Package 'Athlytics'

July 21, 2025

Title Advanced Sports Performance Analysis for 'Strava' Data

Version 0.1.2

Author Ang [aut, cre]

Maintainer Ang <ang@hezhiang.com>

Description Advanced sports performance analysis and modeling for activity data retrieved from 'Strava'. This package focuses on applying established sports science models and statistical methods to gain deeper insights into training load, performance prediction, recovery status, and identifying key performance factors, extending basic data analysis capabilities.

License MIT + file LICENSE

URL https://hezhiang.com/Athlytics/,
https://github.com/HzaCode/Athlytics

BugReports https://github.com/HzaCode/Athlytics/issues

Encoding UTF-8

LazyData true

Depends R (>= 3.6.0)

Imports dplyr (>= 1.0.0), ggplot2, httr, jsonlite, lubridate, purrr, rlang (>= 0.4.0), rStrava, tidyr, viridis, zoo

Suggests devtools, knitr, pkgdown, rmarkdown, testthat (>= 3.0.0), mockery

VignetteBuilder knitr

RoxygenNote 7.3.2

NeedsCompilation no

Repository CRAN

Date/Publication 2025-05-16 09:40:23 UTC

Contents

`athlytics_sample_acwr` *Sample ACWR Data for Athlytics*

Description

A dataset containing pre-calculated Acute:Chronic Workload Ratio (ACWR) and related metrics, derived from simulated Strava data. Used in examples and tests.

Usage

```
athlytics_sample_acwr
```

Format

A tibble with X rows and 5 variables:

date Date of the metrics, as a Date object.

atl Acute Training Load, as a numeric value.

ctl Chronic Training Load, as a numeric value.

acwr Acute:Chronic Workload Ratio, as a numeric value.

acwr_smooth Smoothed ACWR, as a numeric value.

Source

Simulated data generated for package examples.

athlytics_sample_decoupling *Sample Aerobic Decoupling Data for Athlytics*

Description

A dataset containing pre-calculated aerobic decoupling percentages, derived from simulated Strava data. Used in examples and tests.

Usage

```
athlytics_sample_decoupling
```

Format

A tibble with X rows and 2 variables:

date Date of the activity, as a Date object.

decoupling Calculated decoupling percentage, as a numeric value.

Source

Simulated data generated for package examples.

athlytics_sample_ef *Sample Efficiency Factor (EF) Data for Athlytics*

Description

A dataset containing pre-calculated Efficiency Factor (EF) values, derived from simulated Strava data. Used in examples and tests.

Usage

```
athlytics_sample_ef
```

Format

A data.frame with X rows and 3 variables:

date Date of the activity, as a Date object.

activity_type Type of activity (e.g., "Run", "Ride"), as a character string.

ef_value Calculated Efficiency Factor, as a numeric value.

Source

Simulated data generated for package examples.

athlytics_sample_exposure *Sample Training Load Exposure Data for Athlytics*

Description

This dataset contains daily training load, ATL, CTL, and ACWR, derived from simulated Strava data. Used in examples and tests, particularly for ‘plot_exposure‘.

Usage

```
athlytics_sample_exposure
```

Format

A tibble with X rows and 5 variables:

date Date of the metrics, as a Date object.

daily_load Calculated daily training load, as a numeric value.

ctl Chronic Training Load, as a numeric value.

atl Acute Training Load, as a numeric value.

acwr Acute:Chronic Workload Ratio, as a numeric value.

Source

Simulated data generated for package examples.

athlytics_sample_pbs *Sample Personal Bests (PBs) Data for Athlytics*

Description

A dataset containing pre-calculated Personal Best (PB) times for various distances, derived from simulated Strava data. Used in examples and tests.

Usage

```
athlytics_sample_pbs
```

Format

A tibble with X rows and 10 variables:

activity_id ID of the activity where the effort occurred, as a character string.

activity_date Date and time of the activity, as a POSIXct object.

distance Target distance in meters for the best effort, as a numeric value.

elapsed_time Elapsed time for the effort in seconds, as a numeric value.

moving_time Moving time for the effort in seconds, as a numeric value.

time_seconds Typically the same as elapsed_time for best efforts, in seconds, as a numeric value.

cumulative_pb_seconds The personal best time for that distance up to that date, in seconds, as a numeric value.

is_pb Logical, TRUE if this effort set a new personal best.

distance_label Factor representing the distance (e.g., "1k", "5k").

time_period Formatted time of the effort, as a Period object from lubridate.

Source

Simulated data generated for package examples.

`calculate_acwr` *Calculate ACWR Data*

Description

Calculates the Acute:Chronic Workload Ratio (ACWR) from Strava data.

