International Journal of Strength and Conditioning

Jovanović, M., Vescovi, J. D. (2022).shorts: An R Package for Modeling Short Sprints. International Journal of Strength and Conditioning
https://doi.org/10.47206/ijsc.v2i1.74

{shorts}: An R Package for Modeling Short Sprints

Mladen Jovanović[1]; Jason D. Vescovi[2]

[1]Faculty of Sport and Physical Education, University of Belgrade, Serbia; [2]Faculty of Kinesiology and Physical Education, Graduate School of Exercise Science, Toronto, ON Canada

ABSTRACT

Short sprint performance is one of the most distinguishable and admired physical traits in sports. Short sprints have been modeled using the mono-exponential equation that involves two parameters: (1) maximum sprinting speed (MSS) and (2) relative acceleration (TAU). The most common methods to assess short sprint performance are with a radar gun or timing gates. In this paper, we: 1) provide the **shorts** package that can model sprint timing data from these two sources; 2) discuss potential issues with assessing sprint time (synchronization and flying start, respectively); and 3) provide model definitions within the **shorts** package to help alleviate errors within the subsequent parameter outcomes.

INTRODUCTION

Short sprint performance is one of the most distinguishable and admired physical traits in sports. Short sprints, commonly performed in most team sports (e.g., soccer, field hockey, handball, football, etc.), are defined as maximal running from a stand still position over a distance that doesn't result in deceleration at the end. Peak anaerobic power is achieved within the first few seconds (<5 s) of maximal efforts (Mangine et al. 2014), whereas the ability to achieve maximal sprint speed varies based on the athlete and their sport. For example, track and field sprinters are trained to achieve maximal speed later in a race (i.e., 50-60 m) (Ward-Smith 2001), but team sport athletes have sport-specific attributes and reach it much sooner (i.e., 30-40 m) (Brown, Vescovi, and Vanheest 2004). Regardless of the differences in kinematics between athletes, evaluating short sprint performance is routinely included within a battery of fitness tests for a wide range of sports.

The use of force plates is considered the gold standard for assessing mechanical properties of sprinting; however, there are logistical and financial challenges to capturing the profile of an entire sprint (Jean-Benoit Morin et al. 2019; Samozino et al. 2016). Radar and laser technology are frequently used laboratory-grade methods (Buchheit et al. 2014; Edwards et al. 2020; Jiménez-Reyes et al. 2018; Marcote-Pequeño et al. 2019) but not normally accessible to practitioners working in sports. Undoubtedly, the most common method available and used to evaluate sprint performance are timing gates. Often multiple gates are positioned at varying distances to capture split times (e.g., 5, 10, 20 m), which can now be incorporated into the method for determining sprint mechanical properties (Jean-Benoit Morin et al. 2019; Samozino et al. 2016). This approach presents an advantage to practitioners who can use the outcomes to describe individual differences, quantify the effects of training interventions, and better understanding the limiting factors of performance.

The **shorts** package (Jovanović 2021), written in the R language (R Core Team 2021), represents an open-source tool to help sports scientists translate raw timing data into detailed mechanical outcomes through mathematical modeling (Jean-Benoit Morin et al. 2019; Samozino et al. 2016). To best of our knowledge, scientist, researchers, and coaches have been performing short sprints modeling using the built-in solver function of Excel (Microsoft Corporation, Redmond, Washington, United States) (J. B. Morin 2017; Jean-Benoit Morin and Samozino 2019; Stenroth, Vartiainen, and Karjalainen 2020; Stenroth and Vartiainen 2020; Samozino et al. 2016; Clark et al. 2017; Jean-Benoit Morin et al. 2019), which makes the **shorts** package a major improvement in ease-of-use, speed, transparency, reproducibility, and more feature-rich model fitting.

In the current paper, we will provide an explanation of one commonly used mathematical equation to model short sprints, modeling applications using the **shorts** package, issues that can arise during measurement and estimation, and potential solutions to those problems.

MATHEMATICAL MODEL

Short sprints have been modeled using the mono-exponential equation (1) originally proposed by Furusawa, Hill, and Parkinson (1927), and more recently popularized by Clark et al. (2017), and Samozino et al. (2016). Equation (1) represents the function for instantaneous horizontal velocity v given the time t and two model parameters:

$$v(t) = MSS \times \left(1 - e^{-\frac{t}{TAU}}\right) \qquad (1)$$

The parameters of the equation (1) are *maximum sprinting speed* (MSS; expressed in ms^{-1} and *relative acceleration* (TAU; expressed in *s*). Mathematically, TAU represents the ratio of MSS to initial acceleration (MAC; *maximal acceleration*, expressed in ms^{-2} (2).

$$MAC = \frac{MSS}{TAU} \qquad (2)$$

Given the equation (1), TAU can be interpreted as the time required to reach a sprinting velocity equal to 63.2% of MSS.

Although TAU is used in the equations, and later estimated, we prefer to use MAC instead since it is easier to grasp, particularly for less math inclined coaches.

By derivating equation (1), we can get the equation for horizontal acceleration (3).

$$a(t) = \frac{MSS}{TAU} \times e^{-\frac{t}{TAU}} \qquad (3)$$

By integrating equation (1), we can get the equation for distance covered (4).

$$d(t) = MSS \times \left(t + TAU \times e^{-\frac{t}{TAU}}\right) - MSS \times TAU \qquad (4)$$

Let's consider four athletes with different levels of MSS (high versus low maximal sprinting speed) and MAC (high versus low maximal acceleration; as mentioned previously, using MAC is preferred over using TAU) (Table 1).

Figure 1 depicts distance, velocity, and acceleration over time (from 0 to 6 s).

Table 1. Four athletes with different MSS and MAC parameters.

Athlete	MSS	MAC	TAU
Athlete A	12	10	1.20
Athlete B	12	6	2.00
Athlete C	8	10	0.80
Athlete D	8	6	1.33

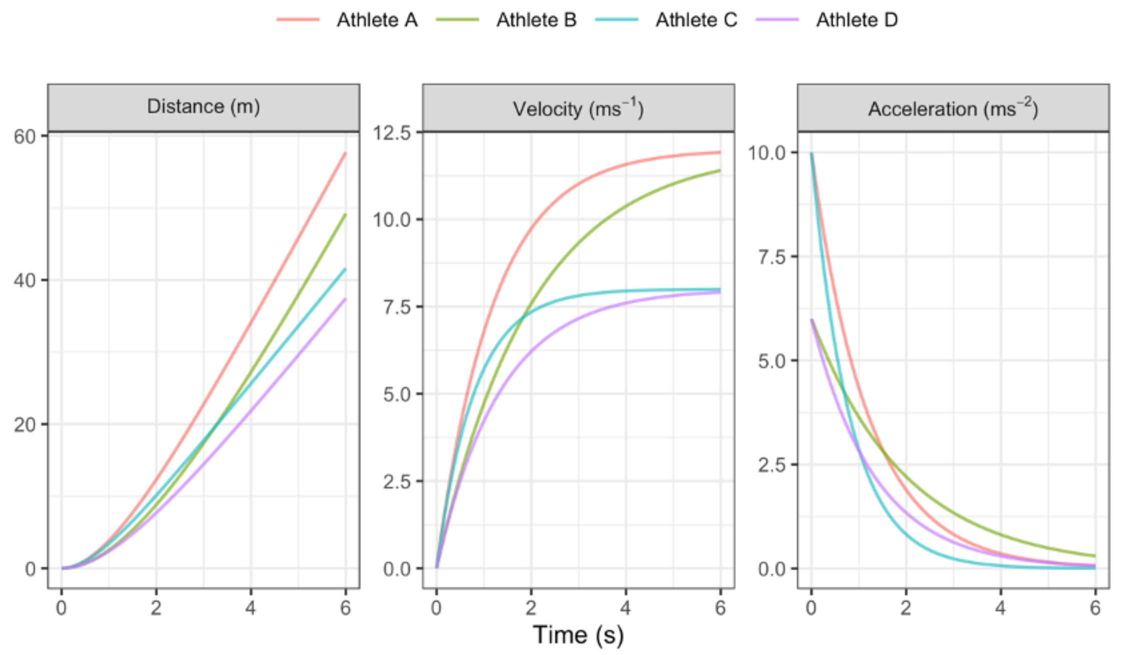


Figure 1. Kinematic characteristic of four athletes with different MSS and MAC parameters over a period of 0 to 6 seconds.

