

Package ‘shorts’

November 29, 2020

Type Package

Title Short Sprints

Version 1.1.4

Description Create short sprint (<6sec) profiles using the split times or the radar gun data. Mono-exponential equation is used to estimate maximal sprinting speed (MSS), relative acceleration (TAU), and other parameters such as maximal acceleration (MAC) and maximal relative power (P_{MAX}). These parameters can be used to predict kinematic and kinetics variables and to compare individuals. The modeling method utilized in this package is based on the works of 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>, Furusawa K, Hill AV, Parkinson JL (1927) <doi: 10.1098/rspb.1927.0035>, Greene PR. (1986) <doi: 10.1016/0025-5564(86)90063-5>, and Samozino P. (2018) <doi: 10.1007/978-3-319-05633-3_11>.

URL <https://mladenjovanovic.github.io/shorts/>

BugReports <https://github.com/mladenjovanovic/shorts/issues>

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends R (>= 2.10)

Imports stats, LambertW, nlme, tidyr, ggplot2

Suggests knitr, rmarkdown, tidyverse

VignetteBuilder knitr

NeedsCompilation no

Author Mladen Jovanović [aut, cre],
Jason D. Vescovi [dct]

Maintainer Mladen Jovanović <coach.mladen.jovanovic@gmail.com>

Repository CRAN

Date/Publication 2020-11-29 21:10:02 UTC

R topics documented:

coef.shorts_mixed_model	2
coef.shorts_model	3
find_functions	4
format_splits	7
get_air_resistance	8
get_FV_profile	9
jb_morin	10
mixed_model_radar	11
mixed_model_split_times	13
model_radar	16
model_split_times	18
plot.shorts_fv_profile	20
plot.shorts_mixed_model	21
plot.shorts_model	22
predict.shorts_mixed_model	23
predict.shorts_model	24
predict_kinematics	24
print.shorts_fv_profile	28
print.shorts_mixed_model	29
print.shorts_model	29
radar_gun_data	30
residuals.shorts_mixed_model	30
residuals.shorts_model	31
split_times	32
summary.shorts_mixed_model	32
summary.shorts_model	33
vescovi	34
Index	36

```
coef.shorts_mixed_model
```

S3 method for extracting model parameters from shorts_mixed_model object

Description

S3 method for extracting model parameters from shorts_mixed_model object

Usage

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

Arguments

object shorts_mixed_model object
... Extra arguments. Not used

Examples

```
data("split_times")

mixed_model <- mixed_model_using_splits(
  data = split_times,
  distance = "distance",
  time = "time",
  athlete = "athlete"
)

# mixed_model$parameters
coef(mixed_model)
```

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_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_using_splits(distance, time)
)
```

```
# unlist(simple_model$parameters)
coef(simple_model)
```

find_functions	<i>Find functions</i>
----------------	-----------------------

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_max_power_distance finds maximum power and distance at which max power occurs

find_max_power_time finds maximum power and time at which max 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 maximal power over percent is achieved

find_power_critical_time finds critical times at which maximal power over percent is achieved

Usage

```
find_max_power_distance(
  MSS,
  TAU,
  time_correction = 0,
  distance_correction = 0,
  ...
)
```

```
find_max_power_time(MSS, TAU, time_correction = 0, ...)
```

```
find_velocity_critical_distance(
  MSS,
  TAU,
  time_correction = 0,
  distance_correction = 0,
  percent = 0.9
)
```

```
find_velocity_critical_time(MSS, TAU, time_correction = 0, percent = 0.9)
```

```
find_acceleration_critical_distance(
```

```

    MSS,
    TAU,
    time_correction = 0,
    distance_correction = 0,
    percent = 0.9
)

find_acceleration_critical_time(MSS, TAU, time_correction = 0, percent = 0.9)

find_power_critical_distance(
  MSS,
  TAU,
  time_correction = 0,
  distance_correction = 0,
  percent = 0.9,
  ...
)