Usage

```
calculate_acwr(
  stoken,
  activity_type = NULL,
  load_metric = "duration_mins",
  acute_period = 7,
  chronic_period = 28,
  start_date = NULL,
  end_date = NULL,
  user_ftp = NULL,
  user_max_hr = NULL,
  user_resting_hr = NULL,
  smoothing_period = 7
)
```

Arguments

stoken	A valid Strava token from ‘rStrava::strava_oauth()‘.
activity_type	Optional. Filter activities by type (e.g., "Run", "Ride"). Default ‘NULL‘ includes all types.
load_metric	Method for calculating daily load (e.g., "duration_mins", "distance_km", "tss", "hrss"). Default "duration_mins".
acute_period	Days for the acute load window (e.g., 7).
chronic_period	Days for the chronic load window (e.g., 28). Must be greater than ‘acute_period‘.
start_date	Optional. Analysis start date (YYYY-MM-DD string or Date). Defaults to one year ago.
end_date	Optional. Analysis end date (YYYY-MM-DD string or Date). Defaults to today.
user_ftp	Required if ‘load_metric = "tss"‘. Your Functional Threshold Power.
user_max_hr	Required if ‘load_metric = "hrss"‘. Your maximum heart rate.
user_resting_hr	Required if ‘load_metric = "hrss"‘. Your resting heart rate.
smoothing_period	Days for smoothing the ACWR using a rolling mean (e.g., 7). Default 7.

Details

Calculates daily load, ATL, CTL, raw ACWR, and smoothed ACWR from Strava activities.

Provides data for ‘plot_acwr‘. Fetches extra prior data for accurate initial CTL. Fetching can be slow for long periods.

Value

A data frame with columns: ‘date‘, ‘atl‘ (Acute Load), ‘ctl‘ (Chronic Load), ‘acwr‘ (raw ACWR), and ‘acwr_smooth‘ (smoothed ACWR) for the specified date range.

Examples

```
# Example using simulated data (Note: sample data is pre-calculated, shown for demonstration)
data(Athlytics_sample_data)
if (!is.null(athlytics_sample_acwr)) {
  print(head(athlytics_sample_acwr))
}

## Not run:
# Example using real data (requires authentication and app setup)
# Replace with your actual app_name, client_id, and secret or ensure stoken is pre-configured
# stoken <- rStrava::strava_oauth(
#   app_name = "YOUR_APP_NAME",
#   client_id = "YOUR_CLIENT_ID",
#   client_secret = "YOUR_SECRET",
#   cache = TRUE
# )
# if (interactive() && exists("stoken")) { # Proceed if stoken is available
```

```
#   # Calculate ACWR for Runs (using duration)
#   run_acwr <- calculate_acwr(stoken = stoken, activity_type = "Run",
#                              load_metric = "duration_mins")
#   print(tail(run_acwr))
#
#   # Calculate ACWR for Rides (using TSS, requires FTP)
#   ride_acwr_tss <- calculate_acwr(stoken = stoken, activity_type = "Ride",
#                                   load_metric = "tss", user_ftp = 280)
#   print(tail(ride_acwr_tss))
# } else {
#  message("Strava token not available or not in interactive session, skipping real data example.")
# }

## End(Not run)
```

`calculate_decoupling` *Calculate Aerobic Decoupling*

Description

Calculates aerobic decoupling for Strava activities.

Usage

```
calculate_decoupling(
  stoken,
  activity_type = c("Run", "Ride"),
  decouple_metric = c("Pace_HR", "Power_HR"),
  start_date = NULL,
  end_date = NULL,
  min_duration_mins = 45,
  max_activities = 50,
  stream_df = NULL
)
```

Arguments

`stoken`	A valid Strava token from ‘rStrava::strava_oauth()‘.
`activity_type`	Type(s) of activities to analyze (e.g., "Run", "Ride").
`decouple_metric`	Basis for calculation: "Pace_HR" or "Power_HR".
`start_date`	Optional. Analysis start date (YYYY-MM-DD string or Date). Defaults to one year ago.
`end_date`	Optional. Analysis end date (YYYY-MM-DD string or Date). Defaults to today.
`min_duration_mins`	Minimum activity duration (minutes) to include. Default 45.
`max_activities`	Maximum number of recent activities to analyze. Default 50.

`stream_df` Optional. A pre-fetched data frame for a *single* activity's stream. If provided, calculates decoupling for this data directly, ignoring 'stoken' and other fetching parameters. Must include columns: 'time', 'heartrate', and either 'velocity_smooth'/'distance' (for Pace_HR) or 'watts' (for Power_HR).

Details

Calculates aerobic decoupling (HR drift relative to pace/power) using detailed Strava activity streams. Fetching streams via API can be slow.

Provides data for 'plot_decoupling'. Compares output/HR efficiency between first and second halves of activities. Positive values indicate HR drift. Fetching streams via API using 'httr' is slow and subject to rate limits.

Value

Returns a data frame with 'date' and 'decoupling' [a single numeric decoupling value if 'stream_df' is provided.

Examples

```
# Example using simulated data
data(Athlytics_sample_data)
print(head(athlytics_sample_decoupling))

## Not run:
# Example using real data (requires authentication)
# To authenticate (replace with your details):
# stoken <- rStrava::strava_oauth(app_name = "YOUR_APP",
#                                 client_id = "YOUR_ID",
#                                 client_secret = "YOUR_SECRET",
#                                 cache = TRUE)

# Calculate Pace/HR decoupling for recent Runs (limit to 10 activities for speed)
# Ensure stoken is defined and valid before running this part
# run_decoupling <- calculate_decoupling(
#     stoken = stoken,
#     activity_type = "Run",
#     decouple_metric = "Pace_HR",
#     max_activities = 10
# )
# print(tail(run_decoupling))

## End(Not run)
```

---

calculate_ef *Calculate Efficiency Factor (EF) Data*

Description

Calculates Efficiency Factor (Pace/HR or Power/HR) from Strava activities.