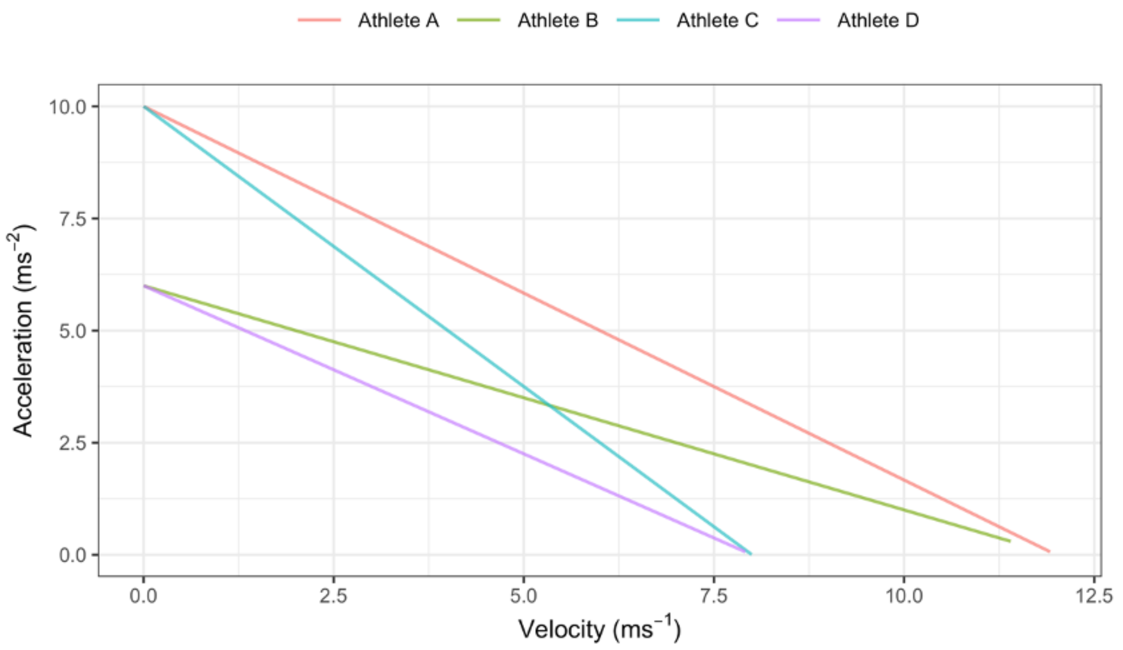


Figure 2. Acceleration-Velocity profile of four athletes with different MSS and MAC parameters.

Plotting acceleration against velocity (Figure 2), we will get *Acceleration-Velocity Profile*, which is linear, according to the mathematical model. If the athlete's body mass (kg) is known, as well as additional air resistance parameters (see Air resistance and the calculation of force and mechanical power section of this paper), *Force-Velocity Profile* can be estimated (see Force-Velocity profile section of this paper).

ESTIMATION USING {SHORTS} PACKAGE

Short sprints profiling is usually performed by: (1) measuring split times using timing gates (i.e., positioned at various distances, e.g., 5, 10, 20, 30, 40 m), or (2) getting a velocity trace using a radar gun. Estimation of MSS and TAU parameters from equation (1) is performed in **{shorts}** package using *non-linear least squares regression* implemented in the nlsLM() function from the **{minpack.lm}** package (Elzhov et al. 2022).

Estimating short sprint parameters using timing gates split times

For timing gates split times, distance is a predictor, and time is the outcome variable. Thus, equation (4) becomes:

$$t(d) = TAU \times W\left(-e^{\frac{-d}{MSS \times TAU}} - 1\right) + \frac{d}{MSS} + TAU \qquad (5)$$

In equation (5), *W* represents Lambert's *W* function (Goerg 2022). Researchers often incorrectly use the equation (4) (J. B. Morin 2017; Jean-Benoit Morin and Samozino 2019; Stenroth and Vartiainen 2020), in which the time is the predictor and distance is the outcome variable, instead of the statistically correct equation (5). This practice should be avoided because switching the predictor and outcome variables in the regression model might produce biased estimated parameters (Motulsky 2018, 341). These biases might not necessarily be significant with the short sprints profiling, but it is nevertheless a bad statistical practice.

Here is an example using a built-in dataset. We are going to filter out one athlete (e.g., John) with the help of the **{tidyverse}** (Wickham 2021) package and estimate MSS, TAU, and MAC parameters using the model_timing_gates() function:

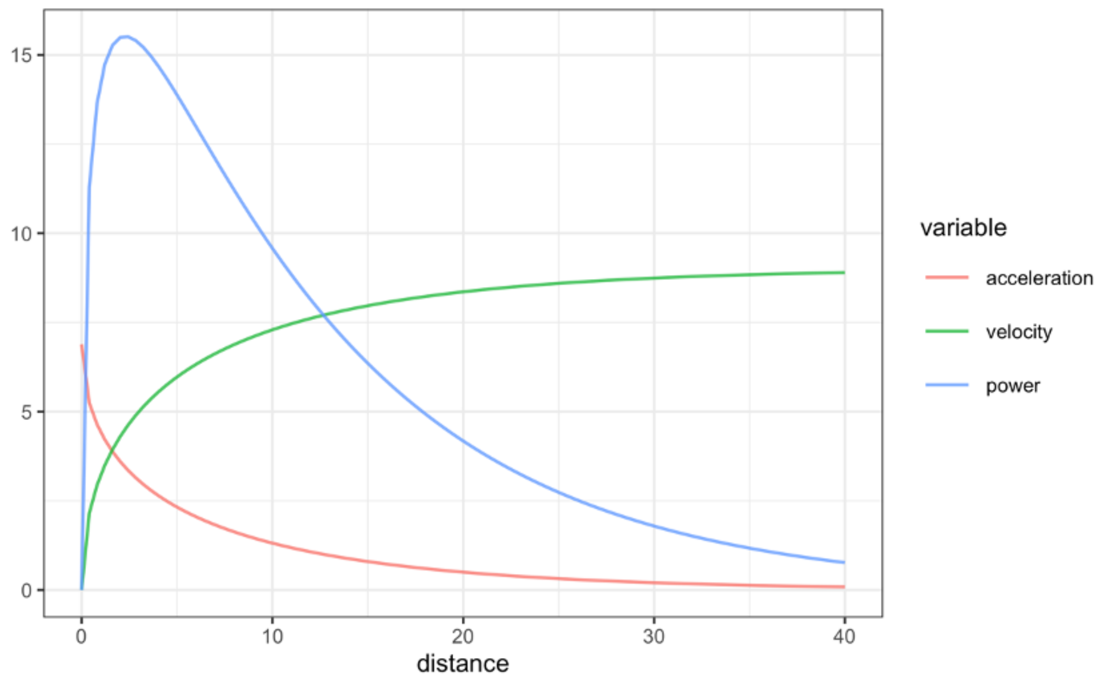
```
require(shorts)
require(tidyverse)

# Load built-in dataset
data(split_times)

# Filter timing gates splits for John
john_TG_data <- split_times %>%
  filter(athlete == "John")

john_TG_data
#> # A tibble: 6 × 4
#>   athlete bodyweight distance     time
#>   <chr>        <dbl>    <dbl> <I<dbl>>
#> 1 John            75        5     1.20
#> 2 John            75       10     1.97
#> 3 John            75       15     2.66
#> 4 John            75       20     3.31
#> 5 John            75       30     4.59
#> 6 John            75       40     5.85

# Estimate John's MSS, TAU, MAC, and PMAX
m1 <- model_timing_gates(
  distance = john_TG_data$distance,
  time = john_TG_data$time
)

m1
#> Estimated model parameters
#> --------------------------
#>    MSS    TAU    MAC   PMAX
#>  7.800  0.737 10.577 20.625
#>
#> Model fit estimators
#> --------------------
#>       RSE R_squared    minErr    maxErr maxAbsErr      RMSE
#>    0.0259    0.9998   -0.0405    0.0208    0.0405    0.0212
#>       MAE      MAPE
#>    0.0173    0.8584
```

Maximal relative power (PMAX) from the output is estimated using (MSS×MAC)/4, disregarding the air resistance.

Besides providing *residual standard error* (RSE), **{shorts}** functions provide additional model fit estimators. Additional information can be gained by exploring the returned object, particularly the object returned from the nlsLM() function (i.e., by using the S3 summary() method). To extract estimated model parameters, use S3 coef() method.

To create a simple plot of the model, use the S3 plot() method, which returns **ggplot2** (Wickham, Chang, et al. 2022) object:

```
plot(m1) + theme_bw(8)
```



Once we have estimated MSS and MAC, we can use predict_XXX() family of functions to predict various relationships (i.e., time at distance, acceleration at distance, velocity at time, etc.):

```
# Predict time at distance
predict_time_at_distance(
  distance = john_TG_data$distance,
  MSS = m1$parameters$MSS,
  MAC = m1$parameters$MAC
)
#> [1] 1.24 1.97 2.64 3.29 4.58 5.87
```

Estimating short sprint parameters using the radar gun

Estimating the short sprint profile using radar gun data takes time as a predictor and velocity as the outcome variable. Estimation using radar gun data is implemented in the **{shorts}** package using model_radar_gun() function.

Here is an example using built-in dataset:

```
# Load built-in dataset
data(radar_gun_data)

# Filter radar gun data for John
john_RG_data <- radar_gun_data %>%
  filter(athlete == "John")

head(john_RG_data)
#> # A tibble: 6 × 4
#>   athlete bodyweight  time velocity
#>   <chr>        <dbl> <dbl>    <dbl>
#> 1 John            75  0        0
#> 2 John            75  0.01     0.075
#> 3 John            75  0.02     0.149
#> 4 John            75  0.03     0.222
#> 5 John            75  0.04     0.291
#> 6 John            75  0.05     0.367

# Estimate John's MSS, TAU, MAC, and PMAX
m2 <- model_radar_gun(
  velocity = john_RG_data$velocity,
  time = john_RG_data$time
)

m2
#> Estimated model parameters
#> --------------------------
#>       MSS       TAU       MAC      PMAX        TC
#>  7.999887  1.068770  7.485133 14.970054 -0.000215
#>
#> Model fit estimators
#> --------------------
#>       RSE R_squared    minErr    maxErr maxAbsErr      RMSE
#>    0.0488    0.9994   -0.1715    0.1444    0.1715    0.0486
#>       MAE      MAPE
#>    0.0364       Inf
```

There is the additional parameter in the output, TC, that is estimated using the model_radar_gun() function. The utility of this parameter is explained in the Problems with time sync with the radar gun section of this paper.

