Package: shorts (via r-universe)

October 27, 2024

Type Package

Title Short Sprints

Version 3.3.0

Description Create short sprint acceleration-velocity (AVP) and force-velocity (FVP) profiles and predict kinematic and kinetic variables using the timing-gate split times, laser or radar gun data, tether devices data, as well as the data provided by the GPS and LPS monitoring systems. The modeling method utilized in this package is based on the works of Furusawa K, Hill AV, Parkinson JL (1927) <doi:10.1098/rspb.1927.0035>, Greene PR. (1986) <doi:10.1016/0025-5564(86)90063-5>, Chelly SM, Denis C. (2001) <doi:10.1097/00005768-200102000-00024>, Clark KP, Rieger RH, Bruno RF, Stearne DJ. (2017) <doi:10.1519/JSC.0000000000002081>, Samozino P. (2018) <doi:10.1007/978-3-319-05633-3_11>, Samozino P. and Peyrot N., et al (2022) <doi:10.1111/sms.14097>, Clavel, P., et al (2023) <doi:10.1016/j.jbiomech.2023.111602>, Jovanovic M. (2023) <doi:10.1080/10255842.2023.2170713>, Jovanovic M., et al (2024) <doi:10.3390/s24092894>, and Jovanovic M., et al (2024) <doi:10.3390/s24196192>.

URL `https://mladenjovanovic.github.io/shorts/`

BugReports `https://github.com/mladenjovanovic/shorts/issues`

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Depends R (>= 2.10)

Imports stats, LambertW, tidyr, ggplot2, minpack.lm, purrr

Config/testthat/edition 3

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

RemoteUrl https://github.com/mladenjovanovic/shorts

RemoteRef HEAD

RemoteSha 5f8ce5554b5aca0e857199e0eb50bf66d7089130

Contents

`coef.shorts_model` *S3 method for extracting model parameters from* `shorts_model` *object*

Description

S3 method for extracting model parameters from `shorts_model` object

Usage

```
## S3 method for class 'shorts_model'
coef(object, ...)
```

Arguments

`object`	`shorts_model` object
`...`	Extra arguments. Not used

Examples

```
split_distances <- c(10, 20, 30, 40, 50)
split_times <- create_timing_gates_splits(
  gates = split_distances,
  MSS = 10,
  MAC = 9,
  FD = 0.25,
  TC = 0
)

# Simple model
simple_model <- model_timing_gates(split_distances, split_times)
coef(simple_model)
```

---

confint.shorts_model — *S3 method for providing confidence intervals for the* `shorts_model`

Description

S3 method for providing confidence intervals for the `shorts_model`

Usage

```
## S3 method for class 'shorts_model'
confint(object, ...)
```

Arguments

`object`	`shorts_model` object
`...`	Forwarded to generic `confint()` function

Examples

```
## Not run:
split_distances <- c(10, 20, 30, 40, 50)
split_times <- create_timing_gates_splits(
  gates = split_distances,
  MSS = 10,
  MAC = 9,
  FD = 0,
  TC = 0,
  noise = 0.01
)

# Simple model
simple_model <- model_timing_gates(split_distances, split_times)
confint(simple_model)

## End(Not run)
```

convert_FVP *Convert Force-Velocity profile back to Acceleration-Velocity profile*

Description

This function converts back the Force-Velocity profile (FVP; F0 and V0 parameters) to Acceleration-Velocity profile (AVP; MSS and MAC parameters)

Usage

```
convert_FVP(
  F0,
  V0,
  bodymass = 75,
  inertia = 0,
  resistance = 0,
  wind_velocity = 0,
  ...
)
```

Arguments

`F0, V0`	Numeric vectors. FV profile parameters
`bodymass`	Body mass in kg. Used to calculate relative power and forwarded to `get_air_resistance`
`inertia`	External inertia in kg (for example a weight vest, or a sled). Not included in the air resistance calculation
`resistance`	External horizontal resistance in Newtons (for example tether device or a sled friction resistance)
`wind_velocity`	In meters per second (m/s). Use negative number as head wind, and positive number as back wind. Default is 0m/s (no wind)
`...`	Forwarded to `predict_power_at_distance`

Value

A list with calculated `MSS` and `MAC` parameters

Examples

```
FVP <- create_FVP(7, 8.3, inertia = 10, resistance = 50)

convert_FVP(FVP$F0, FVP$V0, inertia = 10, resistance = 50)
```

`create_FVP` *Create Force-Velocity Profile*

Description

Creates Force-Velocity Profile (FVP) modified using ideas by Pierre Samozino and JB-Morin, et al. (2016) and Pierre Samozino and Nicolas Peyror, et al (2021).

Usage

```
create_FVP(
  MSS,
  MAC,
  bodymass = 75,
  inertia = 0,
  resistance = 0,
  wind_velocity = 0,
  ...
)
```

Arguments

`MSS, MAC`	Numeric vectors. Model parameters
`bodymass`	Body mass in kg. Used to calculate relative power and forwarded to `get_air_resistance`
`inertia`	External inertia in kg (for example a weight vest, or a sled). Not included in the air resistance calculation
`resistance`	External horizontal resistance in Newtons (for example tether device or a sled friction resistance)
`wind_velocity`	In meters per second (m/s). Use negative number as head wind, and positive number as back wind. Default is 0m/s (no wind)
`...`	Forwarded to `predict_power_at_distance`

Value

List containing the following elements:

bodymass Returned `bodymass` used in FV profiling

F0 Horizontal force when velocity=0

F0_rel `F0` divided by `bodymass`

V0 Velocity when horizontal force=0

Pmax Maximal horizontal power

Pmax_rel `Pmax` divided by `bodymass`

FV_slope Slope of the FV profile. See References for more info

References

Samozino P, Rabita G, Dorel S, Slawinski J, Peyrot N, Saez de Villarreal E, Morin J-B. 2016. A simple method for measuring power, force, velocity properties, and mechanical effectiveness in sprint running: Simple method to compute sprint mechanics. Scandinavian Journal of Medicine & Science in Sports 26:648–658. DOI: 10.1111/sms.12490.

Samozino P, Peyrot N, Edouard P, Nagahara R, Jimenez-Reyes P, Vanwanseele B, Morin J. 2022. Optimal mechanical force-velocity profile for sprint acceleration performance. Scandinavian Journal of Medicine & Science in Sports 32:559–575. DOI: 10.1111/sms.14097.

