Tutorial MTTfireCAL R Package

Prepared by Bruno A. Aparício (Bruno.a.aparicio@gmail.com)

Version 1 - March 2023

Lisbon, Portugal

Disclaimer

This document reports MTTfireCAL, a R package designed to assist in the calibration of Minimum Travel Time (MTT) algorithm. MTTfireCAL is a public domain package and can be used and copied freely. MTTfireCAL has been tested in multiple study areas and by several users. However, no warranty is given that the package is completely error-free. If you encounter problems with the code, or have suggestions for improvement, please report them at https://github.com/bmaparicio/MTTfireCAL/issues or directly contact the authors at Bruno.a.aparicio@gmail.com.

Contents

1. Overview

MTTfireCAL is a free and innovative R package that enables fast calibration of the Minimum Travel Time (MTT) fire spread models. The package was built to support all the steps required to run MTT (or its command line version – FConstMTT), including the i) characterization of the historical fire regime in the study area, ii) download and processing of the fire weaher data, iii) generation of random ignitions across the study area, iv) running FConstMTT with combinations of multiple fire duration values, and v) evaluating the quality of the calibration by comparing historical and simulated fire patterns.

For a complete description of MTTfireCAL and discussion of the method see Aparício et al. (2023).

2. Before MTTfireCAL

Before start using MTTfireCAL, it is of great importance that the use is familiar with fire behavior metrics and with MTT parameters. All of the variables are explained in the Flammap manual and tutorial (https://owfflammaphelp62.firenet.gov/). Also, refer to the article by Finney (2002).

Before start using MTTfireCAL, please consider that every time your machine sleeps or hibernates, the processes are halted. This will lead to a longer period using the functions than expected and may cause some to fail. Hence, it is crucial that the user prevents its machine from sleeping or hibernating when inactive. In a machine with Windows 10, this can be done by assessing the control panel > system and security > power options > change when the computer sleeps

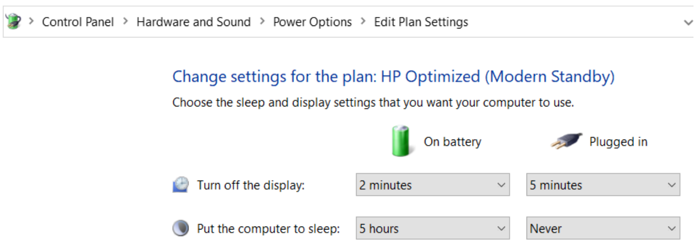


Figure 1 – Screenshot of the definitions recommended in the power options. Please do not allow your computer to sleep as it will interrupt the use of MTTfireCAL and may result in errors.

The user must also download the FConstMTT (MTT command line version) from https://www.alturassolutions.com/FB/FB_API.htm. This is explained in detail in the section 8.1.

3. Installing MTTfireCAL R Package

To use MTTfireCAL you first need to download and install R and RStudio in your machine (follow the instructions at https://posit.co/download/rstudio-desktop/).
Afterwards, if your machine uses Windows, you will need to have a version of Rtools installed that matches your version of R. Rtools can be downloaded here. After downloading the Rtools, install it in your machine following the recommendations here

Finally, the users must install the package devtools to be able to install packages from GitHub:

```
install.packages("devtools")
install.packages("usethis")
```

And then load the devtools package in your R session and install MTTfireCAL:

```
library(devtools)
install_github("bmaparicio/MTTfireCAL")
```

Please choose to update the required packages as you install the MTTfireCAL.

In some circumstances one's machine may not connect with GitHub servers and install MTTfireCAL (e.g. user's machine might be behind a firewall). If this is the case, the user can download the entire package as zip (as shown in Figure 2) and install it following:

```
devtools::install_local("C:/path_to_package/name_of_package.zip")
```

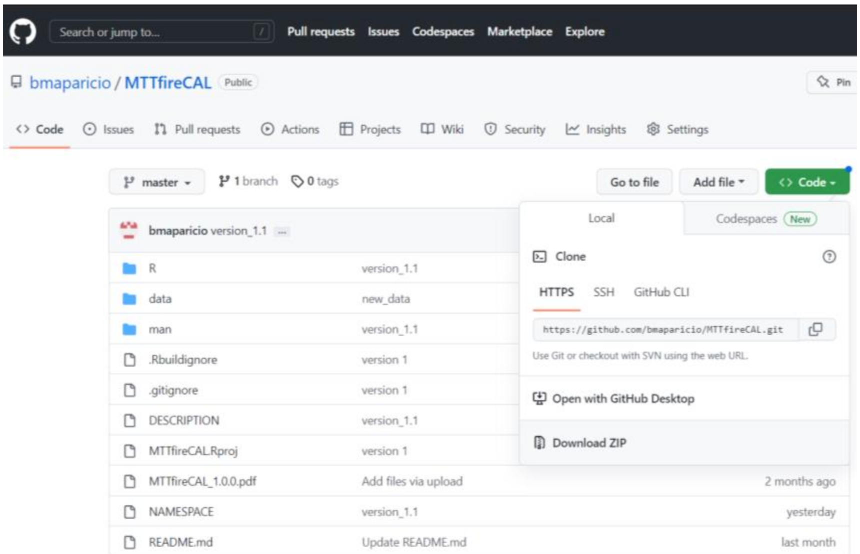


Figure 2 – Screenshot of how to download the entire package from GitHub

4. Getting started

After installing MTTfireCAL, load the package

```
library(MTTfireCAL)
```

The package contains data that will be used throughout this tutorial, including a shapefile of a study area in south Portugal; a shapefile of dated fire perimeters in the same area; an ignition probability surface for the same area; a landscape file; and two maps of fuel models (raster files). These files will be loaded when needed.

First, plot the study area and fire perimeters that are included in the package:

```
library(rgdal)
library(rnaturalearth)
library(sf)
library(ggplot2)
library(dplyr)

#load the study area shapefile
my_study_area <- readOGR("~/study_area_mttfirecal.shp")

#lets take a look at the structure of the shapefile
my_study_area

> class       : SpatialPolygonsDataFrame
> features    : 1
> extent      : -8.864971, -8.247533, 37.08893, 37.51487  (xmin, xmax, ymin, ymax)
> crs         : +proj=longlat +datum=WGS84 +no_defs
> variables   : 1
> names       : id
> value       :  1
```

Note the structure of the shapefile. The shapefile has only one field (named “id”) and only one feature (i.e. only one polygon). It is mandatory that the user defined study areas have a similar structure, with **only one feature**.

Now, we will plot the study area in the world map using the WGS84 coordinate system. It is always important to prevent errors by checking the original data.

```
world_st <- ne_countries(scale = "medium", returnclass = "sf")
country_identified <- subset(world_st,sovereignt=="Portugal")


WorldData <- map_data('world') %>% filter(region == country_identified_use) %>%
fortify
surroundings <- map_data('world') %>% fortify


my_study_area <- spTransform(my_study_area, CRS("+proj=longlat +datum=WGS84
+no_defs +ellps=WGS84 +towgs84=0,0,0"))
```

```
ggplot() +
  coord_fixed(xlim=c(min(WorldData$long),max(WorldData$long)),
              ylim = c(min(WorldData$lat),max(WorldData$lat)))+
  suppressWarnings(geom_map(data = surroundings, map = surroundings,
                            aes(x = long, y = lat, group = group, map_id=region),
                            fill = "grey", colour = "black", size=0.5))+
  suppressWarnings(geom_map(data = WorldData, map = WorldData,
                            aes(x = long, y = lat, group = group, map_id=region),
                            fill = "antiquewhite", colour = "black", size=0.5))+
  suppressMessages(geom_polygon(data = my_study_area, aes(long, lat),
                                colour = alpha("darkred", 1/2), size = 0.7,
fill=NA,linetype = "dashed"))+
  theme(panel.grid.major = element_line(color = "gray", linetype = "dashed", size
= 0.5),
        panel.background = element_rect(fill = "aliceblue"))+
  xlab("Longitude") + ylab("Latitude") + ggtitle("Study area location")
```

Which will return the figure below.

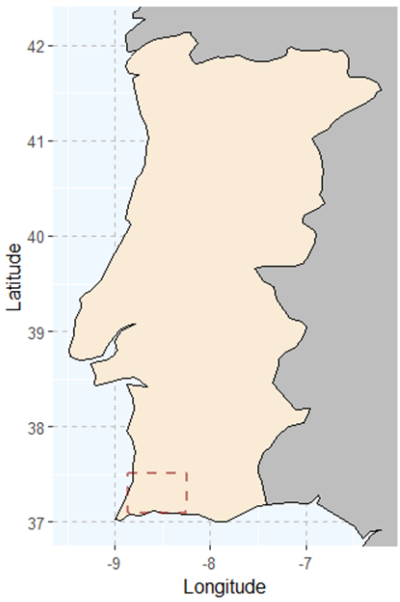


Figure 3 – Location of the study area (in red dashed line).

The Figure 3 confirms that the study area is plotted in the expected region. We can now proceed with the use of MTTfireCAL

The usage of MTTfireCAL starts with the download of meteorological data from the ERA5-Land (Muñoz-Sabater et al., 2021).

5. Downloading and preparing weather data

The first function that we will use is the *get_fire_weather* function.
The function *get_fire_weather* automatically downloads the required weather variables from ERA5-Land dataset (Muñoz-Sabater et al., 2021) at an hourly resolution and stores it as netcdf files. Then, it processes the data and converts it into a text file that will be used by other functions of MTTfireCAL.

To understand how the function works and what inputs are needed, see the help page:

```
?get_fire_weather
```

The help of this function reveals that this function requires a shapefile of the study area (*study.area*), a shapefile containing the dated fire perimeters (*my.fires*), the UTC time zone (*utc.zone*), the source for the weather data, either "era5-land" or "era5" (*data.source*), the Copernicus Climate Data Store login information, namely the user id (*wf_user*) and the key (*wf_key*), and the path to where the outputs should be stored (*output.folder*).

A sample of dated fire perimeters can be found in the tutorial data. Before using the dated fire perimeters shapefile, it is useful to understand its structure

```
#load the dated fire perimeters
my_dated_fires <- readOGR("~/dated_fire_perimeters_mttfirecal.shp")

#lets take a look at the structure of the shapefile
my_dated_fires

> class       : SpatialPolygonsDataFrame
> features    : 221
> extent      : 510436.3, 600468.9, 4104915, 4153136  (xmin, xmax, ymin, ymax)
> crs         : +proj=utm +zone=29 +datum=WGS84 +units=m +no_defs
> variables   : 5
> names       :    ID, Year,   Date_ini,   Date_end,      Area_ha
> min values  : 15661, 2001, 2001-06-23, 2001-06-23,            0
> max values  :  7848, 2021,        NaN,        NaN,          902
```

As shown above, the dated fire perimeters shapefile contains five variables and 221 fire perimeters that occurred between 2001 and 2021. **The variables that are listed are required to be present in the shapefile, with exactly the same names as shown.** Note that not all the fire perimeters are dated as some show NaN in either Date_ini and Date_end. This does not influence the normal usage of the functions as only the dated fire perimeters will be used when downloading the weather variables. The variable ID identifies the different fire perimeters with a unique identifier. Hence, it is crucial that **unique fires have a unique value in the ID field**.

Like in the previous case, it is helpful to plot the fire perimeters to ensure that it overlaps with the study area. First make sure to transform all the layer to the same coordinate system

```
my_dated_fires <- spTransform(my_dated_fires, CRS("+proj=longlat +datum=WGS84
+no_defs +ellps=WGS84 +towgs84=0,0,0"))
```

And then plot both shapefiles

```
ggplot() +
 coord_fixed(xlim = c(-9, -8), ylim = c(37, 37.6)) +
 suppressWarnings(geom_map(data = surroundings, map = surroundings,
 aes(x = long, y = lat, group = group, map_id=region),
 fill = "grey", colour = "black", size=0.5))+
 suppressWarnings(geom_map(data = WorldData, map = WorldData,
 aes(x = long, y = lat, group = group, map_id=region),
 fill = "antiquewhite", colour = "black", size=0.5))+
 suppressMessages(geom_polygon(data = my_study_area, aes(long, lat),
 colour = alpha("darkred", 1/2), size = 0.7, fill=NA,linetype = "dashed"))+
 suppressMessages(geom_polygon(data = my_dated_fires, aes(long, lat, group=group),
 colour = alpha("darkred", 1/2), size = 0.7, fill=NA))+
 theme(panel.grid.major = element_line(color = "gray", linetype = "dashed", size =
0.5),
 panel.background = element_rect(fill = "aliceblue"))+
 xlab("Longitude") + ylab("Latitude")
```

Which will return the figure below

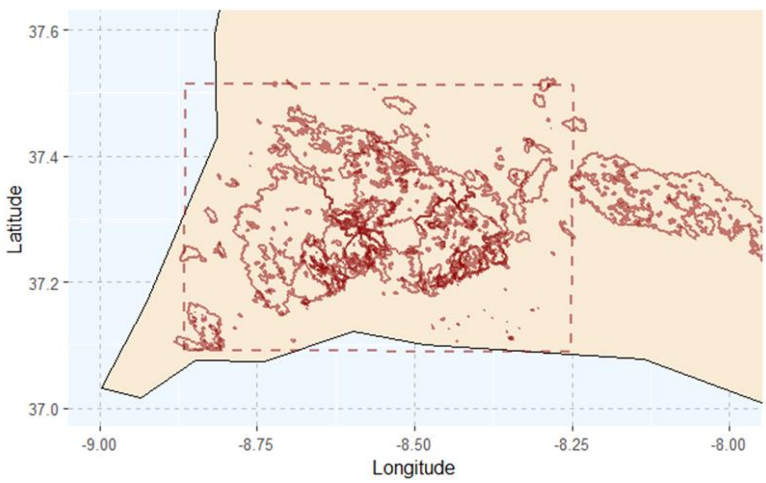


Figure 4 – Study area (in red dashed line) and the wildfires in the database (grey perimeters).

The Figure 4 shows that both the study area and the dated fire perimeters overlap. Hence, we are ok to proceed and use the function *get_fire_weather* to download the meteorological data from ERA5-Land.

Note that a shapefile containing the dated fires at a national or regional scale can be used. Only the fire perimeters intersecting the study area will be considered.

5.1. Function get_fire_weather

Before running the function make sure that the folder where you want to save the meteorological data exists.

```
get_fire_weather(study.area="~/study_area_mttfirecal.shp",
  my.fires="~/dated_fire_perimeters_mttfirecal.shp",
  output.folder="~/meteorological_data",
  data.source = "era5-land",
  utc.zone=+1,
  wf_user="12345",
  wf_key="12345678-1234-1234-1234-123456789123")
```

Please note that both the wf_user and the wf_key are only examples and do not correspond to a valid user and key. The user can retrieve this information from https://cds.climate.copernicus.eu/user after logging in.

Also note that the *utc.zone* was set to +1. This reflects the daylight saving time in Portugal during the fire season (that corresponds to the summer). It is essential to set this parameter correctly as the full ERA5-Land dataset is in UTC+0. A mistake in this parameter might compromise the entire fire weather analysis.

While running the function *get_fire_weather*, the user gets the following message in the R console:

```
User xxxxx for cds service added successfully in keychain
```

This message means that the data is being proceeded for download and the progress can be tracked in the webpage of the CDS at https://cds.climate.copernicus.eu/cdsapp#!/yourrequests

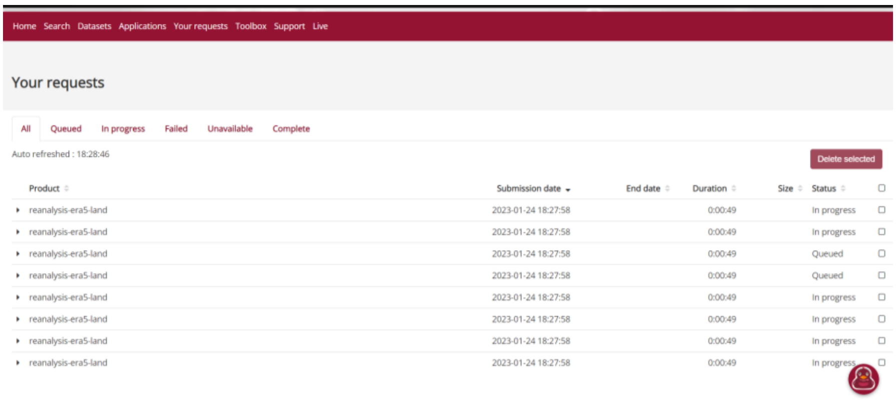


Figure 5 – Screenshot of the users' personal page at the CDS webpage. This shows the data being downloaded.

Figure 5 shows a screen shot of CDS webpage while downloading the files. Note that more than one netcdf file is downloaded. This results from the limitation in download more than 1,000 items at the same time[1].

Please allow some time for the function to download all the data required as this process can take from a few minutes to a few hours (depending on the number of fires in the user study area, user's internet connection, and availability of CDS servers). This function should not take more than 30 minutes when using the study area in this tutorial.

After running the function, a csv file named "fire_weather_study_area" is saved in the directory specified by the user (here ~/meteorological_data). This csv file contains all the meteorological information required and is shown in Table 1.

ID	*row*	*col*	*lon*	*lat*	*day_files*	*day_UTC_corrected*	*temperature*	*RH*	*WS*	*WD*
51	5	9	-8.6	37.2	2001080600	2001080601	22.4	56.01	13.65	354.73
51	5	9	-8.6	37.2	2001080601	2001080602	21.92	57.91	12.96	357.32
51	5	9	-8.6	37.2	2001080602	2001080603	21.42	60.49	12.19	357.98

Table 1 – Example of the text file (csv) generated after using the functions *get_fire_weather* and *fire_weather_nc*. If the user wants to use other meteorological data than ERA5-Land, then they need to provide a similar table in build_report function. The ID represents the id of the fire perimeter in the shapefile used as input in the function *get_fire_weather* or *fire_weather_nc*; the row and col represent the row number and the column number (respectively) where the information regarding the weather conditions is stored in the netcdf file for the fire event specified in the column ID; lon and lat represent the longitude and latitude (respectively) of the ERA5-Land point that is either contained by the fire perimeter or the closest to the fire perimeter; day_files represent the date and time of the specified weather conditions (format yyyymmddhh) in UTC+0; day_UTC_corrected represent the date and time of the specified weather conditions (format yyyymmddhh) in the UTC specified by the user (here is UTC+1); temperature represents the temperature in degrees Celsius in the specified point at the specified time; RH represents the relative humidity in percentage in the specified point at the specified time; WS represents the wind speed in km/h in the specified point at the specified time; and WD represents the wind direction in degrees in the specified point at the specified time.

