Package 'CHCN'

February 19, 2015

Type Package

Title Canadian Historical Climate Network

Version 1.5

Date 2012-06-07

Author Steven Mosher

Maintainer Steven Mosher <moshersteven@gmail.com>

Depends R (>= 2.11.0),methods,bitops, RCurl

Description A compilation of historical through contemporary climate measurements scraped from the Environment Canada Website Including tools for scraping data, creating metadata and formating temperature files.

License GPL (>= 2)

URL http://stevemosher.wordpress.com/

LazyLoad yes

LazyData FALSE

Repository CRAN

Date/Publication 2012-06-08 18:23:46

NeedsCompilation no

R topics documented:

`CHCN-package` *A package for Building Historical Climate data for Canada*

Description

The package provides tools for scraping climate data from the Environment Canada website and turning it into R objects that can easily be used by analysts. The functions provided allow the user to download a master list of all the data at Environment Canada and to build a collection of monthly climate data from those resources. The master list is downloaded and then the stations which provide monthly data is extracted. Those station Ids are used to make http requests and the data is downloaded as csv files. There are function provided to parse the local csv files and create station "Inventories" with limited metadata (station name, latitude, longitude, etc). In addition, functions for selecting and compiling temperature data are provided. Other data, such as snowfall, rainfall, can also be easily extracted with simple R commands. Data can be reformated into a format similar to that provided by NOAA's GHCN (Global Historical Climate Network) and integration with the package RghcnV3 is trivial.

Details

Package:	CHCN
Type:	Package
Version:	1.5
Date:	2012-06-07
License:	GPL (>= 2)
LazyLoad:	yes
LazyData:	FALSE

key functions and building the database

The process of building the database is shown in the demo files. The process starts by downloading the master station list. The format of this list has changed somewhat since the first publication of this code, so version prior to 1.3 will be broken by that change. The first call to make is `downloadMaster()` That call creates a master list of all stations and their web ids. If that download fails there is a backup version of the file shipped with the package. It can be read using the function `readLocalMaster`. See that documentation for directions. Next, we create a list of monthly stations from that master list using `Stations <- writeMonthlyStations()` This writes a file of the stations that report monthly. Then we scrape the website: `scrapeToCsv(Stations)`, passing that function our list of monthly stations. When this process completes we check for missing or empty csv files: `EMPTY <- getEmptyCsv()`. If the list is null then we have no empty downloads. Checking for missing downloads is accomplished by `getMissingScrape`. Assuming that all files are downloaded, then we can proceed to create datasets and inventories. `data <- createDataset()` will create a dataset by reading all the csv files. As there is more than temperature data here, we want to save it all: `writeData(data)`. Next, we create an inventory of all the stations: `inv  <- createInventory()`. This inventory can be saved with a simple `write.table`. When we save this we preserve all the orginal metadata in the format we downloaded it in: `write.table(inv,"masterInventory.inv")`. Next for working with other packages (like RghcnV3) we want to save data in a friendly format: `writeInventory(inv)` will write a GHCN style inventory with variables named appropriately and put into the correct columns for RghcnV3. Lastly, we want to select certain data from the master datafile and write it out. To select Tmean we do the following `Mean <- formatGhcn(data, dataColumn = 7)`. Data column 7 (use colnames(data) to see the entire list of options).

lastly, we write out the data in ghcn format `writeGhcn(data, directory = DATA.DIRECTORY, filename = "TaveCHCN.da`

Author(s)

Steven Mosher

Maintainer: Steven Mosher<moshersteven@gmail.com>

References

http://climate.weatheroffice.gc.ca

Examples

```
## Not run:
     downloadMaster()
     Stations <- writeMonthlyStations()
     scrapeToCsv(Stations)
     EMPTY <- getEmptyCsv()
     if (is.null(EMPTY)){
         data <- createDataset()
         ###  save all the data
         writeData(data)
         inv  <- createInventory()
         write.table(inv,"masterInventory.inv")
         # write a ghcn style inventory
         writeInventory(inv)
         # select Tave data
```

```
          Mean <- formatGhcn(data, dataColumn = 7)
          writeGhcn(Mean)

} else{
  scrapeToCsv(EMPTY)
}


## End(Not run)
```

BASE.URL *The "base" elements of the url for scraping*

Description

The http request is made to Enviroment Canada using a URL that consists of a "base" element, a station Id, a year and a format.

