Third Fitting of Thermal Models

Michael Gilchrist

date: 2022-10-20

Goal

- Fit series of thermal models, including, ultimately, those found in `rTPC` to data collected in the Derryberry lab.

Recap

Previous work suggests that

- 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.

Current Work

- 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?

Next Steps

Which Explanatory Variable: temp, humidity, vpd?

- Liz will look into operative temp
- Use of `vpd_mean` is somewhat arbitrary since it depends on the experimental design.
  - Use of 0 reference in vpd seems less arbitrary than using 0 C.
- Note that we could scale temp_mean relative to C = 45, which is the understood thermal maximum.
  - This is what Liz wants to do since it's easier to interpret

Including beak and mass data

- Include beak size (surface area) as additional explanatory variable. We do have bird mass as well
- Notes from Liz
  - Do birds with larger beaks maintain singing at higher rates at higher temps?
  - We calculated bill surface area approximately as the surface area of a cone: length * pi * (width + depth) / 4.
  - Because larger animals produce more heat (Kleiber 1932), we scaled bill size relative to heat production by dividing bill surface area by expected daily energy consumption (mass0.658; Speakman and Kr´ol 2010, Hudson et al. 2013).
  - So, looks like we need to calculate bill surface area and then scale relative to heat production (bill surface area/mass0.658)

Type of Model Fitted

- Don't worry about random effects for now
- Try the quadratic fit with just temperature and the 0 set at 45C.
  - Quadratic function – pull out the peak, the curvature at the peak, calculate the intercept at some temp – what is the intercept
- Begin fitting rTPC models to combined dataset.

Using Data from Rounds 1 and 2

- Liz: Ask Kayci about `temp_mean` and `humidity_mean` data for rounds 1 and 2.
- Look for order effects in round 1 and 2? Can we use any of this data? Seems like there will be an issue if we throw out, say, first two trials, when using `song_prop`. Note that since total_count is consistent for a bird between rounds, this may not be an issue.
- Begin fitting rTPC models to combined dataset.
- We could include round = 2
  - Would need to down weight song_count values when combining across `count_total_round` values.

Additional Liz

- Follow up with Ray/Juan about ground versus surface temp for operative temperature.
- What is the 0 for operative temperature?
- Ponder utility of other ZF data on panting.

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   0.3.4
## v tibble  3.1.8           v dplyr   1.0.99.9000
## v tidyr   1.2.0.9000      v stringr 1.4.1
## v readr   2.1.2           v forcats 0.5.2
## -- Conflicts ------------------------------------------------------------------ tidyverse_conflic
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
```

```
require(ggplot2)
require(ggpubr)
```

```
## Loading required package: ggpubr
```

```
require(viridisLite)
```

```
## Loading required package: 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)

require(GGally)
```

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

```
require(reshape2)
```

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

```
require(lme4)
```

```
## Loading required package: lme4
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

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

```
## Loading required package: RVAideMemoire
## *** 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
```

```
require(humidity) ## provides VPD
```

```
## Loading required package: humidity
```

```
require(weathermetrics)
```

```
## Loading required package: weathermetrics
```

```
require(latex2exp)
```

```
## Loading required package: 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")) %>% m
    ## 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)) %>%
    ## 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
    mutate(count_wt = 1/(song_count + 1)) %>%
    ## need to rescale wts for song_prop data
    mutate(prop_wt = count_wt * count_total_round^2) %>%
    ## 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
```

