Package ‘activatr’

May 2, 2023

**Type** Package

**Title** Utilities for Parsing and Plotting Activities

**Version** 0.1.1

**Description** This contains helpful functions for parsing, managing, plotting, and visualizing activities, most often from GPX (GPS Exchange Format) files recorded by GPS devices. It allows easy parsing of the source files into standard R data formats, along with functions to compute derived data for the activity, and to plot the activity in a variety of ways.

**License** MIT + file LICENSE

**URL** https://github.com/dschafer/activatr

**BugReports** https://github.com/dschafer/activatr/issues

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 4.0.0)

**Imports** dplyr (>= 1.0.0), geosphere (>= 1.5), ggmap (>= 3.0.0), glue (>= 1.4.0), httr (>= 1.4.0), lubridate (>= 1.7.0), magrittr (>= 2.0.0), rlang (>= 0.4.0), tibble (>= 3.0.0), slider (>= 0.3.0), xml2 (>= 1.3.2)

**RoxygenNote** 7.2.3

**Suggests** covr (>= 3.5.0), lintr (>= 3.0.0), ggplot2 (>= 3.4.0), knitr (>= 1.30), mockery (>= 0.4.2), rmarkdown (>= 2.6), roxygen2 (>= 7.1.0), styler (>= 1.9.0), testthat (>= 3.0.0)

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Daniel Schafer [aut, cph, cre]

**Maintainer** Daniel Schafer <dan.schafer@gmail.com>

**Repository** CRAN

**Date/Publication** 2023-05-01 22:00:02 UTC
R topics documented:

activatr .............................................................. 2
act_tbl ................................................................. 2
get_ggmap_from_df ...................................................... 3
localize_to_time_zone ................................................. 4
mutate_with_distance .................................................. 4
mutate_with_speed ..................................................... 5
pace_formatter .......................................................... 6
parse_gpx ............................................................... 6
parse_tcx ............................................................... 7
running_example_ggmap ............................................... 9
speed_to_mile_pace .................................................... 9

Index 10

activatr  activatr: Utilities for Parsing and Plotting Activities

Description

This contains helpful functions for parsing, managing, plotting, and visualizing activities, most often from GPX (GPS Exchange Format) files recorded by GPS devices. It allows easy parsing of the source files into standard R data formats, along with functions to compute derived data for the activity, and to plot the activity in a variety of ways.

act_tbl

Creates an activatr tibble, abbreviated act_tbl.

Description

act_tbl takes a tibble and returns an act_tbl object.

summary(act_tbl) returns a tibble with canonical information about the activity. Designed to allow for easy creation of activity summary data sets by mapping summary over each act_tbl then using bind_rows to create a complete data set.

Usage

act_tbl(x)

## S3 method for class 'act_tbl'
summary(object, full = FALSE, units = c("imperial", "metric"), ...)
get_ggmap_from_df

Get a ggmap object for a given Activatr DF.

Description

Note that since this calls ggmap::get_googlemap, you must have previously called ggmap::register_google to register an API key.

Usage

get_ggmap_from_df(df, ...)

Arguments

df A Activatr DF: a tibble from parse_gpx or parse_tcx.
...
Additional arguments to pass to ggmap::get_googlemap.

Value

A ggmap object, the result of calling ggmap::get_googlemap, but with the correct center and size to include the entire data frame.
mutate_with_distance

Augments a Activatr DF with a distance variable.

Description
This returns a mutated Activatr DF with a new column representing distance, in meters. The distance is determined by looking at the lat/lon delta between the current point and the previous point: hence, it is always NA for the first row in the data frame.

Usage
mutate_with_distance(df, method = c("2D", "3D"), lead = 0, lag = 1)

mutate_with_time

Uses Google Maps Time Zone APIs to localize the time zone.

Description
This returns a mutated Activatr DF with the time column updated to reflect the correct time zone, using the Google Maps Time Zone APIs.

Usage
localize_to_time_zone(df)

Arguments
df A Activatr DF: a tibble from parse_gpx or parse_tcx.

Details
Note that to avoid overuse of the API, this does an "approximation", in that it finds the correct time zone for the first point in the data frame, and assumes all points in that data frame use that time zone. Runs between time zones (or runs that cross daylight savings time shifts) will hence be recorded using a consistent, but not always pointwise correct, timezone.

Note that you must have previously called ggmap::register_google to register an API key before calling this.

Value
That same Activatr DF, but with the time column updated to be in the local time zone rather than UTC.
mutate_with_speed

Arguments

- **df**: A Activatr DF: a tibble from `parse_gpx` or `parse_tcx`.
- **method**: If 2D (default), ignores elevation. If 3D, includes elevation.
- **lead**: How far ahead to look for the "end" point
- **lag**: How far behind to look for the "start" point

Value

That same Activatr DF, but with a new distance column, in meters.

mutate_with_speed Augments a Activatr DF with a speed variable.

Description

This returns a mutated Activatr DF with a new column representing speed, in meters per second. The speed is determined by looking at the time difference between the current point and the previous point: hence, it is always NA for the first row in the data frame.

Usage

```
mutate_with_speed(df, method = c("2D", "3D"), lead = 0, lag = 1)
```

Arguments

- **df**: A Activatr DF: a tibble from `parse_gpx` or `parse_tcx`.
- **method**: If 2D (default), ignores elevation. If 3D, includes elevation.
- **lead**: How far ahead to look for the "end" point
- **lag**: How far behind to look for the "start" point

Value

That same Activatr DF, but with a new speed column, in meters per second.
**pace_formatter**

A formatter that takes a pace duration and returns a formatted M:SS string.

