Package ‘geoknife’

June 2, 2020

Type  Package
Title  Web-Processing of Large Gridded Datasets
Version  1.6.4
Date  2020-05-20
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 (Read et al. (2015) <doi:10.1111/ecog.01880>).
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.
Depends  R (>= 3.0)
Imports  whisker, xml2, methods, httr (>= 1.0.0), curl, sp, utils, progress (>= 1.1.2)
Suggests  testthat, xtable, knitr, markdown, ggmap, dplyr, rasterVis, ggplot2, rgdal, maps
VignetteBuilder  knitr
RoxygenNote  7.1.0
NeedsCompilation  no
Author  Jordan Read [aut, cre], Jordan Walker [aut], Alison Appling [aut], David Blodgett [aut], Emily Read [aut], Luke Winslow [aut],

1
Lindsay Carr [aut],
David Watkins [aut]

Maintainer  Jordan Read <jread@usgs.gov>
Repository  CRAN
Date/Publication  2020-06-02 08:50:25 UTC

R topics documented:

abstract .................................................. 3
algorithm ............................................... 3
attribute<- ........................................... 4
cancel .................................................. 5
check ................................................... 6
datagroup .............................................. 7
datagroup-class ........................................ 8
defaultProcessInputs .................................. 8
download ............................................... 9
gconfig ................................................ 9
ggeojob ............................................... 10
ggeojob-class ......................................... 12
geknife ............................................... 12
geo<- .................................................. 13
parseCategorical ....................................... 14
parseTimeseries ....................................... 15
query .................................................. 15
result .................................................. 17
simplegeom .......................................... 18
simplegeom-class ..................................... 19
start .................................................. 19
successful .......................................... 20
times .................................................. 21
url<- ................................................ 22
values<- ........................................... 23
variables ............................................. 23
version<- .......................................... 24
wait .................................................. 25
webdata ............................................. 26
webdata-class ....................................... 27
webgeom ............................................. 27
webgeom-class ....................................... 29
webprocess ......................................... 30
webprocess-class ................................... 30
XML .................................................. 31

Index 33
abstract

get abstract from a datagroup

Description
extracts the abstract information from a datagroup object

Usage
abstract(.Object)

algorithm(.Object)

algorithm(.Object) <- value

algorithm(.Object)

title(.Object)

Arguments
. Object a datagroup object

algorithm the algorithm of a webprocess object

Description
Functions to get or set the algorithm of a webprocess object. The algorithm is the type of process that will be used, and can be accessed or modified using the algorithm method.

Usage
algorithm(.Object)

algorithm(.Object) <- value

algorithm(.Object)

algorithm(.Object) <- value

algorithm(.Object) <- value

algorithm(.Object) <- value
**Arguments**

- `.Object`: a `webprocess` object
- `value`: a list with name of algorithm and relative url endpoint

**Examples**

```r
## Not run:
wp <- webprocess()
algorithmp(w)
## End(Not run)
```

---

**attribute<->**

the attribute of an `webgeom` object

---

**Description**

get or set the attribute of a `webgeom` object.

**Usage**

```r
attribute(.Object) <- value
attribute(.Object)
```

### S4 replacement method for signature 'webgeom'

```r
attribute(.Object) <- value
```

### S4 method for signature 'webgeom'

```r
attribute(.Object)
```

**Arguments**

- `.Object`: a `webgeom` object
- `value`: a attribute
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)

## S4 method for signature 'character'
check(.Object)

Arguments
.Object a geojob object with an active GDP process request, or a character URL of an existing job