Usage

```
calculate_ef(
  stoken,
  activity_type = c("Run", "Ride"),
  ef_metric = c("Pace_HR", "Power_HR"),
  start_date = NULL,
  end_date = NULL,
  min_duration_mins = 20
)
```

Arguments

stoken	A valid Strava token object obtained using rStrava::strava_oauth().
activity_type	Character vector or single string specifying activity type(s).
ef_metric	Character string specifying the EF metric ("Pace_HR" or "Power_HR").
start_date	Optional start date (YYYY-MM-DD string or Date object). Defaults to one year ago.
end_date	Optional end date (YYYY-MM-DD string or Date object). Defaults to today.
min_duration_mins	Numeric, minimum activity duration in minutes. Default 20.

Details

Fetches activity summaries and calculates EF (output/HR) for each. Provides the data used by ‘plot_ef‘.

Value

A data frame with columns: date, activity_type, ef_value.

Examples

```
# Example using simulated data
data(Athlytics_sample_data)
print(head(athlytics_sample_ef))

## Not run:
# Example using real data (requires authentication)
# To authenticate (replace with your details):
# stoken <- rStrava::strava_oauth(app_name = "YOUR_APP",
#                                 client_id = "YOUR_ID",
#                                 client_secret = "YOUR_SECRET",
#                                 cache = TRUE)

# Calculate Pace/HR efficiency factor for Runs
# Ensure stoken is defined and valid before running this part
# ef_data_run <- calculate_ef(stoken = stoken, activity_type = "Run", ef_metric = "Pace_HR")
# print(tail(ef_data_run))
```

```
# Calculate Power/HR efficiency factor for Rides
# Ensure stoken is defined and valid before running this part
# ef_data_ride <- calculate_ef(stoken = stoken, activity_type = "Ride", ef_metric = "Power_HR")
# print(tail(ef_data_ride))

## End(Not run)
```

`calculate_exposure` *Calculate Training Load Exposure (ATL, CTL, ACWR)*

Description

Calculates training load metrics like ATL, CTL, and ACWR from Strava data.

Usage

```
calculate_exposure(
  stoken,
  activity_type = c("Run", "Ride", "VirtualRide", "VirtualRun"),
  load_metric = "duration_mins",
  acute_period = 7,
  chronic_period = 42,
  user_ftp = NULL,
  user_max_hr = NULL,
  user_resting_hr = NULL,
  end_date = NULL
)
```

Arguments

`stoken`	A valid Strava token from ‘rStrava::strava_oauth()‘.
`activity_type`	Type(s) of activities to include (e.g., "Run", "Ride"). Default includes common run/ride types.
`load_metric`	Method for calculating daily load (e.g., "duration_mins", "distance_km", "tss", "hrss"). Default "duration_mins".
`acute_period`	Days for the acute load window (e.g., 7).
`chronic_period`	Days for the chronic load window (e.g., 42). Must be greater than ‘acute_period‘.
`user_ftp`	Required if ‘load_metric = "tss"‘. Your Functional Threshold Power.
`user_max_hr`	Required if ‘load_metric = "hrss"‘. Your maximum heart rate.
`user_resting_hr`	Required if ‘load_metric = "hrss"‘. Your resting heart rate.
`end_date`	Optional. Analysis end date (YYYY-MM-DD string or Date). Defaults to today. The analysis period covers the ‘chronic_period‘ days ending on this date.

Details

Calculates daily load, ATL, CTL, and ACWR from Strava activities based on the chosen metric and periods.

Provides data for 'plot_exposure'. Fetches extra prior data for accurate initial CTL. Requires FTP/HR parameters for TSS/HRSS metrics.

Value

A data frame with columns: 'date', 'daily_load', 'atl' (Acute Load), 'ctl' (Chronic Load), and 'acwr' (Acute:Chronic Ratio) for the analysis period.

Examples

```
# Example using simulated data
data(Athlytics_sample_data)
print(head(athlytics_sample_exposure))

## Not run:
# Example using real data (requires authentication)
# Replace YOUR_APP, YOUR_ID, YOUR_SECRET with your Strava application details
# stoken_example <- rStrava::strava_oauth(app_name = "YOUR_APP",
#                                 client_id = "YOUR_ID",
#                                 client_secret = "YOUR_SECRET",
#                                 cache = TRUE)

# Calculate training load for Rides using TSS
# Ensure stoken_example is defined and valid before running this part
# if (exists("stoken_example") && inherits(stoken_example, "Token2.0")) {
#   ride_exposure_tss <- calculate_exposure(
#     stoken = stoken_example,
#     activity_type = "Ride",
#     load_metric = "tss",
#     user_ftp = 280,
#     acute_period = 7,
#     chronic_period = 28
#   )
#   print(head(ride_exposure_tss))
#
#   # Calculate training load for Runs using HRSS
#   run_exposure_hrss <- calculate_exposure(
#       stoken = stoken_example,
#       activity_type = "Run",
#       load_metric = "hrss",
#       user_max_hr = 190,
#       user_resting_hr = 50
#   )
#   print(tail(run_exposure_hrss))
# } else {
#  message("stoken_example not created. Skipping real data examples for calculate_exposure.")
# }
```

```
## End(Not run)
```

`calculate_pbs` *Calculate Personal Bests (PBs)*

Description

Finds personal best times for specified distances from Strava activities.

