setting viewer = browserViewer(browser = getOption("browser")) and the systems default browser is Firefox, the browser window will likely not automatically close when the app is closed (by pressing "done" or "cancel"). To enable automatic closing of tabs/windows in Firefox try the following:

- input "about:config " to your firefox address bar and hit enter
- make sure your "dom.allow_scripts_to_close_windows" is true

editFeatures *Interactively Edit Map Features*

Description

Interactively Edit Map Features

Usage

```
editFeatures(x, ...)

## S3 method for class 'sf'
editFeatures(
  x,
  map = NULL,
  mergeOrder = c("add", "edit", "delete"),
  record = FALSE,
  viewer = shiny::paneViewer(),
  crs = 4326,
  label = NULL,
  title = "Edit Map",
  editor = c("leaflet.extras", "leafpm"),
  editorOptions = list(),
  ...
)

## S3 method for class 'Spatial'
editFeatures(x, ...)
```

Arguments

x	features to edit
...	other arguments
map	a background `leaflet` or `mapview` map to be used for editing. If `NULL` a blank mapview canvas will be provided.
mergeOrder	`vector` or `character` arguments to specify the order of merge operations. By default, merges will proceed in the order of add, edit, delete.
record	`logical` to record all edits for future playback.
viewer	`function` for the viewer. See Shiny `viewer`. NOTE: when using `browserViewer(browser = getOption("browser"))` to open the app in the default browser, the browser window will automatically close when closing the app (by pressing "done" or "cancel") in most browsers. Firefox is an exception. See Details for instructions on how to enable this behaviour in Firefox.
crs	see `st_crs`.
label	`character` vector or `formula` for the content that will appear in label/tooltip.

title	string to customize the title of the UI window. The default is "Edit Map".
editor	character either "leaflet.extras" or "leafpm"
editorOptions	list of options suitable for passing to either leaflet.extras::addDrawToolbar or leafpm::addPmToolbar.

Details

When setting viewer = browserViewer(browser = getOption("browser")) and the systems default browser is Firefox, the browser window will likely not automatically close when the app is closed (by pressing "done" or "cancel"). To enable automatic closing of tabs/windows in Firefox try the following:

- input "about:config " to your firefox address bar and hit enter
- make sure your "dom.allow_scripts_to_close_windows" is true

Examples

```
## Not run:
library(mapedit)
library(mapview)

lf <- mapview()

# draw some polygons that we will select later
drawing <- lf %>%
  editMap()

# little easier now with sf
mapview(drawing$finished)

# especially easy with selectFeatures
selectFeatures(drawing$finished)


# use @bhaskarvk USA Albers with leaflet code
#  https://bhaskarvk.github.io/leaflet/examples/proj4Leaflet.html
#devtools::install_github("hrbrmstr/albersusa")
library(albersusa)
library(sf)
library(leaflet)
library(mapedit)

spdf <- usa_sf()
pal <- colorNumeric(
  palette = "Blues",
  domain = spdf$pop_2014
)

bounds <- c(-125, 24 ,-75, 45)

(lf <- leaflet(
```

```
  options=
    leafletOptions(
      worldCopyJump = FALSE,
      crs=leafletCRS(
        crsClass="L.Proj.CRS",
        code='EPSG:2163',
        proj4def=paste0(
          '+proj=laea +lat_0=45 +lon_0=-100 +x_0=0 +y_0=0 +a=6370997 ',
          '+b=6370997 +units=m +no_defs'
        ),
        resolutions = c(65536, 32768, 16384, 8192, 4096, 2048,1024, 512, 256, 128)
      )
    )
  ) %>%
  fitBounds(bounds[1], bounds[2], bounds[3], bounds[4]) %>%
  setMaxBounds(bounds[1], bounds[2], bounds[3], bounds[4]) %>%
  mapview::addFeatures(
    data=spdf, weight = 1, color = "#000000",
    # adding group necessary for identification
    layerId = ~iso_3166_2,
    fillColor=~pal(pop_2014),
    fillOpacity=0.7,
    label=~stringr::str_c(name,' ', format(pop_2014, big.mark=",")),
    labelOptions= labelOptions(direction = 'auto')
  )
)


# test out selectMap with albers example
selectMap(
  lf,
  styleFalse = list(weight = 1),
  styleTrue = list(weight = 4)
)

## End(Not run)
```