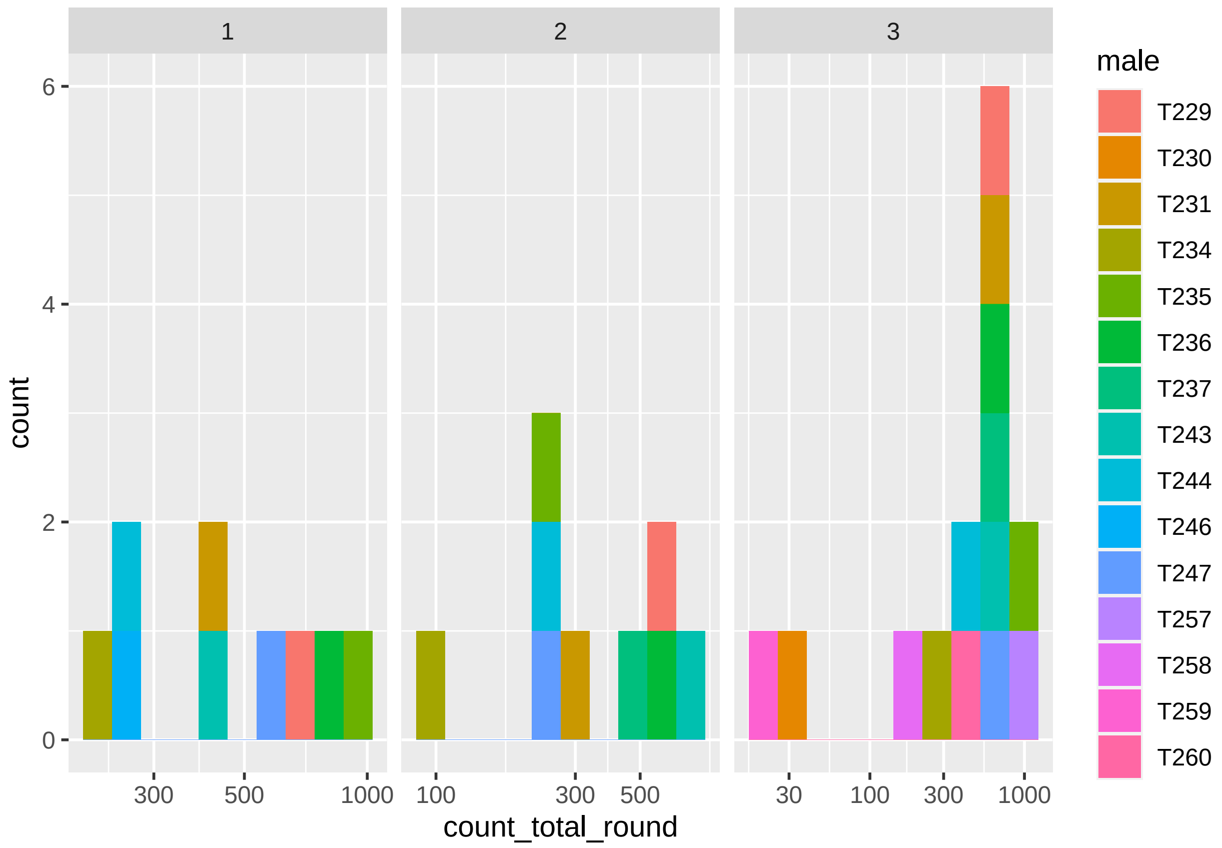
```
## Joining, by = c("male", "chamber", "date", "song_count", "counter", "test_order", "temp_target", "ro
## Joining, by = c("male", "chamber", "date", "song_count", "counter", "test_order", "temp_target", "ro
```

Third Analysis for Liz

Examine Data

```
data_count_total <- data_full %>% group_by(round) %>%
    select(male, round, count_total_round)  %>%
    distinct()

t <- ggplot(data_count_total, aes(count_total_round, fill = male)) +
    geom_histogram(bins = 10) +
    scale_x_log10()

hist_count_total <- t +
    facet_grid(cols =vars(round), scales = "free_x")
hist_count_total
```

