# Package 'geoknife'

October 4, 2015

**Type** Package

**Title** Web-Processing of Large Gridded Datasets

**Version** 1.0.0

**Date** 2015-10-02

**Description** Processes gridded datasets found on the
U.S. Geological Survey Geo Data Portal web application or elsewhere,
using a web-enabled workflow that eliminates the need to download and store
large datasets that are reliably hosted on the Internet. The package provides
access to several data subset and summarization algorithms that are available
on remote web processing servers.

**License** CC0

**URL** https://github.com/USGS-R/geoknife

**BugReports** https://github.com/USGS-R/geoknife/issues

**Copyright** This software is in the public domain because it contains
materials that originally came from the United States
Geological Survey, an agency of the United States Department of
Interior. For more information, see the official USGS copyright
policy at
http://www.usgs.gov/visual-id/credit_usgs.html#copyright

**Depends** R (>= 3.0)

**Imports** XML, methods, httr, sp, utils

**Suggests** testthat, xtable, knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Jordan Read [aut, cre],
Jordan Walker [aut],
Alison Appling [aut],
David Blodgett [aut],
Emily Read [aut],
Luke Winslow [aut]

**Maintainer** Jordan Read <jread@usgs.gov>

**Repository** CRAN

**Date/Publication** 2015-10-04 19:46:23

# R topics documented:



---

`abstract` *get abstract from a datagroup*

---

## Description

extracts the abstract information from a datagroup object

## Usage

```
abstract(.Object)

## S4 method for signature 'datagroup'
abstract(.Object)

title(.Object)

## S4 method for signature 'datagroup'
title(.Object)
```

## Arguments

| | |
|---|---|
| .Object | a datagroup object |

---

`attribute<-` *the attribute of an webgeom object*

---

## Description

the attribute of an webgeom object

## Usage

```
attribute(.Object) <- value

attribute(.Object)

## S4 replacement method for signature 'webgeom'
attribute(.Object) <- value
```

## Arguments

| | |
|---|---|
| .Object | a webgeom object |
| value | a attribute |

cancel *cancel a geo-web processing request*

## Description

Cancel process for geojob

## Usage

```
cancel(.Object)

## S4 method for signature 'geojob'
cancel(.Object)

## S4 method for signature 'missing'
cancel(.Object)
```

## Arguments

| | |
|---|---|
| .Object | a geojob object with an active geo-web processing request. |

## Details

cancel is a method for cancelling a geo-web processing request.

## Value

A geojob object with no active job

## See Also

check, start

## Examples

```
wd <- webdata('prism')
wg <- webgeom('state::New Hampshire')
wp <- webprocess()
gj <- geojob()
xml(gj) <- XML(wg, wd, wp)
url(gj) <- url(wp)
## Not run:
gj <- start(gj)
gj <- cancel(gj)

## End(Not run)
```

check *Check status of processing request*

## Description

Check status of processing request

## Usage

```
check(.Object)

## S4 method for signature 'geojob'
check(.Object)
```

## Arguments

| | |
|---|---|
| .Object | a geojob object with an active GDP process request. |

## Details

check is a method for checking the process status of an active (executed) geojob object. The method returns process, which is a list containing two fields: status and URL. If the geojob object has not been executed (see start), this method returns status='none' and URL=NULL.

## Value

process, a list containing status and URL.

## Author(s)

Jordan S. Read

## See Also

start

## Examples

```
gj <- geojob() # create geojob object
check(gj) # no process for empty geojob object
```

`datagroup` *create datagroup object*

## Description

A class representing a geoknife job (`datagroup`).

## Usage

```
datagroup(...)

## S4 method for signature 'ANY'
datagroup(...)

## S4 method for signature 'datagroup'
length(x)

## S4 method for signature 'datagroup'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'datagroup,ANY,ANY'
x[[i, j, ..., drop = TRUE]]
```

## Arguments

| | |
|---|---|
| `...` | additional arguments passed to initialize method |
| `x` | a datagroup object |
| `i` | index specifying elements to extract or replace. |
| `j` | not implemented |
| `drop` | not implemented |

## Value

the datagroup object

## Author(s)

Jordan S Read

## datagroup-class *datagroup class*

### Description

datagroup class

### Slots

`group` a list of webdata compatable elements

## download *download output from geojob*

### Description

download output from geojob

### Usage

```
download(.Object, destination, ...)

## S4 method for signature 'geojob,missing'
download(.Object, destination, ...)

## S4 method for signature 'geojob,character'
download(.Object, destination, ...)
```