Examples

```
data("jb_morin")

m1 <- model_radar_gun(time = jb_morin$time, velocity = jb_morin$velocity)

fv_profile <- create_FVP(
  MSS = m1$parameters$MSS,
  MAC = m1$parameters$MAC,
  bodyheight = 1.72,
  bodymass = 120,
)

fv_profile
```

`create_sprint_trace` *Create Sprint Trace*

Description

This function creates sprint trace either using `time` or `distance` vectors

Usage

```
create_sprint_trace(
  MSS,
  MAC,
  time = NULL,
  distance = NULL,
  TC = 0,
  DC = 0,
  FD = 0,
  remove_leading = FALSE
)
```

Arguments

`MSS, MAC`	Numeric vector. Model parameters
`time`	Numeric vector.
`distance`	Numeric vector.
`TC`	Numeric vector. Time-shift added to sprint times. Default is 0
`DC`	Numeric vector. Distance-shift added to sprint distance. Default is 0
`FD`	Numeric vector. Flying start distance. Default is 0
`remove_leading`	Should trace leading to sprint be removed? Default is `FALSE`

Value

Data-frame with following 6 columns

time Measurement-scale time vector in seconds. Equal to parameter `time`

distance Measurement-scale distance vector in meters. Equal to parameter `distance`

velocity Velocity vector in m/s

acceleration Acceleration vector in m/s/s

sprint_time Sprint scale time vector in seconds. Sprint always start at t=0s

sprint_distance Sprint scale distance vector in meters. Sprint always start at d=0m

See Also

`create_timing_gates_splits`

Examples

```
df <- create_sprint_trace(8, 7, time = seq(0, 6, by = 0.01))
df <- create_sprint_trace(8, 7, distance = seq(0, 40, by = 1))
```

`create_timing_gates_splits` *Create Timing Gates Splits*

Description

This function is used to generate timing gates splits with predetermined parameters

Usage

```
create_timing_gates_splits(
  MSS,
  MAC,
  gates = c(5, 10, 20, 30, 40),
  FD = 0,
  TC = 0,
  noise = 0
)
```

Arguments

MSS, MAC	Numeric vectors. Model parameters
gates	Numeric vectors. Distances of the timing gates
FD	Numeric vector. Flying start distance. Default is 0
TC	Numeric vector. Time-correction added to split times (e.g., reaction time). Default is 0
noise	Numeric vector. SD of Gaussian noise added to the split times. Default is 0

See Also

create_sprint_trace

Examples

```
create_timing_gates_splits(
  gates = c(10, 20, 30, 40, 50),
  MSS = 10,
  MAC = 9,
  FD = 0.5,
  TC = 0
)
```

---

dynaspeed *DynaSpeed Single Sprint Data*

Description

DynaSpeed(TM) data collected for a single athlete (female, 177cm, 64kg) and a single sprint over 40m. Sampling frequency is 1,000Hz. Additional time and distance shift is added to the dataset to provide a sandbox for potential issues during the analysis

Usage

```
data(dynaspeed)
```

Format

Data frame with 4 variables and 7,251 observations:

time time in seconds

distance Distance in meters

velocity Smoothed velocity in meters per second

raw_velocity Velocity in meters per second

Author(s)

Håkan Andersson
The High-Performance Center
Växjö, Sweden
<hakan.andersson@hpcsweden.com>

`find_functions` *Find functions*

Description

Family of functions that serve a purpose of finding maximal value and critical distances and times at which power, acceleration or velocity drops below certain threshold.

`find_peak_power_distance` finds peak power and `distance` at which peak power occurs

`find_peak_power_time` finds peak power and `time` at which peak power occurs

`find_velocity_critical_distance` finds critical distance at which `percent` of `MSS` is achieved

`find_velocity_critical_time` finds critical time at which `percent` of `MSS` is achieved

`find_acceleration_critical_distance` finds critical distance at which `percent` of `MAC` is reached

`find_acceleration_critical_time` finds critical time at which `percent` of `MAC` is reached

`find_power_critical_distance` finds critical distances at which peak power over `percent` is achieved

`find_power_critical_time` finds critical times at which peak power over `percent` is achieved

Usage

```
find_peak_power_distance(MSS, MAC, inertia = 0, resistance = 0, ...)

find_peak_power_time(MSS, MAC, inertia = 0, resistance = 0, ...)

find_velocity_critical_distance(MSS, MAC, percent = 0.9)

find_velocity_critical_time(MSS, MAC, percent = 0.9)

find_acceleration_critical_distance(MSS, MAC, percent = 0.9)
```

```
find_acceleration_critical_time(MSS, MAC, percent = 0.9)

find_power_critical_distance(
  MSS,
  MAC,
  inertia = 0,
  resistance = 0,
  percent = 0.9,
  ...
)

find_power_critical_time(
  MSS,
  MAC,
  inertia = 0,
  resistance = 0,
  percent = 0.9,
  ...
)
```

Arguments

MSS, MAC	Numeric vectors. Model parameters
inertia	External inertia in kg (for example a weight vest, or a sled). Not included in the air resistance calculation
resistance	External horizontal resistance in Newtons (for example tether device or a sled friction resistance)
...	Arguments passed on to get_air_resistance velocity Instantaneous running velocity in meters per second (m/s) bodymass In kilograms (kg). Default is 75kg bodyheight In meters (m). Default is 1.75m barometric_pressure In Torrs. Default is 760Torrs air_temperature In Celzius (C). Default is 25C wind_velocity In meters per second (m/s). Use negative number as head wind, and positive number as back wind. Default is 0m/s (no wind)
percent	Numeric vector. Used to calculate critical distance. Default is 0.9

Value

find_peak_power_distance returns list with two elements: peak_power and distance at which peak power occurs

find_peak_power_time returns list with two elements: peak_power and time at which peak power occurs

References

Haugen TA, Tønnessen E, Seiler SK. 2012. The Difference Is in the Start: Impact of Timing and Start Procedure on Sprint Running Performance: Journal of Strength and Conditioning Research 26:473–479. DOI: 10.1519/JSC.0b013e318226030b.

Samozino P. 2018. A Simple Method for Measuring Force, Velocity and Power Capabilities and Mechanical Effectiveness During Sprint Running. In: Morin J-B, Samozino P eds. Biomechanics of Training and Testing. Cham: Springer International Publishing, 237–267. DOI: 10.1007/978-3-319-05633-3_11.

Examples

```
dist <- seq(0, 40, length.out = 1000)

velocity <- predict_velocity_at_distance(
  distance = dist,
  MSS = 10,
  MAC = 9
)

acceleration <- predict_acceleration_at_distance(
  distance = dist,
  MSS = 10,
  MAC = 9
)

# Use ... to forward parameters to the shorts::get_air_resistance
pwr <- predict_power_at_distance(
  distance = dist,
  MSS = 10,
  MAC = 9
  # bodyweight = 100,
  # bodyheight = 1.9,
  # barometric_pressure = 760,
  # air_temperature = 25,
  # wind_velocity = 0
)

# Find critical distance when 90% of MSS is reached
plot(x = dist, y = velocity, type = "l")
abline(h = 10 * 0.9, col = "gray")
abline(v = find_velocity_critical_distance(MSS = 10, MAC = 9), col = "red")

# Find critical distance when 20% of MAC is reached
plot(x = dist, y = acceleration, type = "l")
abline(h = 9 * 0.2, col = "gray")
abline(v = find_acceleration_critical_distance(MSS = 10, MAC = 9, percent = 0.2), col = "red")

# Find peak power and location of peak power
plot(x = dist, y = pwr, type = "l")

peak_pwr <- find_peak_power_distance(
```

```
  MSS = 10,
  MAC = 9
  # Use ... to forward parameters to the shorts::get_air_resistance
)
abline(h = peak_pwr$peak_power, col = "gray")
abline(v = peak_pwr$distance, col = "red")

# Find distance in which relative power stays over 75% of PMAX'
plot(x = dist, y = pwr, type = "l")
abline(h = peak_pwr$peak_power * 0.75, col = "gray")
pwr_zone <- find_power_critical_distance(MSS = 10, MAC = 9, percent = 0.75)
abline(v = pwr_zone$lower, col = "blue")
abline(v = pwr_zone$upper, col = "blue")
```