It is important to refer that occasionally this function may return an error similar to the one below:

```
Error in { :
  task 19 failed - "Timeout was reached: [cds.climate.copernicus.eu] Send failure:
Connection was reset"
```

This error is the consequence of either unstable internet connection or unavailability of CDS servers to process the request in a reasonable time. In case of this error, the download of netcdf files will not be finished. The user can repeat the get_fire_weather function. However, if the function is once again not unsuccessful, the user can manually download the remaining files. The files that were not downloaded are stored in the user's request webpage of CDS (https://cds.climate.copernicus.eu/cdsapp#!/yourrequests).

[1] https://confluence.ecmwf.int/pages/viewpage.action?pageId=308052947

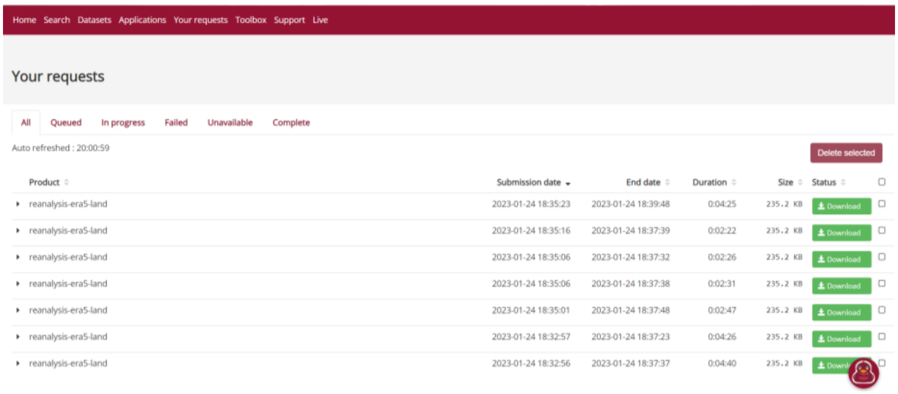


Figure 6 - Screenshot of the users' personal page at the CDS webpage. This shows the data ready to be downloaded. A similar window is only showed when an error occurred while downloading the data.

If the user decides to manually download the files, please consider that the files should be saved with the same name as in the "target" parameter (Figure 7).

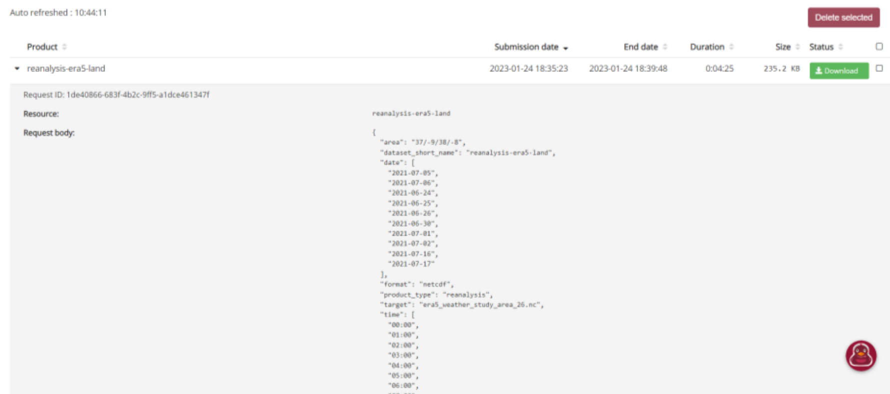


Figure 7 - Screenshot of the users' personal page at the CDS webpage showing the target parameter.

After saving all the files, the csv "fire_weather_study_area" can be generated by using the function fire_weather_nc. This function does not download netcdf files. Instead, it reads and processes the netcdf files in the user's machine.

5.2 Function fire_weather_nc

If climate data was previously downloaded as netcdf from ERA5-Land, the user can use the function *fire_weather_nc*. This function is similar to get_fire_weather, with the difference that it does not require the `wf_user` and `wf_key`. Instead, it requires the path to the folder containing the netcdf files.

Before using this function, create a new folder in your tutorial data named "get_fire_weather_meteorological_data", where the results of this function will be stored.

```
fire_weather_nc(study.area="~/study_area_mttfirecal.shp",
                my.fires="~/dated_fire_perimeters_mttfirecal.shp",
                utc.zone=+1,
                nc.folder="~/meteorological_data",
                output.folder="~/get_fire_weather_meteorological_data")
```

Like in the previous function, the fire_weather_study_area.csv is saved the output folder (get_fire_weather_meteorological_data). This file is equal to the file generated by the function get_fire_weather.

6. Characterizing the fire regime and weather conditions

The next step after downloading and processing the meteorological variables associated with the occurrence of wildfires in the study area is the characterization of the study area. This is done by using the function *build_report*.

The build_report function creates files with meteorological characterization of the study area during fire events (csv and fms files); and a word file with the characterization of the fire regime.

First, one should evaluate the inputs required in the help section:

```
?build_report
```

```
build_report(study.area, my.fires, my.dated.fires, meteo.data, active.period,
  user.period, meteo.aggregation, min.size, min.overlap, output.folder,
  fire.aggregation, fire.size.intervals, manual.dur, create.clusters,
  summarize.per.fire, live.fuel.moisture)
```

The parameter study.area refers to the limits of the study area that was also used to download and process the weather data; the parameter my.fires refers to a shapefile containing the fire perimeters, which do not necessarily need to be dated; the parameter my.dated.fires is the dated fire perimeters used to download the weather data; the active.period defines the hours interval in which the weather variables should be used to characterize the fire propagation; the parameter meteo.aggregation refers to how hourly data should be combined into daily data and; min.size allows to subset the fire sample by setting a minimum fire size to be considered in the analysis; min.overlap refers to the minimal overlap (as percentage) required between the fire perimeters and the study area to consider the fire perimeter in the analysis; the output.folder is the path to the folder where the outputs of the function should be saved; the parameter fire.aggregation refers to how the meteorology of different ERA5-Land points inside the same fire perimeter should be combined; fire.size.intervals sets the fire size classes to be considered in the characterization of the study area; the parameter create.clusters allows the user to specify if weather clusters should be created or if the analysis should consider weather percentiles; and the parameter summarize.per.fire allows to summarize the weather conditions of each fire event in one vale (i.e. multi-days fires will only have one value for the weather conditions; only available if fire.aggregation = "WS"), and the parameter live.fuel.moisture allows the user to set the value for the herbaceous and woody live fuel moisture.

Like in the previous functions, the shapefile containing the fires (dated or not) can be at national or regional scale, since only the fire perimeters intersecting the study area will be considered.

To better understand the function, we can run:

```
build_report(study.area="~/study_area_mttfirecal.shp",
       my.fires="~/dated_fire_perimeters_mttfirecal.shp",
       my.dated.fires="~/dated_fire_perimeters_mttfirecal.shp",
       meteo.data="~/meteorological_data/fire_weather_study_area.csv",
       active.period="energy",
       meteo.aggregation="max.min",
       fire.aggregation="WS",
       min.size=100,
       min.overlap=50,
       fire.size.intervals=c(100, 200, 300, 400, 500, 600, 700, 800, 900, 1000,
2500, 5000, 10000),
       create.clusters=TRUE,
       live.fuel.moisture=c(60,90),
       output.folder="~/characterization")
```

The function returns the outputs listed below in Figure 8. The content in each file and folder is explained below.

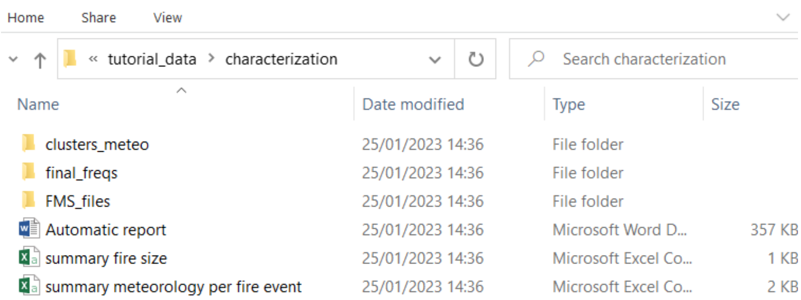


Figure 8 – Screenshot of outputs generated after running the function *build_report*.

The csv file “summary meteorology per fire event” contains the meteorological conditions per fire perimeter. This file is build using the values given by the user in the parameters active.period, meteo.aggregation, fire.aggregation, min.size and min.overlap. In this example, the fire events considered burned a minimum of 100 hectares with a minimum of 50% overlap with the study area. After this initial subset, only the hour-interval defined in the active.period is selected for fire perimeter. Then this data is grouped following the parameters in meteo.aggregation i.e., only the maximum value of temperature and wind speed and minimum relative humidity inside the interval of hours defined will be stored. If the fire perimeter intersects more than one ERA5-Land point, then only the point with the highest wind speed will be considered (as defined at fire.aggregation).

After this aggregation, only one value per weather variable, per day and per fire perimeter is stored and used for the grouping process (cluster classification or percentiles; Table 2).

Fire ID	*year*	*month*	*day*	*T*	*RH*	*WS*	*Fire size*
51	2001	8	7	33.32	29.64	26.8	1018
54	2001	9	8	30.65	39.17	15.53	1319
54	2001	9	9	30.87	29.26	11.61	1319

Table 2 – Example of the file summary meteorology per fire event.csv. Using the settings above will result in one value of temperature, relative humidity and wind speed being used for the weather aggregation process. Fire ID represents the unique id of each fire, T represents temperature in degrees Celsius, RH represents relative humidity in percentage, WS represents wind speed in kmh.

If one wishes to map the fire perimeters that will be used to characterize the typical weather conditions during fire spread, then:

```
#load and re-load the data
used_fire_perimeters <- read.csv("~/characterization/summary meteorology per fire event.csv")
my_dated_fires <- readOGR("~/dated_fire_perimeters_mttfirecal.shp")
my_study_area <- readOGR("~/study_area_mttfirecal.shp")

#subsetting the data
my_dated_fires_used <- my_dated_fires[my_dated_fires$ID %in% used_fire_perimeters$Fire.ID, ]


#plotting
world_st <- ne_countries(scale = "medium", returnclass = "sf")
WorldData <- map_data('world') %>% filter(region == "Portugal") %>% fortify
surroundings <- map_data('world') %>% fortify


my_study_area <- spTransform(my_study_area, CRS("+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"))
my_dated_fires_used <- spTransform(my_dated_fires_used, CRS("+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"))


ggplot() +
  coord_fixed(xlim = c(-9, -8), ylim = c(37, 37.6)) +
  suppressWarnings(geom_map(data = surroundings, map = surroundings,
                            aes(x = long, y = lat, group = group, map_id=region),
                            fill = "grey", colour = "black", size=0.5))+
  suppressWarnings(geom_map(data = WorldData, map = WorldData,
                            aes(x = long, y = lat, group = group, map_id=region),
                            fill = "antiquewhite", colour = "black", size=0.5))+
  suppressMessages(geom_polygon(data = my_study_area, aes(long, lat),
                                colour = alpha("darkred", 1/2), size = 0.7,
fill=NA,linetype = "dashed"))+
```

```
  suppressMessages(geom_polygon(data = my_dated_fires_used, aes(long, lat,
group=group),
                                colour = alpha("darkred", 1/2), size = 0.7,
fill=NA))+
  theme(panel.grid.major = element_line(color = "gray", linetype = "dashed", size
= 0.5),
        panel.background = element_rect(fill = "aliceblue"))+
  xlab("Longitude") + ylab("Latitude")
```

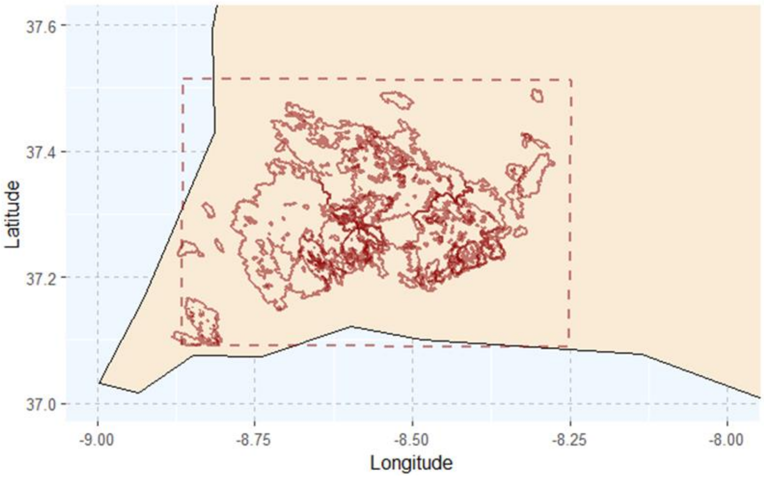


Figure 9 – Study area with the fire perimeters used in weather aggregation process (i.e. fire perimeters that met the settings above and that are dated).

6.1. The report generated

One of the outputs generated from the *build_report* function is the Automatic report. This report is essential to characterize the study area. It is divided into the following sections:

- **Technical details**
- **Conditions of the analysis**, where it is recorded the user's choices in the parametrization of active.period, min.size and min.overlap.
- **Fire regime analysis**, where the study area is characterized in terms of annual burned area, fire size distribution and burned area due to large fires.
- **Fire weather analysis**, where the weather grouping (clustering or percentiles) is shown
- **Number of durations to consider**, where the classes of duration to use in the fire spread simulations are identified.

In this tutorial we briefly explain the most important figures, tables and analysis.

Figure 10 shows the fire size distribution per class of fire size. Note that these fire size classes do not correspond to the classes set by the users. On contrary, it is meant to assist the user in evaluate if the fire size classes set are able to characterize the observed fire regime. For the same reason, it does not reflect the min.size set by the user.

The Figure 10 is only dependent of the parameter min.overlap. Note that this figure is merely illustrative and is not used elsewhere in the MTTfireCAL package.

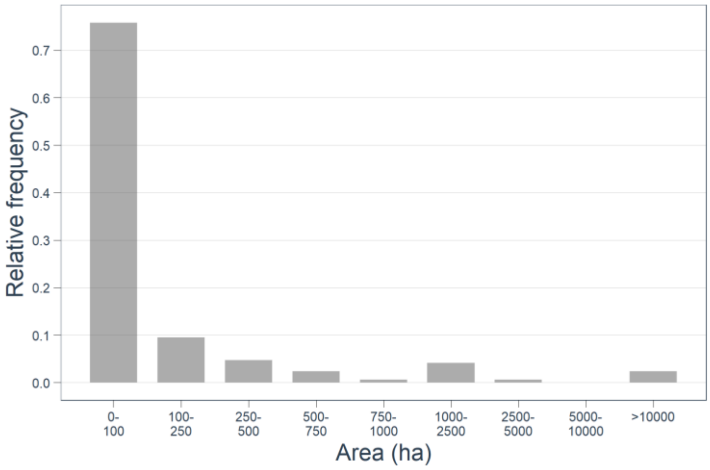


Figure 10 – Historical fire size distribution.

Figure 11 represents distribution of annual burned area over the period of analysis.
Like in the previous analysis, the annual burned area only depends on the parameter min.overlap set by the user. This figure is useful to highlight the years with high incidence of fire events.

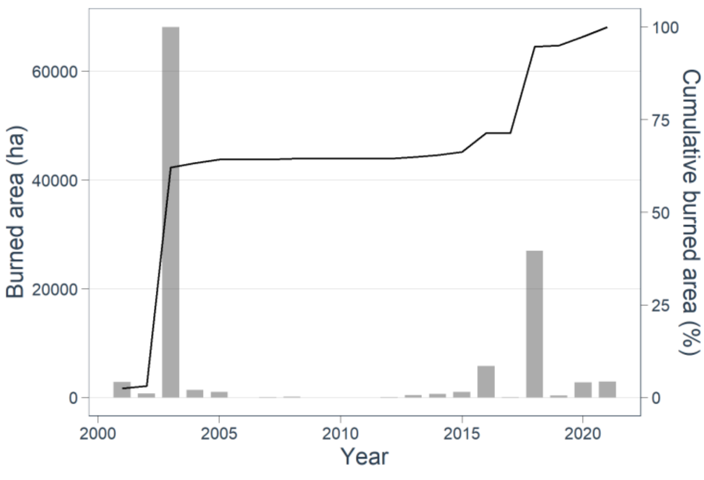


Figure 11 – Historical distribution of burned area in the period of analysis (2001-2021; histogram) and the cumulative burned area as percentage (black line).

Figure 12 illustrates the contribution of burned area in each fire size class to the overall burned area. For instance, fires that burned less than 100 ha contributed with 2% to the total burned area; fires that burned between 100 and 500 ha contributed with 5% to the total burned area, etc. This figure may be helpful when deciding which fire size classes are more important to represent. In this example, fires larger than 1000 hectares account with 90% to the bulk of burned area. For this reason, it is extremely important to represent those fire in the study area.

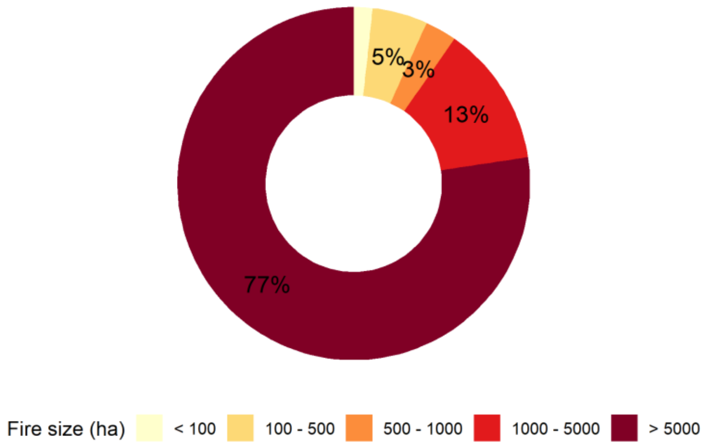


Figure 12 - Contribution of different fire size classes to the bulk of burned area.

