

Overview of Creating SEER Data R Binaries

Tom Radivoyevitch

April 17, 2018

`SEERaBomb` is for SEER and Japanese A-bomb survivor data analyses, but its focus is on SEER, for which it contributes speed to analyses by reducing file sizes to contain only items of interest. To obtain the data please visit the links in `gettingData.pdf` in the package's `doc` folder wherein use cases are also given in R scripts in the `examples` and `papers` directories. Of particular relevance here is the script `SEERaBomb/doc/examples/mkDataBinaries.R`. The goal of that script and this pdf is to help users produce R binaries of the SEER data. This is the first step to using `SEERaBomb` to analyze SEER data.

The `incidence` directory of the SEER data contains a SAS file that defines the field names, their starting positions, and their fixed widths. This file is used by `getFields()` to produce a dataframe output that presents users with field choices. This output and the SEER documentation file `seerdoc.pdf` in the SEER incidence directory should be thoroughly examined to identify fields of interest. Given user choices, the R function `pickFields()` automatically determines the sequence of widths needed to extract the data of interest using the speedy R package `LaF`. The output of `pickFields()` contains not only rows pulled from the input, but also inserted rows with widths computed to fill the gaps of no interest (see output of code below). Knowing these gap sizes enables fast file reading by `LaF` in `mkSEER()`, which produces R binaries that can be then be accessed efficiently from an R script. A common mistake is to send the output of `getFields()` directly to `mkSEER()` in an attempt to obtain all columns. This produces an error because the output of `pickFields()` includes an additional column needed by `mkSEER()` (i.e. the column `type` in the code output below). Retaining all columns is not recommended as it slows daily data loading. A comparison of loading times is provided in `SEERaBomb/doc/examples/mkDataBinaries.R`, which shows that it is best to start with the defaults, and if additional columns are needed, add them later, each time saving the larger binary generated by `mkSEER()` to a database file with a different name.

In the code below the `pickFields()` argument `picks` (not shown) has the following default:

```
picks=c("casenum","reg","race","sex","agedx","yrbrth","seqnum",  
        "modx","yrdx","histo3","ICD9","COD","surv","radiatn","chemo").
```

```
options(width=120)
library(SEERaBomb,quietly=T)
df=getFields()
(df=pickFields(df)) #numeric rowname => skip, else include

##      start width  sasnames  names          desc      type
## casenum    1    8    PUBCSNUM casenum      Patient ID integer
## reg        9   10         REG     reg          SEER registry integer
## 1         19    1
## race       20    2     RACE1V   race          Race/ethnicity integer
## 11         22    2
## sex        24    1         SEX     sex              Sex integer
## agedx      25    3     AGE_DX   agedx          Age at diagnosis integer
## yrbrth     28    4     YR_BRTH yrbrth          Year of birth integer
## 12         32    3
## seqnum     35    2     SEQ_NUM  seqnum          Sequence number integer
```

## modx	37	2	MDXRECOMP	modx	Month of diagnosis	integer
## yr dx	39	4	YEAR_DX	yr dx	Year of diagnosis	integer
## 13	43	10				string
## histo3	53	4	HISTO3V	histo3	Histologic Type ICD-0-3	integer
## 14	57	147				string
## ICD9	204	4	ICDOT09V	ICD9	Recode ICD-0-2 to 9	integer
## 15	208	47				string
## COD	255	5	CODPUB	COD	Cause of death to SEER site recode	integer
## 16	260	41				string
## surv	301	4	SRV_TIME_MON	surv	Survival months	integer
## 17	305	58				string
## radiatn	363	1	RADIATNR	radiatn	Radiation Recode	integer
## 18	364	2				string
## chemo	366	1	CHEMO_RX_REC	chemo	Chemotherapy recode (yes, no/unk)	integer