Package: bikedata (via r-universe)

October 29, 2026

Title Download and Aggregate Data from Public Hire Bicycle Systems

Version 0.2.5.071

Description Download and aggregate data from all public hire bicycle systems which provide open data, currently including 'Santander' Cycles in London, U.K.; from the U.S.A., 'Ford GoBike' in San Francisco CA, 'citibike' in New York City NY, 'Divvy' in Chicago IL, 'Capital Bikeshare' in Washington DC, 'Hubway' in Boston MA, 'Metro' in Los Angeles LA, 'Indego' in Philadelphia PA, and 'Nice Ride' in Minnesota; 'Bixi' from Montreal, Canada; and 'mibici' from Guadalajara, Mexico.

License GPL-3

URL https://docs.ropensci.org/bikedata/, https://github.com/ropensci/bikedata

BugReports https://github.com/ropensci/bikedata/issues

Depends R (>= 3.0)

Imports brio, DBI, httr, lubridate, magrittr, Rcpp, readxl, reshape2, RSQLite, tibble, xml2

Suggests dodgr, knitr, markdown, rmarkdown, testthat

LinkingTo BH, Rcpp

VignetteBuilder knitr

Encoding UTF-8

LazyData true

NeedsCompilation yes

RoxygenNote 7.3.3

X-schema.org-applicationCategory Data Access

X-schema.org-isPartOf https://ropensci.org

X-schema.org-keywords bicycle-hire-systems, bike-hire-systems, bike-hire, bicycle-hire, database, bike-data

Config/pak/sysreqs libicu-dev libxml2-dev libssl-dev

Repository https://ropensci.r-universe.dev

Date/Publication 2026-06-25 14:38:26 UTC

RemoteUrl https://github.com/ropensci/bikedata

RemoteRef HEAD

RemoteSha 5d0098c5645a7d0bc84b1fa006d1cb1068fa1496

Contents

bike_cities . . . 2
bike_daily_trips . . . 3
bike_datelimits . . . 4
bike_db_totals . . . 5
bike_demographic_data . . . 6
bike_distmat . . . 7
bike_latest_files . . . 8
bike_match_matrices . . . 9
bike_rm_db . . . 9
bike_rm_test_data . . . 10
bike_stations . . . 11
bike_stored_files . . . 12
bike_summary_stats . . . 13
bike_test_data . . . 14
bike_tripmat . . . 14
bike_write_test_data . . . 16
bikedata . . . 17
dl_bikedata . . . 18
index_bikedata_db . . . 20
lo_stns . . . 20
store_bikedata . . . 21

Index **23**

`bike_cities` *List of cities currently included in bikedata*

Description

List of cities currently included in bikedata

Usage

```
bike_cities()
```

Value

A `data.frame` of cities, abbreviations, and names of bike systems currently able to be accessed.

Examples

```
bike_cities ()
```

`bike_daily_trips` *Extract daily trip counts for all stations*

Description

Extract daily trip counts for all stations

Usage

```
bike_daily_trips(
  bikedb,
  city,
  station,
  member,
  birth_year,
  gender,
  standardise = FALSE
)
```

Arguments

`bikedb`	A string containing the path to the SQLite3 database. If no directory specified, it is presumed to be in `tempdir()`.
`city`	City for which trips are to be counted - mandatory if database contains data for more than one city
`station`	Optional argument specifying bike station for which trips are to be counted
`member`	If given, extract only trips by registered members (`member = 1` or `TRUE`) or not (`member = 0` or `FALSE`).
`birth_year`	If given, extract only trips by registered members whose declared birth years equal or lie within the specified value or values.
`gender`	If given, extract only records for trips by registered users declaring the specified genders (`f/m/.` or `2/1/0`).
`standardise`	If TRUE, daily trip counts are standardised to the relative numbers of bike stations in operation for each day, so daily trip counts are increased during (generally early) periods with relatively fewer stations, and decreased during (generally later) periods with more stations.