### Arguments

| | |
|---|---|
| `.Object` | a `geojob` that has completed |
| `destination` | a file destination. If missing, a temp directory will be used |
| `...` | additional arguments passed to `write_disk`, such as overwrite = TRUE |

### Value

the file handle

### Author(s)

Jordan S Read

geojob *create geojob object*

## Description

A class representing a geoknife job (geojob).

process id of geojob

## Usage

```
geojob(...)

## S4 method for signature 'ANY'
geojob(...)

xml(.Object) <- value

xml(.Object)

id(.Object)
id(.Object) <- value

id(.Object)

## S4 replacement method for signature 'geojob'
id(.Object) <- value

## S4 method for signature 'geojob'
id(.Object)
```

## Arguments

| | |
|---|---|
| ... | additional arguments passed to initialize method |
| .Object | a geojob object |
| value | a character string of xml |

## Value

the geojob object

## Author(s)

Jordan S Read

## Examples

```
xml <- "<foo> <bar> text <baz/> </bar> </foo>"
gj <- geojob()
xml(gj) <- xml
xml(gj)
xml <- "<foo> <bar> text <baz/> </bar> </foo>"
gj <- geojob(xml = xml)
xml(gj)
id(gj)
```

---

# geojob-class *geojob class*

---

## Description

geojob class

## Slots

`url` URL of web processing endpoint

`xml` XML character for post

`id` job identifier

`package.version` the version of the geoknife package

`algorithm.version` the version of the algorithm used for processing

---

# geoknife *geoknife*

---

## Description

geoknife

## Usage

```
geoknife(stencil, fabric, knife = webprocess(...), ...)
```

## Arguments

| | |
|---|---|
| `stencil` | a `webgeom`, `simplegeom`, or any type that can be coerced into `simplegeom`. |
| `fabric` | a dataset. A `webdata` or any type that can be coerced into `webdata` |
| `knife` | (optional) a `webprocess` object |
| `...` | additional arguments passed to `new webprocess`. Can also be used to modify the `knife` argument, if it is supplied. |

## Details

The `stencil` argument is akin to cookie cutter(s), which specify how the dataset is to be subsampled spatially. Supported types are all geometric in nature, be they collections of points or polygons. Because geoporcessing operations require a non-zero area for `stencil`, if points are used (i.e., the different point collections that can be used in `simplegeom`), there is a negligle automatic point buffer applied to each point to result in a non-zero area.

Naming of the components of the `stencil` will impact the formatting of the result returned by the geoknife processing job (the `geojob`)

geoknife will check the class of the stencil argument, and if stencil's class is not `webgeom`, it will attempt to coerce the object into a `simplegeom`. If no coercion method exists, geoknife will fail.

The `fabric` argument is akin to the dough or fabric that will be subset with the `stencil` argument. At present, this is a web-available gridded dataset that meets a variety of formatting restrictions. Several quick start methods for creating a `webdata` object (only `webdata` or an type that can be coerced into `webdata` are valid arguments for `fabric`).

## Value

and object of class geojob

## Examples

```
## Not run:
job <- geoknife(stencil = c(-89,42), fabric = 'prism')
check(job)

#-- set up geoknife to email user when the process is complete

job <- geoknife(webgeom("state::Wisconsin"), fabric = 'prism', email = 'fake.email@gmail.com')


## End(Not run)
```

---

# geom<- *the geom of an object*

---

## Description

the geom of an object

## Usage

```
geom(.Object) <- value

geom(.Object)

## S4 replacement method for signature 'webgeom'
geom(.Object) <- value
```

```
## S4 method for signature 'webgeom'
geom(.Object)
```

### Arguments

| | |
|---|---|
| .Object | a webgeom object |
| value | a geom |

---

`parseTimeseries` *parse timeseries file into R environment*

---

### Description

parse timeseries file into R environment

### Usage

```
parseTimeseries(file, delim, with.units = FALSE)
```

### Arguments

| | |
|---|---|
| file | a geojob timeseries processing result file location (See check). |
| delim | the file delimiter |
| with.units | boolean for including a units column in returned data.frame |

### Details

a function for loading data into R from a file (or URL) from a completed processing request

### Value

a data.frame of timeseries values.

