

# Package ‘waver’

January 29, 2018

**Type** Package

**Title** Calculate Fetch and Wave Energy

**Version** 0.2.1

**Description** Functions for calculating the  
fetch (length of open water distance along given directions)  
and estimating wave energy from wind and wave monitoring data.

**License** GPL-3

**URL** <https://github.com/pmarchand1/waver>

**BugReports** <https://github.com/pmarchand1/waver/issues>

**Depends** R (>= 3.2.2), rgdal (>= 0.8), sp (>= 1.1)

**Imports** geosphere (>= 1.5), methods, rgeos (>= 0.3)

**LazyData** TRUE

**Suggests** testthat

**RoxygenNote** 6.0.1

**NeedsCompilation** no

**Author** Philippe Marchand [aut, cre],  
David Gill [aut]

**Maintainer** Philippe Marchand <marchand.philippe@gmail.com>

**Repository** CRAN

**Date/Publication** 2018-01-29 19:59:44 UTC

## R topics documented:

fetch_len	2
fetch_len_multi	3
wave_energy	4

<b>Index</b>	<b>6</b>
--------------	----------

---

 fetch\_len

 Calculate the fetch length around a point
 

---

### Description

Given a point, a shoreline layer and a vector of wind directions (bearings), `fetch_len` calculates the distance from point to shore for each bearing.

### Usage

```
fetch_len(p, bearings, shoreline, dmax, spread = 0, projected = FALSE,
         check_inputs = TRUE)
```

### Arguments

<code>p</code>	SpatialPoints* object of length 1 (single point).
<code>bearings</code>	Vector of bearings, in degrees.
<code>shoreline</code>	SpatialLines* or SpatialPolygons* object representing the shoreline.
<code>dmax</code>	Maximum value of fetch length, returned if there is no land within a distance of <code>dmax</code> from a given bearing.
<code>spread</code>	Vector of relative bearings (in degrees) for which to calculate fetch around each main bearing (see details).
<code>projected</code>	Should projected coordinates be used to calculate fetch?
<code>check_inputs</code>	Should the validity of inputs be checked? It is recommended to keep this TRUE, unless this function is called repeatedly from another function that already checks inputs.

### Details

The fetch length (or fetch) is the distance of open water over which the wind can blow in a specific direction. Note that bearings represent the direction from where the wind originates.

The optional `spread` argument defines relative directions that are added to each main bearing to produce a set of sub-bearings. The fetch lengths calculated for each sub-bearing are averaged with weights proportional to  $\cos(\text{spread})$ . By default, `spread = 0` and fetch length is calculated for the main bearings only.

If `projected` is FALSE (the default), the input data must be in WGS84 geographic (longitude, latitude) coordinates. Geodesic distances are calculated using the `distGeo` function from the `geosphere` R package. All distance are expressed in meters.

If `projected` is TRUE, the input data (`p` and `shoreline`) must share the same projection. Projected distances are calculated with the `rgeos` R package. All distances are expressed in the projection's coordinates.

If the shoreline layer is given as `SpatialPolygons*`, the function verifies that the input point is outside all polygons (i.e. in water). If this is not the case, it issues a warning and returns a vector of NA.

**Value**

A named vector representing the fetch length for each direction given in bearings.

**See Also**

[fetch\\_len\\_multi](#) for an efficient alternative when computing fetch length for multiple points.

**Examples**

```
pt <- SpatialPoints(matrix(c(0, 0), ncol = 2),
                        proj4string = CRS("+proj=longlat"))
# Shoreline is a rectangle from (-0.2, 0.25) to (0.3, 0.5)
rect <- Polygon(cbind(c(rep(-0.2, 2), rep(0.3, 2), -0.2),
                      c(0.25, rep(0.3, 2), rep(0.25, 2))))
land <- SpatialPolygons(list(Polygons(list(rect), ID = 1)),
                        proj4string = CRS("+proj=longlat"))
fetch_len(pt, bearings = c(0, 45, 225, 315), land,
          dmax = 50000, spread = c(-10, 0, 10))
```

---

 fetch\_len\_multi

---

*Calculate the fetch length for multiple points*


---

**Description**

fetch\_len\_multi provides two methods to efficiently compute fetch length for multiple points.