`find_optimal_distance` *Function that finds the distance at which the sprint, probe, or FV profile is optimal (i.e., equal to 100 perc)*

Description

Function that finds the distance at which the sprint, probe, or FV profile is optimal (i.e., equal to 100 perc)

Usage

```
find_optimal_distance(..., optimal_func = optimal_FV, min = 1, max = 60)
```

Arguments

`...`	Forwarded to selected `optimal_func`
`optimal_func`	Selected profile optimization function. Default is `optimal_FV`
`min, max`	Distance over which to find optimal profile distance

Value

Distance

Examples

```
MSS <- 10
MAC <- 8
bodymass <- 75

fv <- create_FVP(MSS, MAC, bodymass)

find_optimal_distance(
  F0 = fv$F0,
  V0 = fv$V0,
```

```
  bodymass = fv$bodymass,
  optimal_func = optimal_FV,
  method = "max"
)

find_optimal_distance(
  MSS = MSS,
  MAC = MAC,
  optimal_func = optimal_MSS_MAC
)

find_optimal_distance(
  MSS = MSS,
  MAC = MAC,
  optimal_func = probe_MSS_MAC
)
```

fitted.shorts_model — *S3 method for returning predictions of* shorts_model

Description

S3 method for returning predictions of shorts_model

Usage

```
## S3 method for class 'shorts_model'
fitted(object, ...)
```

Arguments

object	shorts_model object
...	Extra arguments. Not used

Examples

```
split_distances <- c(10, 20, 30, 40, 50)
split_times <- create_timing_gates_splits(
  gates = split_distances,
  MSS = 10,
  MAC = 9,
  FD = 0.25,
  TC = 0
)

# Simple model
simple_model <- model_timing_gates(split_distances, split_times)
fitted(simple_model)
```

format_splits *Format Split Data*

Description

Function formats split data and calculates split distances, split times and average split velocity

Usage

```
format_splits(distance, time)
```

Arguments

`distance`	Numeric vector
`time`	Numeric vector

Value

Data frame with the following columns:

split Split number

split_distance_start Distance at which split starts

split_distance_stop Distance at which split ends

split_distance Split distance

split_time_start Time at which distance starts

split_time_stop Time at which distance ends

split_time Split time

split_mean_velocity Mean velocity over split distance

split_mean_acceleration Mean acceleration over split distance

Examples

```
data("split_times")

john_data <- split_times[split_times$athlete == "John", ]

format_splits(john_data$distance, john_data$time)
```

`get_air_resistance` *Get Air Resistance*

Description

`get_air_resistance` estimates air resistance in Newtons

Usage

```
get_air_resistance(
  velocity,
  bodymass = 75,
  bodyheight = 1.75,
  barometric_pressure = 760,
  air_temperature = 25,
  wind_velocity = 0
)
```

Arguments

`velocity`	Instantaneous running velocity in meters per second (m/s)
`bodymass`	In kilograms (kg). Default is 75kg
`bodyheight`	In meters (m). Default is 1.75m
`barometric_pressure`	In Torrs. Default is 760Torrs
`air_temperature`	In Celzius (C). Default is 25C
`wind_velocity`	In meters per second (m/s). Use negative number as head wind, and positive number as back wind. Default is 0m/s (no wind)

Value

Air resistance in Newtons (N)

References

Arsac LM, Locatelli E. 2002. Modeling the energetics of 100-m running by using speed curves of world champions. Journal of Applied Physiology 92:1781–1788. DOI: 10.1152/japplphysiol.00754.2001.

Samozino P, Rabita G, Dorel S, Slawinski J, Peyrot N, Saez de Villarreal E, Morin J-B. 2016. A simple method for measuring power, force, velocity properties, and mechanical effectiveness in sprint running: Simple method to compute sprint mechanics. Scandinavian Journal of Medicine & Science in Sports 26:648–658. DOI: 10.1111/sms.12490.

van Ingen Schenau GJ, Jacobs R, de Koning JJ. 1991. Can cycle power predict sprint running performance? European Journal of Applied Physiology and Occupational Physiology 63:255–260. DOI: 10.1007/BF00233857.

Examples

```
get_air_resistance(
  velocity = 5,
  bodymass = 80,
  bodyheight = 1.90,
  barometric_pressure = 760,
  air_temperature = 16,
  wind_velocity = -0.5
)
```

---

jb_morin — *JB Morin Sample Dataset*

Description

Sample radar gun data provided by Jean-Benoît Morin on his website. See https://jbmorin.net/2017/12/13/a-spreadsheet-for-sprint-acceleration-force-velocity-power-profiling/ for more details.

Usage

```
data(jb_morin)
```

Format

Data frame with 2 variables and 232 observations:

time Time in seconds

velocity Velocity in m/s

Details

This dataset represents a sample data provided by Jean-Benoît Morin on a single individual running approximately 35m from a stand still position that is measured with the radar gun. Individual's body mass is 75kg, height is 1.72m. Conditions of the run are the following: air temperature 25C, barometric pressure 760mmHg, wind velocity 0m/s.

The purpose of including this dataset in the package is to check the agreement of the model estimates with Jean-Benoît Morin Microsoft Excel spreadsheet.

Author(s)

Jean-Benoît Morin
Inter-university Laboratory of Human Movement Biology
Saint-Étienne, France https://jbmorin.net/

References

Morin JB. 2017.A spreadsheet for Sprint acceleration Force-Velocity-Power profiling. Available at https://jbmorin.net/2017/12/13/a-spreadsheet-for-sprint-acceleration-force-velocity-power-profiling/ (accessed October 27, 2020).

`laser_gun_data` *Laser Gun Data*

Description

Performance of 35m sprint by a youth basketball player done using standing start. Sample was collected by laser gun (CMP3 Distance Sensor, Noptel Oy, Oulu, Finland) and was sampled at a rate of 2.56 KHz. A polynomial function modeling the relationship between distance and time was employed and subsequently resampled at a frequency of 1,000 Hz using Musclelab™ v10.232.107.5298, a software developed by Ergotest Technology AS located in Langesund, Norway. Data was further modified by calculating raw acceleration using dv/dt (using smoothed velocity provided by the system), and then smoothed out using 4th-order Butterworth filter with a cutoff frequency of 1 Hz.

