Christian Tobias Torp-Pedersen & Thomas Alexander Gerds

R-package heaven

Table of Contents

- 1. About
- 2. Prelimnaries
- 3. Philosophy
  - 3.1. What is real
  - 3.2. Ordnung muss sein
  - 3.3. TMP and TEMP folders are temporary
- 4. R-studio users
- 5. Introduction to heaven
  - 5.1. Raw data import
  - 5.2. Hypertension medication
  - 5.3. Charlson Index
  - 5.4. DREAM register
  - 5.5. getAdmLimits
  - 5.6. findCondition
  - 5.7. medicinMacro
  - 5.8. Lexis functions
  - 5.9. Matching
  - 5.10. averageIncome
- 6. Build-in datasets
  - 6.1. ATC codes

1 About

The R package `heaven` includes a series of functions that are useful when working with Danish registry data on Denmark Statistics.This document provides a simple overview and details of each program are found on the help pages for each program.

- Thanks to Jesper Lindhardsen and Anders Munch for their work on `importSAS`
- In case of R-trouble complain to
  - Christian Torp-Pedersen: `christian.tobias.torp-pedersen@regionh.dk`
  - Thomas Alexander Gerds: `tag@biostat.ku.dk`

2 Prelimnaries

- You should know or learn `data.table` because data on Denmark Statistics are large.

3 Philosophy

3.1 What is real

Many R-beginners think the objects in the workspace, Enviroment, Data view, and in the Console are real. However, it is much better to define the R-code (R or Rmd) shown in the Source window and the corresponding files as real.

The idea is that the the raw data and R-code are sufficient to produce the objects in the Enviroment at any time. The advantage of this attitude towards the project work is that one reproduce and update results, if also the project is set up in a good way.

3.2 Ordnung muss sein

Working directories corresponding to a phd or other purpose project are too often like a one-room appartment where the trash is next to the bed. There are few colleagues who are able to use a professional program for backing up files

There is a function to help you set up a clean project:

```
library(heaven)
createProject("v:/data/workdata/701234/YourName/study-I")
```

3.3 TMP and TEMP folders are temporary

To save time when picking up project work at intermediate stages (from last time), it is very useful to save some intermediate results permanently. It is also useful to save other intermediate results only temporary:

- the temporary files can be removed at any time
- the temporary files are easy to reproduce
- the temporary files are superseeded by permanent files

It is very **good style** to remove all temporary files when they are not needed any more, e.g., after inspection.

It is a sin to save a mixture of temporary files and useful files in a folder called `tmp`. It is also a sin to save backup very large or very many intermediate results.

4 R-studio users

- Go to Global options under Tools and tell R to never restore `.RData` at startup and to never save workspace data to `.RData` on exit.
- Learn about projects in R studio and set up an R-studio project for each project. This will make your work much more efficient and less prone to errors.
- Learn about the version control (via git) offered by R-studio. This provides a much better way to keep the project clean still to be able to go back to older versions of your files. Much better than calling the backup versions of your file `analysis-v1.R` ... `analysis-v17.R` where one of them is the current version.

5 Introduction to heaven

5.1 Raw data import

Check the raw in this project:

```
setwd("x:/data/rawdata_hurtig/701234/")
listRawdata()
scanRawdata()
```

Read a specific SAS data file:

```
# check the contents of a sas data file
c <- contentSAS("x:/data/rawdata_hurtig/701234/diag.sas7bdat")
# read the first 10 rows of the sas data file
d <- importSAS("x:/data/rawdata_hurtig/701234/diag.sas7bdat",obs=10)
help(importSAS)
```

The function `importSAS` is designed to read from a SAS dataset and output a R data.table. Presence of SAS is a requirement. The function can be used in a simple manner to read an entire SAS dataset, but the real advantage of the function is the capability to have SAS extract relevant data in the background and only convert these data to R. The function has a range of parameters, but understanding the following are important:

5.2 Hypertension medication

```
help(hypertensionMedication)
```

This function takes as input a prescription type data.table that includes ATC codes of medication and dates for prescriptions. The function has two options:

- When the variable index.date is not NULL then it defines hypertension as present when at least two antihypertensive drugs have been claimed in a period of 180 days before the date.
- When the variables index.date is NULL it findes the first date where two antihypertensive drugs have been claimed during two consecutive quaters (3 months)

By default the ATC codes used for the calculations are defined in the list "hypertensionATC" (see below) however the user can make a modified list if necessary.

