# EPIDEMIA Malaria Forecasting System: Detailed Walk-through Guide

For epidemia-scripts v1.3 using epidemiar v1.1.1

*Dawn Nekorchuk, Michael Wimberly, and EPIDEMIA Team Members*
*Department of Geography and Environmental Sustainability, University of Oklahoma*
*dawn.nekorchuk@ou.edu; mcwimberly@ou.edu*

*Updated April 16, 2019*

## Contents

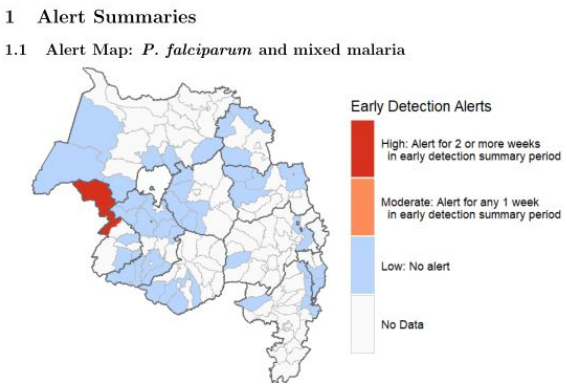


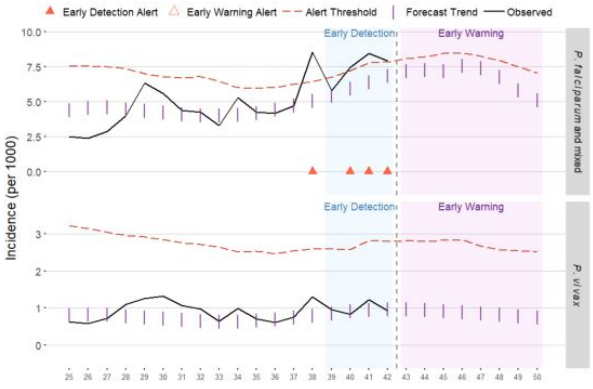
## 1 Introduction

This detailed walk-through will explain each section of the `run_report_amhara.R` script in the `epidemia-scripts` R project.

The final report presents environmental and epidemiological surveillance and forecasting results for 47 woredas in the Amhara region for the past 18 weeks through 8 weeks forecasted into the future.

Example sections of the final report:



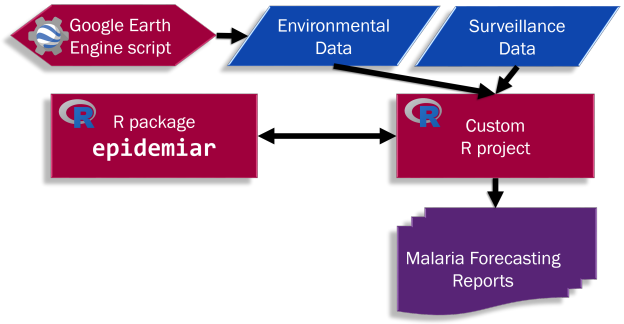




We are going to use an R script to:

- bring in malaria surveillance and remotely-sensed environmental data
- set some model and event detection parameters
- call the epidemiar functions to run the model, forecast, event detection, produce & save the report data ouput

- send the results to a formatting script (`report/epidemia_report_am.Rnw`, a Sweave file) to create the final pdf report.

This script is part of the larger EPIDEMIA Forecasting System. In the diagram below, the "Custom R Project" is `epidemia-scripts`.




# 2 R script

Open the `epidemia_scripts.Rproj` file to open the project in RStudio, and then open the `run_models_amhara.R` script. This script is divided into sections:

1. Loading packages & functions
2. Reading in the data
3. Set up forecast controls
4. Set up early detection controls
5. Run epidemia & create report data
6. Merge species data, save, and create pdf report

## 2.1 Loading packages & functions

First, we need to load the R packages and functions we will use in this R script.

```
## Load packages necessary for script

#make sure pacman is installed
if (!require("pacman")) install.packages("pacman")
```

```
## Loading required package: pacman
```

```
#load packages
  #note: data corrals use dplyr, tidyr, lubridate, readr, readxl
  #note: pdf creation requires knitr, tinytex
pacman::p_load(dplyr,
               knitr,
               lubridate,
               parallel,
               readr,
               readxl,
               tidyr,
               tinytex,
               tools)
library(epidemiar)
```

Using pacman is a convenient way of loading libraries (`library(package-name)`). If a package is missing, it will install it before continuing and prevent the script from stopping with an error.

We use `library()` for the epidemiar package, because this is our project-specific package and does not exist on the R package repository (CRAN), so pacman would fail trying to install epidemiar. See the "How to: Install & Update" documentation for how to install epidemiar.

