Fit rTPC models to `song_prop` after filtering

Michael Gilchrist

date: 2022-10-20

Goal

- Evaluate trends in song_count under a (near) constant temperature.

Set up

Load libraries

```
## load libraries
library(stats)
require(MASS) # provides negative binomial fitting:  glm.nb
```

```
## Loading required package: MASS
```

```
library(RSQLite)  # Don't think we need this.
library(rTPC)  ##
library(nls.multstart)
library(broom)
library(tidyverse)
```

```
## -- Attaching packages
## ---------------------------------------------------------------------------------------------------
## tidyverse 1.3.2 --

## v ggplot2 3.4.0          v purrr   0.3.5
## v tibble  3.1.8          v dplyr   1.0.99.9000
## v tidyr   1.2.1          v stringr 1.4.1
## v readr   2.1.3          v forcats 0.5.2
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
```

```
library(ggplot2)
library(ggpubr)
library(grid) ## provides textGrob
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(viridisLite)

                                        #options(ggplot2.continuous.colour="viridis",
                                        #        ggplot2.discrete.colour="viridis",
                                        #        ggplot2.scale_fill_discrete = scale_fill_viridis_d,
                                        #        ggplot2.scale_fill_continuous = scale_fill_viridis_c)

library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(lme4)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
library(nlme)
```

```
##
## Attaching package: 'nlme'
##
## The following object is masked from 'package:lme4':
##
##     lmList
##
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
library(gnm)
library(rsample) ## provides bootstraps()

library(RVAideMemoire) # provides overdisp.glmer()
```

```
## *** Package RVAideMemoire v 0.9-81-2 ***
##
## Attaching package: 'RVAideMemoire'
##
## The following object is masked from 'package:gnm':
##
##     se
##
## The following object is masked from 'package:lme4':
##
##     dummy
##
## The following object is masked from 'package:broom':
##
##     bootstrap
```

```
library(humidity) ## provides VPD
library(weathermetrics)
library(latex2exp)
```

Local Functions

- Copied from brms-first.fitting.Rmd

```
## Taken from: https://stackoverflow.com/a/51330864/5322644
## Use to get model equations for models in rTPC

help_text <- function(...) {
    file <- help(...)
    path <- dirname(file)
    dirpath <- dirname(path)
    pkgname <- basename(dirpath)
    RdDB <- file.path(path, pkgname)
    rd <- tools:::fetchRdDB(RdDB, basename(file))
    capture.output(tools::Rd2txt(rd, out="", options=list(underline_titles=FALSE)))
}

get_model_eq <- function(model) {
    txt <- help_text(model)
    eqn_line <- grep("^ +rate = .*$", txt, value = TRUE)
    print(paste(model, eqn_line))
    eqn <- gsub("(^ +rate = | *$)", "", eqn_line) %>%
        gsub("([^0-9])\\.(\\w+)", "\\1 * \\2", .) %>%
        gsub("\\.([^0-9])", " * \\1", .) %>%
        gsub("\\_", "", .)
    df <- tibble(model = model, eq = eqn)
```

```
    return(df)
}


plot_brms_fit <- function(brmsfit) {
    plist = list()

    plist[[1]] <- plot(brmsfit, title = paste("brm() summary: ", prior_index), ask = FALSE)
    return(plist)
}

plot_stan_fit <- function(stanfit) {
    plist = list();
    i <- 1

    plist[[i]] <-  plot(stanfit, title = paste("stan() summary: ", prior_index))

    ## Plotting routines from: www.weirdfishes.blog/blog/fitting-bayesian-models-with...
    ##
    mack_diagnostics <- rstan::get_sampler_params(stanfit) %>%
        set_names(1:4) %>%
        map_df(as_tibble,.id = 'chain') %>%
        group_by(chain) %>%
        mutate(iteration = 1:length(chain)) %>%
        mutate(warmup = iteration <= warmup) %>%
        mutate()
    i <- i+ 1

    plist[[i]] <- mack_diagnostics %>%
        group_by(warmup, chain) %>%
        summarise(percent_divergent = mean(divergent__ >0)) %>%
        ggplot() +
        geom_col(aes(chain, percent_divergent, fill = warmup), position = 'dodge', color = 'black') +
        scale_y_continuous(labels = scales::percent, name = "% Divergent Runs")  +
        scale_fill_npg()

    i<- i+1
    plist[[i]] <- mack_diagnostics %>%
        ggplot(aes(iteration, treedepth__, color = chain)) +
        geom_line() +
        geom_hline(aes(yintercept = max_treedepth), color = 'red') +
        scale_color_locuszoom()

    return(plist)

}
```

- Copied from: (https://rmazing.wordpress.com/2012/07/19/a-weighting-function-for-nls-nlslm/)
- Key usage: `wfct(1/fitted^2)` which uses the fitted value in nls

```
wfct <- function(expr)
{
    expr <- deparse(substitute(expr))
```

```
    ## create new environment
    newEnv <- new.env()

    ## get call
    mc <- sys.calls()[[1]]
    mcL <- as.list(mc)

    ## get data and write to newEnv
    DATA <- mcL[["data"]]
    DATA <- eval(DATA)
    DATA <- as.list(DATA)
    NAMES <- names(DATA)
    for (i in 1:length(DATA)) assign(NAMES[i], DATA[[i]], envir = newEnv)

    ## get parameter, response and predictor names
    formula <- as.formula(mcL[[2]])
    VARS <- all.vars(formula)
    RESP <- VARS[1]
    RHS <- VARS[-1]
    PRED <- match(RHS, names(DATA))
    PRED <- names(DATA)[na.omit(PRED)]

    ## calculate variances for response values if "error" is in expression
    ## and write to newEnv
    if (length(grep("error", expr)) > 0) {
        y <- DATA[[RESP]]
        x <- DATA[[PRED]]
        ## test for replication
        if (!any(duplicated(x))) stop("No replicates available to calculate error from!")
        ## calculate error
        error <- tapply(y, x, function(e) var(e, na.rm = TRUE))
        error <- as.numeric(sqrt(error))
        ## convert to original repititions
        error <- rep(error, as.numeric(table(x)))
        assign("error", error, envir = newEnv)
    }

    ## calculate fitted or residual values if "fitted"/"resid" is in expression
    ## and write to newEnv
    if (length(grep("fitted", expr)) > 0 || length(grep("resid", expr)) > 0) {
        mc2 <- mc
        mc2$weights <- NULL
        MODEL <- eval(mc2)
        fitted <- fitted(MODEL)
        resid <- residuals(MODEL)
        assign("fitted", fitted, newEnv)
        assign("resid", resid, newEnv)
    }

    ## return evaluation in newEnv: vector of weights
    OUT <- eval(parse(text = expr), envir = newEnv)
    return(OUT)
}
```

```
## Plotting settings
```

```
## From: https://data-se.netlify.app/2018/12/12/changing-the-default-color-scheme-in-ggplot2/

theme_set(theme_minimal(base_size = 9))
theme_update(
    plot.title = element_text(size = rel(1.1)),
    plot.subtitle = element_text(size = rel(1)))

if(!exists("old_opts")) old_opts <- options()  # save old options

options(ggplot2.continuous.colour="viridis")
options(ggplot2.continuous.fill = "viridis")
options(ggplot2.discrete.colour="viridis")
options(ggplot2.discrete.fill = "viridis")
```