Value

A `data.frame` containing daily dates and total numbers of trips

Examples

```
## Not run:
bike_write_test_data () # by default in tempdir ()
# dl_bikedata (city = "la", data_dir = data_dir) # or some real data!
store_bikedata (data_dir = tempdir (), bikedb = "testdb")
# create database indexes for quicker access:
index_bikedata_db (bikedb = "testdb")

bike_daily_trips (bikedb = "testdb", city = "ny")
bike_daily_trips (bikedb = "testdb", city = "ny", member = TRUE)
bike_daily_trips (bikedb = "testdb", city = "ny", gender = "f")
bike_daily_trips (bikedb = "testdb", city = "ny", station = "173",
                  gender = 1)

bike_rm_test_data ()
bike_rm_db ("testdb")
# don't forget to remove real data!
# file.remove (list.files (".", pattern = ".zip"))

## End(Not run)
```

`bike_datelimits` *Extract date-time limits from trip database*

Description

Extract date-time limits from trip database

Usage

```
bike_datelimits(bikedb, city)
```

Arguments

`bikedb`	A string containing the path to the SQLite3 database. If no directory specified, it is presumed to be in `tempdir()`.
`city`	If given, date limits are calculated only for trips in that city.

Value

A vector of 2 elements giving the date-time of the first and last trips

Examples

```
## Not run:
data_dir <- tempdir ()
bike_write_test_data (data_dir = data_dir)
# dl_bikedata (city = 'la', data_dir = data_dir) # or some real data!
# Remove one London file that triggers an API call which may fail tests:
```

```
file.remove (file.path (tempdir(),
            "01aJourneyDataExtract10Jan16-23Jan16.csv"))
bikedb <- file.path (data_dir, 'testdb')
store_bikedata (data_dir = data_dir, bikedb = bikedb)
# create database indexes for quicker access:
index_bikedata_db (bikedb = bikedb)

bike_datelimits ('testdb') # overall limits for all cities
bike_datelimits ('testdb', city = 'NYC')
bike_datelimits ('testdb', city = 'los angeles')

bike_rm_test_data (data_dir = data_dir)
bike_rm_db (bikedb)
# don't forget to remove real data!
# file.remove (list.files ('.', pattern = '.zip'))

## End(Not run)
```

`bike_db_totals` *Count number of entries in sqlite3 database tables*

Description

Count number of entries in sqlite3 database tables

Usage

```
bike_db_totals(bikedb, trips = TRUE, city)
```

Arguments

`bikedb`	A string containing the path to the SQLite3 database.
`trips`	If true, numbers of trips are counted; otherwise numbers of stations
`city`	Optional city for which numbers of trips are to be counted

Examples

```
## Not run:
data_dir <- tempdir ()
bike_write_test_data (data_dir = data_dir)
bikedb <- file.path (data_dir, 'testdb')
# latest_lo_stns is set to FALSE just to avoid download on CRAN; this should
# normally remain at default value of TRUE:
store_bikedata (data_dir = data_dir, bikedb = bikedb, latest_lo_stns = FALSE)
# create database indexes for quicker access:
index_bikedata_db (bikedb = bikedb)

bike_db_totals (bikedb = bikedb, trips = TRUE) # total trips
bike_db_totals (bikedb = bikedb, trips = TRUE, city = 'ch')
```

```
bike_db_totals (bikedb = bikedb, trips = TRUE, city = 'ny')
bike_db_totals (bikedb = bikedb, trips = FALSE) # total stations
bike_db_totals (bikedb = bikedb, trips = FALSE, city = 'ch')
bike_db_totals (bikedb = bikedb, trips = FALSE, city = 'ny')
# numbers of stations can also be extracted with
nrow (bike_stations (bikedb = bikedb))
nrow (bike_stations (bikedb = bikedb, city = 'ch'))

bike_rm_test_data (data_dir = data_dir)
bike_rm_db (bikedb)
# don't forget to remove real data!
# file.remove (list.files ('.', pattern = '.zip'))

## End(Not run)
```

`bike_demographic_data` *Static summary of which systems provide demographic data*

