

# Package ‘mapedit’

May 8, 2026

**Title** Interactive Editing of Spatial Data in R

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

**Version** 0.8.0

**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** assertthat, dplyr, DT, htmltools (>= 0.3), htmlwidgets, jsonlite, leafem, leaflet (>= 2.0.1), leafpm, leafpop, mapview, methods, miniUI, raster, rstudioapi, scales, sf (>= 0.5-2), shiny, tmaptools, shinyWidgets (>= 0.4.3), magrittr

**Suggests** crayon, sp

**Enhances** geojsonio

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**NeedsCompilation** no

**Author** Tim Appelhans [aut, cre],  
Kenton Russell [aut],  
Lorenzo Busetto [aut],  
Josh O'Brien [ctb],  
Jakob Gutschlhofer [ctb],  
Matt Johnson [ctb],  
Eli Pousson [ctb] (ORCID: <<https://orcid.org/0000-0001-8280-1706>>)

**Maintainer** Tim Appelhans <[tim.appelhans@gmail.com](mailto:tim.appelhans@gmail.com)>

**Repository** CRAN

**Date/Publication** 2026-02-15 12:20:02 UTC

## Contents

mapedit-package	2
addToolbar	3
createFeatures	4
drawFeatures	4
editAttributes	5
editFeatures	7
editMap	9
editMod	11
editModUI	12
processOpts	12
selectFeatures	13
selectMap	14
selectMod	16
selectModUI	16
<b>Index</b>	<b>18</b>

---

mapedit-package	<i>mapedit: interactive editing and selection for geospatial data</i>
-----------------	-----------------------------------------------------------------------

---

## 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]
- Matt Johnson [contributor]
- Eli Pousson ([ORCID](#)) [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(leafletmap, editorOptions, editor = "leafpm", targetLayerId)
```

### Arguments

leafletmap	leaflet map to use for Selection
editorOptions	A list of options to be passed on to leafpm::addPmToolbar.
editor	Character string giving editor to be used for the current map. Defaults to "leafpm" which is the only supported option at the moment.
targetLayerId	string name of the map layer group to use with edit

### Value

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

---

createFeatures	<i>mapedit create features Addin</i>
----------------	--------------------------------------

---

### Description

Create and save spatial objects within the Rstudio IDE. Objects can then be saved to file types such as .geojson or .shp. Objects are also output to the console and can be assigned to a variable using `.Last.value`. If you wish to pass the output directly to a variable simply call the `addin` function, ie. `new_sf <- createFeatures()`.

An existing `sf` data frame can also be passed either indirectly by selecting text in RStudio with the name of the object, or directly by passing the existing `sf` object to `new_sf <- createFeatures(existing_sf)`. When passing an existing `sf` object you can only add and edit additional features, the existing features cannot be changed.

### Usage

```
createFeatures(SF_OBJECT = NULL)
```

### Arguments

`SF_OBJECT`      `sf` Simple feature collection

### Value

`sf` object and/or saved to file

---

drawFeatures	<i>Draw (simple) features on a map</i>
--------------	----------------------------------------

---

### Description

Draw (simple) features on a map

### Usage

```
drawFeatures(
  map = NULL,
  sf = TRUE,
  record = FALSE,
  viewer = shiny::paneViewer(),
  title = "Draw Features",
  editor = "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 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 currently, only "leafpm" is supported.
editorOptions	list of options suitable for passing to leafpm::addPmToolbar.
...	additional arguments passed on to <a href="#">editMap</a> .

**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

---

editAttributes

*Edit Feature Attributes*


---

**Description**

Launches a shiny application where you can add and edit spatial geometry and attributes. Geometry is created or edited within the interactive map, while feature attributes can be added to and edited within the editable table.

Starting with a data.frame or an sf data.frame, a list of sf data.frames or nothing at all. You can add columns, and rows and geometry for each row. Clicking on a row with geometry you can zoom across the map between features.

When you are done, your edits are saved to an sf data.frame for use in R or to be saved to anyformat you wish via [st\\_write](#).

The application can dynamically handle: character, numeric, integer, factor and date fields.

When the input data set is an sf data.frame the map automatically zooms to the extent of the sf object.

When the input has no spatial data, you must tell the function where to zoom. The function uses [geocode\\_OSM](#) to identify the coordinates of your area of interest.

**Usage**

```
editAttributes(
  dat,
  zoomto = NULL,
  col_add = TRUE,
  reset = TRUE,
  provider = "Esri.WorldImagery",
  testing = FALSE
)
```

**Arguments**

dat	input data source, can be a <code>data.frame</code> or an <code>sf data.frame</code> , or it can be left empty. When nothing is passed to <code>dat</code> a basic <code>data.frame</code> is generated with <code>id</code> and <code>comment</code> fields.
zoomto	character area of interest. The area is defined using <a href="#">geocode_OSM</a> , which uses <a href="#">OSM Nominatim</a> . The area can be as ambiguous as a country, or as specific as a street address. You can test the area of interest using the application or the example code below.
col_add	boolean option to enable add columns form. Set to false if you don't want to allow a user to modify the data structure.
reset	boolean option to reset attribute input. Set to false if you don't want the attribute input to reset to NA after each added row. Use this option when features share common attributes
provider	A character string indicating the provider tile of choice, e.g. 'Esri.WorldImagery' (default)
testing	Only relevant for internal testing using <code>shinytest</code> .