### Author(s)

Luke A. Winslow, Jordan S. Read

### Examples

```
local_file <- system.file('extdata','tsv_linear_ring.tsv', package = 'geoknife')
output <- parseTimeseries(local_file, delim = '\t')
```

query *query webdata for various fields*

## Description

query webdata for various fields

## Usage

```
query(.Object, field, ...)

## S4 method for signature 'webdata,character'
query(.Object, field, ...)

## S4 method for signature 'webdata,missing'
query(.Object, field, ...)

## S4 method for signature 'character,missing'
query(.Object, field, ...)

## S4 method for signature 'webgeom,character'
query(.Object, field, ...)

## S4 method for signature 'webprocess,character'
query(.Object, field, ...)
```

## Arguments

| | |
|---|---|
| .Object | a webdata, webgeom, or webprocess object. |
| field | a plural parameter name for fields in .Object (e.g., 'variables', 'times') |
| ... | additional arguments passed to methods |

## Details

a method for finding possible values for a given field

## Value

a character vector of values corresponding to the query field specified

## Author(s)

Jordan S. Read

## Examples

```
fabric <- webdata('prism')
query(fabric, 'variables')
wg <- webgeom()
query(wg, 'geoms')
geom(wg) <- "derivative:CONUS_States"
query(wg, 'attributes')
attribute(wg) <- 'STATE'
query(wg, 'values', rm.duplicates = TRUE)
```

---

`result` *parse process output into R environment*

---

## Description

parse process output into R environment

## Usage

```
result(.Object, ...)

## S4 method for signature 'geojob'
result(.Object, ...)
```

## Arguments

| | |
|---|---|
| `.Object` | a `geojob` object with a successful processID. (See `check`). |
| `...` | additional arguments passed to parsers (e.g., keep.units = TRUE) |

## Details

a `geojob` method for loading data into R from a completed processing request

## Value

data.frame of timeseries values.

## Author(s)

Jordan S. Read

## Examples

```
## Not run:
job <- geoknife(stencil = c(-89,42), fabric = 'prism', wait = TRUE)
result(job, with.units = TRUE) # load and print output

## End(Not run)
```

`simplegeom` *create simplegeom object*

## Description

create simplegeom object

## Usage

```
simplegeom(.Object, ...)

## S4 method for signature 'missing'
simplegeom(.Object, ...)

## S4 method for signature 'ANY'
simplegeom(.Object, ...)
```

## Arguments

| | |
|---|---|
| `.Object` | any object that can be coerced into simplegeom |
| `...` | additional arguments passed to SpatialPolygonsDataFrame |

## Value

the simplegeom object

## Author(s)

Jordan S Read

## Examples

```
simplegeom(c(-88.6, 45.2))
## Not run:
simplegeom(Srl, proj4string = CRS("+proj=longlat +datum=WGS84"))

## End(Not run)
simplegeom(data.frame('point1'=c(-89, 46), 'point2'=c(-88.6, 45.2)))
```

## simplegeom-class *simplegeom class*

### Description

The simplegeom class represents geometries that can be coerced into polygon features. This is one of two stencil types accepted by geoknife (the other being webgeom).

### Details

The difference between webgeom and simplegeom is both in the permenance and the location of the data. webgeom is located on a web server that offers geometries using the web feature service (WFS) specification. simplegeom are typically local data that can be accessed within an R session. Within reason, anything that can be represented as a webgeom (or WFS) can also be represented by a simplegeom For example, a state or watershed can be read in as SpatialPolygons object and turned into a simplegeom.

### Slots

sp a SpatialPolygons object

DRAW_NAMESPACE (_private) web location of draw namespace

DRAW_SCHEMA (_private) web location of draw schema

## start *start a geo-web processing request*

### Description

Start process for geojob

### Usage

```
start(.Object)

## S4 method for signature 'geojob'
start(.Object)
```

### Arguments

| | |
|---|---|
| .Object | a geojob object |

### Details

start is a method for submitting a geo-web processing request.

## Value

A geojob object with an active GDP process request.