There are also some local R functions, which are user-defined functions but not part of any package. The functions are in separate R script files that we `source()` so we can use the function. The data_corrals functions are how the epidemiological and environmental data are each read in and merged together to create datasets to be given to the epidemiar function for modeling. The report_save_create_helpers offer two functions, one to merge the results from *P. falciparum* & mixed species, and *P. vivax* model data, and another to send the result data to the formatting script to produce the pdf.

```
## Locally-defined Functions
source("R/data_corrals.R")
source("R/report_save_create_helpers.R")
```

## 2.2 Reading in the data

The epidemiar modeling and code requires 3 main sets of data: 1) epidemiological data, 2) daily environmental data, and 3) historical environmental reference/climatology data. A few look-up tables (metadata or informational reference sets) are needed as well.

### 2.2.1 Woreda metadata

First, we read in some information about the woredas in the Amhara region. We are going to create a subset of woredas that are going to be included in the report. A column named `report` in the Excel metadata file `data/woredas.xlsx` indicates if the woreda should be in the report. Currently, these are the 47 pilot woredas. To be included in the report, the woreda would need sufficient epidemiolgical data, environmental data, and model cluster ID. The woreda would have to have been included when fitting the model for cluster information and which environmental variables are needed. The list of report woredas is going to be used to filter the epidemiological and environmental data into a smaller dataset. The smaller dataset reduces processing time and the script will run faster.

```
# read in woreda metadata
report_woredas <- readxl::read_xlsx("data/woredas.xlsx", na = "NA") %>%
  dplyr::filter(report == 1)
```

### 2.2.2 Epidemiological data

For the epidemiology data, we will need weekly case counts per woreda of confirmed *P. falciparum* & mixed, and *P. vivax* malaria. To merge data from multiple files, we will use a "data corral" subfolder `data_epidemiological`.

The `corral_epidemiological()` function will merge all xlsx files in this directory, so you can have multiple files, e.g. one for each year. No special file names are expected, so you are free to use your standard naming convention. The script is expecting the file to be in your standard format. Specifically, it is looking for the fields: `Woreda/Hospital`,`Budget Year`, `Epi- Week`, `Blood film P. falciparum`, `RDT P. falciparum`, `Blood film P. vivax`, `RDT P. vivax`. For an example, the file `data_epidemiological\Malaria Data 2018 weeks 28-41.xlsx` is the file we were given during the 2018 workshop and what the script expects.

There should be a line for each week and woreda, even for missing data. Any missing (NA) values in the data will be filled in by linear interpolation inside of the epidemiar modeling functions. Gaps in the data, like missing weeks for a woreda, will trigger an error and stop the script. A log file of missing dates will be written, `log_missing_report_epidemiology.csv`, which can be opened with Excel.

Also in the same folder, there is a file `epi_data_20120712_20170708.xlsx` which contains the condensed epidemiological data from the EPIDEMIA project: 12 July 2012 through 7 July 2017. You may add your own files for some or all of this time period. The function will preferentially use your data.

No other files should be in this folder. An Excel file that is not epidemiological data will cause the script to fail.

The `corral_epidemiological()` function will loop through all of your xlsx files and combine them. Next, the script will append your data to the older EPIDEMIA data. Last, the function will remove any duplicates, choosing data from the file that was most recently modified. The `corral_epidemiological()` function is flexible enough to handle overlapping or partial year files, as long as there are no gaps in dates.

In regards to dates: Converion from Ethiopian date to Gregorian date was done by adding 7 (ISO week >= 28) or 8 (ISO week < 28) years, e.g. budget year 2011 is in year 2018 for weeks 28 - 52, and 2019 for weeks 1 - 27. Currently, no R function exists for converting full Ethiopian dates to full Gregorian calendar dates. The `obs_date` field is the last day of ISO-8601 week and is the standard date used in the epidemiar package.

The epidemiar package needs total malaria case counts per species. To create this variable, we will add the count of positive blood tests and the count of positive rapid diagnositic tests, for each species. `test_pf_tot = Blood film P. falciparum + RDT P. falciparum` and `test_pv_only = Blood film P. vivax + RDT P. vivax`.