Create Model Tibble

```
model_def_tbl <- lapply(get_model_names(), get_model_eq) %>% bind_rows(, .id = NULL) %>% tibble()
```

```
## [1] "beta_2012      rate = (a.((temp - b + ((c.(d-1))/(d + e - 2)))/c)^(d-1).(1 - ((temp - b + ((c.(
## [1] "boatman_2017            rate = rmax.(sin(pi.((temp - tmin)/(tmax - tmin))^a))^b      "
## [1] "briere2_1999               rate = a.temp.(temp - tmin).(tmax - temp)^(1/b)          "
## [1] "delong_2017      rate = c.exp(-(eb-(ef.(1-((temp + 273.15)/tm))+ehc.((temp + 273.15)-tm-((temp +
## [1] "flinn_1991                   rate = 1 / (1 + a + b.temp + c.temp^2)                "
## [1] "gaussian_1987                rate = rmax.exp(-0.5.(abs(temp - topt)/a)^2)           "
## [1] "hinshelwood_1947       rate = a.exp(-e/k.(temp + 273.15)) - b.exp(-eh/k.(temp + 273.15)) "
## [1] "joehnk_2008      rate = rmax.(1 + a.((b^(temp - topt) - 1) - (log(b)/log(c)).(c^(temp - topt) -
## [1] "johnsonlewin_1946      rate = (r0.exp(-e/(k.(temp + 273.15)))) / ((1 + exp((-1/(k.(temp + 273.15
## [1] "kamykowski_1985       rate =  a.(1 - exp(-b.(temp - tmin))).(1 - exp(-c.(tmax - temp))) "
## [1] "lactin2_1995       rate =  exp(a.temp) - exp(a.tmax - ((tmax - temp) / delta_t)) + b "
## [1] "lrf_1991      rate = rmax * ((temp - tmax)(temp - tmin))^2 / ((topt - tmin)((topt - tmin)(temp +
## [1] "modifiedgaussian_2006                 rate = rmax.exp(-0.5.(abs(temp - topt)/a)^b)            "
## [1] "oneill_1972      rate = rmax.(ctmax - temp / ctmax - topt)^2.exp(x.(temp-topt/ctmax-topt))"
## [1] "pawar_2018      rate = r_tref.exp(e/k.(1/tref - 1/(temp + 273.15))) / (1 + (e / (eh-e)) * exp(el
## [1] "quadratic_2008                          rate = a + b.temp + c.temp^2                    "
## [1] "ratkowsky_1983           rate = ((a.(temp - tmin)).(1 - exp(b.(temp - tmax))))^2      "
## [1] "rezende_2019                    rate = (a.10^(log10(q10)/(10/temp)))                 "
## [2] "rezende_2019           rate = (a.10^(log10(q10)/(10/temp))).(1-c.(b - temp)^2))      "
## [1] "sharpeschoolfull_1981      rate = r_tref.exp(e/k.(1/tref - 1/(temp + 273.15))) / (1 + exp(-el/k
## [1] "sharpeschoolhigh_1981      rate = r_tref.exp(e/k.(1/tref - 1/(temp + 273.15))) / (1 + exp(eh/k.
## [1] "sharpeschoollow_1981      rate = r_tref.exp(e/k.(1/tref - 1/(temp + 273.15))) / (1 + exp(-el/k.
## [1] "spain_1982              rate = est = r0 . exp(a.temp) . (1 - b.exp(c.temp))        "
## [1] "thomas_2012           rate = a . exp(b . temp) . (1 - ((temp - topt)/(c/2))^2)      "
## [1] "thomas_2017              rate = a . exp(b . temp) - (c + d.(exp(e.temp)))          "
## [1] "weibull_1995      rate = ((a.(((c-1)/c)^((1-c)/c)).((((temp-topt)/b)+(((c-1)/c)^(1/c)))^(c-1)).
```

```
                                    #print(model_def_tbl, n = 200, width = 200)

str_rm <- c("exp", "[0-9.]+", "log(2|10|)", "sin", "abs", "pi", "temp")
```

```
pattern <- paste0("\\b", paste0(str_rm, collapse = "\\b|\\b"), "\\b")
n_param  <- stringi::stri_extract_all_words(model_def_tbl$eq) %>%
    lapply(., unique) %>%
    lapply(., paste, collapse= " ") %>%
    str_replace_all(., pattern, "") %>%
    str_count(., boundary("word"))
model_tbl <- bind_cols(model_def_tbl, n_param= n_param) %>%
    arrange(n_param, model) %>% relocate(eq, .after = n_param)
## model_tbl
```

Load Data

```
load(file.path("input", "data.processing_2022-11-09.Rda"),
     verbose = TRUE)
```

```
## Loading objects:
##   song_data
##   song_data_40C
##   song_stats
##   song_stats_40C
##   bird_bill_data
```

Examine Data

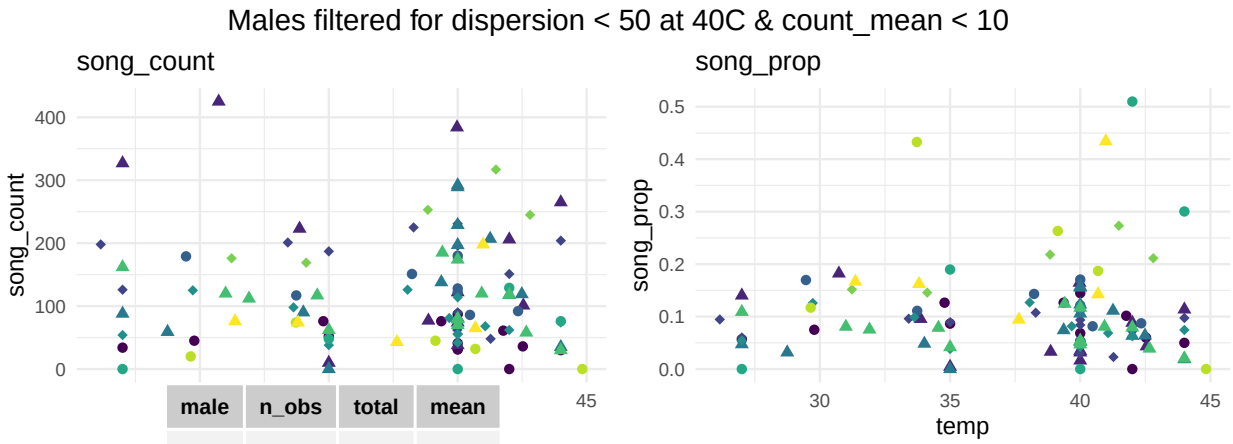
Create Working Dataset

```
males_filtered_disp <- song_stats_40C %>%
    filter(dispersion < 50) %>%
    pull(male)

males_filtered_mean <- song_stats %>%
    filter(mean > 10) %>%
    pull(male)

males_filtered <- intersect(males_filtered_mean, males_filtered_disp)

##males_selected <-

data_ind <- song_data %>%
    filter(male %in% males_filtered) %>%
    arrange(male) %>%
    ##    left_join(male_shape, by = "male") %>%
    mutate()

## copy data frame and assign `male =  "combined")
data_comb <- data_ind %>% mutate(male = "combined")

stats_ind <- song_stats %>%
    filter(male %in% males_filtered)
```

Plot song_count

```
g1 <- ggplot(data = data_ind) +
    aes(x = temp, y = song_count, color = male, shape = male) +
    ## Redefine shapes. Note need to set 'shape = male' above to prevent there from
    ## begin two legends: 1 for shape and 1 for color.
    scale_shape_manual(values = rep(c(16:18), length.out = length(males_filtered))) +
    geom_point() +
    scale_color_viridis_d() +
    labs(title = "song_count") +
    theme(legend.position="none")
```

```
g1 <- ggplot(data = data_ind) +
    aes(x = temp, y = song_count, color = male, shape = male) +
    ## Redefine shapes. Note need to set 'shape = male' above to prevent there from
    ## begin two legends: 1 for shape and 1 for color.
    scale_shape_manual(values = rep(c(16:18), length.out = length(males_filtered))) +
    geom_point() +
    scale_color_viridis_d() +
    labs(title = "song_count") +
    theme(legend.position="none")

g2 <- ggplot(data = data_ind) +
    aes(x = temp, y = song_prop, color = male, shape = male) +
    scale_shape_manual(values = rep(c(16:18), length.out = length(males_filtered))) +
    geom_point() +
    scale_color_viridis_d() +
    labs(title = "song_prop") +
    theme(legend.position="bottom")


legend <- get_legend(g2)

g2 <- g2 + theme(legend.position="none")

g3 <- tableGrob(format(data.frame(stats_ind %>% select(male, n_obs, total, mean) %>% unique() ),
                       digits = 1),
                theme = ttheme_default(base_size = 8))

grid.arrange(g1, g2, g3, legend, ncol = 2,
             top=textGrob("Males filtered for dispersion < 50 at 40C & count_mean < 10",
                          gp=gpar(fontsize = 11))
             )
```



	male	n_obs	total	mean
1	T236	13	2095	161
2	T243	13	1861	143
3	T246	5	253	51
4	T247	14	1487	106
5	T234	13	601	46
6	T244	13	993	76
7	T235	13	2333	179
8	T237	8	1055	132
9	T258	5	171	34
10	T257	5	1160	232

Analyze Data:

Models with 3 or 4 parameters

```
model_set <- model_tbl %>% filter(n_param < 5 & n_param > 2)
model_grob <- tableGrob(model_set)

grid.arrange(model_grob)
```

eq
1 / (1 + a + b * temp + c * temp^2)
rmax * exp(−0.5 * (abs(temp − topt)/a)^2)
a + b * temp + c * temp^2
a * temp * (temp − tmin) * (tmax − temp)^(1/b)
exp(a * temp) − exp(a * tmax − ((tmax − temp) / delta
rmax * ((temp − tmax)(temp − tmin))^2 / ((topt − tmin)((topt − tmin)(temp − topt) − (
rmax * exp(−0.5 * (abs(temp − topt)/a)^b)
rmax * (ctmax − temp / ctmax − topt)^2 * exp(x * (temp−topt
((a * (temp − tmin)) * (1 − exp(b * (temp − tmax)))
(a * 10^(log10(q10)/(10/temp))) * (1−c * (b − temp)
a * exp(b * temp) * (1 − ((temp − topt)/(c/2))^
((a * (((c−1)/c)^((1−c)/c)) * ((((temp−topt)/b)+(((c−1)/c)^(1/c)))^(c−1)) * (exp(−((((temp−t

