Fifth Fitting of Thermal Models

Michael Gilchrist

date: 2022-10-20

Goal

- Simple, but reasonable fits using GLM qpoisson with either
  - `song_count` and `1|male`
  - `song_count` and `offset(total_count)`
  - `song_prop` with weights
- Use `temp_ref = 45` per Liz's request

Notes from 27 Oct 2022 meeting

- TODO Me
  - Document simplest fits of
    - quad qpoisson models to `song_count`
    - general quad with weights to `song_prop`
    - KEY QUESTION: Does `song_prop` work as well as `song_count`?
      - If so, use all `song_prop` data with `rTPC`
      - Fit lactin 2 and other models in fixed effects framework
      - Do model selection amongst
    - What's the diff between `song_count` vs `song_prop`
      - using `male` wiht `song_count` is more precise accommodation for `male`, but costs us 1 df/male
      - not using `male` with `song_prop` is a less precise accommodation with no cost in terms of df. It also allows us to use rTPC models (we think)

Resources

Poisson Regression and its extensions

- Provides some clear examples of R code
  - https://stats.oarc.ucla.edu/r/dae/negative-binomial-regression/
- Gives important details on variance terms
  - file://Rodriguez_2022_models.for.count.data.with.over.dispersion_Lecture.Notes.pdf

Recap

- Using a Gamma model for the error converges, but I don't think it's the appropriate model, it assumes var ~ x^2 See: ## First and Second Fitting
- Temp is hard to control in chambers, so we should work with `temp_mean` (mean value during trial), not `temp_target`.
- `humidity_mean` and `temp_mean` strongly co-vary so consider using aggregate index as explanatory variable
  - Vapor Pressure Deficit (provided by `humidity` package)
  - Heat Index (formulated for humans, and provided by `weathermetrics`)
- Round 3 data only for curve fitting
  - Only round where `temp_mean` and `humidity_mean` exist.
- `count_total_round` are consistent between Rounds 1 and 3, so could use that info to classify birds, estimate variance function in response, etc.
- One bird in `round = 3` collapsed during the trial so it was terminated. We've set the `song_count` from NA to 0 and should consider making at `temp_working` column using `temp_target` in this instance and `temp_mean` in all other instances. Would need to do something similar for relative humidity, i.e. use `mean(humidity_mean)` for the `temp_target` value.

Third Fitting

- Created `temp` and `humidity` variables from `temp_mean` and `humidity_mean` for males who didn't collapse, used `mean(temp_mean)` and `mean(humidity_mean)` values for one male that did.
- Copied `data_full` and set `male = "combined"` so we could look at all of the data at once.
- Learned about using `model` argument for glm models
- Can't fit ME models
  - I don't understand why the ME model with `song_count` where we use a RE for the intercept doesn't have a similar effect as using song_prop. This intercept value is essentially multiplying by a constant, so we could try and specify this value using `offset(log(count_total_round))`. So why doesn't this work?

Fourth Fitting

- Tried fitting lots of variations of `glm` and `glmer`
- Optimizer matters in terms of convergence.
- Adding fixed `male` term rather than treating it as RE via `(1|male)` was very important since the variation in overall male singing varies so greatly.
- poisson fits indicate data is greatly overdispersed.
- Got one particular form of GLMM (`glmer`) working well (i.e. when there's no covariance matrix). I think this is a valid approach, but am unsure and a bit concerned since the `|` fitting shows the RE for $x1$ and $x1^2$ being close to perfectly correlated.
- Overall results indicate taht this approach is problematic
  - Data is extremely overdispersed in these fits.
  - As a result, no signal beyond a `male` effect on overall activity is reliably detected.