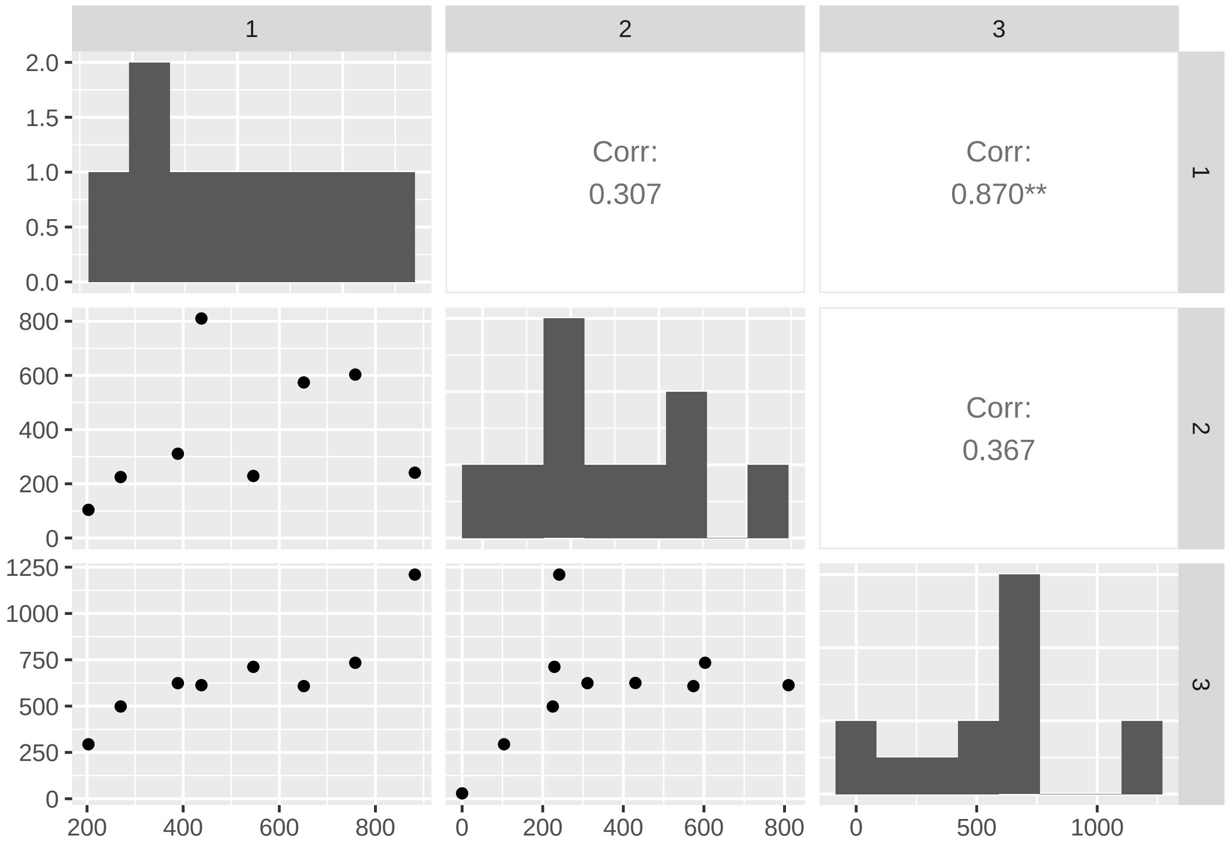
```
## Warning: Transformation introduced infinite values in continuous x-axis
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```



## Compare `count_total_round` between round 1 and 3

```
count_total_by_round <- data_full %>%
      select(male, round, count_total_round) %>%
      unique() %>%
      pivot_wider(names_from = round, values_from = count_total_round)


## Modified from https://stackoverflow.com/a/68553749/5322644
diag_plots <- function(data, mapping, ...) {
    ggplot(data = data, mapping = mapping) +
        # choose color by counter and send bin width argument in
        geom_histogram(...)
}

# pairs plot
ggpairs(count_total_by_round %>% select(-male),
        diag = list(continuous = wrap(diag_plots, bins = 8))
        )
```

```
## Warning: Removed 6 rows containing non-finite values (stat_bin).

## Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
## Removed 7 rows containing missing values
## Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
## Removed 7 rows containing missing values

## Warning: Removed 7 rows containing missing values (geom_point).

## Warning: Removed 5 rows containing non-finite values (stat_bin).

## Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
## Removed 5 rows containing missing values

