Butterfly Monitoring and Analyses

Reto Schmucki

2024-07-22

Table of contents

Preface

This project aims to collate documentation of methods used to analyze and work with Butterfly Monitoring Schemes (BMS) data. We focus on methods used to analyze count data associated with Pollard transects that represent the core of the eBMS database.

> **NOTE - this book project is a work in progress that only started.**
>
> Reto Schmucki, July 2024

Part I

Counting Butterflies

Butterfly Monitoring

Content under construction

Part II

From Counts to FLight Curves

Butterfly Counts

Data Simulation

Why rely on simulated data?

Simulation of Butterfly Counts

To demonstrate and assess the method used to compute butterfly abundance indices and the Grassland Butterfly Indicator which involves calculating collated abundance indices from multiple sites and estimating the population trend from a set of indicator species, the best option is to use simulated datasets that provide realistic data with known parameters. The simulation approach allows us to assess the methods' performance and enables us to control individual parameters and conduct rigorous sensitivity analysis. This will provide useful insight into the method and enable rigorous assessment of its power and limitations.

Simulation Tool for Butterflies and Other Phenologies

To simulate butterfly count across sites and over multiple iterations, we will use the function developed by Collin Edwards and available in the R packages butterflyGamSims.

```r
if(!require("data.table")) install.packages("data.table")
if(!require("ggplot2")) install.packages("ggplot2")
if(!require("devtools")) install.packages("devtools")
if(!require("rbms")) devtools::install_github("RetoSchmucki/rbms")
if(!require("butterflyGamSims")) devtools::install_github("cbedwards/butterflyGamSims")

flc_col <- 'orange'
cnt_col <- 'darkcyan'
missing_col <- 'red'
GAM_col <- '#483d8b'
```

In the first case, we will simulate counts for one site for one year, using a Gaussian curve to depict the adult butterflies' seasonal phenology (activity curve). The daily counts are generated from a random Poisson process.

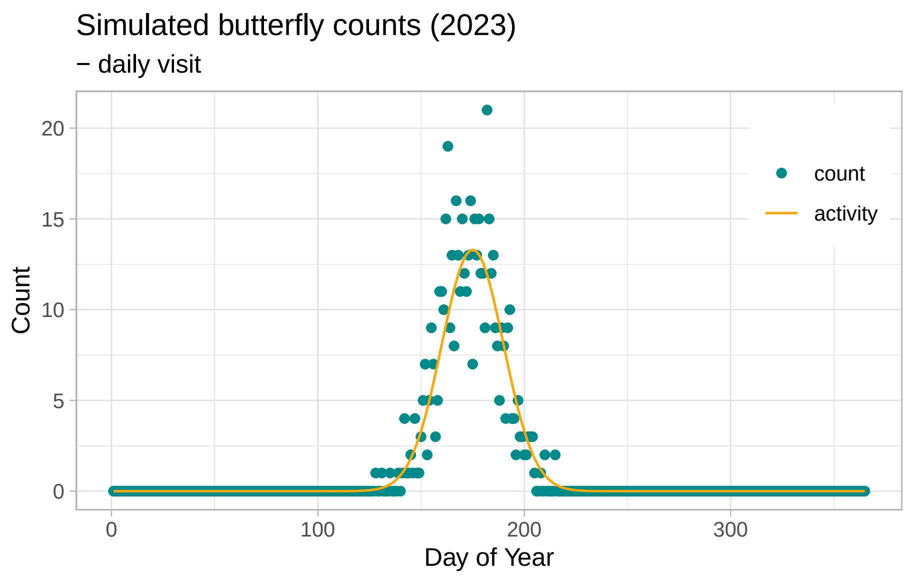
```
btfl_data <- timeseries_sim(nsims=1,
               year = c(2023:2025),
               doy.samples = seq(from=1, to=365, by=1),
               abund.type = "exp",
               activity.type = "gauss",
               sample.type = "pois",
               sim.parms = list(growth.rate = 0,
                                init.size = 500,
                                act.mean = 175,
                                act.sd = 15)
               )
```