Future Steps

Set up

Install libraries

```
## install packages user might not have by replacing FALSE with TRUE
if(FALSE) {
    BiocManager::install("mixOmics") ## needed by RVAideMemoire
    install.packages(c("RSQLite", "nls.multstart", "lme4", "RVAideMemoire"))
    ##  Install the thermal curve package from git_hub, not cran
    remotes::install_github("padpadpadpad/rTPC")

}

## 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(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(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:lme4':
##
##     dummy
##
## The following object is masked from 'package:broom':
##
##     bootstrap
```

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

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

git_root <- system("git rev-parse --show-toplevel", intern = TRUE)

data_raw = list()

data_raw[[1]] <- read.csv(file.path(git_root, "data", "raw_data", "HSPi-Round-1-Heat-Trials.csv")) %>% r
    ## 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))

data_raw[[2]] <- read.csv(file.path(git_root, "data", "raw_data", "HSPi-Repeatability-Song-Count.csv"))
    mutate(round = 2) %>%
    group_by(male) %>%
    mutate(test_order = rank(date)) %>%
    ungroup()

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
data_full <- full_join(data_raw[[1]], data_raw[[2]]) %>%
    full_join(data_raw[[3]]) %>%
    discard(~all(is.na(.) | . =="")) %>% ## get rid of columns of only NA
    mutate(trial_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 global variable trial_order based on individual rounds
    mutate(trial_index = as.integer(round*10+test_order)) %>%
    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)) %>%
    ungroup() %>%
    mutate(song_prop = song_count/count_total_round) %>%
    ## 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() or se() not var()
    mutate(count_wt = sqrt(1/(song_count + 1))) %>%
    ## need to rescale wts for song_prop data
    mutate(prop_wt = count_wt * count_total_round) %>%
    ## Add vpd
    mutate(svp = SVP(t = temp_mean + 273.15, isK = TRUE), vpd = svp*(1-humidity_mean/100) ) %>%
    group_by(round) %>%
    mutate(vpd_offset = vpd - mean(vpd)) %>%
    ungroup() %>%
    relocate(song_count, song_prop, vpd, 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", "test_order", "temp_target", "rou
## Joining, by = c("male", "chamber", "date", "song_count", "counter", "test_order", "temp_target", "rou
```

Examine Data

Create Working Dataset

```
data_ind <- data_full %>%
    filter(round %in% c(2,3)) %>%
    filter(count_total_round >= 1) %>%
    mutate()

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

data <- bind_rows(data_ind, data_comb)
```

Examine How Var(Mean) using `round = 2`

```
song_count_round_2  <- data_ind %>%
    filter(round == 2) %>%
    select(c(male, song_count)) %>%
```

```
    group_by(male) %>%
    mutate(mean = mean(song_count), sd = sd(song_count), var = var(song_count)) %>%
    ungroup() %>%
    unique()

## Bootstrap data to get SE of var and mean
bs_data <- song_count_round_2 %>%
select(male, song_count) %>%
    bootstraps(times=100, strata = male)

bs_moments <- map_dfr(
  bs_data$splits,
  function(x) {
    dat <- as.data.frame(x) %>% group_by(male) %>%
    summarize(mean = mean(song_count), var = var(song_count), sd = sd(song_count))
  }, .id = "sample")

bs_ci <- bs_moments %>%
    group_by(male) %>%
    summarize(mean_low = quantile(mean, 0.025),
              mean_high = quantile(mean, 0.975),
              sd_low = quantile(sd, 0.025),
              sd_high = quantile(sd, 0.975),
              var_low = quantile(var, 0.025),
              var_high = quantile(var, 0.975)
              )

##Clearly we have very little confidence in our moment estimators

song_count_stats <- full_join(song_count_round_2 %>% select(-song_count) %>% unique(), bs_ci, by = "mal


g1 <- ggplot(song_count_round_2) +
    aes(x=song_count,
        color = male,
        fill = male) +
    geom_histogram() +
    labs(title = "")

g2 <- ggplot(song_count_stats) +
    aes(x=mean, y = sd) +
    geom_point(aes(color = male)) +
    geom_smooth(method='lm', formula= y~x) +
    geom_errorbar(aes(ymin = sd_low, ymax = sd_high)) +
    geom_errorbarh(aes(xmin = mean_low, xmax = mean_high)) +
    labs(title = "Mean vs. sd for Round 2: temp_target = 40")

