APPENDIX AVAILABLE ON REQUEST

Communication 12

Internet-Based Health and Air Pollution Surveillance System

Zeger et al

Appendix B NMMAPSdata R Package

Correspondence may be addressed to **Dr Scott L Zeger, Department of Biostatistics, Johns Hopkins University Bloomberg School of Public Health, 615 North Wolfe Street, Room E3132, Baltimore MD 21205-2179.**

Although this document was produced with partial funding by the United States Environmental Protection Agency under Assistance Award R82811201 to the Health Effects Institute, it has not been subjected to the Agency's peer and administrative review and therefore may not necessarily reflect the views of the Agency, and no official endorsement by it should be inferred. The contents of this document also have not been reviewed by private party institutions, including those that support the Health Effects Institute; therefore, it may not reflect the views or policies of these parties, and no endorsement by them should be inferred.

This document was reviewed by some members of the HEI Health Research and Review Committees. It did not undergo the HEI scientific editing and production process.

Package 'NMMAPSdata'

October 5, 2004

Title NMMAPS Data

Depends R (>= 2.0.0), methods

Date 2004-10-05

Version 0.4

LazyLoad yes

Author Roger D. Peng <rpeng@jhsph.edu>, Leah J. Welty, Aidan McDermott

Maintainer Roger D. Peng <rpeng@jhsph.edu>

Description Daily mortality, weather, and air pollution data from the National Morbidity, Mortality, and Air Pollution Study for 108 U.S. cities (1987–2000)

URL http://www.ihapss.jhsph.edu/data/NMMAPS/R/

R topics documented:

NMMAPS *Overview of NMMAPS database*

Description

The NMMAPS database: Daily mortality, weather, and pollution data for 1987–2000.

The database contains dataframes with air pollution, weather, and mortality data for 108 United States cities. Each city dataframe contains daily time series of mortality counts (for various causes of death), pollution levels, and weather (e.g. temperature and humidity).

The data were assembled from publicly available data sources as part of the National Morbidity, Mortality, and Air Pollution Study (NMMAPS) sponsored by the Health Effects Institute. Daily mortality counts were obtained from the National Center for Health Statistics and classified into three age categories (< 65; 65-74; >= 75). Accidental deaths (i.e. ICD-9 >= 800) were excluded. Weather data were obtained from the National Climatic Data Center EarthInfo CD-ROM and pollution data were obtained from the Environmental Protection Agency's Aerometric Information Retrieval System (AIRS) and AirData System.

Note that the data included with this package only contain the daily counts of mortality. Morbidity outcomes (e.g. hospital admissions, etc.) are not included with this package.

Format

Each city dataframe has 15342 rows and 291 columns.

Note that the pollution and weather data are replicated 3 times because the mortality counts are split into three age categories. There are only 5114 *days* of observations, but each day has three mortality counts. The dataframes are stored in this format so that they can be used immediately with a regression procedure such as `lm` or `glm`. If age category stratified counts are not needed, one can use the `collapseEndpoints` preprocessing function to combine the outcomes across age categories.

Pollution variables have been de-trended and their names all have the suffix *tmean (which stands for "trimmed mean"). Briefly, the pollution value for a particular day in a given city is the 10% trimmed mean of the values from all of the monitors in the given city.

Each pollutant also has variable whose name has the suffix *mtrend. This variable stores the median trend of the pollution monitors for a given city. This roughly the value that is subtracted off the original pollution series to center it around zero. See the `PollutantProcess` vignette for more details.

The iHAPSS website (`http://www.ihapss.jhsph.edu/`) contains detailed information on the different variables included with each city dataframe.

References

Samet JM, Dominici F, Zeger SL, Schwartz J, Dockery DW (2000). *The National Morbidity, Mortality, and Air Pollution Study, Part I: Methods and Methodologic Issues*. Research Report 94, Health Effects Institute, Cambridge MA.

`http://www.healtheffects.org/Pubs/Samet.pdf`

Samet JM, Zeger SL, Dominici F, Curriero F, Coursac I, Dockery DM, Schwartz J, Zanobetti A (2000). *The National Morbidity, Mortality, and Air Pollution Study, Part II: Morbidity and Mortality from Air Pollution in the United States*. Research Report 94, Health Effects Institute, Cambridge MA.

http://www.healtheffects.org/Pubs/Samet2.pdf

Dominici F, McDermott A, Daniels M, Zeger SL, Samet JM (2003). "Mortality among residents of 90 cities," in *Revised Analyses of Time-Series Studies of Air Pollution and Health*. Research Report 94, Health Effects Institute, Cambridge MA, pp. 9-24.

http://www.healtheffects.org/Pubs/TimeSeries.pdf

Daniels MJ, Dominici F, Zeger SL, Samet JM (2004). *The National Morbidity, Mortality, and Air Pollution Study, Part III: Concentration-Response Curves and Thresholds for the 20 Largest US Cities*. Health Effects Institute, Cambridge MA.

http://www.healtheffects.org/Pubs/Daniels94-3.pdf

Internet-based Health and Air Pollution Surveillance System (iHAPSS):

http://www.ihapss.jhsph.edu/

NMMAPSdata announcements mailing list:

http://groups-beta.google.com/group/NMMAPSdata-announce

`NMMAPScite` *Citing NMMAPSdata in Publications*

Description

How to cite NMMAPSdata in publications.

