The NMMAPSdata Package

Roger D. Peng   Leah J. Welty

Department of Biostatistics
Johns Hopkins Bloomberg School of Public Health

Introduction

The NMMAPSdata package for R contains daily mortality, air pollution, and weather data that were originally assembled for the National Morbidity, Mortality, and Air Pollution Study (NMMAPS). NMMAPS was a large multi-city time series study of the short-term effects of ambient air pollution on daily mortality and morbidity in the United States. The analyses of the original 90 city, 8 year database can be found in Samet et al. (2000a), Samet et al. (2000b), Dominici et al. (2003), and Daniels et al. (2004). The database has since been updated to contain data on 108 U.S. cities for 14 years (1987–2000). While the original study examined morbidity outcomes such as hospital admissions, the NMMAPSdata package does not include those data.

The NMMAPSdata package can be downloaded from `http://www.ihapss.jhsph.edu/data/NMMAPS/R/`. The package does not currently reside on CRAN, although it passes all `R CMD check` quality control tests. A source package as well as a Windows binary package are available for download. All comments that follow pertain to version 0.3-3 of the package.

In this article we provide a very brief introduction to the data and to the R functions provided in the NMMAPSdata package. As an example of how one may use the package, we present a a small multi-city time series analysis of daily non-accidental mortality and PM_{10}. A more detailed description of the NMMAPSdata package and additional examples of time series models for air pollution and mortality are available in a technical report (Peng et al., 2004).

A Brief Summary of the Data

The data are divided into 108 separate dataframes, one per city. Each dataframe has 15,342 rows and 291 columns. Although there are only 14 years of daily observations (5,114 days), the mortality data are split into three age categories, resulting in each of the weather and pollution variables being repeated three times. The dataframes are set up in this manner so that they can be used immediately in a regression analysis function such as `lm` or `glm` to fit models similar to those used in NMMAPS. Those not interested in using the separate age categories can collapse the outcome counts with the `collapseEndpoints` preprocessing function included in the package (see the next section for how to do this).

The measured pollutants in NMMAPSdata are PM_{10}, $PM_{2.5}$, SO_2, O_3, NO_2, and CO. These are the six "criteria pollutants" defined by the U.S. Environmental Protection Agency. Most cities have measurements for the gases (SO_2, O_3, NO_2, CO) every day and measurements for PM_{10} once every six days. Only a handful of cities have daily measurements of PM_{10}. Beginning in 1999, most cities have daily $PM_{2.5}$ measurements.

The meteorological variables included in the database are temperature, relative humidity, and dew point temperature. We also include as separate variables in the dataframes three day running means of temperature and dew point temperature.

General information about the data and how they were assembled can be found in Samet et al. (2000a). Interested readers are also encouraged to visit the Internet-based Health and Air Pollution Surveillance System (IHAPSS) website at `http://www.ihapss.jhsph.edu/` which contains more details about how the data were originally processed.

Overview of NMMAPSdata

The NMMAPSdata package can be loaded into R in the usual way.

```
> library(NMMAPSdata)

NMMAPS Data (version 0.3-3)
    Type '?NMMAPS' for a brief introduction to the NMMAPS database.
    Type 'NMMAPScite()' for information on how to cite 'NMMAPSdata' in
    publications.  A short tutorial vignette is available and can be
    viewed by typing 'vignette("NMMAPSdata")'
```

Some introductory material regarding the database can be found by typing `?NMMAPS` at the command line.

The primary function in NMMAPSdata is `buildDB`, which can be used to build custom versions of the full NMMAPS database. In particular, most users will not need to use the entire database (291 variables for each of 108 cities) at any given time. The custom versions of the database may also contain transformations of variables or newly created/derived variables. Possible new variables include:

- Exclusions: Remove days with extreme pollution, mortality, or temperature
- Fill in occasional/sporadic missing data
- Create seasonal indicators
- Compute running means of variables

There are, of course, many other possibilities.