Usage

```
BASE.URL
```

Format

The format is: chr "http://climate.weatheroffice.gc.ca/climateData/ bulkdata_e.html?timeframe=3&Prov=XX&StationID="

Details

The base is used to construct the final http request. The various elements are pasted together.

Source

http://climate.weatheroffice.gc.ca

References

The request format was passed from Enviroment Canada via email and verified by performing sample downloads.

`createDataset` *A function to create a dataset from csv files on disk*

Description

After files have been scraped to disk they have to processed from cvs files into proper R objects. The first step is to create and inventory and then to create datasets. This function creates datasets. Every csv file has 25 parameters. Creating the entire dataset makes a 350Mb file and takes a long time to process. In the end you have a dataset that conatins all the monthly data from Environment canada.

Usage

```
createDataset(Ids = NULL, filename = MONTHLY.STATION.LIST, directory = "EnvCanada")
```

Arguments

`Ids`	a sequence of unique station Ids. Ids start at 99111111. They are present as variables in both the file names and the inventories. Ids defaults to NULL. If this is not changed by the caller then all station Ids are used to create the dataset. Alternatively, one can create a subset of all the data by subseting the inventory and working with the Ids from that inventory. For example, one could create datasets for every province or for lat/lon combinations.
`filename`	This is the filename of the master list of monthly stations. that file is read in get a list of all unique Ids
`directory`	The directory defaults to EnvCanada where all the csv files are. The csv files all have unique names that are tied to the unique Ids. Given the vector of Ids provided by the caller, and the list of files available, the function then reads in those files to generate a data structure.

Value

The function returns a dataframe of 27 variables, including Ids, climate data, and the file that was used to create the data. This data can then be written out by R's write commands. You can also pass this data through `formatGhcn` and create datasets that can be read by RghcnV3

Author(s)

Steven Mosher

Examples

```
## Not run:

 data <-createDataset(Ids = STARTING.STATION.ID:(STARTING.STATION.ID + 5))
 data <-createDataset() # compiles a complete dataset of all files
```

```
## End(Not run)
```

`createInventory` *Creates an inventory of all stations in the directory*

Description

After all the files have been scraped to EnvCanada the next step is to create a master inventory of every station. The cvs files have 7 lines of metadata which is read by this function and formated into a R data structure for saving

Usage

```
createInventory(directory = "EnvCanada")
```

Arguments

`directory`	Defaults to EnvCanada where all scrapes are written to.

Details

As they are scraped the files are given a unique identifier which is contained in the monthly station list. That identifier is a part of the filename is it saved under. The directory is searched for files matching a specific name pattern. Those files are read in and the metadata is collected and formated into a dataframe

Value

the function returns a dataframe conatining all the metadata of the stations in the directory. this should be saved as a file.

Author(s)

Steven Mosher

Examples

```
## Not run:

  Inv <- createInvetory()
  write.table(Inv,"masterInventory.inv")


## End(Not run)
```

DATA.DIRECTORY *Directory for the final data storage*

Description

Finalized inventories and data files are written here

Usage

```
DATA.DIRECTORY
```

Format

The format is: chr "DataDirectory"

Details

When data is processed from the web or local csv files files are created and put in this directory

downloadMaster *Download the master list of stations*

Description

This function simply downloads the master list and gives it a name that other functions rely on

Usage

```
downloadMaster(url = STATION.URL, localFile = MASTER.STATION.LIST)
```

Arguments

`url`	the url for the csv file that contains all the stations
`localFile`	A local filename to write this data to

Details

The function downloads the output of a scraper written by DrJ of clearclimatecode. That scraper is found here: `http://scraperwiki.com/scrapers/can-weather-stations/` It scrapes the main page at Enviroment Canada and deposits a csv file containing all the information needed to data scrape

