

# Package ‘jpmesh’

January 10, 2022

**Type** Package

**Title** Utilities for Japanese Mesh Code

**Version** 2.1.0

**Maintainer** Shinya Uryu <suika1127@gmail.com>

**Description** Helpful functions for using mesh code (80km to 100m) data in Japan. Visualize mesh code using 'ggplot2' and 'leaflet', etc.

**License** MIT + file LICENSE

**URL** <https://uribo.github.io/jpmesh/>

**BugReports** <https://github.com/uribo/jpmesh/issues/>

**Depends** R (>= 3.1)

**Imports** leaflet (>= 1.1.0), memoise (>= 1.1.0), miniUI (>= 0.1.1), purrr (>= 0.2.4), rlang (>= 0.1.4), sf (>= 0.5-5), shiny (>= 1.0.5), tibble (>= 3.0.0), units (>= 0.5-1), magrittr (>= 1.5), vctrs (>= 0.3.4)

**Suggests** knitr (>= 1.20), lintr (>= 2.0.1), lwgeom (>= 0.1-4), testthat (>= 2.1.0), rmarkdown (>= 1.10), vdiffr (>= 0.3.1)

**VignetteBuilder** knitr

**LazyData** true

**Encoding** UTF-8

**RoxygenNote** 7.1.2

**NeedsCompilation** no

**Author** Shinya Uryu [aut, cre] (<<https://orcid.org/0000-0002-0493-6186>>)

**Repository** CRAN

**Date/Publication** 2022-01-10 03:32:41 UTC

## R topics documented:

administration_mesh . . . . .	2
coarse_gather . . . . .	3

coords_to_mesh . . . . .	3
cut_off . . . . .	4
eval_jp_boundary . . . . .	5
export_mesh . . . . .	5
fine_separate . . . . .	6
is_mesh . . . . .	7
jpnrect . . . . .	7
meshcode_set . . . . .	8
meshcode_sf . . . . .	8
meshcode_vector . . . . .	9
mesh_convert . . . . .	10
mesh_size . . . . .	11
mesh_to_coords . . . . .	11
mesh_viewer . . . . .	12
neighbor_mesh . . . . .	12
rmesh . . . . .	13
sf_jpmesh . . . . .	13

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

---

administration_mesh	<i>Extract administration mesh code</i>
---------------------	---

---

## Description

Extract administration mesh code

## Usage

```
administration_mesh(code, to_mesh_size)
```

## Arguments

code	administration code
to_mesh_size	target mesh size. That is, 1 for 1km, and 0.5 for 500m. From 80km to 0.100km.

## Examples

```
## Not run:
administration_mesh(code = "35201", to_mesh_size = 1)
administration_mesh(code = "08220", to_mesh_size = 80)
administration_mesh(code = c("08220", "08221"), to_mesh_size = 10)
administration_mesh(code = "35", to_mesh_size = 80)
administration_mesh(code = c("33", "34"), to_mesh_size = 80)

## End(Not run)
```

---

coarse_gather	<i>Gather more coarse mesh</i>
---------------	--------------------------------

---

**Description**

Return coarse gather mesh codes

**Usage**

```
coarse_gather(meshcode, distinct = FALSE)
```

**Arguments**

meshcode	character. mesh code
distinct	return unique meshcodes

**Value**

[meshcode](#)

**Examples**

```
m <- c("493214294", "493214392", "493215203", "493215301")
coarse_gather(m)
coarse_gather(coarse_gather(m))
coarse_gather(coarse_gather(m), distinct = TRUE)
```

---

coords_to_mesh	<i>Convert from coordinate to mesh code</i>
----------------	---

---

**Description**

From coordinate to mesh codes.