6.1.1. Number of durations to consider

The last figure in the automatic report highlights the classes of duration set to be used for the calibration of the model. Each vertical red line represents a different duration class that was automatically created. Figure 13 shows the example in this tutorial.

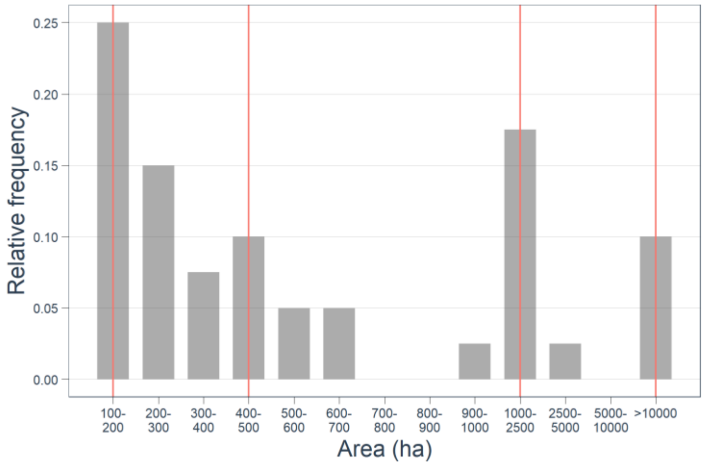


Figure 13 – Historical fire size distribution with the identification of the suggested duration classes (red vertical lines).

In the example shown, there are four duration classes identified:

- the first duration class (or simply called duration_1) starts in the fire size class 100-200 hectares and ends in the fire size class 300-400 hectares
- the second duration class (duration_2) starts in the fire size class 400-500 hectares, and ends in the fire size class 900-1000 hectares
- the third duration class (duration_3) starts in the fire size class 1000-2500 hectares, and ends in the fire size class 5000-10000 hectares
- the forth duration class (duration_4) starts in the fire size class >10,000.

Hence, the duration_1 will be responsible to capture the fire size distribution of fires with burned area between 100 and 400 hectares; duration_4 will represent fire with a size between 400 and 1000 hectares; duration_3 will be responsible to capture the fire size distribution of fires with burned area between 1000 and 10,000 hectares; and duration_4 will be responsible to capture the fire size distribution of fires with burned area larger than 10,000 hectares.

Because the relative frequency inside each interval that the duration classes will characterize is different, this implies that the duration classes will have different relative frequencies (i.e. some fire size classes are more common than others). This is essential to capture the overall fire size distribution. The total relative frequency of each duration class results of the sums of individual fire size classes that are characterized by the duration class. For example:

$$RF\ duration_1 = RF_{[100-200]} + RF_{[200-300]} + RF_{[300-400]}$$

Calculating this relative frequency for all the duration will result in the following distribution of frequencies:

Duration class	Relative frequency
Duration_1	0.475
Duration_2	0.225
Duration_3	0.2

Duration_4	0.1

Table 3 – Relative frequency of each duration class defined in Figure 13.

In practical terms, this means that when simulating fires in the landscape, we will simulate ca. five times more fires of the duration class 1 than of the duration class 4. Or, in other words, our simulations will have five times more fires with size between 100 and 400 hectares than fires with size over 10,000 hectares.

6.1.2. Manually choosing the duration classes to be used

The example shown above represents a fair automatic classification of duration classes. However, sometimes the result of the automatic definition of duration classes is not what one might expect or the user want to set specific duration classes. Manually defined duration classes are easily obtained by using the parameter manual.dur. To use the manual.dur parameter, the user needs to specify the lowest limit of the fire size class where the duration class should be defined. For instance

```
manual.dur=c(100,1000,10000)
```

will create three duration classes: the duration_1 will characterize fires with sizes between 100 and 1000 hectares; the duration_2 will characterize fires sizes between 1000 and 10,000 hectares; and the duration_3 will characterize fires larger than 10,000 hectares. The use of manual.dur will override the automatic definition of duration classes.

Let's try this setting:

```
build_report(study.area="~/study_area_mttfirecal.shp",
             my.fires="~/dated_fire_perimeters_mttfirecal.shp",
             my.dated.fires="~/dated_fire_perimeters_mttfirecal.shp",
             meteo.data="~/meteorological_data/fire_weather_study_area.csv",
             active.period="energy",
             meteo.aggregation="max.min",
             fire.aggregation="WS",
             min.size=100,
             min.overlap=50,
             fire.size.intervals=c(100, 200, 300, 400, 500, 600, 700, 800, 900,
1000, 2500, 5000, 10000),
             create.clusters=TRUE,
             manual.dur=c(100,1000,10000),
             live.fuel.moisture=c(60,90),
             output.folder="~/characterization_manual_dur")
```

Note that the output.folder is now different. You can choose to overwrite the files in the previous folder. Here, we chose to save them in a different folder to allow for comparisons between using and not using the manual.dur.

After running the build_report function with the manual.dur parameter, we can examine the Automatic report file. The only figure that is different from the previous report is the last figure showing the duration classes to use (compare Figure 13 and Figure 14).

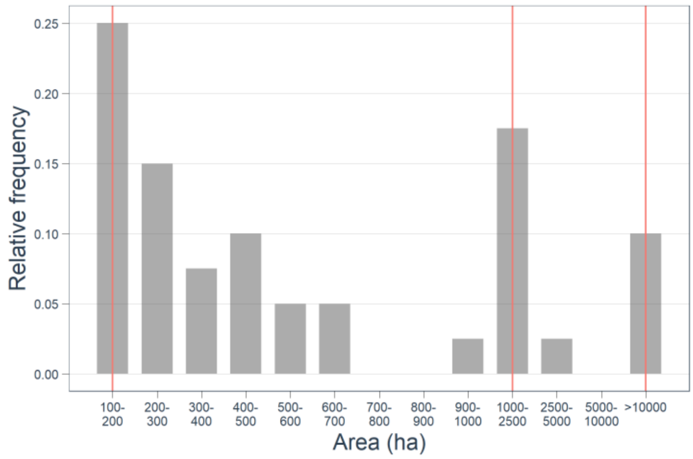


Figure 14 – Historical fire size distribution with the identification of the manually inserted duration classes (red vertical lines).

As explained above, we have now defined four duration classes:

- the first duration class (or simply called duration_1) starts in the fire size class 100-200 hectares and ends in the fire size class 900-1000 hectares
- the second duration class (duration_2) starts in the fire size class 1000-2500 hectares, and ends in the fire size class 5000-10000 hectares
- the third duration class (duration_3) starts in the fire size class >10,000.

This change will also imply changes in the relative frequency of the durations, resulting in the following distribution of frequencies:

Duration class	Relative frequency
Duration_1	0.7
Duration_2	0.2
Duration_3	0.1

Table 4 - Relative frequency of each duration class defined in Figure 14.

Note that the relative duration that represent fire sizes larger than 10,000 hectares is, naturally, the same between the two methods (Duration 3 in Table 4 and Duration 4 in Table 3).

6.1.3. Exploring the clusters_meteo folder

As shown in Figure 8, some folders are created after using the build_report function. One of those folders is the clusters_meteo. This folder has several csv files describing the weather conditions in each cluster. For each clustering classification there are two files: kmeans_N_clusters or model_based_clustering and kmeans_N_clusters_wind_directions or model_based_clustering_wind_directions. Taking the kmeans clustering classification with 2 clusters as example:

cluster	T	RH	WS	Cluster size	Cluster RF
1	27	43	26	26	0.31
2	34	24	18	58	0.69

Table 5 – Characterization of weather variables after the aggregation process (clustering in this case). The column cluster refers to the unique id of the cluster, T refers to temperature in degrees Celsius, RH refers to the relative humidity, WS refers to the wind speed in kmh, Cluster size refers to the number of observation in each cluster, and Cluster RF refers to the relative frequency of each cluster.

In this example, the cluster 1 is characterized by a temperature of 27ºC, 43% of relative humidity and a wind speed of 26 kmh. Cluster 2 is characterized by a temperature of 34ºC, 24% of relative humidity and 18 kmh of wind speed. Cluster 2 is more than two times more frequent than cluster 1, with a relative frequency of 0.69 and 0.31, respectively.

Cluster_id_km_2	WD_letter	Freq
1	N	0.026
	NE	0.003
	E	0.001
	SE	0.052
	S	0.024
	SW	0.017
	W	0.041
	NW	0.146
2	N	0.054
	NE	0.009
	E	0.034
	SE	0.107
	S	0.041
	SW	0.049
	W	0.077
	NW	0.319
	Total	**1**

Table 6 – Characterization of the wind direction in each weather cluster. Cluster_id_km_2 refers to the cluster id, WD_letter refers to the wind direction and Freq refers to the relative frequency of the wind direction and the cluster id.

Table 5 shows the weather variables and the relative frequency for the two clusters considered; while Table 6 shows the relative frequency of the different wind directions in each cluster. Note that the relative frequency of the different wind directions is calculated individually for each cluster (e.g. wind blowing from southeast is the second most common wind direction in the cluster 2, but is infrequent when weather conditions fall in cluster 2).

Both files are only informative and are not used anywhere else in MTTfireCAL.

6.1.4. Exploring the final_freqs folder

In the final_freqs folder, one csv file is stored for each clustering classification. Like in the previous example, we will use the file "kmean_2_clusters_final_freqs.csv" to explain its content. Here we used the files generated in section 6, i.e., with the automatic duration classes.

				Duration				
cluster_id_km_2	**WD_use**	**WD_letter**	**Freq**	**1**	**2**	**3**	**4**	**Total cluster**
1	0	N	0.026	0.013	0.006	0.005	0.003	0.31
	45	NE	0.003	0.001	0.001	0.001	0.000	
	90	E	0.001	0.001	0.000	0.000	0.000	
	135	SE	0.052	0.025	0.012	0.010	0.005	
	180	S	0.024	0.011	0.005	0.005	0.002	
	225	SW	0.017	0.008	0.004	0.003	0.002	
	270	W	0.041	0.019	0.009	0.008	0.004	
	315	NW	0.146	0.069	0.033	0.029	0.015	
2	0	N	0.054	0.026	0.012	0.011	0.005	0.69
	45	NE	0.009	0.004	0.002	0.002	0.001	
	90	E	0.034	0.016	0.008	0.007	0.003	
	135	SE	0.107	0.051	0.024	0.021	0.011	
	180	S	0.041	0.019	0.009	0.008	0.004	
	225	SW	0.049	0.023	0.011	0.010	0.005	
	270	W	0.077	0.036	0.017	0.015	0.008	
	315	NW	0.319	0.151	0.072	0.064	0.032	
		Total	**1**	**0.475**	**0.225**	**0.2**	**0.1**	

Table 7 – Example of the kmean_2_clusters_final_freqs.csv file. The column cluster_id_km_2 represents the cluster id, WD_use represents the wind direction in degrees, WD_letter represents the wind direction, Freq represents the frequency of the cluster with the wind direction, Duration refers to the different duration classes and contains the relative frequency of each duration class in each cluster and with the different wind direction, Total cluster refers to the relative frequency of each cluster.

Table 7 shows a combination of Table 4, Table 5 and Table 6. This file will be used when generating the ignitions.

6.1.5. Exploring the FMS_files folder

The last folder that is created after running build_report is the FMS_files. This folder contains several folders inside, one for each clustering classification. Once again we will use kmeans_2_clusters as an example to explain the content of the FMS files.

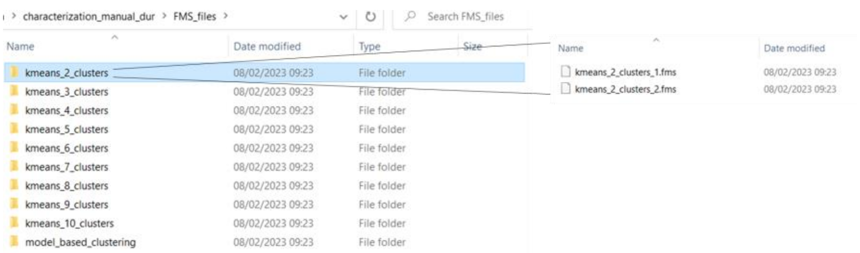


Figure 15 – Screenshot of the FMS folder created after using the function build_report

There are two files inside the folder kmeans_2_clusters, one per cluster. These files have the fuel moisture files for each cluster. Although these are fms format files, they are essentially text files that can be opened with any text editor (here we use Notepad++).

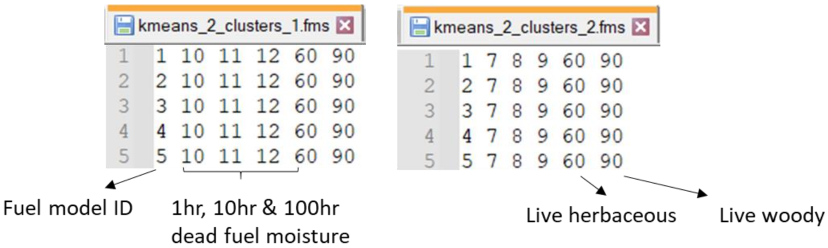


Figure 16 – Example of the FMS file for the two clusters considered. Since cluster 2 has a higher temperature and lower relative humidity than cluster 1 (see Table 5), it shows lower dead fuel moisture.

The dead fuel moisture is calculated for all fuel models following the equations in Anderson et al. (2015) and Nelson (2000), while the live fuel moisture is directly set by the user in the parameter live.fuel.moisture.

6.1.6. A note of caution

When automatically setting the duration classes in the build_report function, the user should be aware that this not always produces the expected outcomes. A clear example is shown in Figure 13. The example shown in Figure 17 results from the application of MTTfireCAL to a different study area.

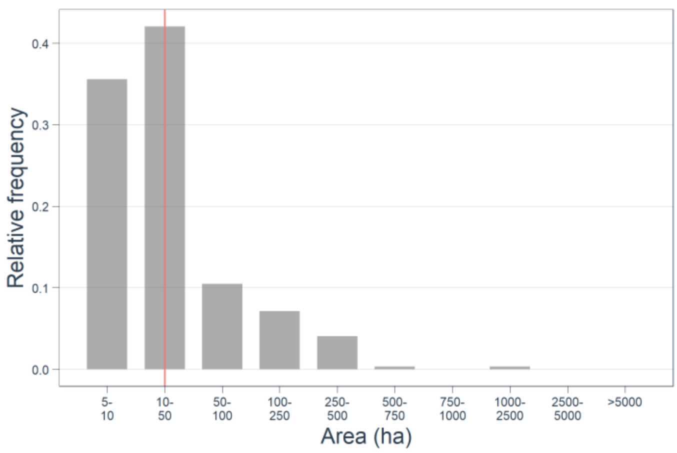


Figure 17 – Example of the historical fire size distribution of a different study area, with the automatic duration class suggested (red vertical line).

As shown, the automatic detection of duration classes only identifies one duration for the entire fire size distribution. It is highly unlikely that a duration value capable of replicating the peaks in the fire sizes intervals of 5 to 10 hectares is capable of producing fires larger than, for example, 250 hectares. Hence, in this situation, the user must adjust the duration classes manually, by using the parameter manual.dur. The user could, for instance, set the manual.dur as:

```
manual.dur=c(5,100,10000)
```

6.2. Evaluate the influence of the parameterization

The parameterization in the function *build_report* has a great importance in the quality of the calibration and in the outputs from MTT. Hence, it is important to fully understand the implications of the settings before continuing with the tutorial.

In this section we will change the parameters `active.period`, `meteo.aggregation`, `fire.aggregation`, `min.size`, `min.overlap`, and `fire.size.intervals`. These are the most subjective parameters in the function.

The comparisons were made using the definitions in section 6, this is:

```
build_report(study.area="~/study_area_mttfirecal.shp",
       my.fires="~/dated_fire_perimeters_mttfirecal.shp",
       my.dated.fires="~/dated_fire_perimeters_mttfirecal.shp",
       meteo.data="~/meteorological_data/fire_weather_study_area.csv",
       active.period="energy",
       meteo.aggregation="max.min",
       fire.aggregation="WS",
       min.size=100,
       min.overlap=50,
       fire.size.intervals=c(100, 200, 300, 400, 500, 600, 700, 800, 900, 1000,
2500, 5000, 10000),
       create.clusters=TRUE,
       output.folder="~/characterization")
```

6.2.1. Parameter active.period

As stated above, the parameter active.period defines the hours interval in which the weather variables should be used to characterize the typical fire propagation. This means that only the hours that fall inside the interval of time defined in this parameter will be used in the weather aggregation. There are essentially two option in this parameter: either setting it to “energy” or defined it manually. The “energy” setting is the one used in this tutorial and is based on the average energy released during fire spread, which peaks from 12h to 20h in Portugal (see Aparício et al., 2023). The user may choose to specify the period of hours to be used in the aggregation of fire weather by setting the `active.period = c(11,13)`. By doing so, only the values for the hours 11, 12, and 13 are used in the weather aggregation. Table 8 highlights the difference in both periods.

Fire_ID	Day	Hour	Temperature	Relative humidity	Wind speed
51	2001-08-06	11	30	38	5
		12	32	34	2
		13	33	31	4
		14	34	30	7
		15	34	29	11
		16	34	30	15
		17	33	32	18
		18	32	34	18
		19	31	38	18
		20	29	43	17
		21	27	48	16
		22	26	50	15

Table 8 – Implications of choosing different active.period. Green rows represent the user defined active.period (between 11 and 13 hours) and the red rows represent the energy defined active.period (between 12 and 20 hours).