---

editMap *Interactively Edit a Map*

Description

Interactively Edit a Map

Usage

```
editMap(x, ...)

## S3 method for class 'leaflet'
editMap(
```

```
  x = NULL,
  targetLayerId = NULL,
  sf = TRUE,
  ns = "mapedit-edit",
  record = FALSE,
  viewer = shiny::paneViewer(),
  crs = 4326,
  title = "Edit Map",
  editor = c("leaflet.extras", "leafpm"),
  editorOptions = list(),
  ...
)

## S3 method for class 'mapview'
editMap(
  x = NULL,
  targetLayerId = NULL,
  sf = TRUE,
  ns = "mapedit-edit",
  record = FALSE,
  viewer = shiny::paneViewer(),
  crs = 4326,
  title = "Edit Map",
  editor = c("leaflet.extras", "leafpm"),
  editorOptions = list(),
  ...
)

## S3 method for class '`NULL`'
editMap(x, editor = c("leaflet.extras", "leafpm"), editorOptions = list(), ...)
```

Arguments

x	leaflet or mapview map to edit
...	other arguments for leafem::addFeatures() when using editMap.NULL or selectFeatures
targetLayerId	string name of the map layer group to use with edit
sf	logical return simple features. The default is TRUE. If sf = FALSE, GeoJSON will be returned.
ns	string name for the Shiny namespace to use. The ns is unlikely to require a change.
record	logical to record all edits for future playback.
viewer	function for the viewer. See Shiny viewer. NOTE: when using browserViewer(browser = getOption("browser")) to open the app in the default browser, the browser window will automatically close when closing the app (by pressing "done" or "cancel") in most browsers. Firefox is an exception. See Details for instructions on how to enable this behaviour in Firefox.

crs	see st_crs.
title	string to customize the title of the UI window. The default is "Edit Map".
editor	character either "leaflet.extras" or "leafpm"
editorOptions	list of options suitable for passing to either leaflet.extras::addDrawToolbar or leafpm::addPmToolbar.

Details

When setting viewer = browserViewer(browser = getOption("browser")) and the systems default browser is Firefox, the browser window will likely not automatically close when the app is closed (by pressing "done" or "cancel"). To enable automatic closing of tabs/windows in Firefox try the following:

- input "about:config " to your firefox address bar and hit enter
- make sure your "dom.allow_scripts_to_close_windows" is true

Value

sf simple features or GeoJSON

Examples

```
## Not run:
library(leaflet)
library(mapedit)
editMap(leaflet() %>% addTiles())

## End(Not run)
## Not run:
  # demonstrate Leaflet.Draw on a layer
  library(sf)
  library(mapview)
  library(leaflet.extras)
  library(mapedit)

  # ?sf::sf
  pol = st_sfc(
    st_polygon(list(cbind(c(0,3,3,0,0),c(0,0,3,3,0)))),
    crs = 4326
  )
  mapview(pol) %>%
    editMap(targetLayerId = "pol")

  mapview(franconia[1:2,]) %>%
    editMap(targetLayerId = "franconia[1:2, ]")

## End(Not run)
```

editMod *Shiny Module Server for Geo Create, Edit, Delete*

Description

Shiny Module Server for Geo Create, Edit, Delete

Usage

```
editMod(
  input,
  output,
  session,
  leafmap,
  targetLayerId = NULL,
  sf = TRUE,
  record = FALSE,
  crs = 4326,
  editor = c("leaflet.extras", "leafpm"),
  editorOptions = list()
)
```

Arguments

input	Shiny server function input
output	Shiny server function output
session	Shiny server function session
leafmap	leaflet map to use for Selection
targetLayerId	character identifier of layer to edit, delete
sf	logical to return simple features. sf=FALSE will return GeoJSON.
record	logical to record all edits for future playback.
crs	see st_crs.
editor	character either "leaflet.extras" or "leafpm"
editorOptions	list of options suitable for passing to either leaflet.extras::addDrawToolbar or leafpm::addPmToolbar.

Value

server function for Shiny module

editModUI *Shiny Module UI for Geo Create, Edit, Delete*

Description

Shiny Module UI for Geo Create, Edit, Delete

Usage

```
editModUI(id, ...)
```

Arguments

id	character id for the the Shiny namespace
...	other arguments to leafletOutput()

Value

ui for Shiny module

processOpts *Prepare arguments for addDrawToolbar or addPmToolbar*

Description

Prepare arguments for addDrawToolbar or addPmToolbar

Usage

```
processOpts(fun, args)
```

Arguments

fun	Function used by editor package (leafpm or leaflet.extras) to set defaults
args	Either a (possibly nested) list of named options of the form suitable for passage to fun or (if the chosen editor is "leaflet.extras") FALSE.