**Usage**

```
fetch_len_multi(pts, bearings, shoreline, dmax, spread = 0,
               method = "btree", projected = FALSE)
```

**Arguments**

pts	A SpatialPoints* object.
bearings	Vector of bearings, in degrees.
shoreline	SpatialLines* or SpatialPolygons* object representing the shoreline.
dmax	Maximum value of fetch length, returned if there is no land within a distance of dmax from a given bearing.
spread	Vector of relative bearings (in degrees) for which to calculate fetch around each main bearing.
method	Whether to use the "btree" (default) or "clip" method. See below for more details.
projected	Should projected coordinates be used to calculate fetch?

### Details

With `method = "btree"`, the `gBinarySTRtreeQuery` function from the `rgeos` package is called to determine which polygons in `shoreline` could be within `dmax` of each point. This is a fast calculation based on bounding box overlap.

With `method = "clip"`, the `shoreline` layer is clipped to a polygon formed by the union of rectangular buffers around each point.

In both cases, `fetch_len` is then applied to each point, using only the necessary portion of the shoreline.

Generally, the "clip" method will produce the biggest time savings when points are clustered within distances less than `dmax` (so their clipping rectangles overlap), whereas the "btree" method will be more efficient when the shoreline is composed of multiple polygons and points are distant from each other.

### Value

A matrix of fetch lengths, with one row by point in `pts` and one column by bearing in `bearings`.

### See Also

`fetch_len` for details on the fetch length computation.

---

<code>wave_energy</code>	<i>Calculate the wave energy flux</i>
--------------------------	---------------------------------------

---

### Description

Calculates the wave energy flux (power per meter of wave crest) given either (1) the significant wave height and peak period or (2) the wind speed at 10m, fetch length and (optionally) water depth.

### Usage

```
wave_energy(height = NA, period = NA, wind = NA, fetch = NA,
            depth = NA)
```

### Arguments

<code>height</code>	Significant wave height, in meters.
<code>period</code>	Peak wave period, in seconds.
<code>wind</code>	Wind speed at 10m, in m/s.
<code>fetch</code>	Fetch length, in meters.
<code>depth</code>	Water depth, in meters.

**Details**

Given the significant height ( $H$ ) and peak period ( $T$ ), the wave energy flux is calculated as:

$$\frac{\rho g^2}{64\pi} H^2 T$$

, where  $\rho$  is the density of water (998 kg/m<sup>3</sup>) and  $g$  is the acceleration of gravity (9.81 m/s<sup>2</sup>).

If both height and period are missing, they are estimated from on the wind speed at 10m ( $U_{10}$ ) and the fetch length ( $F$ ) as described in Resio et al. (2003):

$$U_f^2 = 0.001(1.1 + 0.035U_{10})U_{10}^2$$

(friction velocity)

$$\frac{gH}{U_f^2} = \min(0.0413\sqrt{\frac{gF}{U_f^2}}, 211.5)$$

$$\frac{gT}{U_f} = \min(0.651(\frac{gF}{U_f^2})^{1/3}, 239.8)$$

If the depth ( $d$ ) is specified, it imposes a limit on the peak period:

$$T_{max} = 9.78\sqrt{\frac{d}{g}}$$

(in seconds)

**Value**

Wave energy flux, in kW/m.

**References**

Resio, D.T., Bratos, S.M., and Thompson, E.F. (2003). Meteorology and Wave Climate, Chapter II-2. Coastal Engineering Manual. US Army Corps of Engineers, Washington DC, 72pp.

**Examples**

```
# With height and period arguments
wave_energy(8, 1)
```

```
# With wind, fetch and depth arguments (must be named)
wave_energy(wind = 12, fetch = 15000, depth = 10)
```

# Index

`distGeo`, [2](#)

`fetch_len`, [2](#), [4](#)

`fetch_len_multi`, [3](#), [3](#)

`gBinarySTRtreeQuery`, [4](#)

`wave_energy`, [4](#)