g3 <- ggplot(song_count_stats) +
    aes(x=mean, y = var) +
    geom_point(aes(color = male)) +
    geom_smooth(method='lm', formula= y~x) +
##    geom_errorbar(aes(ymin = var_low, ymax = var_high)) +
##    geom_errorbarh(aes(xmin = mean_low, xmax = mean_high)) +
```

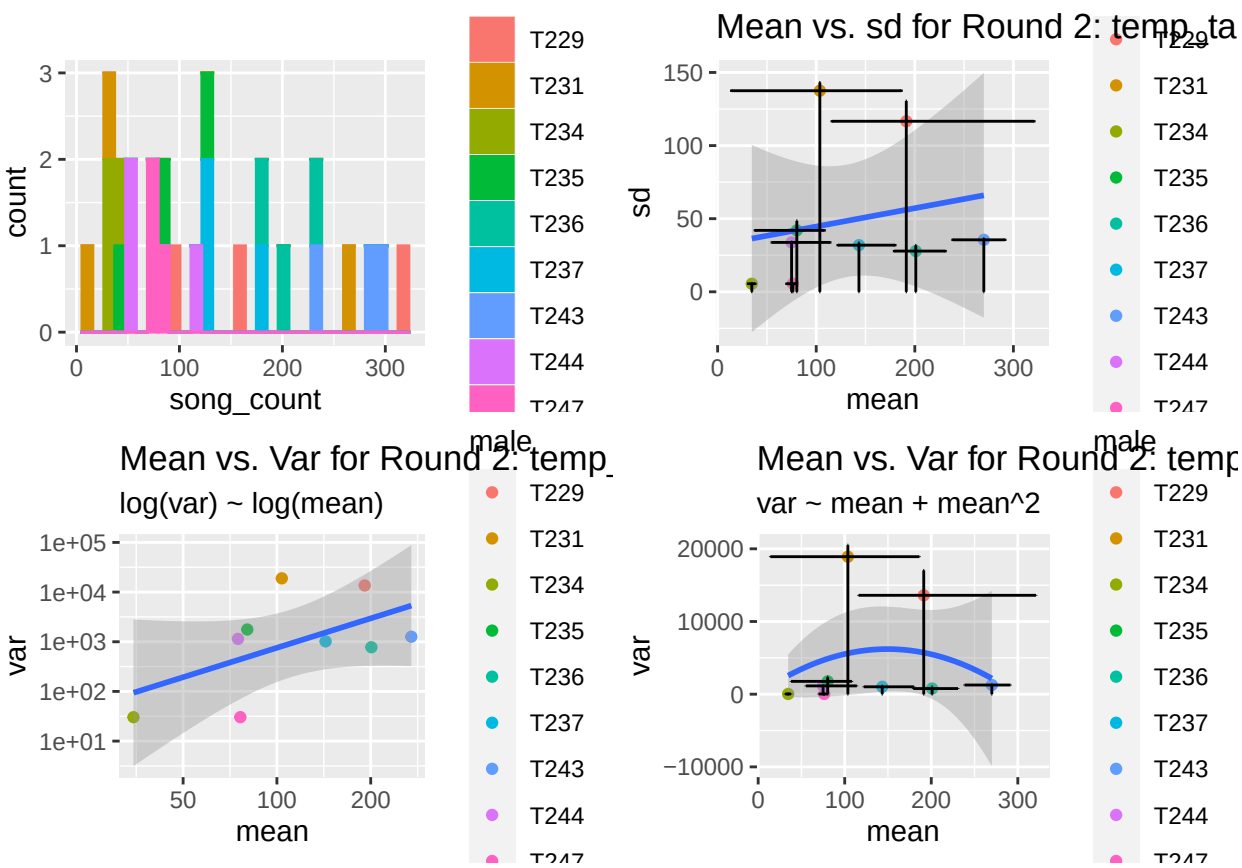
```
    labs(title = "Mean vs. Var for Round 2: temp_target = 40",
         subtitle = "log(var) ~ log(mean)") +
    scale_x_log10() +
    scale_y_log10()