find_power_critical_time(MSS, TAU, time_correction = 0, percent = 0.9, ...)

```

Arguments

MSS, TAU	Numeric vectors. Model parameters
time_correction	Numeric vector. Used for correction. Default is 0. See references for more info
distance_correction	Numeric vector. Used for correction. Default is 0. See vignettes for more info
...	Forwarded to predict_power_at_distance for the purpose of calculation of air resistance
percent	Numeric vector. Used to calculate critical distance. Default is 0.9

Value

find_max_power_distance returns list with two elements: max_power and distance at which max power occurs

find_max_power_time returns list with two elements: max_power and time at which max 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,
  TAU = 0.9
)

acceleration <- predict_acceleration_at_distance(
  distance = dist,
  MSS = 10,
  TAU = 0.9
)

# Use ... to forward parameters to the shorts::get_air_resistance
pwr <- predict_relative_power_at_distance(
  distance = dist,
  MSS = 10,
  TAU = 0.9
  # 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, TAU = 0.9), col = "red")

# Find critical distance when 20% of MAC is reached
plot(x = dist, y = acceleration, type = "l")
abline(h = (10 / 0.9) * 0.2, col = "gray")
abline(v = find_acceleration_critical_distance(MSS = 10, TAU = 0.9, percent = 0.2), col = "red")

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

max_pwr <- find_max_power_distance(
  MSS = 10,
  TAU = 0.9
  # Use ... to forward parameters to the shorts::get_air_resistance
)
abline(h = max_pwr$max_power, col = "gray")
abline(v = max_pwr$distance, col = "red")

# Find distance in which relative power stays over 75% of PMAX'
plot(x = dist, y = pwr, type = "l")
abline(h = max_pwr$max_power * 0.75, col = "gray")
pwr_zone <- find_power_critical_distance(MSS = 10, TAU = 0.9, percent = 0.75)

```

```
abline(v = pwr_zone$lower, col = "blue")
abline(v = pwr_zone$upper, col = "blue")
```

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

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)
bodyheight	In meters (m)
barometric_pressure	In Torrs
air_temperature	In Celcius (C)
wind_velocity	In meters per second (m/s). Use negative number as head wind, and positive number as back 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/jappphysiol.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
)

```

get_FV_profile	<i>Get Force-Velocity Profile</i>
----------------	-----------------------------------

Description

Provides Force-Velocity (FV) profile suggested by Pierre Samozino and JB-Morin, et al.

Usage

```

get_FV_profile(
  MSS,
  TAU,
  bodymass = 75,
  max_time = 6,
  frequency = 100,
  RFmax_cutoff = 0.3,
  ...
)

```

Arguments

MSS	Numeric vectors. Model parameters
TAU	Numeric vectors. Model parameters
bodymass	Body mass in kg. Used to calculate relative power and forwarded to get_air_resistance
max_time	Predict from 0 to max_time. Default is 6seconds
frequency	Number of samples within one second. Default is 100Hz
RFmax_cutoff	Time cut-off used to estimate RFmax and Drf. Default is 0.3s
...	Forwarded to get_air_resistance for the purpose of calculation of air resistance and power

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 horizonatl force=0
Pmax Maximal horizontal power
Pmax_rel Pmax divided by bodymass
FV_slope Slope of the FV profile. See References for more info
RFmax Maximal force ratio after 0.3sec. See References for more info
RFmax_cutoff Time cut-off used to estimate RFmax
Drf Slope of Force Ratio (RF) and velocity. See References for more info
RSE_FV Residual standard error of the FV profile.
RSE_Drf Residual standard error of the RF-velocity profile
data Data frame containing simulated data used to estimate parameters

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.

Examples

```
data("jb_morin")

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

fv_profile <- get_FV_profile(
  MSS = m1$parameters$MSS,
  TAU = m1$parameters$TAU,
  bodyheight = 1.72,
  bodymass = 120)

print(fv_profile)
plot(fv_profile)
plot(fv_profile, "time")
```

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).

mixed_model_radar

Mixed Model Using Instantaneous Velocity

Description

This function models the sprint instantaneous velocity using mono-exponential equation and non-linear mixed model using `nlme` to estimate fixed and random maximum sprinting speed (MSS) and relative acceleration (TAU) parameters. In mixed model, fixed and random effects are estimated for MSS and TAU parameters using athlete as levels. velocity is used as target or outcome variable, and time as predictor.

Usage