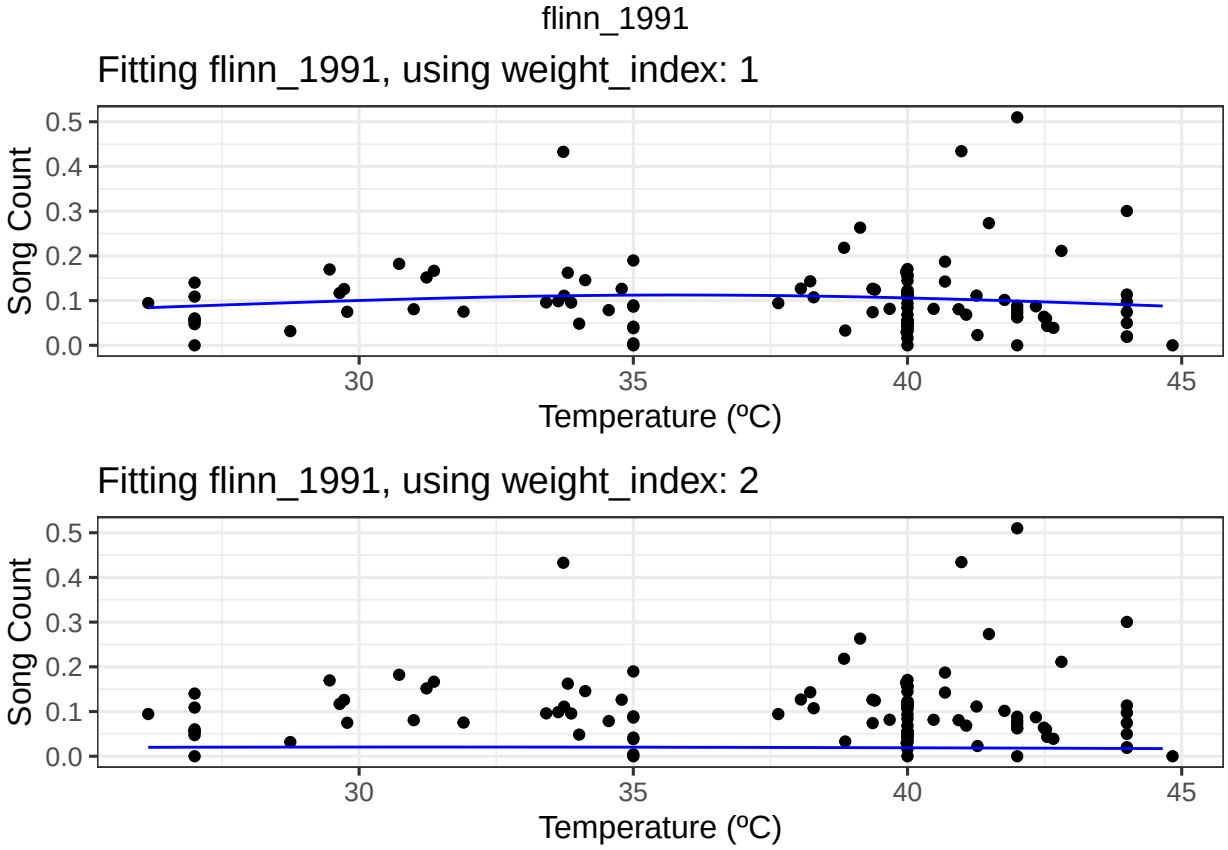
Flinn

```
data <- data_ind %>% rename(rate = song_prop) %>%
    select(temp, rate, weights, song_count_plus_1) %>%
    data.frame()

fit_list <- list()
graph_list <- list()
tgrob_list <- list()

for(model_str in model_set$model) {

    n_param <- formals(model_str) %>% length() - 1
    iter <- 500 ##rep(5, n_param)

    model_fits <- list()
    model_graphs <- list()
    model_tgrobs <- list()

    start_vals <- get_start_vals(data$temp, data$rate, model_name = model_str)
    lower <- get_lower_lims(data$temp, data$rate, model_name = model_str)
    upper <- get_upper_lims(data$temp, data$rate, model_name = model_str)

    my.formals <- names(formals(model_str)) %>% paste(., collapse = ", ") %>% sub("temp,", "temp = temp
    formula <- paste0("rate ~ ", model_str, "(", my.formals, ")")
    for(weights_index in 1:2) {

        ## WARNING: if you use 'weights' as a variable, it uses the column in data$weights even if weig

        local.weights <- switch(weights_index,
                                rep(1, nrow(data)),
                                data$weights
                                )
```

```
        fit <- nls_multstart(formula = formula,
                             data = data,
                             iter = iter,
                             start_lower = lower*1.01,
                             start_upper = upper*0.99,
                             lower = lower,
                             upper = upper,
                             supp_errors = 'Y',
                             modelweights =  local.weights,
                             convergence_count = 100, ## only used if iter = INT
                             control = c(maxiter = 1024, maxfev = 100000)
                             )


        summary(fit) %>% print()
        ##fit_list[[model_str]] <- fit

        ## calculate additional traits
        if(FALSE) calc_params(fit) %>% mutate_all(round, 2)

        ## Get predictions of our model using broom::augment(), which is similar to predict(). These ar

                                        # predict new data
        new_data <- data.frame(temp = seq(min(data$temp), max(data$temp), 0.5))
        preds <- augment(fit, newdata = new_data)

                                        # plot data and model fit
        g <- ggplot(data, aes(temp, rate)) +
            geom_point() +
            geom_line(aes(temp, .fitted), preds, col = 'blue') +
            theme_bw(base_size = 12) +
            labs(x = 'Temperature (ºC)',
                 y = 'Song Count',
                 title = paste0("Fitting ", model_str, ", using weight_index: ", weights_index))

        model_fits[[weights_index]] <- fit
        model_graphs[[weights_index]] <- g
##        model_tgrobs[[weights_index]] <- text_grob(label = summary(fit))

    }

    ##grid.arrange(grobs = c(model_graphs, model_tgrobs), top = model_str)
    grid.arrange(grobs = model_graphs, top = model_str)

    fit_list[[model_str]] <- model_fits;
    graph_list[[model_str]] <- model_graphs;
    tgrob_list[[model_str]] <- model_tgrobs;

}
```

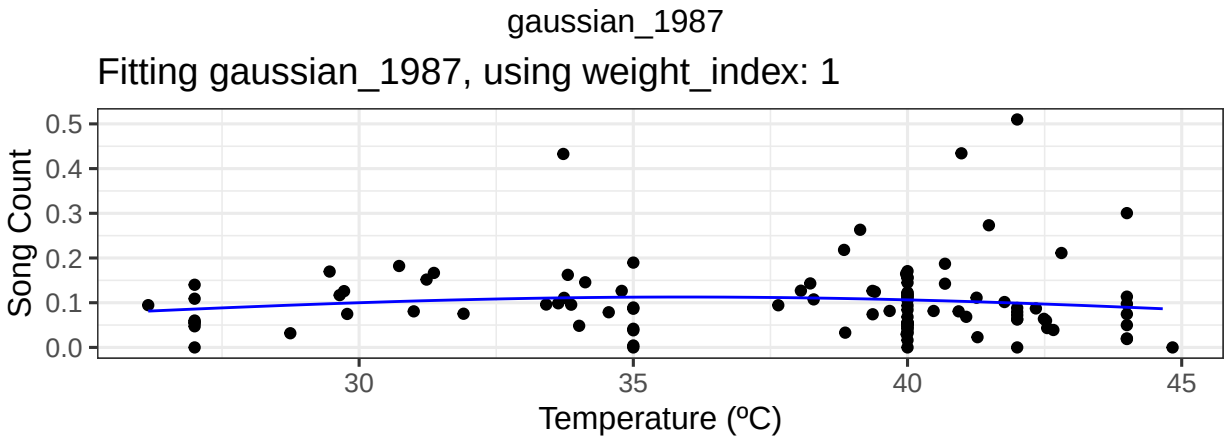
```
##
## Formula: rate ~ flinn_1991(temp = temp, a, b, c)
##
```