Simulate Data Sampling Process

```
# data: Time series resulting from the simulation generated for 365 days
# weeklySample: TRUE or FALSE; should the daily count in the time series be resample
# weekdayKeep: vector of days c(1,2,..., 7) to be sampled from for the weekly count.
# contains c(2,3,4), the sampling process will be restricted to Tuesday, Wednesday c
# monitoringSeason: vector of months that define the monitoring season (e.g. April t

sim2bms <- function(data, yearKeep = NULL, weeklySample = FALSE, weekdayKeep = NULL,

                  btfl_ts <- data.table::data.table(data)[, site_id := paste0("site_
                  if(!is.null(yearKeep)){
                        btfl_ts <- btfl_ts[years %in% yearKeep, ]
                  }
                  btfl_ts[ , date := as.Date(doy, origin = paste0(years, "-01-01"))-
                  btfl_ts[ , week := isoweek(date)]
                  btfl_ts[month(date) != 1 | week < 50, weekday := rowid(week), by =
                  if(isTRUE(weeklySample)){
                        if(!is.null(weekdayKeep)){
                              btfl_ts <- btfl_ts[weekday %in% weekdayKeep, ]
                        }
                        btfl_ts <- btfl_ts[btfl_ts[,.I[sample(.N, 1)]], by = .(week,
                  }
                  if(!is.null(monitoringSeason)){
                  btfl_ts <- btfl_ts[month(date) %in% monitoringSeason, ]
                  }
            return(btfl_ts)
```

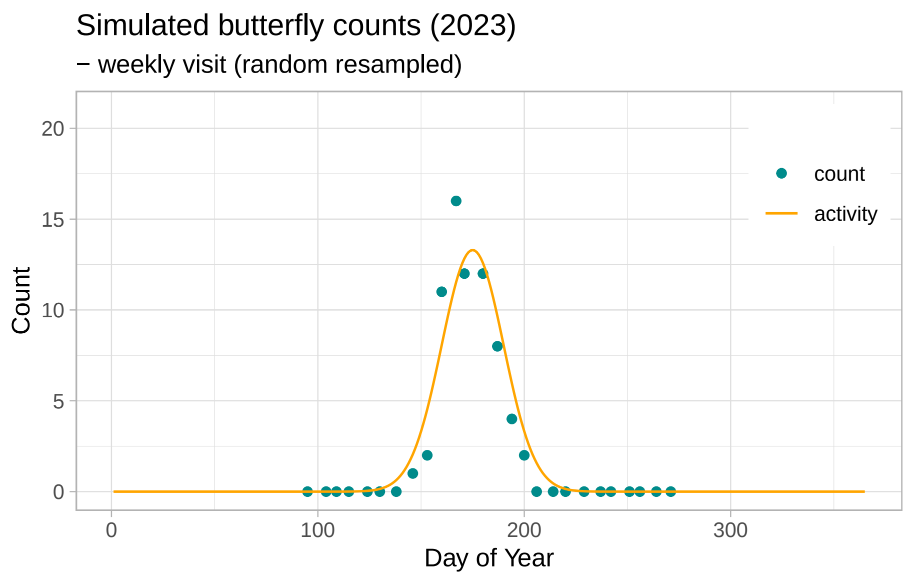
```
              }
```

```
y <- c(2023)

btfl_ts <- sim2bms(data = btfl_data$timeseries, yearKeep = y)

btfl_fig <- ggplot() +
                geom_point(data=btfl_ts, aes(x=doy, y=count, colour = "count")) +
                geom_line(data = btfl_ts,
                aes(x = doy, y = act, colour = "activity")) +
                xlim(1,365) + ylim(0, max(btfl_ts$count, btfl_ts$act)) +
                scale_colour_manual("",
                      breaks = c("count", "activity"),
                      values = c(cnt_col, flc_col)) +
                theme_light() +
                theme(legend.position = "inside", legend.position.inside = c(0.9, 0.8)) +
                labs(title = paste0("Simulated butterfly counts (", y,")"),
                     subtitle = "- daily visit",
                     x = "Day of Year",
                     y = "Count")
btfl_fig
```