**Value**

sf data.frame

**Note**

Editing of feature geometries does not work for multi-geometry inputs. For this use case it is advisable to split the data set by geometry type and edit separately

**Examples**

```
## Not run:

# with no input
data_sf <- editAttributes(zoomto = 'germany')

# a data.frame input
dat <- data.frame(name = c('SiteA', 'SiteB'),
                  type = factor(
                    c('park', 'zoo')
```

```
      , levels = c('park', 'factory', 'zoo', 'warehouse')
    ),
    size = c(35, 45))

data_sf <- editAttributes(dat, zoomto = 'berlin')

# an sf data.frame input
data_sf <- editAttributes(data_sf)

# test zoomto area of interest
zoomto_area <- tmaptools::geocode_OSM('paris')
mapview(st_as_sf(zoomto_area$bbox))

## End(Not run)
```

---

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 = "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 <a href="#">viewer</a> . NOTE: when using <code>browserViewer(browser = getOption("browser"))</code> 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 <a href="#">st_crs</a> .
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 currently, only "leafpm" is supported.
editorOptions	list of options suitable for passing to <code>leafpm::addPmToolbar</code> .

### 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)

## End(Not run)
```

---

editMap	<i>Interactively Edit a Map</i>
---------	---------------------------------

---

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

## Arguments

x leaflet or mapview map to edit

...	other arguments for <code>leafem::addFeatures()</code> when using <code>editMap.NULL</code> or <code>selectFeatures</code>
<code>targetLayerId</code>	string name of the map layer group to use with <code>edit</code>
<code>sf</code>	logical return simple features. The default is <code>TRUE</code> . If <code>sf = FALSE</code> , GeoJSON will be returned.
<code>ns</code>	string name for the Shiny namespace to use. The <code>ns</code> is unlikely to require a change.
<code>record</code>	logical to record all edits for future playback.
<code>viewer</code>	function for the viewer. See Shiny <a href="#">viewer</a> . NOTE: when using <code>browserViewer(browser = getOption("browser"))</code> 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 <a href="#">Details</a> for instructions on how to enable this behaviour in Firefox.
<code>crs</code>	see <a href="#">st_crs</a> .
<code>title</code>	string to customize the title of the UI window. The default is "Edit Map".
<code>editor</code>	character currently, only "leafpm" is supported.
<code>editorOptions</code>	list of options suitable for passing to <code>leafpm::addPmToolbar</code> .

### 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(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 = "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 <a href="#">st_crs</a> .
editor	character currently, only "leafpm" is supported.
editorOptions	list of options suitable for passing to leafpm: :addPmToolBar.

**Value**

server function for Shiny module

---

editModUI	<i>Shiny Module UI for Geo Create, Edit, Delete</i>
-----------	-----------------------------------------------------

---

**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	<i>Prepare arguments for addPmToolbar</i>
-------------	-------------------------------------------

---

**Description**

Prepare arguments for addPmToolbar

**Usage**

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

**Arguments**

fun	Function used by editor package (leafpm) to set defaults
args	Either a (possibly nested) list of named options of the form suitable for passage to fun.

**Value**

An object suitable for passing in as the supplied argument to leafpm::addPmToolbar.

---

selectFeatures	<i>Interactively Select Map Features</i>
----------------	------------------------------------------

---

## 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 <code>sf::geos_binary_pred</code> . 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 <a href="#">viewer</a> . NOTE: when using <code>browserViewer(browser = getOption("browser"))</code> 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)

## 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 <a href="#">viewer</a> . 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)

## 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**

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

<code>id</code>	character id for the the Shiny namespace
<code>...</code>	other arguments to <code>leafletOutput()</code>

**Value**

ui for Shiny module

# Index

[addToolbar](#), [3](#)

[createFeatures](#), [4](#)

[drawFeatures](#), [4](#)

[editAttributes](#), [5](#)

[editFeatures](#), [2](#), [7](#)

[editMap](#), [2](#), [5](#), [9](#)

[editMod](#), [2](#), [11](#)

[editModUI](#), [2](#), [12](#)

[geocode\\_OSM](#), [5](#), [6](#)

[mapedit \(mapedit-package\)](#), [2](#)

[mapedit-package](#), [2](#)

[processOpts](#), [12](#)

[selectFeatures](#), [2](#), [13](#)

[selectMap](#), [2](#), [14](#)

[selectMod](#), [2](#), [16](#)

[selectModUI](#), [2](#), [16](#)

[st\\_crs](#), [8](#), [10](#), [11](#)

[st\\_write](#), [5](#)

[viewer](#), [8](#), [10](#), [13](#), [15](#)