Getting started with palmsplusr

Tom Stewart

2018-01-12

Loading the PALMS dataset

A PALMS dataset (in csv format) is read in using the `read_palms()` function. This function checks that all required column names are present before converting the csv file to a simple features (spatial) object. If any columns are missing you will receive an error message. For a list of required column names, please see `read_palms()`.

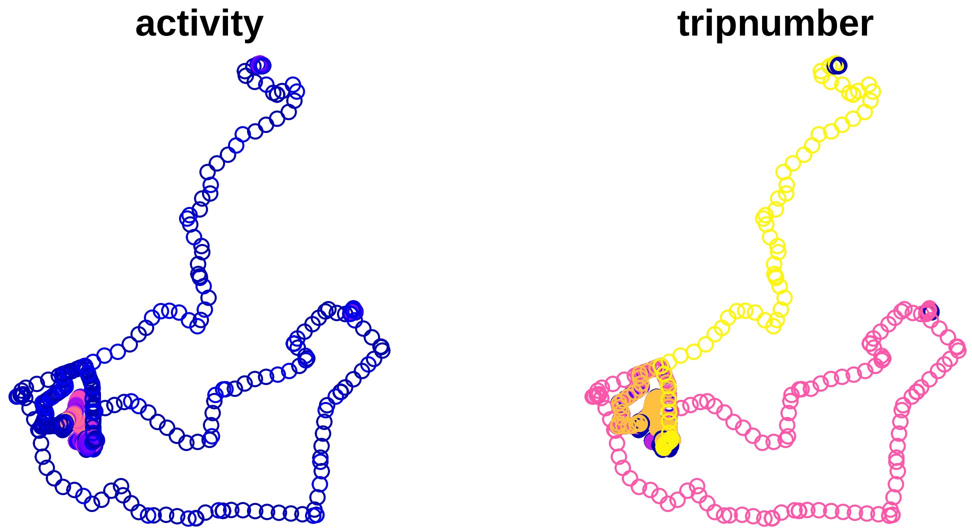
```r
library(palmsplusr)

palms <- read_palms(system.file("extdata", "one_participant.csv", package = "palmsplusr"))
names(palms)
```

```
##  [1] "identifier"         "datetime"           "dow"
##  [4] "fixtypecode"        "iov"                "tripnumber"
##  [7] "triptype"           "tripmot"            "activity"
## [10] "activityintensity"  "activityboutnumber" "sedentaryboutnumber"
## [13] "geometry"
```

This `palms` object contains 13 columns. Notice how the `lon` and `lat` columns that were present in the csv have been replaced by a `geometry` column. This is POINT geometry, as each row in `palms` represents a point.

In this example, the `palms` dataset contains data from one participant. You can plot this data to look at the distribution of points in space. Here I have chosen to plot two columns:

```r
plot(palms[, c("activity", "tripnumber")])
```



Building palmsplus

The palmsplus build process adds additional columns (i.e., fields) to the input `palms` dataset shown above. However, the user needs to specify what columns to add, and how to calculate them. This is done by creating a table with the name of the new column and the formula used to calculate it.

The function `palms_add_field(name, formula, domain_field = FALSE)` is used to add a field:

```
palms_add_field("weekday",    "dow < 6")
palms_add_field("weekend",    "dow > 5")
palms_add_field("indoors",    "iov == 3")
palms_add_field("outdoors",   "iov == 1")
palms_add_field("in_vehicle", "iov == 2")
palms_add_field("inserted",   "fixtypecode == 6")
palms_add_field("pedestrian", "tripmot == 1")
palms_add_field("bicycle",    "tripmot == 2")
palms_add_field("vehicle",    "tripmot == 3")
palms_add_field("nonwear",    "activityintensity < 0",  TRUE)
palms_add_field("wear",       "activityintensity >= 0", TRUE)
palms_add_field("sedentary",  "activityintensity == 0", TRUE)
palms_add_field("light",      "activityintensity == 1", TRUE)
palms_add_field("moderate",   "activityintensity == 2", TRUE)
palms_add_field("vigorous",   "activityintensity == 3", TRUE)
palms_add_field("mvpa",       "moderate + vigorous",    TRUE)
```

The code above can be replicated using `palms_load_defaults()`; however, this example demonstrates building field tables from scratch, as it is likely users will do this at some point.