The `corral_epidemiological()` function also needs three additional metadata files:

- Spelling crosswalk: `data/woreda_spellings.xlsx`: This is a crosswalk between the spellings found in the shapefile and the ones found in the epidemiological xlsx file. Additional alternative spellings can be added into this metafile as needed: `woreda_name` is the shapefile spelling, and `to_replace` is the alternative spelling, and a woreda may have multiple entries.
- Split woreda information: `data/woredas_split.xlsx`: This is a list of woredas that have split since the model was fitted and produced. Seperate environmental data does not yet exist for the split woredas. A new shapefile will need to be added to GEE first and historical information gathered. Model fitting with cluster identification was done on the pre-split woreda and until the model has been updated, the report can only be generated on the combined woreda. The split woredas will be added back together, to match the original woreda before the split. More splits can be added: `woreda_name` is the original woreda, and `split_1` and `split_2` are the two woredas that it was split into.
- Population data: `data/population_weekly_2012_2030.csv`: This population data is from the EPIDEMIA project: population living in malarious areas, called population at risk, and total population numbers. The population numbers were estimated out into the past and future, using an estimated population growth factor of 1.018.

The script will use the most recent week of data as the malaria report date, i.e. it will produce a report with forecasting starting the following future week.

```
# read & process case data needed for report
am_epi_data <- corral_epidemiological(report_woreda_names = report_woredas$woreda_name)
```

```
## Reading epidemiological data...
```

```
## Processing epidemiological data...
```

```
## Epidemiological data date range is 2012-07-15 to 2019-01-20 (YYYY-MM-DD).
```

### 2.2.3 Environmental data

For the environmental data, daily data is expected for each environmental variable for each woreda.

We are using the previous 181 days of environmental data in modeling the effects of the environmental variables to the case numbers, and so we must have environmental data at least that number of days *before* the first epidemiology data date. The included EPIDEMIA project environmental dataset has environmental data starting 2012 W1. We will set the lag length of 181 days later in the script.

Updated environmental data will be obtained from Google Earth Engine, which is a repository for many different types of satellite imagery and will process and summarize this data for us, so that we only have to

download a small text-based file.

The script will run with more recent epidemiological data than environmental data. Missing environmental data, up to the date of the last epidemiological data date, will be filled in using a persistence scheme. This scheme uses the last known value of the environmental variable and copies it forward in time for each missing week. If there are many values being estimated, the accuracy of the results may be reduced. We highly recommend getting the latest environmental data available before running the report.

The `corral_environment()` function works similarly to the epidemiolgy corral function `corral_epidemiological()`. The `corral_environment()` function is designed specifically for combining environmental data from Google Earth Engine and our previous EPIDEMIA environmental data.

All environmental data files are stored in subfolder `data_environmental/`. The `corral_environment()` function will loop through all csv (GEE) files and combine them. Next, the script will append the GEE data to the older EPIDEMIA data. Then the function will remove any duplicates, choosing data from the file that was most recently modified. Finally, it will check for any gaps in data. Missing days for any environmental variable will cause an error and stop the script. A log file of the missing days will be written, `log_missing_environmental.csv` which can be opened in Excel. To fix the error, run the GEE script for the missing dates and copy the files into the `data_environmental` subfolder.

Also in the `data_environmental/` folder is a file `env_data_20120101_20180930.RDS`. This RDS file is a R native format file, and contains all the EPIDEMIA project environmental data from 1 January 2012 through 30 September 2018. The RDS format is very small efficient so the file size is much smaller than a csv or xlsx file would be. This file may be replaced in the future by GEE data for the same time period.

Only GEE downloaded data in csv format should be added to the `data_environmental/` folder. An csv file that is not GEE-formated environmental data will cause the script to fail.

Planning for the far future, the function is flexible to accept new GEE variables and files. The new file would need: wid, doy, year, [env variable(s)]. There may be any number of variables, but they must have a unique name. To be included in modeling, the variable must have an entry in environ_info.xlsx and also be listed in the model variables, e.g. `falciparum_model.csv`. The `wid` field is the shapefile ID of the woreda used by GEE to get woreda-level data, and the `doy` field is the day of year, and `year` is the Gregorian calendar year.

The `corral_environment()` function takes a single argument, a tibble of woredas that will be included in the report. This list of woredas is used to filter the environmental data into a smaller subset of data. The smaller dataset will take less time to process and the script will run faster.

```
# read & process environmental data for woredas in report
am_env_data <- corral_environment(report_woredas = report_woredas)
```

```
## Reading environmental data...
```

```
## Processing environmental data...
```

```
## Environmental data date range (YYYY-MM-DD):
```

```
## # A tibble: 9 x 3
##   environ_var_code start_dt   end_dt
##   <chr>            <date>     <date>
## 1 evi              2001-01-01 2019-03-16
## 2 lst_day          2002-01-01 2019-03-21
## 3 lst_mean         2002-01-01 2019-03-21
## 4 lst_night        2002-01-01 2019-03-21
## 5 ndvi             2001-01-01 2019-03-16
## 6 ndwi5            2001-01-01 2019-03-16
## 7 ndwi6            2001-01-01 2019-03-16
## 8 savi             2001-01-01 2019-03-16
## 9 totprec          2001-01-01 2019-03-24
```