**Usage**

```
coords_to_mesh(longitude, latitude, to_mesh_size = 1, geometry = NULL, ...)
```

**Arguments**

longitude	longitude that approximately to .120.0 to 154.0 (double)
latitude	latitude that approximately to 20.0 to 46.0 (double)
to_mesh_size	target mesh size. That is, 1 for 1km, and 0.5 for 500m. From 80km to 0.100km.
geometry	XY sfg object
...	other parameters

**Value**

mesh code (default 3rd meshcode aka 1km mesh)

**References**

Akio Takenaka: [http://takenaka-akio.org/etc/j\\_map/index.html](http://takenaka-akio.org/etc/j_map/index.html) # nolint

**See Also**

[mesh\\_to\\_coords\(\)](#) for convert from meshcode to coordinates

**Examples**

```

coords_to_mesh(141.3468, 43.06462, to_mesh_size = 1)
coords_to_mesh(139.6917, 35.68949, to_mesh_size = 0.250)
coords_to_mesh(139.71475, 35.70078)
coords_to_mesh(139.71475, 35.70078, to_mesh_size = 0.1)
coords_to_mesh(c(141.3468, 139.71475),
               c(43.06462, 35.70078),
               mesh_size = c(1, 10))
# Using sf (point as sfg object)
library(sf)
coords_to_mesh(geometry = st_point(c(139.71475, 35.70078)))
coords_to_mesh(geometry = st_point(c(130.4412895, 30.2984335)))

```

---

cut\_off

*Cutoff mesh of outside the area*

---

**Description**

Cutoff mesh of outside the area

**Usage**

```
cut_off(meshcode)
```

**Arguments**

meshcode          character. mesh code

---

eval_jp_boundary	<i>Check include mesh areas</i>
------------------	---------------------------------

---

**Description**

It roughly judges whether the given coordinates are within the mesh area.

**Usage**

```
eval_jp_boundary(longitude = NULL, latitude = NULL, ...)
```

**Arguments**

longitude	longitude that approximately to .120.0 to 154.0 (double)
latitude	latitude that approximately to 20.0 to 46.0 (double)
...	other parameters

**Examples**

```
eval_jp_boundary(139.71471056, 35.70128943)
```

---

export_mesh	<i>Export meshcode to geometry</i>
-------------	------------------------------------

---

**Description**

Convert and export meshcode area to sfc\_POLYGON and sf.

**Usage**

```
export_mesh(meshcode)

export_meshes(meshcode, .keep_class = FALSE)
```

**Arguments**

meshcode	character. mesh code
.keep_class	Do you want to assign a class to the meshcode column in data.frame? If FALSE, it will be treated as a character type.

**Value**

[sfc](#) object  
[sf](#) object

## Examples

```
export_mesh("6441427712")
export_meshes("4128")
find_neighbor_mesh("37250395") %>%
  export_meshes()
```

---

fine_separate	<i>Separate more fine mesh order</i>
---------------	--------------------------------------

---

## Description

Return contains fine mesh codes

## Usage

```
fine_separate(meshcode = NULL, .type = "standard", ...)
```

## Arguments

meshcode	character. mesh code
.type	Specify the subdivision if you want to get a 100m mesh.
...	other parameters for <a href="#">paste</a>

## Value

[meshcode](#)

## Examples

```
fine_separate("5235")
fine_separate("523504")
fine_separate("52350432")
fine_separate("523504321")
fine_separate("5235043211")
# to 100m mesh code
fine_separate("64414315", .type = "subdivision")
```

---

is_mesh	<i>Predict meshcode format and positions</i>
---------	--

---

**Description**

Predict meshcode format and positions for utility and certain.

**Usage**

```
is_meshcode(meshcode)
```

```
is_corner(meshcode)
```

**Arguments**

meshcode	character. mesh code
----------	----------------------

---

jpnrect	<i>Simple displaed as rectangel for Japan (fortified)</i>
---------	---

---

**Description**

Rectangle Japanese prefectures positions.

