Package ‘ptvapi’

June 3, 2020

Title Access the ‘Public Transport Victoria’ Timetable API
Version 1.1.2
Description Access the ‘Public Transport Victoria’ Timetable API
with results returned as familiar R data structures. Retrieve information on
stops, routes, disruptions, departures, and more.
License MIT + file LICENSE
Encoding UTF-8
LazyData true
RoxygenNote 7.1.0
Imports httr, glue, digest, jsonlite, purrr, tibble, assertthat
Suggests testthat (>= 2.1.0), dplyr, lubridate
URL https://github.com/mdneuzerling/ptvapi
BugReports https://github.com/mdneuzerling/ptvapi/issues
NeedsCompilation no
Author David Neuzerling [aut, cre, cph]
Maintainer David Neuzerling <david@neuzerling.com>
Repository CRAN
Date/Publication 2020-06-03 16:40:02 UTC

R topics documented:

- departures ................................................................. 2
- directions ................................................................. 4
- directions_on_route .................................................... 6
- disruptions ............................................................... 7
- disruptions_at_stop .................................................... 8
- disruptions_on_route .................................................. 10
- disruption_information ............................................... 11
- disruption_modes ..................................................... 12
departures

Description

departures retrieves all upcoming departures for a given stop ID and route type.

Usage

departures(
    stop_id,
    route_type,
    route_id = NULL,
    direction_id = NULL,
    platform_numbers = NULL,
    departs = Sys.time(),
    look_backwards = FALSE,
    max_results = 5,
    include_cancelled = FALSE,
    validate_results = TRUE,
    user_id = determine_user_id(),
    api_key = determine_api_key()
)

Arguments

stop_id An integer stop ID returned by the stops_on_route or stops_nearby functions.
A route type which can be provided either as a non-negative integer code, or as a character: "Tram", "Train", "Bus", "Vline" or "Night Bus". Character inputs are not case-sensitive. Use the route_types function to extract a vector of all route types.

Optionally filter by a route ID. These can be obtained with the routes function.

Optionally filter by a direction ID. These can be obtained with the directions_on_route function.

Character vector. Optionally filter results by platform number. Despite the name, these are characters.

POSIXct or Character. Optionally filter results to departures on or after the given value, according to either scheduled or estimated departure time. Characters are automatically converted to datetimes, and are assumed to be given as Melbourne time. Defaults to the current system time.

Boolean. Whether to look before deports. Use with caution (see Details). Defaults to FALSE.

Integer. The maximum number of departures to return for each route_id. Departures are ordered by estimated departure time, when available, and scheduled departure time otherwise. When set to 0, all departures after the given deports for the entire day are shown, and potentially some in the early hours of the next morning. Defaults to 5.

Logical. Whether results should be returned if they have been cancelled. Metropoli-
tan train services only. Defaults to FALSE.

Boolean. If TRUE (the default), will apply additional filters to ensure that the arguments to deports, max_results, and route_id are respected if given.

Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to ?ptvapi for more details.

Character. An API key, with dashes, provided by Public Transport Victoria. Refer to ?ptvapi for more details.

Filtering departures: The API supports filtering by departure time, to show the departures after the given time. However, its behaviour is unpredictable, returning departures around the given time, both before and after. We apply an additional filter once the results are retrieved to ensure that only departures at or after the given deports datetime are shown.

It’s not clear what functionality look_backwards has. It’s included here regardless. Moreover, it’s not clear how the API treats route_id or max_results. We filter the results after retrieval, to ensure that deports, max_results, and route_id are respected. This additional validation can be disabled by setting validate_results = TRUE.

A tibble consisting of the following columns:
### directions

#### Description

This function returns all directions with a given ID. Directions that share an ID are not necessarily related, especially if not filtering by route type. It's advised to use the `directions_on_route` function to search for directions of interest.
Usage

directions(
  direction_id,
  route_type = NULL,
  user_id = determine_user_id(),
  api_key = determine_api_key()
)

Arguments

direction_id  Integer.
route_type     Optionally filter results by a route type. A route type can be provided either as
                a non-negative integer code, or as a character: "Tram", "Train", "Bus", "Vline"
                or "Night Bus". Character inputs are not case-sensitive. Use the `route_types`
                function to extract a vector of all route types.
user_id        Integer or character. A user ID or devid provided by Public Transport Victoria.
                Refer to ?ptvapi for more details.
api_key        Character. An API key, with dashes, provided by Public Transport Victoria.
                Refer to ?ptvapi for more details.