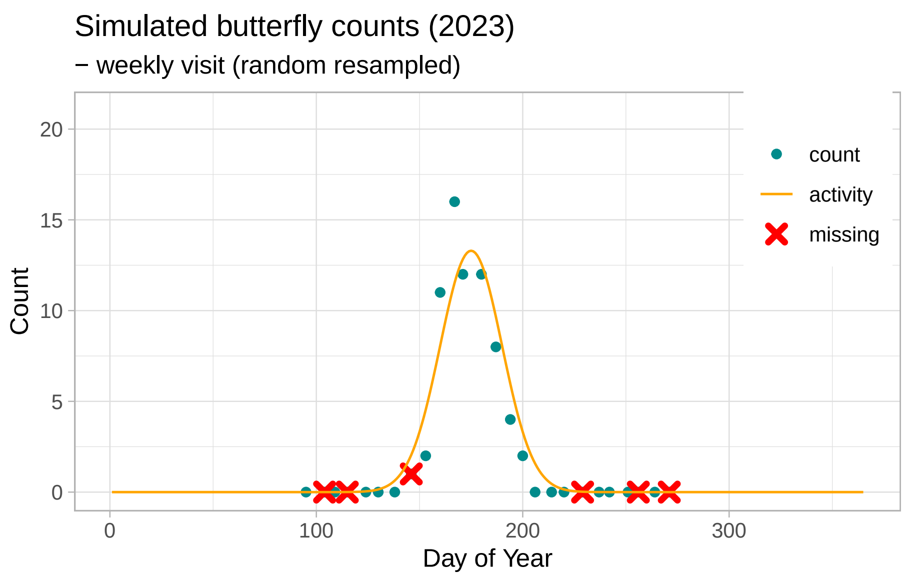
In the example above, the activity curve represented by the line has a Gaussian shape and counts presented by the points along the curve are independent random samples from a Poisson distribution. Because we sampled a count value for 365 days (day-of-year; doy), the counts are representative of the population of active adult butterflies as if the site was visited every. This implies that a proportion of butterflies are counted more than one day as their lifespan exceeds one day. On Pollard transect, this is how butterfly counts are likely to be counted and reported, but with a different frequency as visits are generally weekly, fortnightly, or even monthly. We can replicate this value by resampling the daily count weekly.

```r
set.seed(13276)

btfl_week_smpl <- sim2bms(data = btfl_data$timeseries, yearKeep = y,
                  weeklySample = TRUE,
                  weekdayKeep = c(2:5),
                  monitoringSeason = c(4:9))

btfl_fig2 <- ggplot() +
                geom_point(data=btfl_week_smpl, aes(x=doy, y=count, colour = "count'
                geom_line(data = btfl_ts,
                aes(x = doy, y = act, colour = "activity")) +
                xlim(1,365) + ylim(0, max(btfl_ts$count, btfl_ts$act)) +
                scale_colour_manual("",
                     breaks = c("count", "activity"),
                     values = c(cnt_col, flc_col)) +
                theme_light() +
                theme(legend.position = "inside", legend.position.inside = c(0.9, 0.
                labs(title = paste0("Simulated butterfly counts (", y,")"),
                     subtitle = "- weekly visit (random resampled)",
                     x = "Day of Year",
                     y = "Count")
btfl_fig2
```



From the weekly visits, counts outside of the monitoring period will not be informed, in many cases these are 'zeros' as we expect the monitoring season to align with butterflies' activity. Some other weeks might be missing from the time series, potentially due to unsuitable weather conditions for monitoring or recorder's unavailability. We can include these missing visits by resampling a subset of the weekly counts.

```
missing_prob <- function(data, mu=NULL, alpha = 5, theta = 0.3){
                              x_ <- seq_len(nrow(data))
                              mu_ <- ifelse(is.null(mu), length(x_) / 2, mu)
                              std_ <- sqrt(mu_ / theta)
                              y_ <- abs((alpha * exp((-(x_ - mu_)^2) / std_^2)) - alpha) + alp
                              yn_ <- y_ / (sum(y_))
                        return(yn_)
                  }

sample_missing <- function(data, propMissing = 0.25){

            missing.prob <- data.table::data.table()
            for(i in data[, unique(years)]){
                  for(j in data[, unique(site_id)]){
                  missing.prob <- rbind(missing.prob, missing_prob(data[years == i & site_id =
                  }
            }

            missing.week <- data[sample(seq_len(.N), round(propMissing * .N), prob = unlist(mi
      return(missing.week)
}

btfl_week_missing <- sample_missing(data = btfl_week_smpl, propMissing = 0.25)

btfl_fig3 <- ggplot() +
                geom_point(data=btfl_week_smpl, aes(x=doy, y=count, colour = "count'
                geom_point(data=btfl_week_missing, aes(x=doy, y=count, colour = "mis
                            shape=4, size=2, stroke=2) +
                geom_line(data = btfl_ts,
                aes(x = doy, y = act, colour = "activity")) +
                xlim(1,365) + ylim(0, max(btfl_ts$count, btfl_ts$act)) +
                scale_colour_manual("",
                      breaks = c("count", "activity", "missing"),
                      values = c(cnt_col, flc_col, missing_col)) +
                theme_light() +
                theme(legend.position = "inside", legend.position.inside = c(0.9, 0.
                labs(title = paste0("Simulated butterfly counts (", y,")"),
                     subtitle = "- weekly visit (random resampled)",
                     x = "Day of Year",
                     y = "Count")
btfl_fig3
```