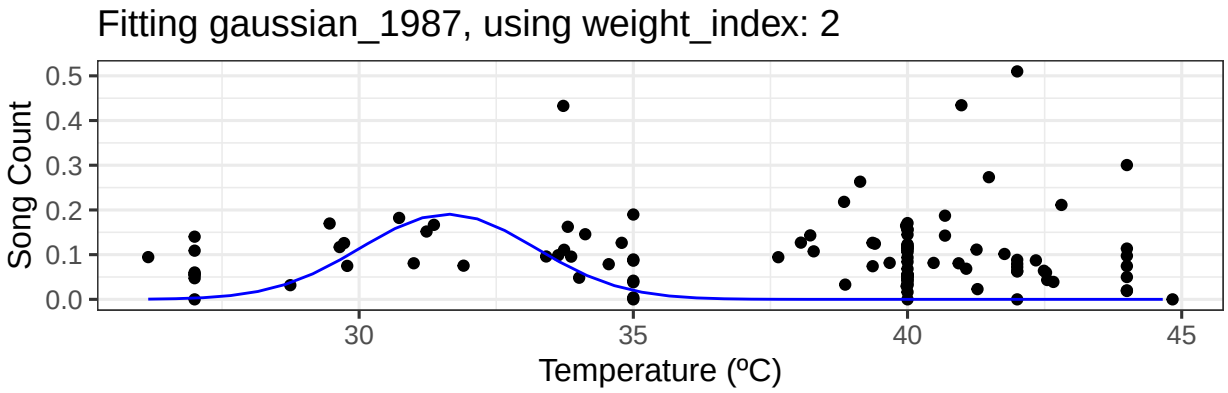
```
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## a 48.85927   49.71516   0.983    0.328
## b -2.28687    2.79500  -0.818    0.415
## c  0.03191    0.03884   0.822    0.413
##
## Residual standard error: 0.0849 on 104 degrees of freedom
##
## Number of iterations to convergence: 28
## Achieved convergence tolerance: 1.49e-08
##
##
## Formula: rate ~ flinn_1991(temp = temp, a, b, c)
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## a 100.00000  536.61314   0.186    0.853
## b  -3.36423   31.12324  -0.108    0.914
## c   0.05397    0.44053   0.123    0.903
##
## Residual standard error: 0.01245 on 104 degrees of freedom
##
## Number of iterations to convergence: 14
## Achieved convergence tolerance: 1.49e-08
```




```
##
```

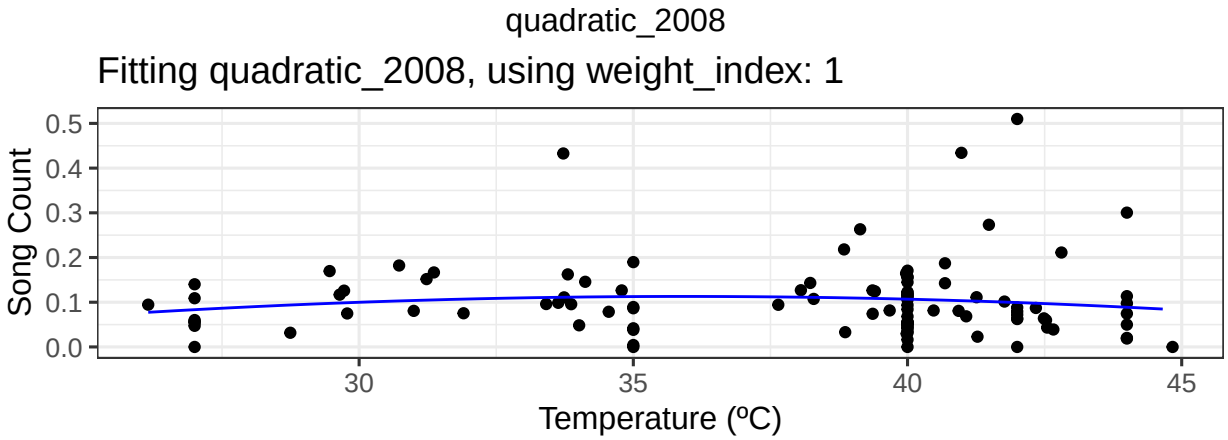
```
## Formula: rate ~ gaussian_1987(temp = temp, rmax, topt, a)
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## rmax   0.1127     0.0143   7.882 3.34e-12 ***
## topt  35.8962     2.5645  13.997  < 2e-16 ***
## a     12.0370     6.5466   1.839   0.0688 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08487 on 104 degrees of freedom
##
## Number of iterations to convergence: 14
## Achieved convergence tolerance: 1.49e-08
##
##
## Formula: rate ~ gaussian_1987(temp = temp, rmax, topt, a)
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## rmax  0.19050    0.06165    3.09  0.00257 **
## topt 31.61928    0.37900   83.43  < 2e-16 ***
## a     1.58786    0.27908    5.69 1.18e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01218 on 104 degrees of freedom
##
## Number of iterations to convergence: 82
## Achieved convergence tolerance: 1.49e-08
```

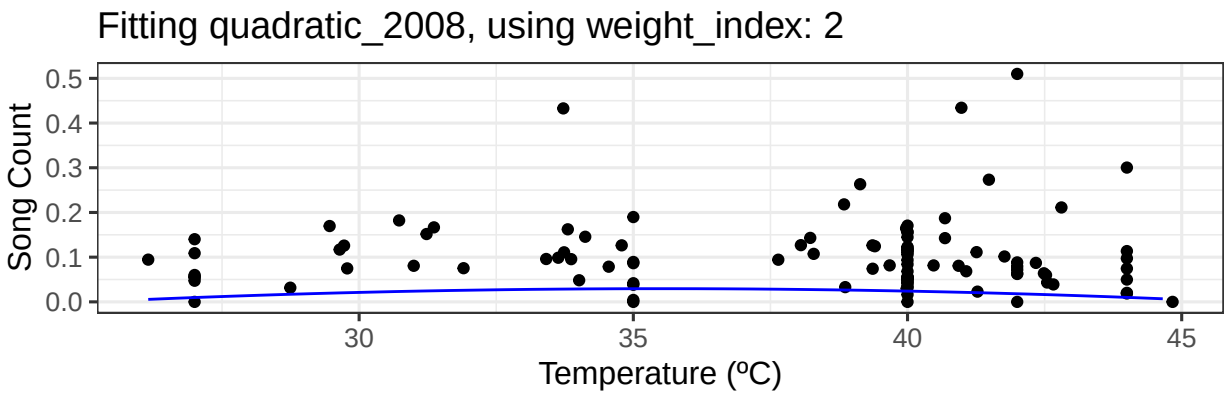





```
##
## Formula: rate ~ quadratic_2008(temp = temp, a, b, c)
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## a -0.366110   0.444217  -0.824    0.412
## b  0.026664   0.025367   1.051    0.296
## c -0.000371   0.000356  -1.042    0.300
##
## Residual standard error: 0.08483 on 104 degrees of freedom
##
## Number of iterations to convergence: 3
## Achieved convergence tolerance: 1.49e-08
##
##
## Formula: rate ~ quadratic_2008(temp = temp, a, b, c)
##
## Parameters:
##     Estimate Std. Error t value Pr(>|t|)
## a -0.3138741  0.1962339  -1.599   0.1127
## b  0.0193259  0.0112450   1.719   0.0887 .
## c -0.0002719  0.0001572  -1.730   0.0866 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01229 on 104 degrees of freedom
##
```

