Package 'rwalkr'

July 19, 2018

Type Package

Title API to Melbourne Pedestrian Data

Version 0.3.4

Description Provides API to Melbourne pedestrian data in tidy data form.

Depends R (>= 3.1.3)

Imports tidyr, dplyr, httr, tibble

Suggests shiny (>= 1.0.4), plotly

URL http://pkg.earo.me/rwalkr

BugReports https://github.com/earowang/rwalkr/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Author Earo Wang [aut, cre] (<https://orcid.org/0000-0001-6448-5260>)

Maintainer Earo Wang <earo.wang@gmail.com>

Repository CRAN

Date/Publication 2018-07-19 10:20:03 UTC

R topics documented:

lookup_sensor *Look up sensor names between* run_melb() *and* walk_melb()

Description

One-to-one corresponding sensor names between run_melb() and walk_melb()

Usage

```
lookup_sensor()
```

Details

Two APIs (Socrata and compedapi) code some sensors using different names. This functions returns a data frame that allows to compare sensor names obtained from these two APIs.

Value

A data frame including three columns:

- run: Sensor names obtained from the run_melb() function using Socrata
- walk: Sensor names obtained from the walk_melb() function using compedapi
- match: whether sensor names are identical or not

Examples

```
lookup_sensor()
```

pull_sensor *API using Socrata to Melbourne pedestrian sensor locations*

Description

Provides API using Socrata to Melbourne pedestrian sensor locations.

Usage

```
pull_sensor(app_token = NULL)
```

Arguments

app_token	Characters giving the application token. A limited number of requests can be made without an app token (NULL), but they are subject to much lower throttling limits than request that do include one. Sign up for an app token here.

Details

It provides API using Socrata.

Value

A data frame including these variables as follows:

- Sensor: Sensor name (43 sensors up to date)
- Sensor_ID: Sensor identifier
- Longitude: Longitude
- Latitude: Latitude
- Location_Type: Location type
- Year_Installed: Year installed

See Also

run_melb

Examples

```
## Not run:
  pull_sensor()

## End(Not run)
```

---

run_melb *API using Socrata to Melbourne pedestrian data*

Description

Provides API using Socrata to Melbourne pedestrian data in a tidy data form.

Usage

```
run_melb(year = NULL, sensor = NULL, tz = "", na.rm = FALSE,
  app_token = NULL)
```

Arguments

year	An integer or a vector of integers. By default, it's the current year.
sensor	Sensor names. By default, it pulls all the sensors. Use lookup_sensor to see the available sensors.
tz	Time zone. By default, "" is the current time zone. For this dataset, the local time zone is "Australia/Melbourne" which would be the most appropriate, depending on OS.

na.rm	Logical. FALSE is the default suggesting to include NA in the dataset. TRUE removes the NAs.
app_token	Characters giving the application token. A limited number of requests can be made without an app token (NULL), but they are subject to much lower throttling limits than request that do include one. Sign up for an app token here.

Details

It provides API using Socrata, where counts are uploaded on a monthly basis. The up-to-date data would be till the previous month. The data is sourced from Melbourne Open Data Portal. Please refer to Melbourne Open Data Portal for more details about the dataset and its policy.

Value

A tibble including these variables as follows:

- Sensor: Sensor name (46 sensors up to date)
- Date_Time: Date time when the pedestrian counts are recorded
- Date: Date associated with Date_Time
- Time: Time of day
- Count: Hourly counts

See Also

walk_melb

Examples

```
## Not run:
  # Retrieve the year 2017
  ped_df17 <- run_melb(year = 2017)
  head(ped_df17)

  # Retrieve the year 2017 for Southern Cross Station
  sx_df17 <- run_melb(year = 2017, sensor = "Southern Cross Station")
  head(sx_df17)

## End(Not run)
```

---

shine_melb *A simple shiny app for pedestrian data*

Description

Provides a GUI to download data of selected sensors over a specified period as a CSV file, accompanied with basic visualisation.

Usage

```
shine_melb()
```

Details

It offers some basic plots to give a glimpse of the data over a short time period. In order to be reproducible, scripting using `walk_melb` or `run_melb` is recommended.

Value

A shiny app.

See Also

walk_melb, run_melb

Examples

```
## Not run:
  shine_melb()

## End(Not run)
```

---

walk_melb *API using compedapi to Melbourne pedestrian data*

Description

Provides API using compedapi to Melbourne pedestrian data in a tidy data form.

Usage

```
walk_melb(from = to - 6L, to = Sys.Date() - 1L, tz = "", na.rm = FALSE,
  session = NULL)
```

Arguments

`from`	Starting date.
`to`	Ending date.
`tz`	Time zone. By default, "" is the current time zone. For this dataset, the local time zone is "Australia/Melbourne" which would be the most appropriate, depending on OS.
`na.rm`	Logical. `FALSE` is the default suggesting to include `NA` in the dataset. `TRUE` removes the `NA`s.
`session`	`NULL` or "shiny". For internal use only.

Details

It provides API using compedapi, where counts are uploaded on a daily basis. The up-to-date data would be till the previous day. The data is sourced from Melbourne Open Data Portal. Please refer to Melbourne Open Data Portal for more details about the dataset and its policy.

Value

A tibble including these variables as follows:

- Sensor: Sensor name (43 sensors up to date)
- Date_Time: Date time when the pedestrian counts are recorded
- Date: Date associated with Date_Time
- Time: Time of day
- Count: Hourly counts

See Also

run_melb

Examples

```
## Not run:
  # Retrieve last week data
  ped_df1 <- walk_melb()
  head(ped_df1)

  # Retrieve data of a speficied period
  start_date <- as.Date("2017-07-01")
  end_date <- start_date + 6L
  ped_df2 <- walk_melb(from = start_date, to = end_date)
  head(ped_df2)

## End(Not run)
```

Index