```

mixed_model_using_radar(
  data,
  time,
  velocity,
  athlete,
  time_correction = 0,
  random = MSS + TAU ~ 1,
  LOOCV = FALSE,
  na.rm = FALSE,
  ...
)

mixed_model_using_radar_with_time_correction(
  data,
  time,
  velocity,
  athlete,
  random = MSS + TAU ~ 1,
  LOOCV = FALSE,
  na.rm = FALSE,
  ...
)

```

Arguments

<code>data</code>	Data frame
<code>time</code>	Character string. Name of the column in data
<code>velocity</code>	Character string. Name of the column in data
<code>athlete</code>	Character string. Name of the column in data. Used as levels in the nlme
<code>time_correction</code>	Numeric vector. Used to filter out noisy data from the radar gun. This correction is done by adding <code>time_correction</code> to <code>time</code> . Default is 0. See more in Samozino (2018)
<code>random</code>	Formula forwarded to nlme to set random effects. Default is <code>MSS + TAU ~ 1</code>
<code>LOOCV</code>	Should Leave-one-out cross-validation be used to estimate model fit? Default is FALSE. This can be very slow process due high level of samples in the radar data
<code>na.rm</code>	Logical. Default is FALSE
<code>...</code>	Forwarded to nlme function

Value

List object with the following elements:

parameters List with two data frames: fixed and random containing the following estimated parameters: MSS, TAU, MAC, and PMAX

- model_fit** List with the following components: RSE, R_squared, minErr, maxErr, and RMSE
- model** Model returned by the `nlme` function
- data** Data frame used to estimate the sprint parameters, consisting of athlete, time, velocity, and pred_velocity columns

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.

Examples

```
data("radar_gun_data")

mixed_model <- mixed_model_using_radar(radar_gun_data, "time", "velocity", "athlete")

print(mixed_model)
coef(mixed_model)
plot(mixed_model)

mixed_model_correction <- mixed_model_using_radar_with_time_correction(
  radar_gun_data,
  "time",
  "velocity",
  "athlete")

print(mixed_model_correction)
coef(mixed_model_correction)
plot(mixed_model_correction)
```

mixed_model_split_times

Mixed Models Using Split Times

Description

These functions model the sprint split times using mono-exponential equation, where time is used as target or outcome variable, and distance as predictor. Function `mixed_model_using_splits` provides the simplest model with estimated MSS and TAU parameters. Time correction using heuristic rule of thumbs (e.g., adding 0.3s to split times) can be implemented using `time_correction` function parameter. Function `mixed_model_using_splits_with_time_correction`, besides estimating MSS and TAU, estimates additional parameter `time_correction`. Function `mixed_model_using_splits_with_correction`, besides estimating MSS, TAU and `time_correction`, estimates additional parameter `distance_correction`. For more information about these function please refer to accompanying vignettes in this package.

Usage

```

mixed_model_using_splits(
  data,
  distance,
  time,
  athlete,
  time_correction = 0,
  random = MSS + TAU ~ 1,
  LOOCV = FALSE,
  na.rm = FALSE,
  ...
)

mixed_model_using_splits_with_time_correction(
  data,
  distance,
  time,
  athlete,
  random = MSS + TAU ~ 1,
  LOOCV = FALSE,
  na.rm = FALSE,
  ...
)

mixed_model_using_splits_with_corrections(
  data,
  distance,
  time,
  athlete,
  random = MSS + TAU ~ 1,
  LOOCV = FALSE,
  na.rm = FALSE,
  ...
)

```

Arguments

<code>data</code>	Data frame
<code>distance</code>	Character string. Name of the column in data
<code>time</code>	Character string. Name of the column in data
<code>athlete</code>	Character string. Name of the column in data. Used as levels in the nlme
<code>time_correction</code>	Numeric vector. Used to correct for different starting techniques. This correction is done by adding <code>time_correction</code> to <code>time</code> . Default is 0. See more in Haugen et al. (2018)
<code>random</code>	Formula forwarded to nlme to set random effects. Default is <code>MSS + TAU ~ 1</code>

LOOCV	Should Leave-one-out cross-validation be used to estimate model fit? Default is FALSE
na.rm	Logical. Default is FALSE
...	Forwarded to nlme function

Value

List object with the following elements:

parameters List with two data frames: fixed and random containing the following estimated parameters: MSS, TAU, time_correction, distance_correction, MAC, and PMAX

model_fit List with the following components: RSE, R_squared, minErr, maxErr, and RMSE

model Model returned by the [nlme](#) function

data Data frame used to estimate the sprint parameters, consisting of athlete, distance, time, and pred_time columns

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.

Examples

