Welcome to {midsprint}

Aaron Pearson, Dani Chu, and Patrick Ward

2021-02-15

{midsprint}

This package provides coaches with a low barrier of entry into athlete profiling and positional tracking data. This file provides you with a simple walkthrough that allows you to model athlete:

- Speed-Acceleration
- Speed and acceleration over time from game data
- Speed and acceleration over time from Combine data

Current Updates

Major changes: some functions are not updated from version 0.1.1 and are not included in this update. This is a temporary change and will be rectified shortly. Functions omitted do not affect modeling and reporting aspects of the package, and were omitted because of inconsistency in how they are called upon. They are functions that are (presumably) not often called like `time_to_position_compare()` which calculates how long a players will take to reach a given distance.

Version 0.2.0 simplifies the functionality of the package, allowing you to create simple coach's reports in minutes. This is accomplished by relying on the following packages: `ggplot2`, `patchwork`, and `glue`.

Not much has changed in terms of workflow. However, functions within the workflow have been updated to provide greater personalization. For example, game_data() initially required the athlete's speed and acceleration values. Now, you can set the deault speed units (initially set as m/s) and can include the athlete's name. This allows the functions to return more accurate player values.

It is encouraged that you include speed metrics because the functions on the back-end assume all values are in metric. If your default speeds and acceleration are not in metric, you can type in either "yd/s", "ft/s", "mi/h", or "km/h". There is some leeway if your spelling is different, i.e. "km/h" can be typed in as "kilometers/hour" but this is not advised.

Please note: speed and acceleration must be in the same units. The functions cannot convert speed and acceleration values separately. To do so is unfeasible and would disrupt the workflow.

Finally, the updated vignette (walthrough) is more compact. The "must-knows" are set at the beginning of each section with more details proceeding them for those interested.

Installing and Loading midsprint

To install midsprint, you'll need to copy and paste the following code into your 'Console'.

```
devtools::install_github("aaronzpearson/midsprint")
```

Loading in the package allows you to make use of the functions that are kept within. Without loading them, they are much more difficult to reach and utilize. Loading midsprint into your current session is simply:

```
library(midsprint)
```

Loading Data

The package includes sample GPS and Combine data. If you are unsure of how to clean and analyze your current tracking data, I suggest following this walkthrough. You can always call upon the sample data to try things out before working on your own data set. The sample data will always be at your disposal and you cannot affect it permanently.

To use the sample data in the following example, set the game data to either midsprint::player_a or player_b and the combine data to player_40yd.

To load your data, I suggest download the tidyverse package by typing “install.packages(”tidyverse“)” into your console. Downloading the package will take a few minutes. This package is intuitive and easy to learn, and the `read_csv()` function is more versatile than the `read.csv()` function used below

Workflow Summary

In-Game Player Profiling

Single player

```
player_raw_data <- read.csv("path_to_data")
player_game_data <- game_data(player_raw_data$speed, player_raw_data$acceleration,
                              units = "m/s", player_name = "John Doe")
player_game_profile <- game_profile(player_game_data)
player_game_visual <- game_player_plot(player_game_profile)
```

Multiple players

```
player_one <- read.csv("path_to_data_one")
player_game_data_one <- game_data(player_one$speed, player_one$acceleration,
                              units = "m/s", player_name = "John Doe")
player_game_profile_one <- game_profile(player_game_data_one)

player_two <- read.csv("path_to_data_two")
player_game_data_two <- game_data(player_two$speed, player_two$acceleration,
                              units = "m/s", player_name = "Jane Doe")
player_game_profile_two <- game_profile(player_game_data_two)


player_game_compare <- game_player_plot(player_game_profile_one,
                                        player_game_profile_two)

# For player plots with 3+ athletes, repeat the workflow and include any number of player profiles
into player_game_plot()

player_game_compare <- game_player_plot(player_game_profile_one,
                                        player_game_profile_two
                                        player_game_profile_three,
                                        ...)
```

