Package 'ccafs'

August 26, 2020

Type Package

Title Client for 'CCAFS' 'GCM' Data

Description Client for Climate Change, Agriculture, and Food Security ('CCAFS') General Circulation Models ('GCM') data. Data is stored in Amazon 'S3', from which we provide functions to fetch data.

Version 0.3.0

License MIT + file LICENSE

URL https://docs.ropensci.org/ccafs/, https://github.com/ropensci/ccafs

BugReports https://github.com/ropensci/ccafs/issues

Encoding UTF-8

Language en-US

Imports rappdirs, crul, raster, tibble, xml2, jsonlite, data.table

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.1.1

X-schema.org-keywords data, climate change, agriculture, gcm, general circulation model, ccafs

X-schema.org-isPartOf ``https://ropensci.org"

X-schema.org-applicationCategory Data Access

NeedsCompilation no

Author Scott Chamberlain [aut, cre] (<https://orcid.org/0000-0003-1444-9135>)

Maintainer Scott Chamberlain <myrmecocystus@gmail.com>

Repository CRAN

Date/Publication 2020-08-26 13:40:12 UTC

R topics documented:

ccafs-package 2
ccafs-search 3
cc_cache 8
cc_data_fetch 9
cc_data_read 10
cc_list_keys 11
cc_search 13

Index **15**

`ccafs-package` *Client for CCAFS GCM Data*

Description

Client for Climate Change, Agriculture, and Food Security (CCAFS) General Circulation Models (GCM) data. Data is stored in Amazon S3, from which we provide functions to fetch data.

About CCAFS

Client for Climate Change, Agriculture, and Food Security (CCAFS) General Circulation Models (GCM) data. Data is stored in Amazon S3, from which we provide functions to fetch data.

R client for Climate Change, Agriculture, and Food Security (CCAFS) General Circulation Models (GCM) data.

CCAFS website: http://ccafs-climate.org/

CCAFS GCM data for this package comes from Amazon S3 http://cgiardata.s3.amazonaws.com More about Amazon S3 below.

CCAFS data can be used for studying climate change, and how climate impacts various aspects of the earth. Search google scholar with "CCAFS" "GCM" to see example uses.

As far as I can tell, CCAFS GCM data comes from IPCC data.

About Amazon S3

Amazon S3 stands for "Simple Storage Service" - it's like a file system, and they give you links to the files and metadata around those links.

S3 is split up into buckets, essentially folder. All CCAFS data is in one bucket. Within the CCAFS bucket on S3 are a series of nested folders. To get to various files we need to navigate down the tree of folders. Keys are file paths with all their parent folders, e.g., "/foo/bar/1/2". Unfortunately, there's no meaningful search of the CCCAFS data as they have on their website http://ccafs-climate.org/ . However, you can set a prefix for a search of these keys, e.g., "/foo/bar" for the key above.

Check out https://aws.amazon.com/s3/ for more info.

About the package

ccafs is a client to work with the data CCAFS provides via Amazon Web Services S3 data.

The **ccafs** data has access to is the "Spatial Downscaling" data that you see on the http://ccafs-climate.org/data/ page. The other data sets are not open.

Currently, we don't provide a way to search for what data is available. You have to know what you want, or you can list what is available, and then pick files from the list. Though there's not a lot of information in the metadata returned from S3.

We'll work on incorporating a way to search - currently there is no solution.

raster

The main useful output are `raster` package objects of class `RasterLayer` or `RasterBrick` - so in general have `raster` loaded in your session to maximize happiness.

Citations

Cite CCAFS data following their guidelines at http://ccafs-climate.org/about/

Get a citation for this package like `citation(package = 'ccafs')`

Vignette

ccafs has the following vignettes:

- `vignette("ccafs",package = "ccafs")`
- `vignette("amazon_s3_keys",package = "ccafs")`

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

---

ccafs-search *cc_search options*

Description

cc_search options

Usage

```
cc_params
```

Format

An object of class `list` of length 9.

Details

`cc_params` is a list with slots matching the parameters in `cc_search()` - so you can more easily get to the options you want to pass in to that function.