### 2.2.4 Environmental reference/climatology data

The environmental reference / climate data file `"data/env_ref_data.csv"` contains a average value for each the environmental variable for each woreda for each week of the year. These data are used with the most recently known daily environmental values to estimate future values of the environmental variables for forecasting. These data are also used to calculate anomalies. Anomalies are the difference between the current value and the historical average value. The anomalies are used in the modelling. The current reference file is a 17 year average (2001 - 2018).

```
# read in climatology / environmental reference data
am_env_ref_data <- read_csv("data/env_ref_data.csv", col_types = cols())
```

### 2.2.5 Environmental variable information

The environmental variable information file `data/environ_info.xlsx` lists the environmental variables and important information about the variable.

```
# read in environmental info file
am_env_info <- read_xlsx("data/environ_info.xlsx", na = "NA")
```

- `environ_var_code`: Short name for the environmental variable, using the GEE variable names
- `reference_method`: 'sum' or 'mean' for how to aggregate daily values into weekly values. For example, rainfall would be the 'sum' of the daily values, while LST would be the 'mean' value during that week
- `report_label`: The axis label to be used in creating the formatted report graphs
- `old_env_var_id`: the old EASTWeb/EPIDEMIA variable code name. Old data has been relabeled, this column is only for historical reference.

## 2.3 Set up Forecast controls

In this section, we first set the number of weeks in the report and set up for modeling and forecasting.

The malaria forecasting report will show a total of 26 weeks. We want to forecast eight (8) weeks out into the future from the last known epidemiological data date. The total 26 weeks is therefore 18 weeks of known data and 8 weeks of future forecasts.

```
#total number of weeks in report (including forecast period)
am_report_period <- 26

# forecast 8 weeks into the future
am_forecast_future <- 8
```

Next, we are going to read in two files that contain which environmental variables to use, and the woreda clustering information. The model is based on a general additive model (GAM) regression of multiple factors, including the woreda cluster, lagged environmental drivers (anomalies of the environmental variables), long terms trends, and seasonality.

```
#read in model environmental variables to use
am_pfm_model_env <- read_csv("data/falciparum_model_envvars.csv", col_types = cols())
am_pv_model_env <- read_csv("data/vivax_model_envvars.csv", col_types = cols())
```

- These are just a list of which `environ_var_code` variables to use in the modeling.

For the *P. falciparum* and mixed malaria model, the environmental variables are rainfall, daytime Land Surface Temperature (LST), and Normalized Difference Water Index (NDWI6; a satellite- derived index for vegetation water content). For the *P. vivax* model, the environmental variables are rainfall, the mean of daytime and nighttime LST, and NDWI6.

Woredas were clustered by the pattern of how the malaria incidence responds to the environmental variables. Woredas where these *interactions* between malaria incidence and environment variables were similar were placed in the same cluster. This gives us greater power for the forecast modeling because we have more datapoints in that cluster than in a single woreda alone.

```
#read in model cluster information
am_pfm_clusters <- read_csv("data/falciparum_model_clusters.csv", col_types = cols())
am_pv_clusters <- read_csv("data/vivax_model_clusters.csv", col_types = cols())
```

- The field `cluster_id` gives the cluster value for each woreda, by `woreda_name`.

Now we set some of the length parameters in the forecasting. We want to forecast eight (8) weeks out into the future from the last known epidemiological data date. We want to use the previous 181 days, the "lag length", of environmental data in modeling the effects of the environmental variables on malaria case numbers.

Each woreda and week is associated with environmental data on the day the week began, up to 180 days in the past, so that each woreda-week has a 181-day history of weather data. A distributed lag basis is created with the natural cubic splines function, including intercept, with knots at 25%, 50%, and 75% of the lag length. The 5 basis functions that result are multiplied by each woreda's history, so that there are just 5 summary statistics, instead of 181, for every combination of woreda, week, and environmental covariate.

```
#set maximum environmental lag length (in days)
am_lag_length <- 181
```

Next are a few technical settings for how the modeling itself is run. There is an option to fit the model week by week, but this is very slow, and is not noticeably different from fitting the model once in our data. We are also taking advantage of parallel processing. We are splitting the modeling into pieces that run *at the same time* on different processors on the computer.

```
#model fit frequency: fit once ("once), or fit every week ("week")
am_fit_freq <- "once"

#set number of cores to use on computer for parallel processing
  #default value is the number of physical cores minus 1, minimum 1 core.
am_cores <- max(detectCores(logical=FALSE) - 1, 1)
```

Finally, we just put all these forecasting controls into a list, so that we have one list object with all the controls for the forecasting.

```
#make control lists
am_pfm_fc_control <- list(env_vars = am_pfm_model_env,
                          clusters = am_pfm_clusters,
                          lag_length = am_lag_length,
                          fit_freq = am_fit_freq,
                          ncores = am_cores)

am_pv_fc_control <- list(env_vars = am_pv_model_env,
                         clusters = am_pv_clusters,
                         lag_length = am_lag_length,
                         fit_freq = am_fit_freq,
                         ncores = am_cores)
```

## 2.4 Set up Early Detection controls

In this section, we set up the parameters for the early detection algorithm.