Speed-Acceleration Profile

Single player

```
player_raw_data <- read.csv("path_to_data")
player_game_data <- game_data(player_raw_data$speed, player_raw_data$acceleration,
                              units = "m/s", player_name = "John Doe")
player_speed_accel <- speed_accel(player_game_data)
player_speed_accel_plot <- speed_accel_plot(player_game_data)
```

Multiple players

```
player_one <- read.csv("path_to_data_one")
player_game_data_one <- game_data(player_one$speed, player_one$acceleration,
                              units = "m/s", player_name = "John Doe")

player_two <- read.csv("path_to_data_two")
player_game_data_two <- game_data(player_two$speed, player_two$acceleration,
                              units = "m/s", player_name = "Jane Doe")

# As above, the speed_accel_plot() function can include any number of player game data
player_speed_accel_plot_compare <- speed_accel_plot(player_game_data_one,
                                                    player_game_data_two,
                                                    ...)
```

40 yard dash/ NFL Combine Profile

Single player

```
player_raw_data <- read.csv("path_to_data")
player_combine_data <- combine_data(player_raw_data$speed, player_raw_data$acceleration,
                              units = "yd/s", player_name = "John Doe")
player_combine_profile <- combine_profile(player_combine_data)
player_combine_visual <- combine_player_plot(player_combine_profile)
```

Multiple players

```
player_one <- read.csv("path_to_data_one")
player_combine_data_one <- combine_data(player_one$speed, player_one$acceleration,
                              units = "yd/s", player_name = "John Doe")
player_combine_profile_one <- combine_profile(player_combine_data_one)

player_two <- read.csv("path_to_data_two")
player_combine_data_two <- combine_data(player_two$speed, player_two$acceleration,
                              units = "yd/s", player_name = "Jane Doe")
player_combine_profile_two <- combine_profile(player_combine_data_two)

# As above, the combine_player_plot() function can include any number of player game data
player_combine_compare <- speed_accel_plot(player_combine_profile_one,
                                           player_combine_profile_two,
                                                    ...)
```

Details

Coach's Reports

To create .pdf reports, you must start and end reports with `start_report()` and `save_report()`, otherwise it will not save to your working directory and can affect future work.

The report functions between `start_report()` and `save_report()` cannot be objects (do not put them in the form of `report_one <- game_report()`). Also, each report function creates a separate page in the .pdf. Therefore, if you'd like to have a single player one page one and multiple players on page two, call the function twice. The example below will clarify a report pipeline.

When the report pipeline is built, highlight and run the entire chunk of code.

```
start_report("my_report_name.pdf") # do not include .pdf at the end of the report name

game_report(game_profile_one) # single player game report on page 1
game_report(game_profile_one, # multiple player game report on page 2
            game_profile_two,
            ...)
speed_accel_report(game_data_one, # multiple player speed-acceleration report on page 3
                   game_data_two)
combine_report(combine_profile_one) # singple player combine report on page 4

save_report()
```

Data Structure

Game data is typically in the form of:

```
head(player_a)
#> # A tibble: 6 x 2
#>   speed accel
#>   <dbl> <dbl>
#> 1  0.02  0.03
#> 2  0.03  0.03
#> 3  0.02  0.03
#> 4  0.02  0.02
#> 5  0.02  0.02
#> 6  0.01  0.02
```

and Combine data in the form of:

```
head(player_40yd)
#>   Distance Split
#> 1        0  0.00
#> 2       10  1.50
#> 3       20  2.59
#> 4       40  4.47
```

In-Game & Combine Profile Breakdown

Using the same code as above, the following displays how the intermediate steps should appear. This example includes multiple players but is no different than when working with a single player.