Value

A tibble consisting of the following columns:

- direction_id
- direction_name,
- route_id
- route_type
- route_direction_description

Examples

## Not run:
```
directions(direction_id = 5)
directions(direction_id = 5, route_type = "Train")
directions(direction_id = 5, route_type = 0)
```

## End(Not run)
directions_on_route  

*Directions on a given route*

**Description**

Directions on a given route

**Usage**

```r
directions_on_route(
  route_id,
  user_id = determine_user_id(),
  api_key = determine_api_key()
)
```

**Arguments**

- **route_id**  
  Integer. These can be listed and described with the `routes` function.

- **user_id**  
  Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to `?ptvapi` for more details.

- **api_key**  
  Character. An API key, with dashes, provided by Public Transport Victoria. Refer to `?ptvapi` for more details.

**Value**

A tibble consisting of the following columns:

- `direction_id`
- `direction_name`
- `route_id`
- `route_type`
- `route_direction_description`

**Examples**

```r
## Not run:
directions_on_route(6)
```

```r
## End(Not run)
```
disruptions

Information for all disruptions

Description

Information for all disruptions

Usage

disruptions(
  route_types = NULL,
  disruption_modes = NULL,
  disruption_status = NULL,
  user_id = determine_user_id(),
  api_key = determine_api_key()
)

Arguments

route_types Integer or character vector. Optionally filter by a vector of route types. A route type can be provided either as a non-negative integer code, or as a character: “Tram”, “Train”, “Bus”, “Vline” or “Night Bus”. Character inputs are not case-sensitive. Use the route_types function to extract a vector of all route types. The filter is applied to the disruption mode, rather than the routes that are affected by the disruption. For example, filtering by the “train” route type will restrict the disruptions returned to those with a mode corresponding to “metro_train”.

disruption_modes Integer vector. Optionally filter by disruption modes. For a full list of modes and their corresponding descriptions, use the disruptions_modes function.

disruption_status Character. Can be used to filter to either “current” or “planned” disruptions. Defaults to NULL, in which case no filter is applied.

user_id Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to ?ptvapi for more details.

api_key Character. An API key, with dashes, provided by Public Transport Victoria. Refer to ?ptvapi for more details.

Value

A tibble with the following columns:

- disruption_mode
- disruption_mode_description
- disruption_id
- title
disruptions_at_stop

Disruptions at a given stop

Description
Disruptions at a given stop

Usage

```r
disruptions_at_stop(
  stop_id,
  disruption_status = NULL,
  user_id = determine_user_id(),
  api_key = determine_api_key()
)
```

Examples

```r
## Not run:
disruptions()
disruptions(route_types = c("Train", "Tram"))
disruptions(disruption_modes = c(0, 1))
disruptions(disruption_status = "current")

## End(Not run)
```
Arguments

- **stop_id**: Integer stop ID.
- **disruption_status**: Character. Can be used to filter to either "current" or "planned" disruptions. Defaults to NULL, in which case no filter is applied.
- **user_id**: Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to ?ptvapi for more details.
- **api_key**: Character. An API key, with dashes, provided by Public Transport Victoria. Refer to ?ptvapi for more details.

Value

A tibble with the following columns:

- disruption_mode
- disruption_mode_description
- disruption_id
- title
- url
- description
- disruption_status
- disruption_type
- published_on
- last_updated
- from_date
- to_date
- routes
- stops
- colour
- display_on_board
- display_status

Examples

```r
## Not run:
disruptions_at_stop(1071)
disruptions_at_stop(1071, disruption_status = "current")

## End(Not run)
```
disruptions_on_route  Disruptions on a given route

Description

Disruptions on a given route

Usage

disruptions_on_route(
    route_id,
    stop_id = NULL,
    disruption_status = NULL,
    user_id = determine_user_id(),
    api_key = determine_api_key()
)

Arguments

route_id  Integer. These can be listed and described with the routes function.
stop_id   Integer. Optionally filter results to a specific stop ID. These can be searched for with the stops_on_route and stops_nearby functions.
disruption_status  Character. Can be used to filter to either "current" or "planned" disruptions. Defaults to NULL, in which case no filter is applied.
user_id   Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to ?ptvapi for more details.
api_key   Character. An API key, with dashes, provided by Public Transport Victoria. Refer to ?ptvapi for more details.

Value

A tibble with the following columns:

- disruption_mode
- disruption_mode_description
- disruption_id
- title
- url
- description
- disruption_status
- disruption_type
- published_on
- last_updated
disruption_information

- from_date
- to_date
- routes
- stops
- colour
- display_on_board
- display_status

Examples

## Not run:

disruptions_on_route(6)
disruptions_on_route(6, stop_id = 1071)
disruptions_on_route(6, disruption_status = "current")

## End(Not run)

disruption_information

Information on a particular disruption

Description

This function can be used when the integer disruption ID is already known. This can be searched for with either disruptions, disruptions_on_route, or disruptions_at_stop functions.