## Warning: Removed 7 rows containing missing values (geom_point).

## Warning: Removed 5 rows containing missing values (geom_point).

## Warning: Removed 1 rows containing non-finite values (stat_bin).
```

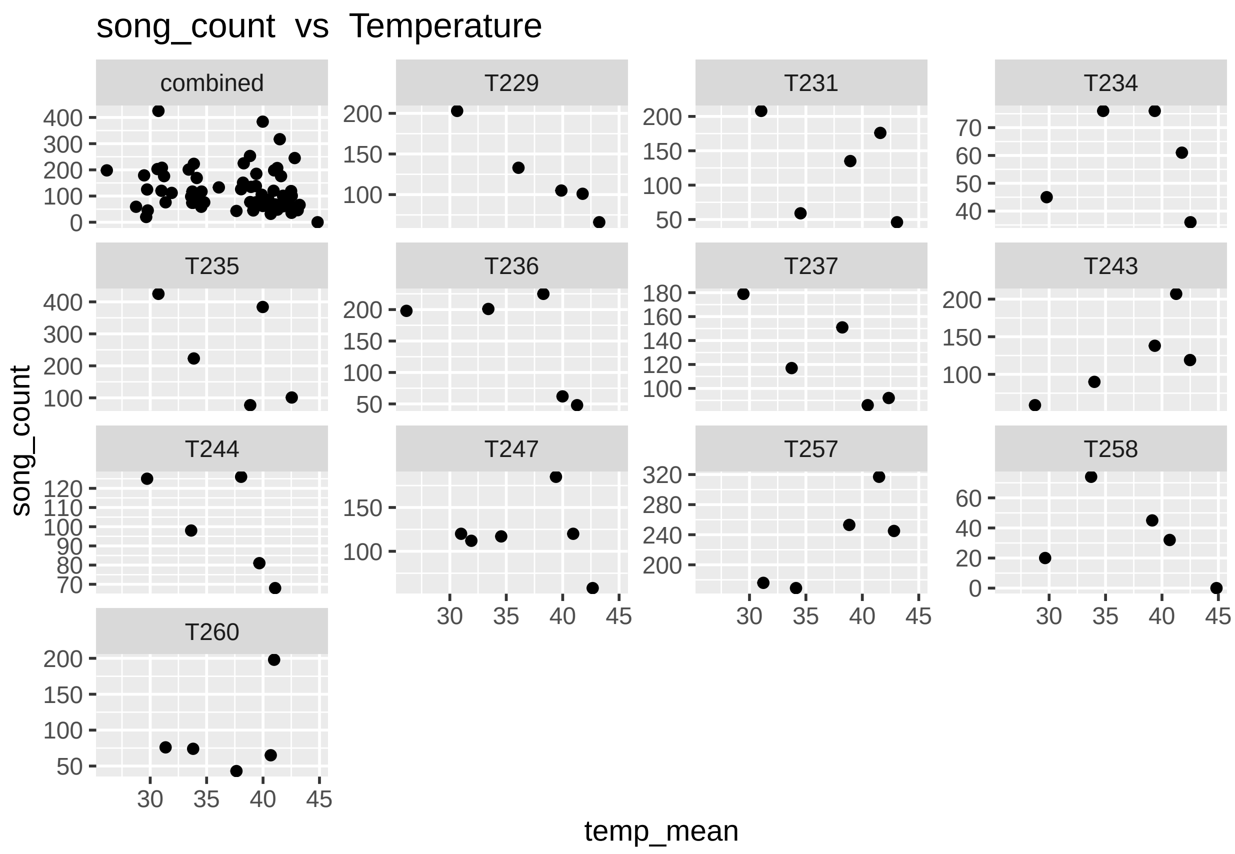


Result

- As before, we see strong consistancy between `round` 1 and 3.
- Consistency with round 2 is weaker, but sample sizes are smaller: 3 trials/male in round 2 vs 6 trials/male in round 3.

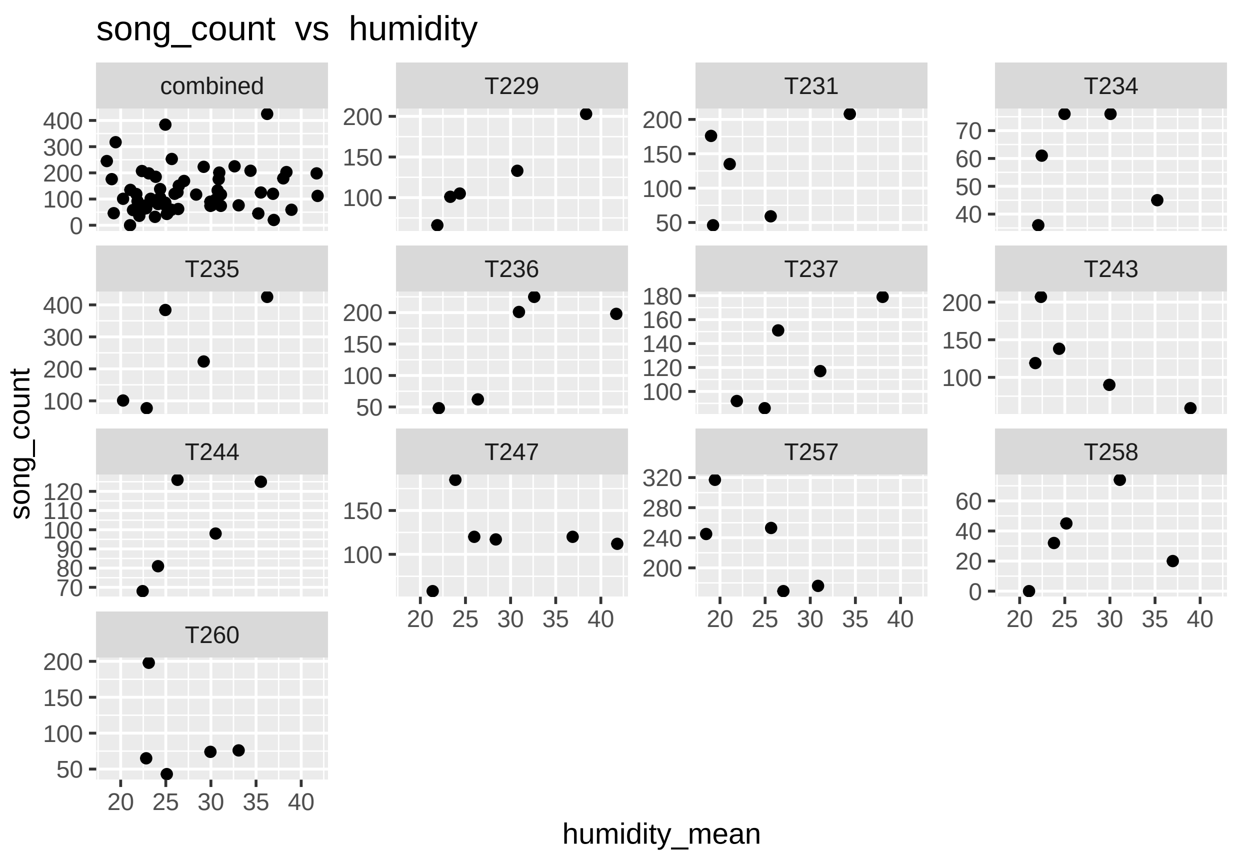
Create & Plot Filtered Data