#ggcoefstats(

g4 <- ggplot(song_count_stats) +
    aes(x=mean, y = var) +
    geom_point(aes(color = male)) +
    geom_smooth(method='glm', formula= y~-1 + x+I(x^2)) +
    geom_errorbar(aes(ymin = var_low, ymax = var_high)) +
    geom_errorbarh(aes(xmin = mean_low, xmax = mean_high)) +
    labs(title = "Mean vs. Var for Round 2: temp_target = 40",
         subtitle = "var ~ mean + mean^2")
#    scale_x_log10() +
#    scale_y_log10()

grid.arrange(g1, g2, g3, g4, ncol=2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Compare Rounds 2 and 3

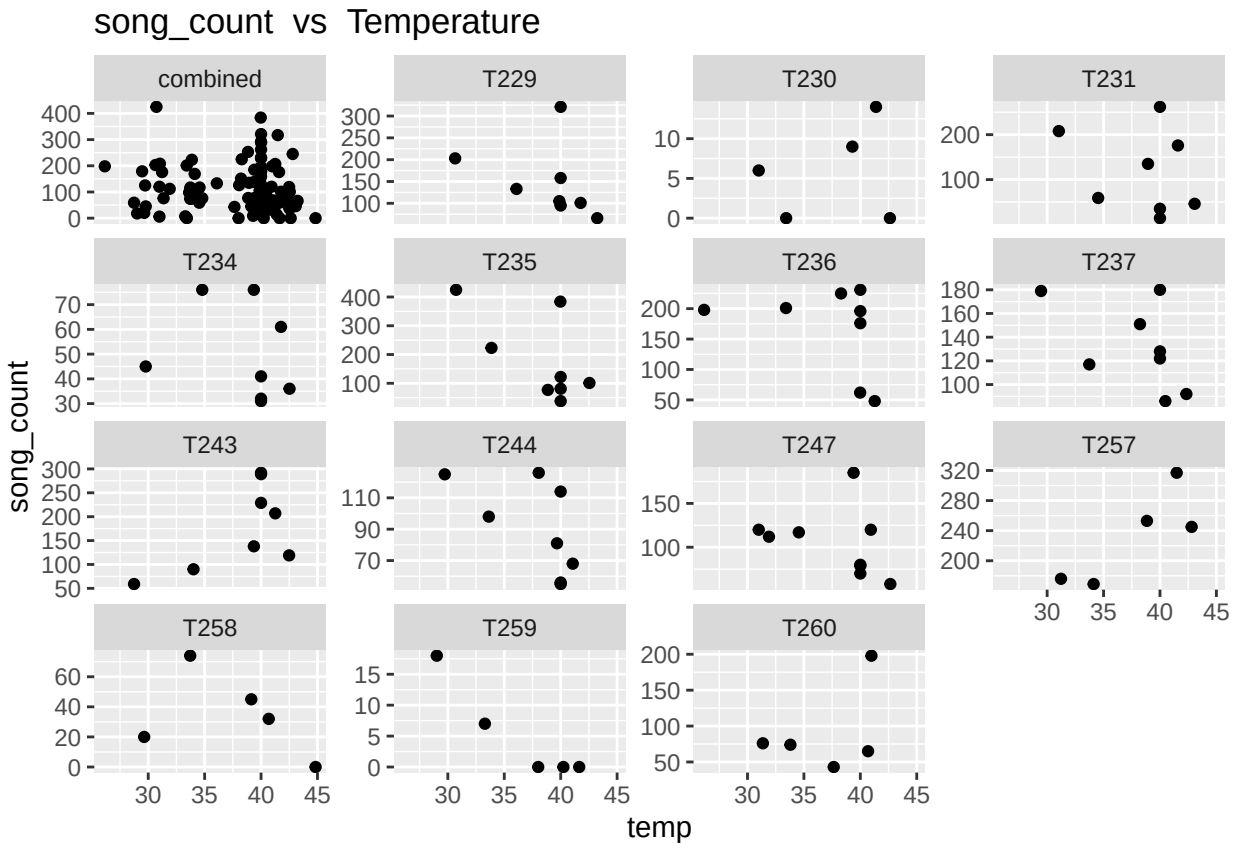
```
count_total_round_2_and_3  <- data_ind %>%
    select(c(male, count_total_round, round)) %>%
    unique() %>%
    pivot_wider(names_from = "round", values_from = "count_total_round")%>%
    select(male, `2`, `3`) %>%
    print(n=100)
```

```
## # A tibble: 14 x 3
##    male    `2`   `3`
##    <fct> <dbl> <dbl>
##  1 T247    229   712
##  2 T236    603   734
##  3 T243    810   613
##  4 T234    104   294
##  5 T244    225   498
##  6 T231    311   624
##  7 T235    241  1210
##  8 T229    574   608
##  9 T237    430   625
## 10 T230     NA    29
## 11 T259     NA    25
## 12 T258     NA   171
## 13 T257     NA  1160
## 14 T260     NA   456
```

