

# Package ‘jpmesh’

May 9, 2019

**Type** Package

**Title** Utilities for Japanese Mesh Code

**Version** 1.1.3

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

**Description** Helpful functions for using mesh code (80km to 125m) data in Japan. Visualize mesh code using 'ggplot2' and 'leaflet', etc.

**License** MIT + file LICENSE

**URL** <https://github.com/uribo/jpmesh#readme>

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

**Depends** R (>= 3.1)

**Imports** leaflet (>= 1.1.0), miniUI (>= 0.1.1), purrr (>= 0.2.4), rlang (>= 0.1.4), sf (>= 0.5-5), shiny (>= 1.0.5), tibble (>= 1.3.4), units (>= 0.5-1), magrittr (>= 1.5)

**Suggests** knitr (>= 1.20), lwgeom (>= 0.1-4), testthat (>= 1.0.2), rmarkdown (>= 1.10)

**VignetteBuilder** knitr

**LazyData** true

**Encoding** UTF-8

**RoxygenNote** 6.1.1.9000

**NeedsCompilation** no

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

**Repository** CRAN

**Date/Publication** 2019-05-09 07:50:03 UTC

## R topics documented:

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

cut_off . . . . .	4
eval_jp_boundary . . . . .	5
export_mesh . . . . .	5
export_meshes . . . . .	6
fine_separate . . . . .	6
is_mesh . . . . .	7
jpnrct . . . . .	7
meshcode_set . . . . .	8
mesh_to_coords . . . . .	8
mesh_viewer . . . . .	9
neighbor_mesh . . . . .	10
rmesh . . . . .	10
sf_jpmesh . . . . .	11

<b>Index</b>	<b>12</b>
--------------	-----------

---

administration\_mesh    *Extract administration mesh*

---

## Description

Extract administration mesh

## Usage

```
administration_mesh(code, type = c("city", "prefecture"))
```

## Arguments

code	administration code
type	administration type. Should be one of "prefecture" or "city".

## Examples

```
## Not run:
administration_mesh(code = c(35201), type = "city")
administration_mesh(code = "08220", type = "city")
administration_mesh(code = c("08220", "08221"), type = "city")
administration_mesh(code = 35, type = "prefecture")
administration_mesh(code = c(33, 34), type = "prefecture")

## End(Not run)
```

---

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

---

**Description**

Return coarse gather mesh codes

**Usage**

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

**Arguments**

meshes	mesh code sets
distinct	return unique meshcodes

**Value**

character vector

**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, mesh_size = "1km", 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)
mesh_size	mesh type. From 80km to 125m
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)

**See Also**

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

**Examples**

```
coords_to_mesh(141.3468, 43.06462, mesh_size = "10km")
coords_to_mesh(139.6917, 35.68949, mesh_size = "250m")
coords_to_mesh(139.71475, 35.70078)

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      integer. 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**

```
## Not run:  
eval_jp_boundary(139.71471056, 35.70128943)  
  
## End(Not run)
```

---

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

---

**Description**

Convert and export meshcode area to sfc\_POLYGON.

**Usage**

```
export_mesh(meshcode)
```

**Arguments**

meshcode	integer. mesh code
----------	--------------------

**Value**

sf object

**Examples**

```
export_mesh(6441427712)
```

---

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

---

**Description**

Convert and export meshcode area to sf.

**Usage**

```
export_meshes(meshes)
```

**Arguments**

meshes	mesh code sets
--------	----------------

**Examples**

```
export_meshes("4128")  
## Not run:  
find_neighbor_mesh(37250395) %>%  
  export_meshes()  
  
## End(Not run)
```

---

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

---

**Description**

Return contains fine mesh codes

**Usage**

```
fine_separate(meshcode = NULL, ...)
```

**Arguments**

meshcode	integer. mesh code
...	other parameters for paste

**Value**

character vector

**Examples**

```

fine_separate(5235)
fine_separate(523504)
fine_separate(52350432)
fine_separate(523504321)
fine_separate(5235043211)

```

---

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	integer. 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("80km", "10km", "1km"))
```

### Arguments

mesh\_size      Export mesh size from 80km to 1km.

### Examples

```
meshcode_set(mesh_size = "80km")
```

---

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

---

### Description

mesh centroid

### Usage

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



### Arguments

meshcode        integer. 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)

### See Also

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

### Examples

```
mesh_to_coords(64414277)
```

---

mesh\_viewer

*interactive meshcode check*

---

### 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)
```

**Arguments**

meshcode	integer. mesh code
contains	logical. contains input meshcode (default TRUE)

**Value**

mesh code vectors (character)

**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 = c("1km"))
```

**Arguments**

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

**Examples**

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

---

`sf_jpmesh`*1:200,000 Scale Maps Name with Meshcode of Japan.*

---

**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

## \*Topic **datasets**

jpnrect, [7](#)

sf\_jpmesh, [11](#)

administration\_mesh, [2](#)

coarse\_gather, [3](#)

coords\_to\_mesh, [3](#)

coords\_to\_mesh(), [9](#)

cut\_off, [4](#)

eval\_jp\_boundary, [5](#)

export\_mesh, [5](#)

export\_meshes, [6](#)

fine\_separate, [6](#)

is\_corner (is\_mesh), [7](#)

is\_mesh, [7](#)

is\_meshcode (is\_mesh), [7](#)

jpnrect, [7](#)

mesh\_to\_coords, [8](#)

mesh\_to\_coords(), [4](#)

mesh\_viewer, [9](#)

meshcode\_set, [8](#)

neighbor\_mesh, [10](#)

rmesh, [10](#)

sf\_jpmesh, [11](#)