Usage

```
NMMAPScite()
```

Details

Execute the function `NMMAPScite()` for information on how to cite NMMAPSdata in publications.

Note

The same citation information can be obtained by using the `citation` function. `NMMAPScite` will likely be removed in future releases.

`NMMAPSdata-internal`
Internal functions for NMMAPSdata

Description

Internal functions for NMMAPSdata

Usage

```
setDBPath(fullpath)
```

Arguments

`fullpath`	character, full path to the database directory

Details

This function sets the directory path for a database.

Value

Nothing useful is returned.

Author(s)

Roger D. Peng ⟨rpeng@jhsph.edu⟩

Examples

```
## None
```

`NMMAPSdbInfo-class` *Class "NMMAPSdbInfo"*

Description

Information about NMMAPS databases

Objects from the Class

Objects can be created by calls of the form `new("NMMAPSdbInfo", ...)`. However, users will generally not have any need to do this directly, but rather will call `buildDB`.

Slots

`cityList`: character vector of (abbreviated) names of cities contained in the database

`procFunc`: function used to preprocess the data

`dbPath`: character, full directory path for the database

`call`: the call to `buildDB` used to construct the database

`.Environment`: the associated environent of the processing function `procFunc`

Methods

rebuildDB `signature(object = "NMMAPSdbInfo")`: rebuild an NMMAPS database using information stored in the `NMMAPSdbInfo` object.

show `signature(object = "NMMAPSdbInfo")`: prints some summary information about the database

Author(s)

Roger D. Peng ⟨rpeng@jhsph.edu⟩

See Also

buildDB

Examples

```
## None right now
```

---

buildDB *Build a version of the NMMAPS database*

Description

Build a version of the NMMAPS database.

Usage

```
buildDB(procFunc, dbName,
        path = system.file("db", package = "NMMAPSdata"),
        cityList = NULL, compress = FALSE, verbose = TRUE, ...)
```

Arguments

procFunc	A function (or function name, quoted) for preprocessing a dataframe. See Details
dbName	character, a name for the new database
path	character, directory in which to create the new database
cityList	character, vector of abbreviated city names
compress	logical, should the new database be stored in a compressed format?
verbose	logical, should messages be printed as database is being constructed?
...	other arguments passed to procFunc

Details

buildDB can be used to make derivatives of the NMMAPS database from the raw data. If dbName is not specified, the name of the new database is taken from the name of the function procFunc. buildDB attempts to create the directory given by the value of dbName in the directory specified in path. The default for path is the directory db/ in the installation directory for the NMMAPSdata package.

Once the directory for the new database is created, buildDB cycles through each city dataframe and applies the function procFunc to each of them. The function procFunc should either return a modified dataframe or NULL. If procFunc returns a dataframe, then buildDB saves the dataframe in the new database directory. If procFunc returns NULL when processing a city dataframe, buildDB then skips that city and does not write out anything.

After building the database, buildDB calls registerDB to register the location of the new database.

Value

An object of class NMMAPSdbInfo. Currently, there is a show method for NMMAPSdbInfo. There is also an experimental function rebuildDB for rebuilding a database.

Warning

If `path` is a relative path, then changing the working directory (i.e. via `setwd`) will result in functions such as `loadCity`, etc. not finding the current database.

Author(s)

Roger D. Peng ⟨rpeng@jhsph.edu⟩

See Also

`registerDB`, `NMMAPSdbInfo-class`

Examples

```
## Build a database containing data for the 10 largest cities
data(cities)
cityList <- with(cities, city[order(pop, decreasing = TRUE)][1:10])
buildDB(procFunc = function(x) { x }, dbName = "Largest10", cityList = cityList)
```

`cityData` *U.S. Cities Data*

Description

Pollutant, meteorology, and mortality data for U.S. cities (1987–2000)

Details

Individual city data frames can be loaded using `loadCity`, `attachCity` or `readCity`.

Data for a particular city are loaded by using the abbreviated name for the city. The list of abbreviations can be found by using `listAllCities` or `listDBCities` for the currently registered database.

The city data frames are stored as compressed R workspace files with the compression done using the bzip2 algorithm to save space. The compression used is not the same as that used by `save(compress = TRUE)`.