Usage

```
data(laser_gun_data)
```

Format

Data frame with 6 variables and 4805 observations:

time Time vector in seconds

distance Distance vector in meters

velocity Smoothed velocity vector in m/s; this represent step-averaged velocity

raw_velocity Raw velocity vector in m/s

raw_acceleration Raw acceleration vector in m/s/s; calculated using difference in smoothed velocity divided by time difference (i.e., dv/dt method of derivation)

butter_acceleration Smoothed acceleration vector in m/s/s; smoothed out using 4th-order Butterworth filter with a cutoff frequency of 1 Hz

LPS_session *LPS Basketball Session Dataset*

Description

LPS Basketball Session Dataset

Usage

```
data(LPS_session)
```

Format

Data frame with 5 variables and 91,099 observations:

time Time in seconds from the start of the session

x x-coordinate in meters provided by the LPS

y y-coordinate in meters provided by the LPS

velocity Velocity provided by LPS in m/s

acceleration Acceleration provided by LPS in m/s

Details

This dataset represents a sample data provided by Local Positioning System (LPS) on a single individual performing a single basketball practice session (aprox. 90min). Sampling frequency is 20Hz.

model_functions *Model functions*

Description

Family of functions that serve a purpose of estimating short sprint parameters

`model_in_situ` estimates short sprint parameters using velocity-acceleration trace, provided by the monitoring systems like GPS or LPS. See references for the information

`model_radar_gun` estimates short sprint parameters using time-velocity trace, with additional parameter `TC` serving as intercept

`model_laser_gun` alias for `model_radar_gun`

`model_tether` estimates short sprint parameters using distance-velocity trace (e.g., tether devices).

`model_tether_DC` estimates short sprint parameters using distance-velocity trace (e.g., tether devices) with additional distance correction `DC` parameter

`model_time_distance` estimates short sprint parameters using time distance trace model_time_distance_FD estimates short sprint parameters using time-distance trace with additional estimated flying distance correction parameter FD

model_time_distance_FD_fixed estimates short sprint parameters using time-distance trace with additional flying distance correction parameter FD which is fixed by the user

model_time_distance estimates short sprint parameters using time distance trace with additional time correction parameter TC

model_time_distance estimates short sprint parameters using time distance trace with additional distance correction parameter DC

model_time_distance estimates short sprint parameters using time distance trace with additional time correction TC and distance correction TC parameters

model_timing_gates estimates short sprint parameters using distance-time trace (e.g., timing gates/photo cells)

model_timing_gates_TC estimates short sprint parameters using distance-time trace (e.g., timing gates/photo cells), with additional time correction parameter TC

model_timing_gates_FD estimates short sprint parameters using distance-time trace (e.g., timing gates/photo cells), with additional estimated flying distance correction parameter FD

model_timing_gates_FD_fixed estimates short sprint parameters using distance-time trace (e.g., timing gates/photo cells), with additional flying distance correction parameter FD which is fixed by the user

model_timing_gates_DC estimates short sprint parameters using distance-time trace (e.g., timing gates/photo cells), with additional distance correction parameter DC

model_timing_gates_TC_DC estimates short sprint parameters using distance-time trace (e.g., timing gates/photo cells), with additional time correction TC and distance correction DC parameters

Usage

```
model_in_situ(
  velocity,
  acceleration,
  weights = 1,
  acceleration_threshold = 0,
  velocity_threshold = NULL,
  velocity_step = 0.2,
  n_observations = 2,
  filter_func = function(x) {
     x
 },
  CV = NULL,
  na.rm = FALSE,
  ...
)

model_radar_gun(
  time,
  velocity,
  weights = 1,
```

```
  CV = NULL,
  use_observed_MSS = FALSE,
  na.rm = FALSE,
  ...
)

model_laser_gun(
  time,
  velocity,
  weights = 1,
  CV = NULL,
  use_observed_MSS = FALSE,
  na.rm = FALSE,
  ...
)

model_tether(
  distance,
  velocity,
  weights = 1,
  CV = NULL,
  use_observed_MSS = FALSE,
  na.rm = FALSE,
  ...
)

model_tether_DC(
  distance,
  velocity,
  weights = 1,
  CV = NULL,
  use_observed_MSS = FALSE,
  na.rm = FALSE,
  ...
)

model_time_distance(time, distance, weights = 1, CV = NULL, na.rm = FALSE, ...)

model_time_distance_FD(
  time,
  distance,
  weights = 1,
  CV = NULL,
  na.rm = FALSE,
  ...
)

model_time_distance_FD_fixed(
```

```
  time,
  distance,
  weights = 1,
  FD = 0,
  CV = NULL,
  na.rm = FALSE,
  ...
)

model_time_distance_TC(
  time,
  distance,
  weights = 1,
  CV = NULL,
  na.rm = FALSE,
  ...
)

model_time_distance_DC(
  time,
  distance,
  weights = 1,
  CV = NULL,
  na.rm = FALSE,
  ...
)

model_time_distance_TC_DC(
  time,
  distance,
  weights = 1,
  CV = NULL,
  na.rm = FALSE,
  ...
)

model_timing_gates(distance, time, weights = 1, CV = NULL, na.rm = FALSE, ...)

model_timing_gates_TC(
  distance,
  time,
  weights = 1,
  CV = NULL,
  na.rm = FALSE,
  ...
)

model_timing_gates_FD(
```

```
  distance,
  time,
  weights = 1,
  CV = NULL,
  na.rm = FALSE,
  ...
)

model_timing_gates_FD_fixed(
  distance,
  time,
  weights = 1,
  FD = 0,
  CV = NULL,
  na.rm = FALSE,
  ...
)

model_timing_gates_DC(
  distance,
  time,
  weights = 1,
  CV = NULL,
  na.rm = FALSE,
  ...
)

model_timing_gates_TC_DC(
  distance,
  time,
  weights = 1,
  CV = NULL,
  na.rm = FALSE,
  ...
)
```

Arguments

`weights`	Numeric vector. Default is 1
`acceleration_threshold`	Acceleration cutoff. Default is 0
`velocity_threshold`	Velocity cutoff. If `NULL` (default), velocity of the observation with the fastest acceleration is taken as the cutoff value (after applying the `filter_func`)
`velocity_step`	Velocity increment size for finding max acceleration. Default is 0.2 m/s
`n_observations`	Number of top acceleration observations to keep in velocity bracket. Default is 2

`filter_func` Function to filter outliers within each velocity bracket. This is used to remove noise in the data. Default is `function(x){x}` which doesn't involve any filtering.

`CV` Should cross-validation be used to estimate model fit? Default is `NULL`. Otherwise use integer indicating number of folds

`na.rm` Logical. Default is FALSE

`...` Forwarded to `nlsLM` function

`time, velocity, distance, acceleration`
Numeric vector

`use_observed_MSS`
Should observed peak `velocity` be used as `MSS` parameter? Default is `FALSE`

`FD` Flying distance parameter. Default is 0

Value

List object with the following elements:

data Data frame used to estimate the sprint parameters

model_info Extra information regarding model used

model Model returned by the `nlsLM` function

parameters List with the following estimated parameters: `MSS`, `MAC`, `TAU`, and `PMAX`

correction List with additional model correcitons

predictions Data frame with `.predictor`, `.observed`, `.predicted`, and `.residual` columns

model_fit List with multiple model fit estimators

CV If cross-validation is performed, this will included the data as above, but for each fold

References

Samozino P. 2018. A Simple Method for Measuring Force, Velocity and Power Capabilities and Mechanical Effectiveness During Sprint Running. In: Morin J-B, Samozino P eds. Biomechanics of Training and Testing. Cham: Springer International Publishing, 237–267. DOI: 10.1007/978-3-319-05633-3_11.