The options are also listed below.

file set options

- 12: Delta Method IPCC AR5
- 4: Delta Method IPCC AR4
- 9: MarkSim Pattern Scaling
- 10: Eta South America
- 7: PRECIS Andes
- 8: CORDEX
- 11: Disaggregation IPCC AR4
- 3: Delta Climgen
- 2: Delta Method IPCC AR4 (Climgen Data)
- 5: Delta Method IPCC AR4 (Stanford Data)
- 6: Delta Method IPCC AR3

scenario options

- 1: Baseline
- 2: SRES A1B
- 3: SRES A2A
- 4: SRES B2A
- 5: SRES A2
- 6: SRES B1
- 7: RCP 2.6
- 8: RCP 4.5
- 9: RCP 6.0
- 10: RCP 8.5

model options

- 1: baseline
- 42: bcc_csm1_1
- 43: bcc_csm1_1_m
- 2: bccr_bcm2_0
- 44: bnu_esm
- 45: cccma_cancm4
- 46: cccma_canesm2

- 3: cccma_cgcm2_0
- 4: cccma_cgcm3_1_t47
- 5: cccma_cgcm3_1_t63
- 47: cesm1_bgc
- 48: cesm1_cam5
- 49: cesm1_cam5_1_fv2
- 50: cesm1_fastchem
- 51: cesm1_waccm
- 52: cmcc_cesm
- 53: cmcc_cm
- 54: cmcc_cms
- 6: cnrm_cm3
- 55: cnrm_cm5
- 56: csiro_access1_0
- 57: csiro_access1_3
- 7: csiro_mk2
- 8: csiro_mk3_0
- 25: csiro_mk3_5
- 58: csiro_mk3_6_0
- 59: ec_earth
- 60: fio_esm
- 9: gfdl_cm2_0
- 61: gfdl_cm2_1
- 10: gfdl_cm2_1
- 62: gfdl_cm3
- 63: gfdl_esm2g
- 64: gfdl_esm2m
- 11: giss_aom
- 65: giss_e2_h
- 66: giss_e2_h_cc
- 67: giss_e2_r
- 68: giss_e2_r_cc
- 26: giss_model_eh
- 27: giss_model_er
- 38: hadcm_cntrl
- 39: hadcm_high
- 40: hadcm_low

- 41: hadcm_midi
- 12: hccpr_hadcm3
- 13: iap_fgoals1_0_g
- 28: ingv_echam4
- 37: inm_cm3_0
- 69: inm_cm4
- 14: ipsl_cm4
- 70: ipsl_cm5a_lr
- 71: ipsl_cm5a_mr
- 72: ipsl_cm5b_lr
- 73: lasg_fgoals_g2
- 74: lasg_fgoals_s2
- 15: miroc3_2_hires
- 16: miroc3_2_medres
- 75: miroc_esm
- 76: miroc_esm_chem
- 77: miroc_miroc4h
- 78: miroc_miroc5
- 17: miub_echo_g
- 34: mohc_hadam3p_2
- 35: mohc_hadam3p_3
- 79: mohc_hadcm3
- 31: mohc_hadcm3q0
- 33: mohc_hadcm3q16
- 32: mohc_hadcm3q3
- 80: mohc_hadgem2_cc
- 81: mohc_hadgem2_es
- 29: mpi_echam4
- 18: mpi_echam5
- 30: mpi_echam5
- 82: mpi_esm_lr
- 83: mpi_esm_mr
- 84: mpi_esm_p
- 19: mri_cgcm2_3_2a
- 85: mri_cgcm3
- 20: ncar_ccsm3_0
- 86: ncar_ccsm4

- 21: ncar_pcm1
- 87: ncc_noresm1_m
- 88: ncc_noresm1_me
- 36: ncep_r2
- 22: nies99
- 89: nimr_hadgem2_ao
- 23: ukmo_hadcm3
- 24: ukmo_hadgem1

period options

- 1: 1970s
- 10: 1990s
- 2: 2000s
- 3: 2020s
- 4: 2030s
- 5: 2040s
- 6: 2050s
- 7: 2060s
- 8: 2070s
- 9: 2080s

variable options

- 1: Bioclimatics
- 6: Diurnal Temperature Range
- 3: Maximum Temperature
- 4: Mean Temperature
- 5: Minimum Temperature
- 2: Precipitation
- 7: Solar Radiation
- 9999: Other

resolution options

- 1: 30 seconds
- 2: 2.5
- 3: 5 minutes
- 4: 10 minutes
- 5: 30 minutes
- 6: 25 minutes
- 7: 20 minutes

cc_cache *Manage cached CCAFS files*

Description

Manage cached CCAFS files

Usage

```
cc_cache_list()

cc_cache_delete(files, force = TRUE)

cc_cache_delete_all(force = TRUE)

cc_cache_details(files = NULL)
```