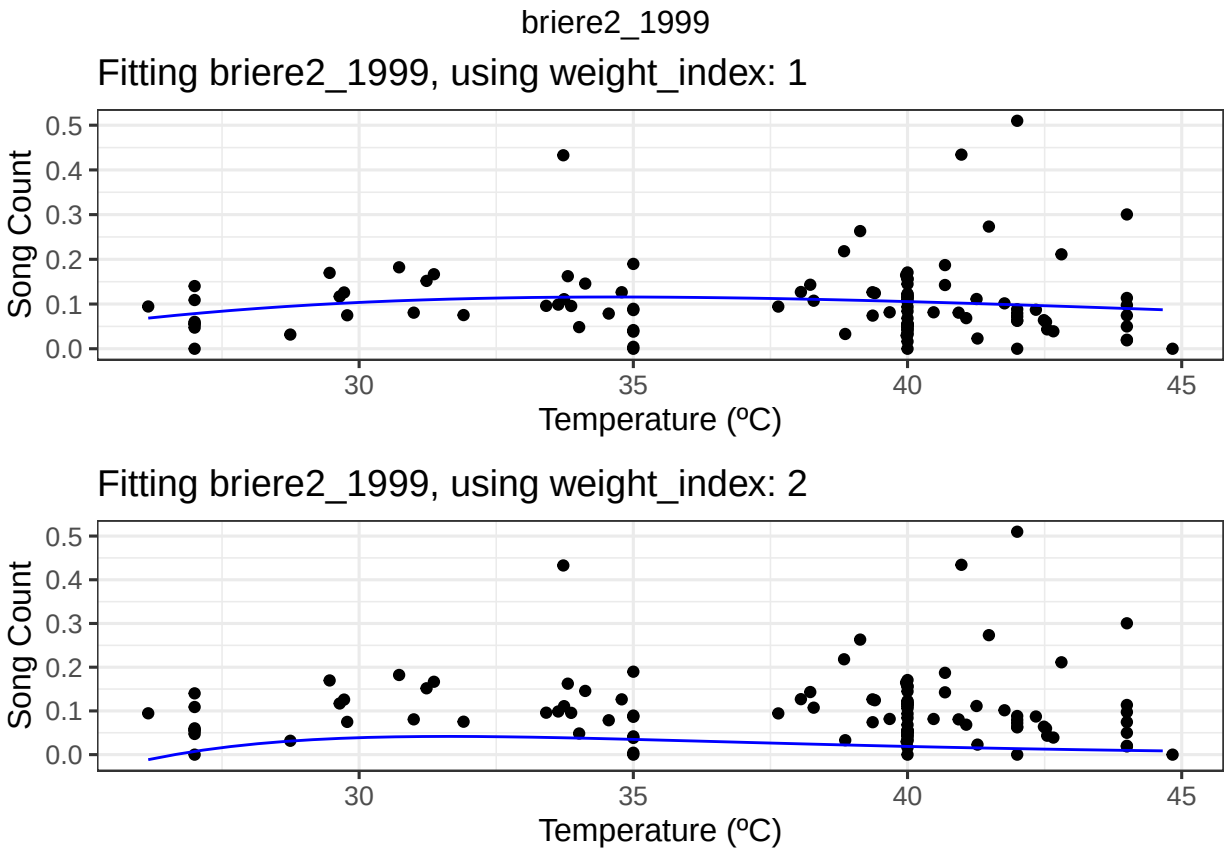
```
## Number of iterations to convergence: 3
## Achieved convergence tolerance: 1.49e-08
```





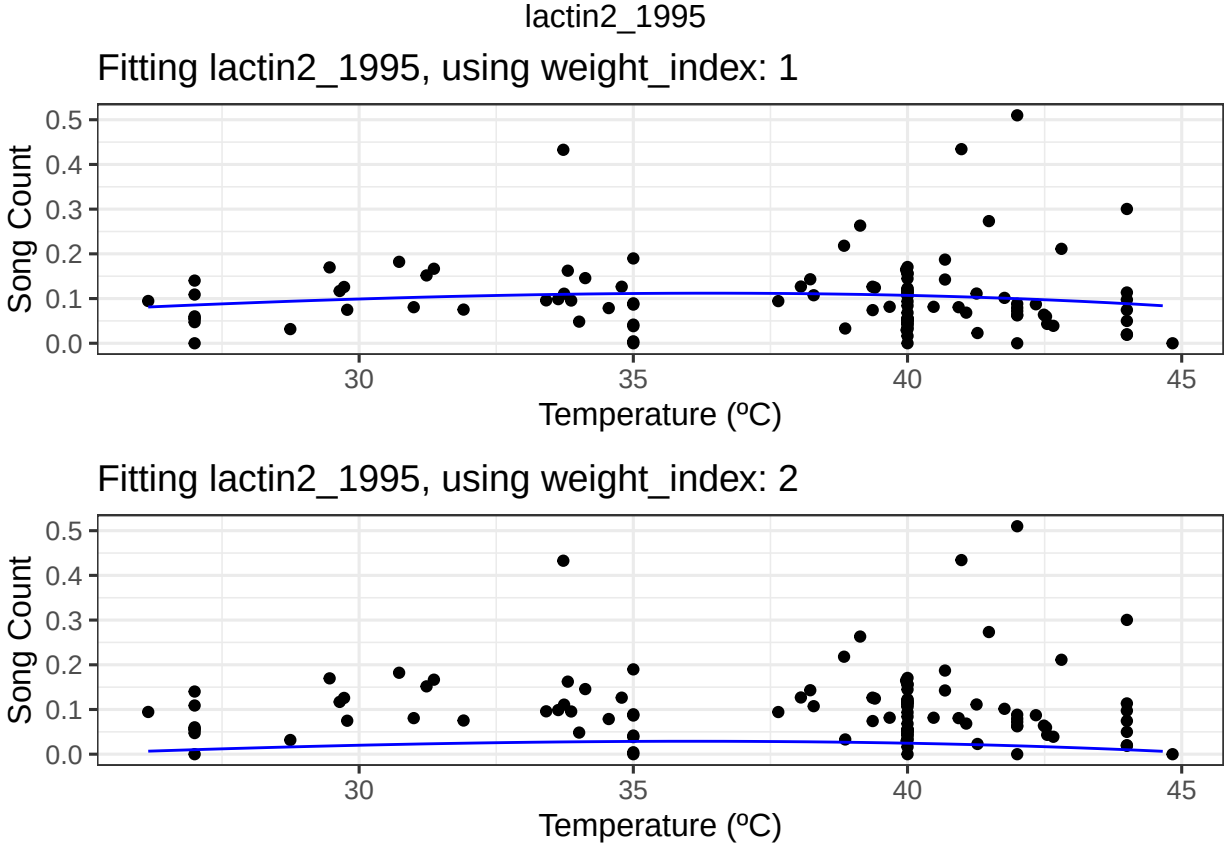

```
##
## Formula: rate ~ briere2_1999(temp = temp, tmin, tmax, a, b)
##
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
## tmin 2.222e+01  1.108e+01   2.005   0.0476 *
## tmax 2.080e+02  4.706e+03   0.044   0.9648
## a    9.673e-47  3.149e-43   0.000   0.9998
## b    5.275e-02  1.480e+00   0.036   0.9716
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08514 on 103 degrees of freedom
##
## Number of iterations till stop: 1024
## Achieved convergence tolerance: 1.49e-08
## Reason stopped: Number of iterations has reached 'maxiter' == 1024.
##
##
## Formula: rate ~ briere2_1999(temp = temp, tmin, tmax, a, b)
##
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
```

```
## tmin 2.664e+01  9.292e-01  28.665   <2e-16 ***
## tmax 1.535e+02  1.907e+03   0.081    0.936
## a    1.849e-63  4.968e-60   0.000    1.000
## b    3.526e-02  5.808e-01   0.061    0.952
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01232 on 103 degrees of freedom
##
## Number of iterations till stop: 1024
## Achieved convergence tolerance: 1.49e-08
## Reason stopped: Number of iterations has reached 'maxiter' == 1024.
```