The malaria forecasting report shows a total of 26 weeks. This will be 18 weeks of known data, and 8 weeks of forecasted values. Within the 18 weeks of known data, the most recent four (4) weeks are designated as

the 'early detection period'.

```
#number of weeks in early detection period
# (last n weeks of known epidemiological data to summarize alerts)
am_ed_summary_period <- 4
```

Next, we are going to set up the parameters used in the Farrington improved event detection algorithm. We will be using the `farringtonFlexible()` function as implemented in the `surveillance` package.

The central idea of event detection is to identify when the numer of cases exceeds a baseline threshold, and this detection happens as close to real-time as possible. Event detection is done in a prospective manner, where the algorithm knows past data up to the present. This is different from retrospective methods, where events are identified from historical data. There are many different prospective event detection algorithms. Each algorithm calculates baseline thresholds differently and have different assumptions about the pattern of disease transmission, speed of outbreak development, seasonality, and trends.

The Farrington Original method was developed in 1996 and an Improved version was released in 2013. The original method is used at Statens Serum Institut in Denmark, Centre for Infections of the Health Protection Agency (HPA) UK, National Institute for Public Health and the Environment (RIVM) the Netherlands (as of 2010). The Farrington methods are based on quasi-Poisson regression, and allows for long-term trend adjustments, seasonality, and re-weighting of past event case numbers.

The parameters below are the Farrington Improved versions we tested that had the highest percent of events caught and the lowest rate of false alarms. The Farrington method performed better than the other methods we compared: Center for Disease Control and Prevention (CDC) Early Aberation Reporting System (EARS) and the first EPIDEMIA system using dynamic linear modeling. We tested each species separately, so there are different settings for *P. falciparum* (and mixed) and *P. vivax*. The parameters are all added to a list so that we have one object that has all the controls for Farrington event detection.

```
#settings for Farrington event detection algorithm
am_pfm_ed_control <- list(
  w = 3, reweight = TRUE, weightsThreshold = 2.58,
  trend = TRUE, pThresholdTrend = 0,
  populationOffset = TRUE,
  noPeriods = 12, pastWeeksNotIncluded = 4,
  thresholdMethod = "nbPlugin")

am_pv_ed_control <- list(
  w = 4, reweight = TRUE, weightsThreshold = 2.58,
  trend = TRUE, pThresholdTrend = 0,
  populationOffset = TRUE,
  noPeriods = 10, pastWeeksNotIncluded = 4,
  thresholdMethod = "nbPlugin")
```

Farrington flexible settings:

- w: the number of timepoints in the window
- reweight: if identified past events are reweighted lower (so past events don't raise the new thresholds too high)
- weightsThreshold: the default value 2.58 in the Farrington revised algorithm that was found to be best performing by the authors who updated the algorithm
- trend: Use trend weighting over the past years
- pThresholdTrend: 0 value means always use trend adjustment
- populationOffset: Use population to adjust case numbers
- noPeriods: break up the year into 10 periods for modeling
- pastWeeksNotIncluded: Discount the first 4 weeks when testing for events to avoid incorrect results when an event is occuring right at the beginning of the period
- thresholdMethod: Use the recommended statistical method for calculating thresholds.

For more details, please run `?surveillance::farringtonFlexible` in the RStudio console to get the help file for this function.

## 2.5 Run epidemiar & create report data

Now to actually run the model and generate the report data! Basically, we gather up all the data and settings, and feed it to the `run_epidemia()` function in our `epidemiar` package. We are also setting that we want incidence reported as per 1000 persons.

Each malaria species is run on its own, with their respective settings. The main epidemiological dataset contains both species in different columns, and therefore the `casefield` changes between "test_pf_tot" for *P. falciparum* and mixed species, and "test_pv_only" for *P. vivax*. Similarly, the controls for event detection (`ed_control`) and forecasting (`am_pfm_fc_control`) also change.