The function `buildDB` has one required argument, `procFunc`, a processing function (or function name) which will be applied to the city dataframes. By default, `buildDB` applies the processing function to all cities in the NMMAPS package. However, if a character vector with abbreviated city names is supplied to argument `cityList`, the processing function will be applied only to the cities in that list.

```
> args(buildDB)

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

By default, `buildDB` builds a new database in the package installation directory. If installing the new database in this location is not desirable, one can specify another directory via the `path` argument.

The function specified in the `procFunc` argument should return a (possibly modified) dataframe or `NULL`. If `procFunc` returns `NULL` when it is applied to a particular city, `buildDB` will skip that city and not include the dataframe in the new database. This is useful for excluding cities that do not contain observations for a particular pollutant without having to directly specify a list of cities to include.

Once a database is constructed using `buildDB`, it is *registered* via call to `registerDB`. When `registerDB` is called with no arguments it sets the full (unprocessed) NMMAPS database as the currently registered database. The argument `dbName` can be used to register other previously built databases, however, only one database can be registered at a time. The processing function used to create the new database is always stored with the newly created database, ensuring that all of the transformations to the original data are documented with code.

```
> registerDB()
> showDB()

Currently using full NMMAPS database
```

Each of the city dataframes can be loaded, read, or attached using `loadCity`, `readCity`, or `attachCity`, respectively. For example we can load, read, or attach the full New York City dataframe.

```
> loadCity("ny")
> ny[1:5, 1:10]

  city     date dow agecat accident copd cvd death inf pneinf
1   ny 19870101   5      1       10    3  22    73   0      3
2   ny 19870102   6      1        4    4  20    68   0      1
3   ny 19870103   7      1        5    0  17    56   0      3
4   ny 19870104   1      1        5    1  18    55   0      2
5   ny 19870105   2      1        2    2  14    60   0      2

> dframe <- readCity("ny")
> identical(dframe, ny)

[1] TRUE

> attachCity("ny")
> search()

 [1] ".GlobalEnv"         "ny"                 "package:NMMAPSdata"
 [4] "package:tools"      "package:methods"    "package:stats"
 [7] "package:graphics"   "package:utils"      "Autoloads"
[10] "package:base"
```

We can print the first 10 days of death counts from cardiovascular disease and non-accidental deaths for people < 65 years old:

```
> cvd[1:10]

 [1] 22 20 17 18 14 18 17 16 25 20

> death[1:10]

 [1] 73 68 56 55 60 80 64 63 64 65
```

The function `attachCity` will mostly likely only be useful for interactive work. Furthermore, only one city dataframe can be usefully attached at a time since all of the variables in the most recently attached dataframe will mask the variables in previously attached dataframes.

Example: Analysis of $\mathbf{PM_{10}}$ and Mortality

In this section we illustrate how to fit models similar to those used in Dominici et al. (2002a,b, 2003). The basic NMMAPS model for a single city is an overdispersed Poisson model of the following form

$$\begin{array}{rcl} Y_t & \sim & \text{Poisson}(\mu_t) \\ \log \mu_t & = & \text{DOW}_t + \text{AgeCat} \\ & & +ns(\text{temp}_t, df = 6) + ns(\text{temp}_{t,1-3}, df = 6) \\ & & +ns(\text{dewpt}_t, df = 3) + ns(\text{dewpt}_{t,1-3}, df = 3) \\ & & +ns(t,\ df = 7 \times \ \#\ \text{years}) + ns(t,\ df = 1 \times \ \#\ \text{years}) \times \text{AgeCat} \\ & & +\beta \text{PM}_t \\ \text{Var}(Y_t) & = & \phi \mu_t \end{array} \qquad (1)$$

where Y_t is the number of non-accidental deaths on day t for a particular age category, AgeCat is an indicator for the age category, temp_t is the average temperature on day t, $\text{temp}_{t,1-3}$ is a running mean of temperature for the previous 3 days, and PM_t is the PM_{10} level for day t. The variables dewpt_t and $\text{dewpt}_{t,1-3}$ are current day and running mean of dew point temperature. The age categories used here are ≥ 75 years old, 65–74, and < 65. Each of the temperature and dewpoint temperature variables are related to mortality in a flexible manner via the smooth function $ns(\cdot, df)$, which indicates a natural spline with df degrees of freedom.

To process the data in preparation for fitting model (1) to PM_{10} and mortality data, one can use the built-in `basicNMMAPS` function as the argument to `procFunc` in `buildDB`. The function first checks the dataframe to see if it contains any PM_{10} data. If there is no PM_{10} data, then `NULL` is returned and `buildDB` skips the city. For cities with PM_{10} data, days with extreme mortality counts are set to `NA` (missing) using an indicator variable included in the dataframe. Then the function coerces the day-of-week and age category variables to `factor` type and creates some age category indicators. Finally, a subset of the pollution (seven lags of PM_{10}), weather (temperature and dewpoint), and mortality (total non-accidental deaths, deaths from cardiovascular disease, and deaths from respiratory diseases) variables are retained and the reduced dataframe is returned.

In order to illustrate how `basicNMMAPS` works, we use it outside `buildDB` to build a customized dataframe for New York. After looking at the body of `basicNMMAPS`, we register the full NMMAPS database, load the database for New York specifically, then using `basicNMMAPS` create the customized dataframe called `ny2`.