Value

An object suitable for passing in as the supplied argument to either leaflet.extras::addDrawToolbar or leafpm::addPmToolbar.

`selectFeatures` *Interactively Select Map Features*

Description

Interactively Select Map Features

Usage

```
selectFeatures(x, ...)

## S3 method for class 'sf'
selectFeatures(
  x = NULL,
  mode = c("click", "draw"),
  op = sf::st_intersects,
  map = NULL,
  index = FALSE,
  viewer = shiny::paneViewer(),
  label = NULL,
  title = "Select features",
  ...
)

## S3 method for class 'Spatial'
selectFeatures(x, ...)
```

Arguments

`x`	features to select
`...`	other arguments
`mode`	one of "click" or "draw".
`op`	the geometric binary predicate to use for the selection. Can be any of `geos_binary_pred`. In the spatial operation the drawn features will be evaluated as x and the supplied feature as y. Ignored if `mode = "click"`.
`map`	a background `leaflet` or `mapview` map to be used for editing. If `NULL` a blank mapview canvas will be provided.
`index`	`logical` with `index=TRUE` indicating return the index of selected features rather than the actual selected features
`viewer`	`function` for the viewer. See Shiny `viewer`. NOTE: when using `browserViewer(browser = getOption("browser"))` to open the app in the default browser, the browser window will automatically close when closing the app (by pressing "done" or "cancel") in most browsers. Firefox is an exception. See Details for instructions on how to enable this behaviour in Firefox.
`label`	`character` vector or `formula` for the content that will appear in label/tooltip.
`title`	`string` to customize the title of the UI window. The default is "Select features".

Details

When setting `viewer = browserViewer(browser = getOption("browser"))` and the systems default browser is Firefox, the browser window will likely not automatically close when the app is closed (by pressing "done" or "cancel"). To enable automatic closing of tabs/windows in Firefox try the following:

- input "about:config " to your firefox address bar and hit enter
- make sure your "dom.allow_scripts_to_close_windows" is true

Examples

```
## Not run:
library(mapedit)
library(mapview)

lf <- mapview()

# draw some polygons that we will select later
drawing <- lf %>%
  editMap()

# little easier now with sf
mapview(drawing$finished)

# especially easy with selectFeatures
selectFeatures(drawing$finished)


# use @bhaskarvk USA Albers with leaflet code
#  https://bhaskarvk.github.io/leaflet/examples/proj4Leaflet.html
#devtools::install_github("hrbrmstr/albersusa")
library(albersusa)
library(sf)
library(leaflet)
library(mapedit)

spdf <- usa_sf()
pal <- colorNumeric(
  palette = "Blues",
  domain = spdf$pop_2014
)

bounds <- c(-125, 24 ,-75, 45)

(lf <- leaflet(
  options=
    leafletOptions(
      worldCopyJump = FALSE,
      crs=leafletCRS(
        crsClass="L.Proj.CRS",
        code='EPSG:2163',
        proj4def=paste0(
```

```
          '+proj=laea +lat_0=45 +lon_0=-100 +x_0=0 +y_0=0 +a=6370997 ',
          '+b=6370997 +units=m +no_defs'
        ),
        resolutions = c(65536, 32768, 16384, 8192, 4096, 2048,1024, 512, 256, 128)
      )
    )
  ) %>%
  fitBounds(bounds[1], bounds[2], bounds[3], bounds[4]) %>%
  setMaxBounds(bounds[1], bounds[2], bounds[3], bounds[4]) %>%
  mapview::addFeatures(
    data=spdf, weight = 1, color = "#000000",
    # adding group necessary for identification
    layerId = ~iso_3166_2,
    fillColor=~pal(pop_2014),
    fillOpacity=0.7,
    label=~stringr::str_c(name,' ', format(pop_2014, big.mark=",")),
    labelOptions= labelOptions(direction = 'auto')
  )
)


# test out selectMap with albers example
selectMap(
  lf,
  styleFalse = list(weight = 1),
  styleTrue = list(weight = 4)
)

## End(Not run)
```

