# Package 'cbpR'

February 1, 2015

**Type** Package

**Title** This package creates panel data sets from the Census County Business Pattern database

**Version** 1.0

**Date** 2015-01-31

**Author** Jan Tilly

**Maintainer** Jan Tilly <jtilly@sas.upenn.edu>

**Description** This package creates panel data sets from the Census County Business Pattern database

**License** GPL (>= 2)

**Depends** sqldf

## R topics documented:



---

checkCbp *Check if the data directories are set*

---

### Description

This function produces an error if the data directories were not set.

### Usage

```
checkCbp()
```

### Value

`true` if the function concludes without error

---

cpbR *Get the firm count data from the County Business Patterns (CBP)*

---

### Description

This is an R Package that downloads and prepares panel data sets from the Census County Business Patterns (CBP).

### Details

It downloads the CPB data on the county level and then allows the user to aggregate the data up into larger geographic entities such as Metropolitan Statistical Areas, Micropolitan Statistical Areas, or some user defined collection of counties.

The file demo/cardealers.R contains a demonstration. It generates a panel data set for "New Car Dealers" (NAICS 441110). The data set ranges from 2000 to 2009. It aggregates the firm count data from the County Business Patterns into Micropolitan Statistical Areas and returns a dataset with annual data on the firm count, employment (if available), firm count by employment, and population figures for each Micropolitan Statistical Area. The population estimates are taken from the Annual Estimates of the Population of Metropolitan and Micropolitan Statistical Areas: April 1, 2000 to July 1, 2009 (CBSA-EST2009-01). The Micropolitan Statistical Area definitions are taken from the Census 2003-2009 Delineation Files.

The data frame `firms` contains the following columns

- `fips` the fips code
- `fipsstate` the state fips code
- `fipscty` the county fips code within the state
- `year` the year of observation
- `est` the number of establishments
- `n1_4` the number of establishments with between 1 and 4 employees
- `n5_9` the number of establishments with between 5 and 9 employees
- `n10_19` ... with between 10 and 19 employees
- `n20_49` ... with between 20 and 49 employees
- `n50_99`... with between 50 and 99 employees
- `n100_249` ... with between 100 and 249 employees
- `n250_499` ... with between 250 and 499 employees
- `n500_999` ... with between 500 and 999 employees
- `n1000plus` ... with between more than 1000 employees

## downloadCbp *Download the data*

### Description

This function downloads the data from the Census website. All Downloads are stored in the directory that was specified using `setCbpPath()`

### Usage

```
downloadCbp()
```

### Value

`true` if the function concludes without error

## getCbpPath *Get the path to the data directories*

### Description

Get the path to the data directories

### Usage

```
getCbpPath()
```

### Value

a list with the paths to the data directories

## getCBSAs *Returns a list of Core-Based Statistical Areas and counties*

### Description

This function uses the 2003-2009 definitions of Core-Based Statistical Areas and returns a map of counties into Micro- and Metropolitan Statistical Areas.

### Usage

```
getCBSAs(drop_states = NA, drop_cbsas = NA, metro = TRUE, micro = TRUE)
```

### Arguments

| | |
|---|---|
| drop_states | is a vector of state abbreviations to drop from the data |
| drop_cbsas | is a vector of CBSA IDs (as strings) to drop from the data |
| metro | is a boolean whether to include metropolitan statistical areas |
| micro | is a boolean whether to include micropolitan statistical areas |

### Value

a list of data frames with micropolitan statistical areas, metropolitan statistical areas and counties

---

## getFirmCount *Get the firm count data from the County Business Patterns (CBP)*

---

### Description

This function processes the downloaded CPB data and generates a long panel data set. The data frame contains the following columns

- fips the fips code
- fipsstate the state fips code
- fipscty the county fips code within the state
- year the year of observation
- est the number of establishments
- n1_4 the number of establishments with between 1 and 4 employees
- n5_9 the number of establishments with between 5 and 9 employees
- n10_19 ... with between 10 and 19 employees
- n20_49 ... with between 20 and 49 employees
- n50_99... with between 50 and 99 employees
- n100_249 ... with between 100 and 249 employees
- n250_499 ... with between 250 and 499 employees
- n500_999 ... with between 500 and 999 employees
- n1000plus ... with between more than 1000 employees

### Usage

```
getFirmCount(naics, years = c("09", "08", "07", "06", "05", "04", "03", "02",
  "01", "00"))
```

### Arguments

| | |
|---|---|
| naics | a string with the naics code of the industry |
| years | a vector of strings with the last two digits of years for which to process the data |

### Value

a data frame of a long panel with the CPB data on the county level

## getPopulationData *Returns a long panel of the population data*

### Description

Reshapes the population data for the Core-Based Statistical Areas into a long panel.

### Usage

```
getPopulationData()
```

### Value

a data frame with population data for micro- and metropolitan statistical areas

## getStatesFips *State abbreviations, fips codes and state names*

### Description

State abbreviations, fips codes and state names

### Usage

```
getStatesFips()
```

### Value

a data frame with state abbreviations, fips codes, and state names

## setCbpPath *Set the environmental variables to the data directories*

### Description

This function sets the environmental variables that point to the directories where the downloaded and processed data is stored

### Usage

```
setCbpPath(data_in, data_out)
```

## Arguments

| | |
|---|---|
| `data_in` | a string with the path to the directory where the downloaded data will be stored |
| `data_out` | a string with the path to the directory where the processed data will be stored |

## Value

`true` if the function concludes successfully

# Index