Usage

disruption_information(
  disruption_id,
  user_id = determine_user_id(),
  api_key = determine_api_key()
)

Arguments

disruption_id  Integer.
user_id        Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to ?ptvapi for more details.
api_key        Character. An API key, with dashes, provided by Public Transport Victoria. Refer to ?ptvapi for more details.
disruption_modes

Value
A tibble with the following columns:

- disruption_mode
- disruption_mode_description
- disruption_id
- title
- url
- description
- disruption_status
- disruption_type
- published_on
- last_updated
- from_date
- to_date
- routes
- stops
- colour
- display_on_board
- display_status

Examples

```r
## Not run:
disruption_information(206639)
## End(Not run)
```

Description
Disruption mode types (e.g. "metro_train", "metro_tram", "school_bus", "taxi") have corresponding integer IDs. This function retrieves a named vector in which the values are the disruption mode descriptions, and the names of the vector are the description mode numbers. Note that disruption mode names are in snake case, that is, all lower case with underscores between words.

Usage

```r
disruption_modes(user_id = determine_user_id(), api_key = determine_api_key())
```
#### fare_estimate

**Calculate a fare estimate between zones**

**Description**

Retrieve fare information for a journey through the given zones. Also supports journey touch on and off times, to accommodate for discounts.

**Usage**

```r
fare_estimate(
  min_zone,
  max_zone,
  journey_touch_on = NULL,
  journey_touch_off = NULL,
  journey_in_free_tram_zone = FALSE,
  travelled_route_types = NULL,
  user_id = determine_user_id(),
  api_key = determine_api_key()
)
```

**Arguments**

- **min_zone**: Integer. Minimum zone travelled through.
- **max_zone**: Integer. Maximum zone travelled through.
- **journey_touch_on**, **journey_touch_off**: POSIXct or Character. Optionally filter results to a journey time. Values to both must be provided. Characters are automatically converted to datetimes, and are assumed to be given as Melbourne time.
journey_in_free_tram_zone
  Boolean. Defaults to FALSE.

travelled_route_types
  Integer or character vector. Optionally filter by a vector of route types. A route
type can be provided either as a non-negative integer code, or as a character:
"Tram", "Train", "Bus", "Vline" or "Night Bus". Character inputs are not case-
sensitive. Use the route_types function to extract a vector of all route types.

user_id
  Integer or character. A user ID or devid provided by Public Transport Victoria.
  Refer to ?ptvapi for more details.

api_key
  Character. An API key, with dashes, provided by Public Transport Victoria.
  Refer to ?ptvapi for more details.

Value

A data frame consisting of one row for each passenger_type, and the following columns:

- min_zone
- max_zone
- unique_zones
- early_bird
- free_tram_zone
- weekend_journey
- passenger_type
- fare_2_hour_peak
- fare_2_hour_off_peak
- fare_daily_peak
- fare_daily_off_peak
- pass_7_days
- pass_28_to_69_day_per_day
- pass_70_plus_day_per_day
- weekend_cap
- holiday_cap

Examples

## Not run:
fare_estimate(min_zone = 1, max_zone = 2)

fare_estimate(min_zone = 1, max_zone = 1, journey_in_free_tram_zone = TRUE)

fare_estimate(
  min_zone = 1,
  max_zone = 2,
  travelled_route_types = c("Train", "Tram"))
```r
fare_estimate(
  min_zone = 1,
  max_zone = 2,
  journey_touch_on = "2020-06-21 07:31:00",
  journey_touch_off = "2020-06-21 08:45:00"
)
```

## End(Not run)

---

### outlets

**Information for a all outlets**

**Description**

Information for a all outlets

**Usage**

`outlets(user_id = determine_user_id(), api_key = determine_api_key())`

**Arguments**

- **user_id** `Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to ?ptvapi for more details.`
- **api_key** `Character. An API key, with dashes, provided by Public Transport Victoria. Refer to ?ptvapi for more details.`

**Details**

The business hours are reported as characters. Usually they take on a format of "8.00AM - 10.00PM", but there variants such as "7.30AM - 11.00AM and 1.30PM - 6.00PM". For days on which an outlet is closed, the opening hours are usually reported as "CLOSED", but can also be an empty character. Some opening hours are "24 Hours". These fields are also filled with missing values and empty characters.