## See Also

check

## Examples

```
wd <- webdata('prism')
wg <- webgeom('state::New Hampshire')
wp <- webprocess()
gj <- geojob()
## Not run:
xml(gj) <- XML(wg, wd, wp)
url(gj) <- url(wp)
gj <- start(gj)

## End(Not run)
```

---

# successful *Convenience function for GDP process state*

---

## Description

Simple wrapper to check process status

## Usage

```
successful(.Object, retry)
error(.Object, retry)
running(.Object, retry)

## S4 method for signature 'geojob'
successful(.Object, retry = FALSE)

## S4 method for signature 'geojob'
running(.Object, retry = FALSE)

## S4 method for signature 'geojob'
error(.Object, retry = FALSE)
```

## Arguments

| | |
|---|---|
| .Object | a geojob object |
| retry | attempt to retry again if communication failed with the server |

## Value

TRUE/FALSE indicating if process is in the given state (error, processing, successful)

## Author(s)

Luke Winslow, Jordan S Read

## See Also

check

## Examples

```
## Not run:
wp <- quick_wp()
job <- geoknife(stencil = c(-89,42), fabric = 'prism', knife = wp)
check(job)

running(job)
error(job)
successful(job)

## End(Not run)
```

---

# times *the times of an webdata object*

---

## Description

Functions to get or set the times of a webdata object

## Usage

```
times(.Object)

times(.Object) <- value

## S4 replacement method for signature 'webdata'
times(.Object) <- value

## S4 method for signature 'webdata'
times(.Object)
```

## Arguments

| | |
|---|---|
| .Object | a webdata object |
| value | a POSIXct vector |

## Examples

```
wd <- webdata('prism')
times(wd) <- as.POSIXct(c("2012-11-04", "2012-11-12"))
times(wd)[1] <- as.POSIXct("2012-11-04")
times(wd)
```

---

`url<-` *the url of an object*

---

## Description

the url of an object

## Usage

```
url(.Object)
url(.Object)<- value

url(.Object)

## S4 replacement method for signature 'ANY'
url(.Object) <- value

## S4 replacement method for signature 'webprocess'
url(.Object) <- value

## S4 method for signature 'datagroup'
url(.Object)

## S4 method for signature 'ANY'
url(.Object)
```

## Arguments

| | |
|---|---|
| `.Object` | a webgeom, webdata, geojob, or webprocess object |
| `value` | a url |

---

`values<-` *the values of an object*

---

## Description

the values of an object

## Usage

```
values(.Object) <- value

values(.Object)

## S4 replacement method for signature 'webgeom'
values(.Object) <- value

## S4 method for signature 'webgeom'
values(.Object)
```

## Arguments

| | |
|---|---|
| .Object | a webgeom object |
| value | a values |

---

`variables` *the variables of a webdata object*

---

## Description

the variables of a webdata object

## Usage

```
variables(.Object)
variables(.Object) <- value

variables(.Object) <- value

## S4 method for signature 'webdata'
variables(.Object)

## S4 replacement method for signature 'webdata'
variables(.Object) <- value
```

## Arguments

| | |
|---|---|
| .Object | a webdata object |
| value | a character vector for variables |

## version<- *the version of an object*

### Description

the version of an object

### Usage

```
version(.Object) <- value

version(.Object)

## S4 replacement method for signature 'ANY'
version(.Object) <- value

## S4 method for signature 'ANY'
version(.Object)
```

### Arguments

| | |
|---|---|
| .Object | a webgeom or webprocess object |
| value | a version |

## webdata *create webdata object*

### Description

A class representing a web dataset.

### Usage

```
webdata(.Object, ...)

## S4 method for signature 'missing'
webdata(.Object, ...)

## S4 method for signature 'ANY'
webdata(.Object, ...)
```

### Arguments

| | |
|---|---|
| .Object | any object that can be coerced into webdata |
| ... | additional arguments passed initialize method (e.g., times, or any other in the webdata object. |

## Value

the webdata object representing a dataset and parameters

## Slots

`times` value of type `"POSIXct"`, start and stop dates for data

`url` value of type `"character"`, the web location for the dataset

`variable` value of type `"character"`, the variable(s) for data

## Author(s)

Jordan S Read

## Examples

```
webdata('prism')
webdata('prism', times=as.POSIXct(c('1990-01-01', '1995-01-01')))
webdata(list(times = as.POSIXct(c('1895-01-01 00:00:00','1899-01-01 00:00:00')),
 url = 'http://cida.usgs.gov/thredds/dodsC/prism',
 variables = 'ppt'))
```

---

# webdata-class *webdata class*

---

## Description

webdata class

## Slots

`times` vector of POSIXct dates (specifying start and end time of processing)

`url` URL of web data

`variables` variable(s) used for processin from dataset

webgeom *create webgeom object*

## Description

A class representing a web dataset.