The third parameter `domain_field` specifies whether the field should be summarized when creating `days` (see Building days below for more info).

Each time you add a new field, a new row is appended to the global `palmsplus_fields` table. If this table is printed, you will see it contains the fields that were just added:

```
palmsplus_fields
```

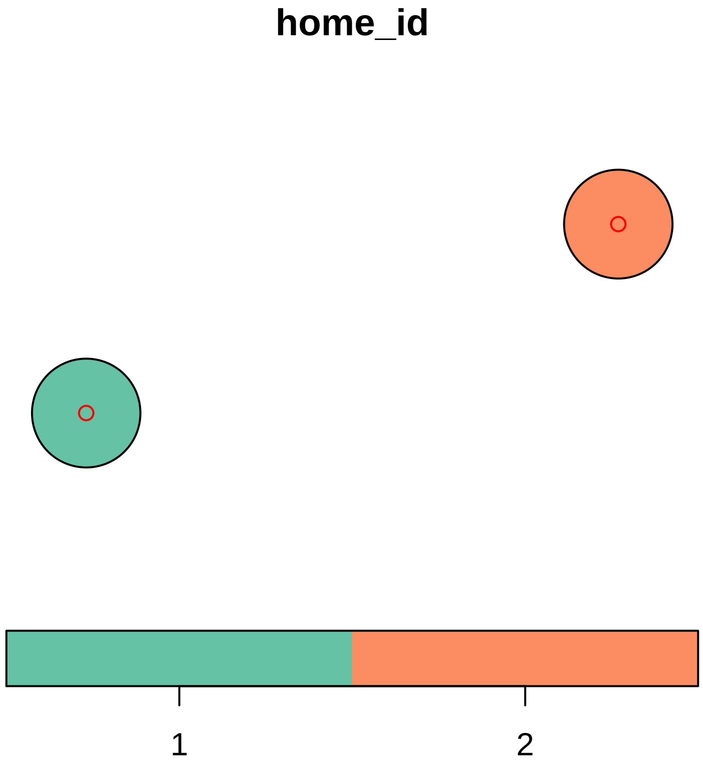
Table 1: palmsplus_fields

name	formula	domain_field
weekday	dow < 6	FALSE
weekend	dow > 5	FALSE
indoors	iov == 3	FALSE
outdoors	iov == 1	FALSE
in_vehicle	iov == 2	FALSE
inserted	fixtypecode == 6	FALSE
pedestrian	tripmot == 1	FALSE
bicycle	tripmot == 2	FALSE
vehicle	tripmot == 3	FALSE
nonwear	activityintensity < 0	TRUE
wear	activityintensity >= 0	TRUE
sedentary	activityintensity == 0	TRUE
light	activityintensity == 1	TRUE
moderate	activityintensity == 2	TRUE
vigorous	activityintensity == 3	TRUE
mvpa	moderate + vigorous	TRUE

Any variable from the **palms** dataset can be used to build formulas, although the true power of **palmsplusr** comes from integrating external data into these calculations.

In the next code snippet, a shapefile that represents home points is read in and buffered by 100 m to create polygons. These polygons are going to be used in a field formula. When this data is plotted, you will notice this person has two homes.

```
home.points <- read_sf(system.file("extdata/shapefiles/", "home.shp", package = "palmsplusr"))
home.buffer <- palms_buffer(point = home.points, distance = 100)

# Plot
plot(home.buffer[, "home_id"], key.pos = 1)
plot(home.points[, "home_id"], col = "red", add = TRUE)
```



Below, a new field called *at_home* is added. The formula for this field checks whether each point in the **palms** dataset falls inside the home.buffer polygons.