**Value**

A tibble with the following columns:

- `outlet_slid_spid`
- `outlet_name`
- `outlet_business`
- `outlet_latitude`
- `outlet_longitude`
- `outlet_suburb`
• outlet_postcode
• outlet_business_hour_mon
• outlet_business_hour_tue
• outlet_business_hour_wed
• outlet_business_hour_thu
• outlet_business_hour_fri
• outlet_business_hour_sat
• outlet_business_hour_sun
• outlet_notes

Examples

```r
## Not run:
outlets()

## End(Not run)
```

---

### outlets_nearby

**Information for outlets near a given location**

**Description**

Information for outlets near a given location

**Usage**

```r
outlets_nearby(
  latitude, longitude,
  max_distance = NULL,
  max_results = 30,
  user_id = determine_user_id(),
  api_key = determine_api_key()
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>latitude</td>
<td>Numeric. Latitude in decimal degrees. For example, Flinders Street Station is at approximately -37.8183 latitude.</td>
</tr>
<tr>
<td>longitude</td>
<td>Numeric. Longitude in decimal degrees. For example, Flinders Street Station is at approximately 144.9671 longitude.</td>
</tr>
<tr>
<td>max_distance</td>
<td>Integer. Optionally filter by maximum distance from the given location, in metres.</td>
</tr>
</tbody>
</table>
outlets_nearby

max_results  Integer. Defaults to 30. Caps the number of results returned.
user_id  Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to ?ptvapi for more details.
api_key  Character. An API key, with dashes, provided by Public Transport Victoria. Refer to ?ptvapi for more details.

Details
The business hours are reported as characters. Usually they take on a format of "8.00AM - 10.00PM", but there variants such as "7.30AM - 11.00AM and 1.30PM - 6.00PM". For days on which an outlet is closed, the opening hours are usually reported as "CLOSED", but can also be an empty character. Some opening hours are "24 Hours". These fields are also filled with missing values and empty characters.

Value
A tibble with the following columns:

- outlet_slid_spid
- outlet_name
- outlet_business
- outlet_latitude
- outlet_longitude
- outlet_suburb
- outlet_postcode
- outlet_business_hour_mon
- outlet_business_hour_tue
- outlet_business_hour_wed
- outlet_business_hour_thu
- outlet_business_hour_fri
- outlet_business_hour_sat
- outlet_business_hour_sun
- outlet_notes

Examples

## Not run:
outlets_nearby(latitude = -37.8183, longitude = 144.9671)

## End(Not run)
**Description**

A pattern consists of all departures, stops, routes, runs, directions and disruptions associated with a particular run ID. This is returned as a list of tibbles, with output corresponding to their respective API calls.

**Usage**

```r
patterns(
  run_id,
  route_type,
  stop_id = NULL,
  departs = Sys.time(),
  user_id = determine_user_id(),
  api_key = determine_api_key()
)
```

**Arguments**

- **run_id**: An integer run ID. This may retrieved from the `departures` or `runs_on_route` functions.
- **route_type**: Optionally filter results by a route type. A route type can be provided either as a non-negative integer code, or as a character: "Tram", "Train", "Bus", "Vline" or "Night Bus". Character inputs are not case-sensitive. Use the `route_types` function to extract a vector of all route types.
- **stop_id**: Integer. Optionally filter results to a specific stop ID. These can be searched for with the `stops_on_route` and `stops_nearby` functions.
- **departs**: POSIXct or character. Optionally filter by date. See Details. Characters are automatically converted to departs, and are assumed to be given as Melbourne time. The behaviour of the API is unpredictable when using this argument — see details. Defaults to the current system time.
- **user_id**: Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to `?ptvapi` for more details.
- **api_key**: Character. An API key, with dashes, provided by Public Transport Victoria. Refer to `?ptvapi` for more details.

**Details**

The stops tibble has an output similar to that returned by `stops_on_route`. The routes tibble does not contain service status information.

Departures: The API seems to return the earliest 7 departures. While the PTV Timetable API supports filtering patterns by datetimes, the behaviour of this argument is not reliable — it appears to filter by day only, returning the earliest 7 departures of a different day. It is recommended that departures are retrieved via the `departures` function.
Value

An object of class "ptvapi", which is effectively a list with the following names:

• departures
• stops
• routes
• runs
• directions
• disruptions

Examples

```r
## Not run:
patterns(run_id = 1, route_type = 0)
patterns(run_id = 1, route_type = "Train")

## End(Not run)
```

Description

Accessing the Public Transport Victoria Timetable API requires a user ID (also called a devid) and an API key. These can be accessed by contacting Public Transport Victoria. See https://www.ptv.vic.gov.au/footer/data-and-reporting/datasets/ptv-timetable-api/

The user ID and API key can be entered directly into all functions. Alternatively, all functions will pick up on the PTV_USER_ID and API_KEY environment variables, if defined.

Details

This is an unofficial wrapper of the Public Transport Victoria Timetable API. The author(s) of this package are unaffiliated with Public Transport Victoria.

Examples