Description

Static summary of which systems provide demographic data

Usage

```
bike_demographic_data()
```

Value

A `data.frame` detailing the kinds of demographic data provided by the different systems

Examples

```
bike_demographic_data ()
# Examples of filtering data by demographic parameters:
## Not run:
data_dir <- tempdir ()
bike_write_test_data (data_dir = data_dir)
bikedb <- file.path (data_dir, "testdb")
store_bikedata (data_dir = data_dir, bikedb = bikedb)
# create database indexes for quicker access:
index_bikedata_db (bikedb = bikedb)

sum (bike_tripmat (bikedb = bikedb, city = "bo")) # 200 trips
sum (bike_tripmat (bikedb = bikedb, city = "bo", birth_year = 1990)) # 9
sum (bike_tripmat (bikedb = bikedb, city = "bo", gender = "f")) # 22
sum (bike_tripmat (bikedb = bikedb, city = "bo", gender = 2)) # 22
sum (bike_tripmat (bikedb = bikedb, city = "bo", gender = 1)) # = m; 68
sum (bike_tripmat (bikedb = bikedb, city = "bo", gender = 0)) # = n; 9
# Sum of gender-filtered trips is less than total because \code{gender = 0}
# extracts all registered users with unspecified genders, while without
```

```
# gender filtering extracts all trips for registered and non-registered
# users.

# The following generates an error because Washinton DC's DivvyBike system
# does not provide demographic data
sum (bike_tripmat (bikedb = bikedb, city = "dc", birth_year = 1990))
bike_rm_test_data (data_dir = data_dir)
bike_rm_db (bikedb)

## End(Not run)
```

`bike_distmat` *Extract station-to-station distance matrix*

Description

Extract station-to-station distance matrix

Usage

```
bike_distmat(bikedb, city, expand = 0.5, long = FALSE, quiet = TRUE)
```

Arguments

`bikedb`	A string containing the path to the SQLite3 database. If no directory specified, it is presumed to be in `tempdir()`.
`city`	City for which tripmat is to be aggregated
`expand`	Distances are calculated by routing through the OpenStreetMap street network surrounding the bike stations, with the street network expanded by this amount to ensure all stations can be connected.
`long`	If FALSE, a square distance matrix of (num-stations, num_stations) is returned; if TRUE, a long-format matrix of (stn-from, stn-to, distance) is returned.
`quiet`	If FALSE, progress is displayed on screen

Value

If `long = FALSE`, a square matrix of numbers of trips between each station, otherwise a long-form **tibble** with three columns of of (start_station_id, end_station_id, distance)

Note

Distance matrices returned from `bike_distamat` use all stations listed for a given system, while trip matrices extracted with bike_tripmat will often have fewer stations because operational station numbers commonly vary over time. The two matrices may be reconciled with the `match_trips2dists` function, enabling then to be directly compared.

`bike_latest_files` *Check whether files in database are the latest published files*

Description

Check whether files in database are the latest published files

Usage

```
bike_latest_files(bikedb)
```

Arguments

`bikedb`	A string containing the path to the SQLite3 database. If no directory specified, it is presumed to be in `tempdir()`.

Value

A named vector of binary values: TRUE is files in `bikedb` are the latest versions; otherwise FALSE, in which case `store_bikedata` could be run to update the database.

Examples

```
## Not run:
data_dir <- tempdir ()
bike_write_test_data (data_dir = data_dir)
# or download some real data!
# dl_bikedata (city = 'la', data_dir = data_dir)
# Remove one London file that triggers an API call which may fail tests:
file.remove (file.path (tempdir(),
             "01aJourneyDataExtract10Jan16-23Jan16.csv"))
bikedb <- file.path (data_dir, 'testdb')
store_bikedata (data_dir = data_dir, bikedb = bikedb)
# bike_latest_files (bikedb)
# All false because test data are not current, but would pass with real data

bike_rm_test_data (data_dir = data_dir)
bike_rm_db (bikedb)
# don't forget to remove real data!
# file.remove (list.files (data_dir, pattern = '.zip'))

## End(Not run)
```

`bike_match_matrices` *Match rows and columns of distance and trip matrices*

Description

Match rows and columns of distance and trip matrices

Usage

```
bike_match_matrices(mat1, mat2)
```

Arguments

`mat1`	A wide- or long-form trip or distance matrix returned from `bike_tripmat` or `bike_distmat`.
`mat2`	The corresponding distance or trip matrix.