```
data("split_times")

mixed_model <- mixed_model_using_splits(
  data = split_times,
  distance = "distance",
  time = "time",
  athlete = "athlete"
)

print(mixed_model)
coef(mixed_model)
plot(mixed_model)

mixed_model <- mixed_model_using_splits_with_time_correction(
  data = split_times,
  distance = "distance",
  time = "time",
  athlete = "athlete"
)

print(mixed_model)
coef(mixed_model)
plot(mixed_model)

mixed_model <- mixed_model_using_splits_with_corrections(
  data = split_times,
```

```
distance = "distance",
time = "time",
athlete = "athlete"
)

print(mixed_model)
coef(mixed_model)
plot(mixed_model)
```

model_radar

Model Using Instantaneous Velocity or Radar Gun

Description

This function models the sprint instantaneous velocity using mono-exponential equation that estimates maximum sprinting speed (MSS) and relative acceleration (TAU). velocity is used as target or outcome variable, and time as predictor.

Usage

```
model_using_radar(
  time,
  velocity,
  time_correction = 0,
  weights = 1,
  LOOCV = FALSE,
  na.rm = FALSE,
  ...
)

model_using_radar_with_time_correction(
  time,
  velocity,
  weights = 1,
  LOOCV = FALSE,
  na.rm = FALSE,
  ...
)
```

Arguments

time	Numeric vector
velocity	Numeric vector
time_correction	Numeric vector. Used to filter out noisy data from the radar gun. This correction is done by adding time_correction to time. Default is 0. See more in Samozino (2018)

weights	Numeric vector. Default is 1
LOOCV	Should Leave-one-out cross-validation be used to estimate model fit? Default is FALSE
na.rm	Logical. Default is FALSE
...	Forwarded to nlme function

Value

List object with the following elements:

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

model_fit List with the following components: RSE, R_squared, minErr, maxErr, and RMSE

model Model returned by the [nls](#) function

data Data frame used to estimate the sprint parameters, consisting of time, velocity, weights, and pred_velocity columns

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.

Examples

```
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)
)

sprint_model <- with(
  instant_velocity,
  model_using_radar(time, velocity)
)

print(sprint_model)
coef(sprint_model)
plot(sprint_model)

sprint_model_correction <- with(
  instant_velocity,
  model_using_radar_with_time_correction(time + 0.3, velocity)
)

print(sprint_model_correction)
coef(sprint_model_correction)
plot(sprint_model_correction)
```

Description

These functions model the sprint split times using mono-exponential equation, where time is used as target or outcome variable, and distance as predictor. Function `model_using_splits` provides the simplest model with estimated MSS and TAU parameters. Time correction using heuristic rule of thumbs (e.g., adding 0.3s to split times) can be implemented using `time_correction` function parameter. Function `model_using_splits_with_time_correction`, besides estimating MSS and TAU, estimates additional parameter `time_correction`. Function `model_using_splits_with_corrections`, besides estimating MSS, TAU and `time_correction`, estimates additional parameter `distance_correction`. For more information about these function please refer to accompanying vignettes in this package.

Usage

```
model_using_splits(  
  distance,  
  time,  
  time_correction = 0,  
  weights = 1,  
  LOOCV = FALSE,  
  na.rm = FALSE,  
  ...  
)  
  
model_using_splits_with_time_correction(  
  distance,  
  time,  
  weights = 1,  
  LOOCV = FALSE,  
  na.rm = FALSE,  
  ...  
)  
  
model_using_splits_with_corrections(  
  distance,  
  time,  
  weights = 1,  
  LOOCV = FALSE,  
  na.rm = FALSE,  
  ...  
)
```

Arguments

`distance`, `time` Numeric vector. Indicates the position of the timing gates and time measured

time_correction	Numeric vector. Used to correct for different starting techniques. This correction is done by adding time_correction to time. Default is 0. See more in Haugen et al. (2018)
weights	Numeric vector. Default is vector of 1. This is used to give more weight to particular observations. For example, use 1\distance to give more weight to observations from shorter distances.
LOOCV	Should Leave-one-out cross-validation be used to estimate model fit? Default is FALSE
na.rm	Logical. Default is FALSE
...	Forwarded to nls function

Value

List object with the following elements:

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

model_fit List with the following components: RSE, R_squared, minErr, maxErr, and RMSE

model Model returned by the [nls](#) function

data Data frame used to estimate the sprint parameters, consisting of distance, time, weights, and pred_time columns

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.

Examples