Fire_ID	Day	Hour	Temperature	Relative humidity	Wind speed
51	2001-08-06	11	30	38	5
		12	32	34	2
		13	33	31	4
		14	34	30	7
		15	34	29	11
		16	34	30	15
		17	33	32	18
		18	32	34	18
		19	31	38	18
		20	29	43	17
		21	27	48	16
		22	26	50	15

Commented [a1]: a tabela fica aqui porque pode ser necessário alterar o período energy

To better understand the influence of the setting of active.period, we can use the function build_report again, by changing only the active.period and the output.folder (to prevent overlapping of results). We created a folder named changing_parameters and the subfolder changing_active_period inside.

```
build_report(study.area="~/study_area_mttfirecal.shp",
       my.fires="~/dated_fire_perimeters_mttfirecal.shp",
       my.dated.fires="~/dated_fire_perimeters_mttfirecal.shp",
       meteo.data="~/meteorological_data/fire_weather_study_area.csv",
       active.period=c(11,13),
       meteo.aggregation="max.min",
       fire.aggregation="WS",
       min.size=100,
       min.overlap=50,
       fire.size.intervals=c(100, 200, 300, 400, 500, 600, 700, 800, 900, 1000,
2500, 5000, 10000),
       create.clusters=TRUE,
       output.folder="~/changing_parameters/changing_active_period")
```

Now we can compare the centroid of clusters to confirm that the setting of this parameter indeed influences the weather aggregation process. Here we use the kmeans with 2 clusters. Table 9 shows the comparison.

cluster	T	RH	WS	Cluster size	Cluster RF	active.period
1	27	43	26	26	0.31	energy
2	34	24	18	58	0.69	energy
1	27	42	21	35	0.42	11-13
2	34	25	13	49	0.58	11-13

Table 9 – Implications of choosing different active.period for the clusters' values. Green rows represent the user defined active.period (between 11 and 13 hours) and the red rows represent the energy defined active.period (between 12 and 20 hours).

6.2.2. Parameter meteo.aggregation

The parameter meteo.aggregation refers to how hourly data should be combined into daily data. There are several options: "mean" (calculates the mean of each variable for the active.period), "max.min" (calculates the maximum value of temperature and wind speed and the minimum relative humidity for the active.period), "none" (does not combine hourly data into daily data and uses all the data in the creation of clusters or percentiles). The option max.min often results in more severe weather conditions.

Like in the previous case, we can compare the influence of the different types of meteorological aggregation. Here, we will compare the "max.min" (calculated above) with the "none" using the active.period defined by "energy".

```
build_report(study.area="~/study_area_mttfirecal.shp",
       my.fires="~/dated_fire_perimeters_mttfirecal.shp",
       my.dated.fires="~/dated_fire_perimeters_mttfirecal.shp",
       meteo.data="~/meteorological_data/fire_weather_study_area.csv",
       active.period="energy",
       meteo.aggregation="none",
       fire.aggregation="WS",
       min.size=100,
       min.overlap=50,
       fire.size.intervals=c(100, 200, 300, 400, 500, 600, 700, 800, 900, 1000,
2500, 5000, 10000),
       create.clusters=TRUE,
       output.folder="~/changing_parameters/changing_meteo_aggregation")
```

Table 10 shows the comparison between the two settings. It is clear the increase in the size of the clusters. This is the result of not aggregating the meteorological data, and hence using all the hours inside the defined active.period (i.e. from 12 to 20 for each day of the fire duration). The reader may also notice that the values of the meteorological variables become slightly less severe when using meteo.aggregation = "none".

cluster	T	RH	WS	Cluster size	Cluster RF	meteo.aggregation
1	27	43	26	26	0.31	max.min
2	34	24	18	58	0.69	
1	33	27	13	443	0.59	none
2	26	48	22	313	0.41	

Table 10 - Implications of choosing different meteo.aggregation for the clusters' values. Green rows represent the none defined meteo.aggregation (all the hours inside the active.period are considered) and the red rows represent the max.min defined meteo.aggregation (only the value of highest temperature and wind speed and lowest relative humidity inside the active.period is considered).

6.2.3. Parameter fire.aggregation

The parameter fire.aggregation refers to how the meteorology of different points from the ERA5-Land grid that fall inside of the same fire perimeter should be combined. Possible methods include: WS (wind speed; the ERA5-Land grid point falling inside the fire perimeter with highest wind speed is kept and the others ignored), none (uses all the ERA5-Land grid points falling inside the fire perimeter).

To compare both options, we need to set the fire.aggregation as "none" and use the build_report function again (fire.aggregation = "WS" was used above).

```
build_report(study.area="~/study_area_mttfirecal.shp",
       my.fires="~/dated_fire_perimeters_mttfirecal.shp",
       my.dated.fires="~/dated_fire_perimeters_mttfirecal.shp",
       meteo.data="~/meteorological_data/fire_weather_study_area.csv",
       active.period="energy",
       meteo.aggregation="max.min",
       fire.aggregation="none",
       min.size=100,
       min.overlap=50,
       fire.size.intervals=c(100, 200, 300, 400, 500, 600, 700, 800, 900, 1000,
2500, 5000, 10000),
       create.clusters=TRUE,
       output.folder="~/changing_parameters/changing_fire_aggregation")
```

Table 11 shows the changes in the clusters when using fire.aggregation = "WS" or fire.aggregation = "none". Note the change in the clusters' size between the two settings, following the inclusion of all grid points inside the fire perimeters. In this particular case, using fire.aggregation = "none" will result in clusters with higher temperature and lower relative humidity, and lower wind speed than using fire.aggregation = "WS".

cluster	T	RH	WS	Cluster size	Cluster RF	fire.aggregation
1	27	43	26	26	0.31	WS
2	34	24	18	58	0.69	
1	35	22	16	69	0.64	none
2	28	39	25	39	0.36	

Table 11 - Implications of choosing different fire.aggregation for the clusters' values. Green rows represent the none defined fire.aggregation (all the ERA5-Land gridpoints inside the fire perimeter are considered) and the red rows represent the WS defined fire.aggregation (only the ERA5-Land gridpoint with the highest value of wind speed is considered).

Important note: Setting meteo.aggregation and fire.aggregation to other than “none” allows faster clustering and percentile calculation. Moreover, in study areas with high recurrence of wildfires, the fire weather file (fire_weather_study_area.csv) will have hundreds of thousands of hourly data. In such cases, the meteo.aggregation and fire.aggregation must be set to other than “none”, or he clustering option will fail.

6.2.4. Parameter min.size

The parameter min.size refers to the minimum fire size in hectares that should be considered for the characterization of the historical fire regime. All fires with a burned area below the value set in min.size will be ignored in the characterization of both the fire size distribution and the meteorological aggregation. This parameter is particularly useful to ignore small fires in the study area that might not be wildland fires (e.g. might be related with agricultural activities in Mediterranean countries). We can test the influence of this parameter by setting min.size to 10 and compare it against the outputs generated above, with min.size as 100.

```
build_report(study.area="~/study_area_mttfirecal.shp",
       my.fires="~/dated_fire_perimeters_mttfirecal.shp",
       my.dated.fires="~/dated_fire_perimeters_mttfirecal.shp",
       meteo.data="~/meteorological_data/fire_weather_study_area.csv",
       active.period="energy",
       meteo.aggregation="max.min",
       fire.aggregation="WS",
       min.size=10,
       min.overlap=50,
       fire.size.intervals=c(10,100, 200, 300, 400, 500, 600, 700, 800, 900, 1000,
2500, 5000, 10000),
       create.clusters=TRUE,
       output.folder="~/changing_parameters/changing_min_size")
```

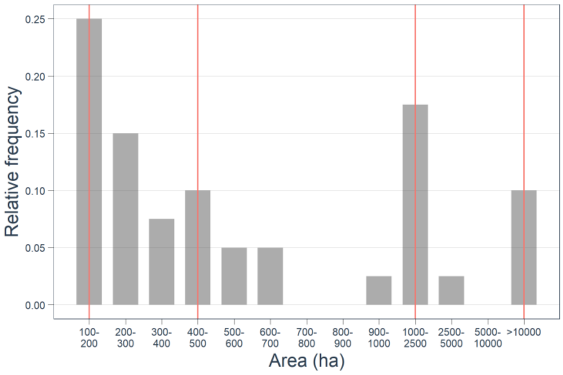
Table 12 shows the changes in the clusters when using min.size = 100 or min.size= 10. Note the change in the clusters’ size between the two settings, following the inclusion of all grid points inside the fire perimeters. Following the change in the min.size parameter, small changes can also be observed in the values of the centroids in each cluster.

cluster	T	RH	WS	Cluster size	Cluster RF	min.size
1	27	43	26	26	0.31	100
2	34	24	18	58	0.69	
1	36	22	17	76	0.57	10
2	27	41	24	58	0.43	

Table 1 – Implications of choosing different min.size for the clusters’ values. Green rows represent min.size as 10 hectares (only fires that burn at least 10 hectares are considered) and the red rows represent min.size as 100 hectares (only fires that burn at least 100 hectares are considered).

Besides the changes in the clustering process, the most relevant modifications are in the fire size distribution that is used to calibrate the MTT (Figure 18).

a)



b)

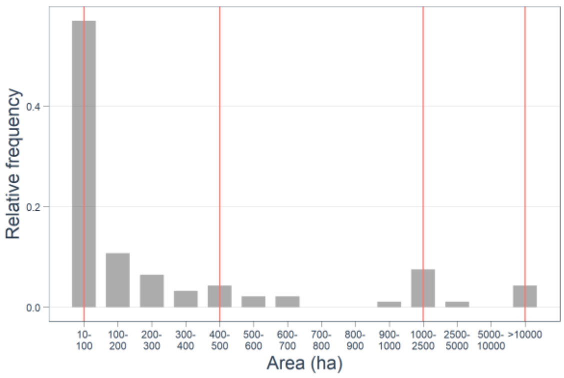


Figure 18 – Illustration of the fire size distribution with the automatic duration classes for two min.size: a) min.size of 100 hectares, and b) min.size of 10 hectares

Naturally, the change in the fire size distribution and in the duration classes detected have a great influence in the relative frequency of each class (Table 13).

Duration class	Relative frequency	min.size
Duration_1	0.475	100
Duration_2	0.225	
Duration_3	0.2	
Duration_4	0.1	
Duration_1	0.774	10
Duration_2	0.97	
Duration_3	0.086	
Duration_4	0.043	

Table 13 - Implications of choosing different min.size for the relative frequency of each duration class. Green rows represent min.size as 10 hectares (only fires that burn at least 10 hectares are considered) and the red rows represent min.size as 100 hectares (only fires that burn at least 100 hectares are considered).

The best way to understand the changes in meteorological aggregation and in the fire size distribution is to plot the fire perimeters that are included in both analyses. First, we will replicate the Figure 9; then, we will add the new fire perimeters that were not accounted before (with min.size = 100) and will be accounted now (with min.size = 10) with blue colour.

We can start by plotting the changes in the fire perimeters considered for the meteorological aggregation (i.e. the ones that are dated).

```
#load and re-load the data
used_fire_perimeters <- read.csv("~/characterization/summary meteorology per fire
event.csv")
my_dated_fires <- readOGR("~/dated_fire_perimeters_mttfirecal.shp")
my_study_area <- readOGR("~/study_area_mttfirecal.shp")

#subsetting the data
my_dated_fires_used <- my_dated_fires[my_dated_fires$ID %in%
used_fire_perimeters$Fire.ID, ]


#plotting
world_st <- ne_countries(scale = "medium", returnclass = "sf")
WorldData <- map_data('world') %>% filter(region == "Portugal") %>% fortify
surroundings <- map_data('world') %>% fortify


my_study_area <- spTransform(my_study_area, CRS("+proj=longlat +datum=WGS84
+no_defs +ellps=WGS84 +towgs84=0,0,0"))
my_dated_fires_used <- spTransform(my_dated_fires_used, CRS("+proj=longlat
+datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"))


ggplot() +
  coord_fixed(xlim = c(-9, -8), ylim = c(37, 37.6)) +
  suppressWarnings(geom_map(data = surroundings, map = surroundings,
                            aes(x = long, y = lat, group = group, map_id=region),
                            fill = "grey", colour = "black", size=0.5))+
  suppressWarnings(geom_map(data = WorldData, map = WorldData,
                            aes(x = long, y = lat, group = group, map_id=region),
                            fill = "antiquewhite", colour = "black", size=0.5))+
  suppressMessages(geom_polygon(data = my_study_area, aes(long, lat),
                                colour = alpha("darkred", 1/2), size = 0.7,
fill=NA,linetype = "dashed"))+
  suppressMessages(geom_polygon(data = my_dated_fires_used, aes(long, lat,
group=group),
                                colour = alpha("darkred", 1/2), size = 0.7,
fill=NA))+
  theme(panel.grid.major = element_line(color = "gray", linetype = "dashed", size
= 0.5),
        panel.background = element_rect(fill = "aliceblue"))+
  xlab("Longitude") + ylab("Latitude")
```

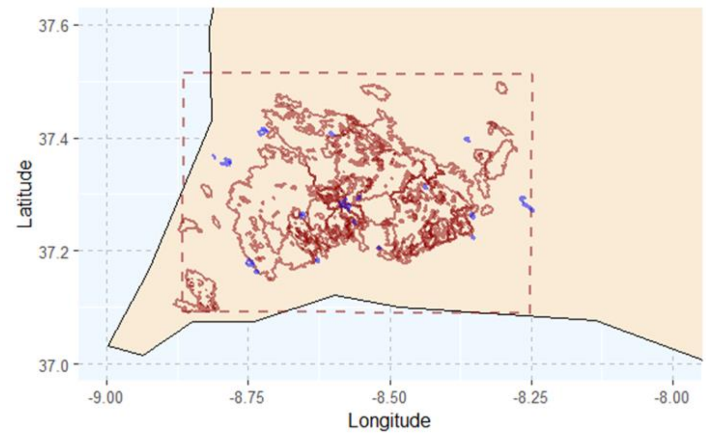


Figure 19 - Red perimeters represent the dated fire events that were considered for the meteorological aggregation when min.size = 100; blue perimeters represent the dated perimeters added to the meteorological aggregation when considering min.size = 10

As shown in the Figure 19, there are several fire perimeters below 100 hectares that are considered for the meteorological aggregation when min.size = 10 hectares (shown in blue). In total, 21 fire perimeters smaller than 100 hectares that are dated.

We can repeat this analysis but plot all the fire perimeters that are used to create the fire size distribution, i.e., independently if they are dated or not.

```
#plot all the fires that are considered in the fire size distribution analisys
used_fire_perimeters <- read.csv("~/changing_parameters/changing_min_size/summary
fire size.csv")
my_dated_fires <- readOGR("~/dated_fire_perimeters_mttfirecal.shp")
my_study_area <- readOGR("~/study_area_mttfirecal.shp")

#subsetting the data
all_my_fires_used <- my_dated_fires[my_dated_fires$ID %in%
used_fire_perimeters$ID, ]


#plotting
world_st <- ne_countries(scale = "medium", returnclass = "sf")
WorldData <- map_data('world') %>% filter(region == "Portugal") %>% fortify
surroundings <- map_data('world') %>% fortify


my_study_area <- spTransform(my_study_area, CRS("+proj=longlat +datum=WGS84
+no_defs +ellps=WGS84 +towgs84=0,0,0"))
all_my_fires_used <- spTransform(all_my_fires_used, CRS("+proj=longlat
+datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"))

all_my_fires_used$Area_ha <- as.numeric(all_my_fires_used$Area_ha)

all_my_fires_used_before <- subset(all_my_fires_used,Area_ha > 100)
```

```
all_my_fires_used_new <- subset(all_my_fires_used,Area_ha <= 100)
```

```
ggplot() +
  coord_fixed(xlim = c(-9, -8), ylim = c(37, 37.6)) +
  suppressWarnings(geom_map(data = surroundings, map = surroundings,
                            aes(x = long, y = lat, group = group, map_id=region),
                            fill = "grey", colour = "black", size=0.5))+
  suppressWarnings(geom_map(data = WorldData, map = WorldData,
                            aes(x = long, y = lat, group = group, map_id=region),
                            fill = "antiquewhite", colour = "black", size=0.5))+
  suppressMessages(geom_polygon(data = my_study_area, aes(long, lat),
                                colour = alpha("darkred", 1/2), size = 0.7,
fill=NA,linetype = "dashed"))+
  suppressMessages(geom_polygon(data = all_my_fires_used_before, aes(long, lat,
group=group),
                                colour = alpha("darkred", 1/2), size = 0.7,
fill=NA))+
  suppressMessages(geom_polygon(data = all_my_fires_used_new, aes(long, lat,
group=group),
                                colour = alpha("blue", 1/2), size = 0.7,
fill=NA))+
  theme(panel.grid.major = element_line(color = "gray", linetype = "dashed", size
= 0.5),
        panel.background = element_rect(fill = "aliceblue"))+
  xlab("Longitude") + ylab("Latitude")
```

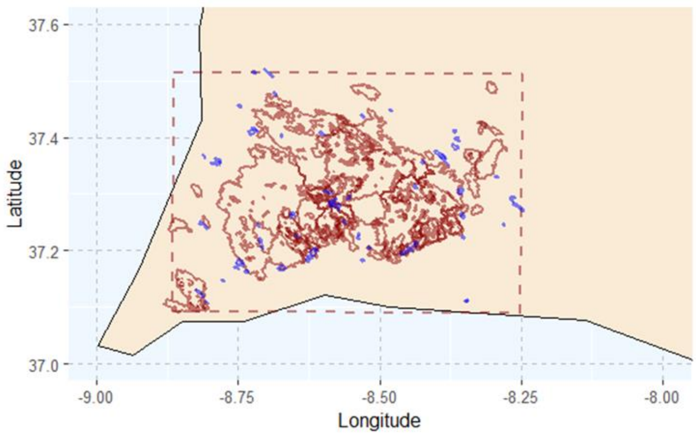


Figure 20 – Red perimeters represent the fire events that were considered for the fire size distribution when min.size = 100; blue perimeters represent the perimeters added to the fire size distribution analysis when considering min.size = 10.

By comparing the Figure 19 and Figure 20, it is clear that in this case the adjustment of min.size will produce greater changes in the fire size distribution than in the meteorological aggregation, since the majority of the added fire perimeters are not dated. In total, 61 fire perimeters (dated or undated) are added to the fire size distribution when min.size is set to 10 hectares.

6.2.5. Parameter min.overlap

The parameter min.overlap represents the minimal overlap (as percentage) between the fire perimeters and the study area so that the fire is considered in the analysis. Low values of min.overlap will include more fire events in the meteorological aggregation and in the historical fire size distribution.