Both timing gates and radar gun models allow *weighted* non-linear regression. Weighting in regression is utilized when the observations have unequal error variance (Gelman, Hill, and Vehtari 2020), which can happen due to instrumental error (e.g., a multiplicative error instead of additive error) or due to biological variability (e.g., higher split time variance on shorter distances versus longer distances). In this case, observations with higher error variance get lower weight when fitting the model (Gelman, Hill, and Vehtari 2020). According to Gelman, Hill, and Vehtari (2020), unequal variances are not typically a significant issue for the goal of estimating regression parameters, but they can become more important when making predictions about individual observations. Further exploring the topic of unequal error variances is beyond the scope of this paper. However, the **{shorts}** package provides a weighting feature for potential research of this topic in the future.

Weighted non-linear regression is performed by setting the weights parameter. For example, we can give more weight to shorter distances or faster velocities:

```
m2_weighted <- model_radar_gun(
  velocity = john_RG_data$velocity,
  time = john_RG_data$time,
  weights = 1 / (john_RG_data$velocity + 1)
)

m2_weighted
#> Estimated model parameters
#> --------------------------
#>        MSS        TAU        MAC       PMAX         TC
#>  7.9999558  1.0693094  7.4814227 14.9627626  0.0000522
#>
#> Model fit estimators
#> --------------------
#>       RSE R_squared    minErr    maxErr maxAbsErr      RMSE
#>    0.0170    0.9994   -0.1714    0.1447    0.1714    0.0486
#>       MAE      MAPE
#>    0.0364       Inf
```

Air resistance and the calculation of force and mechanical power

To estimate force production at distance or time (using predict_force_at_distance() and predict_force_at_time() functions), as well as power production (using predict_power_at_distance() and predict_power_at_time() functions), one needs to take into account the air resistance. Air resistance (measured in Newtons, *N*) is estimated using get_air_resistance() function, which takes velocity, body mass (kg), body height (*m*), barometric pressure (*Torr*), air temperature (*C°*), and wind velocity (ms^{-1}) as parameters (please refer to Arsac and Locatelli (2002), Samozino et al. (2016), and van Ingen Schenau, Jacobs, and de Koning (1991) for more information):

```
get_air_resistance(
  velocity = 5,
  bodymass = 80,
  bodyheight = 1.85,
  barometric_pressure = 780,
  air_temperature = 20,
  wind_velocity = 0.5
)
#> [1] 6.1
```

When estimating force and power, the air resistance parameters can be set using "...", which are forwarded to the get_air_resistance():

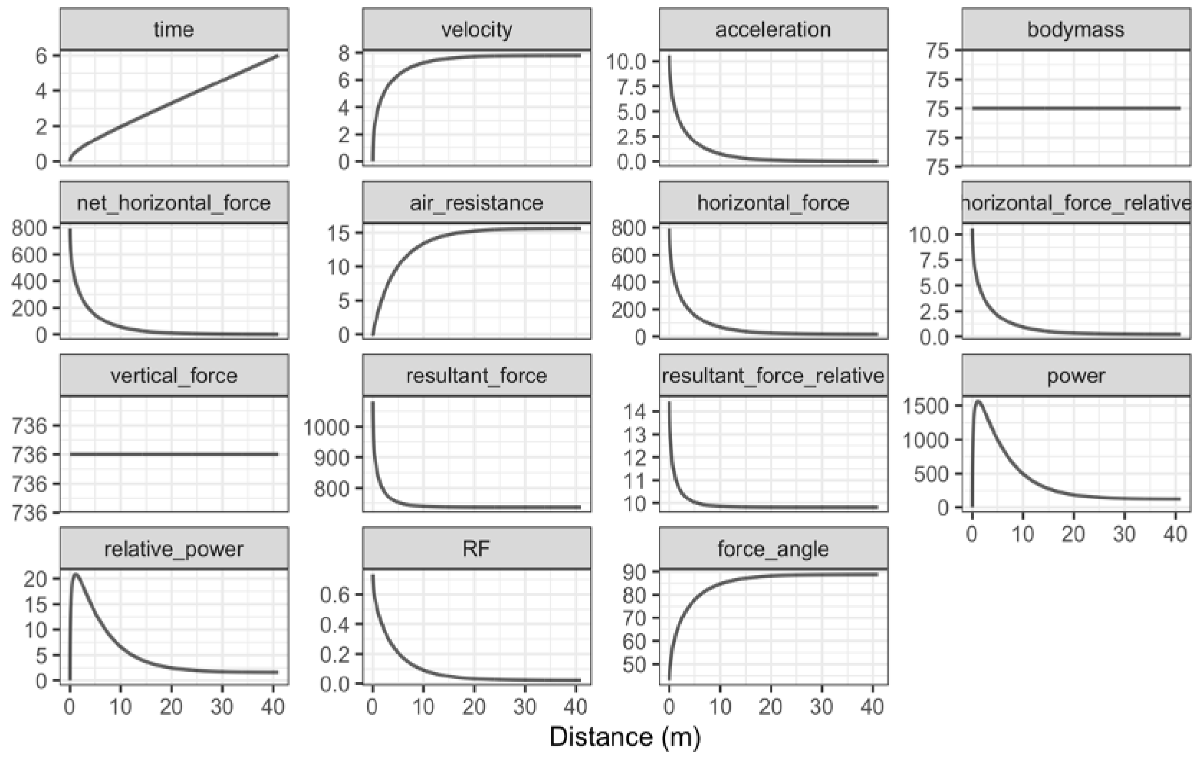
```
# To calculate horizontal force produced
predict_force_at_distance(
  distance = john_TG_data$distance,
  MSS = m1$parameters$MSS,
  MAC = m1$parameters$MAC,
  # Additional parameters forwarded to
get_air_resistance
  # Otherwise, defaults are used
  bodymass = john_TG_data$bodyweight,
  bodyheight = 1.85,
  barometric_pressure = 780,
  air_temperature = 20,
  wind_velocity = 0.5
)
#> [1] 157.3  68.4  36.8  24.4  17.1  15.9
```

The easiest way to get all kinematics and kinetics for short sprints is to use predict_kinematics() function:

```
df <- predict_kinematics(
  m1,
  max_time = 6,
  frequency = 100,
  # Additional parameters forwarded to
get_air_resistance
  # Otherwise, defaults are used
  bodymass = john_TG_data$bodyweight[1],
  bodyheight = 1.85,
  barometric_pressure = 780,
  air_temperature = 20,
  wind_velocity = 0.5
)
```

Plotting the model predictions can be done once we convert data from wide to long with the help of **ggplot2** (Wickham, Chang, et al. 2022), **dplyr** (Wickham, François, et al. 2022), and **tidyr** (Wickham and Girlich 2022) packages (loaded already using the **tidyverse** (Wickham 2021) package):

```
variable_names <- colnames(df)

df <- pivot_longer(data = df, cols = -2)
%>%
  mutate(name = factor(name, levels = var-
iable_names))

ggplot(df, aes(x = distance, y = value)) +
  theme_bw(8) +
  facet_wrap(~name, scales = "free_y") +
  geom_line(alpha = 0.7) +
  ylab(NULL) +
  xlab("Distance (m)")
```



These kinematic and kinetic variables are utilized in Force-Velocity profile estimation, which is covered later in this paper.

Utility Functions

Another valuable addition for sports scientists and coaches is the ability to determine the distances and times where 90% of maximum sprinting speed is reached, or where peak power is within 90% range. To identify these values, the **{shorts}** package comes with the find_XXX() family of functions:

```
# Finds distance where 90% of maximum sprinting speed is reached
find_velocity_critical_distance(
  MSS = m1$parameters$MSS,
  MAC = m1$parameters$MAC,
  percent = 0.9
)
#> [1] 8.07

# Finds maximal power and distance (this time using air resistance)
find_max_power_distance(
  MSS = m1$parameters$MSS,
  MAC = m1$parameters$MAC,
  # Additional parameters forwarded to get_air_resistance
  # Otherwise, defaults are used
  bodymass = 80,
  bodyheight = 1.85,
  barometric_pressure = 780,
  air_temperature = 20,
  wind_velocity = 0.5
)
#> $max_power
#> [1] 1664
#>
#> $distance
#> [1] 1.15

# Finds distance over 90% power range
find_power_critical_distance(
  MSS = m1$parameters$MSS,
  MAC = m1$parameters$MAC,
  # Additional parameters forwarded to get_air_resistance
  # Otherwise, defaults are used
  bodymass = 80,
  bodyheight = 1.85,
  barometric_pressure = 780,
  air_temperature = 20,
  wind_velocity = 0.5
)
#> $lower
#> [1] 0.452
#>
#> $upper
#> [1] 2.51
```

Force-Velocity Profile

To create *Force-Velocity Profile* (FVP) using single athlete estimated sprint model parameters (i.e., MSS and MAC), you can use the make_FV_profile() function. When estimating FVP, athlete body mass (kg) can be set using bodymass parameter, while the air resistance parameters can be set using "...", which are forwarded to the get_air_resistance() function. Details of the FVP method implemented in the **{shorts}** package and the interpretation from a sprint training perspective are covered elsewhere (Thomas A. Haugen, Breitschädel, and Samozino 2020; Jean-Benoit Morin and Samozino 2016; Jean-Benoit Morin et al. 2019; Samozino et al. 2016).

```
# To create Force-Velocity Profile
fvp <- make_FV_profile(
  MSS = m1$parameters$MSS,
  MAC = m1$parameters$MAC,
  bodymass = 80,
  # Additional parameters forwarded to get_air_resistance
  # Otherwise, defaults are used
  bodyheight = 1.85,
  barometric_pressure = 780,
  air_temperature = 20,
  wind_velocity = 0.5
)

fvp
#> Estimated Force-Velocity Profile
#> --------------------------------
#>      bodymass           F0        F0_rel          V0
#>      80.00000    840.73446      10.50918     7.94658
#>          Pmax Pmax_relative     FV_slope  RFmax_cutoff
#>    1670.23989     20.87800      -1.32248     0.30000
#>         RFmax          Drf        RSE_FV       RSE_Drf
#>       0.58380     -0.11766       0.99481     0.00855
```

To plot FVP kinematics and kinetics (which are the same as generated by the predict_kinematics() function), use the S3 plot() function. FVP estimated kinetics are default plotted against velocity (on the x-axis).

```
plot(fvp) + theme_bw(8)
```

To plot FVP estimated kinetics against time, use type = "time" parameter:

```
plot(fvp, "time") + theme_bw(8)
```

PROBLEMS WITH ESTIMATION

There is a challenge when collecting sprint data that could substantially impact modeled outcomes. To ensure accurate parameter outcomes, the initial force production must be synced with the start time (Thomas A. Haugen, Breitschädel, and Samozino 2020; Thomas A. Haugen, Breitschädel, and Seiler 2019). Below we describe this challenge when using radar guns or timing gates and suggest potential solutions within the **{shorts}** package.