(Note: Since this is a long script, and early error messages may have been missed, we've added a simple check to make sure the epidemiological and environmental datasets have been generated, and if not, it'll produce an informative message towards this end of the script.)

```
#Run modeling to get report data
# with check on current epidemiology and environmental data sets

if (exists("am_epi_data") & exists("am_env_data")){

  # P. falciparum & mixed
  message("Running P. falciparum & mixed")
  pfm_reportdata <- run_epidemia(epi_data = am_epi_data,
                                 casefield = test_pf_tot,
                                 populationfield = pop_at_risk,
                                 #incidence rates per 1000
                                 inc_per = 1000,
                                 groupfield = woreda_name,
                                 week_type = "ISO",
                                 report_period = am_report_period,
                                 ed_summary_period = am_ed_summary_period,
                                 ed_method = "Farrington",
                                 ed_control = am_pfm_ed_control,
                                 env_data = am_env_data,
                                 obsfield = environ_var_code,
                                 valuefield = obs_value,
                                 forecast_future = am_forecast_future,
                                 fc_control = am_pfm_fc_control,
                                 env_ref_data = am_env_ref_data,
                                 env_info = am_env_info)

  # P. vivax
  message("Running P. vivax")
  pv_reportdata <- run_epidemia(epi_data = am_epi_data,
                                casefield = test_pv_only,
                                populationfield = pop_at_risk,
                                #incidence rates per 1000
                                inc_per = 1000,
                                groupfield = woreda_name,
                                week_type = "ISO",
                                report_period = am_report_period,
                                ed_summary_period = am_ed_summary_period,
                                ed_method = "Farrington",
```

```
                                ed_control = am_pfm_ed_control,
                                env_data = am_env_data,
                                obsfield = environ_var_code,
                                valuefield = obs_value,
                                forecast_future = am_forecast_future,
                                fc_control = am_pv_fc_control,
                                env_ref_data = am_env_ref_data,
                                env_info = am_env_info)

} else {
  message("Error: Epidemiological and/or environmental datasets are missing.
          Check Section 2 for data error messages.")
}
```

```
## Running P. falciparum & mixed

## Preparing for forecasting

## Generating forecasts

## Running early detection

## Running P. vivax

## Preparing for forecasting

## Generating forecasts

## Running early detection
```

Depending on the power of your computer, this could take anywhere from 3 to 15 minutes or so for each species.

### 2.5.1 Report data output

The `run_epidemia()` function returns one object. For *P. falciparum* we called this object `pfm_reportdata` and `pv_repordata` for *P. vivax*. This object is a list of tibbles, or dataframes. These tibbles are the outputs from the modeling, early detection and early warning algorithms, and other results.

Looking at the list of objects:

```
str(pfm_reportdata, max.level = 1)
```

```
## List of 7
##  $ summary_data         :Classes 'tbl_df', 'tbl' and 'data.frame':   47 obs. of  5 variables:
##  $ epi_summary          :Classes 'tbl_df', 'tbl' and 'data.frame':   47 obs. of  2 variables:
##  $ modeling_results_data:Classes 'tbl_df', 'tbl' and 'data.frame':   4512 obs. of  9 variables:
##  $ environ_timeseries   :Classes 'tbl_df', 'tbl' and 'data.frame':   3666 obs. of  16 variables:
##  $ environ_anomalies    :Classes 'tbl_df', 'tbl' and 'data.frame':   141 obs. of  3 variables:
##  $ params_meta          :List of 8
##  $ regression_object    :List of 55
##   ..- attr(*, "class")= chr [1:4] "bam" "gam" "glm" "lm"
```

#### 2.5.1.1 summary_data

```
pfm_reportdata$summary_data
```

```
## # A tibble: 47 x 5
##    woreda_name     ed_alert_count ed_sum_level ew_alert_count ew_level
##    <chr>                    <dbl> <ord>                 <dbl> <ord>
```

```
##  1 Abargelie                    0 Low                      0 Low
##  2 Alefa                        0 Low                      0 Low
##  3 Andabiet                     0 Low                      0 Low
##  4 Ankesha                      0 Low                      0 Low
##  5 Antsokiya Gemza              0 Low                      0 Low
##  6 Artuma Fursi                 0 Low                      0 Low
##  7 Awabel                       0 Low                      0 Low
##  8 Bahir Dar Zuria              0 Low                      0 Low
##  9 Baso Liben                   0 Low                      0 Low
## 10 Borena                       0 Low                      0 Low
## # ... with 37 more rows
```

This tibble contains the early detection and early warning alert levels for each woreda.

Early detection alerts (`ed_alert_count`) are alerts that are triggered during the early detection period, which is defined as the 4 most recent weeks of known epidemiology data. Similarly, early warning alerts (`ew_alert_count`) are alerts in the future forecast estimates. "High" level indicates two or more weeks in this period had incidences greater than the alert threshold, "Medium" means that one week was in alert status, and "Low" means no weeks had alerts (`ed_sum_level` and `ew_level`, respectively).