First, we need to use the function build_report once again and change only the parameter min.overlap to 0.1. This value means that all fire perimeters that overlap the study-area in at least 0.1% will be considered for the entire analysis.

```
build_report(study.area="~/study_area_mttfirecal.shp",
      my.fires="~/dated_fire_perimeters_mttfirecal.shp",
      my.dated.fires="~/dated_fire_perimeters_mttfirecal.shp",
      meteo.data="~/meteorological_data/fire_weather_study_area.csv",
      active.period="energy",
      meteo.aggregation="max.min",
      fire.aggregation="WS",
      min.size=100,
      min.overlap=0.1,
      fire.size.intervals=c(100, 200, 300, 400, 500, 600, 700, 800, 900, 1000,
2500, 5000, 10000),
      create.clusters=TRUE,
      output.folder="~/changing_parameters/changing_min_overlap")
```

Like in the previous cases, it is useful to plot the fire perimeters that are used when min.overlap = 50 (which was set previously) and min.overlap = 0.1.

```
#plot all the fires that are considered in the fire size distribution analisys
library(rgdal)
library(rnaturalearth)
library(sf)
library(ggplot2)
library(dplyr)
library(tidyverse)

used_fire_perimeters_overlap0_1 <-
read.csv("~/changing_parameters/changing_min_overlap/summary fire size.csv")
my_dated_fires <- readOGR("~/dated_fire_perimeters_mttfirecal.shp")
my_study_area <- readOGR("~/study_area_mttfirecal.shp")

#subsetting the data
all_my_fires_used_overlap0_1 <- my_dated_fires[my_dated_fires$ID %in%
used_fire_perimeters_overlap0_1$ID, ]


#plotting
world_st <- ne_countries(scale = "medium", returnclass = "sf")
WorldData <- map_data('world') %>% filter(region == "Portugal") %>% fortify
surroundings <- map_data('world') %>% fortify


my_study_area <- spTransform(my_study_area, CRS("+proj=longlat +datum=WGS84
+no_defs +ellps=WGS84 +towgs84=0,0,0"))
```

```
all_my_fires_used_overlap0_1 <- spTransform(all_my_fires_used_overlap0_1,
CRS("+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"))


#load the min.overlap=50

used_fire_perimeters_overlap50 <- read.csv("~/characterization_manual_dur/summary
fire size.csv")
my_dated_fires <- readOGR("~/dated_fire_perimeters_mttfirecal.shp")
my_study_area <- readOGR("~/study_area_mttfirecal.shp")

#subsetting the data
all_my_fires_used_overlap50 <- my_dated_fires[my_dated_fires$ID %in%
used_fire_perimeters_overlap50$ID, ]

#plotting
world_st <- ne_countries(scale = "medium", returnclass = "sf")
WorldData <- map_data('world') %>% filter(region == "Portugal") %>% fortify
surroundings <- map_data('world') %>% fortify

my_study_area <- spTransform(my_study_area, CRS("+proj=longlat +datum=WGS84
+no_defs +ellps=WGS84 +towgs84=0,0,0"))
all_my_fires_used_overlap50 <- spTransform(all_my_fires_used_overlap50,
CRS("+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"))


all_my_fires_used_overlap0_1 <-
all_my_fires_used_overlap0_1[!all_my_fires_used_overlap0_1$ID %in%
all_my_fires_used_overlap50$ID, ]


ggplot() +
	coord_fixed(xlim = c(-9, -8), ylim = c(37, 37.6)) +
	suppressWarnings(geom_map(data = surroundings, map = surroundings, aes(x =
long, y = lat, group = group, map_id=region),fill = "grey", colour = "black",
size=0.5))+
	suppressWarnings(geom_map(data = WorldData, map = WorldData, aes(x = long, y
= lat, group = group, map_id=region), fill = "antiquewhite", colour = "black",
size=0.5))+
	suppressMessages(geom_polygon(data = my_study_area, aes(long, lat), colour =
alpha("darkred", 1/2), size = 0.7, fill=NA,linetype = "dashed"))+
	suppressMessages(geom_polygon(data = all_my_fires_used_overlap50, aes(long,
lat, group=group), colour = alpha("darkred", 1/2), size = 0.7, fill=NA))+
	suppressMessages(geom_polygon(data = all_my_fires_used_overlap0_1, aes(long,
lat, group=group), colour = alpha("blue", 1/2), size = 0.7, fill=NA))+
	theme(panel.grid.major = element_line(color = "gray", linetype = "dashed",
size = 0.5), panel.background = element_rect(fill = "aliceblue"))+
	xlab("Longitude") + ylab("Latitude")
```

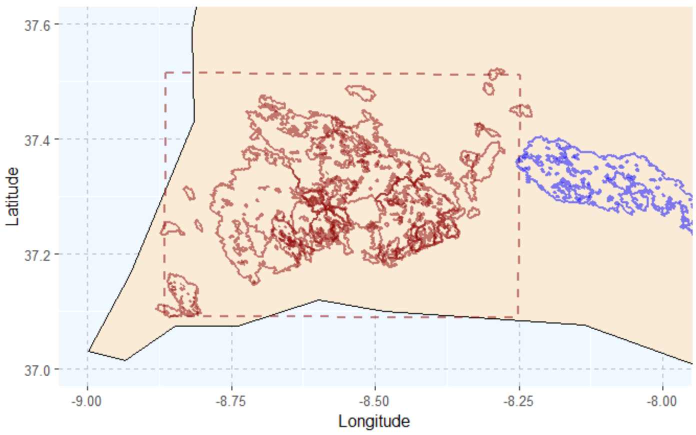


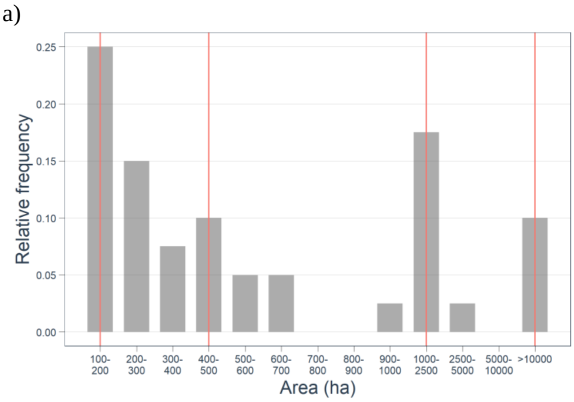
Figure 21 - Red perimeters represent the fire events that were considered for the fire size distribution when min.overlap = 50; blue perimeters represent the perimeters added to the fire size distribution analysis when considering min.overlap = 0.1

Including more fire perimeters in the analysis will modify both the meteorological aggregation (in the case that the fires included are dated) and the fire size distribution (notice the increase of the relative frequency of the fire seize class > 10,000 hectares). This comparison is shown below.

cluster	T	RH	WS	Cluster size	Cluster RF	min.overlap
1	27	43	26	26	0.31	50
2	34	24	18	58	0.69	
1	27	43	26	26	0.31	0.1
2	34	24	17	59	0.69	

Table 142 - Implications of choosing different min.overlap for the relative frequency of each cluster id. Green rows represent min.overlap of 0.1 % between the fire perimeters and the study area, and the red rows represent min.overlap of 50%.

a)



b)

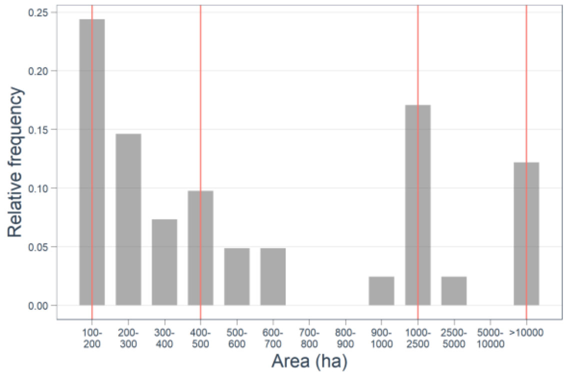


Figure 22 – Illustration of the fire size distribution with the automatic duration classes for two min.overlap: a) min.overlap of 50 %, and b) min.overlap of 0.1 %.

Duration class	Relative frequency	min.overlap
Duration_1	0.475	50
Duration_2	0.225	
Duration_3	0.2	
Duration_4	0.1	
Duration_1	0.463	0.1
Duration_2	0.22	
Duration_3	0.195	
Duration_4	0.122	

Table 15 - Implications of choosing different min.overlap for the relative frequency of each duration class. Green rows represent min.overlap of 0.1 % between the fire perimeters and the study area, and the red rows represent min.overlap of 50%.

6.2.6. Parameter fire.size.intervals

The parameter fire.size.intervals sets the fire size classes to be considered in the characterization of the study area. This parameter is highly important in the calibration process as it will create the historical fire size distribution that will be replicated in the simulations. Hence, the number and size of the bins defined in the parameter fire.size.intervals is of great importance. Below, we show a comparison of the fire size distribution using two settings of fire.size.intervals

```
build_report(study.area="~/study_area_mttfirecal.shp",
      my.fires="~/dated_fire_perimeters_mttfirecal.shp",
      my.dated.fires="~/dated_fire_perimeters_mttfirecal.shp",
      meteo.data="~/meteorological_data/fire_weather_study_area.csv",
      active.period="energy",
      meteo.aggregation="max.min",
      fire.aggregation="WS",
      min.size=100,
      min.overlap=50,
      fire.size.intervals=c(100, 500, 1000, 5000, 10000),
      create.clusters=TRUE,
      output.folder="~/changing_parameters/changing_fire_size_intervals")
```

The Figure 23 shows the comparison of the fire size distribution calculated using the settings in the section 6 and the settings above. As expected, the differences are significant. When including more bins (i.e. more intervals in fire.size.intervals) there is an increase in the accuracy on the characterization of the fire size distribution, which will in turn make the calibration process more complex. On the contrary, few intervals will lead to an easier calibration. This is explained in detail in Aparício et al. (2023).

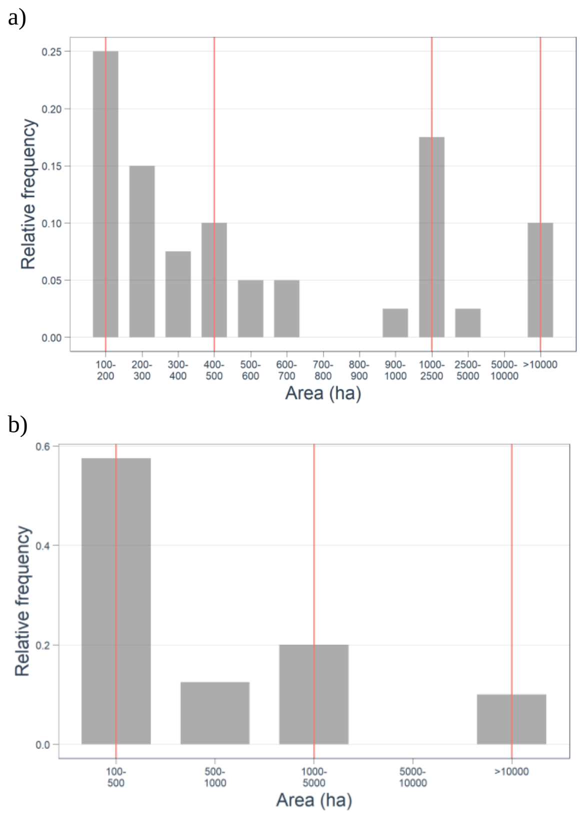


Figure 23 - Illustration of the fire size distribution with the automatic duration classes for two fire.size.intervals: a) fire.size.intervals = c(100,200,300,400,500,600,700,800,900,1000,2500,5000,10000), and b) fire.size.intervals = c(100,500, 1000,5000,10000). Vertical red lines represent the identified duration classes.

6.2.7. Parameter summarize.per.fire

The optional parameter summarize.per.fire allows the user to summarize the weather conditions per fire event. If TRUE, a multi-day fire will show only one set of weather conditions for the entire fire event, instead of multiple set of weather conditions (either one set per day or hourly set, depending on the type of meteo.aggregation). This parameter might be particularly useful to speed-up the meteorological aggregation in study areas with a large number of fire events. The default for this parameter is FALSE

Currently, the summarize.per.fire is only available if fire.aggregation = “WS”.

Below we show the implications of summarize.per.fire TRUE and FALSE.

```
build_report(study.area="~/study_area_mttfirecal.shp",
      my.fires="~/dated_fire_perimeters_mttfirecal.shp",
      my.dated.fires="~/dated_fire_perimeters_mttfirecal.shp",
      meteo.data="~/meteorological_data/fire_weather_study_area.csv",
      active.period="energy",
      meteo.aggregation="max.min",
      fire.aggregation="WS",
      min.size=100,
      min.overlap=50,
      fire.size.intervals=c(100, 200, 300, 400, 500, 600, 700, 800, 900, 1000,
2500, 5000, 10000),
      create.clusters=TRUE,
      summarize.per.fire = TRUE,
      output.folder="~/changing_parameters/with_ summarize_per_fire")
```

Fire ID	*year*	*month*	*day*	*T*	*RH*	*WS*	*Fire size*	*summarize.per.fire*
51	2001	8	7	33.32	29.64	26.8	1018	FALSE
54	2001	9	8	30.65	39.17	15.53	1319	
54	2001	9	9	30.87	29.26	11.61	1319	
51	2001	8	7	33.32	29.64	26.8	1018	TRUE
54	2001	9	8	30.65	39.17	15.53	1319	
60	2001	6	23	33.38	24.4	20.89	130	

Table 16 – Illustration of the first three rows of the file summary meteorology per fire event.csv when using summarize.per fire = FALSE and summarize.per.fire = TRUE. As shown, when using summarize.per.fire as TRUE, only the day with the highest wind speed is kept for fire ID 54.

Table 17 shows the influence of this parameter in the weather clusters. Setting summarize.per.fire = TRUE leads to an increase in the wind speed in the clusters, since only the observation with the highest wind speed is kept.

cluster	*T*	*RH*	*WS*	*Cluster size*	*Cluster RF*	*summarize.per.fire*
1	27	43	26	26	0.31	FALSE
2	34	24	18	58	0.69	
1	34	24	20	23	0.61	TRUE
2	27	41	27	15	0.39	

Table 17 – Influence of summarize.per fire in the clustering process: red rows represent summarize.per fire = FALSE and green rows represent summarize.per.fire = TRUE.

6.2.8. Exploring the fm.forests parameter

The fm.forests is an optional parameter that allows to identify the forest fuel model types. This is useful to properly calculate the fuel moisture content of forests and shrublands. In the absence of this parameter, the worst case scenario is used and the fuel moisture content of all fuel models is calculated assuming the equation for shrubland.

To evaluate the influence of this parameter, we can run the build_report as follows:

```
build_report(study.area="~/study_area_mttfirecal.shp",
       my.fires="~/dated_fire_perimeters_mttfirecal.shp",
       my.dated.fires="~/dated_fire_perimeters_mttfirecal.shp",
       meteo.data="~/meteorological_data/fire_weather_study_area.csv",
       active.period="energy",
       meteo.aggregation="max.min",
       fire.aggregation="WS",
       min.size=100,
       min.overlap=50,
       fire.size.intervals=c(100, 200, 300, 400, 500, 600, 700, 800, 900, 1000,
2500, 5000, 10000),
       create.clusters=TRUE,
       live.fuel.moisture=c(60,90),
       fm.forests=c(211,213,221,222,223,224,225,226,227),
       output.folder="~/changing_parameters/with_fm.forests")
```

To see the influence of this parameter, one can open the same fms file generated with and without the fm.forests parameter. Below we show the file kmeans_8_clusters_6 as an example. The file that does not consider the fm.forests (Figure 24 left) has the same value of dead fuel moisture for all fuel types, while the file that considers the fm.forests parameter (Figure 24 right) shows different dead fuel moisture content for the forest fuel types.

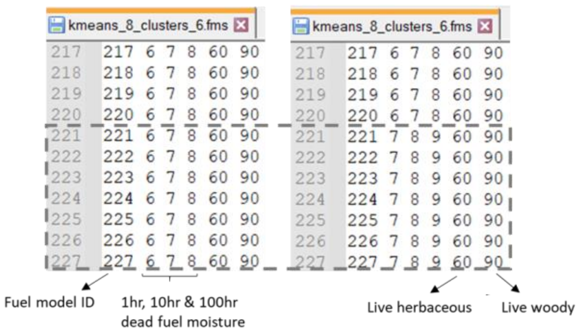


Figure 24 – Example of the FMS file without (left) and with (right) the fm.forests parameter.

6.3. Characterizing the weather conditions using percentiles

In alternative to creating weather clusters to characterize the weather conditions during a fire spread, users may choose to use weather percentiles. To use this option, the user must set the parameter `create.clusters` to `FALSE`:

```
build_report(study.area="~/study_area_mttfirecal.shp",
      my.fires="~/dated_fire_perimeters_mttfirecal.shp",
      my.dated.fires="~/dated_fire_perimeters_mttfirecal.shp",
      meteo.data="~/meteorological_data/fire_weather_study_area.csv",
      active.period="energy",
      meteo.aggregation="max.min",
      fire.aggregation="WS",
      min.size=100,
      min.overlap=50,
      fire.size.intervals=c(100, 200, 300, 400, 500, 600, 700, 800, 900, 1000,
2500, 5000, 10000),
      create.clusters=FALSE,
      output.folder="~/changing_parameters/with_ summarize_per_fire")
```

This will create similar files than when using clustering aggregation. The word report instead of having the clustering analysis shows a single table with the values for each variable (Table 18). The percentiles for the relative humidity are calculated inversely of the other variables. For instance, for percentile 90 (last row in Table 18), the temperature and wind speed represent the percentile 90 of these variables, while the relative humidity represent the percentile 10.

percentile	*T*	*RH*	*WS*
50	32.080	28.560	20.405
60	33.006	25.366	21.492
70	34.404	23.444	24.216
80	35.494	19.870	25.522
90	38.258	16.370	27.145

Table 18 – Characterization of weather variables after the aggregation process (percentile in this case). The column percentile refers to the percentile, T refers to temperature in degrees Celsius, RH refers to the relative humidity, WS refers to the wind speed in kmh,

Note that using the percentile as the aggregation method will only allow to use one set of weather conditions in the calibration process (unlike the clustering aggregation that allows multiple weather conditions depending on the number of clusters).