Value

The function downloads a csv file and writes a local version

Author(s)

Steven Mosher

References

http://scraperwiki.com/scrapers/can-weather-stations/

FORMAT.URL *A string for completing the http request*

Description

htpp requests for data are made by combining 5 strings: a base string, a year string, a station webid, a year, and a format string

Usage

```
FORMAT.URL
```

Format

The format is: chr "&Month=1&Day=1&format=csv"

Details

A string to specify that we want monthly data in a csv file

Source

http://climate.weatheroffice.gc.ca

formatGhcn *creates a 14 column dataset for output*

Description

The entire dataset of Environment Canada has 25 columns of data: Tmax,Tmin,Tave, quality flags and other climate data. For use with the package RghcnV3 a 14 column format is required. This format is similar to the 14 column format of GHCN V2. Id is in the first column, followed by Year, followed by 12 measures, one for each month of the year in column 2. This function takes a column and creates a dataset in that format

Usage

```
formatGhcn(data, dataColumn = 7)
```

Arguments

`data`	A dataset that has been created by `createDataset`
`dataColumn`	The column of the data you want to reformat. column = 7, will collate the Mean air temperature from a full 25 column dataset. Use `colnames(data)` to determine which column you want to format

Details

GHCN data is in 14 column or 51 column format. Every row of data has an Id, a year and 12 months of data. This function takes the datasets of this package and creates formats that are readable by the package `RghcnV3`. The function `createDataset` is used to read from csv files and create a complete dataset with 25 columns of data. Typically you want to work with one data item at a time. This function allows you to extract and reformat a column of that data

Value

Returns a 14 column dataframe

Author(s)

Steven Mosher

See Also

`createDataset`

Examples

```
 ## Not run:
   data <-createDataset()
   Mean <-formatGhcn(data)
   writeGhcn(Mean)

## End(Not run)
```

`getEmptyCsv` *A function to check for empty Csv files*

Description

Before creating inventories or datasets the EnvCanada directory should be checked for completeness. Sometimes the scraping process will result in empty csv files. This function checks for empty files and returns a set of indices for re-scraping the empty files

Usage

```
getEmptyCsv(directory = "EnvCanada", Monthly = MONTHLY.STATION.LIST)
```

Arguments

`directory`	The directory where the csv files live
`Monthly`	The name of the cvs file with the list of stations that report monthly

Details

The function uses `file.info` and `list.files` to find csv files with zero size. Then it seatches the list of monthly stations to find the index (1-7676) of that station Id and returns the set of indices for files that have zero length. Thus the function `scrapeToCsv` can then be called with the results of `getEmptyCsv`

Value

A vector of indicies for empty files OR NULL

Author(s)

Steven Mosher

Examples

```
 ## Not run:
   x<-getEmptyCsv()
   # check if x is non null
   scrapeToCsv(get = x)


## End(Not run)
```

`getMissingScrape` *A function to find which scrapes need to be done*

Description

During the course of scraping the server sometimes loses the connection or returns an empty file. When that happens the `scrapeToCsv` should be run again to complete the scrape. That function takes a sequence of files that have not been downloaded. To figure out the missing files, call `getMissingScrape`

Usage

```
getMissingScrape(monthlyList = MONTHLY.STATION.LIST, directory = "EnvCanada")
```

Arguments

`monthlyList`	The file name of the list of stations that report monthly. This is created by calling `createMonthlyStations`
`directory`	the default directory where scrapes are stored. EnvCanada.

Details

The monthly list of stations contains all the stations that report monthly, numbered from 99111111 upwards. As files are scraped then are downloaded to EnvCanada. If the scrape should fail, or if you want to do it in bits and peices you can find out which files are missing by comparing the master station list with the directory of EnvCanada. This function makes that easy and uses the file names to determin which elements of the monthly station list are missing.