For a more detailed explanation about helper functions, such as **palms_in_polygon()**, and creating formulas, please see this article.

```
palms_add_field("at_home", "palms_in_polygon(., home.buffer, identifier)")
```

Once all of the fields have been added, you can build the **palmsplus** dataset using the **palms_build_palmsplus()** function. This takes the **palms** dataset as input:

```
palmsplus <- palms_build_palmsplus(palms)
```

```
## [1/1] Computed palmsplus for: BC0627
```

When printing the column names of the **palmsplus** dataset, you will notice it contains 30 columns: the original 13 plus the 17 that were added as fields:

```r
names(palmsplus)
```

```
##  [1] "identifier"          "datetime"            "dow"                
##  [4] "fixtypecode"         "iov"                 "tripnumber"         
##  [7] "triptype"            "tripmot"             "activity"           
## [10] "activityintensity"   "activityboutnumber"  "sedentaryboutnumber"
## [13] "weekday"             "weekend"             "indoors"            
## [16] "outdoors"            "in_vehicle"          "inserted"           
## [19] "pedestrian"          "bicycle"             "vehicle"            
## [22] "nonwear"             "wear"                "sedentary"          
## [25] "light"               "moderate"            "vigorous"           
## [28] "mvpa"                "at_home"             "geometry"
```

Now that the `palmsplus` dataset is built, it can be summarized in two ways. Building days, or building trajectories.

Building days

The `days` dataset provides a daily summary of the `domain_fields` present in the `palmsplus` dataset. Recall the `domain_fields` in the dataset above are:

- nonwear
- wear
- sedentary
- light
- moderate
- vigorous
- mvpa

These fields are summarized across several **domains**, which can be thought of as a subset of a day. Examples domains are: *during work hours*, *in greenspace*, *at home*, *in the town centre on weekends*.

Each of the `domain_fields` are summarized separately for each domain.

By default, only the *total* domain is used, which summarizes all data within each 24-hour period. Any additional domains need to be specified by the user. Domains are added the same way fields are added:

```
palms_add_domain("home", "at_home")
palms_add_domain("transport", "pedestrian | bicycle | vehicle")
```

```
palmsplus_domains
```

Table 2: palmsplus_domains

name	formula
home	at_home
transport	pedestrian | bicycle | vehicle

Notice how the domain formulas contain fields created earlier using `palms_add_field()`. Importantly, formulas are evaluated in the order they are specified, so one formula can contain another field name.

As `days` are built from the `palmsplus` dataset, each domain should have a column in `palmsplus` that signifies each point's domain membership. The `palms_build_palmsplus()` function seen above not only adds fields to the `palmsplus` dataset, it also adds domains.

Although I built `palmsplus` earlier, I will need rebuild it so the new domain columns are added. In a normal workflow, you would create domains before building `palmsplus`.

```
palmsplus <- palms_build_palmsplus(palms)
```

```
## [1/1] Computed palmsplus for: BC0627
```

Now when printing the `palmsplus` dataset there are 32 columns; two additional ones that represent the *home* and *transport* domains.

```
names(palmsplus)
```

```
##  [1] "identifier"          "datetime"            "dow"
##  [4] "fixtypecode"         "iov"                 "tripnumber"
##  [7] "triptype"            "tripmot"             "activity"
## [10] "activityintensity"   "activityboutnumber"  "sedentaryboutnumber"
## [13] "weekday"             "weekend"             "indoors"
## [16] "outdoors"            "in_vehicle"          "inserted"
## [19] "pedestrian"          "bicycle"             "vehicle"
## [22] "nonwear"             "wear"                "sedentary"
## [25] "light"               "moderate"            "vigorous"
## [28] "mvpa"                "at_home"             "home"
## [31] "transport"           "geometry"
```

The `palms_build_days()` function can now be used to build `days` from the `palmsplus` dataset:

```
days <- palms_build_days(palmsplus)
```