Value

A list of the same matrices with matching start and end stations, and in the same order passed to the routine (that is, `mat1` then `mat2`). Each kind of matrix will be identified and named accordingly as either "trip" or "dist". Matrices are returned in same format (long or wide) as submitted.

Note

Distance matrices returned from `bike_distamat` use all stations listed for a given system, while trip matrices extracted with bike_tripmat will often have fewer stations because operational station numbers commonly vary over time. This function reconciles the two matrices through matching all row and column names (or just station IDs for long-form matrices), enabling then to be directly compared.

`bike_rm_db` *Remove SQLite3 database generated with 'store_bikedat()'*

Description

If no directory is specified the `bikedb` argument passed to `store_bikedata`, the database is created in `tempdir()`. This function provides a convenient way to remove the database in such cases by simply passing the name.

Usage

```
bike_rm_db(bikedb)
```

Arguments

bikedb	The SQLite3 database containing the bikedata.

Value

TRUE if bikedb successfully removed; otherwise FALSE

Examples

```
## Not run:
data_dir <- tempdir ()
bike_write_test_data (data_dir = data_dir)
# or download some real data!
# dl_bikedata (city = "la", data_dir = data_dir)
bikedb <- file.path (data_dir, "testdb")
store_bikedata (data_dir = data_dir, bikedb = bikedb)

bike_rm_test_data (data_dir = data_dir)
bike_rm_db (bikedb)
# don't forget to remove real data!
# file.remove (list.files (data_dir, pattern = ".zip"))

## End(Not run)
```

---

bike_rm_test_data *Removes test data written with 'bike_write_test_data()'*

Description

The function bike_write_test_data() writes several small zip-compressed files to disk. The default location is tempdir(), in which case these files will be automatically removed on termination of current R session. If, however, any other value for data_dir is passed to bike_write_test_data(), then the resultant files ought be deleted by calling this function.

Usage

```
bike_rm_test_data(data_dir = tempdir())
```

Arguments

data_dir	Directory in which data were extracted.

Value

Number of files successfully removed, which should equal six.

Examples

```
## Not run:
bike_write_test_data ()
list.files (tempdir ())
bike_rm_test_data ()

bike_write_test_data (data_dir = getwd ())
list.files ()
bike_rm_test_data (data_dir = getwd ())

## End(Not run)
```

`bike_stations` *Extract station matrix from SQLite3 database*

Description

Extract station matrix from SQLite3 database

Usage

```
bike_stations(bikedb, city)
```

Arguments

`bikedb`	A string containing the path to the SQLite3 database. If no directory specified, it is presumed to be in `tempdir()`.
`city`	Optional city (or vector of cities) for which stations are to be extracted

Value

Matrix containing data for each station

Examples

```
## Not run:
data_dir <- tempdir ()
bike_write_test_data (data_dir = data_dir)
# or download some real data!
# dl_bikedata (city = 'la', data_dir = data_dir)
bikedb <- file.path (data_dir, 'testdb')
store_bikedata (data_dir = data_dir, bikedb = bikedb)
# create database indexes for quicker access:
index_bikedata_db (bikedb = bikedb)

stations <- bike_stations (bikedb)
head (stations)

bike_rm_test_data (data_dir = data_dir)
```

```
bike_rm_db (bikedb)
# don't forget to remove real data!
# file.remove (list.files (data_dir, pattern = '.zip'))

## End(Not run)
```

`bike_stored_files` *Get names of files read into database*

Description

Get names of files read into database

Usage

```
bike_stored_files(bikedb, city)
```

Arguments

`bikedb`	A string containing the path to the SQLite3 database.
`city`	Optional city for which filenames are to be obtained