Note: in-game and combine profiling are similar. Exchange the `game` functions for `combine` functions when needed.

```
player_raw_data_one <- midsprint::player_a
player_game_data_one <- game_data(player_raw_data_one$speed, player_raw_data_one$accel,
                            units = "m/s", player_name = "John Doe")

player_raw_data_two <- midsprint::player_b
player_game_data_two <- game_data(player_raw_data_two$speed, player_raw_data_two$accel,
                            units = "m/s", player_name = "Jane Doe")
```

Player profile:

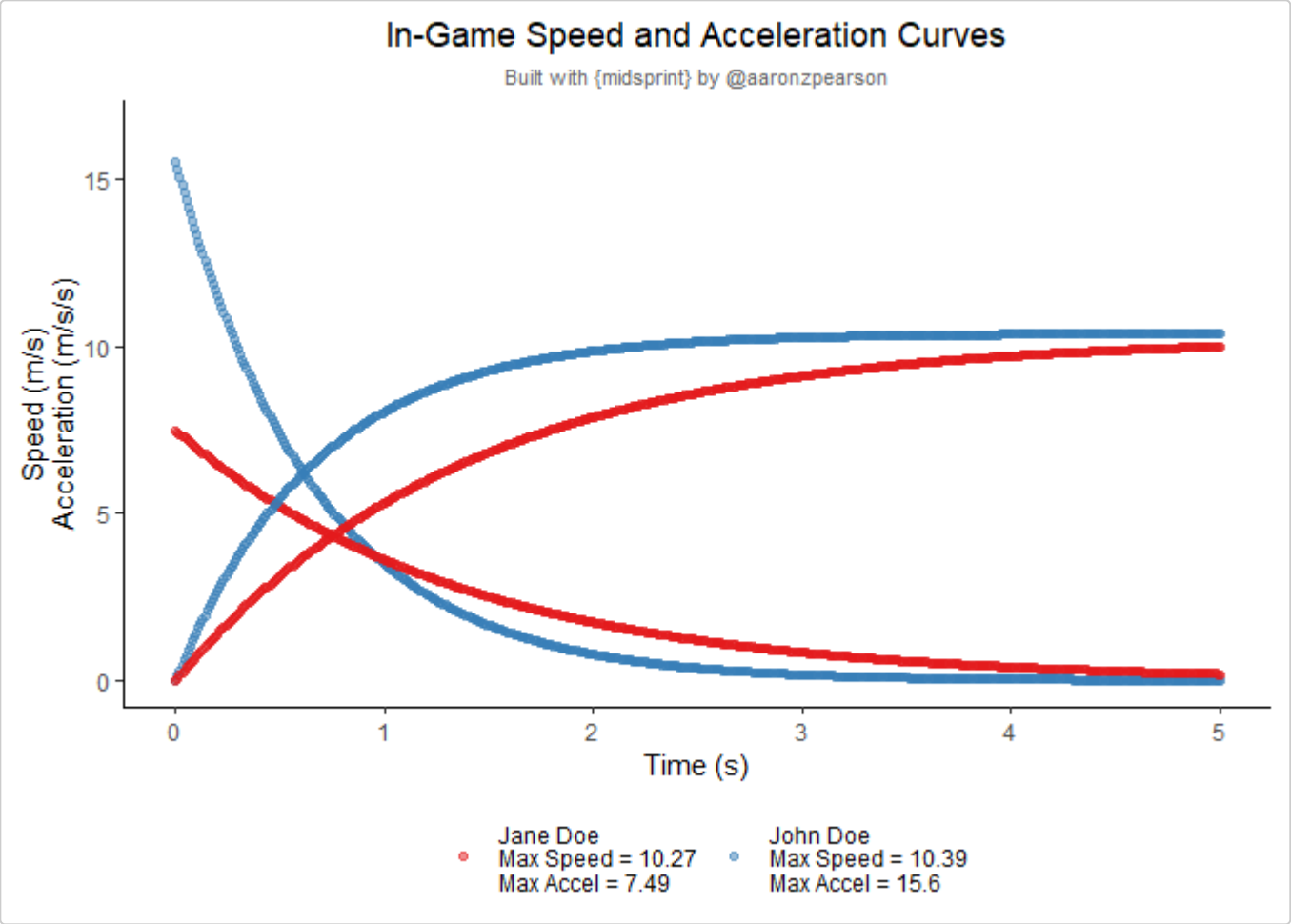
```
player_game_profile_one <- game_profile(player_game_data_one) # this will return player value in the
         console
#> Athlete: John Doe
#> Max Observed Player Speed = 10.39 m/s
#> Max Observed Player Acceleration = 15.6 m/s/s
#> Player Tau Value = 0.67 s


player_game_profile_two <- game_profile(player_game_data_two) # this will return player values in the
         console
#> Athlete: Jane Doe
#> Max Observed Player Speed = 10.27 m/s
#> Max Observed Player Acceleration = 7.49 m/s/s
#> Player Tau Value = 1.37 s
```