**Usage**

```
jpnrect
```

**Format**

A data frame with 235 rows 11 variables:

- long
- lat
- order
- hole
- piece
- id
- group
- mesh\_code
- latitude
- longitude
- abb\_name

**Examples**

```
## Not run:
plot(jpnrect["abb_name"])

## End(Not run)
```

---

meshcode_set	<i>Export meshcode vectors ranges 80km to 1km.</i>
--------------	--

---

**Description**

Unique 176 meshcodes. The output code may contain values not found in the actual mesh code.

**Usage**

```
meshcode_set(mesh_size = c(80, 10, 1), .raw = TRUE)
```

**Arguments**

mesh_size	Export mesh size from 80km to 1km.
.raw	return as character.

**Value**

character or [meshcode](#)

**Examples**

```
meshcode_set(mesh_size = 80)
meshcode_set(mesh_size = 80, .raw = FALSE)
```

---

meshcode_sf	<i>Conversion to sf objects containing meshcode</i>
-------------	---

---

**Description**

Convert and export meshcode area to sf.

**Usage**

```
meshcode_sf(data, mesh_var, .type, .keep_class = FALSE)
```



**Arguments**

data	data.frame
mesh_var	unquoted expressions for meshcode variable.
.type	Specify the subdivision if you want to get a 100m mesh.
.keep_class	Do you want to assign a class to the meshcode column in data.frame? If FALSE, it will be treated as a character type.

**Value**

sf object

**Examples**

```
d <- data.frame(id = seq.int(4),
               meshcode = rmesh(4),
               stringsAsFactors = FALSE)
meshcode_sf(d, meshcode)
```

---

meshcode_vector	<i>Vector of meshcode</i>
-----------------	---------------------------

---

**Description**

Vector of meshcode

**Usage**

```
meshcode_vector(x = character(), size = double(), .type = "standard")
```

```
meshcode(x, .type = "standard")
```

```
as_meshcode(x, ...)
```

```
## S3 method for class 'meshcode'
format(x, ...)
```

```
## S3 method for class 'subdiv_meshcode'
format(x, ...)
```

**Arguments**

x	input meshcode value
size	input meshcode size. Default set to NULL. The decision is automatically made based on the meshsize.
.type	Specify the subdivision if you want to get a 100m mesh.
...	path to another function

**Value**

[meshcode](#)

**Examples**

```
meshcode("6441")
meshcode(c("6441", "6442"))
meshcode(c("6441", "644143"))
meshcode("6441431552", .type = "subdivision")
```

---

mesh\_convert

*Mesh unit converter*

---

**Description**

Return different meshcode values included in the mesh.

**Usage**

```
mesh_convert(meshcode = NULL, to_mesh_size = NULL)
```

**Arguments**

meshcode            character. mesh code  
to\_mesh\_size        target mesh size. That is, 1 for 1km, and 0.5 for 500m. From 80km to 0.100km.

**Details**

If NULL for to\_mesh\_size, the meshcode of one small scale will be returned. If it is the same as the original size, the meshcode of the input will be return.

**Value**

[meshcode](#)

**Examples**

```
mesh_convert(meshcode = "52350432", to_mesh_size = 80)
mesh_convert("52350432", 10)
# Scale down
mesh_convert("52350432", 0.500)
mesh_convert("52350432", 0.250)
mesh_convert(meshcode = "52350432", 0.125)
mesh_convert("523504323", 0.250)
mesh_convert("5235043213", 0.125)
mesh_convert(64414315, 0.1)
# Not changes
mesh_convert("52350432", 1)
mesh_convert("52350432131", 0.125)
```

---

mesh_size	<i>Identifier to mesh size</i>
-----------	--------------------------------

---

**Description**

Returns a unit object of mesh size for the given number.

**Usage**

```
mesh_size(meshcode, .type = "standard")
```

**Arguments**

meshcode	character. mesh code
.type	Specify the subdivision if you want to get a 100m mesh.

