fars_markdown

A M Mahedi Hasan

31 August 2017

Fars Functions:

fars_read function

The "fars_read" function reads a CSV file, given a file name, without showing a progress line. Throws an error line if the file doesn't exist. Name of the file is the only input parameter and it returns a *tibble*.

```
fars_read <- function(filename) {
  if(!file.exists(filename))
    stop("file '", filename, "' does not exist")
  data <- suppressMessages({
    readr::read_csv(filename, progress = FALSE)
  })
  dplyr::tbl_df(data)
}
```

Example: `fars_read("accident_2013.csv.bz2")`

```
## # A tibble: 30,202 x 50
##    STATE ST_CASE VE_TOTAL VE_FORMS PVH_INVL  PEDS PERNOTMVIT PERMVIT
##    <int>   <int>    <int>    <int>    <int> <int>      <int>   <int>
##  1     1   10001        1        1        0     0          0       8
##  2     1   10002        2        2        0     0          0       2
##  3     1   10003        1        1        0     0          0       1
##  4     1   10004        1        1        0     0          0       3
##  5     1   10005        2        2        0     0          0       3
##  6     1   10006        2        2        0     0          0       3
##  7     1   10007        1        1        0     0          0       1
##  8     1   10008        2        2        0     0          0       2
##  9     1   10009        1        1        0     0          0       1
## 10     1   10010        2        2        0     0          0       4
## # ... with 30,192 more rows, and 42 more variables: PERSONS <int>,
## #   COUNTY <int>, CITY <int>, DAY <int>, MONTH <int>, YEAR <int>,
## #   DAY_WEEK <int>, HOUR <int>, MINUTE <int>, NHS <int>, ROAD_FNC <int>,
## #   ROUTE <int>, TWAY_ID <chr>, TWAY_ID2 <chr>, MILEPT <int>,
## #   LATITUDE <dbl>, LONGITUD <dbl>, SP_JUR <int>, HARM_EV <int>,
## #   MAN_COLL <int>, RELJCT1 <int>, RELJCT2 <int>, TYP_INT <int>,
## #   WRK_ZONE <int>, REL_ROAD <int>, LGT_COND <int>, WEATHER1 <int>,
## #   WEATHER2 <int>, WEATHER <int>, SCH_BUS <int>, RAIL <chr>,
## #   NOT_HOUR <int>, NOT_MIN <int>, ARR_HOUR <int>, ARR_MIN <int>,
## #   HOSP_HR <int>, HOSP_MN <int>, CF1 <int>, CF2 <int>, CF3 <int>,
## #   FATALS <int>, DRUNK_DR <int>
```

make_filename function

The "make_filename" function creats a file name based on the year that is provided as a parameter. It returns a string, which is the file name with *bz2* zipped extension.

```
make_filename <- function(year) {
  year <- as.integer(year)
  sprintf("accident_%d.csv.bz2", year)
}
```

Example: `make_filename(2013)`

```
## [1] "accident_2013.csv.bz2"
```

fars_read_years function

The "fars_read_years" function reads a range of valid years, calls the make_filename function and reads data in batch mode. It returns a list of tibble objects given the range of valid years. If a valid year is not given, the function throws an error message.

```
require(magrittr)
```

```
## Loading required package: magrittr
```

```
fars_read_years <- function(years) {
  lapply(years, function(year) {
    file <- make_filename(year)
    tryCatch({
      dat <- fars_read(file)
      dplyr::mutate(dat, year = year) %>%
        dplyr::select(MONTH, year)
    }, error = function(e) {
      warning("invalid year: ", year)
      return(NULL)
    })
  })
}
```

Example: `fars_read_years(2013:2014)`

```
## [[1]]
## # A tibble: 30,202 x 2
##    MONTH  year
##    <int> <int>
##  1     1  2013
##  2     1  2013
##  3     1  2013
##  4     1  2013
##  5     1  2013
##  6     1  2013
##  7     1  2013
##  8     1  2013
##  9     1  2013
## 10     1  2013
## # ... with 30,192 more rows
##
```

```
## [[2]]
## # A tibble: 30,056 x 2
##    MONTH  year
##    <int> <int>
##  1     1  2014
##  2     1  2014
##  3     1  2014
##  4     1  2014
##  5     1  2014
##  6     1  2014
##  7     1  2014
##  8     1  2014
##  9     1  2014
## 10     1  2014
## # ... with 30,046 more rows
```

fars_summarize_years function

The "fars_summarize_years" function counts the occurance by years and months, calls the "fars_read_years" function to do so. It takes a range of integers (years) as input and returns a wide format tibble object, given the range of valid years, of the count of occurance by years and months.

```
library(magrittr)
fars_summarize_years <- function(years) {
  dat_list <- fars_read_years(years)
  dplyr::bind_rows(dat_list) %>%
    dplyr::group_by(year, MONTH) %>%
    dplyr::summarize(n = n()) %>%
    tidyr::spread(year, n)
}
```

Example: `fars_summarize_years(2013:2014)`

```
## # A tibble: 12 x 4
##    MONTH `2013` `2014` `2015`
##  * <int>  <int>  <int>  <int>
##  1     1   2230   2168   2368
##  2     2   1952   1893   1968
##  3     3   2356   2245   2385
##  4     4   2300   2308   2430
##  5     5   2532   2596   2847
##  6     6   2692   2583   2765
##  7     7   2660   2696   2998
##  8     8   2899   2800   3016
##  9     9   2741   2618   2865
## 10    10   2768   2831   3019
## 11    11   2615   2714   2724
## 12    12   2457   2604   2781
```

fars_map_state function

The "fars_map_state" function draws a map for a given state number in the data and a valid year. It returns a *map*. It throws an error "invalid STATE number:" if an invalid state number is given. If no data is available, it throws a message "no accidents to plot".

```r
fars_map_state <- function(state.num, year) {
  filename <- make_filename(year)
  data <- fars_read(filename)
  state.num <- as.integer(state.num)

  if(!(state.num %in% unique(data$STATE)))
    stop("invalid STATE number: ", state.num)
  data.sub <- dplyr::filter(data, STATE == state.num)
  if(nrow(data.sub) == 0L) {
    message("no accidents to plot")
    return(invisible(NULL))
  }
  is.na(data.sub$LONGITUD) <- data.sub$LONGITUD > 900
  is.na(data.sub$LATITUDE) <- data.sub$LATITUDE > 90
  with(data.sub, {
    maps::map("state", ylim = range(LATITUDE, na.rm = TRUE),
              xlim = range(LONGITUD, na.rm = TRUE))
    graphics::points(LONGITUD, LATITUDE, pch = 46)
  })
}
```

Examples `fars_map_state(1, 2013)`

```
## Warning: package 'bindrcpp' was built under R version 3.4.1
```