Player plot:

```
player_game_plot_compare <- game_player_plot(player_game_profile_one,
                                             player_game_profile_two)

player_game_plot_compare
```



Speed-Accel Profile Breakdown

As in the last section, this example includes multiple players. There is one difference when working with a single player versus multiple players and will be shown.

A new function is `speed_accel_observations()` which returns the observations used to build the linear model. This can be of interest to sports scientists when looking to compare or update player models.

Using the `game_data()` objects from above...

Player profile:

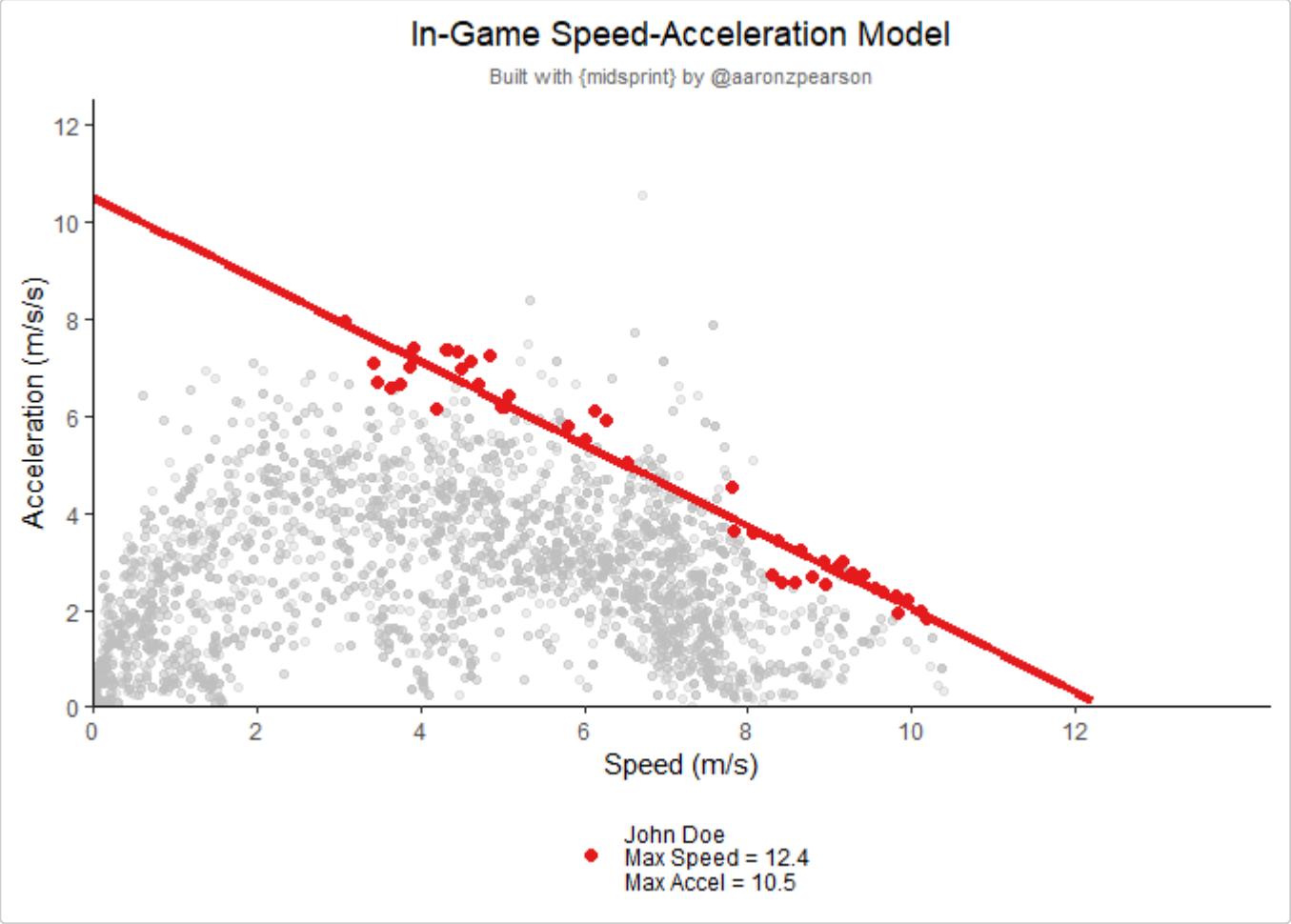
```
player_speed_accel_one <- speed_accel(player_game_data_one) # this will return player values in the
        console
#> Athlete : John Doe
#> Theoretical Max Speed = 12.4 m/s
#> Theoretical Max Accel = 10.5 m/s/s
#> Theoretical Tau Value = 1.18 s
#> r Squared = 0.95
#> Number of Observations = 45
```

```
player_speed_accel_two <- speed_accel(player_game_data_two) # this will return player values in the
        console
#> Athlete : Jane Doe
#> Theoretical Max Speed = 12.8 m/s
#> Theoretical Max Accel = 9.2 m/s/s
#> Theoretical Tau Value = 1.39 s
```

```
#> r Squared = 0.95
#> Number of Observations = 62
```