```
data_ind <- data_full %>%
    filter(round==3) %>%
    filter(count_total_round >=150)
## copy data frame and assign `male =  "combined")
data_comb <- data_ind %>% mutate(male = "combined")

data <- bind_rows(data_ind, data_comb)

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

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

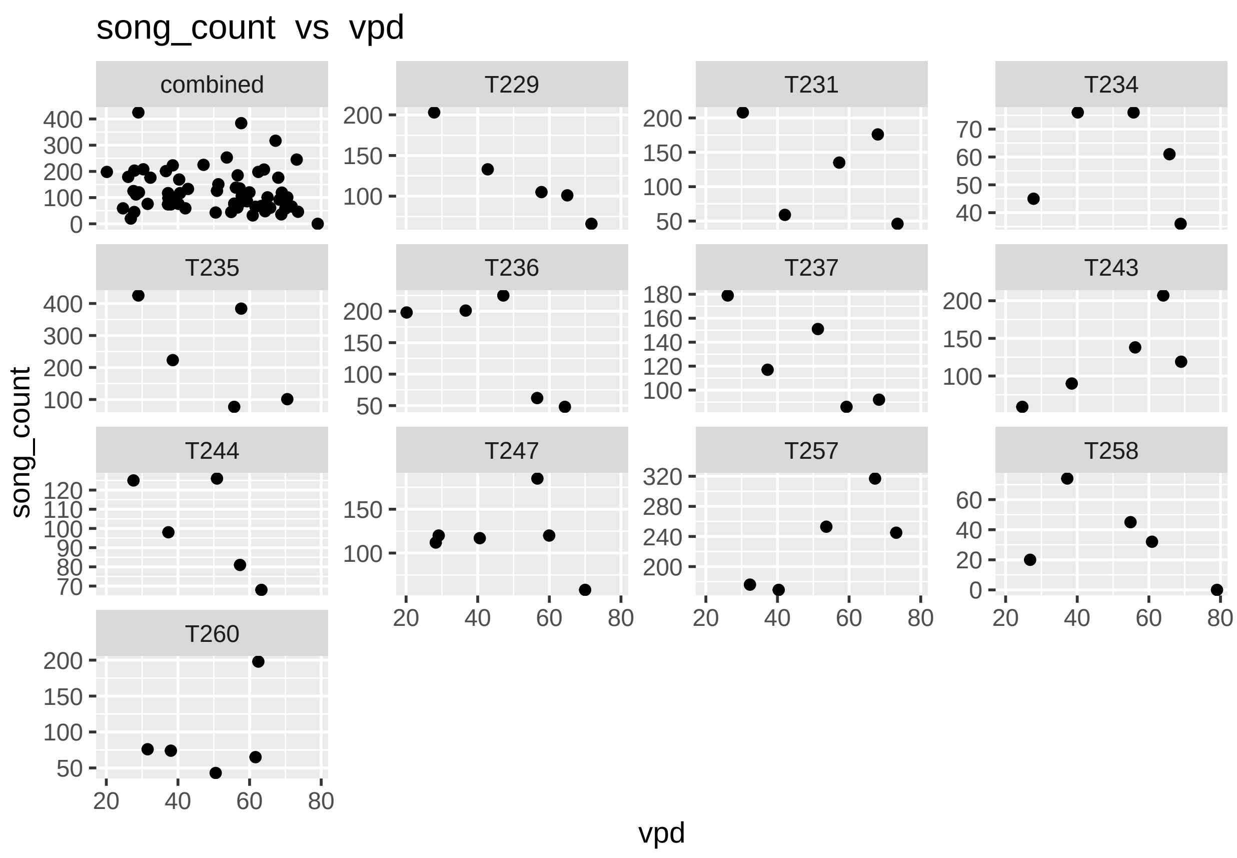


```
## Create again for humidity

xlab <- "humidity"

plot_humidity_data <-
```