Generalized Additive Models with rbms

We will use the simulation to test the GAM method implemented in the R package rbms. Because recorders only report the number of observed butterflies, zeros are generally not reported but can be derived from the visit dates.

Organising BMS count data

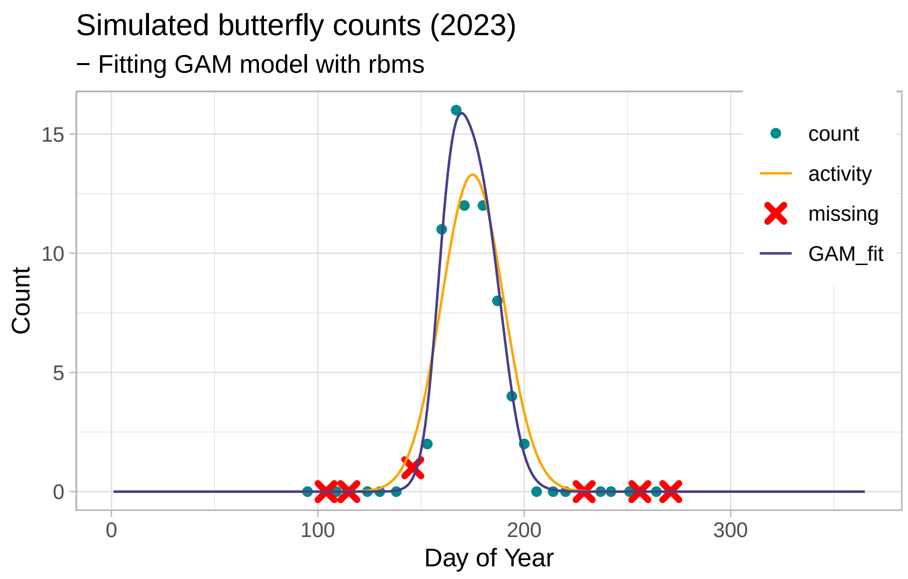
```r
visit_sim <- btfl_week_smpl[!date %in% btfl_week_missing$date, .(site_id, date, count)]
count_sim <- visit_sim[count>=1,][, species := "sp1"]

names(visit_sim) <- toupper(names(visit_sim))
names(count_sim) <- toupper(names(count_sim))

ts_date <- rbms::ts_dwmy_table(InitYear = 2023, LastYear = 2023, WeekDay1 = 'monday')

ts_season <- rbms::ts_monit_season(ts_date,
                      StartMonth = 4,
                      EndMonth = 9,
                      StartDay = 1,
                      EndDay = NULL,
                      CompltSeason = TRUE,
                      Anchor = TRUE,
                      AnchorLength = 2,
                      AnchorLag = 2,
                      TimeUnit = 'd')

ts_season_visit <- rbms::ts_monit_site(ts_season, visit_sim)

ts_season_count <- rbms::ts_monit_count_site(ts_season_visit, count_sim, sp = "sp1")
```

Fitting a GAM to Butterfly Counts

```r
ts_flight_curve <- rbms::flight_curve(ts_season_count,
                      NbrSample = 300,
                      MinVisit = 5,
                      MinOccur = 3,
                      MinNbrSite = 1,
                      MaxTrial = 4,
                      GamFamily = 'nb',
                      SpeedGam = FALSE,
                      CompltSeason = TRUE,
                      SelectYear = NULL,
                        TimeUnit = 'd')
```