```
> body(basicNMMAPS)

{
    if (all(is.na(dataframe[, "pm10tmean"])))
        return(NULL)
    is.na(dataframe[, "death"]) <- as.logical(dataframe[, "markdeath"])
    is.na(dataframe[, "cvd"]) <- as.logical(dataframe[, "markcvd"])
    is.na(dataframe[, "resp"]) <- as.logical(dataframe[, "markresp"])
    Age2Ind <- as.numeric(dataframe[, "agecat"] == 2)
    Age3Ind <- as.numeric(dataframe[, "agecat"] == 3)
    dataframe[, "dow"] <- as.factor(dataframe[, "dow"])
    dataframe[, "agecat"] <- as.factor(dataframe[, "agecat"])
    varList <- c("cvd", "death", "resp", "tmpd", "rmtmpd", "dptp",
        "rmdptp", "time", "agecat", "dow", "pm10tmean", paste("l",
            1:7, "pm10tmean", sep = ""))
    data.frame(dataframe[, varList], Age2Ind = Age2Ind, Age3Ind = Age3Ind)
}

> registerDB(NULL)
> loadCity("ny")
> ny2 <- basicNMMAPS(ny)
> str(ny2)

`data.frame':        15342 obs. of  20 variables:
 $ cvd       : num  22 20 17 18 14 18 17 16 25 20 ...
 $ death     : num  73 68 56 55 60 80 64 63 64 65 ...
 $ resp      : num  6 5 3 3 4 3 5 2 5 3 ...
 $ tmpd      : num  34.5 36.5 35.8 30.2 31.8 ...
 $ rmtmpd    : num     NA    NA    NA 2.970 0.567 ...
 $ dptp      : num  33.2 29.8 23.3 20.4 21.6 ...
 $ rmdptp    : num    NA   NA   NA 9.70 4.28 ...
 $ time      : num  -2556 -2556 -2554 -2554 -2552 ...
 $ agecat    : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
 $ dow       : Factor w/ 7 levels "1","2","3","4",..: 5 6 7 1 2 3 4 5 6 7 ...
 $ pm10tmean : num     NA    NA -17.1    NA    NA ...
 $ l1pm10tmean: num     NA    NA    NA -17.1    NA ...
 $ l2pm10tmean: num     NA    NA    NA    NA -17.1 ...
 $ l3pm10tmean: num  NA NA NA NA NA ...
 $ l4pm10tmean: num  NA NA NA NA NA ...
 $ l5pm10tmean: num  NA NA NA NA NA ...
 $ l6pm10tmean: num  NA NA NA NA NA ...
 $ l7pm10tmean: num  NA NA NA NA NA ...
 $ Age2Ind    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Age3Ind    : num  0 0 0 0 0 0 0 0 0 0 ...
```

For building a multi-city database, the steps above may be avoided by directly using `buildDB`.

As an example, we use `buildDB` with processing function `basicNMMAPS` to build a small four city database that includes New York City, Los Angeles, Chicago, and Seattle. Each of the city dataframes are processed with the `basicNMMAPS` function.