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



```
## Create for vpd

xlab <- "vpd"

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

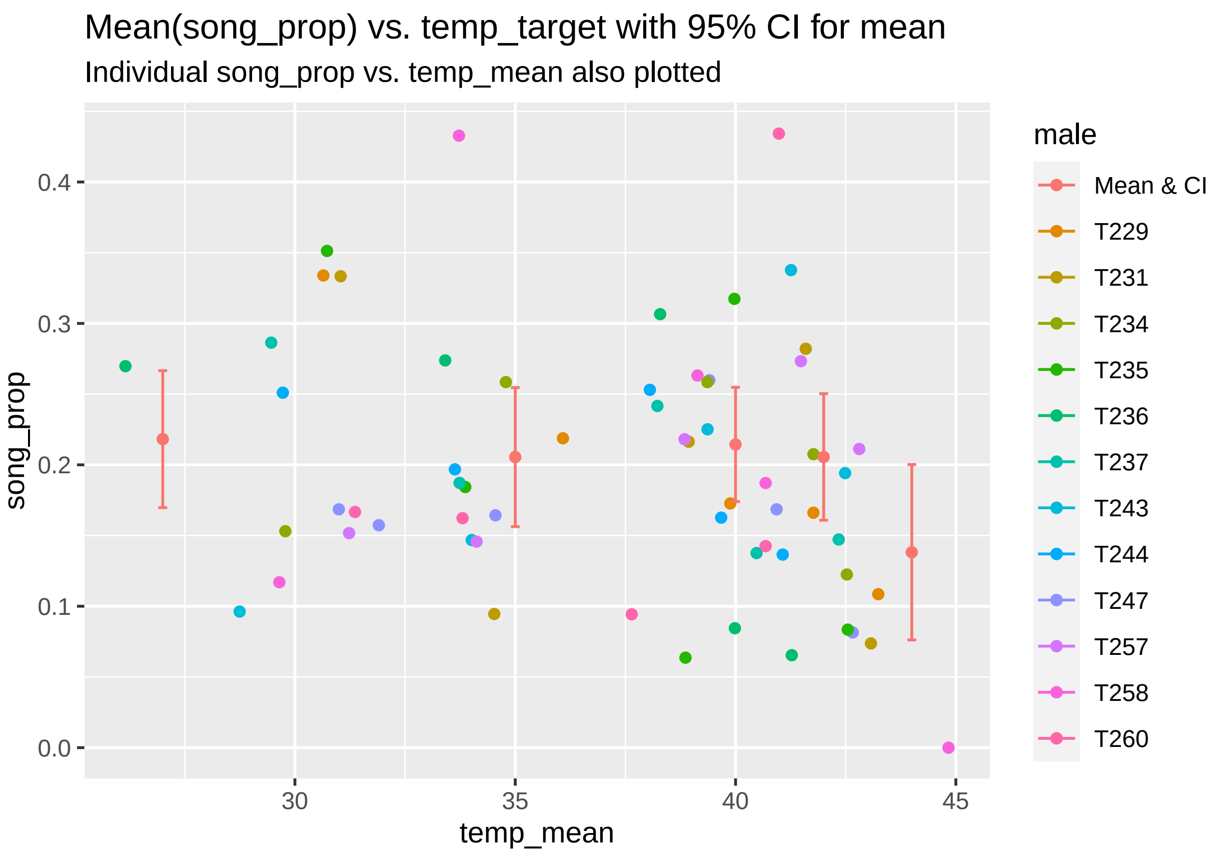


Work with mean and song_prop

- This approach won't work because `temp_target` varies greatly from `temp_mean`.