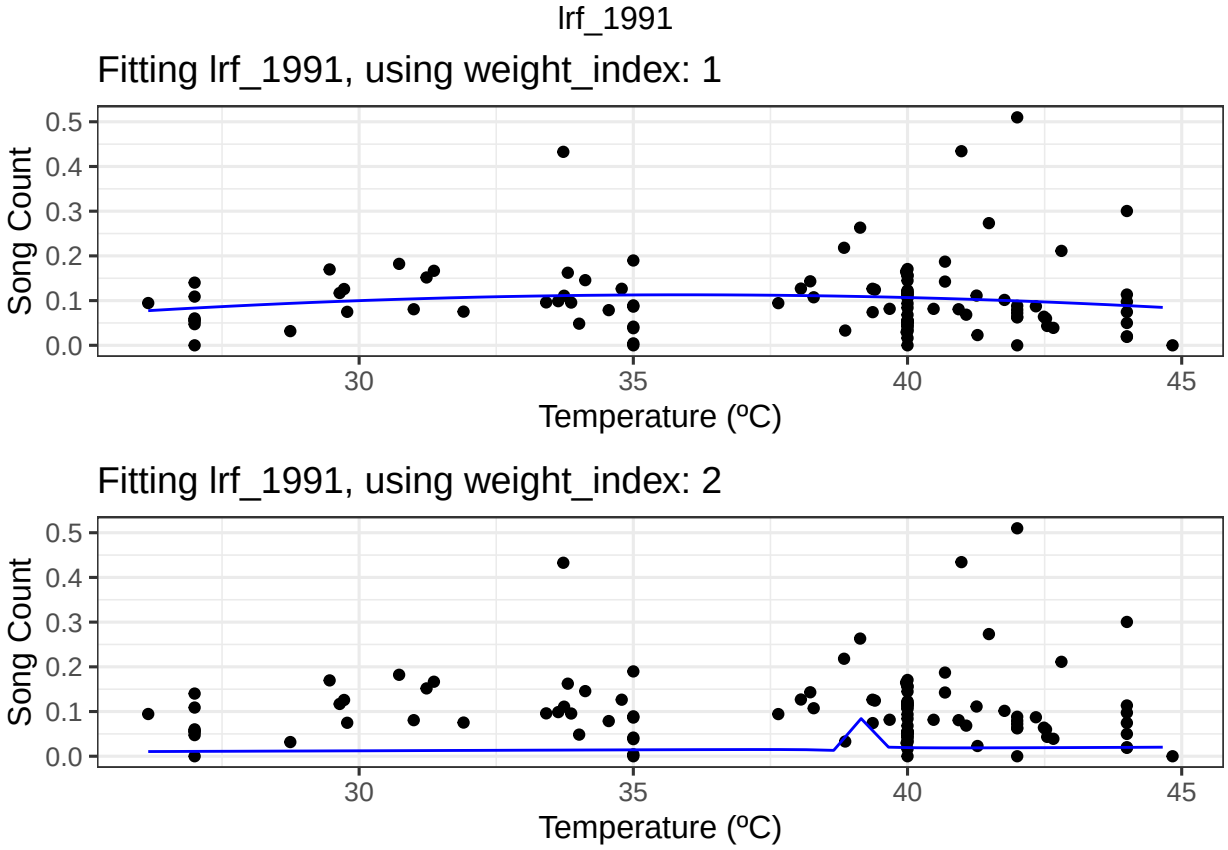

```
##
## Formula: rate ~ lactin2_1995(temp = temp, a, b, tmax, delta_t)
##
## Parameters:
##           Estimate Std. Error t value Pr(>|t|)
## a          0.01725   10.03811   0.002    0.999
## b         -0.93617  322.97064  -0.003    0.998
## tmax      73.85713 3635.63344   0.020    0.984
## delta_t   25.55493 6962.82039   0.004    0.997
##
## Residual standard error: 0.08528 on 103 degrees of freedom
##
## Number of iterations to convergence: 414
```

```
## Achieved convergence tolerance: 1.49e-08
##
##
## Formula: rate ~ lactin2_1995(temp = temp, a, b, tmax, delta_t)
##
## Parameters:
##           Estimate Std. Error t value Pr(>|t|)
## a          0.01454    1.51409   0.010    0.992
## b         -0.98006   51.67837  -0.019    0.985
## tmax      78.28407  597.77577   0.131    0.896
## delta_t   27.71693 1456.87184   0.019    0.985
##
## Residual standard error: 0.01235 on 103 degrees of freedom
##
## Number of iterations to convergence: 451
## Achieved convergence tolerance: 1.49e-08
```



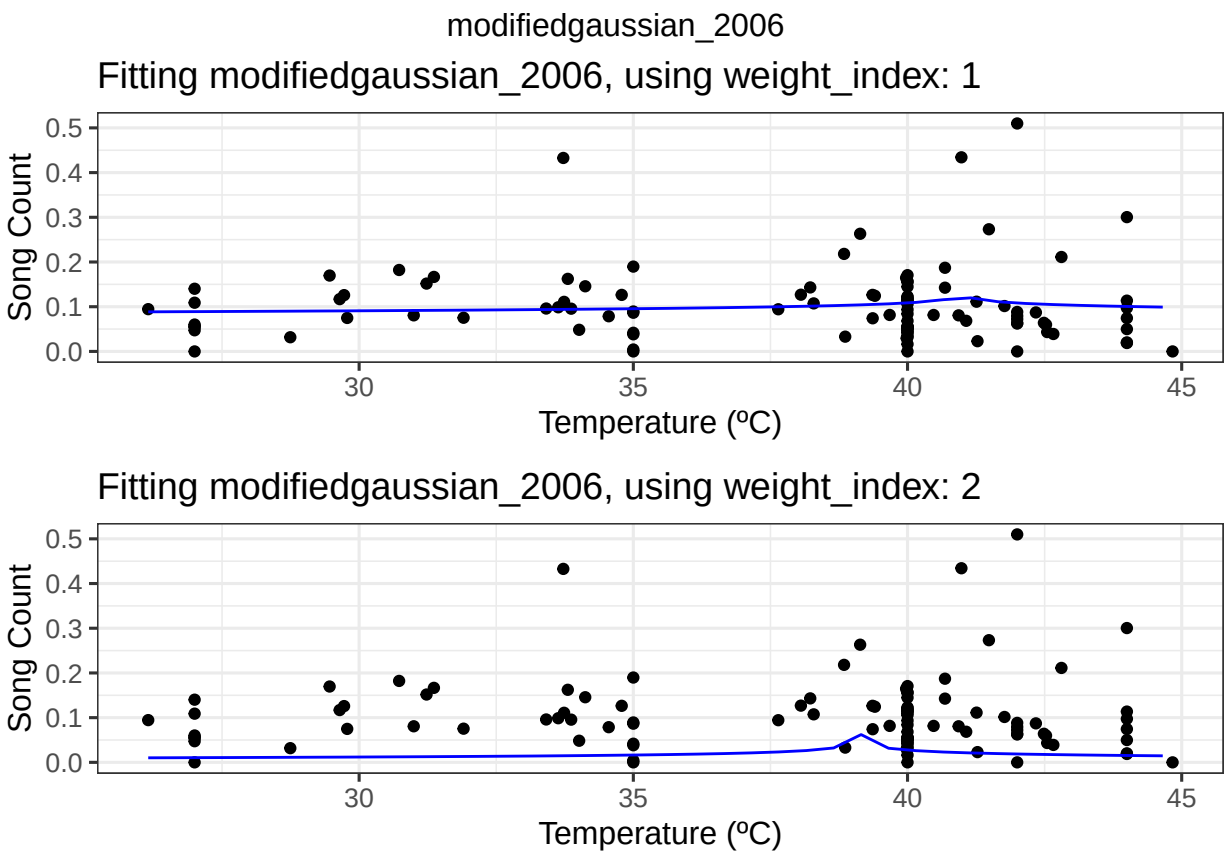

```
##
## Formula: rate ~ lrf_1991(temp = temp, rmax, topt, tmin, tmax)
##
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
## rmax 1.130e-01  1.357e-02   8.325 3.79e-13 ***
## topt 3.593e+01  4.813e+00   7.466 2.75e-11 ***
## tmin 1.849e+01  1.358e+05   0.000 0.999892
## tmax 5.338e+01  1.523e+01   3.505 0.000678 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08524 on 103 degrees of freedom
##
## Number of iterations to convergence: 19
## Achieved convergence tolerance: 1.49e-08
##
##
## Formula: rate ~ lrf_1991(temp = temp, rmax, topt, tmin, tmax)
##
## Parameters:
##        Estimate Std. Error t value Pr(>|t|)
## rmax   0.018741   0.005551   3.376  0.00104 **
## topt  40.857641   5.257753   7.771 6.08e-12 ***
## tmin -23.846027 108.950152  -0.219  0.82718
## tmax  39.032162   0.568949  68.604  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01205 on 103 degrees of freedom
##
## Number of iterations to convergence: 63
## Achieved convergence tolerance: 1.49e-08
```