When looking at the structure of the `days` dataset, you will notice the `domain_fields` have been summarized for each domain. Note that the *duration* field and the *total* domain are included by default.

```
str(days)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    9 obs. of  26 variables:
##  $ identifier         : chr  "BC0627" "BC0627" "BC0627" "BC0627" ...
##  $ date               : Date, format: "2013-08-26" "2013-08-27" ...
##  $ total_nonwear      : num  344 954 501 354 1306 ...
##  $ total_wear         : num  427 486 557 0 134 ...
##  $ total_sedentary    : num  273 328 388 0 49 ...
##  $ total_light        : num  101 120.2 126.8 0 81.2 ...
##  $ total_moderate     : num  31 25 26.75 0 4.25 ...
##  $ total_vigorous     : num  22 13.2 15.8 0 0 ...
##  $ total_mvpa         : num  53 38.25 42.5 0 4.25 ...
##  $ total_duration     : num  771 1440 1058 354 1440 ...
##  $ home_nonwear       : num  344 871 501 354 1306 ...
##  $ home_wear          : num  4.5 58 150.2 0 134.5 ...
##  $ home_sedentary     : num  0 41.2 102.5 0 49 ...
##  $ home_light         : num  2.5 13.8 41.5 0 81.2 ...
##  $ home_moderate      : num  0.75 3 4.5 0 4.25 4.5 0.75 1.25 1.25
##  $ home_vigorous      : num  1.25 0 1.75 0 0 0 0.25 1 0.25
##  $ home_mvpa          : num  2 3 6.25 0 4.25 4.5 1 2.25 1.5
##  $ home_duration      : num  348 929 652 354 1440 ...
##  $ transport_nonwear  : num  0 0 0 NA 2.75 0 NA 0 0
##  $ transport_wear     : num  48.2 41.5 44.5 NA 0 ...
##  $ transport_sedentary: num  2 5.75 5 NA 0 ...
##  $ transport_light    : num  13.75 9.75 14.75 NA 0 ...
##  $ transport_moderate : num  16.5 15.8 14.8 NA 0 ...
##  $ transport_vigorous : num  16 10.2 10 NA 0 ...
##  $ transport_mvpa     : num  32.5 26 24.8 NA 0 ...
##  $ transport_duration : num  48.25 41.5 44.5 NA 2.75 ...
```

Building trajectories

The `trajectories` dataset contains individual trips, and trip-level summaries. Fields that you wish to calculate for each trajectory can be specified with `palms_add_trajectoy_field(name, formula, after_conversion = FALSE)`.

Notice the `palms_epoch()` helper function is used here. This is so the output is in seconds, rather than the number of rows.

```
epoch <- palms_epoch(palms)

palms_add_trajectory_field("mot",       "first(tripmot)")
palms_add_trajectory_field("date",      "first(as.Date(datetime))")
palms_add_trajectory_field("start",     "datetime[triptype==1]")
palms_add_trajectory_field("end",       "datetime[triptype==4]")
palms_add_trajectory_field("duration",  "as.numeric(difftime(end, start, units = \"secs\") + epoch)")
palms_add_trajectory_field("nonwear",   "sum(activityintensity < 0) * epoch")
palms_add_trajectory_field("wear",      "sum(activityintensity >= 0) * epoch")
palms_add_trajectory_field("sedentary", "sum(activityintensity == 0) * epoch")
palms_add_trajectory_field("light",     "sum(activityintensity == 1) * epoch")
palms_add_trajectory_field("moderate",  "sum(activityintensity == 2) * epoch")
palms_add_trajectory_field("vigorous",  "sum(activityintensity == 3) * epoch")
palms_add_trajectory_field("mvpa",      "moderate + vigorous")
palms_add_trajectory_field("length",    "as.numeric(st_length(.))",  TRUE)
palms_add_trajectory_field("speed",     "(length / duration) * 3.6", TRUE)
```

Again, adding all of these fields can be achieved with the `palms_load_defaults()` function; this is just used for illustration. Because `trajectories` are built from the `palmsplus` dataset, any variables used in the trajectory field formulas should be present in the `palmsplus` dataset.

```
trajectory_fields
```

Table 3: trajectory_fields

name	formula	after_conversion
mot	first(tripmot)	FALSE
date	first(as.Date(datetime))	FALSE
start	datetime[triptype==1]	FALSE
end	datetime[triptype==4]	FALSE
duration	as.numeric(difftime(end, start, units = “secs”) + epoch)	FALSE
nonwear	sum(activityintensity < 0) * epoch	FALSE
wear	sum(activityintensity >= 0) * epoch	FALSE
sedentary	sum(activityintensity == 0) * epoch	FALSE
light	sum(activityintensity == 1) * epoch	FALSE
moderate	sum(activityintensity == 2) * epoch	FALSE
vigorous	sum(activityintensity == 3) * epoch	FALSE
mvpa	moderate + vigorous	FALSE
length	as.numeric(st_length(.))	TRUE
speed	(length / duration) * 3.6	TRUE

The `after_conversion` parameter dictates whether the fields are calculated before or after the trip points are converted to `LINESTRING` geometry. Some fields can only be calculated on `LINESTRING` objects, such as the length of the line.