The first column of each city dataframe contains the city's abbreviated name. This column is of class `character`. Every other column is of class `numeric`. Certain variables, such as `agecat` (age category) and `dow` (day of the week) may need to be coerced to class `factor` when fitting models.

Note

City dataframes cannot be loaded using the function `data`.

See Also

`attachCity`, `readCity`, and `loadCity` for how to load the data; `listAllCities`; `variableDesc` for short descriptions of the variables.

Examples

```
## Use the full/raw database
registerDB()

## Load the Seattle data frame
seattle <- readCity("seat")

## Attach Los Angeles to the search list
attachCity("la")
```

`cityApply` *Apply a function to cities*

Description

Apply a function to all city dataframes in the currently registered database.

Usage

```
cityApply(FUN, cityList = listDBCities(), verbose = FALSE,
          hookFun = NULL, wrapInTry = FALSE, ...)
```

Arguments

`FUN`	a function whose first argument is a dataframe
`cityList`	character vector containing (abbreviated) names of cities
`verbose`	logical, should the city names be printed?
`hookFun`	a function or a (named) list of functions to be applied to the object returned by `FUN`
`wrapInTry`	should the call to `FUN` be wrapped with a call to `try`? The default is `TRUE`.
`...`	other arguments for `FUN`

Details

This function is a simple wrapper for a call to `lapply` which loops through the cities in `cityList`, reads in each city dataframe one at a time, and calls the function `FUN` on the dataframe.

The argument `hookFun` can be used to pass a single function or a (named) list of functions to be called on the object returned by `FUN`. Often it will not be necessary to worry about `hookFun` argument since the output from `FUN` can just be collected and returned. However, it may be the case that `FUN` returns a large object (e.g. a fitted model object) and returning one such object for every city in `cityList` would be unwield. For example, one may be interested only in extracting coefficients and standard errors from a fitted model object.

If `wrapInTry = TRUE` then the call to `FUN` will be wrapped by a call to `try`. That way, if there is an error in `FUN`, for example, for a particular city, the other cities will still be processed.

Value

A named list of length `length(cityList)` where each element of the list contains the results of a calling `FUN` on a city dataframe. The names of the lists correspond to the names in `cityList`.

Note

Right now the `verbose` argument, if set to `TRUE` just prints the city names as they are processed by `FUN`.

Author(s)

Roger D. Peng ⟨rpeng@jhsph.edu⟩

Examples

```
buildDB(NMMAPSlite, dbName = "myDB", cityList = c("ny", "chic"))
stats <- function(x) { with(x, tapply(cvd, as.factor(agecat), mean)) }

## Calculate average daily CVD deaths by age category
cityApply(stats, verbose = TRUE)
```

`citycensus` *U.S. Census 2000 data*

Description

Data from the 2000 U.S. Census

Usage

```
data(citycensus)
```

Format

A data frame with 113 observations on the following 78 variables.

city Four letter city abbreviation

pop100 Population

hu100 Housing units

arealand Land Area

areawater Water Area

p006001 Race Denominator

p006002 White alone

p006003 Black or African American alone

p006004 American Indian and Alaska Native alone

p006005 Asian alone

p006006 Native Hawaiian and Other Pacific Islander alone

p006007 Some other race alone

p006008 Two or more races

p007001 Hispanic or Latino Denominator

p007002 Not Hispanic or Latino

p007010 Hispanic or Latino

hct007002 Owner occupied units
hct007001 Occupied housing units
hct007052 Renter Occupied units
p087001 Poverty Denominator
Purban Proportion Urban
Prural Proportion Rural
malel65 Males under 65
male65to75 Males [65,75)
male75p Males 75+
femalel65 Females under 65
female65to75 Females [65,75)
female75p Females 75+
Pdrive Proportion Drive to work
Ppublic Proportion Public transport to work
Pmalehigh Proportion males high school plus
Pmaledeg Proportion males degree plus
Pfemalehigh Proportion females high school plus
Pfemaledeg Proportion females degree plus
Phigh Proportion high school plus
Pdeg Proportion degree plus
Pmaleunem Proportion males unemployed
Pfemaleunem Proportion females unemployed
Punem Proportion unemployed
Ppovertyl5 Proportion poverty age <5 years
Ppovertye5 Proportion poverty age 5 years
Ppoverty6to11 Proportion poverty age 6-11 years
Ppoverty12to17 Proportion poverty age 12-17 years
Ppoverty18to64 Proportion poverty age 18-64 years
Ppoverty65to74 Proportion poverty age 65-74 years
Ppoverty75p Proportion poverty age 75+ years
Ppoverty Proportion poverty
Pdiffhouse Proportion in different house in 1995
Pdiffcounty Proportion in different county in 1995
Pdiffstate Proportion in different state in 1995
Powner Proportion owner households
Prented Proportion renter households
Powner65p Proportion owner households age 65+
Prented65p Proportion renter households age 65+
moved165p 65+ Moved in 99 to March 00
moved265p 65+ Moved in 95-98