---

selectMap *Interactively Select Map Features*

Description

Interactively Select Map Features

Usage

```
selectMap(x, ...)

## S3 method for class 'leaflet'
selectMap(
  x = NULL,
  styleFalse = list(fillOpacity = 0.2, weight = 1, opacity = 0.4),
  styleTrue = list(fillOpacity = 0.7, weight = 3, opacity = 0.7),
  ns = "mapedit-select",
  viewer = shiny::paneViewer(),
  title = "Select features",
  ...
)
```

Arguments

x	leaflet or mapview map to use for selection
...	other arguments
styleFalse, styleTrue	names list of CSS styles used for selected (styleTrue) and deselected (styleFalse)
ns	string name for the Shiny namespace to use. The ns is unlikely to require a change.
viewer	function for the viewer. See Shiny viewer. NOTE: when using browserViewer(browser = getOption("browser")) to open the app in the default browser, the browser window will automatically close when closing the app (by pressing "done" or "cancel") in most browsers. Firefox is an exception. See Details for instructions on how to enable this behaviour in Firefox.
title	string to customize the title of the UI window. The default is "Select features".

Details

When setting viewer = browserViewer(browser = getOption("browser")) and the systems default browser is Firefox, the browser window will likely not automatically close when the app is closed (by pressing "done" or "cancel"). To enable automatic closing of tabs/windows in Firefox try the following:

- input "about:config " to your firefox address bar and hit enter
- make sure your "dom.allow_scripts_to_close_windows" is true

Examples

```
## Not run:
library(mapedit)
library(mapview)

lf <- mapview()

# draw some polygons that we will select later
drawing <- lf %>%
  editMap()

# little easier now with sf
mapview(drawing$finished)

# especially easy with selectFeatures
selectFeatures(drawing$finished)


# use @bhaskarvk USA Albers with leaflet code
#  https://bhaskarvk.github.io/leaflet/examples/proj4Leaflet.html
#devtools::install_github("hrbrmstr/albersusa")
library(albersusa)
library(sf)
library(leaflet)
```

```
library(mapedit)

spdf <- usa_sf()
pal <- colorNumeric(
  palette = "Blues",
  domain = spdf$pop_2014
)

bounds <- c(-125, 24 ,-75, 45)

(lf <- leaflet(
  options=
    leafletOptions(
      worldCopyJump = FALSE,
      crs=leafletCRS(
        crsClass="L.Proj.CRS",
        code='EPSG:2163',
        proj4def=paste0(
          '+proj=laea +lat_0=45 +lon_0=-100 +x_0=0 +y_0=0 +a=6370997 ',
          '+b=6370997 +units=m +no_defs'
        ),
        resolutions = c(65536, 32768, 16384, 8192, 4096, 2048,1024, 512, 256, 128)
      )
    )
  ) %>%
  fitBounds(bounds[1], bounds[2], bounds[3], bounds[4]) %>%
  setMaxBounds(bounds[1], bounds[2], bounds[3], bounds[4]) %>%
  mapview::addFeatures(
    data=spdf, weight = 1, color = "#000000",
    # adding group necessary for identification
    layerId = ~iso_3166_2,
    fillColor=~pal(pop_2014),
    fillOpacity=0.7,
    label=~stringr::str_c(name,' ', format(pop_2014, big.mark=",")),
    labelOptions= labelOptions(direction = 'auto')
  )
)


# test out selectMap with albers example
selectMap(
  lf,
  styleFalse = list(weight = 1),
  styleTrue = list(weight = 4)
)

## End(Not run)
```

`selectMod` *Shiny Module Server for Geo Selection*

Description

Shiny Module Server for Geo Selection

Usage

```
selectMod(
  input,
  output,
  session,
  leafmap,
  styleFalse = list(fillOpacity = 0.2, weight = 1, opacity = 0.4),
  styleTrue = list(fillOpacity = 0.7, weight = 3, opacity = 0.7)
)
```

Arguments

input	Shiny server function input
output	Shiny server function output
session	Shiny server function session
leafmap	leaflet map to use for Selection
styleFalse	named list of valid CSS for non-selected features
styleTrue	named list of valid CSS for selected features

Value

server function for Shiny module

---

selectModUI *Shiny Module UI for Geo Selection*

Description

Shiny Module UI for Geo Selection

Usage

```
selectModUI(id, ...)
```

Arguments

id	character id for the the Shiny namespace
...	other arguments to leafletOutput()

Value

ui for Shiny module

Index