**Description**

A formatter that takes a pace duration and returns a formatted M:SS string.

**Usage**

```r
pace_formatter(pace)
```

**Arguments**

- **pace**: a lubridate duration.

**Value**

A formatted string representing the pace.

**Examples**

```r
pace_formatter(lubridate::dseconds(390))
```

---

**parse_gpx**

Parses a GPX file into a tibble.

**Description**

This parses a standard GPS Exchange Format XML (GPX) file into an act_tbl.

**Usage**

```r
parse_gpx(filename, detail = c("basic", "latlon", "advanced"), every = NA)
```

**Arguments**

- **filename**: The GPX file to parse
- **detail**: How much detail to parse from the GPX. * If "basic", the default, this will load lat / lon / ele / time. * If "latlon", it will only load lat/lon: useful for GPX files exported without time information. * If "advanced", it will load everything from basic, plus hr / cad / atemp: useful for files with HR information.
- **every**: Optional. If provided, determines how frequently points will be sampled from the file, so if 10 is provided, every tenth point will be selected. If omitted or set to 1, every point will be selected. Must be a positive integer.
parse_tcx

Value

A `act_tbl` with one row for each trackpoint in the GPX (modified by `every`), and with the columns determined by `detail`.

- **lat**: latitude, a dbl in degrees between -90 and 90
- **lon**: longitude, a dbl in degrees between -180 and 180
- **ele**: elevation, a dbl in meters
- **time**: time, a dttm representing the time of the point
- **hr**: heart rate, an int in beats per minute
- **cad**: cadence, an int in one-foot steps per minute

Additionally, attributes are set on the tibble containing top level data from the GPX. Each of these will be NA when not provided in the file.

- **filename**: the filename this was parsed from. This is always present, and is always the value of the `filename` argument.
- **time**: time, a dttm representing the time of the GPX
- **title**: title, a chr
- **desc**: description, a chr
- **type**: type, a chr

See Also

https://www.topografix.com/gpx.asp

Examples

```r
running_file <- system.file(  
  "extdata",  
  "running_example.gpx.gz",  
  package = "activatr"  
)
running_df <- parse_gpx(running_file)
```

---

### parse_tcx

Parses a TCX file into a tibble.

**Description**

This parses a standard Training Center XML (TCX) file into an `act_tbl`.

**Usage**

```r
parse_tcx(filename, detail = c("basic", "latlon", "advanced"), every = NA)
```
parse_tcx

Arguments

filename  The TCX file to parse

detail  How much detail to parse from the TCX. * If "basic", the default, this will load
        lat / lon / ele / time. * If "latlon", it will only load lat/lon: useful for TCX
        files exported without time information. * If "advanced", it will load everything
        from basic, plus hr / cad / atemp: useful for files with HR information.

every  Optional. If provided, determines how frequently points will be sampled from
        the file, so if 10 is provided, every tenth point will be selected. If omitted or set
        to 1, every point will be selected. Must be a positive integer.

Value

A act_tbl with one row for each trackpoint in the TCX (modified by every), and with the columns
determined by detail.

lat  latitude, a dbl in degrees between -90 and 90
lon  longitude, a dbl in degrees between -180 and 180
ele  elevation, a dbl in meters
time  time, a dttm representing the time of the point
hr  heart rate, an int in beats per minute
cad  cadence, an int in one-foot steps per minute

Additionally, attributes are set on the tibble containing top level data from the TCX. Each of these
will be NA when not provided in the file.

filename  the filename this was parsed from. This is always present is always the value of
the filename argument.
time  time, a dttm representing the time of the TCX
type  type, a chr

See Also

https://en.wikipedia.org/wiki/Training_Center_XML

Examples

running_file <- system.file("extdata",
    "running_example.tcx.gz",
    package = "activatr"
)
running_df <- parse_gpx(running_file)
**running_example_ggmap**

The result of calling `get_ggmap_from_df` on `running_example`.

**Description**

This is the result of running:

**Usage**

```r
running_example_ggmap
```

**Format**

An object of class `ggmap` (inherits from `raster`) with 1280 rows and 1280 columns.

**Details**

```r
"' running_file <- system.file("extdata", "running_example.gpx", package = "activatr") running_df <- parse_gpx(running_file) running_example_ggmap <- get_ggmap_from_df(running_df) "'
```

except using that in vignettes or examples is hard, because `get_ggmap_from_df` requires an api key be passed to `ggmap`. So this is the result of running that with a valid API key.

**speed_to_mile_pace**

Converts a speed (in meters per second) to a mile pace.

**Description**

Converts a speed (in meters per second) to a mile pace

**Usage**

```r
speed_to_mile_pace(speed)
```

**Arguments**

- `speed` a vector of speed values in meters per second, as from `mutate_with_speed`.

**Value**

A corresponding vector of lubridate durations, representing the mile pace.

**Examples**

```r
speed_to_mile_pace(1)
```
Index

* datasets
  running_example_ggmap, 9

act_tbl, 2
activatr, 2
get_ggmap_from_df, 3
localize_to_time_zone, 4
mutate_with_distance, 4
mutate_with_speed, 5
pace_formatter, 6
parse_gpx, 6
parse_tcx, 7
running_example_ggmap, 9
speed_to_mile_pace, 9
summary.act_tbl(act_tbl), 2