moved365p 65+ Moved in 90-94

moved465p 65+ Moved in 90-89

moved565p 65+ Moved in 70-79

moved665p 65+ Moved before 70

Pmoved165p Proportion 65+ Moved in 99 to March 00

Pmoved265p Proportion 65+ Moved in 95-98

Pmoved365p Proportion 65+ Moved in 90-94

Pmoved465p Proportion 65+ Moved in 90-89

Pmoved565p Proportion 65+ Moved in 70-79

Pmoved665p Proportion 65+ Moved before 70

ownermoved165p 65+ owner moved in 99 to March 00

ownermoved265p 65+ owner moved in 95-98

ownermoved365p 65+ owner moved in 90-94

ownermoved465p 65+ owner moved in 90-89

ownermoved565p 65+ owner moved in 70-79

ownermoved665p 65+ owner moved before 70

rentermoved165p 65+ renter moved in 99 to March 00

rentermoved265p 65+ renter moved in 95-98

rentermoved365p 65+ renter moved in 90-94

rentermoved465p 65+ renter moved in 90-89

rentermoved565p 65+ renter moved in 70-79

rentermoved665p 65+ renter moved before 70

Details

Source

References

Examples

```
data(citycensus)
getVarDesc(citycensus)
```

`dataInput` *Load city data*

Description

Read/Attach city data from NMMAPS database

Usage

```
readCity(name)
attachCity(name, pos = 2)
loadCity(name, envir = parent.frame())
```

Arguments

`name`	the abbreviated name (quoted) of a city in the NMMAPS database
`pos`	the position on the search list at which to attach the city dataframe
`envir`	environment into which the city data should be loaded

Details

Each of these functions first obtains the location of the currently registered database via the function `getDBPath`. The database from which to read/load/attach should have already been registered using the function `registerDB`.

`readCity` returns a dataframe from the currently registered database containing the relevant data for the city specified in `name`.

`attachCity` loads the city's dataframe and attaches it the search list. The city data can subsequently be `detach`ed if desired. The variables in the attached dataframe can be accessed via their individual names (see the examples). `attachCity` may be more useful for interactive work.

`loadCity` loads the dataframe for a specified city into the user's workspace. The variable name of the dataframe will be the abbreviated city name.

Value

`readCity` returns a dataframe. `attachCity` attaches a dataframe to the search list and returns nothing useful. `loadCity` returns (invisibly) a character vector containing the name of the object that was loaded.

Warning

You cannot usefully attach more than one city dataframe at a time since all of the dataframes have the same variable names. If you attach more than one city dataframe (without first detaching), you will only be able to access the variables in the dataframe that was most recently attached.

Author(s)

Roger D. Peng ⟨rpeng@jhsph.edu⟩

See Also

`search`, `registerDB`

Examples

```
## Register full/raw database
registerDB()

## Load Los Angeles city data (use the default raw data)
LAdata <- readCity("la")

## Mean daily non-accidental deaths by age category
tapply(LAdata$death, as.factor(LAdata$agecat), mean)

## Attach New York data to the search list
attachCity("ny")
search()

## Median daily non-accidental deaths by age category
## Variables can be accessed directly
tapply(death, agecat, median)

## Load the Pittsburgh data into the user's workspace
loadCity("pitt")

## Compute a 2-day running mean for PM10
pitt <- subset(pitt, agecat == 3) ## Only use people > 75 years old
rm2pm10 <- filter(pitt$pm10tmean, filter = c(1/2, 1/2), sides = 1)
```

---

utils *Utilities for NMMAPS database*

Description

Utility functions for managing NMMAPS database and derivatives

Usage

```
getDBInfo()
showDB()
getDBPath()
clearDB(dbName)
registerDB(dbName = NULL,
           path = system.file("db", package = "NMMAPSdata"))
listDBCities(full.names = FALSE)
```

Arguments

dbName	character, the name of the database
path	character, the path to the directory containing the database
full.names	logical, should full city names be returned?

Details

`getDBInfo` returns an object of class `NMMAPSdbInfo` which is stored with the currently registered database.

`showDB` prints to the screen the path the the currently registered database (unless the full/raw database is being used, in which case a simple message is printed).

`getDBPath` returns a character string containing the full path to the currently registered database. If the full/raw database is being used, `getDBPath` returns `NULL`.

`clearDB` removes the database specified in `dbName` from the system installation directory (if one has been built with `buildDB`). It cannot be used to remove databases installed in other (user-specified) locations.

`registerDB` registers the location of (and makes currently active) the database specified in `dbName`. Calling `registerDB` with the default arguments registers the full/raw database.