```r
## Not run:
# tibble of all routes
routes()

# Search for routes by name (case insensitive, partial matching supported)
routes(route_name = "Frankston")

# All current disruptions
disruptions(disruption_status = "current")
```
# Train stops near Flinders Street Station
stops_nearby(
  latitude = -37.8183,
  longitude = 144.9671,
  route_types = "Train"
)

# Upcoming train departures from Flinders Street Station
departures(stop_id = 1071, route_type = "Train")

## End(Not run)

columns

---

## Description

Information for all routes

## Usage

routes(
  route_types = NULL,
  route_name = NULL,
  user_id = determine_user_id(),
  api_key = determine_api_key()
)

## Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>route_types</td>
<td>Integer or character vector. Optionally filter by a vector of route types. A route type can be provided either as a non-negative integer code, or as a character: &quot;Tram&quot;, &quot;Train&quot;, &quot;Bus&quot;, &quot;Vline&quot; or &quot;Night Bus&quot;. Character inputs are not case-sensitive. Use the route_types function to extract a vector of all route types.</td>
</tr>
<tr>
<td>route_name</td>
<td>Character. Optionally filter by route name. Partial matches are accepted, and the matches are not case sensitive.</td>
</tr>
<tr>
<td>user_id</td>
<td>Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to ?ptvapi for more details.</td>
</tr>
<tr>
<td>api_key</td>
<td>Character. An API key, with dashes, provided by Public Transport Victoria. Refer to ?ptvapi for more details.</td>
</tr>
</tbody>
</table>
Value

A tibble of routes, with the following columns:

• route_id
• route_gtfs_id
• route_name
• route_type
• route_number
• service_status
• service_status_timestamp

Examples

```r
## Not run:
routes()
routes(route_types = "Train")
routes(route_types = 0)
routes(route_types = c("Train", "Tram"))
routes(route_name = "Frankston")
routes(route_name = "Craigie")
routes(route_name = "werribee")
## End(Not run)
```

route_information

Information for a given route

Description

Information for a given route

Usage

```r
route_information(  
  route_id,  
  user_id = determine_user_id(),  
  api_key = determine_api_key()  
)
```

Arguments

- **route_id**: Integer. These can be listed and described with the `routes` function.
- **user_id**: Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to ?ptvapi for more details.
- **api_key**: Character. An API key, with dashes, provided by Public Transport Victoria. Refer to ?ptvapi for more details.
Value

A tibble of routes, with the following columns:

- route_id
- route_gtfs_id
- route_name
- route_type
- route_number
- service_status
- service_status_timestamp

Examples

## Not run:
route_information(6)

## End(Not run)

---

### route_types

Retrieve a translation from route type number to name

Description

Route types (tram, train, etc.) are provided to the PTV API as an integer code. This function retrieves a named vector in which the values are the route type descriptions, and the names of the vector are the route type numbers. Note that "Night Bus" is a separate route type.

Usage

route_types(user_id = determine_user_id(), api_key = determine_api_key())

Arguments

- **user_id**: Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to `?ptvapi` for more details.
- **api_key**: Character. An API key, with dashes, provided by Public Transport Victoria. Refer to `?ptvapi` for more details.

Value

A named integer vector in which the values are the route type descriptions, and the names of the vector are the route type numbers.
## Not run:

```r
routes_types()
```

## End(Not run)

---

### runs_onroute

**Runs on a given route**

#### Description

Runs on a given route

#### Usage

```r
runs_on_route(  
  route_id,  
  route_type = NULL,  
  user_id = determine_user_id(),  
  api_key = determine_api_key()  
)
```

#### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>route_id</code></td>
<td>Integer. These can be listed and described with the <code>routes</code> function.</td>
</tr>
<tr>
<td><code>route_type</code></td>
<td>Optionally filter results by a route type. A route type can be provided either as a non-negative integer code, or as a character: “Tram”, “Train”, “Bus”, “Vline” or “Night Bus”. Character inputs are not case-sensitive. Use the <code>route_types</code> function to extract a vector of all route types.</td>
</tr>
<tr>
<td><code>user_id</code></td>
<td>Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to <code>?ptvapi</code> for more details.</td>
</tr>
<tr>
<td><code>api_key</code></td>
<td>Character. An API key, with dashes, provided by Public Transport Victoria. Refer to <code>?ptvapi</code> for more details.</td>
</tr>
</tbody>
</table>

#### Value

A tibble with the following columns:

- `run_id`
- `route_id`
- `route_type`
- `direction_id`
- `run_sequence`
- `final_stop_id`
run_information

- destination_name
- status
- express_stop_count
- vehicle_position
- vehicle_descriptor

Examples