Problems with time sync with the radar gun

Time synchronization is one error in the modeled estimation using a radar gun. In theory, synchronization is ideal when a sprint is initiated at t=0 s (i.e., $v(t=0)=0$ ms^{-1}). In practice, this is often not the case. This might happen when the measurement starts before the sprint starts. Although these data should be *trimmed*, the time variable might not be synchronized perfectly (i.e., sprint starts exactly at t=0 s) with the sprint initiation afterward.

The potential solution incorporated into the **{shorts}** package involves estimating the *time correction* (TC) parameter using equation (6). TC parameter serves as *intercept*, similar to linear regression, and allows the model to be fitted correctly in the scenario explained.

$$v(t) = MSS \times \left(1 - e^{-\frac{t+TC}{TAU}}\right) \qquad (6)$$

This model is incorporated in the model_radar_gun() function (see Estimating short sprint parameters using the radar gun), and TC is estimated in addition to MSS, MAC, and TAU parameters.

Problems at the start when using timing gates

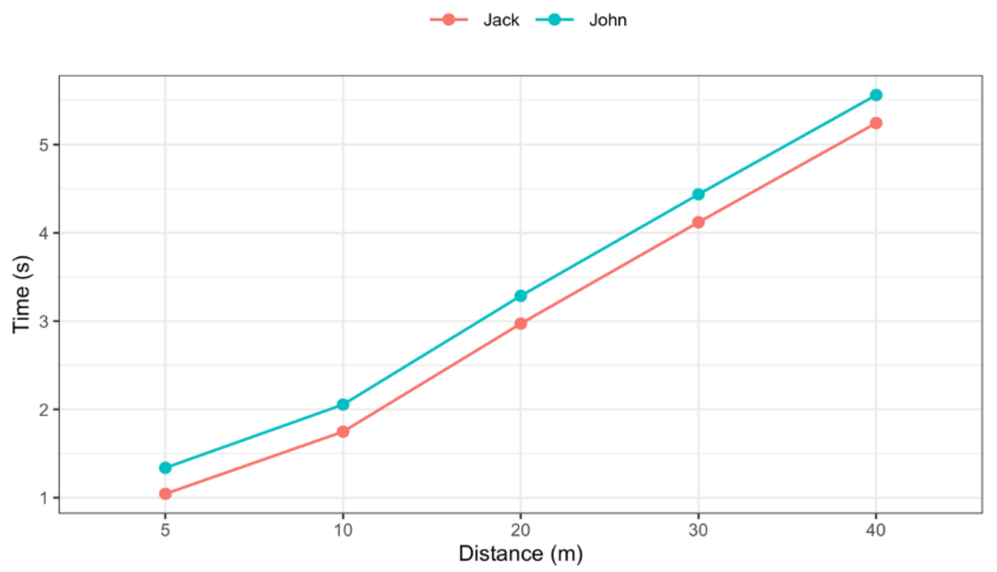
Let us imagine we have two twin brothers with the same short sprint characteristics: MSS equal to 9 ms^{-1} and MAC equal to 8 ms^{-2}. Let us call them John and Jack. They perform 40 m sprints using timing gates set at 5, 10, 20, 30, and 40 m. The initial timing gate at the start (i.e., d=0 m) activates the timing system (i.e., when they cross the beam).

John represents the *theoretical model*, in which we assume that the initial force production and the timing initiation are perfectly synchronized. On the other hand, Jack represents a *practical model* and decides to move slightly behind the initial timing gate (i.e., for 0.5 m) and use body rocking to initiate the sprint start. In other words, Jack uses a *flying start*, a common scenario when testing field sports athletes. Flying start distance is often recommended to avoid premature timing system triggering by lifting knees or swinging arms (Altmann et al. 2018, 2015, 2017; Thomas A. Haugen, Breitschädel, and Samozino 2020; T. Haugen and Buchheit 2016).

Data for this scenario is generated using create_timing_gates_splits() function:

```
split_times <- tibble(
  distance = c(5, 10, 20, 30, 40),
  john_time = create_timing_gates_splits(
    MSS = 9,
    MAC = 8,
    gates = distance
  ),
  jack_time = create_timing_gates_splits(
    MSS = 9,
    MAC = 8,
    gates = distance,
    FD = 0.5
  )
)
```

Here is a graphical representation of the sprint splits (please refer to the Supplemental Material for the R code):



We can see the differences in estimated MSS and MAC parameters using the following code. For better table output, we are going to use the kable() function from the **{knitr}** package (Xie 2022b).

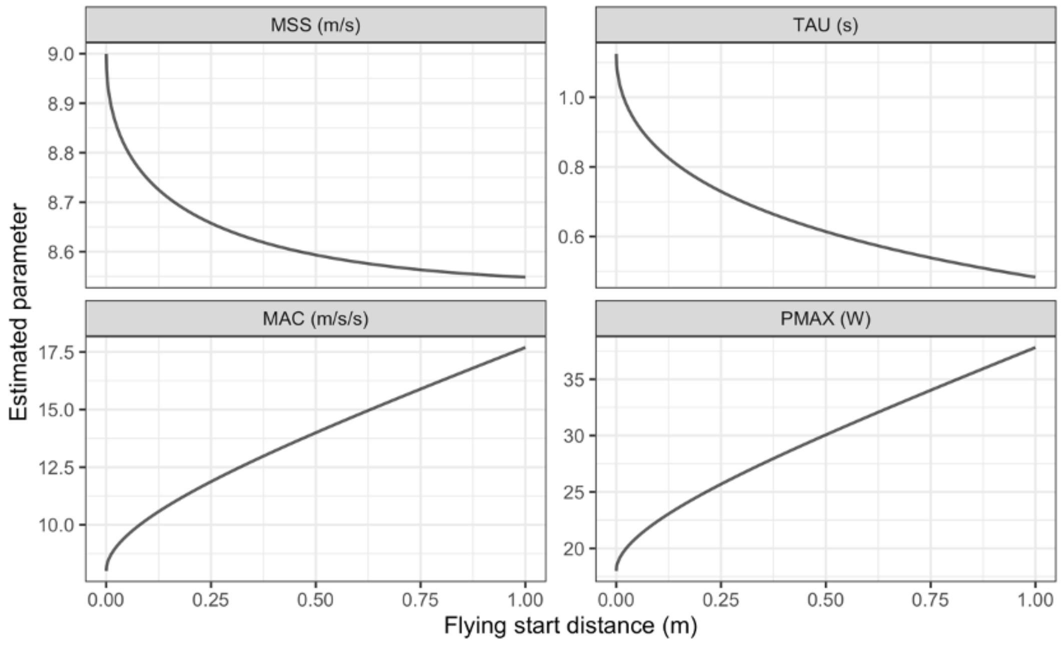
```
john_profile <- model_timing_gates(
  distance = split_times$distance,
  time = split_times$john_time
)

jack_profile <- model_timing_gates(
  distance = split_times$distance,
  time = split_times$jack_time
)

sprint_parameters <- rbind(
  coef(john_profile),
  coef(jack_profile)
)

rownames(sprint_parameters) <- c("John", "Jack")

kable(sprint_parameters, digits = 2, booktabs = TRUE)
```

	MSS	TAU	MAC	PMAX
John	9.00	1.12	8	18.0
Jack	8.59	0.61	14	30.1

As can be seen from the results, a flying start yields biased estimates, particularly for the TAU, MAC, and PMAX.