`listDBCities` returns a character vector containing the names of the cities included in the currently registered database. If `full.names = TRUE` then the full city names are returned. Otherwise, the abbreviated names are returned.

Value

`getDBInfo` returns an object of class `NMMAPSdbInfo`. `getDBPath` returns a character object containing the path to the currently registered database. The other functions do not return anything useful.

Author(s)

Roger D. Peng ⟨rpeng@jhsph.edu⟩

Examples

```
## Build a simple database
buildDB(procFunc = function(x) { x[, c("death", "pm10tmean")] },
        dbName = "simpleDB", cityList = c("det", "minn"))
registerDB("simpleDB")
dbInfo <- getDBInfo()
show(dbInfo)
getDBPath()
showDB()

## Remove database from package directory
clearDB("simpleDB")
```

---

descriptives *Descriptives for NMMAPS data*

Description

Descriptive information for NMMAPS data.

Usage

```
data(agecat)
data(variables)
data(tables)
data(cities)
data(regions)
data(counties)
data(dow)
data(latlong)
```

Format

`agecat`: Description of the three age categories used in NMMAPS.

`dow`: Day of the week indicators.

`regions`: Region abbreviations and full names.

`variables`: Variable names and descriptions.

`cities`: City specific information.

`latlong`: Latitude and longitude coordinates for each city.

`counties`: County information.

`tables`: Descriptive information about these tables

`getVarDesc` *Get variable descriptions for city dataframe*

Description

Retrieve variable descriptions for a city dataframe.

Usage

```
getVarDesc(dataframe)
```

Arguments

`dataframe`	a city dataframe

Details

This function retrieves metadata from the `variables` table for each of the variables in the city dataframe.

Value

A list containing variable names and a short description for each.

Author(s)

Roger D. Peng ⟨rpeng@jhsph.edu⟩

Examples

```
registerDB()
loadCity("ny")
getVarDesc(ny)
```

---

listAllCities *List all NMMAPS cities*

Description

List all city names in NMMAPS database

Usage

```
listAllCities(full.names = FALSE)
```

Arguments

full.names	If FALSE the abbreviated city names are returned. If TRUE the full names for the cities are returned. The default is FALSE

Details

The abbreviated names returned by `listAllCities` can be used to load data using `readCity`, `attachCity`, or `loadCity`.

Value

A character vector containing the names of all the cities in the data base.

Examples

```
## Register full/raw database
registerDB()

## Return the abbreviated names
listAllCities()

## Return the full city names
listAllCities(full.names = TRUE)
```

`preprocessEx` *Some example preprocessing functions*

Description

Some example preprocessing functions for building NMMAPS databases.

Usage

```
collapseEndpoints(dataframe)
basicNMMAPS(dataframe)
NMMAPSlite(dataframe)
reduced(dataframe)
```

Arguments

`dataframe`	A dataframe, i.e. from the NMMAPS database

Details

`collapseEndpoints` takes all of the mortality variables in a dataframe and collapses (i.e. sums) the counts across age categories. The result is a dataframe with only 5114 rows where the mortality counts are for persons of all ages. The pollution and weather variables remain unchanged.

`basicNMMAPS` prepares a dataframe for a basic analysis of PM10 and either total, CVD, or respiratory mortality. If a dataframe does not contain any PM10 at all, the function returns `NULL`

`NMMAPSlite` creates a reduced dataframe with only a subset of the variables. Variables that are excluded include hourly measurements of some pollutants and lagged variables.

`reduced` take another subset of variables and returns a reduced dataframe.

Value

For each function, a modified dataframe is returned, or `NULL`.

Author(s)

Roger D. Peng ⟨rpeng@jhsph.edu⟩

Examples

```
## Create `lite' version
dbInfo <- buildDB(NMMAPSlite, cityList = c("ny", "la", "det"))
show(dbInfo)
loadCity("ny")
dim(ny)  ## Should have fewer columns

dbInfo <- buildDB(collapseEndpoints, cityList = c("seat", "chic"))
show(dbInfo)

loadCity("seat")
dim(seat)  ## Should only have 5114 rows
with(seat, c(mean(death), mean(cvd), mean(resp)))
```

rebuildDB-methods *Methods for Function rebuildDB in Package 'NMMAPSdata'*

Description

Methods for function `rebuildDB` in package **NMMAPSdata**

Methods

object = **"NMMAPSdbInfo"** object that stores information about a database derived from the full/raw NMMAPS database. This object is returned by the `buildDB` function or can be retrieved with a call to `getDBInfo`.

Warning

This function is still experimental and will likely work in some cases and not in others.

seasonal *Preprocessing function for seasonal models*

Description

Preprocess NMMAPS database for seasonal models of PM10 and mortality.

Usage

```
seasonal(dataframe)
```

Arguments

`dataframe`	a city dataframe

Details

`seasonal` prepares an NMMAPS city dataframe for a seasonal analysis of PM10 and mortality. See the References for details about the models.