Clavel, P., Leduc, C., Morin, J.-B., Buchheit, M., & Lacome, M. (2023). Reliability of individual acceleration-speed profile in-situ in elite youth soccer players. Journal of Biomechanics, 153, 111602. https://doi.org/10.1016/j.jbiomech.2023.111602

Morin, J.-B. (2021). The "in-situ" acceleration-speed profile for team sports: testing players without testing them. JB Morin, PhD – Sport Science website. Accessed 31. Dec. 2023. https://jbmorin.net/2021/07/29/the-in-situ-sprint-profile-for-team-sports-testing-players-without-testing-them/

Examples

```
# Model In-Situ (Embedded profiling)
data("LPS_session")
m1 <- model_in_situ(
  velocity = LPS_session$velocity,
  acceleration = LPS_session$acceleration,
```

```
  # Use specific cutoff value
  velocity_threshold = 4)
m1
plot(m1)


# Model Radar Gun (includes Time Correction)
df <- create_sprint_trace(MSS = 8, MAC = 6, time = seq(0, 6, 0.1))

# Add some noise
df$velocity <- df$velocity + rnorm(n = nrow(df), 0, 10^-2)

m1 <- model_radar_gun(time = df$time, velocity = df$velocity)
m1
plot(m1)


# Model Laser Gun (includes Time Correction)
df <- create_sprint_trace(MSS = 8, MAC = 6, time = seq(0, 6, 0.1))

# Add some noise
df$velocity <- df$velocity + rnorm(n = nrow(df), 0, 10^-2)

m1 <- model_laser_gun(time = df$time, velocity = df$velocity)
m1
plot(m1)


# Model Tether
df <- create_sprint_trace(MSS = 8, MAC = 6, time = seq(0, 6, 0.5))
m1 <- model_tether(distance = df$distance, velocity = df$velocity)
m1
plot(m1)


# Model Tether with Distance Correction (DC)
df <- create_sprint_trace(MSS = 8, MAC = 6, time = seq(0.001, 6, 0.5), DC = 5)
m1 <- model_tether_DC(distance = df$distance, velocity = df$velocity)
m1
plot(m1)


# Model Time-Distance trace (simple, without corrections)
df <- create_sprint_trace(MSS = 8, MAC = 6, time = seq(0, 5, by = 0.5))
m1 <- model_time_distance(time = df$time, distance = df$distance)
m1
plot(m1)


# Model Time-Distance trace (with Flying Distance Correction)
df <- create_sprint_trace(MSS = 8, MAC = 6, time = seq(0, 5, by = 0.5), FD = 0.5)
m1 <- model_time_distance_FD(time = df$time, distance = df$distance)
m1
```

```
plot(m1)


# Model Time-Distance trace (with Flying Distance Correction fixed)
df <- create_sprint_trace(MSS = 8, MAC = 6, time = seq(0, 5, by = 0.5), FD = 0.5)
m1 <- model_time_distance_FD_fixed(time = df$time, distance = df$distance, FD = 0.5)
m1
plot(m1)


# Model Time-Distance trace (with Time Correction)
df <- create_sprint_trace(MSS = 8, MAC = 6, time = seq(0, 5, by = 0.5), TC = 1.5)
m1 <- model_time_distance_TC(time = df$time, distance = df$distance)
m1
plot(m1)


# Model Time-Distance trace (with Distance Correction)
df <- create_sprint_trace(MSS = 8, MAC = 6, time = seq(0, 5, by = 0.5), DC = -5)
m1 <- model_time_distance_DC(time = df$time, distance = df$distance)
m1
plot(m1)


# Model Time-Distance trace (with Time and Distance Corrections)
df <- create_sprint_trace(MSS = 8, MAC = 6, time = seq(0, 5, by = 0.5), TC = -1.3, DC = 5)
m1 <- model_time_distance_TC_DC(time = df$time, distance = df$distance)
m1
plot(m1)


# Model Timing Gates (simple, without corrections)
df <- create_sprint_trace(MSS = 8, MAC = 6, distance = c(5, 10, 20, 30, 40))
m1 <- model_timing_gates(distance = df$distance, time = df$time)
m1
plot(m1)


# Model Timing Gates (with Time Correction)
df <- create_sprint_trace(MSS = 8, MAC = 6, distance = c(5, 10, 20, 30, 40), TC = 0.2)
m1 <- model_timing_gates_TC(distance = df$distance, time = df$time)
m1
plot(m1)


# Model Timing Gates (with Flying Distance Correction)
df <- create_sprint_trace(MSS = 8, MAC = 6, distance = c(5, 10, 20, 30, 40), FD = 0.5)
m1 <- model_timing_gates_FD(distance = df$distance, time = df$time)
m1
plot(m1)


# Model Timing Gates (with Flying Distance Correction fixed)
```

```
df <- create_sprint_trace(MSS = 8, MAC = 6, distance = c(5, 10, 20, 30, 40), FD = 0.5)
m1 <- model_timing_gates_FD_fixed(distance = df$distance, time = df$time)
m1
plot(m1)


# Model Timing Gates (with Distance Correction)
df <- create_sprint_trace(MSS = 8, MAC = 6, distance = c(5, 10, 20, 30, 40), DC = 1.5)
m1 <- model_timing_gates_DC(distance = df$distance, time = df$time)
m1
plot(m1)


# Model Timing Gates (with Time and Distance Corrections)
df <- create_sprint_trace(MSS = 8, MAC = 6, distance = c(5, 10, 20, 30, 40), TC = 0.25, DC = 1.5)
m1 <- model_timing_gates_TC_DC(distance = df$distance, time = df$time)
m1
plot(m1)
```

---

optimal_functions *Optimal profile functions*

Description

Family of functions that serve a purpose of finding optimal sprint or force-velocity profile

`optimal_FV` finds "optimal" `F0` and `V0` where time at distance is minimized, while keeping the power the same

`optimal_MSS_MAC` finds "optimal" `MSS` and `MAS` where time at distance is minimized, while keeping the `Pmax` the same

Usage

```
optimal_FV(
  distance,
  F0,
  V0,
  bodymass = 75,
  inertia = 0,
  resistance = 0,
  method = "max",
  ...
)

optimal_MSS_MAC(distance, MSS, MAC)
```

Arguments

`distance`	Numeric vector
`F0, V0`	Numeric vectors. FV profile parameters
`bodymass`	Body mass in kg
`inertia`	External inertia in kg (for example a weight vest, or a sled). Not included in the air resistance calculation
`resistance`	External horizontal resistance in Newtons (for example tether device or a sled friction resistance)
`method`	Method to be utilized. Options are "peak" and "max" (default)
`...`	Arguments passed on to `get_air_resistance` `velocity` Instantaneous running velocity in meters per second (m/s) `bodyheight` In meters (m). Default is 1.75m `barometric_pressure` In Torrs. Default is 760Torrs `air_temperature` In Celzius (C). Default is 25C `wind_velocity` In meters per second (m/s). Use negative number as head wind, and positive number as back wind. Default is 0m/s (no wind)
`MSS, MAC`	Numeric vectors. Model parameters

