

# Process song\_count data

Michael Gilchrist

date: 2022-10-20

## Goal

- Developing code for processing `song_count`
- Idea is to index processing by date and then have other code load data in `output/` found in this directory.

## 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 3.3.6      v purrr  0.3.5
## v tibble  3.1.8      v dplyr  1.0.10
## 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.glmr()
```

```
## *** 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

```
kprint <- function(input, ...) {
  print(knitr::kable(input, ...))
  cat('\n\n<!-- -->\n\n')
}

if(interactive()) {
  file_name <- "testing.Rda"
} else {
  file_name <- knitr::current_input() %>% sub("\\.Rmd", ".Rda", .)
}
```

## 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")
```

## Load Data

```
## Read in ZEFI Data sets
## Treat 'repeatability' as round = 0
## Add round info

## Repeatability was done between round 1 and 2, female was present, but only one temp. so treating as

output_dir <- "output"

if(!dir.exists(output_dir)) dir.create(output_dir)
git_root <- system("git rev-parse --show-toplevel", intern = TRUE)

song_data_raw = list()

song_data_raw[[1]] <- read.csv(file.path(git_root, "data", "raw_data", "HSPi-Round-1-Heat-Trials.csv"))
  ## Note T237 and T230 are missing numbers in the song_count column
  ## so we are filtering these observations out until they are found
  filter(!is.na(song_count))

song_data_raw[[2]] <- read.csv(file.path(git_root, "data", "raw_data", "HSPi-Repeatability-Song-Count.csv"))
  mutate(round = 2) %>%
  ungroup()

song_data_raw[[3]] <- read.csv(file.path(git_root, "data", "raw_data", "HSPi-Round-2-Heat-Trials.csv"))
  mutate(round = 3) %>%
  ## Deal with missing temp_mean and humidity_mean values
  ## in round == 3
  ## 2022/10/19 - code no longer needed
  ## group_by(temp_target) %>%
  ##mutate(temp = if_else((round == 3 & is.na(temp_mean)),
  ##                      mean(temp_mean, na.rm = TRUE),
  ##                      temp_mean)) %>%
  ##mutate(humidity = if_else((round == 3 & is.na(humidity_mean)),
  ##                          mean(humidity_mean, na.rm = TRUE),
  ##                          humidity_mean)) %>%
  ungroup()
```

```

## Join data and discard empty columns
song_data_full <- full_join(song_data_raw[[1]], song_data_raw[[2]]) %>%
  full_join(song_data_raw[[3]]) %>%
  discard(~all(is.na(.) | . == "")) %>% ## get rid of columns of only NA
  mutate(n_obs_completed = !(is.na(song_count)) ) %>%
  mutate(song_count = ifelse(is.na(song_count), 0, song_count)) %>%
  mutate(song_count = song_count*1.0) %>% ## convert to a double so it's not treated as an integer
  mutate(chamber = as.factor(chamber), male = as.factor(male)) %>%
  ## create a male specific round and global trial index `trial`
  group_by(male, round) %>%
  mutate(trial_round = rank(date)) %>%
  ungroup() %>%
  mutate(song_count_plus_1 = (song_count + 1)) %>%
  mutate(log_song_count_plus_1 = log(song_count + 1)) %>%
  mutate(temp_target = as.numeric(temp_target)) %>%
  ## Create generic 'temp' column which is either
  ## temp_mean, if it exists, or temp_target, if it doesn't
  mutate(temp = if_else(is.na(temp_mean),
                        temp_target,
                        temp_mean)) %>%
  ## Add column with total song_count for a given round
  group_by(male, round) %>%
  mutate(count_total_round = sum(song_count),
         n_obs_round = length(song_count),
         count_mean_round = count_total_round/n_obs_round,
         count_sd_round = sd(song_count),
         count_cv_round = count_sd_round/count_mean_round) %>%
  ungroup(round) %>%
  mutate(count_total = sum(song_count),
         n_obs = length(song_count),
         count_mean = count_total/n_obs) %>%
  mutate(trial = rank(date)) %>%
  mutate(song_prop_round = song_count/count_total_round) %>%
  mutate(song_prop = song_count/count_total) %>%
  ## assuming poisson error
  ## From glm man page
  ## > Non-'NULL' 'weights' can be used to indicate that different
  ## > observations have different dispersions (with the values in
  ## > 'weights' being inversely proportional to the dispersions);
  ## add +1 to deal with single 0
  ## Interpret dispersion as ~ sd(), wt ~ 1/disp = 1/sd var()
  mutate(weight_count = sqrt(1/(song_count + 1))) %>%
  ## need to rescale wts for song_prop data
  mutate(weight_prop = weight_count * count_total_round) %>%
  ## Add vpd
  mutate(svp = SVP(t = temp_mean + 273.15, isK = TRUE), vpd = svp*(1-humidity_mean/100) ) %>%
  rename(bill_date = date_morph_data_collected) %>%
  group_by(round) %>%
  mutate(vpd_offset = vpd - mean(vpd)) %>%
  ungroup() %>%
  relocate(round, trial_round, song_count, song_prop, temp_mean, humidity_mean, .after = male) %>%
  mutate() ## Dummy function so we can comment out lines above it w/o any issues

```