Usage

```
calculate_pbs(
  stoken,
  activity_type = "Run",
  distance_meters,
  max_activities = 500,
  date_range = NULL
)
```

Arguments

`stoken`	A valid Strava token from ‘rStrava::strava_oauth()‘.
`activity_type`	Type(s) of activities to search for PBs (e.g., "Run"). Note: Current logic relies on Strava’s ‘best_efforts‘, primarily available for Runs.
`distance_meters`	Numeric vector of distances (in meters) to find PBs for (e.g., ‘c(1000, 5000, 10000)‘).
`max_activities`	Maximum number of recent activities to check. Default 500. Reducing this can speed up the process and help avoid API rate limits.
`date_range`	Optional. Filter activities within a date range ‘c("YYYY-MM-DD", "YYYY-MM-DD")‘.

Details

Fetches detailed activity data, extracts Strava’s ’best efforts’, and calculates cumulative PBs for specified distances.

Provides data for ‘plot_pbs‘. Processes activities chronologically. Fetching detailed data is slow due to API limits (includes 1s delay per activity).

Value

A data frame containing all found best efforts for the specified distances. Includes columns: ‘activity_id‘, ‘activity_date‘, ‘distance‘, ‘time_seconds‘ (elapsed time), ‘cumulative_pb_seconds‘ (the PB for that distance as of that date), ‘is_pb‘ (TRUE if this effort set a new PB), ‘distance_label‘ (e.g., "5k"), and ‘time_period‘ (formatted time).

Examples

```
# Example using simulated data
data(Athlytics_sample_data)
print(head(athlytics_sample_pbs))

## Not run:
# Example using real data (requires authentication and YOUR credentials)
# NOTE: The following rStrava::strava_oauth call is a placeholder.
# You MUST replace "YOUR_APP_NAME", "YOUR_CLIENT_ID", "YOUR_CLIENT_SECRET"
# with your actual Strava API credentials for this to work.
# This is a placeholder and will likely require user interaction or fail
# if not properly configured with actual credentials.
# For R CMD check, the main thing is that the syntax is valid.
tryCatch({
  stoken_example <- rStrava::strava_oauth(
    app_name = "YOUR_APP_NAME_PLACEHOLDER",
    client_id = "YOUR_CLIENT_ID_PLACEHOLDER",
    client_secret = "YOUR_CLIENT_SECRET_PLACEHOLDER",
    cache = TRUE
  )

  # Proceed only if a token object is created
  if (inherits(stoken_example, "Token2.0")) {
    # Shortened message for line width
    message("Placeholder stoken obtained. Real data features may not work.")
    # Calculate PBs for 1k, 5k (limit activities for speed in example)
    pb_data <- calculate_pbs(stoken = stoken_example,
                             activity_type = "Run",
                            # Ensure activity_type matches your default or intended type
                             distance_meters = c(1000, 5000, 10000),
                             max_activities = 10) # Reduced for example speed
    if (nrow(pb_data) > 0) {
      print(head(pb_data))
      # Show only new PB records
      new_pbs <- pb_data[pb_data$is_pb, ]
      print(new_pbs)
    } else {
      message("calculate_pbs example (placeholder token) returned no PB data.")
    }
  } else {
    message("Placeholder stoken creation failed. Check rStrava setup.")
  }
}, error = function(e) {
  # Shortened error messages as well
  message("Error in rStrava::strava_oauth() example: ", e$message)
  message("This can happen with placeholder credentials.")
})

## End(Not run)
```

plot_acwr *Plot ACWR Trend*

Description

Visualizes the Acute:Chronic Workload Ratio (ACWR) trend over time.

Usage

```
plot_acwr(
  stoken,
  activity_type = NULL,
  load_metric = "duration_mins",
  acute_period = 7,
  chronic_period = 28,
  start_date = NULL,
  end_date = NULL,
  user_ftp = NULL,
  user_max_hr = NULL,
  user_resting_hr = NULL,
  smoothing_period = 7,
  highlight_zones = TRUE,
  acwr_df = NULL
)
```

Arguments

stoken	A valid Strava token from ‘rStrava::strava_oauth()‘. Required unless ‘acwr_df‘ is provided.
activity_type	Type(s) of activities to analyze (e.g., "Run", "Ride").
load_metric	Method for calculating daily load (e.g., "duration_mins", "distance_km", "tss", "hrss").
acute_period	Days for the acute load window (e.g., 7).
chronic_period	Days for the chronic load window (e.g., 28). Must be greater than ‘acute_period‘.
start_date	Optional. Analysis start date (YYYY-MM-DD string or Date). Defaults to ~1 year ago.
end_date	Optional. Analysis end date (YYYY-MM-DD string or Date). Defaults to today.
user_ftp	Required if ‘load_metric = "tss"‘ and ‘acwr_df‘ is not provided. Your Functional Threshold Power.
user_max_hr	Required if ‘load_metric = "hrss"‘ and ‘acwr_df‘ is not provided. Your maximum heart rate.
user_resting_hr	Required if ‘load_metric = "hrss"‘ and ‘acwr_df‘ is not provided. Your resting heart rate.

smoothing_period
: Days for smoothing the ACWR using a rolling mean (e.g., 7). Default 7.

highlight_zones
: Logical, whether to highlight different ACWR zones (e.g., sweet spot, high risk) on the plot. Default 'TRUE'.

acwr_df
: Optional. A pre-calculated data frame from 'calculate_acwr'. If provided, 'stoken' and other calculation parameters are ignored.