Examples

```
## Not run:
data_dir <- tempdir ()
bike_write_test_data (data_dir = data_dir)
bikedb <- file.path (data_dir, 'testdb')
store_bikedata (data_dir = data_dir, bikedb = bikedb)
files <- bike_stored_files (bikedb = bikedb)
# returns a tibble with names of all stored files

bike_rm_test_data (data_dir = data_dir)
bike_rm_db (bikedb)
# don't forget to remove real data!
# file.remove (list.files ('.', pattern = '.zip'))

## End(Not run)
```

`bike_summary_stats` *Extract summary statistics of database*

Description

Extract summary statistics of database

Usage

```
bike_summary_stats(bikedb)
```

Arguments

`bikedb`	A string containing the path to the SQLite3 database. If no directory specified, it is presumed to be in `tempdir()`.

Value

A `data.frame` containing numbers of trips and stations along with times and dates of first and last trips for each city in database and a final column indicating whether the files match the latest published versions.

Examples

```
## Not run:
data_dir <- tempdir ()
bike_write_test_data (data_dir = data_dir)
# dl_bikedata (city = "la", data_dir = data_dir) # or some real data!
# Remove one London file that triggers an API call which may fail tests:
file.remove (file.path (tempdir(),
             "01aJourneyDataExtract10Jan16-23Jan16.csv"))
bikedb <- file.path (data_dir, "testdb")
store_bikedata (data_dir = data_dir, bikedb = bikedb)
# create database indexes for quicker access:
index_bikedata_db (bikedb = bikedb)

bike_summary_stats ("testdb")

bike_rm_test_data (data_dir = data_dir)
bike_rm_db (bikedb)
# don't forget to remove real data!
# file.remove (list.files (".", pattern = ".zip"))

## End(Not run)
```

bike_test_data *Test data for all 6 cities*

Description

A data set containing for each of the six cities a `data.frame` object of 200 trips.

Usage

```
bike_test_data
```

Format

A list of one data frame for each of the five cities of (bo, dc, la, lo, ny), plus two more for chicago stations and trips (ch_st, ch_tr). Each of these (except "ch_st") contains 200 representative trips.

Note

These data are only used to convert to `.zip`-compressed files using `bike_write_test_data()`. These `.zip` files can be subsequently read into an SQLite3 database using `store_bikedata`.

bike_tripmat *Extract station-to-station trip matrix or data.frame from SQLite3 database*

Description

Extract station-to-station trip matrix or data.frame from SQLite3 database

Usage

```
bike_tripmat(
  bikedb,
  city,
  start_date,
  end_date,
  start_time,
  end_time,
  weekday,
  member,
  birth_year,
  gender,
  standardise = FALSE,
  long = FALSE,
  quiet = FALSE
)
```

Arguments

bikedb	A string containing the path to the SQLite3 database. If no directory specified, it is presumed to be in tempdir().
city	City for which tripmat is to be aggregated
start_date	If given (as year, month, day) , extract only those records from and including this date
end_date	If given (as year, month, day), extract only those records to and including this date
start_time	If given, extract only those records starting from and including this time of each day
end_time	If given, extract only those records ending at and including this time of each day
weekday	If given, extract only those records including the nominated weekdays. This can be a vector of numeric, starting with Sunday=1, or unambiguous characters, so "sa" and "tu" for Saturday and Tuesday.
member	If given, extract only trips by registered members (member = 1 or TRUE) or not (member = 0 or FALSE).
birth_year	If given, extract only trips by registered members whose declared birth years equal or lie within the specified value or values.
gender	If given, extract only records for trips by registered users declaring the specified genders (f/m/. or 2/1/0).
standardise	If TRUE, numbers of trips are standardised to the operating durations of each stations, so trip numbers are increased for stations that have only operated a short time, and vice versa.
long	If FALSE, a square tripmat of (num-stations, num_stations) is returned; if TRUE, a long-format matrix of (stn-from, stn-to, ntrips) is returned.
quiet	If FALSE, progress is displayed on screen

Value

If long = FALSE, a square matrix of numbers of trips between each station, otherwise a long-form **tibble** with three columns of of (start_station_id, end_station_id, numtrips).