```r
## Not run:
runs_on_route(6)
runs_on_route(6, route_type = "Train")
runs_on_route(6, route_type = 0)

## End(Not run)
```

---

run_information Information for a given run

Description

Run IDs are not unique across the network. If you are interested in a specific run, consider supplying a value to the optional route_type argument.

Usage

```r
run_information(
  run_id,
  route_type = NULL,
  user_id = determine_user_id(),
  api_key = determine_api_key()
)
```

Arguments

- **run_id**: An integer run ID. This may retrieved from the departures or runs_on_route functions.
- **route_type**: Optionally filter results by a route type. A route type can be provided either as a non-negative integer code, or as a character: "Tram", "Train", "Bus", "Vline" or "Night Bus". Character inputs are not case-sensitive. Use the route_types function to extract a vector of all route types.
- **user_id**: Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to ?ptvapi for more details.
- **api_key**: Character. An API key, with dashes, provided by Public Transport Victoria. Refer to ?ptvapi for more details.
search_outlets

Value

A tibble with the following columns:

- run_id
- route_id
- route_type
- direction_id
- run_sequence
- final_stop_id
- destination_name
- status
- express_stop_count
- vehicle_position
- vehicle_descriptor

Examples

```r
## Not run:
run_information(100)

## End(Not run)
```

search_outlets(  
  search_term,
  latitude = NULL,
  longitude = NULL,
  max_distance = NULL,
  route_types = NULL,
  user_id = determine_user_id(),
  api_key = determine_api_key()
)

Description

This function will search outlets in which the search term can be found in either the outlet name, outlet business or outlet suburb. The search is case-insensitive. The search term must contain at least 3 characters, and cannot be numeric.

Usage

```r
search_outlets(  
  search_term,
  latitude = NULL,
  longitude = NULL,
  max_distance = NULL,
  route_types = NULL,
  user_id = determine_user_id(),
  api_key = determine_api_key()
)
```
Arguments

- **search_term**: Character. Term used to perform search.
- **latitude**: Numeric. Latitude in decimal degrees. For example, Flinders Street Station is at approximately -37.8183 latitude.
- **longitude**: Numeric. Longitude in decimal degrees. For example, Flinders Street Station is at approximately 144.9671 longitude.
- **max_distance**: Integer. Optionally filter by maximum distance from the given location, in metres.
- **route_types**: Integer or character vector. Optionally filter by a vector of route types. A route type can be provided either as a non-negative integer code, or as a character: "Tram", "Train", "Bus", "Vline" or "Night Bus". Character inputs are not case-sensitive. Use the `route_types` function to extract a vector of all route types.
- **user_id**: Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to `?ptvapi` for more details.
- **api_key**: Character. An API key, with dashes, provided by Public Transport Victoria. Refer to `?ptvapi` for more details.

Value

A tibble with the following columns:

- `outlet_slid_spid`
- `outlet_name`
- `outlet_business`
- `outlet_latitude`
- `outlet_longitude`
- `outlet_suburb`
- `outlet_postcode`
- `outlet_business_hour_mon`
- `outlet_business_hour_tue`
- `outlet_business_hour_wed`
- `outlet_business_hour_thu`
- `outlet_business_hour_fri`
- `outlet_business_hour_sat`
- `outlet_business_hour_sun`
- `outlet_notes`
### Examples

```r
## Not run:
search_outlets("St Kilda")
search_outlets("St Kilda", route_types = c("Train", "Tram"))
search_outlets("St Kilda", route_types = 1)

search_outlets(
  "St Kilda",
  latitude = -37.867647,
  longitude = 144.976809
)
search_outlets(
  "St Kilda",
  latitude = -37.867647,
  longitude = 144.976809,
  max_distance = 100
)

## End(Not run)
```

---

**search_routes**

*Search for routes using text*

**Description**

This function will search routes in which the search term can be found in one of many fields, such as `route_id`, `route_gtfs_id`, or `route_name`. The search is case-insensitive. Unlike `search_stops` and `search_outlets`, this function supports searching for numerics, and has no minimum character requirement for `search_term`.

**Usage**