Arguments

files	(character) one or more complete file names
force	(logical) Should files be force deleted? Default: TRUE

Details

cache_delete only accepts 1 file name, while cache_delete_all doesn't accept any names, but deletes all files. For deleting many specific files, use cache_delete in a lapply() type call

We cache using rappdirs::user_cache_dir(), find your cache folder by executing rappdirs::user_cache_dir("ccafs'

Functions

- cc_cache_list() returns a character vector of full path file names
- cc_cache_delete() deletes one or more files, returns nothing
- cc_cache_delete_all() delete all files, returns nothing
- cc_cache_details() prints file name and file size for each file, supply with one or more files, or no files (and get details for all available)

Examples

```
## Not run:
# list files in cache
cc_cache_list()

# List info for single files
cc_cache_details(files = cc_cache_list()[1])
cc_cache_details(files = cc_cache_list()[2])

# List info for all files
```

```
cc_cache_details()

# delete files by name in cache
# cc_cache_delete(files = cc_cache_list()[1])

# delete all files in cache
# cc_cache_delete_all()

## End(Not run)
```

`cc_data_fetch` *Download CCAFS data*

Description

Download CCAFS data

Usage

```
cc_data_fetch(key, overwrite = FALSE, ...)
```

Arguments

`key`	(character) a character string specifying a S3 key or a URL (the output from a call to `cc_search()`. the key can have spaces and newlines, which are removed internally - this allows keys to break across lines as keys can be very long
`overwrite`	(logical) Whether to overwrite files if they already exist on your machine. Default: `FALSE`
`...`	Curl options passed on to crul::verb-GET

Details

Note that data is not read into R as data can be very large. See `cc_data_read()`

Look in `rappdirs::user_cache_dir("ccafs")` for what files are cached and to delete any.

Note that we've made it so that you can index into the return object, getting either one or many results and the S3 class will be retained, so that you can pass the result down to `cc_data_read()`

Value

A character vector of full file paths. A print method makes a tidy return object in an S3 class.

Examples

```
## Not run:
key <- "ccafs/ccafs-climate/data/ipcc_5ar_ciat_downscaled/rcp2_6/
 2030s/bcc_csm1_1_m/10min/
 bcc_csm1_1_m_rcp2_6_2030s_prec_10min_r1i1p1_no_tile_asc.zip"

(res <- cc_data_fetch(key = key))
# indexing maintains class for easier subsetting
res[1]
res[[1]]
res[1:2]

res <- cc_list_keys()
zips <- grep("\\.zip", res$Key, value = TRUE)
x <- cc_data_fetch(zips[1])
unclass(x)
cc_data_read(x[1])
cc_data_read(x[1:3])
cc_data_read(x)

library(raster)
plot(cc_data_read(x[1]))
plot(cc_data_read(x[1:3]))

## End(Not run)
```

`cc_data_read` *Read CCAFS data*

Description

Read CCAFS data

Usage

```
cc_data_read(x, unreadable = "filter")
```

Arguments

`x`	A `ccafs_files` object, the output from a call to `cc_data_fetch()`
`unreadable`	(character) what to do when unreadable files are passed in. default is to filter them out and proceed ("filter") - alternatively, you can choose "stop", in which case we'll stop with a message.

Details

Look in `rappdirs::user_cache_dir("ccafs")` for what files are cached and to delete any.

`cc_data_fetch()` downloads data to your machine, and this function reads the data into your R session.

For more control over vizualizations of raster data, check out the `rasterVis` package (https://CRAN.R-project.org/package=rasterVis)

Value

`RasterLayer` or `RasterStack` class object. See their help files in `raster` package documentation.

Examples

```
## Not run:
key <- "ccafs/ccafs-climate/data/ipcc_5ar_ciat_downscaled/rcp2_6/
 2030s/bcc_csm1_1_m/10min/
 bcc_csm1_1_m_rcp2_6_2030s_prec_10min_r1i1p1_no_tile_asc.zip"
res <- cc_data_fetch(key = key)

# a single file
cc_data_read(res[1])

# select individual files
cc_data_read(res[1:2])

# all files
cc_data_read(res)

# character path input
## you can also pass in a path to a file(s)
cc_data_read(unclass(res[1]))

# plot data
library(raster)
plot(cc_data_read(res[1:3]))

## End(Not run)
```

`cc_list_keys` *List CCAFS keys*

Description

List CCAFS keys

Usage

```
cc_list_keys(prefix = NULL, delimiter = NULL, max = 1000, marker = NULL, ...)
```