7. Generating ignitions

In the following sections, we will use the example produced in the section 6 using the following setting:

```
build_report(study.area="~/study_area_mttfirecal.shp",
      my.fires="~/dated_fire_perimeters_mttfirecal.shp",
      my.dated.fires="~/dated_fire_perimeters_mttfirecal.shp",
      meteo.data="~/meteorological_data/fire_weather_study_area.csv",
      active.period="energy",
      meteo.aggregation="max.min",
      fire.aggregation="WS",
      min.size=100,
      min.overlap=50,
      fire.size.intervals=c(100, 200, 300, 400, 500, 600, 700, 800, 900, 1000,
2500, 5000, 10000),
      create.clusters=TRUE,
      output.folder="~/characterization")
```

Ignitions are essential do run MTT or FConstMTT. Ignitions can be created within the MTTfireCAL by using the Gen_ign function

First, let's see what parameters are required to generate the ignitions in the landscape

```
?Gen_ign
```

Which shows in the help tab:

```
Gen_ign(
  ign.raster,
  IgnitionData,
  fuelmap,
  nIgnitions,
  unburnable,
  LandCover.yr,
  LandCover.wgt,
  shapeOut,
  allow.zero,
  random.ignitions,
  output.folder
)
```

The ign.raster is the raster file with the probability of ignition. This file was generated by applying the ArcGIS tool Kernel Density to the ignition points of fires that burned more than 100 hectares. This raster file is included in the tutorial folder.

The `IgnitionData` is the text file (csv) with the relative frequency of each meteorological cluster (or percentile) and duration to be simulated. If the function build_report was used in the process, then the file to be used should is stored in the folder final_freqs. In this example, and because we chose the clustering method kmeans and accepted two clusters, the file loaded should be “kmean_2_clusters_final_freqs.csv”.

The `fuelmap` represents the fuelmaps that will be used in the calibration process. As explained above, the study area had two critical fire seasons: 2003 and 2018. In this example, the fuel maps where created following the methodology presented in Sá et al. (2023) and the fuel models characterized in Fernandes et al. (2009).

nIgnitions represent the total number of ignitions to use in the calibration process. As discussed in Aparício et al. (2023), the minimum recommended nIgnitions should represent a ratio between 50 and 20 burnable hectares per ignition (i.e. one ignition per each 50-20 burnable hectares in the study area). This value can be easily accessed by using the function *get_number_ign*.

`unburnable` represents the fuel model class(es) that are unburnable and where, for that reason, ignitions should not be placed. When using Scott and Burgen (2005) fuel classification, this value is 98.

`LandCover.yr` represents a vector with the years of each fuel model provided. In this example, the vector should contain 2003 and 2018.

`LandCover.wgt` represents a vector with the weights of each fuel model. In our example, the 2003 fuel map must have more weight than 2018 because it accounted more for the overall burned area (see Figure 11).

`shapeOut` indicates if a point shapefile containing the ignitions should be exported or not.

The allow.zero parameter permits the user to set a minimum ignition probability in all burnable areas in the landscape. Whenever true, all burnable pixels have a minimum ignition probability equal to 0the 5th percentile of the original ignition probability.

Whenever True, The `random.ignitions` parameter generates random ignitions in the landscape, independently of `ign.raster`. If True, then no `ign.raster` is required.

Before using the function *Gen_ign* to generate the ignitions, one can identify the minimum number of ignitions required for the calibration process as follows:

```
get_number_ign("~/fm_2003.asc",
               unburnable=98)
```

Which will return the following message:

```
Minimum number of ignitions: 4439.

The value was calculated using the rule of thumb outlined in Aparício et al. 2023
(doi:).
```

We can use this value to generate ignitions. Here, we chose to round the 4,439 ignitions to 5,000 ignitions. The function *Gen_ign* can then be used as follows:

```
Gen_ign(ign.raster="~/ignition_probability.tif",
  IgnitionData= "~/characterization/final_freqs/kmean_2_clusters_final_freqs.csv",
  fuelmap=c("~/fm_2003.asc",
            "~/fm_2018.asc"),
  nIgnitions=5000,
  unburnable=98,
  LandCover.yr=c(2003,2018),
  LandCover.wgt=c(0.65,0.35),
  shapeOut=1,
  allow.zero=FALSE,
  random.ignitions=FALSE,
  output.folder="~/characterization")
```

This will create a folder named ignitions that contain the ignition point shapefiles, a text file with the coordinates of the ignition points and a csv file named “clusters_freqs_final”. The clusters_freqs_final shows the relative frequency of all scenarios (i.e. combination of meteorological cluster, wind direction, duration class and fuel map) and the total number of ignitions in each scenario.

Each scenario is identified by the conditions it represents. For instance, the first scenario in the Figure 25 (ignitions_cluster_1_duration_1_0_land_cover_2003) represents the ignitions of the cluster 1 (ignitions_cluster_1), duration class 1 (duration_1), wind blowing from north (0 degrees), and the land cover of 2003 (land_cover_2003).

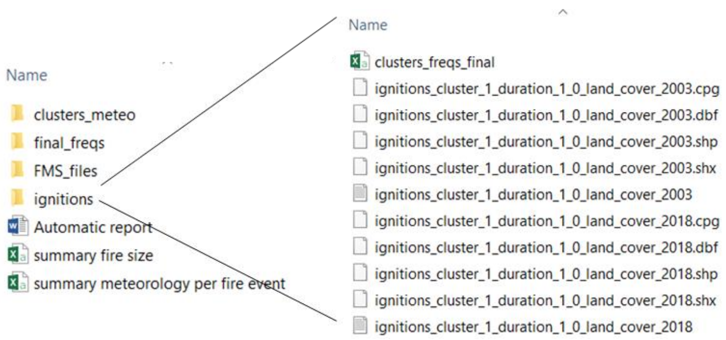


Figure 25 – Screenshot of the ignitions folder created after using the function Geg_ign.

7.1. Plotting the ignitions

The number of the ignitions generated differs between scenarios. This is essential to maintain the historical proportion between fire sizes and to properly replicate the fire size distribution.

```
library(raster)
library(rgdal)
ign_prob <- raster("~/tutorial_data/ignition_probability.tif")

plot(ign_prob)
```

```
sce_a <-
readOGR("~/characterization/ignitions/ignitions_cluster_2_duration_1_315_land_cove
r_2003.shp")

plot(sce_a,add=T)

sce_b <-
readOGR("~/characterization/ignitions/ignitions_cluster_2_duration_3_225_land_cove
r_2003.shp")

plot(sce_b,pch=1, col="blue", add=T)
```

The figure 26 shows the difference between the number of ignitions in two scenarios, with more ignitions being sampled in more frequent scenarios. It also shows how ignitions are randomly sampled following the historical ignition probability.

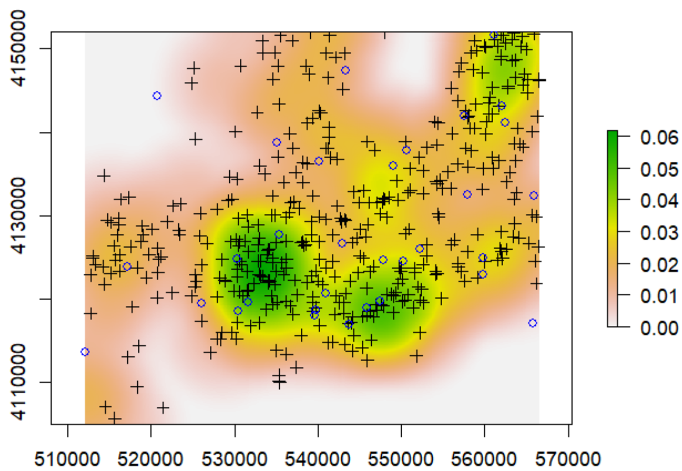


Figure 26 – Distribution of the sampled ignitions for the scenario cluster 2, duration class 1 with wind blowing from NW in the landscape of 2003 (ignitions_cluster_2_duration_1_315_land_cover_2003; black crosses) and cluster 2, duration class 3 with wind blowing from SW in the landscape of 2003 (ignitions_cluster_2_duration_3_225_land_cover_2003; blue circles). The ignition probability surface is represented as the basemap.

8. Running FConstMTT

At this stage, we are now ready to start using FConstMTT. The function to run the FConstMTT from R with multiple combinations of duration values is run_fconstmtt.

Note that to use this function the user **must have** the FConstMTT in their machine. FConstMTT can be downloaded at https://www.alturassolutions.com/FB/FB_API.htm. When using FConstMTT please cite it as "Fire behavior and First Order Fire Effects calculations produced with the command line applications developed by the Missoula Fire Sciences Laboratory, Missoula, MT".

8.1. Downloading and storing FConstMTT

To download FConstMTT go to https://www.alturassolutions.com/FB/FB_API.htm. This hyperlink will redirect you to the Missoula fire Lab Applications. To download the MTT command line simply click on FB_x64.zip

Figure 27 – Screenshot of the webpage where to download the FConstMTT.

After downloading the file, unzip it using winrar or 7zip (or any other software) and save the folder in an easily accessible location (in this tutorial we saved it in the Desktop under the name of FB_x64). You will have a folder named FB_x64. This folder contains not only the FConstMTT but also other fire growth algorithms, such as Randig and Farsite. The folder of each algorithm has the corresponding manuals and tutorial data. The bin file contains the executables of the algorithms. This is the folder that will be accessed in MTTfireCAL to run FConstMTT; the user will later be asked to provide the path to this folder in the MTTfireCAL.

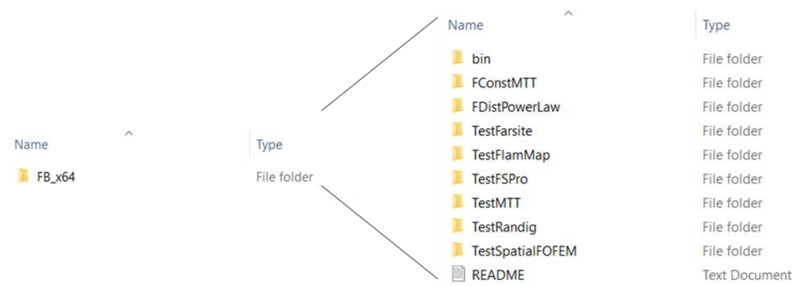


Figure 28 – Screenshot of the downloaded file containing the FConstMTT.

8.2. Explaining the function run_fconstmtt

To understand the parameters in the function run_fconstmtt:

```
?run_fconstmtt
```

Which displays the following parameters in the help tab:

```
run_fconstmtt(
  Folder.fconstmtt
  Folder,
  landscape,
  CrownFireMethod,
  customfmd,
  FmsFolder,
  Resolution,
  GridDistanceUnits,
  MeteoFile,
  Duration.1,
  Duration.2,
  Duration.3,
  Duration.4,
  Duration.5,
  WS_unit,
  FireListFile,
  SpotProbability,
  OutputFirePerims,
  MetricFLP,
  Run.fconstmtt
)
```

`Folder.fconstmtt` corresponds to the path to the folder containing the bin folder with the FContMTT executable.

`Folder` is the path to the folder where the the input and output folder will be created.

`landscape` is the path to the folder contained the landscape file(s) to be used (either .tif or .lcp).

`CrownFireMethod` is the equation that should be used to calculate crown fires. Either 0 for the Finney's method, or 1 for use of Scott and Rheinhardt's method.

`customfmd` is the file containing the characterization of custom fuel models (.fmd file).

`FmsFold`er is the path to the folder containing the fms file(s) to be use in the simulation.

`Resolution` corresponds to the desired resolution of calculations. Usually is the same as the resolution of the landscape file.

`GridDistanceUnits` sets grid distance units, where 0 is for meters and 1 is for feet.

`MeteoFile` corresponds to the text file containing the meteorological characterization of the clusters or percentiles to be used in the simulation. This file is stored in the folder "clusters_meteo" after running the function build_report. Here, we will use the kmean_2_clusters.csv.

`Duration.X` represent the duration values to be tested per duration class. The values should be given as a vector of three numbers: the first is the minimum value to be tested, the second is the maximum value to be tested and the third is the step or by value. For example, for Duration.1 = c(50,200,50), the minimum value that will be tested for duration class 1 is 50 minutes, the maximum value is 200 minutes and the step is 50 minutes. This would result in testing the following duration values for duration class 1: 50,100,150,200.

`WS_unit` represents the units of wind speed ("kmh" or "mph"). If the other functions of MTTfireCAL were used, then this parameter must be "kmh".

`FireListFile` is the path to the folder containing the ignitions to be used. Here only the text files with the coordinates of the ignitions are used.

`SpotProbability` sets the probability of spotting. Note that spotting is highly dependent on the data used in the landscape file. If no data about canopy is given in the landscape file, then spotting probability should not be used.

`OutputFirePerims` is a binary parameter: if 1 then the simulated fire perimeters are exported (as shapefile), if 0 then the simulated fire perimeters are not exported. Usually, the simulated fire perimeters are not used in the calibration process.

`MetricFLP` is a binary parameter: if 1 then the FLP is stored in the metric system, if 0 the FLP is stored in the imperial system. It is important to note that the metric FLP has 20 categories of fire intensity level (expressed as flame lengths in 0.5 meter categories), while the imperial FLP has six categories. This makes the imperial FLP faster to save (and smaller in size). Since this output is not usually used in the calibration process, one can export the FLP in the imperial units to speed up the process.

`Run.fconstmtt` is a binary parameter: if 1 then the FConstMTT command line is launched, if 0 then the function only creates the input and batch files that are used to run FConstMTT.

8.3. Using the function run_fconstmtt

To use run_fconstmtt in the study area used in this tutorial, do:

```
run_fconstmtt(Folder.fconstmtt="~/Desktop/FB_x64",
  Folder="~/characterization",
  landscape="~/data/landscape",
  CrownFireMethod=1,
  customfmd="~/data/fmd_pf_adai.fmd",
  FmsFolder="~/characterization/FMS_files/kmeans_2_clusters",
  Resolution=100,
  GridDistanceUnits=0,
  MeteoFile="~/characterization/clusters_meteo/kmeans_2_clusters.csv",
  Duration.1=c(160,200,20),
  Duration.2=c(300,350,25),
  Duration.3=c(600,800,100),
  Duration.4=c(2200,3200,1000),
  WS_unit="kmh",
  FireListFile="~/characterization/ignitions",
  SpotProbability=0,
  OutputFirePerims=1,
  MetricFLP=0,
  Run.fconstmtt=1)
```

In this example, we are using a custom fmd file for Portugal developed by Fernandes et al. (2009). When using the default fuel models from Scott and Burgen (2005), the user must delete this parameter.

To use the function one needs to set the paths to the files previously created, such as the meteorological cluster classification, landscape file and ignitions. Then, we need to set the range of values to be tested in each duration class. Because have four duration classes (as defined in function build_report and shown in Figure 13), we need to set the range for the four durations. This is done by using the parameters `Duration.1`, `Duration.2`, `Duration.3`, and `Duration.4`. Here, we test three value in each duration class, except for `Duration.4` where only two values are tested:

- Duration.1 as 160, 180 and 200 minutes
- Duration.2 as 300, 325 and 350 minutes
- Duration.3 as 600, 700 and 800 minutes
- Duration.4 as 2200 and 3200 minutes

As a good practice, we suggest that works using MTTfireCAL report the duration values tested in any publication, following the example shown in Table 18.

	Duration 1	**Duration 2**	**Duration 3**	**Duration 4**
Minimum	160	300	600	2200
Maximum	200	350	800	3200
Step	20	25	100	1000
				Total combinations = 54

Table 18 – Summary of the tested values in each duration class.

Please take into consideration that the number of values to be tested in each duration class is small as it is for demonstration purposes only. In real studies, a large number of values in a wider range should be tested. If speed is an issue, one can consider running only one value for Duration.3 and Duration.4 in this tutorial. To do so, simply set the minimum and the maximum to the desired duration value. For instance:

```
Duration.3=c(600,600,100)
Duration.4=c(3200,3200,100)
```

will only run one duration value for Duration.3 (600 minutes) and one value for Duration.4 (3200 minutes).

Notice that we've set the `MetricFLP` as 0, indicating that FConstMTT will store the flame length as feet instead of meters. Both FLP files in meters and feet are similar, but the FLP in the metric system is more detailed, with more classes of flame length. Hence, by choosing to run the simulations in imperial system represents a gain in the computational time and allows to save space in the user's disk. Note that the FLP is not used in the calibration process.

8.4. After using run_fconstmtt function

After using the run_fconstmtt function, the following message is shown in the console of RStudio:

```
"Fconsmtt running! Take a break and come back soon"
```

At the same time, the Command Prompt in Windows opens and FConstMTT starts running. Note that FConstMTT does not use R to run.

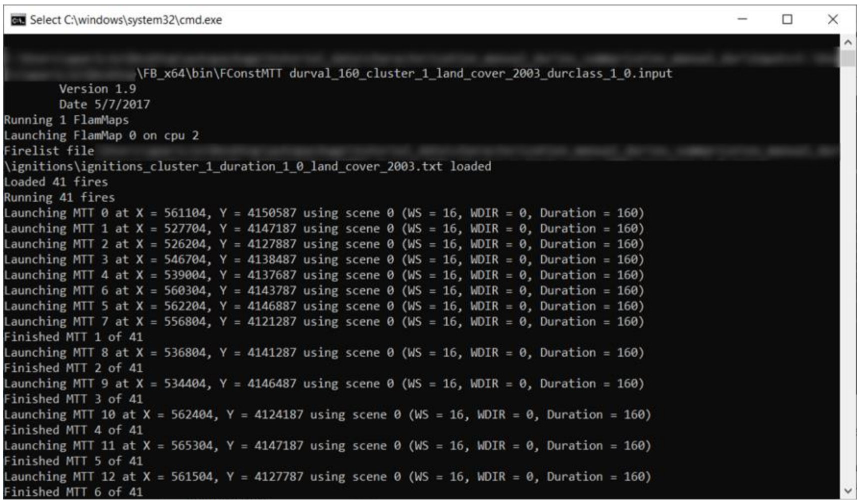


Figure 29 – Screen shot of FConstMTT running

The first lines of the command prompt show the location of the files set by the user. The first part of the first row indicates where the .input files are located. Then, the second part of the same row has the path to the FConstMTT executable (notice that the path corresponds to where the FConstMTT was stored and as specified in run_fconstmtt). The last element of the second row shows the scenario that will be used in the run. In this case, it is the scenario with 160 minutes of fire duration, meteorological conditions of cluster 1, landscape of 2003, duration class 1 and wind direction of 0° (i.e., north).

The Firelist file in the command prompt represents the ignition file that will be used in the FConstMTT. The name of the file must match the name of the scenario specified in the first rows.