Value

A modified dataframe.

Author(s)

Roger D. Peng ⟨rpeng@jhsph.edu⟩

References

Peng RD, Dominici F, Pastor-Barriuso R, Zeger SL, Samet JM (2004). "Seasonal Analyses of PM10 and Mortality," Johns Hopkins University Department of Biostatistics Working Papers, Working paper 41.

http://www.bepress.com/jhubiostat/paper41/

Examples

```
registerDB(NULL)
dframe <- readCity("ny")
modified.dframe <- seasonal(dframe)
```

---

tempDLM *Preprocessing functions for temperature models*

Description

Preprocessing functions for temperature distributed lags models.

Usage

```
tempDLM(dat.raw)
tempDLMCities(pmmiss = FALSE)
```

Arguments

dat.raw	a dataframe
pmmiss	logical, exclude cities with no PM10 data?

Details

`tempDLM` prepares the NMMAPS city dataframes for an analysis of PM10 and mortality with distributed lags for temperature. See the References for details about the models. `tempDLMCities`, for `pmmiss = TRUE`, returns a list of cities suitable for this analysis.

Value

`tempDLM` returns a modified dataframe. `tempDLMCities` returns a character vector of abbreviated city names.

Author(s)

Leah J. Welty ⟨lwelty@jhsph.edu⟩

References

Welty LJ, Zeger SL (2004). "Flexible Distributed Lag Models: Are the Acute Effects of PM10 on Mortality the Result of Inadequate Control for Weather and Season?" Johns Hopkins University Department of Biostatistics Working Papers, Working paper 38.

http://www.bepress.com/jhubiostat/paper38/

Examples

```
registerDB(NULL)
dframe <- readCity("chic")
modified.dframe <- tempDLM(dframe)
```

`variableDesc` *U.S. Cities Variable Descriptions*

Description

Descriptions of pollutant, meteorology, and mortality variables for U.S. cities (1987–2000).

Details

The following information (and more) can be found by loading the `variables` table (i.e. via `data{variables}`)

Each city dataframe contains variables on:

city Four letter city abbreviation

date Date

dow Day of week

agecat 3 age categories

accident Accidental Deaths

copd Chronic Obstructive Pulmonary Disease

cvd Cardiovascular Deaths

death All cause mortality excluding accident

inf Influenza

pneinf Pneumonia and Influenza

pneu Pneumonia

resp Respiratory Deaths

tmpd Mean temperature

tmax Maximum temperature

tmin Minimum temperature

tmean 24 hourly mean temperature

dptp Dew point temperature

rhum Mean relative humidity

mxrh Maximum relative humidity

mnrh Minimum relative humidity

pm10mean PM10 Mean

pm10n No. non-missing

pm10median PM10 Median

pm10max1 Maximum Hourly PM10

pm10max2 2nd Maximum Hourly PM10

pm10max3 3rd Maximum Hourly PM10

pm10max4 4th Maximum Hourly PM10

pm10max5 5th Maximum Hourly PM10

pm10trend Daily mean of 1-year trends

pm10mtrend Daily median of 1-year trends
pm10grandmean Grand Mean
pm10tmean PM10 Trimmed Mean
pm10meanmax Mean of maximum PM10
pm25mean Mean PM2.5
pm25n No. non-missing
pm25median Median PM2.5
pm25max1 Maximum Hourly PM2.5
pm25max2 2nd Maximum Hourly PM2.5
pm25max3 3rd Maximum Hourly PM2.5
pm25max4 4th Maximum Hourly PM2.5
pm25max5 5th Maximum Hourly PM2.5
pm25trend Daily mean of 1-year trends
pm25mtrend Daily median of 1-year trends
pm25grandmean Grand Mean
pm25tmean Trimmed Mean PM2.5
pm25meanmax Mean of maximum PM2.5
o3mean Mean O3
o3n No. non-missing
o3median Median O3
o3h0 0 hour mean
o3h1 1 hour mean
o3h2 2 hour mean
o3h3 3 hour mean
o3h4 4 hour mean
o3h5 5 hour mean
o3h6 6 hour mean
o3h7 7 hour mean
o3h8 8 hour mean
o3h9 9 hour mean
o3h10 10 hour mean
o3h11 11 hour mean
o3h12 12 hour mean
o3h13 13 hour mean
o3h14 14 hour mean
o3h15 15 hour mean
o3h16 16 hour mean
o3h17 17 hour mean
o3h18 18 hour mean
o3h19 19 hour mean