```r
search_routes(
  search_term,
  latitude = NULL,
  longitude = NULL,
  max_distance = NULL,
  route_types = NULL,
  user_id = determine_user_id(),
  api_key = determine_api_key()
)
```

**Arguments**

- **search_term** Character. Term used to perform search.
- **latitude** Numeric. Latitude in decimal degrees. For example, Flinders Street Station is at approximately -37.8183 latitude.
longitude Numeric. Longitude in decimal degrees. For example, Flinders Street Station is at approximately 144.9671 longitude.

max_distance Integer. Optionally filter by maximum distance from the given location, in metres.

event_types Integer or character vector. Optionally filter by a vector of event types. A route type can be provided either as a non-negative integer code, or as a character: "Tram", "Train", "Bus", "Vline" or "Night Bus". Character inputs are not case-sensitive. Use the event_types function to extract a vector of all event types.

user_id Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to ?ptvapi for more details.

api_key Character. An API key, with dashes, provided by Public Transport Victoria. Refer to ?ptvapi for more details.

Value

A tibble of routes, with the following columns:

- route_id
- route_gtfs_id
- route_name
- route_type
- route_number
- service_status
- service_status_timestamp

Examples

```r
## Not run:
search_routes("Pakenham")
search_routes("Pakenham", route_types = c("Train", "Tram"))
search_routes("Pakenham", route_types = 1)

search_routes(
  "Pakenham",
  latitude = -38.077877,
  longitude = 145.484751
)
search_routes(
  "Pakenham",
  latitude = -38.077877,
  longitude = 145.484751,
  max_distance = 100
)
```

## End(Not run)
Description

This function will search stops in which the search term can be found in either the stop name or the stop suburb. The search is case-insensitive. The search term must contain at least 3 characters, and cannot be numeric.

Usage

search_stops(
  search_term,
  latitude = NULL,
  longitude = NULL,
  max_distance = NULL,
  route_types = NULL,
  user_id = determine_user_id(),
  api_key = determine_api_key()
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>search_term</td>
<td>Character. Term used to perform search.</td>
</tr>
<tr>
<td>latitude</td>
<td>Numeric. Latitude in decimal degrees. For example, Flinders Street Station is at approximately -37.8183 latitude.</td>
</tr>
<tr>
<td>longitude</td>
<td>Numeric. Longitude in decimal degrees. For example, Flinders Street Station is at approximately 144.9671 longitude.</td>
</tr>
<tr>
<td>max_distance</td>
<td>Integer. Optionally filter by maximum distance from the given location, in metres.</td>
</tr>
<tr>
<td>route_types</td>
<td>Integer or character vector. Optionally filter by a vector of route types. A route type can be provided either as a non-negative integer code, or as a character: &quot;Tram&quot;, &quot;Train&quot;, &quot;Bus&quot;, &quot;Vline&quot; or &quot;Night Bus&quot;. Character inputs are not case-sensitive. Use the route_types function to extract a vector of all route types.</td>
</tr>
<tr>
<td>user_id</td>
<td>Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to ?ptvapi for more details.</td>
</tr>
<tr>
<td>api_key</td>
<td>Character. An API key, with dashes, provided by Public Transport Victoria. Refer to ?ptvapi for more details.</td>
</tr>
</tbody>
</table>

Value

A tibble with the following columns:

- stop_id
- stop_name
stops_nearby

• stop_suburb
• route_type
• stop_sequence
• stop_latitude
• stop_longitude
• disruption_ids

Examples

```r
## Not run:
search_stops("Ascot Vale")
search_stops("Ascot Vale", route_types = c("Train", "Tram"))
search_stops("Ascot Vale", route_types = 1)

search_stops(
  "Ascot Vale",
  latitude = -37.774240,
  longitude = 144.915518
)
search_stops(
  "Ascot Vale",
  latitude = -37.774240,
  longitude = 144.915518,
  max_distance = 100
)

## End(Not run)
```

stops_nearby  Stops near a given location

Description

Stops near a given location

Usage

```r
stops_nearby(
  latitude,
  longitude,
  max_distance = NULL,
  route_types = NULL,
  user_id = determine_user_id(),
  api_key = determine_api_key()
)
```
**Argument**

- **latitude** Numeric. Latitude in decimal degrees. For example, Flinders Street Station is at approximately -37.8183 latitude.
- **longitude** Numeric. Longitude in decimal degrees. For example, Flinders Street Station is at approximately 144.9671 longitude.
- **max_distance** Integer. Optionally filter by maximum distance from the given location, in metres.
- **route_types** Integer or character vector. Optionally filter by a vector of route types. A route type can be provided either as a non-negative integer code, or as a character: "Tram", "Train", "Bus", "Vline" or "Night Bus". Character inputs are not case-sensitive. Use the route_types function to extract a vector of all route types.
- **user_id** Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to ?ptvapi for more details.
- **api_key** Character. An API key, with dashes, provided by Public Transport Victoria. Refer to ?ptvapi for more details.

**Value**

A tibble with the following columns:

- stop_id
- stop_name
- stop_suburb
- route_type
- stop_sequence
- stop_latitude
- stop_longitude
- disruption_ids

**Examples**