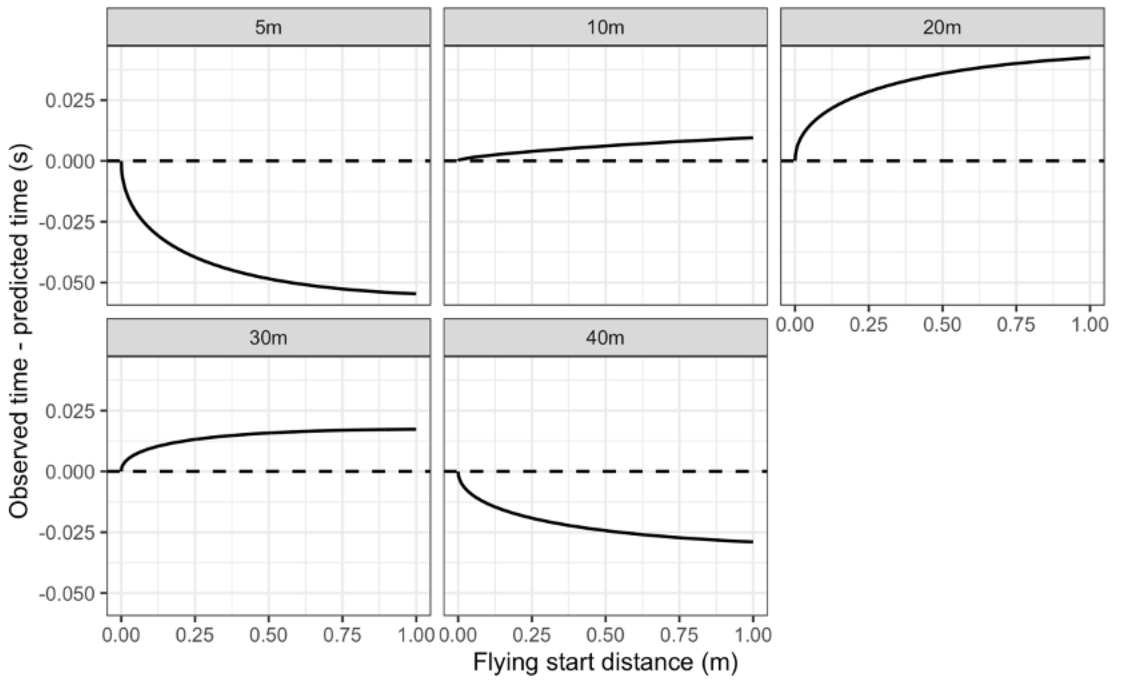
Simple Simulation

To explore this further, we have run a simple simulation by increasing Jack's flying start distance from 0 to 1 m and depicting the estimated MSS, TAU, MAC, and PMAX parameters (please refer to the Supplemental Material for the R code).

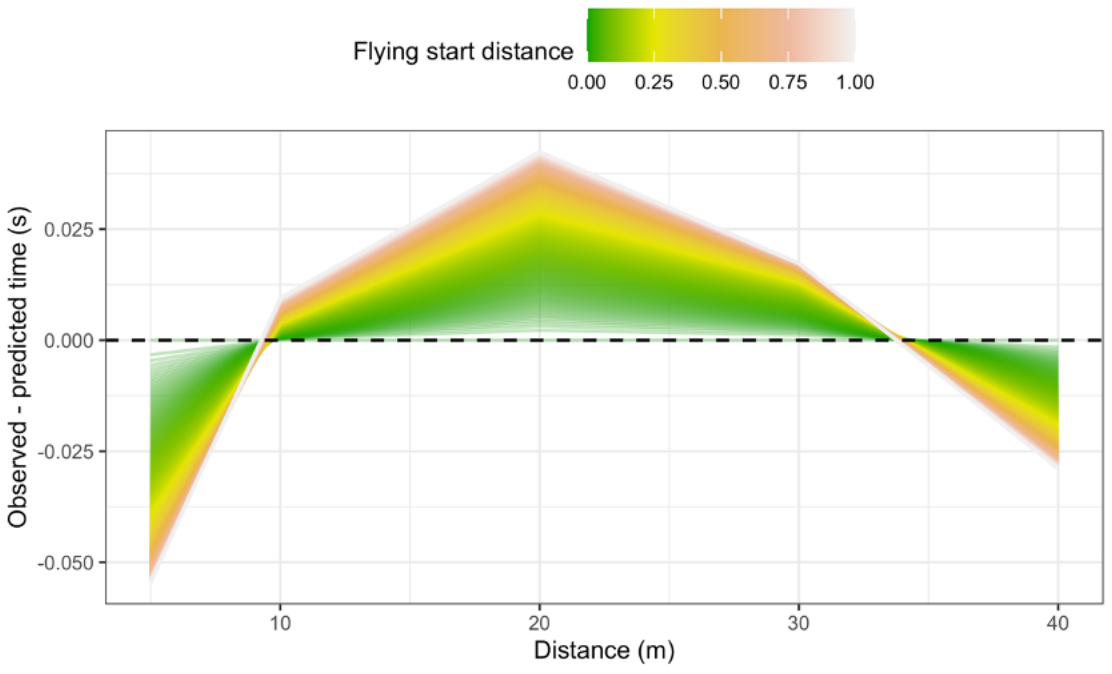


As can be seen from the figure, MSS and TAU are underestimated, while MAC and PMAX are overestimated as the flying start distance increases.

Model residuals are also affected by the flying start distance. The shape of the distribution of the residuals depends on the number and splits utilized (e.g., 10, 20, 30, 40 m versus 5, 15, 30 m). However, here we can see the effect of the flying start distance on the model residuals per split distance utilized in our simple simulation:



Another way to visualize the effect of the flying start distance on split distance residuals can be found in the following figure:



Any flying start with a difference between initial force production and start time can result in biased parameters and predictions. Since maximal sprint speed is difficult to improve, the effects of start inconsistencies can mask the effects of the training intervention. It is thus crucial to standardize the start when testing and implementing the following techniques when using the **{shorts}** package.

How to overcome missing the initial force production when using timing gates?

A potential solution is to use a correction factor - the recommendation in the literature is +0.5 s (Thomas A. Haugen, Breitschädel, and Seiler 2020, 2019). Interestingly, the average difference between timing gates and a block start for 40 m sprint time was 0.27 s (Thomas A. Haugen, Tønnessen, and Seiler 2012). So, while a timing correction factor is warranted to avoid subsequent errors in estimates of kinetic variables (e.g., overestimate power), a correction factor that is too large will have the opposite effect (e.g., underestimate power).

Rather than providing *apriori* time correction from the literature, the **{shorts}** package provides an estimation of this parameter from the data provided, together with MSS and MAC. The same method is suggested by Stenroth, Vartiainen, and Karjalainen (2020), named the *time shift method*, and the estimated parameter is named the *time shift parameter*. We have named this parameter *time correction* (TC) to agree with the parameter introduced in the Problems with time sync with the radar gun section of this paper and the available literature. The model that implements the TC parameter is termed the *estimated TC model*.

When implementing time correction, equation (5) becomes:

$$t(d) = TAU \times W\left(-e^{\frac{-d}{MSS \times TAU}} - 1\right) + \frac{d}{MSS} + TAU - TC \qquad (7)$$

In **{shorts}** package, TC model is implemented in the model_timing_gates_TC() function. Here is how we can estimate Jack parameters using fixed time corrections (e.g., +0.3 and +0.5 *s*) or using estimated TC model:

```
jack_profile_fixed_time_short <- model_timing_gates(
  distance = split_times$distance,
  time = split_times$jack_time + 0.3
)

jack_profile_fixed_time_long <- model_timing_gates(
  distance = split_times$distance,
  time = split_times$jack_time + 0.5
)

jack_profile_time_estimated <- model_timing_gates_TC(
  distance = split_times$distance,
  time = split_times$jack_time
)

jack_parameters <- full_join(
  rbind(
    data.frame(athlete = "John", t(coef(john_profile))),
    data.frame(athlete = "Jack - No corrections", t(coef(jack_profile))),
    data.frame(athlete = "Jack - Fixed TC (+0.3s)", t(coef(jack_profile_fixed_time_
short))),
    data.frame(athlete = "Jack - Fixed TC (+0.5s)", t(coef(jack_profile_fixed_time_
long)))
  ),
  data.frame(athlete = "Jack - Estimated TC", t(coef(jack_profile_time_estimated)))
)

kable(jack_parameters, digits = 2, booktabs = TRUE)
```

athlete	MSS	TAU	MAC	PMAX	TC
John	9.00	1.12	8.00	18.0	
Jack - No corrections	8.59	0.61	14.00	30.1	
Jack - Fixed TC (+0.3s)	9.04	1.13	8.02	18.1	
Jack - Fixed TC (+0.5s)	9.61	1.61	5.98	14.4	
Jack - Estimated TC	8.97	1.05	8.51	19.1	0.26

In Jack's case, both +0.3*s* fixed time correction and estimated TC model yield parameters closer to John's (i.e., *true* parameters). We have used these models in a retrospective pilot study (Vescovi and Jovanović 2021), demonstrating statistically significant differences in estimated FVP parameters.

Instead of using time correction as a simple intercept in equation (7), we can estimate the *flying distance* (FD) using the following equation:

$$\begin{aligned} t(d) &= \left(TAU \times W\left(-e^{\frac{-d+FD}{MSS \times TAU}} - 1\right) + \frac{d+FD}{MSS} + TAU\right) \\ &\quad - \left(TAU \times W\left(-e^{\frac{FD}{MSS \times TAU}} - 1\right) + \frac{FD}{MSS} + TAU\right) \end{aligned} \quad (8)$$

We have named this model *estimated FD model*, and in the **{shorts}** package it is implemented in the model_timing_gates_FD() function. Here is how we can estimate Jack parameters using estimated FD model:

```
jack_profile_FD <- model_timing_gates_FD(
  distance = split_times$distance,
  time = split_times$jack_time
)

jack_parameters <- full_join(
  jack_parameters,
  data.frame(athlete = "Jack - Estimated FD", t(coef(jack_profile_FD)))
)

kable(jack_parameters, digits = 2, booktabs = TRUE)
```