Rounds 2 and 3 vs Temp

```
xlab <- "Temperature"
ylab <- "song_count"

plot_temp_data <-
    ggplot(data) +
    aes(x = temp,
        y = song_count) +
    facet_wrap("male", scales = "free_y") +
    geom_point() +
labs( title = paste( ylab, " vs ", xlab))
last_plot()
```




```
pivot_wider(data_ind, names_from = "round", values_from = "count_total_round") %>% select(male, `2`, `3
```

```
## # A tibble: 14 x 3
##    male    `2`   `3`
##    <fct> <dbl> <dbl>
##  1 T247     NA   712
##  2 T236     NA   734
##  3 T230     NA    29
##  4 T231     NA   624
##  5 T235     NA  1210
##  6 T229     NA   608
##  7 T234     NA   294
##  8 T244     NA   498
##  9 T237     NA   625
## 10 T243     NA   613
## 11 T259     NA    25
## 12 T258     NA   171
## 13 T257     NA  1160
## 14 T260     NA   456
```

Formal Model Fits to song_count

Using

- x = temp -45C

- use `offset(log(count_total_round))` instead of `male` as a factor

```
temp_ref <- 45
verbose <- 0
trace <- FALSE

## Try filtering the data a bit more
## Goal is to get good starting values

data <- data_ind %>%
    mutate(x1 = (temp - temp_ref)) %>%
    filter(
    (round == 3 & count_total_round >= 400) |
    (round == 2 & count_total_round > 30)
    ) %>%
    mutate() %>%
#    filter( !(male %in% c("T231", "T260"))) %>%
    mutate()

make_plot = TRUE;

glm_poisson_1 <- glm(song_count ~
                         (1 + male  +  (x1) + I(x1^2)),
                     data = data,
                     family = poisson(link = "log")
                     )

summary(glm_poisson_1)
```

```
##
## Call:
## glm(formula = song_count ~ (1 + male + (x1) + I(x1^2)), family = poisson(link = "log"),
##     data = data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -12.6200   -3.6484   -0.4908    2.5768   13.5329
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.7580949  0.0504036  94.400  < 2e-16 ***
## maleT231    -0.2435758  0.0437871  -5.563 2.66e-08 ***
## maleT234    -1.4392007  0.1023694 -14.059  < 2e-16 ***
## maleT235     0.1846495  0.0392826   4.701 2.59e-06 ***
## maleT236     0.1016570  0.0401238   2.534  0.01129 *
## maleT237    -0.1344800  0.0424249  -3.170  0.00153 **
## maleT243     0.1747338  0.0393992   4.435 9.21e-06 ***
## maleT244    -0.5228284  0.0473814 -11.034  < 2e-16 ***
## maleT247    -0.3732199  0.0438226  -8.517  < 2e-16 ***
## maleT257     0.4257667  0.0414650  10.268  < 2e-16 ***
## maleT260    -0.5349934  0.0555356  -9.633  < 2e-16 ***
## x1          -0.0556144  0.0115458  -4.817 1.46e-06 ***
## I(x1^2)     -0.0020475  0.0006101  -3.356  0.00079 ***
## ---
```