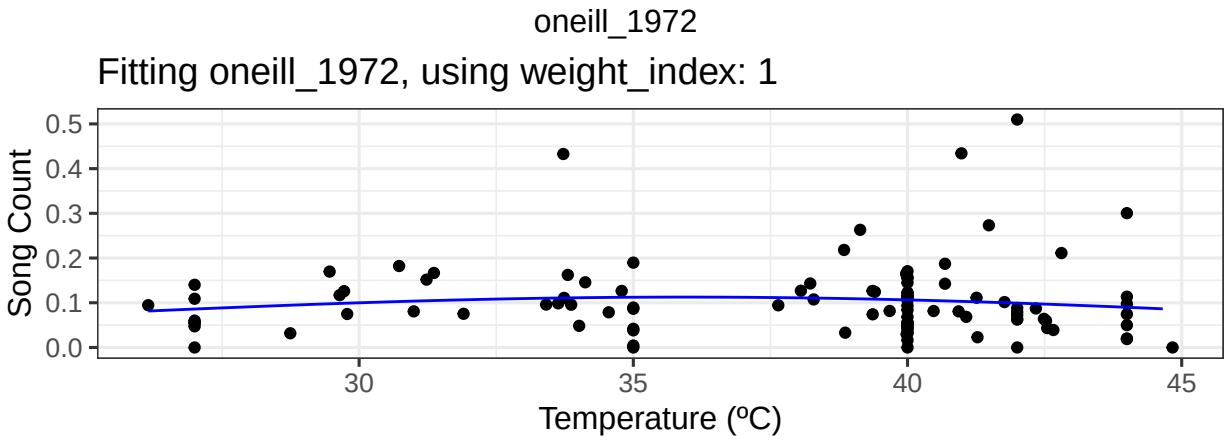

```
##
```

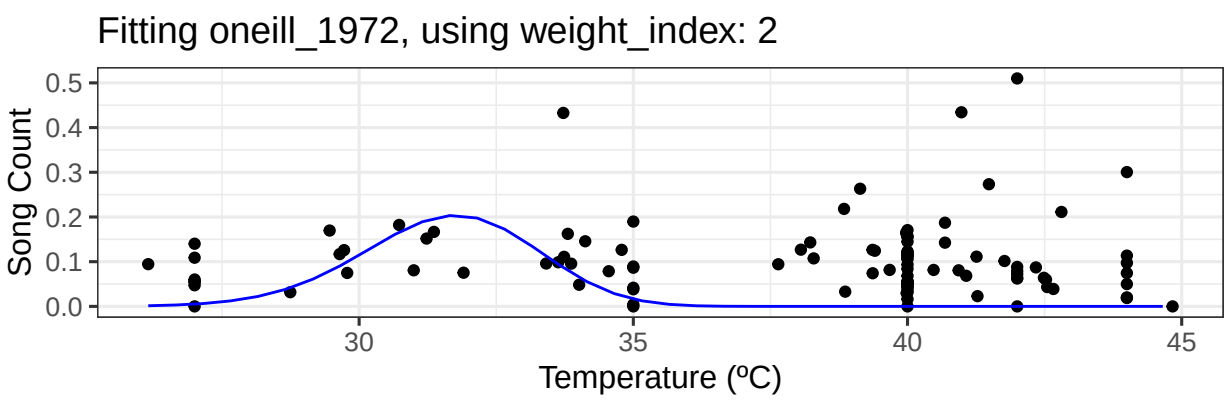
```
## Formula: rate ~ modifiedgaussian_2006(temp = temp, rmax, topt, a, b)
##
## Parameters:
##       Estimate Std. Error   t value Pr(>|t|)
## rmax 1.808e-01  1.043e+00 1.730e-01    0.863
## topt 4.098e+01  1.653e-04 2.479e+05   <2e-16 ***
## a    8.273e-01  7.759e+01 1.100e-02    0.992
## b    1.235e-01  1.276e+00 9.700e-02    0.923
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.084 on 103 degrees of freedom
##
## Number of iterations to convergence: 15
## Achieved convergence tolerance: 1.49e-08
##
##
## Formula: rate ~ modifiedgaussian_2006(temp = temp, rmax, topt, a, b)
##
## Parameters:
##       Estimate Std. Error   t value Pr(>|t|)
## rmax 1.486e-01  2.998e+00 5.000e-02    0.961
## topt 3.914e+01  1.991e-04 1.966e+05   <2e-16 ***
## a    6.702e-04  9.558e-02 7.000e-03    0.994
## b    1.701e-01  1.726e+00 9.900e-02    0.922
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01209 on 103 degrees of freedom
##
## Number of iterations to convergence: 23
## Achieved convergence tolerance: 1.49e-08
```




```
##
## Formula: rate ~ oneill_1972(temp = temp, rmax, ctmax, topt, q10)
##
## Parameters:
##        Estimate Std. Error t value Pr(>|t|)
## rmax  1.127e-01  1.456e-02   7.739 7.12e-12 ***
## ctmax 4.483e+02  5.284e+04   0.008    0.993
## topt  3.594e+01  5.469e+00   6.571 2.09e-09 ***
## q10   1.459e-01  3.392e+00   0.043    0.966
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08528 on 103 degrees of freedom
##
## Number of iterations to convergence: 7
## Achieved convergence tolerance: 1.49e-08
##
##
## Formula: rate ~ oneill_1972(temp = temp, rmax, ctmax, topt, q10)
##
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
## rmax   0.20372    0.06253   3.258  0.00152 **
## ctmax 44.83334   34.94105   1.283  0.20233
## topt  31.76690    0.63838  49.762  < 2e-16 ***
## q10    6.64805    2.11767   3.139  0.00221 **
## ---
```

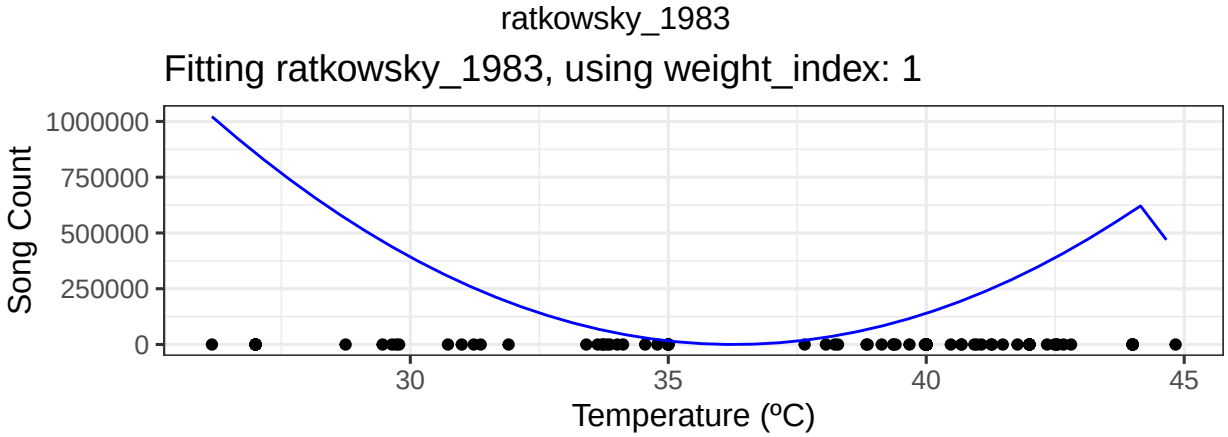
```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01219 on 103 degrees of freedom
##
## Number of iterations to convergence: 32
## Achieved convergence tolerance: 1.49e-08
```

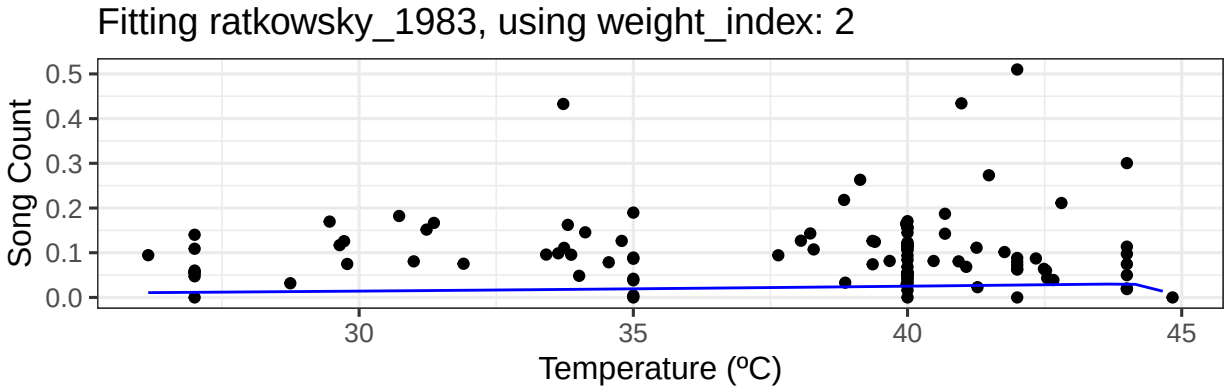





```
##
## Formula: rate ~ ratkowsky_1983(temp = temp, tmin, tmax, a, b)
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## tmin  36.2626     0.3777  96.021   <2e-16 ***
## tmax  44.8234     0.6568  68.243   <2e-16 ***
## a    100.0000     5.6022  17.850   <2e-16 ***
## b     10.0000   627.7206   0.016    0.987
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 356000 on 103 degrees of freedom
##
## Number of iterations to convergence: 189
## Achieved convergence tolerance: 1.49e-08
##
##
## Formula: rate ~ ratkowsky_1983(temp = temp, tmin, tmax, a, b)
```