The `palms_build_trajectories()` function is used to build trajectories from the `palmsplus` dataset:

```
trajectories <- palms_build_trajectories(palmsplus)
```

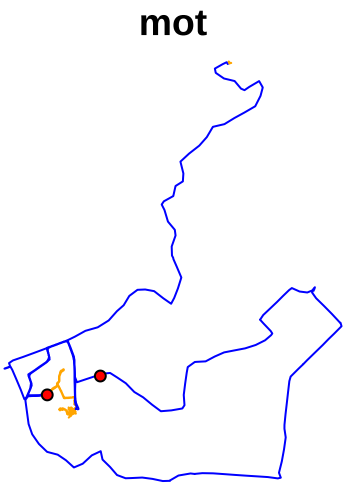
This creates the `trajectories` dataset, which has one row per trajectory, each containing the fields created above:

```
str(trajectories)
```

```
## Classes 'sf' and 'data.frame':   38 obs. of  17 variables:
##  $ identifier: chr  "BC0627" "BC0627" "BC0627" "BC0627" ...
##  $ tripnumber: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ mot       : int  1 1 1 1 1 1 1 3 1 1 ...
##  $ date      : Date, format: "2013-08-26" "2013-08-26" ...
##  $ start     : POSIXct, format: "2013-08-26 11:08:45" "2013-08-26 13:29:45" ...
##  $ end       : POSIXct, format: "2013-08-26 11:11:30" "2013-08-26 13:32:15" ...
##  $ duration  : num  180 165 555 255 675 1230 555 480 165 225 ...
##  $ nonwear   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ wear      : num  180 165 555 255 675 1230 555 480 165 225 ...
##  $ sedentary : num  30 0 60 60 0 90 15 285 0 45 ...
##  $ light     : num  45 75 285 135 435 195 75 150 0 105 ...
##  $ moderate  : num  0 45 150 45 195 510 195 45 60 75 ...
##  $ vigorous  : num  105 45 60 15 45 435 270 0 105 0 ...
##  $ mvpa      : num  105 90 210 60 240 945 465 45 165 75 ...
##  $ length    : num  198 125 518 129 401 ...
##  $ speed     : num  3.97 2.74 3.36 1.82 2.14 ...
##  $ geometry  :sfc_LINESTRING of length 38; first list element:  XY [1:12, 1:2] 175 175 175 175 175 .
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr  "1" "2" "3" "4" ...
##   .. ..$ : NULL
##  - attr(*, "sf_column")= chr "geometry"
##  - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA NA ...
##   ..- attr(*, "names")= chr  "identifier" "tripnumber" "mot" "date" ...
```

As the `trajectories` dataset contains `LINESTRING` geometry, we can plot it:

```
plot(trajectories[, "mot"], pal = c("orange", "blue"), key.pos = NULL)

# Add the home buffer polygons to the plot
plot(home.buffer[, 1], col = "red", key.pos = NULL, add = TRUE)
```



Adding trajectory locations

Trajectory start and end locations can also be calculated. This is used to identify specific trips, such as trips to work or school. This is done with the function `palms_add_trajectory_location(name, start_criteria, end_criteria)`.

The `start_criteria` and `end_criteria` parameters should be fields already calculated in `palmsplus`.

To demonstrate, I'm going identify all trajectories that start at home and end at school. I will need to read in an additional shapefile that contains the schoolyard polygon, and add a new *at_school* field to `palmsplus`.

```
school <- read_sf(system.file("extdata/shapefiles/", "school.shp", package = "palmsplusr"))

palms_add_field("at_school", "palms_in_polygon(., school)")

palmsplus <- palms_build_palmsplus(palms)
```

```
## [1/1] Computed palmsplus for: BC0627
```

Now that `palmsplus` contains the *at_school* field, I'm going to add a `trajectory_location` that starts *at_home* and ends *at_school*. Recall the *at_home* field was created earlier.

```
palms_add_trajectory_location("home_school", "at_home", "at_school")
```

```
trajectory_locations
```