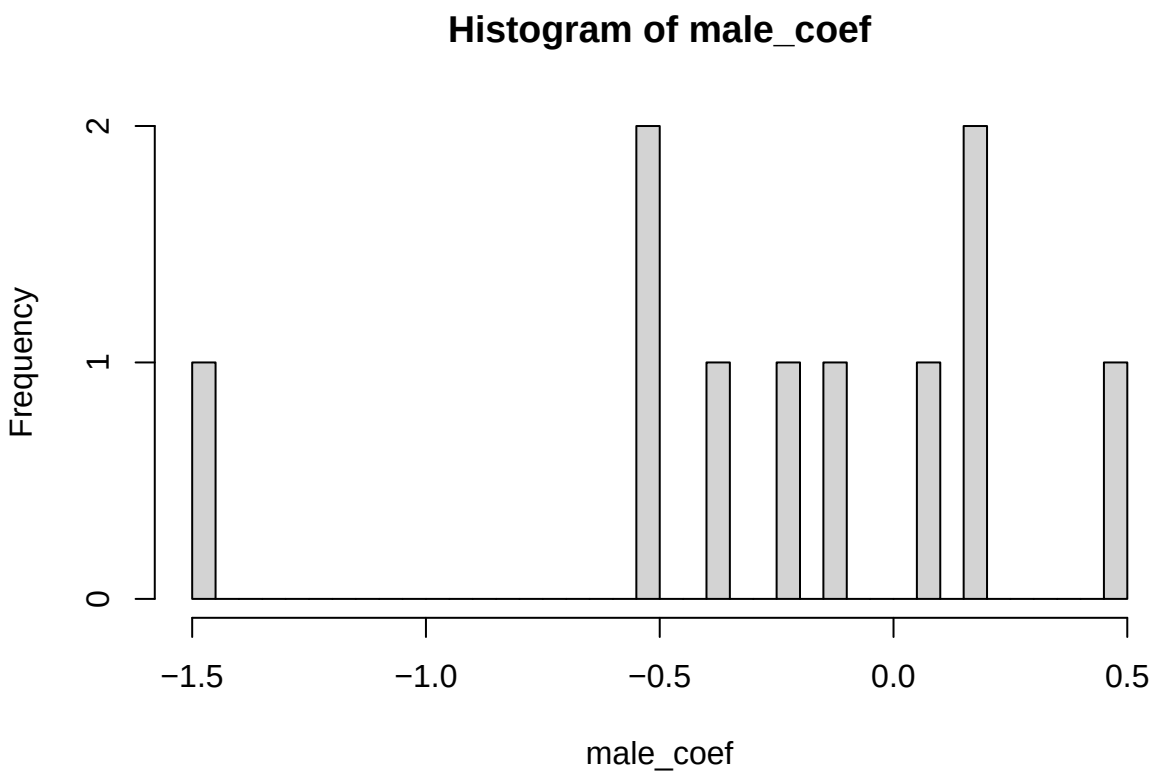
```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 3837.1  on 77  degrees of freedom
## Residual deviance: 2586.5  on 65  degrees of freedom
## AIC: 3125
##
## Number of Fisher Scoring iterations: 5
```

```
## Add round effect
glm_poisson_2 <- glm(song_count ~
                       (1 + male + round +  (x1) + I(x1^2)),
                     data = data,
                     family = poisson(link = "log")
                     )
## Results support round effect
summary(glm_poisson_2)
```

```
##
## Call:
## glm(formula = song_count ~ (1 + male + round + (x1) + I(x1^2)),
##     family = poisson(link = "log"), data = data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -13.1613   -3.8931   -0.4917    2.5266   14.1571
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.9970194  0.0806312  61.974  < 2e-16 ***
## maleT231    -0.2436248  0.0437865  -5.564 2.64e-08 ***
## maleT234    -1.4883981  0.1031520 -14.429  < 2e-16 ***
## maleT235     0.1845115  0.0392786   4.698 2.63e-06 ***
## maleT236     0.0958580  0.0401748   2.386 0.017032 *
## maleT237    -0.1361389  0.0424249  -3.209 0.001332 **
## maleT243     0.1720600  0.0394070   4.366 1.26e-05 ***
## maleT244    -0.5236896  0.0473742 -11.054  < 2e-16 ***
## maleT247    -0.3712532  0.0438229  -8.472  < 2e-16 ***
## maleT257     0.4554935  0.0422389  10.784  < 2e-16 ***
## maleT260    -0.5048986  0.0561333  -8.995  < 2e-16 ***
## round       -0.0899197  0.0236740  -3.798 0.000146 ***
## x1          -0.0516251  0.0116725  -4.423 9.74e-06 ***
## I(x1^2)     -0.0016452  0.0006215  -2.647 0.008118 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 3837.1  on 77  degrees of freedom
## Residual deviance: 2572.1  on 64  degrees of freedom
## AIC: 3112.6
##
## Number of Fisher Scoring iterations: 5
```

```r
male_coef <-  coef(glm_poisson_2) %>% keep(str_detect(names(.), 'male'))

hist(male_coef, breaks = 30)
```




```r
## Remove male effect
glm_poisson_3 <- glm(song_count ~
                       (1 + round +  (x1) + I(x1^2)),
                     data = data,
                     family = poisson(link = "log")
                     )
## Note that using qpoisson doesn't affect MLE

summary(glm_poisson_3)
```

```