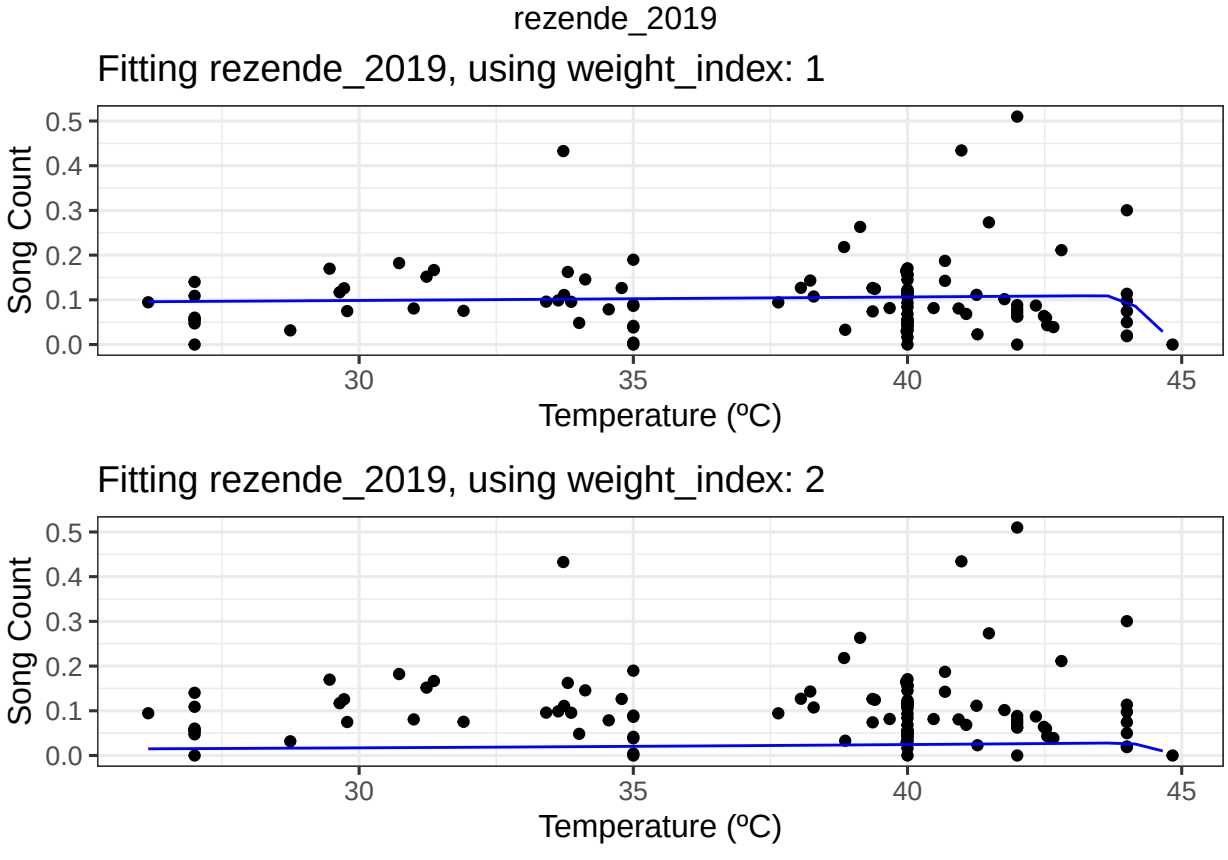
```
##
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
## tmin  0.000000  37.268738   0.000    1.000
## tmax 44.858082   0.477610  93.922   <2e-16 ***
## a     0.003966   0.003876   1.023    0.309
## b     5.380372  72.744879   0.074    0.941
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01234 on 103 degrees of freedom
##
## Number of iterations to convergence: 35
## Achieved convergence tolerance: 1.49e-08
```





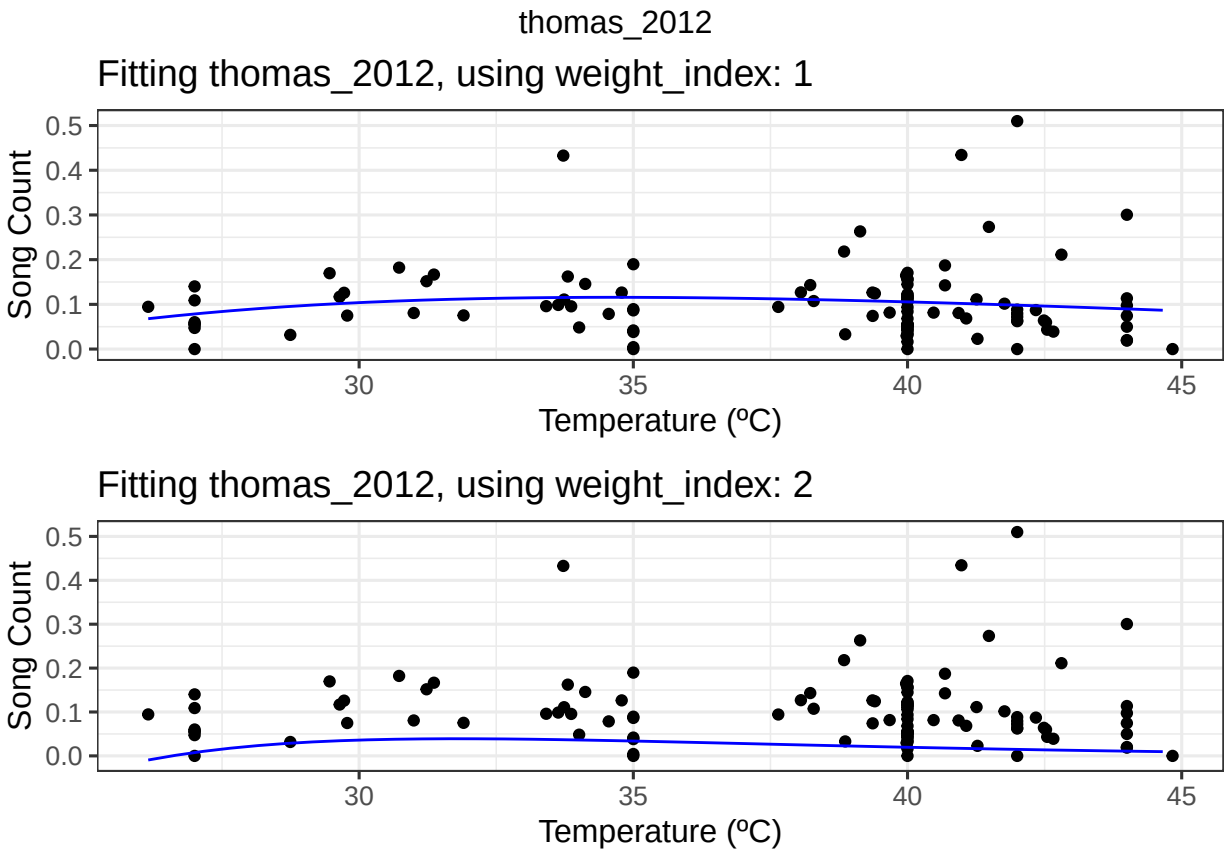

```
##
## Formula: rate ~ rezende_2019(temp = temp, q10, a, b, c)
##
## Parameters:
##     Estimate Std. Error t value Pr(>|t|)
## q10  1.07888    0.19333   5.581 1.95e-07 ***
## a    0.07854    0.05331   1.473    0.144
## b   43.55493    0.93321  46.672  < 2e-16 ***
## c    0.61188    1.12114   0.546    0.586
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.08498 on 103 degrees of freedom
##
## Number of iterations to convergence: 7
## Achieved convergence tolerance: 1.49e-08
##
##
## Formula: rate ~ rezende_2019(temp = temp, q10, a, b, c)
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## q10  1.429017   0.750999   1.903   0.0599 .
## a    0.005854   0.011816   0.495   0.6213
## b   43.846633   4.966307   8.829 2.95e-14 ***
## c    1.000000  10.075992   0.099   0.9211
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01233 on 103 degrees of freedom
##
## Number of iterations to convergence: 24
## Achieved convergence tolerance: 1.49e-08
```




```
##
## Formula: rate ~ thomas_2012(temp = temp, a, b, c, topt)
##
```

```
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
## a      0.88804   10.68965   0.083    0.934
## b     -0.05205    0.26218  -0.199    0.843
## c     44.60410  185.92274   0.240    0.811
## topt  44.83333   99.45336   0.451    0.653
##
## Residual standard error: 0.08514 on 103 degrees of freedom
##
## Number of iterations to convergence: 61
## Achieved convergence tolerance: 1.49e-08
##
##
## Formula: rate ~ thomas_2012(temp = temp, a, b, c, topt)
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## a     10.0000   161.9339   0.062    0.951
## b     -0.1551     0.3756  -0.413    0.681
## c     32.3169   153.8221   0.210    0.834
## topt  42.7433    77.5016   0.552    0.582
##
## Residual standard error: 0.01231 on 103 degrees of freedom
##
## Number of iterations to convergence: 11
## Achieved convergence tolerance: 1.49e-08
```



```
## Length  Class   Mode
##      0   NULL   NULL
## Length  Class   Mode
##      0   NULL   NULL
```

```
## Error in ‘geom_line()‘:
## ! Problem while computing aesthetics.
## i Error occurred in the 2nd layer.
## Caused by error in ‘FUN()‘:
## ! object ’temp’ not found
```