Value

the function returns a sequence of integers that are references into the monthly station list. That list has 7676 stations. If elements 52,78,954, and 3215 do not have their associated files in envCanada, then those files can be scraped by calling `scrapeToCsv` and passing the sequence of elements to that function

Author(s)

Steven Mosher

Examples

```
 ## Not run:
   missing <- getMissingScrape()
   scrapeToCsv(Stations,get=missing)

## End(Not run)
```

`MASTER.STATION.LIST` *Name for the local version of the master station list*

Description

defines the name of the local version of the master list

Usage

```
MASTER.STATION.LIST
```

Format

The format is: chr "EnvCanadaMaster.csv"

Details

Name of the local file. used in other functions

`MONTHLY.STATION.LIST` *Name of the local file for Monthly stations*

Description

This file will contain a subset of master list. feilds have been edited to make scraping easier. Only stations with monthly data are in the list

Usage

```
MONTHLY.STATION.LIST
```

Format

The format is: chr "MonthlyStations.Env.csv"

Details

The name used for the local version of the stations that report monthly

`readLocalMaster` *A function to read a local copy of the master csv file*

Description

The Process of building the dataset depends upon downloading a csv file that lists all the data on environment canada. If for some reason the function `downloadMaster` does not function, the file can be read from a local copy shipped with the package. This version should be current to the last release date of the package.

Usage

```
readLocalMaster()
```

Details

The function reads a stored copy of EnvCanadaMaster.csv that ships with the package. After reading the file you should write it out to your working directory. The following call will read the local copy and write it out to your working directory. write.csv(readLocalMaster(),MASTER.STATION.LIST)

Value

the function returns a data.frame that contains the information neeed to create a master list of monthly stations (as opposed to daily stations and hourly stations which are also in the master list) That data.frame should be written as a csv file to the working directory It must be given the name EnvCanadaMaster.csv . The constant MASTER.STATION.LIST is predefined to this strings value

Author(s)

Steven Mosher

`readMonthlyCsv` *reads the csv file that contains all the monthly stations*

Description

A simple function that wraps a `read.csv` call to read the stations in the monthly station list

Usage

```
readMonthlyCsv(filename = MONTHLY.STATION.LIST)
```

Arguments

`filename`	Default name oof the file. Should not be changed

Details

simplys wraps a `read.csv` call

Value

returns a data frame of the stations that report monthly

Author(s)

Steven Mosher

Examples

```
 ## Not run:
   Stations <- readMonthlyCsv()

## End(Not run)
```

scrapeToCsv *A function to scrape files to local csv files*

Description

This function uses the monthly list of stations and downloads them to a local directory. There are 7676 files as of July 2011. The function throws warnings about wrong files sizes. These can be ignored or suppressed by setting warning options

Usage

```
scrapeToCsv(Stations, get = seq(from = 1, to = 1e+05), directory = "EnvCanada")
```

Arguments

`Stations`	A data structure returned from `readMonthlyStations` If the monthly station file already exists, it can simply be read from disk with `read.csv`
`get`	get is assigned to a sequence of numbers that is used to index the monthly station list. It defaults to 1:100000. This results in the function trying to download all 7676 files from Env Canada. Alternatively, one can download the files in chunks, for example setting `get` to 1:1000, or any other sequence of numbers. Internal checking ensures that the sequence sought is available for download. Irregular sequences are also supported: `get = c( 23,65,257,7000)` would get those elements from the list of stations in monthly.env.csv
`directory`	The local directory to write the csv files to. "EnvCanada"

Details

When `createMonthlyStations` is executed the master list is parsed and only those stations that report monthly are copied into a file. The file contains a web Id that is used when downloading. To scrape the files in the monthly data structure youc all `scrapeToCsv` and provide a sequence of stations you want to download. The download will occasionally fail for server timeouts. By using the function `getMissingScrapes` you can determine which files are missing from the directory. So if you try to download all 7676 files and the server times out after 2365, the function `getMissingScrapes` will provide a sequence of files to be downloaded to complete your scrape.

Value

function downloads files according to the sequence of values in the "get" parameter.