##
## Call:
## glm(formula = song_count ~ (1 + round + (x1) + I(x1^2)), family = poisson(link = "log"),
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -13.066   -5.905   -1.983    3.809   17.600
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.7454034  0.0736686  64.416   <2e-16 ***
## round        0.0226110  0.0223986   1.009    0.313
## x1          -0.0153263  0.0116210  -1.319    0.187
```

```
## I(x1^2)      0.0002044  0.0006179   0.331    0.741
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 3837.1  on 77  degrees of freedom
## Residual deviance: 3762.8  on 74  degrees of freedom
## AIC: 4283.3
##
## Number of Fisher Scoring iterations: 5
```

```
if(make_plot){
    ifelse(length(dev.list()) < 3, {dev.new(); dev.next()}, dev.next())
    par(mfrow = c(2, 2))
    plot(glm_poisson_2, ask = FALSE)
    page_label <- "glm with poisson error"
    mtext(page_label, outer=TRUE,  cex=1, line=-1.1)
}

glm_qpoisson_1 <- update(glm_poisson_1,
                      family = quasipoisson(link = "log")
                      )
summary(glm_qpoisson_1)
```

```
##
## Call:
## glm(formula = song_count ~ (1 + male + (x1) + I(x1^2)), family = quasipoisson(link = "log"),
##     data = data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -12.6200   -3.6484   -0.4908    2.5768   13.5329
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.758095   0.319628  14.886   <2e-16 ***
## maleT231    -0.243576   0.277670  -0.877   0.3836
## maleT234    -1.439201   0.649162  -2.217   0.0301 *
## maleT235     0.184650   0.249105   0.741   0.4612
## maleT236     0.101657   0.254439   0.400   0.6908
## maleT237    -0.134480   0.269032  -0.500   0.6189
## maleT243     0.174734   0.249845   0.699   0.4868
## maleT244    -0.522828   0.300463  -1.740   0.0866 .
## maleT247    -0.373220   0.277895  -1.343   0.1839
## maleT257     0.425767   0.262945   1.619   0.1102
## maleT260    -0.534993   0.352172  -1.519   0.1336
## x1          -0.055614   0.073216  -0.760   0.4502
## I(x1^2)     -0.002048   0.003869  -0.529   0.5984
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 40.21295)
##
```

```
##     Null deviance: 3837.1  on 77  degrees of freedom
## Residual deviance: 2586.5  on 65  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