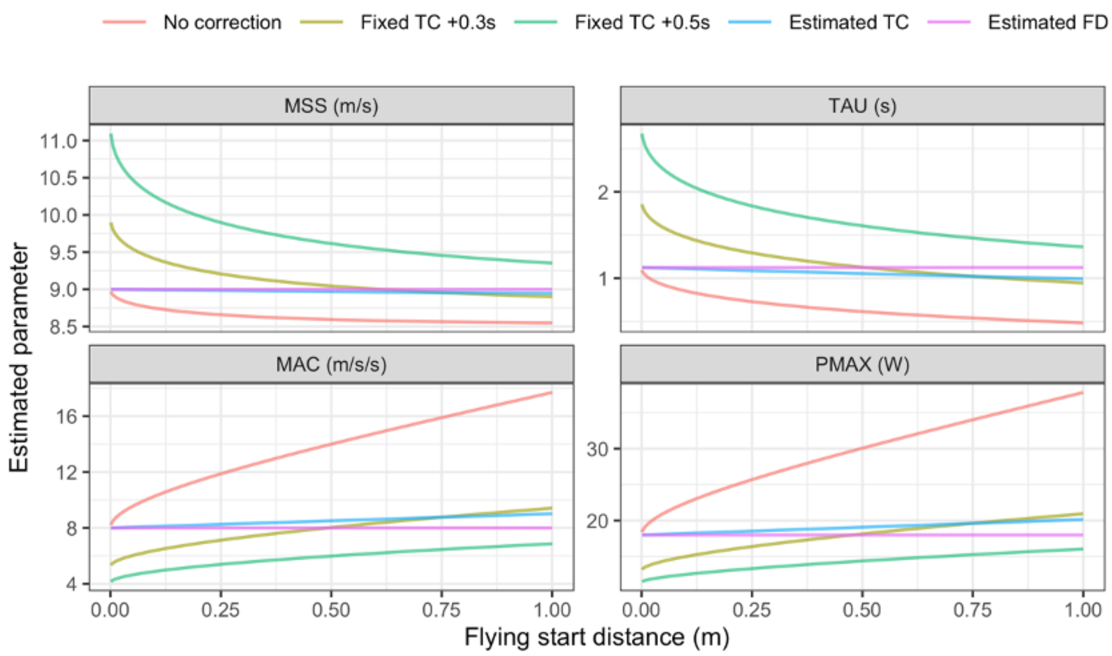
athlete	MSS	TAU	MAC	PMAX	TC	FD
John	9.00	1.12	8.00	18.0		
Jack - No corrections	8.59	0.61	14.00	30.1		
Jack - Fixed TC (+0.3s)	9.04	1.13	8.02	18.1		
Jack - Fixed TC (+0.5s)	9.61	1.61	5.98	14.4		
Jack - Estimated TC	8.97	1.05	8.51	19.1	0.26	
Jack - Estimated FD	9.00	1.12	8.00	18.0		0.5

As can be seen from the results, the estimated FD model correctly estimated Jack's sprint parameters. There are a few issues with this model definition. Besides being novel and still not validated with actual data, the estimated FD model has three parameters to estimate, which implies that at least four sprint splits are needed. This imposes practical limitations since acquiring five timing gates (one for a start and four for splits) might be practically troublesome (which is also the case with the estimated TC model). In addition, the estimated FD model can be ill-defined in scenarios involving reaction time and no actual flying sprint involved. This is often the case when a gunshot is used to initiate the timing system.

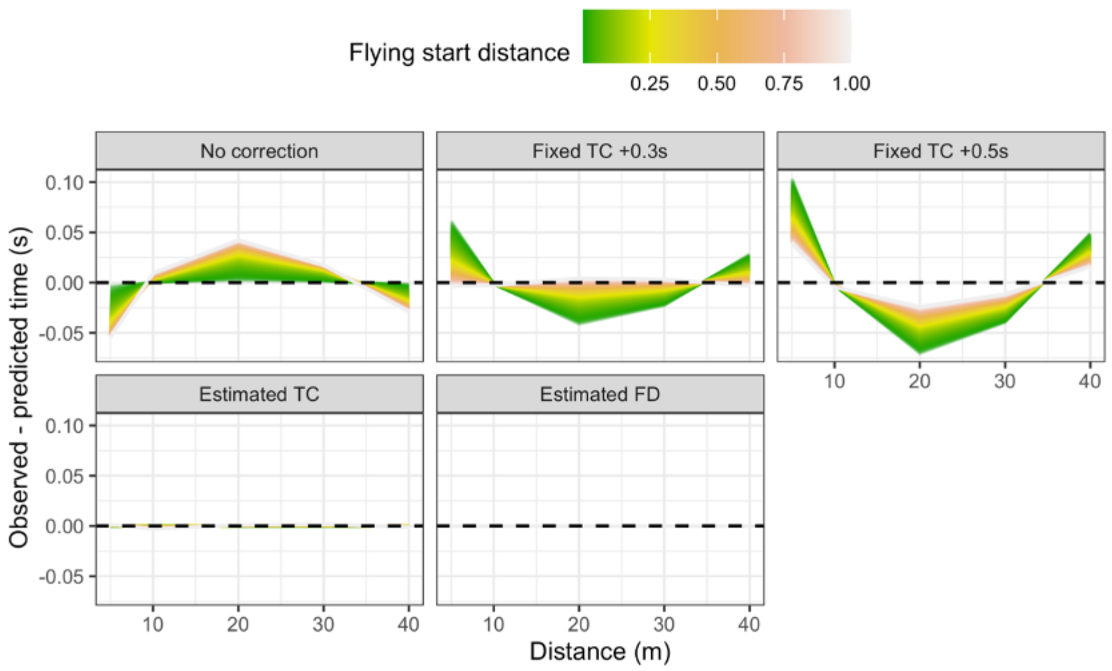
Simple simulation using proposed models

In this section, we examine how these models (no correction, fixed time correction, estimated TC, and estimated FD) perform using simulated data with varying flying start distances (please refer to the Supplemental Material for the R code), assuming *true* MSS equal to 9 ms^{-1}, and MAC equal to 8 ms^{-2} (i.e., sprint characteristics of Jack and John). Timing gates are set at 5, 10, 20, 30, and 40 m.

As can be seen from the following figure, the estimated TC model estimates sprint parameters with increasing bias as the flying distance increases. In contrast, the estimated FD model estimates sprint parameters perfectly. If correct TC is not applied, fixed time correction models have a much larger bias. This value depends on sprint characteristics, the flying distance, and timing gate splits. It is thus impossible to provide a single fixed time correction value that will serve its corrective purpose across different scenarios and athletes measured.



The following figure depicts residuals across split distances for each simulated flying start distance. As can be seen, estimated TC and FD models perform much better than the no correction and fixed time correction models.



The results of this simple simulation demonstrate that the estimated TC and FD models represent sound improvements in parameter estimation and model fit compared to the no corrections model and fixed TC models when attempting to overcome the flying start issues. A more detailed simulation study is ongoing, and the results will be reported in another paper.

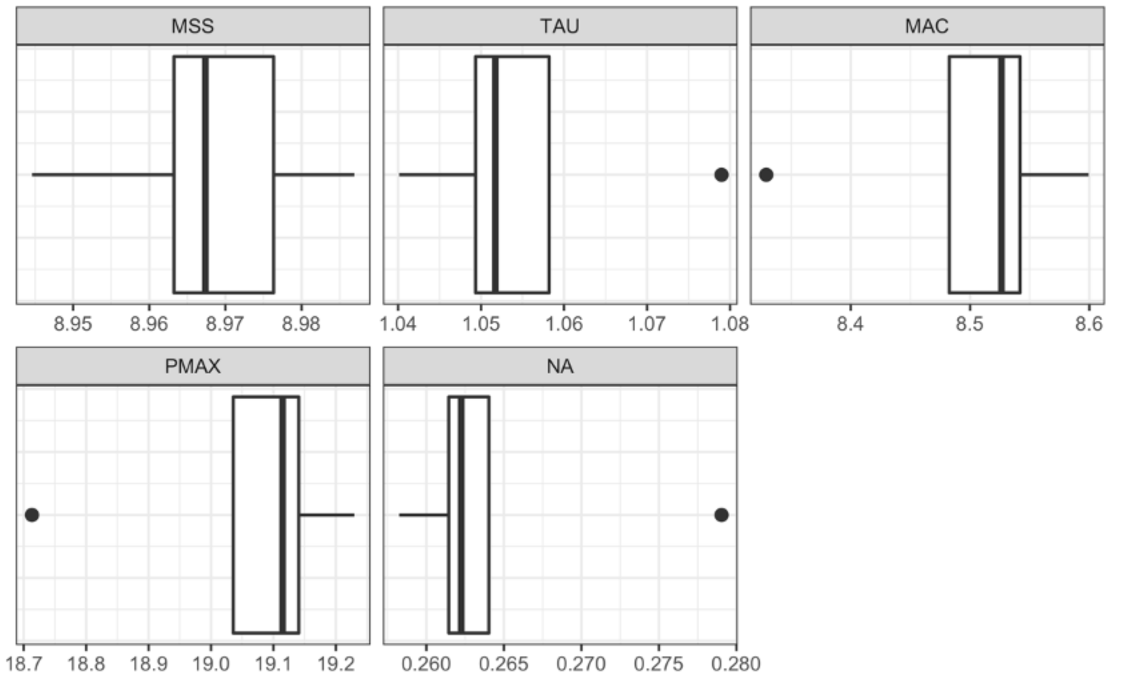
CROSS-VALIDATION

To estimate parameter stability, model over-fitting, and performance on unseen data, **{shorts}** model functions come with implemented *leave-one-out cross-validation* (LOOCV) for the timing gates models and *n-fold cross-validation* (CV) for the radar gun models (James et al. 2017; Jovanović 2020; Kuhn and Johnson 2018). LOOCV involves a simple yet powerful procedure of removing each observation, rebuilding the model, and making predictions for that removed observation. This process is repeated for each observation in the model dataset. LOOCV allows one to check estimated parameters' stability and model performance on unseen data.
Let's perform LOOCV using Jack's data and the estimated TC model:

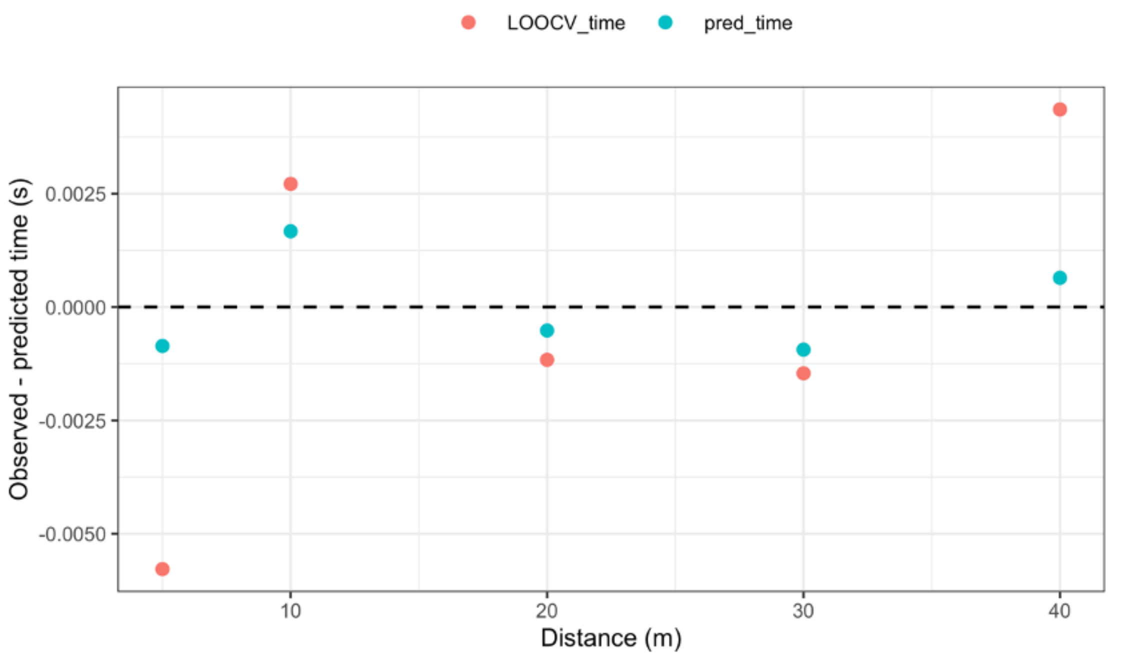
```
jack_LOOCV <- model_timing_gates_TC(
  distance = split_times$distance,
  time = split_times$jack_time,
  LOOCV = TRUE
)
```

The model print output provides training dataset estimates and model performance, as well as LOOCV estimates and model performance.

Next, we plot estimated parameters across LOOCV folds (please refer to the Supplemental Material for the R code):



Here is the plot of the training and LOOCV residuals:



As expected, the model has more issues predicting unseen split times for short or long distances. Please note that since LOOCV removes one observation, if the model estimates three parameters, then at least five observations are needed since we need to ensure the model can be estimated once a single observation is removed.

Since there are much more observations in the radar gun data, n-fold CV is implemented instead of LOOCV. CV procedure randomly splits the dataset into n-folds, leave one fold-out for prediction, and makes a model using the remaining folds. This is then repeated for all the folds.

```
# Estimate John's MSS, TAU, MAC, and PMAX
m2_CV <- model_radar_gun(
  velocity = john_RG_data$velocity,
  time = john_RG_data$time,
  # Eneter number of folds
  CV = 10
)

# Estimated parameters for each fold
m2_CV$CV$parameters
#> # A tibble: 10 × 5
#>      MSS   TAU   MAC  PMAX         TC
#>    <dbl> <dbl> <dbl> <dbl>      <dbl>
#>  1  8.00  1.07  7.49  15.0 -0.000321
#>  2  8.00  1.07  7.49  15.0 -0.000348
#>  3  8.00  1.07  7.49  15.0 -0.000363
#>  4  8.00  1.07  7.48  15.0 -0.0000164
#>  5  8.00  1.07  7.48  15.0 -0.0000593
#>  6  8.00  1.07  7.49  15.0 -0.000281
#>  7  8.00  1.07  7.48  15.0  0.0000791
#>  8  8.00  1.07  7.49  15.0 -0.000298
#>  9  8.00  1.07  7.49  15.0 -0.000417
#> 10  8.00  1.07  7.48  15.0 -0.000108
```

EXAMPLE ANALYSIS

Let us use real-world data to demonstrate the functionalities of the **{shorts}** package. We will use the dataset from Usain Bolt's performance at the IAAF World Championship finals in London, 2017.

Since reaction time enters the splits, we want to see how that will affect the model estimates, mainly if the estimated time correction model will pick up reaction time.

For the sake of this analysis, only 10 m splits over 60 m race distance are used.

```
bolt_reaction_time <- 0.183

bolt_distance <- c(10, 20, 30, 40, 50, 60)
bolt_time <- c(1.963, 2.983, 3.883, 4.763, 5.643, 6.493)

# No corrections model
bolt_m1 <- model_timing_gates(
  distance = bolt_distance,
  time = bolt_time
)

# Model with reaction time as fixed time correction
bolt_m2 <- model_timing_gates(
  distance = bolt_distance,
  time = bolt_time - bolt_reaction_time
)

# Model with estimated time correction
bolt_m3 <- model_timing_gates_TC(
  distance = bolt_distance,
  time = bolt_time
)

# Model with flying distance correction
# THIS CANNOT BE ESTIMATED SINCE IT IS ILL-DEFINED MODEL
# bolt_m4 <- model_timing_gates_FD(
#   distance = bolt_distance,
#   time = bolt_time
# )


bolt_model <- full_join(
  rbind(
    data.frame(model = "No correction", t(coef(bolt_m1))),
    data.frame(model = "No correction - RT", t(coef(bolt_m2)))
  ),
  data.frame(model = "Estimated TC", t(coef(bolt_m3)))
)

kable(bolt_model, digits = 2, booktabs = TRUE)
```

model	MSS	TAU	MAC	PMAX	TC
No correction	12.1	1.56	7.77	23.6	
No correction - RT	11.7	1.21	9.74	28.6	
Estimated TC	11.7	1.20	9.76	28.6	-0.18

Here is the model estimate of the time and distance it takes for Bolt to reach 99% of MSS. These estimated times and distances represent *true* time and distance when the sprint is initiated at *t*=0 *s* and *d*=0 m.

```
bolt_model <- bolt_model %>%
  group_by(model) %>%
  mutate(
    `99% MSS (m)` = find_velocity_criti-
cal_distance(
      MSS = MSS, MAC = MAC,
      percent = 0.99
    ),
    `99% MSS (s)` = find_velocity_criti-
cal_time(
      MSS = MSS, MAC = MAC,
      percent = 0.99
    )
  )

kable(bolt_model[c(1, 7, 8)], digits = 2,
booktabs = TRUE)
```

model	99% MSS (m)	99% MSS (s)
No correction	68.7	7.20
No correction - RT	51.1	5.55
Estimated TC	51.0	5.54

CONCLUSION

Which model should be used? Although providing novel theoretical models in this paper, we acknowledge the need for validating them in practice against gold-standard methods, assessing their agreement, and their power in detecting and adjusting for timing inconsistencies. A more thorough theoretical simulation study is currently in development, intending to explore the behavior of these models under different scenarios.

We hope that the **{shorts}** package will help fellow sports scientists and coaches explore short sprint profiles and help in driving research, particularly in devising measuring protocols that are sensitive enough to capture training intervention changes but also robust enough to take into account potential sprint initiation and timing inconsistencies.

SUPPLEMENTAL MATERIAL

The *R Markdown* (Xie 2022a; Allaire et al. 2022; Xie, Allaire, and Grolemund 2018; Xie, Dervieux, and Riederer 2020) source code for the paper can be found on the GitHub repository: https://github.com/mladenjovanovic/shorts-paper.

REFERENCES

1. Allaire, JJ, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. 2022. Rmarkdown: Dynamic Documents for r. https://CRAN.R-project.org/package=rmarkdown.
2. Altmann, Stefan, Marian Hoffmann, Gunther Kurz, Rainer Neumann, Alexander Woll, and Sascha Haertel. 2015. "Different Starting Distances Affect 5-m Sprint Times." Journal of Strength and Conditioning Research 29 (8): 2361–66. https://doi.org/10.1519/JSC.0000000000000865.
3. Altmann, Stefan, Max Spielmann, Florian Azad Engel, Rainer Neumann, Steffen Ringhof, Doris Oriwol, and Sascha Haertel. 2017. "Validity of Single-Beam Timing Lights at Different Heights." Journal of Strength and Conditioning Research 31 (7): 1994–99. https://doi.org/10.1519/JSC.0000000000001889.
4. Altmann, Stefan, Max Spielmann, Florian Azad Engel, Steffen Ringhof, Doris Oriwol, Sascha Härtel, and Rainer Neumann. 2018. "Accuracy of Single Beam Timing Lights for Determining Velocities in a Flying 20-m Sprint: Does Timing Light Height Matter?" Journal of Human Sport and Exercise 13 (3). https://doi.org/10.14198/jhse.2018.133.10.
5. Arsac, Laurent M., and Elio Locatelli. 2002. "Modeling the Energetics of 100-m Running by Using Speed Curves of World Champions." Journal of Applied Physiology 92 (5): 1781–88. https://doi.org/10.1152/japplphysiol.00754.2001.
6. Brown, Todd D., Jason D. Vescovi, and Jaci L. Vanheest. 2004. "Assessment of Linear Sprinting Performance: A Theoretical Paradigm." Journal