```
plot_combine <- ggplot(data_ind) +
                  aes(x = temp_mean, y = song_prop, color = male) +
                  geom_point()
# last_plot()

data_summarize <-
    data_ind %>% group_by(temp_target) %>%
    summarize(song_mean = mean(song_prop),
              song_sd = sd(song_prop),
              song_n= dplyr::n(),
              song_se = song_sd/sqrt(song_n),
              song_ci = song_se*1.96
              )


plot_summarize <- ggplot(data_summarize) +
    aes(x = temp_target, y = song_mean) +
    geom_point()+
    geom_errorbar(aes(ymin=song_mean - song_ci, ymax=song_mean+song_ci), width=.2)
# last_plot()
```

```
plot_combine +
    geom_point(data = data_summarize, aes(x = temp_target, y = song_mean, colour = "Mean & CI"))+
    geom_errorbar(data = data_summarize, aes(x = temp_target,
                                             y = song_mean,
                                             ymin=song_mean - song_ci,
                                             ymax=song_mean+song_ci,
                                             colour= "Mean & CI"), width=.2) +
    labs(title = "Mean(song_prop) vs. temp_target with 95% CI for mean",
         subtitle = "Individual song_prop vs. temp_mean also plotted")
```



Result

- Data is noisy.
- Using `song_prop` reduces impact of male `T225` on lower temps.
- Note `temp_target` is not necessarily accurate.

Humidity, Temp, and VPD

Previous Results from `second.fitting.pdf`

- Values clearly co-vary.
- Should consider using
  - Vapor Pressure Deficit
    - Uses temp to calculate max humidity and then looks at difference with relative humidity.

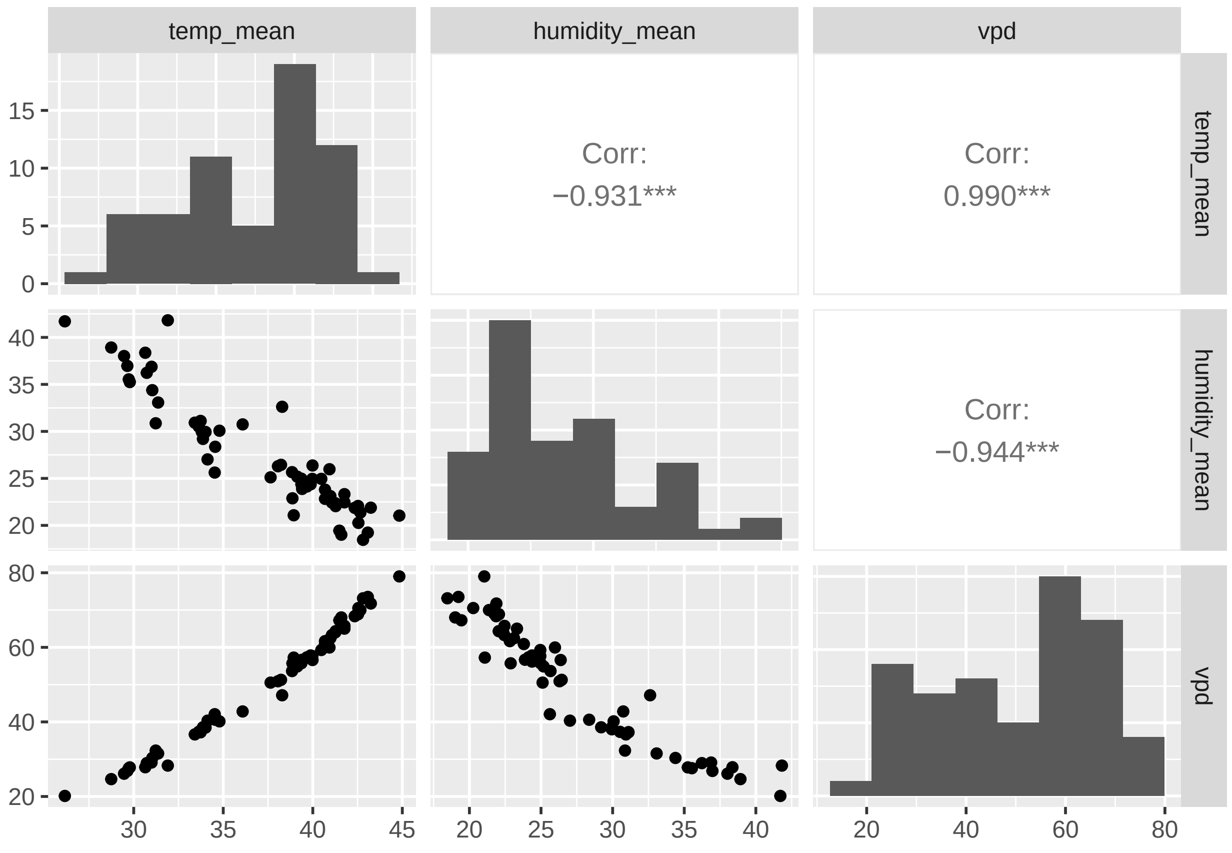
- Heat Index
  - Adjusted for birds if available
  - Use just first terms (`c_1 + c_2 T + c_3 R + c_4 T R + ...`)

Using VPD as Predictor

- VPD = vapor pressure deficit = actual vapor pressure - saturated vapor pressure
  - VPD = $VPD = vp_{\text{sat}} \times (1 - \text{relative humidity}/100)$
- `humidity` package provide saturated vapor pressure `SVP()`
  - Note doesn't work with temps in C despite `isK` argument.

Compare Temp, Humidity, and VPD as predictors