The flight curve computed by the `rbms::flight_curve()` function is stored in the ...$pheno object where the days of year are stored under the variable `trimDAYNO` and the standardized flight curve under the variable `NM`. The `NM` variable is scaled to an Area Under the Curve (AUC) that sum to 1. To compare the flight curve derived from the GAM with the activity curve used for the simulation, we must rescale them to the same AUC, in other words, we must rescale the activity curve to have an AUC of 1 or rescale the `NM` to the population size used for the simulation. Here we will rescale the `NM` to match the simulation population size, this will allow us to display the curves and the counts on the same plot with the correct scale.

```
pheno <- ts_flight_curve$pheno

btfl_fig4 <- ggplot() +
               geom_point(data=ts_season_count[ANCHOR == 0 & !is.na(COUNT), ], aes(
               geom_point(data=btfl_week_missing, aes(x=doy, y=count, colour = "mis
                           shape=4, size=2, stroke=2) +
               geom_line(data = btfl_ts, aes(x = doy, y = act, colour = "activity")
               geom_line(data = pheno,
                   aes(x = trimDAYNO, y = btfl_ts[,unique(abund.true)]*NM, colour =
               xlim(1,365) + ylim(0, max(btfl_ts$act,
                                          pheno$NM*btfl_ts[,unique(abund.true)],
                                          btfl_week_missing$count,
                                          ts_season_count[!is.na(COUNT), COUNT] )) +
               scale_colour_manual("",
                     breaks = c("count", "activity", "missing", "GAM_fit"),
                     values = c(cnt_col, flc_col, missing_col, GAM_col)) +
               theme_light() +
               theme(legend.position = "inside", legend.position.inside = c(0.9, 0.
               labs(title = paste0("Simulated butterfly counts (", y,")"),
                    subtitle = "- Fitting GAM model with rbms",
                    x = "Day of Year",
                    y = "Count")
btfl_fig4
```



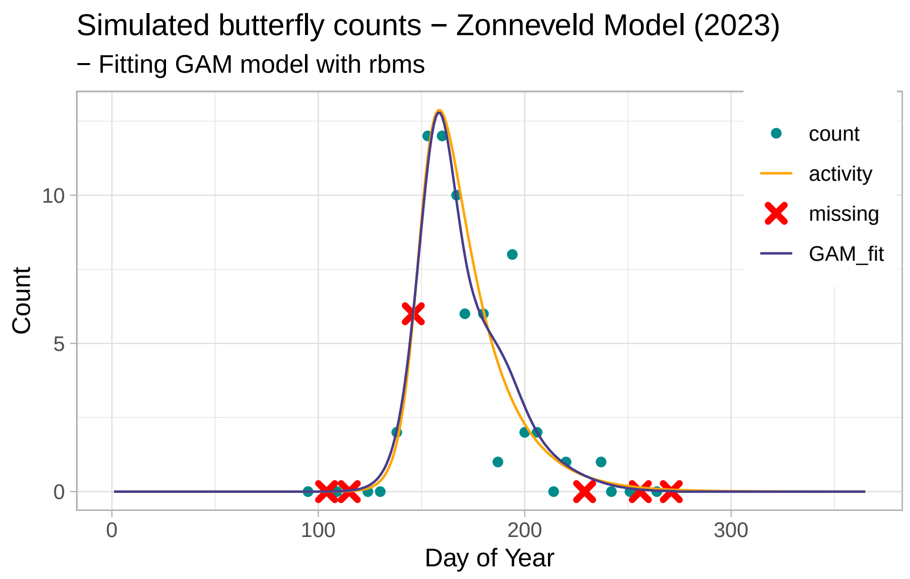
To compare the fitted curve with the activity curve, we should use a standard AUC of 1 to enable a fair comparison between models fitted to different population sizes. Using the standardized activity curve and the GAM-generated flight curve (`NM`), we can calculate the Root Mean Squared Error (RMSE) to estimate the goodness of fit of the flight curve generated with the rbms package.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)}$$