```r
## Not run:
starts_nearby(latitude = -37.8183, longitude = 144.9671)
starts_nearby(latitude = -37.8183, longitude = 144.9671, max_distance = 1000)
starts_nearby(
  latitude = -37.8183,
  longitude = 144.9671,
  route_types = c("Train", "Tram")
)

starts_nearby(
  latitude = -37.8183,
  longitude = 144.9671,
  route_types = 0
)

## End(Not run)
```
Stops on a given route and route type

Usage

```r
stops_on_route(
  route_id,
  route_type,
  direction_id = NULL,
  user_id = determine_user_id(),
  api_key = determine_api_key()
)
```

Arguments

- **route_id**: Integer. These can be listed and described with the `routes` function.
- **route_type**: A route type which can be provided either as a non-negative integer code, or as a character: "Tram", "Train", "Bus", "Vline" or "Night Bus". Character inputs are not case-sensitive. Use the `route_types` function to extract a vector of all route types.
- **direction_id**: Optionally filter by a direction ID. These can be obtained with the `directions_on_route` function.
- **user_id**: Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to `?ptvapi` for more details.
- **api_key**: Character. An API key, with dashes, provided by Public Transport Victoria. Refer to `?ptvapi` for more details.

Value

A tibble with the following columns:

- stop_id
- stop_name
- stop_suburb
- route_type
- stop_sequence
- stop_latitude
- stop_longitude
- disruption_ids
stop_information

Examples

```r
## Not run:
stops_on_route(6, route_type = "Train")
stops_on_route(6, route_type = 0)

## End(Not run)
```

---

**stop_information**  
*Information for a given stop (metropolitan and V/Line stations only)*

**Description**

This function can be used when integer stop ID is already known. This can be searched for with either the `stops_on_route` or `stops_nearby` functions.

**Usage**

```r
stop_information(
  stop_id,  
  route_type,  
  user_id = determine_user_id(),  
  api_key = determine_api_key()
)
```

**Arguments**

- **stop_id**: Integer stop ID.
- **route_type**: A route type which can be provided either as a non-negative integer code, or as a character: "Tram", "Train", "Bus", "Vline" or "Night Bus". Character inputs are not case-sensitive. Use the `route_types` function to extract a vector of all route types.
- **user_id**: Integer or character. A user ID or devid provided by Public Transport Victoria. Refer to `?ptvapi` for more details.
- **api_key**: Character. An API key, with dashes, provided by Public Transport Victoria. Refer to `?ptvapi` for more details.

**Value**

A single-row tibble with the following columns:

- `stop_id`
- `stop_name`
- `route_type`
- `station_details_id`
• station_type
• station_description
• point_id
• mode_id
• operating_hours
• flexible_stop_opening_hours
• stop_contact
• stop_ticket
• stop_location
• stop_amenities
• stop_accessibility
• stop_staffing
• disruption_ids
# Index

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>departures</td>
<td>2</td>
</tr>
<tr>
<td>directions</td>
<td>4</td>
</tr>
<tr>
<td>directions_on_route</td>
<td>6</td>
</tr>
<tr>
<td>disruption_information</td>
<td>11</td>
</tr>
<tr>
<td>disruption_modes</td>
<td>12</td>
</tr>
<tr>
<td>disruptions</td>
<td>7</td>
</tr>
<tr>
<td>disruptions_at_stop</td>
<td>8</td>
</tr>
<tr>
<td>disruptions_on_route</td>
<td>10</td>
</tr>
<tr>
<td>fare_estimate</td>
<td>13</td>
</tr>
<tr>
<td>outlets</td>
<td>15</td>
</tr>
<tr>
<td>outlets_nearby</td>
<td>16</td>
</tr>
<tr>
<td>patterns</td>
<td>18</td>
</tr>
<tr>
<td>ptvapi</td>
<td>19</td>
</tr>
<tr>
<td>route_information</td>
<td>21</td>
</tr>
<tr>
<td>route_types</td>
<td>22</td>
</tr>
<tr>
<td>routes</td>
<td>20</td>
</tr>
<tr>
<td>run_information</td>
<td>24</td>
</tr>
<tr>
<td>runs_on_route</td>
<td>23</td>
</tr>
<tr>
<td>search_outlets</td>
<td>25</td>
</tr>
<tr>
<td>search_routes</td>
<td>27</td>
</tr>
<tr>
<td>search_stops</td>
<td>29</td>
</tr>
<tr>
<td>stop_information</td>
<td>33</td>
</tr>
<tr>
<td>stops_nearby</td>
<td>30</td>
</tr>
<tr>
<td>stops_on_route</td>
<td>32</td>
</tr>
</tbody>
</table>