#### 2.5.1.2 epi_summary

```
pfm_reportdata$epi_summary
```

```
## # A tibble: 47 x 2
##    woreda_name     mean_inc
##    <chr>              <dbl>
##  1 Abargelie        1.20
##  2 Alefa            0.376
##  3 Andabiet         0.0405
##  4 Ankesha          0.152
##  5 Antsokiya Gemza  0.00756
##  6 Artuma Fursi     0.0147
##  7 Awabel           0.0354
##  8 Bahir Dar Zuria  0.141
##  9 Baso Liben       0.649
## 10 Borena           0.0358
## # ... with 37 more rows
```

This tibble holds the mean incidence of malaria in the early detection period, and is used to generate maps in the third section of the pdf report.

#### 2.5.1.3 modeling_results_data

```
pfm_reportdata$modeling_results_data
```

```
## # A tibble: 4,512 x 9
##    woreda_name obs_date   series value lab   upper lower week_epidemiar
##    <chr>       <date>     <chr>  <dbl> <chr> <dbl> <dbl>          <dbl>
##  1 Abargelie   2018-09-23 obs    0.787 Obse~    NA    NA             38
##  2 Abargelie   2018-09-30 obs    0.844 Obse~    NA    NA             39
##  3 Abargelie   2018-10-07 obs    1.31  Obse~    NA    NA             40
##  4 Abargelie   2018-10-14 obs    0.975 Obse~    NA    NA             41
##  5 Abargelie   2018-10-21 obs    2.36  Obse~    NA    NA             42
##  6 Abargelie   2018-10-28 obs    2.74  Obse~    NA    NA             43
##  7 Abargelie   2018-11-04 obs    2.79  Obse~    NA    NA             44
##  8 Abargelie   2018-11-11 obs    3.24  Obse~    NA    NA             45
```

```
##  9 Abargelie   2018-11-18 obs    3.84  Obse~    NA    NA             46
## 10 Abargelie   2018-11-25 obs    2.59  Obse~    NA    NA             47
## # ... with 4,502 more rows, and 1 more variable: year_epidemiar <dbl>
```

This tibble dataset contains multiple timeseries values for observed, forecast, and alert thresholds of malaria incidence, for each woreda. These data are used in creating the individual woreda control charts in the pdf report.

#### 2.5.1.4 environ_timeseries

```
pfm_reportdata$environ_timeseries
```

```
## # A tibble: 3,666 x 16
##    woreda_name environ_var_code year_epidemiar week_epidemiar obs_date
##    <chr>       <chr>                     <dbl>          <dbl> <date>
##  1 Abargelie   lst_day                    2018             38 2018-09-23
##  2 Abargelie   ndwi6                      2018             38 2018-09-23
##  3 Abargelie   totprec                    2018             38 2018-09-23
##  4 Abargelie   lst_day                    2018             39 2018-09-30
##  5 Abargelie   ndwi6                      2018             39 2018-09-30
##  6 Abargelie   totprec                    2018             39 2018-09-30
##  7 Abargelie   lst_day                    2018             40 2018-10-07
##  8 Abargelie   ndwi6                      2018             40 2018-10-07
##  9 Abargelie   totprec                    2018             40 2018-10-07
## 10 Abargelie   lst_day                    2018             41 2018-10-14
## # ... with 3,656 more rows, and 11 more variables: val_epidemiar <dbl>,
## #   reference_method <chr>, data_source <chr>, ref_value <dbl>,
## #   ref_sd <dbl>, ref_n <dbl>, ref_max <dbl>, ref_uq <dbl>,
## #   ref_median <dbl>, ref_lq <dbl>, ref_min <dbl>
```

This tibble dataset contains multiple timeseries for the environmental variables for each woreda, and are used to generate the environmental timeseries graphs on the individual woreda report pages.

#### 2.5.1.5 environ_anomalies

```
pfm_reportdata$environ_anomalies
```

```
## # A tibble: 141 x 3
##    woreda_name environ_var_code anom_ed_mean
##    <chr>       <chr>                   <dbl>
##  1 Abargelie   lst_day              0.459
##  2 Abargelie   ndwi6               -0.00511
##  3 Abargelie   totprec             -8.44
##  4 Alefa       lst_day              1.66
##  5 Alefa       ndwi6                0.00640
##  6 Alefa       totprec             -9.44
##  7 Andabiet    lst_day              0.402
##  8 Andabiet    ndwi6               -0.000361
##  9 Andabiet    totprec            -10.6
## 10 Ankesha     lst_day              1.56
## # ... with 131 more rows
```

This tibble dataset contains the differences of the environmental variable values from the climatology/reference average during the early detection period. These data are used to make anomaly maps in the third section of the pdf report.