Finally, the rows bellow indicate the X and Y coordinates of the ignition used, as well as the wind speed (WS), wind direction (WDIR) and Duration used. Note that the wind speed displayed is in miles per hour.

Once all the ignitions were generated for a given scenario, a message saying `Fires done. Assembling results. Calculations complete` is shown in command prompt; and the outputs are stored in the Output folder created.

Figure 30 – Screenshot of the outputs stored after running FConstMTT. Only one scenario is shown in the figure. Each scenario will have one firelist file, one burn probability file, one FLP file, and one shapefile of the simulated fire perimeters.

Afterwards, the fire growth simulation of a new scenario starts (in the example shown in Figure 31, the scenario represents the same conditions as before but this time with the landscape of 2018).

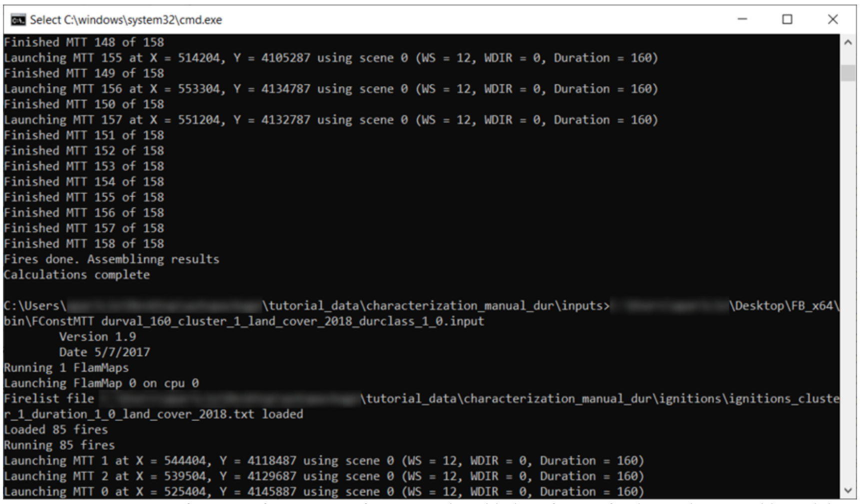


Figure 31 – Screenshot of the end of simulation of one scenario, with the reference for the results' assemble, and the start of the simulation of a new scenario.

Please allow a few hours to run all the scenarios. For a machine with an AMD Ryzen 9 3950X 16-Core Processor 3.49 GHz and 32 GB of RAM it ran in 2 hours. This process will require more time for machines with less cores.

8.5. Analysing the .input and batch files

After using the run_fconstmtt function, two folders are created in the folder specified by the user (characterization_manual_dur folder in this tutorial): inputs and Outputs folder. As the name indicates, the inputs folder stores the input and batch files required to run FConstMTT. Any of these files can be opened using a text editor software. We recommend using [Notepad++](). To open and edit the .input files, right click on it and open in the notepad.

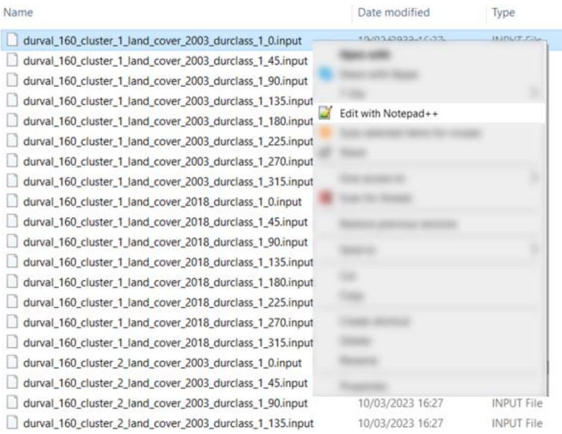


Figure 32 – Screenshot exemplifying how to open na .input file.

A total of 345 files are stored in the inputs folder, one for each scenario simulated (N=343), one text file that indicates the progress in the simulations (fconstProgress.txt) and one batch file.

Figure 33 shows a screenshot of an .input file. The .input file represents the settings and the path to the files that will be used in FConstMTT.

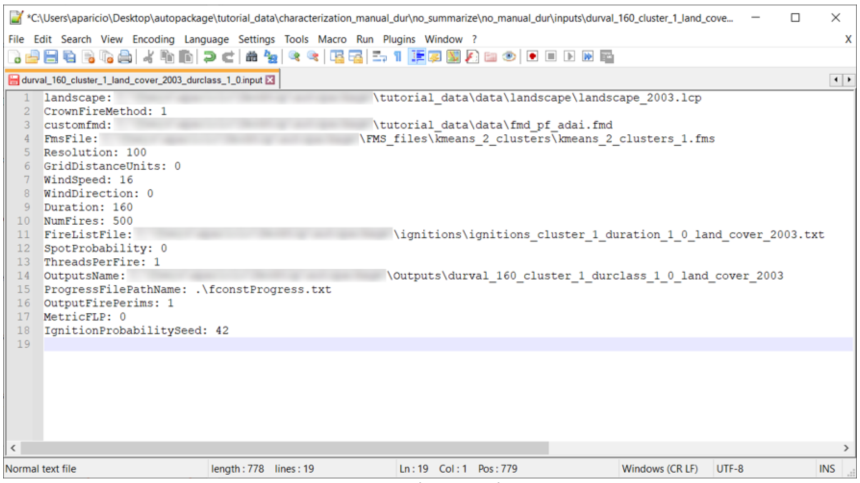


Figure 33 - Screenshot of an .input file opened in notepad++.

The input file should be read as follows:

landscape: Path to the landscape file that will be used in this simulation

CrownFireMethod: method for crown fire calculation. 0 if Finney's method, or 1 if Scott and Rheinhardt's method.

customfmd: path to the custom fmd file.

FmsFile: path to the FMS file to be used in the simulation

Resolution: Value of the desired resolution of calculations.

GridDistanceUnits: The units to be used in the calculations: 0 for meters or 1 for feet

WindSpeed: Value of wind speed to be used in the simulation. This value is in mph.

WindDirection: Value of wind direction in degrees.

Duration: Value of the duration (or maximum simulation time) in minutes to be used in this simulation.

NumFires: Value of the number of fires to simulate.

FireListFile: Path and file containing the ignitions (as text file) to be used in this simulation.

SpotProbability: Value of the probability of having spotting.

ThreadsPerFire: Number of threads per fire. Default is 1.

OutputsName: Path and name of where and how the outputs of this scenario will be named.

ProgressFilePathName: Path and name of the file for progress information.

OutputFirePerims: Binary. If 1 then the simulated fire perimeters will be saved. If 0, then the simulated fire perimeters will not be saved.

MetricFLP: Binary. If 1 then the FLP will be stored in the metric system. If 0, then the FLP will be stored in feet.

IgnitionProbabilitySeed: Sets the random seed for ignition random number generator to be used with the ignition probability grid.

Note that some of the information in the input file may seem contradictory. For example, the parameter **NumFires** is set to 500 in all input files, and the parameter **IgnitionProbabilitySeed** is present in all input files. This would result in generating 500 random ignitions in all scenarios. But because we are specifying which ignitions to use (in **FireListFile**), this information overwrites the previous information of generating 500 random ignitions. Nevertheless, the parameters of **NumFires** and **IgnitionProbabilitySeed** are required to run FConstMTT, even when they are ignored like in this case.
The user can also open the batch file by right click on it and select the notepad. The batch file is essentially a file containing a list of instructions to be carried out in turn, something like the maestro. Figure 34 shows the batch file. The scenarios are simulated in FConstMTT following the order of appearance in the batch file. In the bottom there is also the indication of the number of lines, i.e., the number of scenarios that will be simulated (N=309).

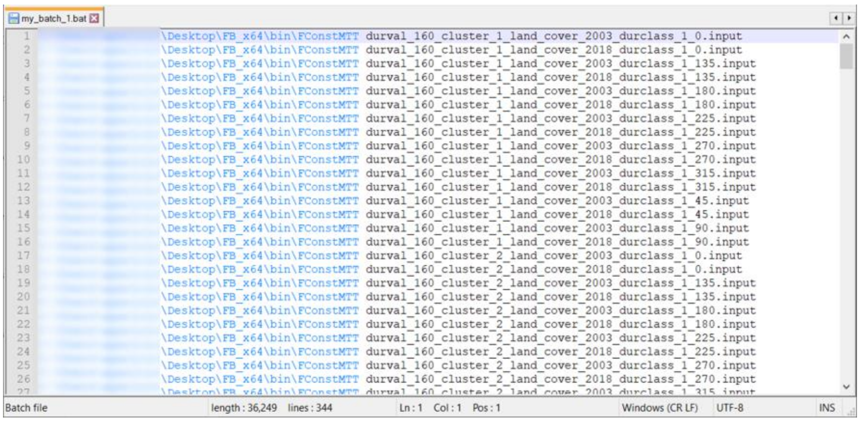


Figure 34 - Screenshot of the batch file opened in notepad++.

9. Running a single meteorological scenario in FConstMTT

Some users may want to use MTTfireCAL to calibrate the MTT using a single scenario. In this case, all the steps explained above can be simplified.

Let's use as an example the same study area in south Portugal and the meteorological files already produced.

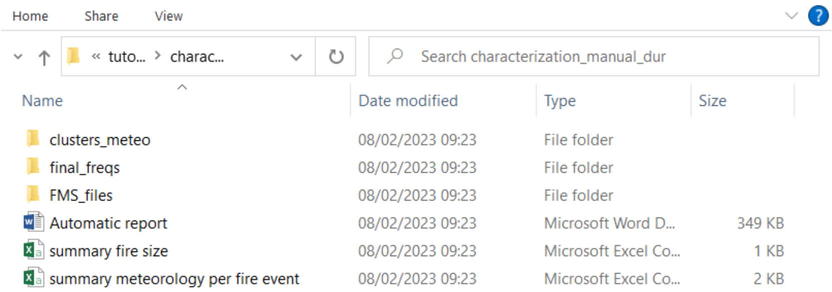


Figure 35 – Screenshot of outputs generated after running the function *build_report*.

As explained above, the folder clusters_meteo contains the meteorological conditions in each cluster. Opening
We can also assess that wind blowing from NW corresponds to ca. 50% of all observations.

Hence, and considering the objective of the study, the user can determine that running a single meteorological scenario is adequate. To do this within MTTfireCAL package, the function to use is run_fconstmtt_simple. First, one can evaluate the required inputs by:

```
?run_fconstmtt_simple
```

Which shows in the help tab:

```
run_fconstmtt_simple(Folder,
  landscape,
  CrownFireMethod,
  customfmd,
  FmsFolder,
  Resolution,
  GridDistanceUnits,
  wind.speed,
  wind.direction,
  Duration.1,
  Duration.2,
  Duration.3,
  Duration.4,
  Duration.5,
  Duration.1.weight,
  Duration.2.weight,
  Duration.3.weight,
  Duration.4.weight,
  Duration.5.weight,
```

```
  WS_unit,
  FireListFile,
  SpotProbability,
  output.folder,
  OutputFirePerims,
  MetricFLP,
  Run.fconstmtt)
```

The inputs required are similar to those required to use run_fconstmtt. However, there are a few exceptions, such as the need to define a value for wind speed (wind.speed) and a value for wind direction (wind.direction). These can be based on user's own data or on the data generated by the build_report function. In this example, we will the values obtained by the build_report function.

Additional inputs about the duration classes are required, namely the parameter Duration. weight.file. This file should have the relative frequency (or weight) of each duration (e.g. any csv file saved in the folder final_freqs). Alternatively, the user can specify the relative frequency (or weight) of each duration class using the duration.X.weight (where X is the number of the duration).

Commented [a2]: falta fazer

Below the example of using the parameter Duration. weight.file:

And below the example of using the duration.X.weight:

Note that this function can be used without previously using any other function from MTTfireCAL. The user can use a local weather station to characterize the meteorological conditions and use them to generate the FMS files and to define the wind.speed and wind.direction. Then, the user can define the number of durations that best describe the historical fire size distribution and that best suit their research objectives; and then assign a weight value to each duration class.

10. Evaluating the calibration quality

The last set of functions in MTTfireCAL were developed to assess the calibration quality. The user will be able to rank the several combinations ran and choose the one that best replicate the historical fire pattern. Note that it is common that even after testing several duration values, all combinations fail to satisfactorily reproduce the historical fire pattern. When this occurs, the process of setting new duration values and combinations must be repeated.

There are two functions to evaluate the quality of the calibration: *evaluate_fire_size* and *evaluate_BP_nxburned*. The first function is used to combine the several duration values set and compares the simulated and historical fire size distribution. The function calculates several performance metrics such as Pearson correlation coefficient, the Root Mean Square Error (RMSE), the percentage of the Normalized Root Mean Square Error (NRMSE), the Mean Absolute Error (MAE), the Relative Absolute Error (RAE), and the Nash-Sutcliffe model efficiency (NSE). The *evaluate_BP_nxburned* is used to correlate the simulated burn probability with the historical number of times burned.

10.1. Explaining the function *evaluate_fire_size*

To understand the parameters in the function run_fconstmtt:

```
?evaluate_fire_size
```

Which displays the following parameters in the help tab:

```
evaluate_fire_size(Folder.Outputs,
  intervals,
  all.dist,
  hist.fire.sizes,
  freqs.durclass,
  plot.all)
```

`Folder.Outputs` corresponds to the path to the folder containing the outputs of the FConstMTT runs.

`intervals` correspond to the intervals of fire size to be considered in the comparison between simulated and the historical fire size distribution.

`all.dist` Logical. If true, then all the fire size distribution is used to calculate the RMSE and the correlation. If false, then the fire size distribution will only be considered starting from the first numeric value identified in intervals.

`hist.fire.sizes` corresponds to the text file (csv) containing the historical fire size ("summary fire size.csv" if the function get_fire_weather was used).

`freqs.durclass` corresponds to the text file (csv) with the relative frequency of each meteorological and fuel map scenario used. If the function Gen_ign was used in the process, then the file to be used here should be "clusters_freqs_final.csv", which is located in the ignition folder.

`plot.all` If TRUE, then the fire size distribution of all combinations will be plotted and exported in multiple png files. If FALSE, only the combination with the lowest RMSE is plotted and exported in a single png file.

10.2. Using the function *evaluate_fire_size*

To apply the function *evaluate_fire_size* to the study case in this tutorial, simply:

```
evaluate_fire_size(Folder.Outputs="~/characterization_manual_dur/Outputs",
      intervals=c(100,200,300,400,500,600,700,800,900,1000,2500,5000,10000),
      all.dist=FALSE,
      hist.fire.sizes="~/characterization_manual_dur/summary fire size.csv",
      freqs.durclass="~/characterization_manual_dur/ignitions/clusters_freqs_final
.csv",
      plot.all=TRUE)
```

Note that the values used to define the `intervals` is the same as the ones used to define the duration classes to be used in the parameter `fire.size.intervals` of the function build_report (see here). This is essential as the entire calibration is dependent on the historical fire size distribution.

Also note that the parameter `all.dist` was set to `FALSE.` Likewise, this is because in the function build_report we only considered fires that burned more than 100 hectares. As a result, the file `summary fire size only.csv` only stored fires larger than 100 hectares. To allow for a fair comparison between the simulated and historical fire size distribution, the user should set the `all.dist` as `False`, otherwise the interval of fire size between 0-100 hectares will also be used in the comparison. This would return a historical relative frequency of zero, which does not correspond to the reality.

After running the evaluate_fire_size function, multiple files are saved in the Outputs folder (`~/characterization_manual_dur/Outputs";` Figure 36): a csv file containing the performance metrics for all the combinations (.csv); a csv file containing the relative frequencies of each fire size class for each combination (simulated_frequencies_fire_size.csv); one png file showing the historical and simulated fire size distribution of the combination with the lowest RMSE (fire size distribution with lowest RMSE.png); and multiple png files showing the simulated and historical fire size distribution (e.g. fire size distribution part1.png). Each png file has the historical and simulated fire size distribution from 11 different combinations.

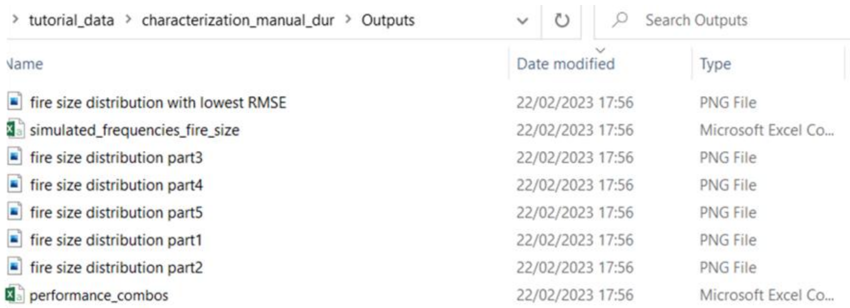


Figure 36 – Screenshot of the outputs generated after using the function evaluate_fire_size. The files include a csv file with the performance metrics for each combination tested (performance_combos), one csv file with the relative frequency of the simulated fire size (simulated_frequencies_fire_size), several png files showing the fire size distribution of multiple combinations (e.g. fire size distribution part1), and one png file showing the simulated fire size distribution of the combination with the lowest RMSE.

The performance_combos.csv shows the performance metrics for all combinations (Figure 37). The performance metrics include the RMSE, Pearson correlation coefficient, percentage NRMSE, MAE, RAE and NSE. The combinations are ordered by their RMSE, with the combination with the lowest RMSE at the top. Hence, from all the combinations ran, the combination 32 is the one showing the best performance metrics. This combination results from simulations that used 180 minutes of fire spread duration for duration class 1, 325 minutes for duration class 2, 600 minutes for duration class 3, and 3200 minutes for duration class 4.