```
> buildDB(procFunc = basicNMMAPS, cityList = c("ny", "la", "chic",
+     "seat"))

Creating directory /home/rpeng/R-local/lib/NMMAPSdata/db/basicNMMAPS
Creating database: basicNMMAPS
Processing cities...
+ ny ---> /home/rpeng/R-local/lib/NMMAPSdata/db/basicNMMAPS/ny.rda
+ la ---> /home/rpeng/R-local/lib/NMMAPSdata/db/basicNMMAPS/la.rda
+ chic ---> /home/rpeng/R-local/lib/NMMAPSdata/db/basicNMMAPS/chic.rda
+ seat ---> /home/rpeng/R-local/lib/NMMAPSdata/db/basicNMMAPS/seat.rda
Saving city information
Registering database location: /home/rpeng/R-local/lib/NMMAPSdata/db/basicNMMAPS

> showDB()

basicNMMAPS in /home/rpeng/R-local/lib/NMMAPSdata/db
```

The database created with a given processing function need only be built once for each city. When `buildDB` is finished building the database it automatically calls `registerDB` to make the newly built database the currently registered one and therefore ready for analysis. To use a database for subsequent analyses not immediately following its creation, the database need only be registered using `registerDB`.

`buildDB` returns (invisibly) an object of class `NMMAPSdbInfo` which has a `show` method. This object is also stored with the database and can be retrieved with the `getDBInfo` function.

```
> getDBInfo()

NMMAPS Database with cities:
  ny la chic seat

Call:
buildDB(procFunc = basicNMMAPS, cityList = c("ny", "la", "chic",
    "seat"))
```

The `NMMAPSdbInfo` object currently contains slots for the processing function, the list of cities included in the database, the full path to the database, the environment of the processing function, and the original call to `buildDB`. A character vector containing the abbreviated names of the cities included in the new database can be retrieved with the `listDBCities` function. `listDBCities` always lists the names of the cities in the currently registered database.

```
> listDBCities()

[1] "chic" "la"   "ny"   "seat"
```

The file `simple.R` contains the code for fitting model (1) and can be downloaded from the IHAPSS website or sourced directly:

```
> source(url("http://www.ihapss.jhsph.edu/data/NMMAPS/R/scripts/simple.R"))
```

It contains a function `fitSingleCity` which can be used for fitting NMMAPS-style time series models to air pollution and mortality data. There are number of arguments to `fitSingleCity`; the default values fit model (1) to a city dataframe.

```
> registerDB("basicNMMAPS")
> loadCity("la")
> fit <- fitSingleCity(data = la, pollutant = "l1pm10tmean", cause = "death")
```

One can examine the formula for `fit` to see the exact model fit to the data by `fitSingleCity`.

```
> formula(fit)

death ~ dow + agecat + ns(time, 98) + I(ns(time, 15) * Age2Ind) +
    I(ns(time, 15) * Age3Ind) + ns(tmpd, 6) + ns(rmtmpd, 6) +
    ns(dptp, 3) + ns(rmdptp, 3) + l1pm10tmean
```

The primary difference between using `fitSingleCity` and calling `glm` directly is that `fitSingleCity` will adjust the number of degrees of freedom for the smooth function of time if there are large contiguous blocks of missing data

The full summary output from the model fit is lengthy, but we can examine the estimate of the pollution effect (and its standard error) via:

```
> summary(fit)$coefficients["l1pm10tmean", ]