Table 4: trajectory_locations

name	start_criteria	end_criteria
home_school	at_home	at_school

Now the `trajectories` dataset can be rebuilt. Additional columns will be added for each entry in the `trajectory_locations` table indicating whether the trajectory meets both the start and end criteria.

```
trajectories <- palms_build_trajectories(palmsplus)
names(trajectories)
```

```
##  [1] "identifier"  "tripnumber"  "mot"         "date"        "start"
##  [6] "end"         "duration"    "nonwear"     "wear"        "sedentary"
## [11] "light"       "moderate"    "vigorous"    "mvpa"        "home_school"
## [16] "length"      "speed"       "geometry"
```

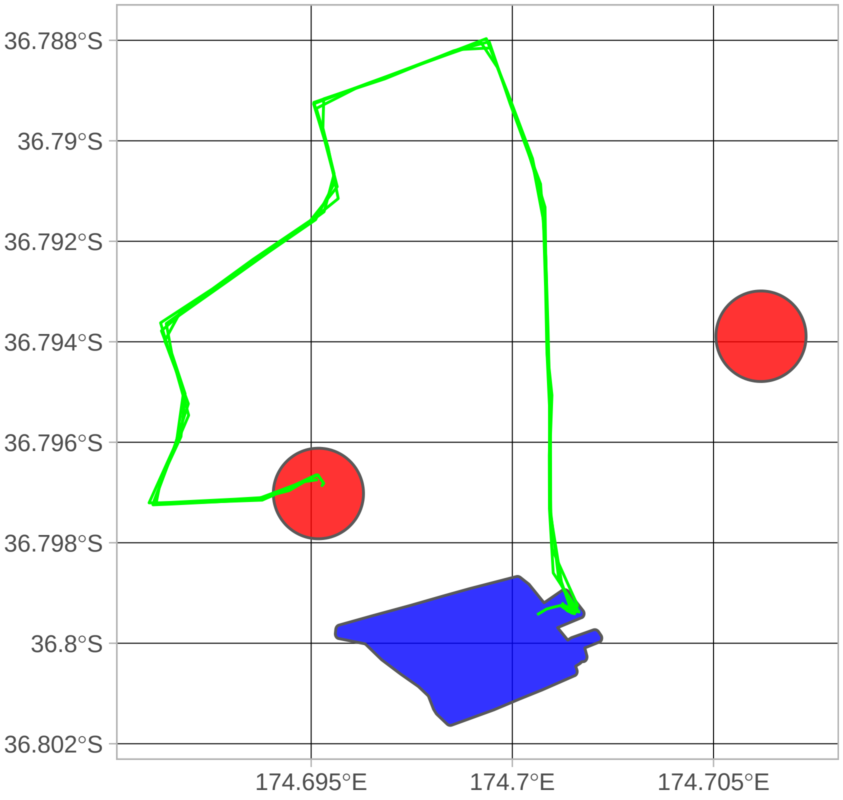
Notice how the `trajectories` dataset now contains an extra column `home_school` which signifies whether the trip started at home and ended at school. In total, 4/38 trajectories meet this criteria:

```
table(trajectories$home_school)
```

```
##
##  0  1
## 34  4
```

We can double check the results by plotting these trajectories. The ggplot2 package can also be used instead of R's base plot. Please see the article ggplot2 and palmsplusr.

```
library(ggplot2)

ggplot() +
  geom_sf(data=home.buffer, fill = "red", alpha = 0.8) +
  geom_sf(data=school, fill = "blue", alpha = 0.8) +
  geom_sf(data=trajectories %>% filter(home_school == 1), colour = "green") +
  theme_light()
```



It looks like they all start at home and end at school!

Building multimodal trajectories

The `trajectories` dataset can be further processed into multimodal trips using the `palms_build_multimodal()` function. This will join two or more trajectories together if they are within a spatial and temporal threshold.

Multimodal trajectories are important, because PALMS assigns a new trip number each time the travel mode changes. A change in travel mode part way along the trip could cause none of the trip 'segments' to meet the `start_criteria` and `end_criteria`.

This is also useful for identifying trip chains and transit use (e.g., walk-vehicle-walk).

The fields that are summarized for each multimodal trajectory are specified using `palms_add_multimodal_field(name, func)`.