Arguments

prefix	(character) string that limits the response to keys that begin with the specified prefix. the string can have spaces and newlines, which are removed internally - this allows the prefix to break across lines as prefixes can be very long
delimiter	(character) string used to group keys. Read the AWS doc for more detail.
max	(integer) number indicating the maximum number of keys to return (max 1000).
marker	(character) string that specifies the key to start with when listing objects in a bucket. Amazon S3 returns object keys in alphabetical order, starting with key after the marker in order.
...	Curl options passed on to crul::verb-GET

Details

This function lists keys from the CCAFS Amazon S3 bucket. Keys are essentially file paths. You can request data from any key that is a file (with a file extension, and has size > 0). Other keys are directories.

Value

A tibble (a data.frame, basically), with the columms:

- Key - object key
- LastModified - Object creation date or the last modified date, whichever is the latest.
- ETag - "entity tag", used for cache validation
- Size - Size of the object, in bytes, divide by 10^6 to get mb (megabytes)
- StorageClass - ignore, just useful for CCAFS maintainers

Examples

```
## Not run:
cc_list_keys(max = 1)

cc_list_keys()
cc_list_keys(max = 10)
cc_list_keys(prefix = "ccafs/ccafs-climate/data/ipcc_5ar_ciat_downscaled/")
cc_list_keys(prefix = "ccafs/ccafs-climate/data/ipcc_5ar_ciat_downscaled/
rcp2_6/2030s/bcc_csm1_1/10min/")

## End(Not run)
```

`cc_search` *Search CCAFS data*

Description

Search CCAFS data

Usage

```
cc_search(
  file_set = NULL,
  scenario = NULL,
  model = NULL,
  extent = NULL,
  format = NULL,
  period = NULL,
  variable = NULL,
  resolution = NULL,
  tile = NULL
)
```

Arguments

`file_set`	(integer) a file set, 2 through 12
`scenario`	(integer) a scenario, 1 through 10
`model`	(integer) a model, 1 through 89
`extent`	(character) an extent, 'global' or 'region'
`format`	(character) a format, 'ascii' or 'esri'
`period`	(integer) a period, 1 through 10
`variable`	(integer) a variable, 1 through 7, or 9999
`resolution`	(integer) a resolutions, 1 through 7
`tile`	(character) a tile defining a spatial area on the globe. one of A1-6, B1-6, or C1-6. See web interface for where those are located.

Details

See ccafs-search for details on parameters.

note that some URLs will be for Amazon S3 and others will have different base URLS (e.g., http://gisweb.ciat.cgiar.org)

Output can be passed to `cc_data_fetch()`, and subsequently to `cc_data_read()`

Value

character strings, one or more urls

Examples

```
## Not run:
(res <- cc_search(file_set = 12, extent = "global", format = "ascii",
  period = 4, variable = 1, resolution = 4))

res <- cc_search(file_set = 7, extent = "region", format = "ascii",
  period = 9, variable = 5, resolution = 6)
cc_data_fetch(res[3])

# Alternatively, you can use the helper list
# where you can reference options by name
# the downside is that this is very verbose
(res <- cc_search(file_set = cc_params$file_set$`Delta method IPCC AR4`,
                  scenario = cc_params$scenario$`SRES B1`,
                  model = cc_params$model$bccr_bcm2_0,
                  extent = cc_params$extent$global,
                  format = cc_params$format$ascii,
                  period = cc_params$period$`2040s`,
                  variable = cc_params$variable$Precipitation,
                  resolution = cc_params$resolution$`5 minutes`))

## End(Not run)
```

Index

∗ **datasets**
ccafs-search, 3
∗ **package**
ccafs-package, 2

cc_cache, 8
cc_cache_delete *(*cc_cache*)*, 8
cc_cache_delete_all *(*cc_cache*)*, 8
cc_cache_details *(*cc_cache*)*, 8
cc_cache_list *(*cc_cache*)*, 8
cc_data_fetch, 9
cc_data_fetch(), *10, 11*, *13*
cc_data_read, 10
cc_data_read(), *9*, *13*
cc_list_keys, 11
cc_params *(*ccafs-search*)*, 3
cc_search, 13
cc_search(), *4*, *9*
ccafs *(*ccafs-package*)*, 2
ccafs-package, 2
ccafs-search, 3, *13*
crul::verb-GET, *9*, *12*

lapply(), *8*

rappdirs::user_cache_dir(), *8*