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
**Description**

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

**Usage**

```r
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
contains collections of webdata that can be processed with geoknife

Slots

group a list of webdata compatible elements

defaultProcessInputs Default Process Inputs

Description
parses DescribeProcess request

Usage
defaultProcessInputs(algorithm, wps_url, wps_version)

Arguments
algorithm the WPS algorithm to get process inputs for
wps_url the service base URL for the WPS
wps_version the service version to use

Value
list of default, optional, and required process inputs for use in the webprocess object.
**download**

Download output from geojob

**Description**

download the result of a processing job to a local destination.

**Usage**

download(.Object, destination, ...)

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

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

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

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

**Arguments**

.Object
  a `geojob` or job id 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 location of the downloaded file

**Author(s)**

Jordan S Read

**gconfig**

configure geoknife settings

**Description**

access and set defaults for geoknife configuration
Usage

gconfig(..., no.readonly = FALSE)

Arguments

... values for gconfig
no.readonly currently not implemented for TRUE

Value

Borrowed text and functionality from \texttt{par}. When parameters are set, their previous values are returned in an invisible named list. Such a list can be passed as an argument to \texttt{par} to restore the parameter values. Use \texttt{gconfig(no.readonly = TRUE)} for the full list of parameters that can be restored. When just one parameter is queried, the value of that parameter is returned as (atomic) vector. When two or more parameters are queried, their values are returned in a list, with the list names giving the parameters. Note the inconsistency: setting one parameter returns a list, but querying one parameter returns a vector.

Examples

\begin{verbatim}
gconfig # all config
gconfig('wait')
gconfig('sleep.time' = 10)
gconfig('sleep.time' = 8, wait=TRUE)
\end{verbatim}

---

\texttt{geojob} \hspace{1em} \textit{create geojob object}

Description

A class representing a geoknife job (\texttt{geojob}).

Usage

\begin{verbatim}
geojob(xml, ...)

## S4 method for signature 'missing'
geojob(xml, ...)

## S4 method for signature 'xml_document'
geojob(xml, ...)

## S4 method for signature 'character'
geojob(xml, ...)

xml(.Object) <- value
\end{verbatim}
geojob

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)

## S4 method for signature 'character'
id(.Object)

Arguments

xml location of xml (URL or local path)
...
,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 version="1.0.0"> <bar> text <baz/> </bar> </foo>"
gj <- geojob(xml = xml)
xml(gj)
id(gj)
Description

contains the information for processing the job, and the versions of the resources used.

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

Description

Creates the processing job and allows specifying the processing details.

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 sub-sampled spatially. Supported types are all geometric in nature, be they collections of points or polygons. Because geoprocessing 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 negligible 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`).

Making concurrent requests to the Geo Data Portal will NOT result in faster overall execution times. The data backing the system is on high performance storage, but that storage is not meant to support parallelized random access and can be significantly slower under these conditions. Read more: https://my.usgs.gov/confluence/display/GeoDataPortal/Geo+Data+Portal+Scalability+Guidelines

Value

and object of class `geojob`

Examples

```r
## 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)
```

---

```r
geom<- the geom of an object
```

Description

The "feature" of a webgeom. This is the key mapping to the web resource that is used as the spatial feature of reference. Other details specified in `attribute` and `values`.

Usage

```r
geom(.Object) <- value
geom(.Object)

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

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

---

```
geom<- the geom of an object
```

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

- `.Object`: a `webgeom` object
- `value`: a `geom`

See Also

- `attribute` and `values`

Description

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

Usage

```r
parseCategorical(file, delim)
```

Arguments

- `file`: a `geojob` categorical processing result file location (See `download`).
- `delim`: the file delimiter

Value

A `data.frame` of categorical fraction (and/or count) values.

See Also

- `check`, `download`, `parseTimeseries`

Examples

```r
local.file <- system.file('extdata', 'csv_categorical_multifeature.csv', package = 'geoknife')
output <- parseCategorical(local.file, delim = ',')
```
**parseTimeseries**  
*parse timeseries file into R environment*

**Description**

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

**Usage**

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

**Arguments**

- `file` (a geojob timeseries processing result file location [See `download`].)
- `delim` (the file delimiter)
- `with.units` (boolean for including a units column in returned data.frame, default = FALSE)

**Value**

A data.frame of timeseries values.

**Author(s)**

Luke A. Winslow, Jordan S. Read

**See Also**

- `check`, `download`, `parseCategorical`

**Examples**

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

**query**  
*query webdata for various fields*

**Description**

A method for finding possible values for a given field.
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

Value

a character vector of values corresponding to the query field specified

Author(s)

Jordan S. Read

Examples

## Not run:
fabric <- webdata('prism')
query(fabric, 'variables')
wg <- webgeom()
query(wg, 'geoms')
geom(wg) <- "sample:CONUS_states"
query(wg, 'attributes')
attribute(wg) <- 'STATE'
query(wg, 'values', rm.duplicates = TRUE)

## End(Not run)
Description

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

Usage

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

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

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

Arguments

- `.Object` - A `geojob` object with a successful processID, or a character URL of a completed job. (See `check`).
- `...` - Additional arguments passed to parsers (e.g., `with.units = TRUE`).