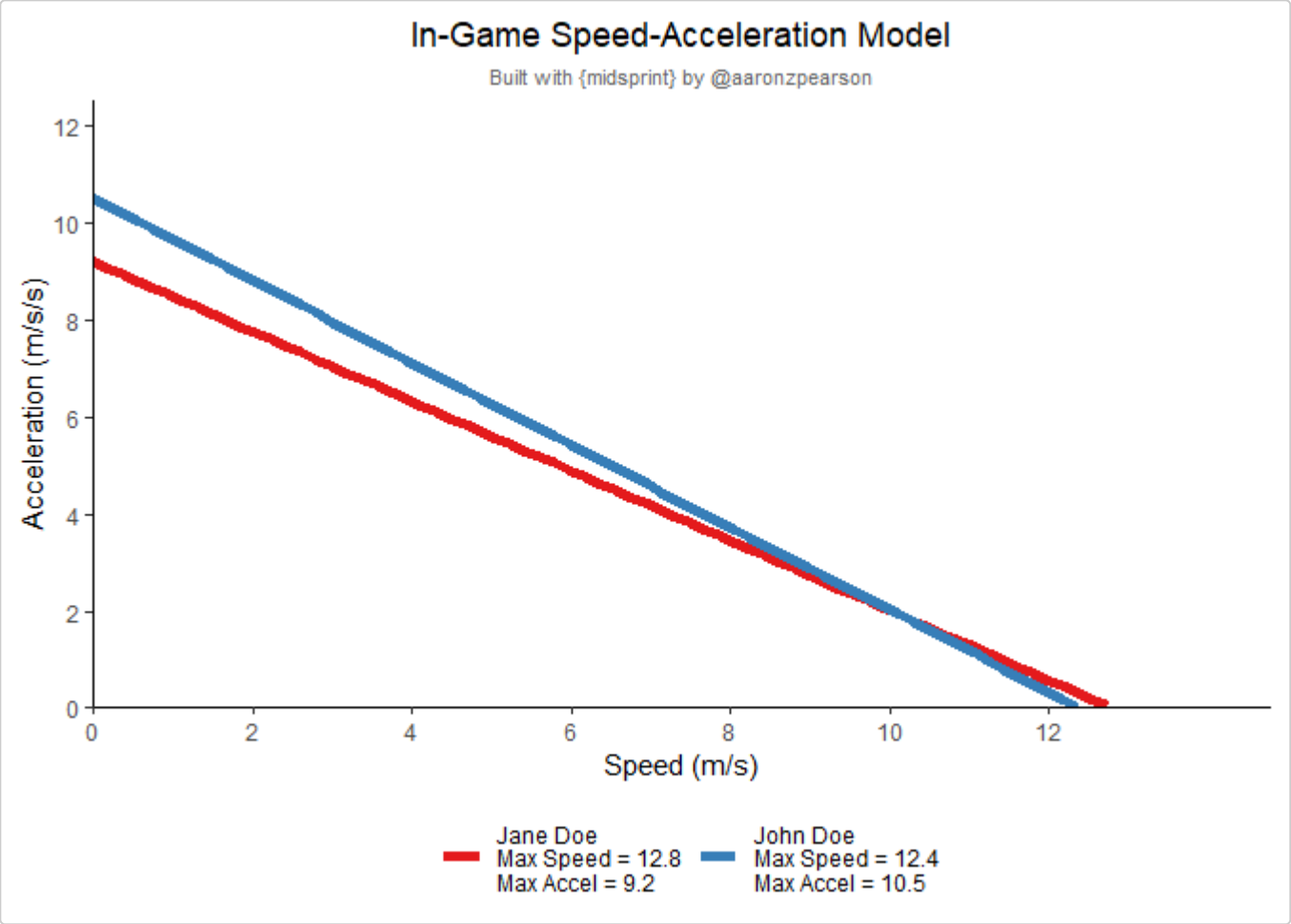
Single player plot:

```
player_speed_accel_plot <- speed_accel_plot(player_game_data_one)
player_speed_accel_plot
#> `geom_smooth()` using formula 'y ~ x'
#> Warning: Removed 11 rows containing missing values (geom_point).
#> Warning: Removed 12 rows containing missing values (geom_smooth).
```