Non Gaussian flight curve

The same procedure can be applied to flight curves having more complex shapes. Here we will generate a time series of butterfly counts drawn from a known flight curve (adult activity), using simulation from a Zonneveld model.

```
btfl_data_zn <- timeseries_sim(nsims=1,
               year = c(2023),
               doy.samples = seq(from=1, to=365, by=1),
               abund.type = "exp",
               activity.type = "zon",
               sample.type = "pois",
               sim.parms = list(growth.rate = 0,
                                init.size = 500,
                                act.mean = 175,
                                act.sd = 15,
                                  #theta = 5,
                                  zon.theta = 50,
                                  t0 = 100,
                                  beta = 5,
                                  alpha = 0.05)
                 )


set.seed(13276)
btfl_ts <- sim2bms(data = btfl_data_zn$timeseries, yearKeep = y)

btfl_week_smpl <- sim2bms(data = btfl_data_zn$timeseries, yearKeep = y,
                  weeklySample = TRUE,
                  weekdayKeep = c(2:5),
                  monitoringSeason = c(4:9))

btfl_week_missing <- sample_missing(data = btfl_week_smpl, propMissing = 0.25)

visit_sim <- btfl_week_smpl[!date %in% btfl_week_missing$date, .(site_id, date, cour
count_sim <- visit_sim[count>=1,][, species := "sp1"]

names(visit_sim) <- toupper(names(visit_sim))
names(count_sim) <- toupper(names(count_sim))

ts_date <- rbms::ts_dwmy_table(InitYear = 2023, LastYear = 2023, WeekDay1 = 'monday'

ts_season <- rbms::ts_monit_season(ts_date,
                       StartMonth = 4,
                       EndMonth = 9,
                       StartDay = 1,
                       EndDay = NULL,
                       CompltSeason = TRUE,
                       Anchor = TRUE,
                       AnchorLength = 2,
                       AnchorLag = 2,
                       TimeUnit = 'd')

ts_season_visit <- rbms::ts_monit_site(ts_season, visit_sim)

ts_season_count <- rbms::ts_monit_count_site(ts_season_visit, count_sim, sp = "sp1")

# mod_k <- "COUNT ~ s(DAY_SINCE, bs =\"cr\", k = 5) + factor(SITE_ID)"

ts_flight_curve <- rbms::flight_curve(ts_season_count,
                         NbrSample = 300,
                         MinVisit = 5,
                         MinOccur = 3,
                         MinNbrSite = 1,
                         MaxTrial = 4,
                         GamFamily = 'nb',
                         SpeedGam = FALSE,
                         CompltSeason = TRUE,
                         SelectYear = NULL,
                         #mod_form = mod_k,
                         TimeUnit = 'd')

pheno <- ts_flight_curve$pheno

btfl_fig5 <- ggplot() +
                geom_point(data=ts_season_count[ANCHOR == 0 & !is.na(COUNT), ], aes(x=DAY_SINC
                geom_point(data=btfl_week_missing, aes(x=doy, y=count, colour = "missing"),
                            shape=4, size=2, stroke=2) +
                geom_line(data = btfl_ts, aes(x = doy, y = act, colour = "activity")) +
                geom_line(data = pheno, aes(x = trimDAYNO, y = btfl_ts[,unique(abund.true)]*NM
                xlim(1,365) + ylim(0, max(btfl_ts$act,
                                          pheno$NM*btfl_ts[,unique(abund.true)],
                                          btfl_week_missing$count,
                                          ts_season_count[!is.na(COUNT), COUNT] )) +
                scale_colour_manual("",
                      breaks = c("count", "activity", "missing", "GAM_fit"),
                      values = c(cnt_col, flc_col, missing_col, GAM_col)) +
                theme_light() +
                theme(legend.position = "inside", legend.position.inside = c(0.9, 0.8)) +
                labs(title = paste0("Simulated butterfly counts - Zonneveld Model (", y,")"),
                     subtitle = "- Fitting GAM model with rbms",
                     x = "Day of Year",
                     y = "Count")
btfl_fig5
```



Simple trend case

In the first scenario, we will apply the method to a simple case where we have one univoltine species that is monitored over 15 years across 100 sites where the populations follow the same trend with a known growth rate.