Value

`optimal_FV` returns s data frame with the following columns

F0 Original F0

V0 Original F0

bodymass Bodymass

inertia Inertia

resistance Resistance

Pmax Maximal power estimated using F0 * V0 / 4

Pmax_rel Relative maximal power

slope FV profile slope

distance Distance

time Time to cover distance

Ppeak Peak power estimated quantitatively

Ppeak_rel Relative peak power

Ppeak_dist Distance at which peak power is manifested

Ppeak_time Time at which peak power is manifested

F0_optim Optimal F0

F0_coef Ratio between F0_optim an F0

V0_optim Optimal V0

V0_coef Ratio between V0_optim an V0

Pmax_optim Optimal maximal power estimated F0_optim * V0_optim / 4

Pmax_rel_optim Optimal relative maximal power

slope_optim Optimal FV profile slope

profile_imb Percent ratio between slope and optimal slope

time_optim Time to cover distance when profile is optimal

time_gain Difference in time to cover distance between time_optimal and time

Ppeak_optim Optimal peak power estimated quantitatively

Ppeak_rel_optim Optimal relative peak power

Ppeak_dist_optim Distance at which optimal peak power is manifested

Ppeak_time_optim Time at which optimal peak power is manifested

`optimal_MSS_MAC` returns a data frame with the following columns

MSS Original MSS

MAC Original MAC

Pmax_rel Relative maximal power estimated using MSS * MAC / 4

slope Sprint profile slope

distance Distance

time Time to cover distance

MSS_optim Optimal MSS

MSS_coef Ratio between MSS_optim an MSS

MAC_optim Optimal MAC

MAC_coef Ratio between MAC_optim an MAC

Pmax_rel_optim Optimal relative maximal power estimated using MSS_optim * MAC_optim / 4

slope_optim Optimal sprint profile slope

profile_imb Percent ratio between slope and optimal slope

time_optim Time to cover distance when profile is optimal

time_gain Difference in time to cover distance between time_optimal and time

References

Samozino P, Peyrot N, Edouard P, Nagahara R, Jimenez-Reyes P, Vanwanseele B, Morin J. 2022. Optimal mechanical force-velocity profile for sprint acceleration performance. Scandinavian Journal of Medicine & Science in Sports 32:559–575. DOI: 10.1111/sms.14097.

Examples

```
MSS <- 10
MAC <- 8
bodymass <- 75

fv <- create_FVP(MSS, MAC, bodymass)
```

```
dist <- seq(5, 40, by = 5)

opt_MSS_MAC_profile <- optimal_MSS_MAC(
  distance = dist,
  MSS,
  MAC
)[["profile_imb"]]

opt_FV_profile <- optimal_FV(
  distance = dist,
  fv$F0,
  fv$V0,
  fv$bodymass
)[["profile_imb"]]

opt_FV_profile_peak <- optimal_FV(
  distance = dist,
  fv$F0,
  fv$V0,
  fv$bodymass,
  method = "peak"
)[["profile_imb"]]

plot(x = dist, y = opt_MSS_MAC_profile, type = "l", ylab = "Profile imbalance")
lines(x = dist, y = opt_FV_profile, type = "l", col = "blue")
lines(x = dist, y = opt_FV_profile_peak, type = "l", col = "red")
abline(h = 100, col = "gray", lty = 2)
```

---

plot.shorts_model *S3 method for plotting* shorts_model *object*

Description

S3 method for plotting `shorts_model` object

Usage

```
## S3 method for class 'shorts_model'
plot(x, type = "model", ...)
```

Arguments

x	`shorts_model` object
type	Type of plot. Can be "model" (default), "kinematics-time", "kinematics-distance", or "residuals"
...	Not used

Value

`ggplot` object

Examples

```
# Simple model with radar gun data
instant_velocity <- data.frame(
  time = c(0, 1, 2, 3, 4, 5, 6),
  velocity = c(0.00, 4.99, 6.43, 6.84, 6.95, 6.99, 7.00)
)

radar_model <- with(
  instant_velocity,
  model_radar_gun(time, velocity)
)

plot(radar_model)
plot(radar_model, "kinematics-time")
plot(radar_model, "kinematics-distance")
plot(radar_model, "residuals")
```

---

predict.shorts_model *S3 method for making predictions using* shorts_model

Description

S3 method for making predictions using shorts_model

Usage

```
## S3 method for class 'shorts_model'
predict(object, ...)
```

Arguments

object	shorts_model object
...	Forwarded to generic predict() function

Examples

```
split_distances <- c(10, 20, 30, 40, 50)
split_times <- create_timing_gates_splits(
  gates = split_distances,
  MSS = 10,
  MAC = 9,
  FD = 0.25,
  TC = 0
)

# Simple model
simple_model <- model_timing_gates(split_distances, split_times)
predict(simple_model)
```

`predict_kinematics` *Kinematics prediction functions*

Description

Predicts kinematic from known MSS and MAC parameters

Usage

```
predict_velocity_at_time(time, MSS, MAC)

predict_distance_at_time(time, MSS, MAC)

predict_acceleration_at_time(time, MSS, MAC)

predict_time_at_distance(distance, MSS, MAC)

predict_time_at_distance_FV(
  distance,
  F0,
  V0,
  bodymass = 75,
  inertia = 0,
  resistance = 0,
  ...
)

predict_velocity_at_distance(distance, MSS, MAC)

predict_acceleration_at_distance(distance, MSS, MAC)

predict_acceleration_at_velocity(velocity, MSS, MAC)

predict_air_resistance_at_time(time, MSS, MAC, ...)

predict_air_resistance_at_distance(distance, MSS, MAC, ...)

predict_force_at_velocity(
  velocity,
  MSS,
  MAC,
  bodymass = 75,
  inertia = 0,
  resistance = 0,
  ...
)
```

```
predict_force_at_time(
  time,
  MSS,
  MAC,
  bodymass = 75,
  inertia = 0,
  resistance = 0,
  ...
)

predict_force_at_distance(
  distance,
  MSS,
  MAC,
  bodymass = 75,
  inertia = 0,
  resistance = 0,
  ...
)

predict_power_at_distance(
  distance,
  MSS,
  MAC,
  bodymass = 75,
  inertia = 0,
  resistance = 0,
  ...
)

predict_power_at_time(
  time,
  MSS,
  MAC,
  bodymass = 75,
  inertia = 0,
  resistance = 0,
  ...
)

predict_relative_power_at_distance(
  distance,
  MSS,
  MAC,
  bodymass = 75,
  inertia = 0,
  resistance = 0,
  ...
```

```
)

predict_relative_power_at_time(
  time,
  MSS,
  MAC,
  bodymass = 75,
  inertia = 0,
  resistance = 0,
  ...
)

predict_work_till_time(time, ...)

predict_work_till_distance(distance, ...)

predict_kinematics(
  object = NULL,
  MSS,
  MAC,
  max_time = 6,
  frequency = 100,
  bodymass = 75,
  inertia = 0,
  resistance = 0,
  add_inertia_to_vertical = TRUE,
  ...
)
```