Note

The city parameter should be given for databases containing data from multiple cities, otherwise most of the resultant trip matrix is likely to be empty. Both dates and times may be given either in numeric or character format, with arbitrary (or no) delimiters between fields. Single numeric times are interpreted as hours, with 24 interpreted as day's end at 23:59:59.

If standardise = TRUE, the trip matrix will have the same number of trips, but they will be re-distributed as described, with more recent stations having more trips than older stations. Trip number are also non-integer in this case, whereas they are always integer-valued for standardise = FALSE.

Examples

```
## Not run:
data_dir <- tempdir ()
bike_write_test_data (data_dir = data_dir)
# or download some real data!
# dl_bikedata (city = "la", data_dir = data_dir)
bikedb <- file.path (data_dir, "testdb")
store_bikedata (data_dir = data_dir, bikedb = bikedb)
# create database indexes for quicker access:
index_bikedata_db (bikedb = bikedb)


tm <- bike_tripmat (bikedb = bikedb, city = "ny") # full trip matrix
tm <- bike_tripmat (bikedb = bikedb, city = "ny",
                    start_date = 20161201, end_date = 20161201)
tm <- bike_tripmat (bikedb = bikedb, city = "ny", start_time = 1)
tm <- bike_tripmat (bikedb = bikedb, city = "ny", start_time = "01:00")
tm <- bike_tripmat (bikedb = bikedb, city = "ny", end_time = "01:00")
tm <- bike_tripmat (bikedb = bikedb, city = "ny",
                    start_date = 20161201, start_time = 1)
tm <- bike_tripmat (bikedb = bikedb, city = "ny", start_date = 20161201,
                    end_date = 20161201, start_time = 1, end_time = 2)
tm <- bike_tripmat (bikedb = bikedb, city = "ny", weekday = 5)
tm <- bike_tripmat (bikedb = bikedb, city = "ny",
                    weekday = c("f", "sa", "th"))
tm <- bike_tripmat (bikedb = bikedb, city = "ny",
                    weekday = c("f", "th", "sa"))
tm <- bike_tripmat (bikedb = bikedb, city = "ny", member = 1)
tm <- bike_tripmat (bikedb = bikedb, city = "ny", birth_year = 1976)
tm <- bike_tripmat (bikedb = bikedb, city = "ny", birth_year = 1976:1990)
tm <- bike_tripmat (bikedb = bikedb, city = "ny", gender = "f")
tm <- bike_tripmat (bikedb = bikedb, city = "ny",
                    gender = "m", birth_year = 1976:1990)

bike_rm_test_data (data_dir = data_dir)
bike_rm_db (bikedb)
# don't forget to remove real data!
# file.remove (list.files (data_dir, pattern = ".zip"))

## End(Not run)
```

`bike_write_test_data` *Writes test data bundled with package to zip files*

Description

Writes very small test files to disk that can be used to test the package. The entire package works by reading zip-compressed data files provided by the various hire bicycle systems. This function generates some equivalent data that can be read into an `SQLite` database by the `store_bikedata()` function, so that all other package functionality can then be tested from the resultant database. This function is also used in the examples of all other functions.

Usage

```
bike_write_test_data(data_dir = tempdir())
```

Arguments

data_dir	Directory in which data are to be extracted. Defaults to tempdir(). If any other directory is specified, files ought to be removed with bike_rm_test_data().

Examples

```
## Not run:
bike_write_test_data ()
list.files (tempdir ())
bike_rm_test_data ()

bike_write_test_data (data_dir = '.')
list.files ()
bike_rm_test_data (data_dir = '.')

## End(Not run)
```

---

bikedata *Download and aggregate data from public bicycle hire systems*

Description

Download data from all public bicycle hire systems which provide open data, currently including

- Santander Cycles London, U.K.
- citibike New York City NY, U.S.A.
- Divvy Chicago IL, U.S.A.
- Capital BikeShare Washingon DC, U.S.A.
- Hubway Boston MA, U.S.A.
- Metro Los Angeles CA, U.S.A.