Details

Plots the ACWR trend over time. Uses pre-calculated data or calls 'calculate_acwr' (can be slow). ACWR is calculated as acute load / chronic load. A ratio of 0.8-1.3 is often considered the "sweet spot". If 'acwr_df' is not provided, calls 'calculate_acwr' first (can be slow and hit API limits).

Value

A ggplot object showing the ACWR trend.

Examples

```
# Example using simulated data
data(Athlytics_sample_data)
if (!is.null(athlytics_sample_acwr)) {
 # Ensure acwr_df is named and other necessary parameters are provided if plot_acwr expects them
  p <- plot_acwr(acwr_df = athlytics_sample_acwr)
  print(p)
}

## Not run:
# Example using real data (requires authentication)
# Please replace with your actual Strava application details for this to work.
# stoken_example <- rStrava::strava_oauth(
#   app_name = "YOUR_APP_NAME_PLACEHOLDER",
#   client_id = "YOUR_CLIENT_ID_PLACEHOLDER",
#   client_secret = "YOUR_CLIENT_SECRET_PLACEHOLDER",
#   cache = TRUE
# )

# If you have a valid stoken_example, you can then use it:
# Plot ACWR trend for Runs (using duration as load metric)
# if (exists("stoken_example") && inherits(stoken_example, "Token2.0")) {
#   plot_acwr(stoken = stoken_example,
#             activity_type = "Run",
#             load_metric = "duration_mins",
#             acute_period = 7,
#             chronic_period = 28)
#
#   # Plot ACWR trend for Rides (using TSS as load metric)
#   plot_acwr(stoken = stoken_example,
#             activity_type = "Ride",
#             load_metric = "tss",
#             user_ftp = 280)  # FTP value is required
```

```
# } else {
# message("stoken_example not created or invalid. Skipping real data example for plot_acwr.")
# }

## End(Not run)
```

---

plot_decoupling *Plot Aerobic Decoupling Trend*

Description

Visualizes the trend of aerobic decoupling over time.

Usage

```
plot_decoupling(
  stoken,
  activity_type = c("Run", "Ride"),
  decouple_metric = c("Pace_HR", "Power_HR"),
  start_date = NULL,
  end_date = NULL,
  min_duration_mins = 45,
  max_activities = 50,
  add_trend_line = TRUE,
  smoothing_method = "loess",
  decoupling_df = NULL
)
```

Arguments

stoken	A valid Strava token from ‘rStrava::strava_oauth()‘. Required unless ‘decoupling_df‘ is provided.
activity_type	Type(s) of activities to analyze (e.g., "Run", "Ride").
decouple_metric	Metric basis: "Pace_HR" or "Power_HR".
start_date	Optional. Analysis start date (YYYY-MM-DD string or Date). Defaults to ~1 year ago.
end_date	Optional. Analysis end date (YYYY-MM-DD string or Date). Defaults to today.
min_duration_mins	Minimum activity duration (minutes) to include. Default 45.
max_activities	Max number of recent activities to fetch/analyze when ‘stoken‘ is used. Default 50.
add_trend_line	Add a smoothed trend line (‘geom_smooth‘)? Default ‘TRUE‘.
smoothing_method	Smoothing method for trend line (e.g., "loess", "lm"). Default "loess".
decoupling_df	Optional. A pre-calculated data frame from ‘calculate_decoupling‘. If provided, ‘stoken‘ and other calculation parameters are ignored. Must contain ’date’ and ’decoupling’ columns.

Details

Plots the aerobic decoupling trend over time. Uses pre-calculated data or calls 'calculate_decoupling' (can be slow).

Plots decoupling percentage ((EF_1st_half - EF_2nd_half) / EF_1st_half * 100). Positive values mean HR drifted relative to output. A 5 used as reference. If 'decoupling_df' is not provided, calls 'calculate_decoupling' first (can be slow and hit API limits).

Value

A ggplot object showing the decoupling trend.