```
if(make_plot){
    ifelse(length(dev.list()) < 3, dev.new(), dev.next())
    par(mfrow = c(2, 2))
    plot(glm_qpoisson_1, ask = FALSE)
    page_label <- "glm with poisson error"
    mtext(page_label, outer=TRUE,  cex=1, line=-1.1)
##    dev.new()
##    ggplot(data, aes(temp, song_count))
}
```

```
## Warning in rep(yes, length.out = len): 'x' is NULL so the result will be NULL
```

```
## Error in ans[ypos] <- rep(yes, length.out = len)[ypos]: replacement has length zero
```

```
glm_qpoisson_2 <- update(glm_poisson_2,
                      family = quasipoisson(link = "log")
                      )
summary(glm_qpoisson_2)
```

```
##
## Call:
## glm(formula = song_count ~ (1 + male + round + (x1) + I(x1^2)),
##     family = quasipoisson(link = "log"), data = data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -13.1613   -3.8931   -0.4917    2.5266   14.1571
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.997019   0.512793   9.745 2.89e-14 ***
## maleT231    -0.243625   0.278471  -0.875   0.3849
## maleT234    -1.488398   0.656020  -2.269   0.0267 *
## maleT235     0.184512   0.249802   0.739   0.4628
## maleT236     0.095858   0.255501   0.375   0.7088
## maleT237    -0.136139   0.269811  -0.505   0.6156
## maleT243     0.172060   0.250618   0.687   0.4949
## maleT244    -0.523690   0.301288  -1.738   0.0870 .
## maleT247    -0.371253   0.278702  -1.332   0.1876
## maleT257     0.455493   0.268628   1.696   0.0948 .
## maleT260    -0.504899   0.356993  -1.414   0.1621
## round       -0.089920   0.150560  -0.597   0.5525
## x1          -0.051625   0.074234  -0.695   0.4893
## I(x1^2)     -0.001645   0.003953  -0.416   0.6786
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for quasipoisson family taken to be 40.44634)
##
##     Null deviance: 3837.1  on 77  degrees of freedom
## Residual deviance: 2572.1  on 64  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

```
if(make_plot){
    ifelse(length(dev.list()) < 3, dev.new(), dev.next())
    par(mfrow = c(2, 2))
    plot(glm_qpoisson_2, ask = FALSE)
    page_label <- "glm with poisson error"
    mtext(page_label, outer=TRUE,  cex=1, line=-1.1)
##    dev.new()
##    ggplot(data, aes(temp, song_count))
}
```

```
glm_qpoisson_3 <- update(glm_poisson_3,
                         family = quasipoisson(link = "log")
                         )
summary(glm_qpoisson_3)
```

```
##
## Call:
## glm(formula = song_count ~ (1 + round + (x1) + I(x1^2)), family = quasipoisson(link = "log"),
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -13.066   -5.905   -1.983    3.809   17.600
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.7454034  0.5437615   8.727 5.46e-13 ***
## round        0.0226110  0.1653278   0.137    0.892
## x1          -0.0153263  0.0857765  -0.179    0.859
## I(x1^2)      0.0002044  0.0045605   0.045    0.964
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 54.48182)
##
##     Null deviance: 3837.1  on 77  degrees of freedom
## Residual deviance: 3762.8  on 74  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

```
if(make_plot){
    ifelse(length(dev.list()) < 3, dev.new(), dev.next())
```

```
    par(mfrow = c(2, 2))
    plot(glm_qpoisson_3, ask = FALSE)
    page_label <- "glm with poisson error"
    mtext(page_label, outer=TRUE,  cex=1, line=-1.1)
##    dev.new()
##    ggplot(data, aes(temp, song_count))
}
```

Result

- Overdispersion of data makes parameters non-significant
- Don't how a `song_prop` approach will solve this issue.