```
## Joining, by = c("male", "chamber", "date", "song_count", "counter", "temp_target", "round")
## Joining, by = c("male", "chamber", "date", "song_count", "counter", "test_order", "temp_target", "round")
```

```
## Extract information on each male's bill
bird_bill_data <- song_data_full %>%
  select(male, bill_depth, bill_length, bill_width, bill_date)

song_stats <- song_data_full %>%
  select(male, round, n_obs, starts_with("count_")) %>%
  unique()
names(song_stats) <- names(song_stats) %>% sub("^count_", "", .)

## Remove columns on male bills
song_data <- song_data_full %>%
  select( -starts_with("bill_")) %>%
  select( -starts_with(c("count_")))
## Create small subset of data.

song_data_40C <- song_data %>%
  filter(temp_target == 40) %>%
  unique()

dim(song_data_40C)
```

```
## [1] 53 28
```

```
## Get stats for subset
## Original approach
song_stats_40C <- song_data_40C %>%
  group_by(male) %>%
  summarize(n_obs = length(song_count),
            mean = mean(song_count),
            var = var(song_count, na.rm = TRUE),
            cv = sqrt(var)/mean,
            dispersion = var/mean) %>%
  replace(is.na(.), 0)

d[is.na(d)] <- 0
```

```
## Error in d[is.na(d)] <- 0: object 'd' not found
```

```
print("We have 15 males, 5 of which we only have 1 observation at 40C")
```

```
## [1] "We have 15 males, 5 of which we only have 1 observation at 40C"
```

```
song_stats_40C
```

```
## # A tibble: 15 x 6
##   male n_obs mean var cv dispersion
##   <fct> <int> <dbl> <dbl> <dbl> <dbl>
```

```
## 1 T229      5 159.   8820.  0.592    55.6
## 2 T230      4  2.25   20.2  2        9
## 3 T231      5 106.   9770.  0.929    91.8
## 4 T234      5  53.4   688.  0.491    12.9
## 5 T235      5  78.4   890.  0.381    11.4
## 6 T236      5 184.   3233.  0.310    17.6
## 7 T237      4 145.    693.  0.181     4.77
## 8 T243      5 229    4218.  0.284    18.4
## 9 T244      5  78.6  1477.  0.489    18.8
## 10 T246     1  0       0  0         0
## 11 T247     5 118.   3223.  0.483    27.4
## 12 T257     1 253       0  0         0
## 13 T258     1  45       0  0         0
## 14 T259     1  0       0  0         0
## 15 T260     1  43       0  0         0
```

```
comment <- paste0("Data created using file ", file_name, " on ", date())

save_obj <- c("song_data", "song_data_40C", "song_stats", "song_stats_40C", "bird_bill_data")

## Add comments
## Note usage of eval and parse!!
for(x in save_obj) {
  my_exp <- paste0("comment(", x, ") <- \"", comment, "\"")
                                #eval(parse(text = my_exp))
  parse(text = my_exp)
}

save(list = save_obj, file = paste0(file.path("output", file_name)) )
```

## Plot song\_prop

```
g1 <- ggplot(data = song_data) +
  aes(x = temp, y = song_count, color = male) +
  geom_point() +
  labs(title = "Males unfiltered") +
  theme(legend.position="bottom")

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

data_filtered <- song_data %>% filter(male %in% males_filtered_disp)

g2 <- ggplot(data = data_filtered) +
  aes(x = temp, y = song_count, color = male) +
  geom_point() +
  labs(title = "Males filtered for dispersion < 50 at 40C") +
  theme(legend.position="bottom")

males_filtered_mean <- song_stats %>% filter(mean > 10) %>% pull(male)
males_filtered <- intersect(males_filtered_mean, males_filtered_disp)
data_filtered <- song_data %>% filter(male %in% males_filtered)
```

```
g3 <- ggplot(data = data_filtered) +
  aes(x = temp, y = song_count, color = male) +
  geom_point() +
  labs(title = "Males filtered for dispersion < 50 at 40C & count_mean < 10)") +
  theme(legend.position="bottom")
```

```
grid.arrange(g1, g2, g3, ncol = 2)
```