Arguments

`time, distance, velocity`
: Numeric vectors

`MSS, MAC`
: Numeric vectors. Model parameters

`F0, V0`
: Numeric vectors. FV profile parameters

`bodymass`
: Body mass in kg. Used to calculate relative power and forwarded to `get_air_resistance`

`inertia`
: External inertia in kg (for example a weight vest, or a sled). Not included in the air resistance calculation

`resistance`
: External horizontal resistance in Newtons (for example tether device or a sled friction resistance)

`...`
: Arguments passed on to `get_air_resistance`

  `bodyheight` In meters (m). Default is 1.75m

  `barometric_pressure` In Torrs. Default is 760Torrs

  `air_temperature` In Celzius (C). Default is 25C

  `wind_velocity` In meters per second (m/s). Use negative number as head wind, and positive number as back wind. Default is 0m/s (no wind) `object` If `shorts_model` object is provided, estimated parameters will be used. Otherwise provide `MSS` and `MAC` parameters

`max_time` Predict from 0 to `max_time`. Default is 6seconds

`frequency` Number of samples within one second. Default is 100Hz

`add_inertia_to_vertical`
Should inertia be added to `bodymass` when calculating vertical force? Use `TRUE` (Default) when using weight vest, and `FALSE` when dragging sled

Value

Numeric vector

Data frame with kinetic and kinematic variables

References

Haugen TA, Tønnessen E, Seiler SK. 2012. The Difference Is in the Start: Impact of Timing and Start Procedure on Sprint Running Performance: Journal of Strength and Conditioning Research 26:473–479. DOI: 10.1519/JSC.0b013e318226030b.

Jovanović, M., Vescovi, J.D. (2020). shorts: An R Package for Modeling Short Sprints. Preprint available at SportRxiv. https://doi.org/10.31236/osf.io/4jw62

Samozino P. 2018. A Simple Method for Measuring Force, Velocity and Power Capabilities and Mechanical Effectiveness During Sprint Running. In: Morin J-B, Samozino P eds. Biomechanics of Training and Testing. Cham: Springer International Publishing, 237–267. DOI: 10.1007/978-3-319-05633-3_11.

Examples

```
MSS <- 8
MAC <- 9

time_seq <- seq(0, 6, length.out = 10)

df <- data.frame(
  time = time_seq,
  distance_at_time = predict_distance_at_time(time_seq, MSS, MAC),
  velocity_at_time = predict_velocity_at_time(time_seq, MSS, MAC),
  acceleration_at_time = predict_acceleration_at_time(time_seq, MSS, MAC)
)

df$time_at_distance <- predict_time_at_distance(df$distance_at_time, MSS, MAC)
df$velocity_at_distance <- predict_velocity_at_distance(df$distance_at_time, MSS, MAC)
df$acceleration_at_distance <- predict_acceleration_at_distance(df$distance_at_time, MSS, MAC)
df$acceleration_at_velocity <- predict_acceleration_at_velocity(df$velocity_at_time, MSS, MAC)

# Power calculation uses shorts::get_air_resistance function and its defaults
# values to calculate power. Use the ... to setup your own parameters for power
# calculations
df$power_at_time <- predict_power_at_time(
  time = df$time, MSS = MSS, MAC = MAC,
```

```
  # Check shorts::get_air_resistance for available params
  bodymass = 100, bodyheight = 1.85
)

df

# Example for predict_kinematics
split_times <- data.frame(
  distance = c(5, 10, 20, 30, 35),
  time = c(1.20, 1.96, 3.36, 4.71, 5.35)
)

# Simple model
simple_model <- with(
  split_times,
  model_timing_gates(distance, time)
)

predict_kinematics(simple_model)
```

print.shorts_model   *S3 method for printing* shorts_model *object*

Description

S3 method for printing shorts_model object

Usage

```
## S3 method for class 'shorts_model'
print(x, ...)
```

Arguments

x	shorts_model object
...	Not used

Examples

```
split_distances <- c(10, 20, 30, 40, 50)
split_times <- create_timing_gates_splits(
  gates = split_distances,
  MSS = 10,
  MAC = 9,
  FD = 0.25,
  TC = 0
)
```

```
# Simple model
simple_model <- model_timing_gates(split_distances, split_times)
simple_model
```

---

probe_functions *Probe profile functions*

Description

Family of functions that serve a purpose of probing sprint or force-velocity profile. This is done by increasing individual sprint parameter for a percentage and calculating which parameter improvement yield biggest deduction in sprint tim

`probe_FV` "probes" `F0` and `V0` and calculates which one improves sprint time for a defined `distance`

`probe_MSS_MAC` "probes" `MSS` and `MAC` and calculates which one improves sprint time for a defined `distance`

Usage

```
probe_FV(
  distance,
  F0,
  V0,
  bodymass = 75,
  inertia = 0,
  resistance = 0,
  perc = 2.5,
  ...
)

probe_MSS_MAC(distance, MSS, MAC, perc = 2.5)
```

Arguments

`distance`	Numeric vector
`F0`, `V0`	Numeric vectors. FV profile parameters
`bodymass`	Body mass in kg
`inertia`	External inertia in kg (for example a weight vest, or a sled). Not included in the air resistance calculation
`resistance`	External horizontal resistance in Newtons (for example tether device or a sled friction resistance)
`perc`	Numeric vector. Probing percentage. Default is 2.5 percent
`...`	Arguments passed on to `get_air_resistance` `velocity` Instantaneous running velocity in meters per second (m/s) `bodyheight` In meters (m). Default is 1.75m

`barometric_pressure` In Torrs. Default is 760Torrs

`air_temperature` In Celzius (C). Default is 25C

`wind_velocity` In meters per second (m/s). Use negative number as head wind, and positive number as back wind. Default is 0m/s (no wind)

`MSS`, `MAC` Numeric vectors. Model parameters

Value

`probe_FV` returns a data frame with the following columns

F0 Original F0
V0 Original F0
bodymass Bodymass
inertia Inertia
resistance Resistance
Pmax Maximal power estimated using F0 * V0 / 4
Pmax_rel Relative maximal power
slope FV profile slope
distance Distance
time Time to cover distance
probe_perc Probe percentage
F0_probe Probing F0
F0_probe_time Predicted time for distance when F0 is probed
F0_probe_time_gain Difference in time to cover distance between time_optimal and time
V0_probe Probing V0
V0_probe_time Predicted time for distance when V0 is probed
V0_probe_time_gain Difference in time to cover distance between time_optimal and time
profile_imb Percent ratio between V0_probe_time_gain and F0_probe_time_gain

`probe_MSS_MAC` returns a data frame with the following columns

MSS Original MSS
MAC Original MAC
Pmax_rel Relative maximal power estimated using MSS * MAC / 4
slope Sprint profile slope
distance Distance
time Time to cover distance
probe_perc Probe percentage
MSS_probe Probing MSS
MSS_probe_time Predicted time for distance when MSS is probed
MSS_probe_time_gain Difference in time to cover distance between probe time and time
MAC_probe Probing MAC
MAC_probe_time Predicted time for distance when MAC is probed
MAC_probe_time_gain Difference in time to cover distance between probing time and time
profile_imb Percent ratio between MSS_probe_time_gain and MAC_probe_time_gain

Examples

```
MSS <- 10
MAC <- 8
bodymass <- 75

fv <- create_FVP(MSS, MAC, bodymass)

dist <- seq(5, 40, by = 5)

probe_MSS_MAC_profile <- probe_MSS_MAC(
  distance = dist,
  MSS,
  MAC
)[["profile_imb"]]

probe_FV_profile <- probe_FV(
  distance = dist,
  fv$F0,
  fv$V0,
  fv$bodymass
)[["profile_imb"]]

plot(x = dist, y = probe_MSS_MAC_profile, type = "l", ylab = "Profile imbalance")
lines(x = dist, y = probe_FV_profile, type = "l", col = "blue")
abline(h = 100, col = "gray", lty = 2)
```

---

radar_gun_data *Radar Gun Data*

Description

Data generated from known MSS and TAU and measurement error for N=5 athletes using radar gun with sampling frequency of 100Hz over 6 seconds.