1	combo	durval1	durval2	durval3	durval4	RMSE	Correlation	percentage NRMSE	MAE	RAE	NSE
2	32	180	325	600	3200	0.022	0.965	28.942	0.019	0.310	0.915
3	29	180	300	600	3200	0.023	0.969	29.583	0.019	0.308	0.911
4	5	180	325	600	2200	0.023	0.961	30.096	0.020	0.328	0.908
5	2	180	300	600	2200	0.024	0.965	30.699	0.020	0.325	0.905
6	35	180	350	600	3200	0.026	0.951	33.152	0.020	0.335	0.889
						●●●					
51	49	160	325	800	3200	0.059	0.799	76.259	0.042	0.690	0.411
52	19	160	300	800	2200	0.059	0.808	76.869	0.041	0.681	0.402
53	52	160	350	800	3200	0.059	0.788	77.223	0.044	0.724	0.396
54	22	160	325	800	2200	0.059	0.792	77.327	0.044	0.718	0.394
55	25	160	350	800	2200	0.060	0.780	78.299	0.046	0.752	0.379

Figure 37 – Example of the performance_combos.csv file. The figure shows the combinations with the lowest RMSE in the top panel and the combinations with the highest RMSE in the bottom panel. Combo refers to the combination of the duration values, durval1 refers to the value set for duration class 1, durval2 refers to the value set for duration class 2, durval3 refers to the value set for duration class 3, durval4 refers to the value set for duration class 4, RMSE represents the Root Mean Square Error, Correlation is the Perason Correlation Coeficient, percentage NRMSE is percentage of the normalizes RMSE, MAE is the Mean Absolute Error, RAE is the Relative Absolute Error and NSE is the Nash–Sutcliffe model efficiency coefficient.

To visualize the historical and simulated fire size distribution, we can open the file fire size distribution part1.png (Figure 38). This figure shows the histogram of historical fire size distribution and the simulated fire size distribution by 11 of the 54 combinations. Each combination has a unique number that matches the performance_combos.csv (Figure 37). Figure 38 shows how changes in fire spread duration changes the distribution outcome.

Figure 38 – Example of the simulated fire size distribution of multiple combinations and its comparison with the historical fire size distribution.

The png file “fire size distribution with lowest RMSE” facilitate the visual interpretation of the simulated and historical fire size distribution (Figure 39). The visual comparison between both distributions and the performance values in Figure 37 allow to conclude that the MTT is reproducing the historical fire size distribution.

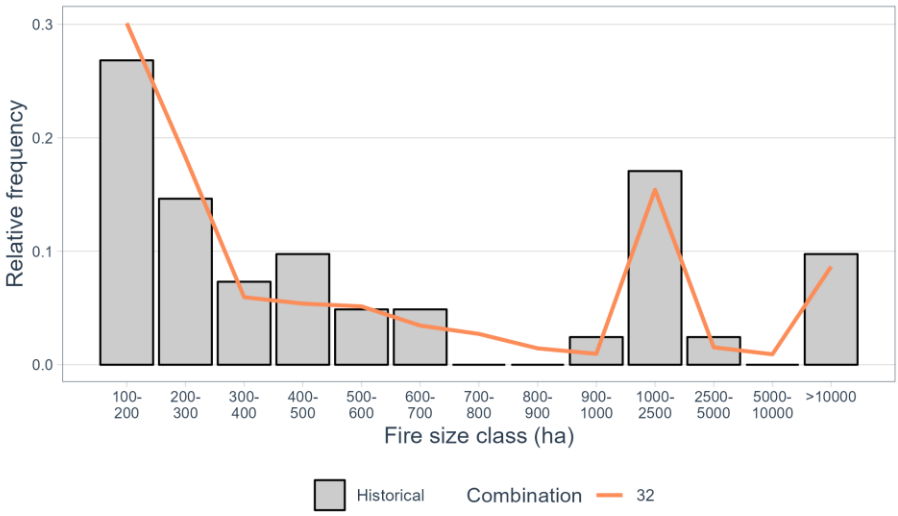


Figure 39 – Fire size distribution simulated by the combination showing the lowest RMSE when compared with the historical fire size distribution.

Note that the best combination obtained by the user might be slightly different from the one shown here. This is due to a certain degree of equifinality that is both dependent on the classes of fire sizes that the user defined in the function *build_report* and on the randomness of the fire spread simulations itself (from the ignitions). Also note that the combination chosen may not correspond to the best possible combination that reproduces the historical fire size distribution. In turn, it simply represents the best combination of the **tested** combinations.

10.2.1. Comparing the best and the worst combinations in reproducing the historical fire size distribution

It might be interesting to compare the fire size distribution created by the combination with the lowest and the highest RMSE (combination 32 and combination 25, respectively). To do so, we will use the file simulated_frequencies_fire_size.csv that was generated by the function evaluate_fire_size.

First we load the required libraries and the csv file. Then, we keep only the combinations that we want to plot. In this case, the combination 0 that corresponds to the historical fire size distribution, and the combination 25 and 32 that correspond to the combination with the poorest and the best performance metrics, respectively (as shown in Figure 37).

```
library(ggplot2)
library(dplyr)
library(tidyquant)

all_freqs <-
read.csv("~/characterization_manual_dur/Outputs/simulated_frequencies_fire_size.cs
v")

#select combos 0 (historical), 25 and 32

best_and_worst_combos <- subset(all_freqs, combo==0 | combo==25 | combo==32)

#plot
ggplot(best_and_worst_combos, aes(x=class, y=relative.frequency,
group=factor(combo),fill = factor(combo))) +
	geom_bar(data = filter(best_and_worst_combos, combo  == 0),
	aes(fill="Historical"),col="black",stat = "identity") +
scale_fill_manual("",values=c("Historical" = "grey80"))+
geom_line(data = filter(best_and_worst_combos, combo  != 0), aes(col =
factor(combo), group = factor(combo)),size=1)+
	scale_color_brewer(palette="Accent",
	direction=-1,
	name="Combination",
	labels = c("Worst", "Best"))+
scale_x_continuous(breaks= c(1:nrow(subset(best_and_worst_combos,combo==0))),
	labels=  gsub("- ","-\n",filter(best_and_worst_combos, combo  == 0)$area))+
theme_tq()+
theme(panel.grid.minor = element_blank(), axis.title=element_text(size=12),
	panel.grid.major.x = element_blank(), axis.text = element_text(size = 8),
	legend.text = element_text(size=12)) +
ylab("Relative frequency") + xlab("Fire size class (ha)")
```

Which will return the following figure:

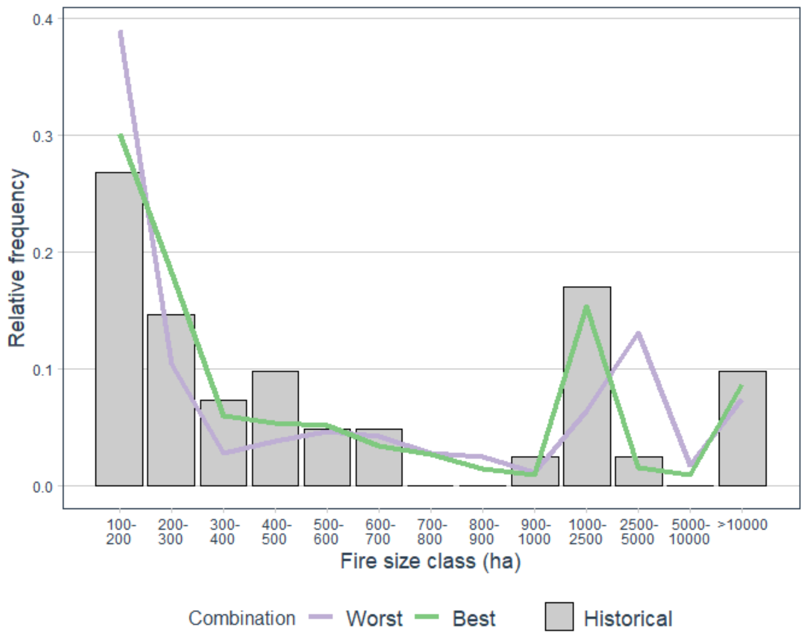


Figure 40 – Comparison of the combinations showing the poorest (blue) and the best (green) performance metrics.

10.3. Explaining the function *evaluate_BP_nxburned*

To understand the function evaluate_BP_nxburned, we can run:

```
?evaluate_BP_nxburned

evaluate_BP_nxburned(
  Folder.Outputs,
  freq.scenario,
  choose.combos,
  combos.file,
  obs.nxburned,
  export.plots)
```

`Folder.Outputs` corresponds to the path to the folder containing the outputs of the FConstMTT runs.

`freq.scenario` corresponds to the text file (csv) with the relative frequency of each meteorological and fuel map scenario used. If the function Gen_ign was used in the process, then the file to be used here should be "clusters_freqs_final.csv", which is located in the ignition folder.

`choose.combos` allows the user to specify the combinations to be tested. Because this process may require some time, the user can set it to "best" so only the combination with the lowest RMSE is used.

`combos.file` corresponds to the text file (csv) with the numeric identification of the different combinations stored as "performance_combos.csv".

`obs.nxburned` corresponds to the raster file with the historical number of times burned.

`export.plot` informs if a boxplot showing the correlation between the estimated burn probability and the number of times burned is saved.

Commented [a3]: se calhar isto não faz sentido. É o objectivo da função

As identified above, this function needs information on the historical number of times each pixel burned. This file must be created by the user prior to using this function. In this tutorial, this information was already produced.

Commented [a4]: era fixe meter aqui código onde se fazia isto

To visualize this raster file, do:

```
library(raster)
obs.nxburned<-raster("~/data/nxburned_monchique_reclass.asc")
plot(obs.nxburned)
```

Which will return the figure below.

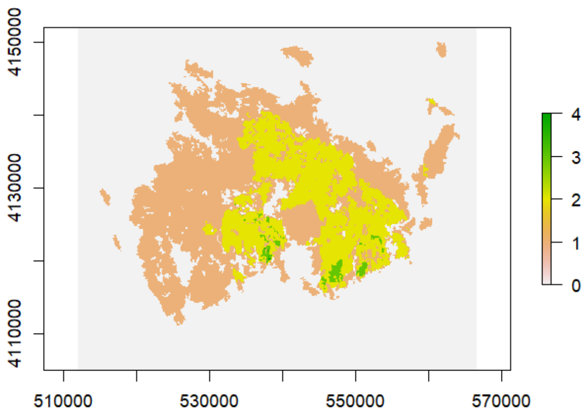


Figure 41 – Historical number of times burned

The number of times burned will be correlated with the estimated burn probability. Since we are replicating the historical fire pattern, using ignitions randomly sampled following the historical probability of ignition, and using the landscape of the years with the highest contribution to the burn area in the period of analysis, we expect that the estimated burn probability is well correlated with the historical number of times burned.

10.4. Using the function *evaluate_BP_nxburned*

To apply the function *evaluate_BP_nxburned* to the study case in this tutorial, simply:

```
evaluate_BP_nxburned(Folder.Outputs="~/characterization_manual_dur/Outputs",
       freq.scenario="~/characterization_manual_dur/ignitions/clusters_freqs_final.
csv",
       choose.combos="best",
       combos.file="~/characterization_manual_dur/Outputs/performance_combos.csv",
       obs.nxburned="~/data/nxburned_monchique_reclass.asc",
       export.plots=1)
```

Commented [a5]: eventualmente tirar. Não faz mesmo sentido

A new folder in the Outputs named correlation_BP_NxBurned is created after running this function. This folder contains a raster file of the estimated burn probability (BP_combo_32.asc), a png file showing the correlation between the estimated burn probability and the historical number of times burned (correlation_BP_nxburned_combo_32.png), a csv file with the Pearson correlation coefficient (correlation_BP_NxBurned.csv), and a csv file with the percentage of the landscape that burned X amount of times in the historical period (proportion_nxburned_original.csv).

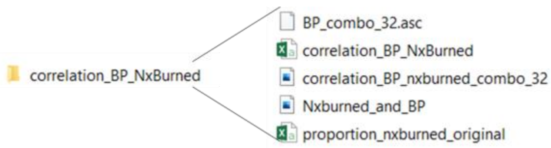


Figure 42 – Screenshot of the folder and files created by the function evaluate_BP_nxburned.

We can start by visualizing the estimated burn probability. Figure 43 shows the file Nxburned_and_BP.jpg. By simply comparing the both figures, it is possible to notice the overlap between the areas with highest number of times burned and with the highest burn probability. In some cases (as the one showed below), it is possible that the estimated burn probability is not a smooth surface. This is due to the low number of ignitions generated to speed-up the calibration process. To create reliable fire behaviour metrics, one must use more ignitions and use the duration values that correspond to the best combination / calibrated MTT.

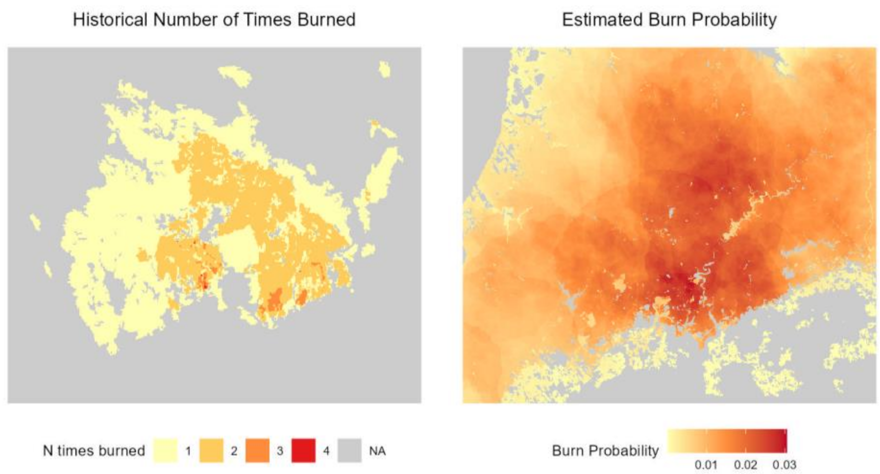


Figure 43 – Comparison between the historical number of times burned (left) and the estimated burn probability (right).

This visual comparison may be useful to detect deviations in the estimated burn probability. However, this simple comparison is not enough to conclude that the MTT model is accurately reproducing the historical spatial pattern. The file BP_nxburned_combo_32 (Figure 44) can be used to confirm if the MTT is reproducing the historical spatial pattern. The figure below shows the correlation boxplots of the simulated burn probability and the historical number of times burned. As expected, there is a positive correlation between the two variables, with areas that burned less times showing the smallest values of burn probability and vice-versa. Both variables have a Pearson Correlation Coefficient of 0.63 (reported in the file correlation_BP_NxBurned.csv).

Commented [a6]: melhorar o nome. Devia chamar-se boxplot_BP_nxburned ou assim

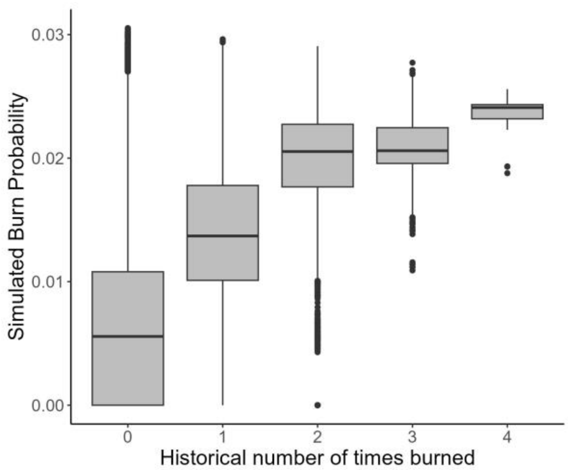


Figure 44 – Boxplots showing the correlation between the historical number of times burned and the simulated burn probability.

The final file saved is the proportion_nxburned_original.csv (Table 19). This file contains the proportion (as percentage) of the landscape in each class of number of times burned. In this example, the majority of the study area did not burn in the period of analysis (69%), around 21% burned once and around 10% burned twice. Areas that burned three or four times are rare, with both classes occupying less than 0.5%.

nxburned	percentage of landscape
0	69.021
1	20.702
2	9.857
3	0.412
4	0.007

Table 19 – Percentage of the landscape in each class of number of times burned.

11. References

Anderson, W. R., Cruz, M. G., Fernandes, P. M., McCaw, L., Vega,J. A., Bradstock, R. A., Fogarty, L., Gould, J., McCarthy, G., & Marsden-Smedley, J. B. (2015). A generic, empirical-based model for predicting rate of fire spread in shrublands. International Journal of Wildland Fire, 24,443–460. https://doi.org/10.1071/WF14130

Aparício, B. A., Benali, A., Pereira, J. M. C. & Sá, A. C. L. (2023). 'MTTfireCAL' package for R – An innovative, comprehensive and fast procedure to calibrate the MTT fire spread modelling system. Manuscript submitted for publication

Fernandes, P., Gonçalves, H., Loureiro, C., Fernandes, M., Costa, T., Cruz, M., Botelho, H. (2009) Modelos de combustível florestal para Portugal, in: Actas Do 6o Congr. Florest. Nac. Soc. Port. Ciências Florestais; SPCF Lisboa, Port. 2009. https://www.researchgate.net/profile/Paulo-Fernandes-6/publication/261708410_Modelos_de_Combustivel_Florestal_para_Portugal/links/00b7d53524bec08267000000/Modelos-de-Combustivel-Florestal-para-Portugal.pdf (accessed March 03, 2023).

Finney, M. A. (2002). Fire growth using minimum travel time methods. Canadian Journal of Forest Research. 32(8): 1420-1424. https://doi.org/10.1139/x02-068

Muñoz-Sabater, J., Dutra, E., Agustí-Panareda, A., Albergel, C., Arduini, G., Balsamo, G., Boussetta, S., Choulga, M., Harrigan, S., Hersbach, H., Martens, B., Miralles, D. G., Piles, M., Rodríguez-Fernández, N. J., Zsoter, E., Buontempo, C., & Thépaut, J.-N. (2021). ERA5-Land: a state-of-the-art global reanalysis dataset for land applications. Earth Syst. Sci. Data, 13(9), 4349–4383. https://doi.org/10.5194/essd-13-4349-2021

Nelson, R. M. J. (2000). Prediction of diurnal change in 10-h fuel stick moisture content. Canadian Journal of Forest Research, 30, 1071–1087. https://doi.org/10.1139/x00-032

Sá, A. C. L., Benali, A., Aparicio, B. A., Bruni, C., Mota, C., Pereira, J. M. C., Fernandes, P. M. (2023). A customized method to produce adaptable fuel models dataset. Manuscript submitted for publication

Scott, J. H. & Burgan, R. E. (2005). Standard Fire Behavior Fuel Models: A Comprehensive Set for Use with Rothermel's Surface Fire Spread Model; USDA Forest Service, Rocky Mountain Research Station: Fort Collins, CO, USA.