Examples

```
# Example using simulated data
data(Athlytics_sample_data)
# Explicitly name decoupling_df and provide activity_type
if (!is.null(athlytics_sample_decoupling) && nrow(athlytics_sample_decoupling) > 0) {
  p <- plot_decoupling(decoupling_df = athlytics_sample_decoupling, activity_type = "Run")
  print(p)
}

## Not run:
# Example using real data (requires authentication)
# NOTE: The following rStrava::strava_oauth call is a placeholder.
# You MUST replace placeholders with your actual Strava API credentials.
tryCatch({
  stoken_example <- rStrava::strava_oauth(
    app_name = "YOUR_APP_NAME_PLACEHOLDER",
    app_client_id = "YOUR_CLIENT_ID_PLACEHOLDER",     # CORRECTED
    app_secret = "YOUR_CLIENT_SECRET_PLACEHOLDER",  # CORRECTED
    cache = TRUE,
    app_scope = "activity:read_all" # Recommended scope
  )

  if (inherits(stoken_example, "Token2.0")) {
    message("Placeholder stoken_example created for Athlytics examples.")

    # Example 1: Plot Decoupling trend for Runs (last 6 months)
    # This first calculates the data, then plots it.
    message("Calculating decoupling for Runs (last 6 months) - may take a moment...")
    decoupling_runs_6mo <- tryCatch({
        calculate_decoupling(
            stoken = stoken_example,
            activity_type = "Run",
            decouple_metric = "Pace_HR",
            date_range = c(format(Sys.Date() - lubridate::months(6), "%Y-%m-%d"),
                           format(Sys.Date(), "%Y-%m-%d")),
            max_activities = 5 # Keep low for example
        )
    }, error = function(e_calc) {
        message(paste("Could not calculate decoupling data in example:", e_calc$message))
```

```
        return(dplyr::tibble()) # Return empty tibble on error
      })

      if (nrow(decoupling_runs_6mo) > 0 && "decoupling" %in% names(decoupling_runs_6mo)) {
      p_runs_6mo <- plot_decoupling(decoupling_df = decoupling_runs_6mo, activity_type = "Run")
        print(p_runs_6mo)
      } else {
       message("No decoupling data for Runs (last 6 months) to plot, or calculation failed.")
      }

      # Example 2: Plot Decoupling trend for Rides
      # decoupling_rides <- calculate_decoupling(
      #   stoken = stoken_example,
      #   activity_type = "Ride",
      #   decouple_metric = "Power_HR",
      #   max_activities = 5
      # )
      # if (nrow(decoupling_rides) > 0 && "decoupling" %in% names(decoupling_rides)) {
     #  p_rides <- plot_decoupling(decoupling_df = decoupling_rides, activity_type = "Ride")
      #   print(p_rides)
      # } else {
      #   message("No decoupling data for Rides to plot, or calculation failed.")
      # }

      # Example 3: Plot Decoupling trend for multiple Run types (no trend line)
      # decoupling_multi_run <- calculate_decoupling(
      #   stoken = stoken_example,
      #   activity_type = c("Run", "VirtualRun"),
      #   decouple_metric = "Pace_HR",
      #   max_activities = 5
      # )
     # if (nrow(decoupling_multi_run) > 0 && "decoupling" %in% names(decoupling_multi_run)) {
      #   p_multi_run <- plot_decoupling(
      #     decoupling_df = decoupling_multi_run,
      #     activity_type = c("Run", "VirtualRun"),
      #     add_trend_line = FALSE
      #   )
      #   print(p_multi_run)
      # } else {
      #  message("No decoupling data for multi-run types to plot, or calculation failed.")
      # }

    } else {
      message("Failed to create placeholder stoken for plot_decoupling examples.")
    }
  }, error = function(e_auth) {
    message(paste("Error during rStrava authentication in example:", e_auth$message))
  })

  ## End(Not run)
```

plot_ef *Plot Efficiency Factor (EF) Trend*

Description

Visualizes the trend of Efficiency Factor (EF) over time.

Usage

```
plot_ef(
  stoken,
  activity_type = c("Run", "Ride"),
  ef_metric = c("Pace_HR", "Power_HR"),
  start_date = NULL,
  end_date = NULL,
  min_duration_mins = 20,
  add_trend_line = TRUE,
  smoothing_method = "loess",
  ef_df = NULL
)
```

Arguments

stoken	A valid Strava token from ‘rStrava::strava_oauth()‘. Required unless ‘ef_df‘ is provided.
activity_type	Type(s) of activities to analyze (e.g., "Run", "Ride").
ef_metric	Metric to calculate: "Pace_HR" (Speed/HR) or "Power_HR" (Power/HR).
start_date	Optional. Analysis start date (YYYY-MM-DD string or Date). Defaults to ~1 year ago.
end_date	Optional. Analysis end date (YYYY-MM-DD string or Date). Defaults to today.
min_duration_mins	Minimum activity duration (minutes) to include. Default 20.
add_trend_line	Add a smoothed trend line (‘geom_smooth‘)? Default ‘TRUE‘.
smoothing_method	Smoothing method for trend line (e.g., "loess", "lm"). Default "loess".
ef_df	Optional. A pre-calculated data frame from ‘calculate_ef‘. If provided, ‘stoken‘ and other calculation parameters are ignored.

Details

Plots the Efficiency Factor (EF) trend over time. Uses pre-calculated data or calls ‘calculate_ef‘.

Plots EF (output/HR based on activity averages). An upward trend often indicates improved aerobic fitness. Points colored by activity type. If ‘ef_df‘ is not provided, calls ‘calculate_ef‘ first.

Value

A ggplot object showing the EF trend.