Usage

```
data(radar_gun_data)
```

Format

Data frame with 4 variables and 3000 observations:

athlete Character string

bodyweight Bodyweight in kilograms

time Time reported by the radar gun in seconds

velocity Velocity reported by the radar gun in m/s

`residuals.shorts_model` *S3 method for returning residuals of* `shorts_model`

Description

S3 method for returning residuals of `shorts_model`

Usage

```
## S3 method for class 'shorts_model'
residuals(object, ...)
```

Arguments

`object`	`shorts_model` object
`...`	Extra arguments. Not used

Examples

```
split_distances <- c(10, 20, 30, 40, 50)
split_times <- create_timing_gates_splits(
  gates = split_distances,
  MSS = 10,
  MAC = 9,
  FD = 0.25,
  TC = 0
)

# Simple model
simple_model <- model_timing_gates(split_distances, split_times)
residuals(simple_model)
```

`split_times` *Split Testing Data*

Description

Data generated from known MSS and TAU and measurement error for N=5 athletes using 6 timing gates: 5m, 10m, 15m, 20m, 30m, 40m

Usage

```
data(split_times)
```

Format

Data frame with 4 variables and 30 observations:

athlete Character string

bodyweight Bodyweight in kilograms

distance Distance of the timing gates from the sprint start in meters

time Time reported by the timing gate

---

summary.shorts_model *S3 method for providing summary for the* shorts_model *object*

Description

S3 method for providing summary for the `shorts_model` object

Usage

```
## S3 method for class 'shorts_model'
summary(object, ...)
```

Arguments

`object`	`shorts_model` object
`...`	Not used

Examples

```
split_distances <- c(10, 20, 30, 40, 50)
split_times <- create_timing_gates_splits(
  gates = split_distances,
  MSS = 10,
  MAC = 9,
  FD = 0.25,
  TC = 0
)

# Simple model
simple_model <- model_timing_gates(split_distances, split_times)
summary(simple_model)
```

`vescovi` *Vescovi Timing Gates Sprint Times*

Description

Timing gates sprint times involving 52 female athletes. Timing gates were located at 5m, 10m, 20m, 30m, and 35m. See **Details** for more information.

Usage

```
data(vescovi)
```

Format

Data frame with 17 variables and 52 observations:

Team Team or sport. Contains the following levels: 'W Soccer' (Women Soccer), 'FH Sr' (Field Hockey Seniors), 'FH U21' (Field Hockey Under 21), and 'FH U17' (Field Hockey Under 17)

Surface Type of testing surface. Contains the following levels: 'Hard Cours' and 'Natural Grass'

Athlete Athlete ID

Age Athlete age in years

Height Body height in cm

Bodyweight Body weight in kg

BMI Body Mass Index

BSA Body Surface Area. Calculated using Mosteller equation `sqrt((height/weight)/3600)`

5m Time in seconds at 5m gate

10m Time in seconds at 10m gate

20m Time in seconds at 20m gate

30m Time in seconds at 30m gate

35m Time in seconds at 35m gate

10m-5m split Split time in seconds between 10m and 5m gate

20m-10m split Split time in seconds between 20m and 10m gate

30m-20m split Split time in seconds between 30m and 20m gate

35m-30m split Split time in seconds between 35m and 30m gate

Details

This data-set represents sub-set of data from a total of 220 high-level female athletes (151 soccer players and 69 field hockey players). Using a random number generator, a total of 52 players (35 soccer and 17 field hockey) were selected for this data-set. Soccer players were older (24.6±3.6 vs. 18.9±2.7 yr, $p < 0.001$), however there were no differences for height (167.3±5.9 vs. 167.0±5.7 cm, $p = 0.886$), body mass (62.5±5.9 vs. 64.0±9.4 kg, $p = 0.500$) or any sprint interval time ($p > 0.650$).

The protocol for assessing linear sprint speed has been described previously (Vescovi 2014, 2016, 2012) and was identical for each cohort. Briefly, all athletes performed a standardized warm-up that included general exercises such as jogging, shuffling, multi-directional movements, and dynamic stretching exercises. Infrared timing gates (Brower Timing, Utah) were positioned at the start line and at 5, 10, 20, and 35 meters at a height of approximately 1.0 meter. Participants stood with their lead foot positioned approximately 5 cm behind the initial infrared beam (i.e., start line). Only forward movement was permitted (no leaning or rocking backwards) and timing started when the laser of the starting gate was triggered. The best 35 m time, and all associated split times were kept for analysis. The assessment of linear sprints using infrared timing gates does not require familiarization (Moir, Button, Glaister, and Stone 2004).

Author(s)

Jason D. Vescovi
University of Toronto
Faculty of Kinesiology and Physical Education
Graduate School of Exercise Science
Toronto, ON Canada
<vescovij@gmail.com>

References

Moir G, Button C, Glaister M, Stone MH (2004). "Influence of Familiarization on the Reliability of Vertical Jump and Acceleration Sprinting Performance in Physically Active Men." The Journal of Strength and Conditioning Research, 18(2), 276. ISSN 1064-8011, 1533-4287. doi:10.1519/R-13093.1.

Vescovi JD (2012). "Sprint Speed Characteristics of High-Level American Female Soccer Players: Female Athletes in Motion (FAiM) Study." Journal of Science and Medicine in Sport, 15(5), 474-478. ISSN 14402440. doi:10.1016/j.jsams.2012.03.006.

Vescovi JD (2014). "Impact of Maximum Speed on Sprint Performance During High-Level Youth Female Field Hockey Matches: Female Athletes in Motion (FAiM) Study." International Journal of Sports Physiology and Performance, 9(4), 621-626. ISSN 1555-0265, 1555-0273. doi:10.1123/ijspp.2013-0263.

Vescovi JD (2016). "Locomotor, Heart-Rate, and Metabolic Power Characteristics of Youth Women's Field Hockey: Female Athletes in Motion (FAiM) Study." Research Quarterly for Exercise and Sport, 87(1), 68-77. ISSN 0270-1367, 2168-3824. doi:10.1080/02701367.2015.1124972.

Index