Author(s)

Steven Mosher

See Also

`getMissingScrape`

Examples

```
## Not run:
   Stations <- writeMonthlyStations()
   scrapeToCsv(Stations,get=1:100)
   scrapeToCsv(Stations,get=100:2075)


## End(Not run)
```

`STARTING.STATION.ID` *The starting unique Id created for these stations*

Description

Stations are numbered sequentially when they are taken out of the master list. Obviously if the master list changes this unquie Id will change. It is used only for keeping track of files during processing. It is not an officially recognized Id and should not be confused with WMo numbers or the like

Usage

```
STARTING.STATION.ID
```

Format

The format is: num 99111111

Details

Stations are numbered sequentially

Examples

```
data(STARTING.STATION.ID)
print(STARTING.STATION.ID)
```

`STATION.URL` *url for getting master station list*

Description

url to get csv file of master list

Usage

```
STATION.URL
```

Format

The format is: chr "http://scraperwiki.com/api/1.0/datastore/ sqlite?format=csv&name=can-weather-stations&query=select+*+from+‘swdata‘"

Details

Output of the scraper written by DrJ of clearclimatecode.org

Source

http://scraperwiki.com/scrapers/can-weather-stations/

`writeData` *Writes a file from a dataframe of climate data*

Description

A function that simply writes the data in a dataframe to the data directory. Typically this would be your master data file containing all stations and all data

Usage

```
writeData(Data, filename = "EnvCanadaData.dat", directory = DATA.DIRECTORY)
```

Arguments

`Data`	The data created by scraping and processing all the web page requests
`filename`	Default file name for your master data file
`directory`	Default directory for your data

Details

Function merely wraps a `write.csv` call and creates a data directory if you need one

Value

Side efffect is a file is written

Author(s)

Steven Mosher

Examples

```
 ## Not run:
  writeData(data)

## End(Not run)
```

`writeGhcn` *A simple wrapper to* `write.table`

Description

Simply writes a file to the data directory using `write.table`

Usage

```
writeGhcn(data, directory = DATA.DIRECTORY, filename = "TaveCHCN.dat")
```

Arguments

`data`	The data you want to write
`directory`	defaults to the processed data directory
`filename`	The filename you want for the data

Details

Simply uses `write.table` to write the data

Value

side effect is a file is written

Author(s)

Steven Mosher

Examples

```
## Not run:

  writeGhcn(data)

## End(Not run)
```

`writeInventory` *Writes inventory data in a format usable by RghcnV3 package*

Description

The RGhcnV3 package expects certain fields – Id, lat,lon in a specific order. This write function insures the data is read in a compatible manner with that package

Usage

```
writeInventory(Inventory, filename = "Inventory.inv", directory = DATA.DIRECTORY)
```

Arguments

`Inventory`	A inventory of stations
`filename`	A default name
`directory`	the default data directory

Details

the function merely reorders the columns so that id is in the first column followed by latitude, and longitude

Value

The side effect is writing a file

Author(s)

Steven Mosher

`writeMonthlyStations` *Function to read the master list and write a monthly list of stations. This file is critical to the operation of the scraper.*

Description

the function reads the master list and writes a local list of stations that report monthly data. It also returns an object dataframe. That dataframe can be fed to downstream processes

Usage

```
writeMonthlyStations(filename = MASTER.STATION.LIST,
                                 outfile =   MONTHLY.STATION.LIST)
```

Arguments

`filename`	The filename of the master list. This is set to a default that should not be changed.
`outfile`	the local filename. Used by other functions. It should not be changed

Details

reads the master list. Extracts those stations that report monthly. Selects the first year reporting and the web Id for the http request builder. Assigns an Id to every station for file naming and tracking. The id it assigns is used for tracking the scrape progress and recovering from scrape failures.

Value

returns a dataframe and writes that dataframe to disk. can be read with `read.csv`

Author(s)

Steven Mosher

Examples

```
 ## Not run:
   Stations <- writeMonthlyStations()

## End(Not run)
```

`YEAR.URL` *A string for making the http request*

Description

1 of 5 strings concatenated to make the final request

Usage

```
YEAR.URL
```

Format

The format is: chr "&Year="

Details

used to make the http request

Index