    Estimate   Std. Error      t value     Pr(>|t|)
0.0003722357 0.0001874975 1.9852832959 0.0472094754
```

The estimated effect is 0.0003722, which can be interpreted as approximately a 0.37% increase in mortality with a 10 μg/m^3 increase in PM$_{10}$.

For a single city analysis, returning the entire `glm` object from `fitSingleCity` is not too burdensome with respect to memory usage. However, in a multi-city analysis, with possibly up to 100 cities, it may not be desirable to store 100 `glm` objects at once, each of which can be 10–20 MB large. The function `fitSingleCity` has an argument `extractors`, which by default is `NULL`. One can pass a list of hook functions via the `extractors` argument and these functions will be applied to the object returned from the call to `glm`. This way, one can obtain relevant quantities (coefficients, standard errors, etc.) from the model fit and discard the rest.

```
> extractFun <- list(coef = function(x) summary(x)$coeff["l1pm10tmean",
+     1], std = function(x) summary(x)$coeff["l1pm10tmean", 2])
> fit <- fitSingleCity(data = la, pollutant = "l1pm10tmean", cause = "death",
+     extractors = extractFun)
> fit

$coef
[1] 0.0003722357

$std
[1] 0.0001874975
```

We can now run our multi-city analysis by calling `cityApply` with `fitSingleCity` and the list of extractor functions in `extractFun`.

```
> results <- cityApply(fitSingleCity, extractors = extractFun)
```

By default, `cityApply` applies the function specified in the `FUN` argument on all of the city dataframes in the currently registered database.

The effect estimates from the 4 cities can be pooled using a simple fixed effects model:

```
> beta <- sapply(results, "[[", "coef")
> std <- sapply(results, "[[", "std")
> weighted.mean(beta, 1/std^2) * 1000

[1] 0.2005406

> sqrt(1/sum(1/std^2)) * 1000

[1] 0.07230552
```

Future Directions

The NMMAPSdata package is a data package and we purposely omit any code for time series modeling. We are currently developing a separate package specifically designed for fitting time series models to air pollution and health data. For now, we hope that users will find the NMMAPSdata package useful for either reproducing results from previous studies or for implementing their own methods. Comments and suggestions are welcome.

References

Daniels, M. J., Dominici, F., Zeger, S. L., and Samet, J. M. (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.

Dominici, F., Daniels, M., Zeger, S. L., and Samet, J. M. (2002a), "Air Pollution and Mortality: Estimating Regional and National Dose-Response Relationships," *Journal of the American Statistical Association*, 97, 100–111.

Dominici, F., McDermott, A., Daniels, M., Zeger, S. L., and Samet, J. M. (2003), "Mortality Among Residents of 90 Cities," in *Revised Analyses of Time-Series Studies of Air Pollution and Health*, The Health Effects Institute, Cambridge, MA, pp. 9–24.

Dominici, F., McDermott, A., Zeger, S. L., and Samet, J. M. (2002b), "On the Use of Generalized Additive Models in Time-Series Studies of Air Pollution and Health," *American Journal of Epidemiology*, 156, 193–203.

Peng, R. D., Welty, L. J., and McDermott, A. (2004), "The National Morbidity, Mortality, and Air Pollution Study Database in R," Tech. Rep. 44, Johns Hopkins University Department of Biostatistics, `http://www.bepress.com/jhubiostat/paper44/`.

Samet, J. M., Dominici, F., Zeger, S. L., Schwartz, J., and Dockery, D. W. (2000a), *The National Morbidity, Mortality, and Air Pollution Study, Part I: Methods and Methodological Issues*, Health Effects Institute, Cambridge MA.

Samet, J. M., Zeger, S. L., Dominici, F., Curriero, F., Coursac, I., Dockery, D. W., Schwartz, J., and Zanobetti, A. (2000b), *The National Morbidity, Mortality, and Air Pollution Study, Part II: Morbidity and Mortality from Air Pollution in the United States*, Health Effects Institute, Cambridge, MA.