Download and store data

- dl_bikedata Download data for particular cities and dates
- store_bikedata Store data in SQLite3 database

Sample data for testing package

- bike_test_data Description of test data included with package
- bike_write_test_data Write test data to disk in form precisely reflecting data provided by all systems
- bike_rm_test_data Remove data written to disk with bike_write_test_data

Functions to aggregate trip data

- `bike_daily_trips` Aggregate daily time series of total trips
- `bike_stations` Extract table detailing locations and names of bicycle docking stations
- `bike_tripmat` Extract aggregate counts of trips between all pairs of stations within a given city

Summary Statistics

- `bike_summary_stats` Overall quantitative summary of database contents. All of the following functions provide individual aspects of this summary.
- `bike_db_totals` Count total numbers of trips or stations, either for entire database or a specified city.
- `bike_datelimits` Return dates of first and last trips, either for entire database or a specified city.
- `bike_demographic_data` Simple table indicating which cities include demographic parameters with their data
- `bike_latest_files` Check whether files contained in database are latest published versions

Author(s)

Mark Padgham

See Also

Useful links:

- https://docs.ropensci.org/bikedata/
- https://github.com/ropensci/bikedata
- Report bugs at https://github.com/ropensci/bikedata/issues

`dl_bikedata` *Download hire bicycle data*

Description

Download data for subsequent storage via store_bikedata.

Usage

```
dl_bikedata(city, data_dir = tempdir(), dates = NULL, quiet = FALSE)

download_bikedata(city, data_dir = tempdir(), dates = NULL, quiet = FALSE)
```

Arguments

city	City for which to download bike data, or name of corresponding bike system (see Details below).
data_dir	Directory to which to download the files
dates	Character vector of dates to download data with dates formated as YYYYMM.
quiet	If FALSE, progress is displayed on screen

Details

This function produces (generally) zip-compressed data in R's temporary directory. City names are not case sensitive, and must only be long enough to unambiguously designate the desired city. Names of corresponding bike systems can also be given. Currently possible cities (with minimal designations in parentheses) and names of bike hire systems are:

Boston (bo)	Hubway
Chicago (ch)	Divvy Bikes
Washington, D.C. (dc)	Capital Bike Share
Los Angeles (la)	Metro Bike Share
London (lo)	Santander Cycles
Minnesota (mn)	NiceRide
New York City (ny)	Citibike
Philadelphia (ph)	Indego
San Francisco Bay Area (sf)	Ford GoBike

Ensure you have a fast internet connection and at least 100 Mb space

Note

Only files that don't already exist in data_dir will be downloaded, and this function may thus be used to update a directory of files by downloading more recent files. If a particular file request fails, downloading will continue regardless. To ensure all files are downloaded, this function may need to be run several times until a message appears declaring that 'All data files already exist'

Examples

```
## Not run:
dl_bikedata (city = 'New York City USA', dates = 201601:201613)

## End(Not run)
```

index_bikedata_db *Add indexes to database created with store_bikedata*

Description

Add indexes to database created with store_bikedata

Usage

```
index_bikedata_db(bikedb)
```

Arguments

bikedb	The SQLite3 database containing the bikedata.

Examples

```
## Not run:
data_dir <- tempdir ()
bike_write_test_data (data_dir = data_dir)
# or download some real data!
# dl_bikedata (city = "la", data_dir = data_dir)
bikedb <- file.path (data_dir, "testdb")
store_bikedata (data_dir = data_dir, bikedb = bikedb)
# create database indexes for quicker access:
index_bikedata_db (bikedb = bikedb)

trips <- bike_tripmat (bikedb = bikedb, city = "LA") # trip matrix
stations <- bike_stations (bikedb = bikedb) # station data

bike_rm_test_data (data_dir = data_dir)
bike_rm_db (bikedb)
# don't forget to remove real data!
# file.remove (list.files (data_dir, pattern = ".zip"))

## End(Not run)
```

lo_stns *Docking stations for London, U.K.*

Description

A `data.frame` of station id values, names, and geographic coordinates for 786 stations for London, U.K. These stations are generally (and by default) downloaded automatically to ensure they are always up to date, but such downloading can be disabled in the `store_bikedata()` function by setting `latest_lo_stns = FALSE`.