o3h20 20 hour mean
o3h21 21 hour mean
o3h22 22 hour mean
o3h23 23 hour mean
o3max1 Maximum Hourly O3
o3max2 2nd Maximum Hourly O3
o3max3 3rd Maximum Hourly O3
o3max4 4th Maximum Hourly O3
o3max5 5th Maximum Hourly O3
o3trend Daily mean of 1-year trends
o3mtrend Daily median of 1-year trends
o3grandmean Grand Mean
o3tmean Trimmed Mean O3
o3meanmax Mean of maximum O3
so2mean Mean SO2
so2n No. non-missing
so2median Median SO2
so2h0 0 hour mean
so2h1 1 hour mean
so2h2 2 hour mean
so2h3 3 hour mean
so2h4 4 hour mean
so2h5 5 hour mean
so2h6 6 hour mean
so2h7 7 hour mean
so2h8 8 hour mean
so2h9 9 hour mean
so2h10 10 hour mean
so2h11 11 hour mean
so2h12 12 hour mean
so2h13 13 hour mean
so2h14 14 hour mean
so2h15 15 hour mean
so2h16 16 hour mean
so2h17 17 hour mean
so2h18 18 hour mean
so2h19 19 hour mean
so2h20 20 hour mean
so2h21 21 hour mean
so2h22 22 hour mean

so2h23 23 hour mean
so2max1 Maximum Hourly SO2
so2max2 2nd Maximum Hourly SO2
so2max3 3rd Maximum Hourly SO2
so2max4 4th Maximum Hourly SO2
so2max5 5th Maximum Hourly SO2
so2trend Daily mean of 1-year trends
so2mtrend Daily median of 1-year trends
so2grandmean Grand Mean
so2tmean Trimmed Mean SO2
so2meanmax Mean of maximum SO2
no2mean Mean NO2
no2n No. non-missing
no2median Median NO2
no2h0 0 hour mean
no2h1 1 hour mean
no2h2 2 hour mean
no2h3 3 hour mean
no2h4 4 hour mean
no2h5 5 hour mean
no2h6 6 hour mean
no2h7 7 hour mean
no2h8 8 hour mean
no2h9 9 hour mean
no2h10 10 hour mean
no2h11 11 hour mean
no2h12 12 hour mean
no2h13 13 hour mean
no2h14 14 hour mean
no2h15 15 hour mean
no2h16 16 hour mean
no2h17 17 hour mean
no2h18 18 hour mean
no2h19 19 hour mean
no2h20 20 hour mean
no2h21 21 hour mean
no2h22 22 hour mean
no2h23 23 hour mean
no2max1 Maximum Hourly NO2
no2max2 2nd Maximum Hourly NO2

no2max3 3rd Maximum Hourly NO2
no2max4 4th Maximum Hourly NO2
no2max5 5th Maximum Hourly NO2
no2trend Daily mean of 1-year trends
no2mtrend Daily median of 1-year trends
no2grandmean Grand Mean
no2tmean Trimmed Mean NO2
no2meanmax Mean of maximum NO2
comean Mean CO
con No. non-missing
comedian Median CO
coh0 0 hour mean
coh1 1 hour mean
coh2 2 hour mean
coh3 3 hour mean
coh4 4 hour mean
coh5 5 hour mean
coh6 6 hour mean
coh7 7 hour mean
coh8 8 hour mean
coh9 9 hour mean
coh10 10 hour mean
coh11 11 hour mean
coh12 12 hour mean
coh13 13 hour mean
coh14 14 hour mean
coh15 15 hour mean
coh16 16 hour mean
coh17 17 hour mean
coh18 18 hour mean
coh19 19 hour mean
coh20 20 hour mean
coh21 21 hour mean
coh22 22 hour mean
coh23 23 hour mean
comax1 Maximum Hourly CO
comax2 2nd Maximum Hourly CO
comax3 3rd Maximum Hourly CO
comax4 4th Maximum Hourly CO
comax5 5th Maximum Hourly CO