**Examples**

```
mesh_size("6740")
```

---

mesh_to_coords	<i>Get from mesh code to latitude and longitude</i>
----------------	---

---

**Description**

mesh centroid

**Usage**

```
mesh_to_coords(meshcode, ...)
```

**Arguments**

meshcode	character. mesh code
...	other parameters

**References**

Akio Takenaka: [http://takenaka-akio.org/etc/j\\_map/index.html](http://takenaka-akio.org/etc/j_map/index.html) # nolint

**See Also**

[coords\\_to\\_mesh\(\)](#) for convert from coordinates to meshcode.

**Examples**

```
mesh_to_coords("64414277")  
mesh_to_coords(c("64414277", "64414278"))
```

---

mesh_viewer	<i>interactive meshcode check</i>
-------------	-----------------------------------

---

**Description**

Shiny gadgets for jpmesh.

**Usage**

```
mesh_viewer(...)
```

**Arguments**

```
...          other parameters
```

**Examples**

```
## Not run:
mesh_viewer()

## End(Not run)
```

---

neighbor_mesh	<i>Find out neighborhood meshes collection</i>
---------------	--

---

**Description**

input should use meshcode under the 1km mesh size.

**Usage**

```
neighbor_mesh(meshcode, contains = TRUE)

find_neighbor_mesh(meshcode = NULL, contains = TRUE)
```

**Arguments**

```
meshcode      character. mesh code
contains      logical. contains input meshcode (default TRUE)
```

**Value**

[meshcode](#)

**Examples**

```
neighbor_mesh(53394501)
neighbor_mesh(533945011)
neighbor_mesh(533945011, contains = FALSE)
```

---

rmesh	<i>Generate random sample meshcode</i>
-------	--

---

**Description**

Generate random sample meshcode

**Usage**

```
rmesh(n, mesh_size = 1)
```

**Arguments**

n	Number of samples
mesh_size	Export mesh size from 80km to 1km.

**Value**

[meshcode](#)

**Examples**

```
rmesh(3, mesh_size = 1)
```

---

sf_jpmesh	<i>1:200,000 Scale Maps Name with Meshcode of Japan.</i>
-----------	--

---

**Description**

Information for the 1:200,000 Scale Maps.

**Usage**

```
sf_jpmesh
```

**Format**

A data frame with 175 rows 9 variables:

- meshcode: 80km meshcode
- name: names for map
- name\_roman: names for map (roman)
- lng\_center: centroid coordinates of mesh
- lat\_center: centroid coordinates of mesh
- lng\_error: mesh area
- lat\_error: mesh area
- type: evaluate value to mesh

**Examples**

```
## Not run:  
plot(sf_jpmesh["name_roman"])  
  
## End(Not run)
```

# Index

- \* **datasets**
  - jpnrect, 7
  - sf\_jpmesh, 13
- administration\_mesh, 2
- as\_meshcode (meshcode\_vector), 9
  
- coarse\_gather, 3
- coords\_to\_mesh, 3
- coords\_to\_mesh(), 11
- cut\_off, 4
  
- eval\_jp\_boundary, 5
- export\_mesh, 5
- export\_meshes (export\_mesh), 5
  
- find\_neighbor\_mesh (neighbor\_mesh), 12
- fine\_separate, 6
- format.meshcode (meshcode\_vector), 9
- format.subdiv\_meshcode (meshcode\_vector), 9
  
- is\_corner (is\_mesh), 7
- is\_mesh, 7
- is\_meshcode (is\_mesh), 7
  
- jpnrect, 7
  
- mesh\_convert, 10
- mesh\_size, 11
- mesh\_to\_coords, 11
- mesh\_to\_coords(), 4
- mesh\_viewer, 12
- meshcode, 3, 6, 8, 10, 12, 13
- meshcode (meshcode\_vector), 9
- meshcode\_set, 8
- meshcode\_sf, 8
- meshcode\_vector, 9
  
- neighbor\_mesh, 12
  
- paste, 6
  
- rmesh, 13
  
- sf, 5, 9
- sf\_jpmesh, 13
- sfc, 5