

# Package ‘rwalkr’

May 9, 2026

**Type** Package

**Title** API to Melbourne Pedestrian Data

**Version** 0.5.7

**Description** Provides API to Melbourne pedestrian and weather data  
<<https://data.melbourne.vic.gov.au>> in tidy data form.

**License** MIT + file LICENSE

**URL** <https://pkg.earo.me/rwalkr/>

**BugReports** <https://github.com/earowang/rwalkr/issues>

**Depends** R (>= 3.1.3)

**Imports** dplyr, hms, httr, tidyr, progress

**Suggests** plotly, shiny (>= 1.0.4)

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**NeedsCompilation** no

**Author** Earo Wang [aut, cre] (ORCID: <<https://orcid.org/0000-0001-6448-5260>>),  
Stuart Lee [aut] (ORCID: <<https://orcid.org/0000-0003-1179-8436>>)

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

**Repository** CRAN

**Date/Publication** 2022-11-15 18:30:02 UTC

## Contents

melb_shine . . . . .	2
melb_walk . . . . .	2
melb_walk_directional . . . . .	3
melb_walk_fast . . . . .	4
melb_weather . . . . .	6
pull_sensor . . . . .	7
pull_weather_sensors . . . . .	8
pull_weather_types . . . . .	8

<b>Index</b>	<b>10</b>
--------------	-----------

---

melb_shine	<i>A simple shiny app for pedestrian data</i>
------------	---

---

### Description

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

### Usage

```
melb_shine()
```

### 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 `melb_walk` or `melb_walk_fast` is recommended.

### Value

A shiny app.

---

melb_walk	<i>API using compedapi to Melbourne pedestrian data</i>
-----------	---

---

### Description

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

### Usage

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

### Arguments

from	Starting date.
to	Ending date.
na.rm	Logical. FALSE is the default suggesting to include NA in the dataset. TRUE removes the NAs.
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**

[melb\\_walk\\_fast](#)

**Examples**

```
## Not run:  
# Retrieve last week data  
melb_walk()  
  
# Retrieve data of a specified period  
start_date <- as.Date("2017-07-01")  
end_date <- start_date + 6L  
melb_walk(from = start_date, to = end_date)  
  
## End(Not run)
```

---

melb\_walk\_directional *API using Socrata to Melbourne pedestrian data with directions (per minute)*

---

**Description**

API using Socrata to Melbourne pedestrian data with directions (per minute)

**Usage**

```
melb_walk_directional(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 the API using [Socrata](#), to access minute by minute directional pedestrian counts for *the last hour* from pedestrian sensor devices located across the city. The data is updated every 15 minutes.

Columns `sensor_id`, `direction_1`, and `direction_2` can be used to join the data with the Sensor Locations dataset which details the location, status, and directional readings of sensors, which can be obtained from [pull\\_sensor\(\)](#).

## Value

A tibble including these variables as follows:

- `sensor_id`: Sensor name
- `date_time`: Date time when the pedestrian counts are recorded
- `date`: Date associated with `date_time`
- `time`: Time of day
- `direction_1`: Direction 1 sensor reading (count of pedestrians)
- `direction_2`: Direction 2 sensor reading (count of pedestrians)
- `total_of_directions`: Total sensor reading i.e. direction 1+2 (count of pedestrians)

## See Also

[pull\\_sensor\(\)](#)

## Examples

```
## Not run:  
melb_walk_directional()  
  
## End(Not run)
```

---

melb\_walk\_fast

*API using Socrata to Melbourne pedestrian data (per hour)*

---

## Description

API using Socrata to Melbourne pedestrian data (per hour)

## Usage

```
melb_walk_fast(year = NULL, sensor = NULL, 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 <a href="#">pull_sensor</a> to see the available sensors.
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 <a href="#">here</a> .

## Details

It provides the 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
- 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

[melb\\_walk](#)

## Examples

```
## Not run:  
# Retrieve the year 2017  
melb_walk_fast(year = 2017)  
  
# Retrieve the year 2017 for Southern Cross Station  
melb_walk_fast(year = 2017, sensor = "Southern Cross Station")  
  
## End(Not run)
```

---

`melb_weather`*API to access Melbourne microclimate sensor data*

---

## Description

API to access Melbourne microclimate sensor data

## Usage

```
melb_weather(  
  from = to - 6L,  
  to = Sys.Date(),  
  site = NULL,  
  sensor_type = NULL,  
  app_token = NULL  
)
```

## Arguments

<code>from</code>	Starting date. Earliest measurement is 2019-11-15
<code>to</code>	Ending date.
<code>site</code>	The site identifier. By default will pull in all locations that have weather sensors <a href="#">pull_weather_sensors()</a> .
<code>sensor_type</code>	The type of microclimate measurement to extract see <a href="#">pull_weather_types()</a> for details.
<code>app_token</code>	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 <a href="#">here</a> .

## Details

It provides the API using [Socrata](#), where microclimate measurements are updated on a daily basis. For data documentation, including a data dictionary see the [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:

- `site`: Site identifier, this is the location of the weather sensor
- `date_time`: Date time when the measurement was recorded
- `date`: Date associated with `date_time`
- `sensor_type`: The type of microclimate sensor reading
- `units`: The units that value is in
- `value`: The value of the reading

**See Also**

[melb\\_walk](#), [pull\\_weather\\_sensors](#), [pull\\_weather\\_types](#)

**Examples**

```
## Not run:
# Retrieve the last weeks data
melb_weather()

# Retrieve the last week but for a single location (Pelham St, Carlton)
melb_weather(site = "arc1047")

# Retrieve the last week but only ambient air temperature
melb_weather(sensor_type = "TPH.TEMP")

## End(Not run)
```

---

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](#).

**See Also**

[melb\\_walk\\_fast](#)

**Examples**

```
## Not run:
pull_sensor()

## End(Not run)
```

pull\_weather\_sensors *API using Socrata to extract Melbourne microclimate sensor locations*

---

### Description

API using Socrata to extract Melbourne microclimate sensor locations

### Usage

```
pull_weather_sensors(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 <a href="#">here</a> .
-----------	---

### Details

Extract all available climate sensor types and their identifiers [Socrata](#).

### See Also

[melb\\_weather](#)

### Examples

```
## Not run:  
pull_weather_types()  
  
## End(Not run)
```

---

pull\_weather\_types *API using Socrata to Melbourne microclimate measurement types*

---

### Description

API using Socrata to Melbourne microclimate measurement types

### Usage

```
pull_weather_types(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 <a href="#">here</a> .
-----------	---

**Details**

Extract all available climate sensor types and their identifiers [Socrata](#).

**See Also**

[melb\\_weather](#)

**Examples**

```
## Not run:  
pull_weather_types()  
  
## End(Not run)
```

# Index

melb\_shine, 2  
melb\_walk, 2, 2, 5, 7  
melb\_walk\_directional, 3  
melb\_walk\_fast, 2, 3, 4, 7  
melb\_weather, 6, 8, 9  
  
pull\_sensor, 5, 7  
pull\_sensor(), 4  
pull\_weather\_sensors, 7, 8  
pull\_weather\_sensors(), 6  
pull\_weather\_types, 7, 8  
pull\_weather\_types(), 6