```
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_using_splits(distance, time)
)

print(simple_model)
coef(simple_model)
plot(simple_model)

# Model with correction of 0.3s
model_with_correction <- with(
  split_times,
```

```
  model_using_splits(distance, time, time_correction = 0.3)
)

print(model_with_correction)
coef(model_with_correction)
plot(model_with_correction)

# Model with time_correction estimation
model_with_time_correction_estimation <- with(
  split_times,
  model_using_splits_with_time_correction(distance, time)
)

print(model_with_time_correction_estimation)
coef(model_with_time_correction_estimation)
plot(model_with_time_correction_estimation)

# Model with time and distance correction estimation
model_with_time_distance_correction_estimation <- with(
  split_times,
  model_using_splits_with_corrections(distance, time)
)

print(model_with_time_distance_correction_estimation)
coef(model_with_time_distance_correction_estimation)
plot(model_with_time_distance_correction_estimation)
```

plot.shorts_fv_profile

S3 method for plotting shorts_fv_profile object

Description

S3 method for plotting shorts_fv_profile object

Usage

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

Arguments

x	shorts_fv_profile object
type	Type of plot. Options are "velocity" (default) and "time"
...	Not used

Value

[ggplot](#) object

Examples

```
data("jb_morin")

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

fv_profile <- get_FV_profile(
  MSS = m1$parameters$MSS,
  TAU = m1$parameters$TAU,
  bodyheight = 1.72,
  bodymass = 120)

plot(fv_profile)
plot(fv_profile, "time")
```

```
plot.shorts_mixed_model
```

S3 method for plotting shorts_mixed_model object

Description

S3 method for plotting shorts_mixed_model object

Usage

```
## S3 method for class 'shorts_mixed_model'
plot(x, type = NULL, ...)
```

Arguments

x	shorts_mixed_model object
type	Not used
...	Not used

Value

[ggplot](#) object

Examples

```
# Split times
data("split_times")
mixed_model_splits <- mixed_model_using_splits(
  data = split_times,
  distance = "distance",
  time = "time",
  athlete = "athlete"
)
```

```
plot(mixed_model_splits)

# Radar gun data
data("radar_gun_data")
mixed_model_radar <- mixed_model_using_radar(radar_gun_data, "time", "velocity", "athlete")

plot(mixed_model_radar)
```

plot.shortcuts_model *S3 method for plotting shortcuts_model object*

Description

S3 method for plotting shortcuts_model object

Usage

```
## S3 method for class 'shortcuts_model'
plot(x, type = NULL, ...)
```

Arguments

x	shortcuts_model object
type	Not used
...	Not used

Value

[ggplot](#) object

Examples

```
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 with time splits
simple_model <- with(
  split_times,
  model_using_splits(distance, time)
)

coef(simple_model)
plot(simple_model)

# 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_using_radar(time, velocity)
)

# sprint_model$parameters
coef(radar_model)
plot(radar_model)
```

```
predict.shorts_mixed_model
```

S3 method for returning predictions of shorts_mixed_model

Description

S3 method for returning predictions of shorts_mixed_model

Usage

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

Arguments

object	shorts_mixed_model object
...	Extra arguments. Not used

Examples

```
data("split_times")

mixed_model <- mixed_model_using_splits(
  data = split_times,
  distance = "distance",
  time = "time",
  athlete = "athlete"
)

predict(mixed_model)
```

predict.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'  
predict(object, ...)
```