Multiple player plot:

```
player_speed_accel_plot_compare <- speed_accel_plot(player_game_data_one,
                                                    player_game_data_two)
player_speed_accel_plot_compare
#> `geom_smooth()` using formula 'y ~ x'
#> Warning: Removed 24 rows containing missing values (geom_smooth).
```



Coach's Reports

This section uses the same code as above. Once run, you can open the sample report .pdf in your working directory. To see where it is saved, type `getwd()` into your console.

```
start_report("my_report_name") # do not include .pdf at the end of the report name

game_report(player_game_profile_one) # single player game report on page 1
game_report(player_game_profile_one, # multiple player game report on page 2
            player_game_profile_two)

speed_accel_report(player_game_data_one) # single player speed-accel report on page 3
speed_accel_report(player_game_data_one, # multiple player speed-acceleration report on page 4
                   player_game_data_two)

combine_report(player_combine_profile) # single player combine report on page 5

save_report()
#> png
#>   2
```

Other

To reach out to report an error or consult, please email `aaronzpearson@outlook.com`.

To cite midsprint in your work, please use `citation("midsprint")`

```
citation("midsprint")
#> Warning in citation("midsprint"): no date field in DESCRIPTION file of package
#> 'midsprint'
#> Warning in citation("midsprint"): could not determine year for 'midsprint' from
#> package DESCRIPTION file
#>
#> To cite package 'midsprint' in publications use:
#>
#>   Aaron Pearson, Dani Chu and Patrick Ward (NA). midsprint: {midsprint}
#>   makes building in-game player profiles easy. R package version 0.2.0.
#>
#> A BibTeX entry for LaTeX users is
#>
#>   @Manual{,
#>     title = {midsprint: {midsprint} makes building in-game player profiles easy},
#>     author = {Aaron Pearson and Dani Chu and Patrick Ward},
#>     note = {R package version 0.2.0},
#>   }
```