Usage

```
lo_stns
```

Format

A `data.frame` of the four columns described above.

`store_bikedata` *Store hire bicycle data in SQLite3 database*

Description

Store previously downloaded data (via the dl_bikedata function) in a database for subsequent extraction and analysis.

Usage

```
store_bikedata(
  bikedb,
  city,
  data_dir,
  dates = NULL,
  latest_lo_stns = TRUE,
  quiet = FALSE
)
```

Arguments

`bikedb`	A string containing the path to the SQLite3 database to use. If it doesn't already exist, it will be created, otherwise data will be appended to existing database. If no directory specified, it is presumed to be in `tempdir()`.
`city`	One or more cities for which to download and store bike data, or names of corresponding bike systems (see Details below).
`data_dir`	A character vector giving the directory containing the data files downloaded with `dl_bikedata` for one or more cities. Only if this parameter is missing will data be downloaded.
`dates`	If specified and no `data_dir` is given, data are downloaded and stored only for these dates specified as vector of YYYYMM values.
`latest_lo_stns`	If `TRUE` (default), download latest version of London stations; otherwise use potentially obsolete internal version. (This parameter should not need to be changed, but can be set to `FALSE` to avoid external calls; for example when not online.)
`quiet`	If FALSE, progress is displayed on screen

Value

Number of trips added to database

Details

City names are not case sensitive, and must only be long enough to unambiguously designate the desired city. Names of corresponding bike systems can also be given. Currently possible cities (with minimal designations in parentheses) and names of bike hire systems are:

Boston (bo)	Hubway
Chicago (ch)	Divvy Bikes
Washington, D.C. (dc)	Capital Bike Share
Los Angeles (la)	Metro Bike Share
London (lo)	Santander Cycles
Minnesota (mn)	NiceRide
New York City (ny)	Citibike
Philadelphia (ph)	Indego
San Francisco Bay Area (sf)	Ford GoBike

Note

Data for different cities may all be stored in the same database, with city identifiers automatically established from the names of downloaded data files. This function can take quite a long time to execute, and may generate an SQLite3 database file several gigabytes in size.

Examples

```
## Not run:
data_dir <- tempdir ()
bike_write_test_data (data_dir = data_dir)
# or download some real data!
# dl_bikedata (city = "la", data_dir = data_dir)
bikedb <- file.path (data_dir, "testdb")
store_bikedata (data_dir = data_dir, bikedb = bikedb)
# create database indexes for quicker access:
index_bikedata_db (bikedb = bikedb)

trips <- bike_tripmat (bikedb = bikedb, city = "LA") # trip matrix
stations <- bike_stations (bikedb = bikedb) # station data

bike_rm_test_data (data_dir = data_dir)
bike_rm_db (bikedb)
# don't forget to remove real data!
# file.remove (list.files (data_dir, pattern = ".zip"))

## End(Not run)
```

Index

∗ **datasets**
bike_test_data, 14
lo_stns, 20

bike_cities, 2
bike_daily_trips, 3
bike_datelimits, 4
bike_db_totals, 5
bike_demographic_data, 6
bike_distmat, 7, *9*
bike_latest_files, 8
bike_match_matrices, 9
bike_rm_db, 9
bike_rm_test_data, 10
bike_stations, 11
bike_stored_files, 12
bike_summary_stats, 13
bike_test_data, 14
bike_tripmat, *7*, *9*, 14
bike_write_test_data, 16
bikedata, 17
bikedata-package *(*bikedata*)*, 17

dl_bikedata, 18, *21*
download_bikedata *(*dl_bikedata*)*, 18

index_bikedata_db, 20

lo_stns, 20

store_bikedata, *18*, 21