Arguments

object	shorts_model object
...	Extra arguments. Not used

Examples

```
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_using_splits(distance, time)  
)  
  
predict(simple_model)
```

predict_kinematics *Kinematics prediction functions*

Description

Predicts kinematic from known MSS and TAU parameters

Usage

```
predict_velocity_at_time(time, MSS, TAU, time_correction = 0)  
  
predict_distance_at_time(  
  time,  
  MSS,
```



```
    TAU,
    time_correction = 0,
    distance_correction = 0
)

predict_acceleration_at_time(time, MSS, TAU, time_correction = 0)

predict_time_at_distance(
  distance,
  MSS,
  TAU,
  time_correction = 0,
  distance_correction = 0
)

predict_velocity_at_distance(
  distance,
  MSS,
  TAU,
  time_correction = 0,
  distance_correction = 0
)

predict_acceleration_at_distance(
  distance,
  MSS,
  TAU,
  time_correction = 0,
  distance_correction = 0
)

predict_acceleration_at_velocity(velocity, MSS, TAU)

predict_air_resistance_at_time(time, MSS, TAU, time_correction = 0, ...)

predict_air_resistance_at_distance(
  distance,
  MSS,
  TAU,
  time_correction = 0,
  distance_correction = 0,
  ...
)

predict_force_at_time(time, MSS, TAU, time_correction = 0, bodymass = 75, ...)

predict_force_at_distance(
  distance,
```

```

    MSS,
    TAU,
    time_correction = 0,
    distance_correction = 0,
    bodymass = 75,
    ...
)

predict_power_at_distance(
    distance,
    MSS,
    TAU,
    time_correction = 0,
    distance_correction = 0,
    bodymass = 75,
    ...
)

predict_power_at_time(time, MSS, TAU, time_correction = 0, bodymass = 75, ...)

predict_relative_power_at_distance(
    distance,
    MSS,
    TAU,
    time_correction = 0,
    distance_correction = 0,
    bodymass = 75,
    ...
)

predict_relative_power_at_time(
    time,
    MSS,
    TAU,
    time_correction = 0,
    bodymass = 75,
    ...
)

predict_kinematics(object, max_time = 6, frequency = 100, bodymass = 75, ...)

```

Arguments

time, distance, velocity
 Numeric vectors

MSS, TAU Numeric vectors. Model parameters

time_correction
 Numeric vector. Used for correction. Default is 0. See references for more info

distance_correction	Numeric vector. Used for correction. Default is 0. See vignettes for more info
...	Forwarded to get_air_resistance for the purpose of calculation of air resistance and power
bodymass	Body mass in kg. Used to calculate relative power and forwarded to get_air_resistance
object	shorts_model or shorts_mixed_model object
max_time	Predict from 0 to max_time. Default is 6seconds
frequency	Number of samples within one second. Default is 100Hz

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.

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
TAU <- 0.7

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

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

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

# 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, TAU = TAU,
  # 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_using_splits(distance, time)
)

predict_kinematics(simple_model)

```

```
print.shorts_fv_profile
```

S3 method for printing shorts_fv_profile object

Description

S3 method for printing shorts_fv_profile object

Usage

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

Arguments

x	shorts_fv_profile object
...	Not used

Examples

```

data("jb_morin")

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

fv_profile <- get_FV_profile(
  MSS = m1$parameters$MSS,
  TAU = m1$parameters$TAU,
  bodyheight = 1.72,
  bodymass = 120)

print(fv_profile)

```

```
print.shortcuts_mixed_model
```

S3 method for printing shortcuts_mixed_model object

Description

S3 method for printing shortcuts_mixed_model object

Usage

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

Arguments

x	shortcuts_mixed_model object
...	Not used

Examples

```
data("split_times")  
  
mixed_model <- mixed_model_using_splits(  
  data = split_times,  
  distance = "distance",  
  time = "time",  
  athlete = "athlete"  
)  
  
print(mixed_model)
```

```
print.shortcuts_model
```

S3 method for printing shortcuts_model object

Description

S3 method for printing shortcuts_model object

Usage

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

Arguments

x	shortcuts_model object
...	Not used

Examples

```
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_using_splits(distance, time)
)

print(simple_model)
```

radar_gun_data	<i>Radar Gun Data</i>
----------------	-----------------------

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_mixed_model
```