of Sports Science & Medicine 3 (4): 203–10.
7. Buchheit, Martin, Pierre Samozino, Jonathan Alexander Glynn, Ben Simpson Michael, Hani Al Haddad, Alberto Mendez-Villanueva, and Jean Benoit Morin. 2014. “Mechanical Determinants of Acceleration and Maximal Sprinting Speed in Highly Trained Young Soccer Players.” Journal of Sports Sciences 32 (20): 1906–13. https://doi.org/10.1080/02640414.2014.965191.
8. Clark, Kenneth P., Randall H. Rieger, Richard F. Bruno, and David J. Stearne. 2017. “The NFL Combine 40-Yard Dash: How Important Is Maximum Velocity?” Journal of Strength and Conditioning Research, June, 1. https://doi.org/10.1519/JSC.0000000000002081.
9. Edwards, Toby, Benjamin Piggott, Harry G. Banyard, G. Gregory Haff, and Christopher Joyce. 2020. “Sprint Acceleration Characteristics Across the Australian Football Participation Pathway.” Sports Biomechanics, August, 1–13. https://doi.org/10.1080/14763141.2020.1790641.
10. Elzhov, Timur V., Katharine M. Mullen, Andrej-Nikolai Spiess, and Ben Bolker. 2022. Minpack.lm: R Interface to the Levenberg-Marquardt Nonlinear Least-Squares Algorithm Found in MINPACK, Plus Support for Bounds. https://CRAN.R-project.org/package=minpack.lm.
11. Furusawa, K., Archibald Vivian Hill, and J. L. Parkinson. 1927. “The Dynamics of "Sprint" Running.” Proceedings of the Royal Society of London. Series B, Containing Papers of a Biological Character 102 (713): 29–42. https://doi.org/10.1098/rspb.1927.0035.
12. Gelman, Andrew, Jennifer Hill, and Aki Vehtari. 2020. Regression and Other Stories. S.l.: Cambridge University Press.
13. Goerg, Georg M. 2022. LambertW: Probabilistic Models to Analyze and Gaussianize Heavy-Tailed, Skewed Data. https://CRAN.R-project.org/package=LambertW.
14. Haugen, Thomas A., Felix Breitschädel, and Pierre Samozino. 2020. “Power-Force-Velocity Profiling of Sprinting Athletes: Methodological and Practical Considerations When Using Timing Gates.” Journal of Strength and Conditioning Research 34 (6): 1769–73. https://doi.org/10.1519/JSC.0000000000002890.
15. Haugen, Thomas A., Felix Breitschädel, and Stephen Seiler. 2019. “Sprint Mechanical Variables in Elite Athletes: Are Force-Velocity Profiles Sport Specific or Individual?” Edited by Leonardo A. Peyré-Tartaruga. PLOS ONE 14 (7): e0215551. https://doi.org/10.1371/journal.pone.0215551.
16. ———. 2020. “Sprint Mechanical Properties in Soccer Players According to Playing Standard, Position, Age and Sex.” Journal of Sports Sciences 38 (9): 1070–76. https://doi.org/10.1080/02640414.2020.1741955.
17. Haugen, Thomas A, Espen Tønnessen, and Stephen K Seiler. 2012. “The Difference Is in the Start: Impact of Timing and Start Procedure on Sprint Running Performance:” Journal of Strength and Conditioning Research 26 (2): 473–79. https://doi.org/10.1519/JSC.0b013e318226030b.
18. Haugen, Thomas, and Martin Buchheit. 2016. “Sprint Running Performance Monitoring: Methodological and Practical Considerations.” Sports Medicine 46 (5): 641–56. https://doi.org/10.1007/s40279-015-0446-0.
19. James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2017. An Introduction to Statistical Learning: With Applications in R. 1st ed. 2013, Corr. 7th printing 2017 edition. New York: Springer.
20. Jiménez-Reyes, Pedro, Pierre Samozino, Amador García-Ramos, Víctor Cuadrado-Peñafiel, Matt Brughelli, and Jean-Benoît Morin. 2018. “Relationship Between Vertical and Horizontal Force-Velocity-Power Profiles in Various Sports and Levels of Practice.” PeerJ 6 (November): e5937. https://doi.org/10.7717/peerj.5937.
21. Jovanović, Mladen. 2020. Bmbstats: Bootstrap Magnitude-Based Statistics for Sports Scientists. Mladen Jovanović.
22. ———. 2022. shorts: Short Sprints. https://CRAN.R-project.org/package=shorts.
23. Kuhn, Max, and Kjell Johnson. 2018. Applied Predictive Modeling. 1st ed. 2013, Corr. 2nd printing 2016 edition. New York: Springer.
24. Mangine, Gerald T., Jay R. Hoffman, Adam M. Gonzalez, Adam J. Wells, Jeremy R. Townsend, Adam R. Jajtner, William P. McCormack, et al. 2014. “Speed, Force, and Power Values Produced From Nonmotorized Treadmill Test Are Related to Sprinting Performance:” Journal of Strength and Conditioning Research 28 (7): 1812–19. https://doi.org/10.1519/JSC.0000000000000316.
25. Marcote-Pequeño, Ramón, Amador García-Ramos, Víctor Cuadrado-Peñafiel, Jorge M. González-Hernández, Miguel Ángel Gómez, and Pedro Jiménez-Reyes. 2019. “Association Between the Force and Performance Variables Obtained in Jumping and Sprinting in Elite Female Soccer Players.” International Journal of Sports Physiology and Performance 14 (2): 209–15. https://doi.org/10.1123/ijspp.2018-0233.
26. Morin, J. B. 2017. “A Spreadsheet for Sprint Acceleration Force-Velocity-Power Profiling.” JB

Morin, PhD - Sport Science. December 13, 2017. https://jbmorin.net/2017/12/13/a-spreadsheet-for-sprint-acceleration-force-velocity-power-profiling/.

27. Morin, Jean-Benoit, and Pierre Samozino. 2016. “Interpreting Power-Force-Velocity Profiles for Individualized and Specific Training.” International Journal of Sports Physiology and Performance 11 (2): 267–72. https://doi.org/10.1123/ijspp.2015-0638.
28. ———. 2019. “Spreadsheet for Sprint Acceleration Force-Velocity-Power Profiling.”
29. Morin, Jean-Benoit, Pierre Samozino, Munenori Murata, Matt R Cross, and Ryu Nagahara. 2019. “A Simple Method for Computing Sprint Acceleration Kinetics from Running Velocity Data: Replication Study with Improved Design.” Journal of Biomechanics 94 (September): 82–87. https://doi.org/10.1016/j.jbiomech.2019.07.020.
30. Motulsky, Harvey. 2018. Intuitive Biostatistics: A Nonmathematical Guide to Statistical Thinking. Fourth edition. New York: Oxford University Press.
31. R Core Team. 2022. R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.
32. Samozino, P., G. Rabita, S. Dorel, J. Slawinski, N. Peyrot, E. Saez de Villarreal, and J.-B. Morin. 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 (6): 648–58. https://doi.org/10.1111/sms.12490.
33. Stenroth, Lauri, and Paavo Vartiainen. 2020. “Spreadsheet for Sprint Acceleration Force-Velocity-Power Profiling with Optimization to Correct Start Time.” https://doi.org/10.13140/RG.2.2.12841.83045.
34. Stenroth, Lauri, Paavo Vartiainen, and Pasi A. Karjalainen. 2020. “Force-Velocity Profiling in Ice Hockey Skating: Reliability and Validity of a Simple, Low-Cost Field Method.” Sports Biomechanics, June, 1–16. https://doi.org/10.1080/14763141.2020.1770321.
35. van Ingen Schenau, Gerrit Jan, Ron Jacobs, and Jos J. de Koning. 1991. “Can Cycle Power Predict Sprint Running Performance?” European Journal of Applied Physiology and Occupational Physiology 63 (3-4): 255–60. https://doi.org/10.1007/BF00233857.
36. Vescovi, Jason D., and Mladen Jovanović. 2021. “Sprint Mechanical Characteristics of Female Soccer Players: A Retrospective Pilot Study to Examine a Novel Approach for Correction of Timing Gate Starts.” Frontiers in Sports and Active Living 3 (May): 629694. https://doi.org/10.3389/fspor.2021.629694.
37. Ward-Smith, A. J. 2001. “Energy Conversion Strategies During 100 m Sprinting.” Journal of Sports Sciences 19 (9): 701–10. https://doi.org/10.1080/02640410152475838.
38. Wickham, Hadley. 2021. Tidyverse: Easily Install and Load the Tidyverse. https://CRAN.R-project.org/package=tidyverse.
39. Wickham, Hadley, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo, Hiroaki Yutani, and Dewey Dunnington. 2022. Ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics. https://CRAN.R-project.org/package=ggplot2.
40. Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2022. Dplyr: A Grammar of Data Manipulation. https://CRAN.R-project.org/package=dplyr.
41. Wickham, Hadley, and Maximilian Girlich. 2022. Tidyr: Tidy Messy Data. https://CRAN.R-project.org/package=tidyr.
42. Xie, Yihui. 2022a. Bookdown: Authoring Books and Technical Documents with r Markdown. https://CRAN.R-project.org/package=bookdown.
43. ———. 2022b. Knitr: A General-Purpose Package for Dynamic Report Generation in r. https://yihui.org/knitr/.
44. Xie, Yihui, J. J. Allaire, and Garrett Grolemund. 2018. R Markdown: The Definitive Guide. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown.
45. Xie, Yihui, Christophe Dervieux, and Emily Riederer. 2020. R Markdown Cookbook. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown-cookbook.