#### 2.5.1.6 params_meta

This lists all the dates, settings, and parameters that were used in the `run_epidemiar()` function. This keeps a record of all the settings so you can view them later.

#### 2.5.1.7 regression_object

This is the regression object from the general additive model (GAM, parallelized with BAM) regression. This is only for statistical investigation of the model, and is usually not saved because it very large object.

### 2.6 Merge species data & save

We now have two different reportdata objects ( *P. falciparum* and mixed species, and *P. vivax*), which need to be merged together into one object to save and to create the single pdf report.

(Note: Since this is a long script, and early error messages may have been missed, we've added a simple check to make sure the report data for each species have been generated, and if not, it'll produce an informative message towards this end of the script.)

```
if (exists("pfm_reportdata") & exists("pv_reportdata")){

  #merging pfm & pv data, save out, and create pdf
  merge_save_report(rpt_data_main = pfm_reportdata,
                    rpt_data_secd = pv_reportdata,
                    #mark sections as P. falciparum and mixed (pfm) or P. vivax (pv)
                    # used in the epidemia_report_am.Rnw file for the formatted report
                    var_labs = c("pfm","pv"),
                    #save out the report data in the file that the formatting file reads
                    save_file = "report/report_data.RData",
                    #save out a second copy of the report data with year and week numbers in the name f
                    second_save = TRUE,
                    #create the pdf
                    create_report = TRUE,
                    #which Rnw file to use to create pdf
                    formatting_file = "epidemia_report_am.Rnw",
                    #show the pdf immediately after creating
                    show_report = TRUE)

} else {
  message("Error: Report data for P. falciparum and/or P. vivax have not been generated and are missing
}
```

```
## Saved results to /report/report_data.RData
```

```
## Saved results to /report/report_data_2019W3.RData
```

```
## Saved report to /report/epidemia_report_am_2019W3.pdf
```

This function will save out the merged report_data object in two places:

- `save_file = "report/report_data.RData"`: this file is the *input* file of the Rnw/formatting script (`epidemia_report_am.Rnw`). The input file of the rnw cannot be particularly changed except by editing the rnw file directly. Therefore this is a 'generic' named file, which is overwritten each time to be used as input to the rnw file.
- `second_save = TRUE`: this saves out a more permanent version of the merged report_data object with an autogenerated year and week numbers in the file name (e.g. "report_data_2018W52.RData" for the report generated with last known epidemiological data of week 2018 Week 52: 2019-12-24 through 2018-12-30.)

This function calls a subfunction `create_pdf()` which calls the Sweave formatting script, `report/epidemia_report_am.Rnw`, to read in the report data (`report/report_data.RData`, as specified in the rnw file) we just created, and to create the formatted pdf report. This `Rnw` Sweave file has been written specifically for the malaria forecasting report in Amhara. The `Rnw` file also uses some additional data: shapefiles (processed into R data files as `data/am.rda` and `data/am_simpl.rda`), and the woreda reference list (`data/woredas.xlsx`).

This subfunction will save two copies of the pdf formatted report:

- `epidemia_report_am.pdf`: default output of an rnw file - name of the rnw file as a pdf
- `epidemia_report_{YYYY"W"WW}.pdf`: a renamed version of the file with autogenerated year and week numbers in the file, same scheme as the second_save of the report_data object, above.

For reference, here is a list of report woreda names:

```
readxl::read_xlsx("data/woredas.xlsx", na = "NA") %>%
  filter(report == 1) %>% pull(woreda_name)
```

```
##  [1] "Abargelie"         "Alefa"             "Andabiet"
##  [4] "Ankesha"           "Antsokiya Gemza"   "Artuma Fursi"
##  [7] "Awabel"            "Bahir Dar Zuria"   "Baso Liben"
## [10] "Borena"            "Bugna"             "Burie Zuria"
## [13] "Debre Elias"       "Dehena"            "Dembecha"
## [16] "Denbia"            "Dera"              "Efratana Gidim"
## [19] "Estea"             "Fagita Lekoma"     "Fogera"
## [22] "Gondar Zuria"      "Gonji Kolela"      "Gozamin"
## [25] "Guagusa Shekudad" "Jabi Tehnan"       "Jawi"
## [28] "Jilie Timuga"      "Kalu"              "Kewet"
## [31] "Kobo Town"         "Lasta"             "Libokemkem"
## [34] "Mecha"             "Mekit"             "Merehabete"
## [37] "Metema"            "Misrak Belesa"     "North Achefer"
## [40] "Quara"             "Raya Kobo"         "Sehela"
## [43] "Shewa Robit"       "South Achefer"     "Tehulederie"
## [46] "Womberma"          "Yilmana Densa"
```