S3 method for providing residuals for the shorts_mixed_model object

Description

S3 method for providing residuals for the shorts_mixed_model object

Usage

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

Arguments

object	shorts_mixed_model object
...	Not used

Examples

```
data("split_times")  
  
mixed_model <- mixed_model_using_splits(  
  data = split_times,  
  distance = "distance",  
  time = "time",  
  athlete = "athlete"  
)  
  
residuals(mixed_model)
```

residuals.shorts_model

S3 method for providing residuals for the shorts_model object

Description

S3 method for providing residuals for the shorts_model object

Usage

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

Arguments

object	shorts_model object
...	Not used

Examples

```
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_using_splits(distance, time)
)

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.shortcuts_mixed_model

S3 method for providing summary for the shortcuts_mixed_model object

Description

S3 method for providing summary for the shortcuts_mixed_model object

Usage

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

Arguments

object	shortcuts_mixed_model object
...	Not used

Examples

```
data("split_times")

mixed_model <- mixed_model_using_splits(
  data = split_times,
  distance = "distance",
  time = "time",
  athlete = "athlete"
)

summary(mixed_model)
```

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_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_using_splits(distance, time)
)

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 $\sqrt{(\text{height}/\text{weight})/3600}$

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/ijsp.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

* datasets

- jb_morin, 10
 - radar_gun_data, 30
 - split_times, 32
 - vescovi, 34
- coef.shorts_mixed_model, 2
- coef.shorts_model, 3
- find_acceleration_critical_distance
(find_functions), 4
- find_acceleration_critical_time
(find_functions), 4
- find_functions, 4
- find_max_power_distance
(find_functions), 4
- find_max_power_time (find_functions), 4
- find_power_critical_distance
(find_functions), 4
- find_power_critical_time
(find_functions), 4
- find_velocity_critical_distance
(find_functions), 4
- find_velocity_critical_time
(find_functions), 4
- format_splits, 7
- get_air_resistance, 8, 9, 27
- get_FV_profile, 9
- ggplot, 20–22
- jb_morin, 10
- mixed_model_radar, 11
- mixed_model_split_times, 13
- mixed_model_using_radar
(mixed_model_radar), 11
- mixed_model_using_radar_with_time_correction
(mixed_model_radar), 11
- mixed_model_using_splits, 13
- mixed_model_using_splits
(mixed_model_split_times), 13
- mixed_model_using_splits_with_corrections,
13
- mixed_model_using_splits_with_corrections
(mixed_model_split_times), 13
- mixed_model_using_splits_with_time_correction,
13
- mixed_model_using_splits_with_time_correction
(mixed_model_split_times), 13
- model_radar, 16
- model_split_times, 18
- model_using_radar (model_radar), 16
- model_using_radar_with_time_correction
(model_radar), 16
- model_using_splits, 18
- model_using_splits (model_split_times),
18
- model_using_splits_with_corrections,
18
- model_using_splits_with_corrections
(model_split_times), 18
- model_using_splits_with_time_correction,
18
- model_using_splits_with_time_correction
(model_split_times), 18
- nlme, 11–15, 17
- nls, 17, 19
- plot.shorts_fv_profile, 20
- plot.shorts_mixed_model, 21
- plot.shorts_model, 22
- predict.shorts_mixed_model, 23
- predict.shorts_model, 24
- predict_acceleration_at_distance
(predict_kinematics), 24
- predict_acceleration_at_time
(predict_kinematics), 24

predict_acceleration_at_velocity
 (predict_kinematics), 24
predict_air_resistance_at_distance
 (predict_kinematics), 24
predict_air_resistance_at_time
 (predict_kinematics), 24
predict_distance_at_time
 (predict_kinematics), 24
predict_force_at_distance
 (predict_kinematics), 24
predict_force_at_time
 (predict_kinematics), 24
predict_kinematics, 24
predict_power_at_distance, 5
predict_power_at_distance
 (predict_kinematics), 24
predict_power_at_time
 (predict_kinematics), 24
predict_relative_power_at_distance
 (predict_kinematics), 24
predict_relative_power_at_time
 (predict_kinematics), 24
predict_time_at_distance
 (predict_kinematics), 24
predict_velocity_at_distance
 (predict_kinematics), 24
predict_velocity_at_time
 (predict_kinematics), 24
print.shorts_fv_profile, 28
print.shorts_mixed_model, 29
print.shorts_model, 29

radar_gun_data, 30
residuals.shorts_mixed_model, 30
residuals.shorts_model, 31

split_times, 32
summary.shorts_mixed_model, 32
summary.shorts_model, 33

vescovi, 34