Value

data.frame of timeseries values.

Author(s)

Jordan S. Read

Examples

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

# or use the job id:
id <- id(job)
result(id, with.units = TRUE) # load and print output
```

## End(Not run)
simplegeom

Create simplegeom object

Description

A simple geom is a simple set of geometries specified locally. See webgeom for web features.

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:

library(sp)

Sr1 <- Polygon(cbind(c(-89.0001,-89,-88.9999,-89,-89.0001),c(46,46.0001,46,45.9999,46)))

Sr2 <- Polygon(cbind(c(-88.6,-88.5999,-88.5999,-88.6,-88.6),c(45.2,45.2,45.1999,45.1999,45.2)))

Srs1 <- Polygons(list(Sr1), "s1")

Srs2 <- Polygons(list(Sr2), "s2")

SP <- SpatialPolygons(list(Srs1,Srs2), proj4string = CRS("+proj=longlat +datum=WGS84"))

result(geoknife(simplegeom(SP), 'prism', wait=TRUE))

## End(Not run)

simplegeom(data.frame('point1'=c(-89, 46), 'point2'=c(-88.6, 45.2)))
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 permanence 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
Submit a GDP 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 a geo-web processing request
start is a method for submitting a geo-web processing request.
successful

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)

running(.Object, retry = FALSE)

error(.Object, retry = FALSE)

Arguments

/Object a geojob object or geojob ID (character)
retry logical, 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
## Description
Functions to get or set the times of a `webdata` object

## Usage
```r
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
```r
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**

get or set the url of an object

**Usage**

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

## S4 replacement method for signature 'ANY'

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

## S4 replacement method for signature 'webprocess'

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

## S4 method for signature 'character'

```r
url(.Object, ...)
```

## S4 method for signature 'missing'

```r
url(.Object, ...)
```

## S4 method for signature 'datagroup'

```r
url(.Object, ...)
```

## S4 method for signature 'ANY'

```r
url(.Object, ...)
```

**Arguments**

- **.Object**
  - a `webgeom`, `webdata`, or `webprocess` object

- **value**
  - a url

- **...**
  - additional arguments that would be passed to the masked `base::url` function. These are only used when the `.Object` argument is character or missing `geojob`, or `webprocess` object
The values of a webgeom are the values of the attributes used in the geometries. For example, if the webgeom's "geom" field is a feature collection containing states and counties, and the "attributes" are the states, then the values would be the specific states.

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

Examples

```
wg <- webgeom('state::Wisconsin')
values(wg)
values(wg) <- c('Wisconsin','New Hampshire')
```

The variables of a webdata object

Access or set the variables of a webdata object

```
### Usage

variables(.Object) <- value

variables(.Object) <- value

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

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

### Arguments

- **.Object**
  - A *webdata* object
- **value**
  - A character vector for variables

---

### Description

get the version of webgeom or webprocess

### Usage

version(.Object) <- value

version(.Object)

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

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

### Arguments

- **.Object**
  - A *webgeom* or *webprocess* object
- **value**
  - A version
wait

hold up R while GDP is processing

Description

keeps R in a loop while GDP works on the request. Checks running. Will drop out of loop whenever !running(geojob)

Usage

wait(.Object, sleep.time)

## S4 method for signature 'geojob,numeric'
wait(.Object, sleep.time)

## S4 method for signature 'geojob,missing'
wait(.Object, sleep.time)

## S4 method for signature 'character,numeric'
wait(.Object, sleep.time)

## S4 method for signature 'character,missing'
wait(.Object, sleep.time)

Arguments

/Object a geojob

sleep.time a number of seconds to wait in between checking the process

Value

invisible return of .Object, unaltered

Examples

## Not run:
job <- geoknife(stencil = c(-89,42), fabric = 'prism')
2+2
wait(job)
check(job) # should be complete

## End(Not run)
**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 'character'
webdata(.Object = c("prism", "iclus", "daymet", "gldas", "nldas", "topowx", "solar", "metobs"), ...)

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

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

Arguments

.Object any object that can be coerced into webdata (currently character, webdata, and list)

... 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

```r
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 = 'https://cida.usgs.gov/thredds/dodsC/prism',
               variables = 'ppt'))
```

---

### Description

A class for specifying details of web datasets (webdata!). These datasets have to be accessible through the OPeNDAP protocol or as WCS (web coverage services).