Examples

```
# Example using simulated data
data(Athlytics_sample_data)
# Explicitly name ef_df and provide activity_type
p <- plot_ef(ef_df = athlytics_sample_ef, activity_type = "Run")
print(p)

## Not run:
# Example using real data (requires authentication)
# stoken <- rStrava::strava_oauth("YOUR_APP_NAME",
#                                 "YOUR_APP_CLIENT_ID",
#                                 "YOUR_APP_SECRET",
#                                 cache = TRUE)

# Plot Pace/HR EF trend for Runs (last 6 months)
# plot_ef(stoken = stoken, # Replace stoken with a valid token object
#         activity_type = "Run",
#         ef_metric = "Pace_HR",
#         start_date = Sys.Date() - months(6))

# Plot Power/HR EF trend for Rides
# plot_ef(stoken = stoken, # Replace stoken with a valid token object
#         activity_type = "Ride",
#         ef_metric = "Power_HR")

# Plot Pace/HR EF trend for multiple Run types (no trend line)
# plot_ef(stoken = stoken, # Replace stoken with a valid token object
#         activity_type = c("Run", "VirtualRun"),
#         ef_metric = "Pace_HR",
#         add_trend_line = FALSE)

## End(Not run)
```

`plot_exposure` *Plot Training Load Exposure (ATL vs CTL)*

Description

Visualizes the relationship between Acute and Chronic Training Load.

Usage

```
plot_exposure(
  stoken,
  activity_type = c("Run", "Ride", "VirtualRide", "VirtualRun"),
```

```
  load_metric = "duration_mins",
  acute_period = 7,
  chronic_period = 42,
  user_ftp = NULL,
  user_max_hr = NULL,
  user_resting_hr = NULL,
  end_date = NULL,
  risk_zones = TRUE,
  exposure_df = NULL
)
```

Arguments

stoken	A valid Strava token from ‘rStrava::strava_oauth()‘. Required unless ‘exposure_df‘ is provided.
activity_type	Type(s) of activities to include (e.g., "Run", "Ride"). Default uses common types.
load_metric	Method for calculating daily load (e.g., "duration_mins", "tss", "hrss"). Default "duration_mins". See ‘calculate_exposure‘ for details on approximate TSS/HRSS calculations.
acute_period	Days for acute load window (e.g., 7).
chronic_period	Days for chronic load window (e.g., 42). Must be > ‘acute_period‘.
user_ftp	Required if ‘load_metric = "tss"‘. Your FTP.
user_max_hr	Required if ‘load_metric = "hrss"‘. Your max HR.
user_resting_hr	Required if ‘load_metric = "hrss"‘. Your resting HR.
end_date	Optional. Analysis end date (YYYY-MM-DD string or Date). Defaults to today.
risk_zones	Add background shading for typical ACWR risk zones? Default ‘TRUE‘.
exposure_df	Optional. A pre-calculated data frame from ‘calculate_exposure‘. If provided, ‘stoken‘ and other calculation parameters are ignored. Must contain ‘date‘, ‘atl‘, ‘ctl‘ (and ‘acwr‘ if ‘risk_zones = TRUE‘).

Details

Plots ATL vs CTL, optionally showing risk zones based on ACWR. Uses pre-calculated data or calls ‘calculate_exposure‘.

Visualizes training state by plotting ATL vs CTL (related to PMC charts). Points are colored by date, latest point is highlighted (red triangle). Optional risk zones (based on ACWR thresholds ~0.8, 1.3, 1.5) can be shaded. If ‘exposure_df‘ is not provided, it calls ‘calculate_exposure‘ first.

Value

A ggplot object showing ATL vs CTL.

Examples

```
# Example using simulated data
data(Athlytics_sample_data)
# Ensure exposure_df is named and other necessary parameters like activity_type are provided
p <- plot_exposure(exposure_df = athlytics_sample_exposure, activity_type = "Run")
print(p)

## Not run:
# Example using real data (requires authentication)
# stoken <- rStrava::strava_oauth("YOUR_APP_NAME",
#                                 "YOUR_APP_CLIENT_ID",
#                                 "YOUR_APP_SECRET",
#                                 cache = TRUE)

# Plot Exposure trend for Runs (last 6 months)
# plot_exposure(stoken = stoken, # Replace stoken with a valid token object
#               activity_type = "Run",
#            end_date = Sys.Date(), # For internal calculate_exposure: fetches prior data.
#               # Note: start_date applies to the internal calculate_exposure call.
#               user_ftp = 280) # Example, if load_metric = "tss"

# Plot Exposure trend for Rides
# plot_exposure(stoken = stoken, # Replace stoken with a valid token object
#               activity_type = "Ride",
#               user_ftp = 280) # Example, provide if load_metric = "tss"

# Plot Exposure trend for multiple Run types (risk_zones = FALSE for this example)
# plot_exposure(stoken = stoken, # Replace stoken with a valid token object
#               activity_type = c("Run", "VirtualRun"),
#               risk_zones = FALSE,
#               user_ftp = 280) # Example, provide if load_metric = "tss"

## End(Not run)
```

`plot_pbs` *Plot Personal Best (PB) Trends*

Description

Visualizes the trend of personal best times for specific running distances.