## Usage

```
webgeom(.Object, ...)

## S4 method for signature 'missing'
webgeom(.Object, ...)

## S4 method for signature 'ANY'
webgeom(.Object, ...)
```

## Arguments

| | |
|---|---|
| .Object | any object that can be coerced into webgeom |
| ... | additional arguments passed initialize method (e.g., url) |

## Value

the webgeom object representing a dataset and parameters

## Slots

`url` value of type "character", the web location for the dataset

`variable` value of type "character", the variable(s) for data

## Author(s)

Jordan S Read

## Examples

```
wg <- webgeom(geom = "sample:CONUS_states",
 attribute = "STATE",
 values = "New Hampshire")
#-- use available state datasets:
wg <- webgeom('state::New Hampshire')
wg <- webgeom('state::New Hampshire,Wisconsin,Alabama')
#-- use available Level III Ecoregion datasets:
wg <- webgeom('ecoregion::Colorado Plateaus,Driftless Area')
#-- use available simplified HUC8s:
wg <- webgeom('HUC8::09020306,14060009')
wg <- webgeom()
```

webgeom-class *webgeom class*

## Description

The webgeom class represents a web feature service (WFS) dataset. WFS is an open geospatial consortium standard for spatial data on the web. WFS supports filtering of spatial elements and this object can support many of those filters.

## Slots

`url` URL of web feature service endpoint. Can be set or accessed using `url`

`geom` character for geometric feature name. Can be set or accessed using `geom`

`attribute` character for feature attribute (used for filtering and naming in output) Can be set or accessed using `attribute`

`values` character vector of attribute values to be used in processing (a subset, or all if NA) Can be set or accessed using `values`

`version` a character that specifies the web feature service (WFS) version to use. Can be set or accessed using `version`

`GML_IDs` (_private) IDs that correspond to `values`. Used internally for processing.

`WFS_NAMESPACE` (_private) web location of web feature service namespace

`GML_NAMESPACE` (_private) web location of GML namespace

`GML_SCHEMA_LOCATION` (_private) web location of GML schema location

## See Also

`webgeom`, `url`, `geom`, `attribute`, `values`, `version`

webprocess *create webprocess object*

## Description

create webprocess object

## Usage

```
webprocess(.Object, ...)

## S4 method for signature 'missing'
webprocess(.Object, ...)

## S4 method for signature 'ANY'
webprocess(.Object, ...)
```

### Arguments

| | |
|---|---|
| .Object | any object that can be coerced into webprocess |
| ... | additional arguments passed initialize method (e.g., url, version) |

### Value

the webprocess object

### Author(s)

Jordan S Read

---

## webprocess-class *webprocess class*

---

### Description

A class representing geoknife web processing specifications

### Slots

`url` URL for webprocessing service. Can be set or accessed using `url`

`algorithm` a list for algorithm used. Can be set or accessed using `algorithm`

`version` a character specifying the wep processing service version to use. Can be set or accessed using `version`

`email` an email to send finished process alert to

`wait` boolean for wait until complete (hold up R until processing is complete)

`processInputs` (_private) a list of required and options process inputs, and their default values (if specified). This is populated (or repopulated) whenever `algorithm` is set.

`WPS_SCHEMA_LOCATION` (_private) location for web processing service schema

`WPS_NAMESPACE` (_private) location for web processing service namespace

`OWS_NAMESPACE` (_private) namespace web location

`XSI_SCHEMA_LOCATION` (_private) schema web location

`XSI_NAMESPACE` (_private) namespace web location

`XLINK_NAMESPACE` (_private) namespace web location

`UTILITY_URL` (_private) web processing service utility url. Uses same base url as public slot `url`

`OGC_NAMESPACE` (_private) namespace web location

`emailK` (_private) relative url for email when complete utility.

### See Also

`webprocess`, `url`, `algorithm`, `version`

XML *XML from set of objects*

## Description

XML from set of objects

## Usage

```
XML(stencil, fabric, knife)

## S4 method for signature 'ANY,webdata,webprocess'
XML(stencil, fabric, knife)
```

## Arguments

| | |
|---|---|
| stencil | a webdata OR simplegeom object |
| fabric | a webdata object |
| knife | a webprocess object |

## Value

XML as ?string?

## Examples

```
wd <- webdata('prism',times = as.POSIXct(c('2001-01-01','2002-02-05')))
wg <- webgeom('state::Wisconsin')
## Not run:
XML(wg, wd, webprocess())
sg <- simplegeom(c(-89,45))
XML(sg, wd, webprocess())

## End(Not run)
```

# Index