### Slots

- `times` vector of POSIXct dates (specifying start and end time of processing)
- `url` URL of web data
- `variables` variable(s) used for processing from dataset

---

### Description

A class representing a web available feature geometry.

### Usage

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

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

```r
## 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`). See the named slots above for arguments for ...
Details
slots can be accessed or set with methods of the same names (e.g., url(webgeom()))

Value
the webgeom object representing a dataset and parameters

Slots
url value of type "character", the web location for the web feature service
geom value of type "character", the feature for webgeom
attribute the attribute (e.g., "State")
values the values of the attribute, (e.g., "Wisconsin") or NA (all)

Author(s)
Jordan S Read

See Also
url, geom, attribute, values

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()

## Not run:
## Steps to find data on Howard County in Texas:
#1) locate the \code{geom} for counties by looking at the options for geoms
query(webgeom(), 'geoms') # discover sample:Counties
#2) locate the \code{attribute} for county names by looking at the options for attributes
query(webgeom(geom='sample:Counties'), 'attributes') # discover FIPS
#3) find the appropriate fip code for the county:
howard.fips <- maps::county.fips %>%
dplyr::filter(polyname == 'texas,howard') %>% .$fips %>% as.character
#4) create a webgeom for the Howard County in Texas
stencil <- webgeom(geom='sample:Counties', attribute='FIPS', values=howard.fips)
#5) get data for Howard County
variables = "Total_precipitation_surface_1_Hour_Accumulation",
}


times = c(as.POSIXct("2016-06-06 05:00:00"),
        as.POSIXct("2016-06-07 05:00:00")))
job <- geoknife(stencil, fabric, wait = TRUE)
precipData <- result(job)
head(precipData)

## End(Not run)

---

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`
Description

create webprocess object

Usage

webprocess(.Object, ...)

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

## S4 method for signature 'character'
webprocess(
  .Object = c("summary", "unweighted summary", "coverage summary", "subset",
            "coverage subset"),
  ...
)

## 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

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)
sleep.time numeric for time to wait in between calls to `check`. Only used if `wait`=TRUE
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**

Extract important parts of stencil, fabric, and knife into POST XML

**Usage**

`XML(stencil, fabric, knife)`

```r
## 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

*Topic methods
  cancel, 5
  parseCategorical, 14
  parseTimeseries, 15
  query, 15
  result, 17
  [,datagroup-method (datagroup), 7
  [[,datagroup,ANY,ANY-method (datagroup), 7

abstract, 3
abstract,datagroup-method (abstract), 3
algorithm, 3, 31
algorithm,webprocess-method (algorithm), 3
algorithm,xml_document-method (algorithm), 3
algorithm<- (algorithm), 3
algorithm<-,webprocess-method (algorithm), 3
attribute, 13, 14, 28, 29
attribute (attributes<), 4
attribute,webgeom-method (attribute<), 4
attribute<-, 4
attribute<-,webgeom-method (attribute<-), 4

cancel, 5
cancel,geojob-method (cancel), 5
cancel,missing-method (cancel), 5
check, 6, 14, 15, 17, 20, 21, 31
check,character-method (check), 6
check,geojob-method (check), 6

datagroup, 7
datagroup,ANY-method (datagroup), 7
datagroup,datagroup-methods (datagroup), 7
datagroup-class, 8
defaultProcessInputs, 8
download, 9, 14, 15
download,character,character-method (download), 9
download,character,missing-method (download), 9
download,geojob,character-method (download), 9
download,geojob,missing-method (download), 9

error (successful), 20
gconfig, 9
geojob, 5, 6, 9, 10, 11–15, 17, 19, 20, 22
geojob,character-method (geojob), 10
geojob,geojob-method (geojob), 10
geojob,missing-method (geojob), 10
geojob,xml_document-method (geojob), 10
geojob-class, 12
goknife, 8, 12, 19
gom, 28, 29
gom (geom<), 13
gom,webgeom-method (geom<), 13
geom<-, 13
geom<-,webgeom-method (geom<), 13

id (geojob), 10
id,character-method (geojob), 10
id,geojob-method (geojob), 10
id<- (geojob), 10
id<-,geojob-method (geojob), 10

length,datagroup-method (datagroup), 7
par, 10
parseCategorical, 14, 15
parseTimeseries, 14, 15
query, 15