Usage

```
plot_pbs(
  stoken,
  activity_type = "Run",
  distance_meters,
  max_activities = 500,
  date_range = NULL,
```

```
  add_trend_line = TRUE,
  pbs_df = NULL
)
```

Arguments

stoken	A valid Strava token from ‘rStrava::strava_oauth()‘. Required unless ‘pbs_df‘ is provided.
activity_type	Type(s) of activities to search (e.g., "Run"). Default "Run".
distance_meters	Numeric vector of distances (meters) to plot PBs for (e.g., ‘c(1000, 5000)‘). Relies on Strava’s ‘best_efforts‘ data.
max_activities	Max number of recent activities to check. Default 500. Reduce for speed.
date_range	Optional. Filter activities by date ‘c("YYYY-MM-DD", "YYYY-MM-DD")‘.
add_trend_line	Logical. Whether to add a trend line to the plot. Default TRUE.
pbs_df	Optional. A pre-calculated data frame from ‘calculate_pbs‘. If provided, ‘stoken‘ and other calculation parameters are ignored.

Details

Plots the trend of best efforts for specified distances, highlighting new PBs. Uses pre-calculated data or calls ‘calculate_pbs‘.

Visualizes data from ‘calculate_pbs‘. Points show best efforts; solid points mark new PBs. Y-axis is MM:SS. If ‘pbs_df‘ is not provided, calls ‘calculate_pbs‘ first (can be slow).

Value

A ggplot object showing PB trends, faceted by distance if multiple are plotted.

Examples

```
# Example using simulated data
data(Athlytics_sample_data)
# athlytics_sample_pbs should contain the PBs to be plotted
if (!is.null(athlytics_sample_pbs) && nrow(athlytics_sample_pbs) > 0) {
  sample_pbs_for_plot <- athlytics_sample_pbs

  # Ensure the date column is named 'activity_date' and is of Date type for plot_pbs
 if ("date" %in% names(sample_pbs_for_plot) && !"activity_date" %in% names(sample_pbs_for_plot)) {
   names(sample_pbs_for_plot)[names(sample_pbs_for_plot) == "date"] <- "activity_date"
 }
 if ("activity_date" %in% names(sample_pbs_for_plot)) {
   sample_pbs_for_plot$activity_date <- as.Date(sample_pbs_for_plot$activity_date)
 } else {
   message("Relevant date column not found in sample PBs for example.")
 }

 # plot_pbs requires distance_meters. Extract from sample data.
 req_dist_meters <- NULL
```

```
  if ("distance" %in% names(sample_pbs_for_plot)) {
    req_dist_meters <- unique(sample_pbs_for_plot$distance)
  } else if ("distance_target_m" %in% names(sample_pbs_for_plot)) {
    req_dist_meters <- unique(sample_pbs_for_plot$distance_target_m)
  }

  can_plot <- "activity_date" %in% names(sample_pbs_for_plot) &&
              !is.null(req_dist_meters) && length(req_dist_meters) > 0

  if (can_plot) {
    p <- plot_pbs(pbs_df = sample_pbs_for_plot, activity_type = "Run",
                  distance_meters = req_dist_meters)
    print(p)
  } else {
    message("Sample PBs data lacks required date or distance info for example.")
  }
} else {
  message("athlytics_sample_pbs is empty or not found, skipping example plot.")
}

## Not run:
# Example using real data (requires authentication)
# Users should first authenticate and obtain a stoken, e.g.:
# To authenticate (replace with your details):
# stoken <- rStrava::strava_oauth(app_name = "YOUR_APP",
#                                 client_id = "YOUR_ID",
#                                 client_secret = "YOUR_SECRET",
#                                 cache = TRUE)

# Plot PBS trend for Runs (last 6 months)
# Note: plot_pbs requires distance_meters.
# This example assumes you want to see all available from calculate_pbs.
# For a specific plot, ensure calculate_pbs was run for those distances
# or specify them here.
# pb_data_run <- calculate_pbs(stoken = stoken, activity_type = "Run",
#                              distance_meters = c(1000,5000,10000),
#                              date_range = c(format(Sys.Date() - months(6)),
#                                             format(Sys.Date())))
# if(nrow(pb_data_run) > 0) {
#   plot_pbs(pbs_df = pb_data_run, distance_meters = c(1000,5000,10000))
# }

# Plot PBS trend for Rides (if applicable, though PBs are mainly for Runs)
# Ensure distance_meters are relevant for Ride PBs if your calculate_pbs handles them.
# pb_data_ride <- calculate_pbs(stoken = stoken, activity_type = "Ride",
#                               distance_meters = c(10000, 20000))
# if(nrow(pb_data_ride) > 0) {
#    plot_pbs(pbs_df = pb_data_ride, distance_meters = c(10000, 20000))
# }

# Plot PBS trend for multiple Run types (no trend line)
# Ensure distance_meters are specified
# pb_data_multi <- calculate_pbs(stoken = stoken,
```

```
#                               activity_type = c("Run", "VirtualRun"),
#                               distance_meters = c(1000,5000))
# if(nrow(pb_data_multi) > 0) {
#   plot_pbs(pbs_df = pb_data_multi, distance_meters = c(1000,5000),
#            add_trend_line = FALSE)
# }

## End(Not run)
```

Index