cotrend Daily mean of 1-year trends
comtrend Daily median of 1-year trends
cograndmean Grand Mean
cotmean Trimmed Mean CO
comeanmax Mean of maximum CO
rmtmpd Adjusted 3-day lag temperature
rmdptp Adjusted 3-day lag Dew point temperature
markaccident Exclusions for Accidental Deaths
markcopd Exclusions for COPD
markcvd Exclusions for Cardiovascular Deaths
markdeath Exclusions for death
markinf Exclusions for Influenza
markpneinf Exclusions for Pneumonia and Influenza
markpneu Exclusions for Pneumonia
markresp Exclusions for Respiratory Deaths
l1pm10tmean Lag 1 PM10 trimmed mean
l1pm25tmean Lag 1 PM25 trimmed mean
l1cotmean Lag 1 CO trimmed mean
l1no2tmean Lag 1 NO2 trimmed mean
l1so2tmean Lag 1 SO2 trimmed mean
l1o3tmean Lag 1 O3 trimmed mean
l2pm10tmean Lag 2 PM10 trimmed mean
l2pm25tmean Lag 2 PM25 trimmed mean
l2cotmean Lag 2 CO trimmed mean
l2no2tmean Lag 2 NO2 trimmed mean
l2so2tmean Lag 2 SO2 trimmed mean
l2o3tmean Lag 2 O3 trimmed mean
l3pm10tmean Lag 3 PM10 trimmed mean
l3pm25tmean Lag 3 PM25 trimmed mean
l3cotmean Lag 3 CO trimmed mean
l3no2tmean Lag 3 NO2 trimmed mean
l3so2tmean Lag 3 SO2 trimmed mean
l3o3tmean Lag 3 O3 trimmed mean
l4pm10tmean Lag 4 PM10 trimmed mean
l4pm25tmean Lag 4 PM25 trimmed mean
l4cotmean Lag 4 CO trimmed mean
l4no2tmean Lag 4 NO2 trimmed mean
l4so2tmean Lag 4 SO2 trimmed mean
l4o3tmean Lag 4 O3 trimmed mean
l5pm10tmean Lag 5 PM10 trimmed mean

l5pm25tmean Lag 5 PM25 trimmed mean
l5cotmean Lag 5 CO trimmed mean
l5no2tmean Lag 5 NO2 trimmed mean
l5so2tmean Lag 5 SO2 trimmed mean
l5o3tmean Lag 5 O3 trimmed mean
l6pm10tmean Lag 6 PM10 trimmed mean
l6pm25tmean Lag 6 PM25 trimmed mean
l6cotmean Lag 6 CO trimmed mean
l6no2tmean Lag 6 NO2 trimmed mean
l6so2tmean Lag 6 SO2 trimmed mean
l6o3tmean Lag 6 O3 trimmed mean
l7pm10tmean Lag 7 PM10 trimmed mean
l7pm25tmean Lag 7 PM25 trimmed mean
l7cotmean Lag 7 CO trimmed mean
l7no2tmean Lag 7 NO2 trimmed mean
l7so2tmean Lag 7 SO2 trimmed mean
l7o3tmean Lag 7 O3 trimmed mean
lm1pm10tmean Lag -1 PM10 trimmed mean
lm1pm25tmean Lag -1 PM25 trimmed mean
lm1cotmean Lag -1 CO trimmed mean
lm1no2tmean Lag -1 NO2 trimmed mean
lm1so2tmean Lag -1 SO2 trimmed mean
lm1o3tmean Lag -1 O3 trimmed mean
lm2pm10tmean Lag -2 PM10 trimmed mean
lm2pm25tmean Lag -2 PM25 trimmed mean
lm2cotmean Lag -2 CO trimmed mean
lm2no2tmean Lag -2 NO2 trimmed mean
lm2so2tmean Lag -2 SO2 trimmed mean
lm2o3tmean Lag -2 O3 trimmed mean
lm3pm10tmean Lag -3 PM10 trimmed mean
lm3pm25tmean Lag -3 PM25 trimmed mean
lm3cotmean Lag -3 CO trimmed mean
lm3no2tmean Lag -3 NO2 trimmed mean
lm3so2tmean Lag -3 SO2 trimmed mean
lm3o3tmean Lag -3 O3 trimmed mean
lm4pm10tmean Lag -4 PM10 trimmed mean
lm4pm25tmean Lag -4 PM25 trimmed mean
lm4cotmean Lag -4 CO trimmed mean
lm4no2tmean Lag -4 NO2 trimmed mean
lm4so2tmean Lag -4 SO2 trimmed mean

lm4o3tmean Lag -4 O3 trimmed mean

lm5pm10tmean Lag -5 PM10 trimmed mean

lm5pm25tmean Lag -5 PM25 trimmed mean

lm5cotmean Lag -5 CO trimmed mean

lm5no2tmean Lag -5 NO2 trimmed mean

lm5so2tmean Lag -5 SO2 trimmed mean

lm5o3tmean Lag -5 O3 trimmed mean

lm6pm10tmean Lag -6 PM10 trimmed mean

lm6pm25tmean Lag -6 PM25 trimmed mean

lm6cotmean Lag -6 CO trimmed mean

lm6no2tmean Lag -6 NO2 trimmed mean

lm6so2tmean Lag -6 SO2 trimmed mean

lm6o3tmean Lag -6 O3 trimmed mean

lm7pm10tmean Lag -7 PM10 trimmed mean

lm7pm25tmean Lag -7 PM25 trimmed mean

lm7cotmean Lag -7 CO trimmed mean

lm7no2tmean Lag -7 NO2 trimmed mean

lm7so2tmean Lag -7 SO2 trimmed mean

lm7o3tmean Lag -7 O3 trimmed mean

See Also

`attachCity`, `readCity`, `loadCity`, `listAllCities`, `cityData`.

Index