5.3 Charlson Index

```
help(charlsonIndex)
```

This function takes as input a data.table of diagnoses and outputs a list with two elements. The first is the Charlson index of each individual at the specified time. The other is presence of components of the Charlson Index.

The variable with the index time needs to be added to the data.table with diagnoses.

5.4 DREAM register

The DREAM register is a Danish register which holds receipt of public funding on week levels and profession at month levels for any Dane that has received public funding. It is a valuable source of estimating working status of Danes. The register is organised with a huge number of variables indicating weekly receips of funding and monthly professional status.

```
help(importDREAM)
```

The function takes the DREAM register as input and outputs in a long form where the periods are provides as dates. The function can output either funding or profession.

5.5 getAdmLimits

```
help(getAdmLimits)
```

This function is designed for admission type data where a dates corresponding to start and end are present. The function will for each individual examine consecutive admissions and when there is overlap the true initial date and true discharge date are added.

5.6 findCondition

```
help(findCondition)
```

This function can from a data.table select records where a character variables partially match selected values. Typical use is to define diseases, operations of treatments based on available codes. The function finds multiple conditions in one step.

`findCondition` produces a long form output and the examples show how this can be transposed to wide form for various purposes.

5.7 medicinMacro

```
help(medicinMacro)
```

This function is named after a SAS macro that has been used for to extract treatment

periods and doses from lists of prescriptions. This function accepts as input prescription type data that includes ATC codes for medication, dates of prescriptions, number of packages provided, number of tablets in packages and stregnth of tablets. The output is treatment periods and dose of drug during periods.

The current form of the function is useful for outcome studies where conditioning on the future is not allowed. Thus calculations at any time only uses prescription information from the past.

```
help(xRecepter)
```

This is a function designed to compare with medicinMacro. It calls the old SAS function for calculations.

5.8 Lexis functions

Lexis functions are made to "split" observations to have new values of variables in time periods. The functions are necessary processing for time dependent analyses. For purely preactical reasons there are three functions:

```
help(lexisTwo)
```

This function can split observations in up to two periods as dependent on time for various conditions. A typical use is dates of comorbidities.

```
help(lexisFromTo)
```

This function can split observations in multiple time periods as dependent on start/end of a sequence of periods. Typical use is a list of intervals where selected medications are used.

```
help(lexisSeq)
```

This function can split observations in multiple periods based on vectors that define periods. Typical use is splitting based on calender periods or age

5.9 Matching

There are two function available for matching

```
help(incidenceMatch)
```

This function performs incidence density matching for nested case control studies in the context of a Cox regression model. A case is an individual who has the event and a corresponding date, the case date. For each case the function chooses a user defined

number of subjects from the at-risk set, i.e., from the subjects that are alive and event free at the case date. Matching on a variable corresponds to stratifying the baseline hazard function of the Cox regression model.

```
help(exposureMatch)
```

This function performs exposure density matching. For each case with a date of exposure a number of controls are selected that are alive and event-free and not yet exposed.

```
help(matchReport)
```

This function takes the result of matching as input and provides a simple table of matching successes and reuse of cases/controls

5.10 averageIncome

```
help(averageIncome)
```

This function calculates average income over a selected period of years prior to a chosen date. The input are two data.tables/frames with ID/dage and with ID/year /income. The default period is 5 years.

6 Build-in datasets

6.1 ATC codes

A named list of character vectors defining selected diseases (ICD8/10), operations and medications (ATC)

```
data(diseasecode)
```

A list of character vectors for ATC codes of antihypertensive medication:

```
data(hypertensionATC)
```

A data.frame with Danish education codes and variables to define education (hfaudd) as well as a standrd devision in 5 levels and division accordint to ISCED (9 levels)

```
edu_code
```

A named list with character codes for Charlson Index

```
charlson.codes
```

Author: Christian Tobias Torp-Pedersen & Thomas Alexander Gerds
Created: 2021-02-06 Sat 19:25
Validate