The `name` refers to a field name in the `trajectories` dataset, while `func` specifies a summary function used to aggregate trajectory fields (usually `sum()` or `mean()`). For example, you probably want to sum the duration of trip segments, but take the average (mean) of the segment speeds.

```r
palms_add_multimodal_field("duration", "sum")
palms_add_multimodal_field("speed", "mean")
```

Alternatively, you can pass in a vector of field names that use the same summary function:

```r
palms_add_multimodal_field(c("nonwear", "wear", "sedentary", "light", "moderate",
                             "vigorous", "mvpa", "length"), "sum")
```

```r
multimodal_fields
```

Table 5: multimodal_fields

name	func
duration	sum
speed	mean
nonwear	sum
wear	sum
sedentary	sum
light	sum
moderate	sum
vigorous	sum
mvpa	sum
length	sum

It should be noted that these `multimodal_fields` are also created by `palms_load_defaults()`.

The `trajectories` dataset is then passed to `palms_build_multimodal(spatial_threshold, temporal_threshold)` to build the `multimodal` dataset.

The `spatial_threshold` is the distance (in meters) between the end of one trajectory and the start of the next, while the `temporal_threshold` is the time between these (in minutes). I've chosen a criteria of 200 m and 10 minutes:

```r
multimodal <- palms_build_multimodal(trajectories, 200, 10)
```

```
## Calculating multimodal eligibility...done
## Assigning trip numbers...done
## Calculating fields...done
```

When printing the column names of `multimodal` you will notice an overall summary for each field, but also for each travel mode (note this participant has no bicycle trips; mot = 2):

```
names(multimodal)
```

```
##  [1] "identifier"      "mmt_number"      "trip_numbers"
##  [4] "n_segments"      "mot_order"       "start"
##  [7] "end"             "home_school"     "mot_1_duration"
## [10] "mot_1_length"    "mot_1_light"     "mot_1_moderate"
## [13] "mot_1_mvpa"      "mot_1_nonwear"   "mot_1_sedentary"
## [16] "mot_1_vigorous"  "mot_1_wear"      "mot_3_duration"
## [19] "mot_3_length"    "mot_3_light"     "mot_3_moderate"
## [22] "mot_3_mvpa"      "mot_3_nonwear"   "mot_3_sedentary"
## [25] "mot_3_vigorous"  "mot_3_wear"      "duration"
## [28] "nonwear"         "wear"            "sedentary"
## [31] "light"           "moderate"        "vigorous"
## [34] "mvpa"            "length"          "mot_1_speed"
## [37] "mot_3_speed"     "speed"           "geometry"
```

Recall there were 38 observations in the `trajectories` dataset. We can see what trajectories were combined by looking at the `trip_numbers` variable

```
multimodal$trip_numbers
```

```
##  [1] "1"        "2"        "3-4"      "5-6"      "7"        "8-9"
##  [7] "10"       "11"       "12"       "13"       "14-15"    "16"
## [13] "17"       "18"       "19"       "20"       "21"       "22"
## [19] "23"       "24"       "25"       "26"       "27"       "28"
## [25] "29"       "30"       "31"       "32"       "33-34"    "35"
## [31] "36-37-38"
```

The `mot_order` variable retains the mode of travel order of each trajectory in the multimodal trips:

```
multimodal$mot_order
```

```
##  [1] "1"     "1"     "1-1"   "1-1"   "1"     "3-1"   "1"     "1"
##  [9] "1"     "1"     "3-1"   "1"     "1"     "1"     "1"     "1"
## [17] "1"     "3"     "3"     "3"     "3"     "1"     "1"     "1"
## [25] "1"     "1"     "1"     "1"     "3-1"   "1"     "1-3-1"
```

If any `trajectory_locations` were created, they will also be added to `multimodal` (notice the *home_school* field in the column name vector above).

Saving geometry and results

You can save all datasets as a csv file. As `palmsplus`, `trajectories` and `multimodal` contain geometry, you can also save these as ESRI shapefiles:

```
write_csv(palmsplus, "palmsplus.csv")
write_csv(days, "days.csv")
write_csv(trajectories, "trajectories.csv")
write_csv(multimodal, "multimodal.csv")

st_write(palmsplus, "palmsplus.shp")
st_write(trajectories, "trajecories.shp")
st_write(multimodal, "multimodal.shp")
```