```
thv <- data_ind %>%
      select(male, temp_mean, humidity_mean, vpd) %>%
      unique()


# pairs plot
ggpairs(thv %>% select(-male),
        diag = list(continuous = wrap(diag_plots, bins = 8))
        )
```



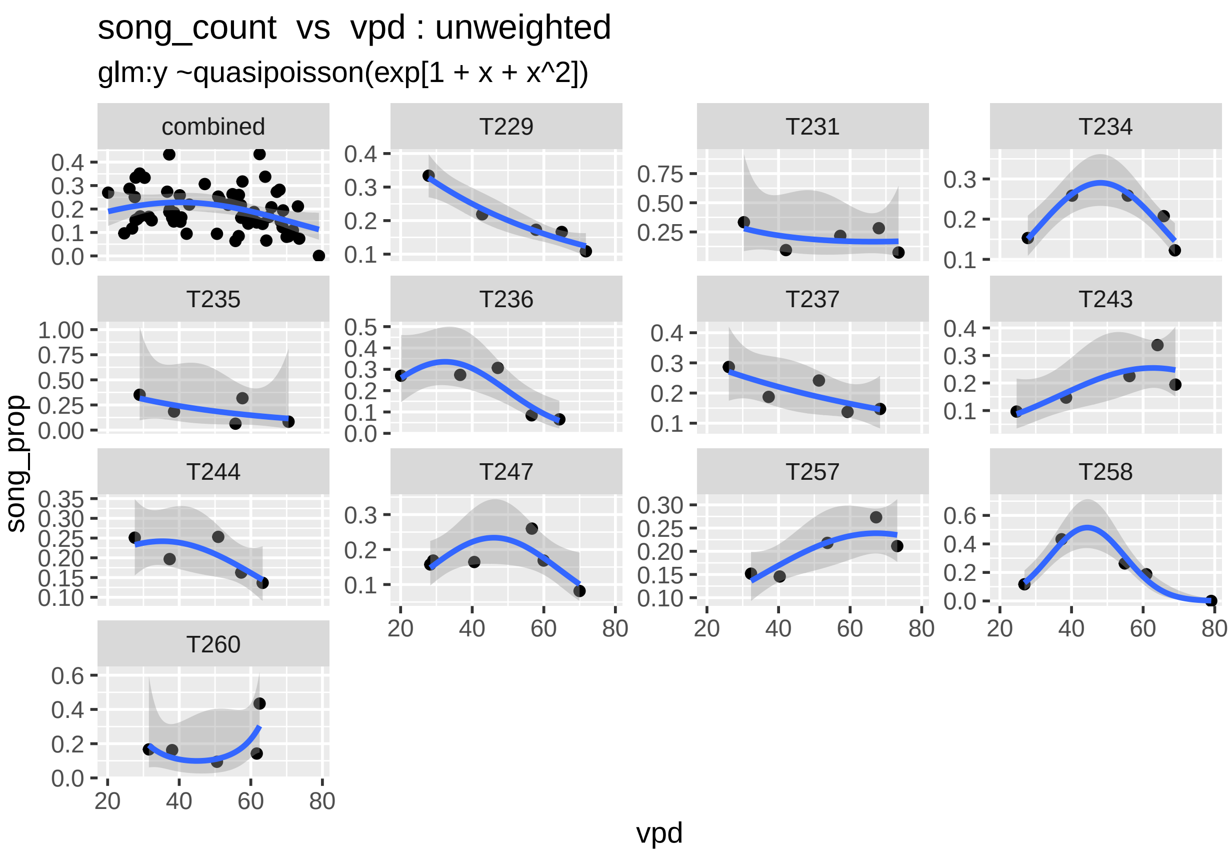
Result

- VPD and temp are *highly* correlated

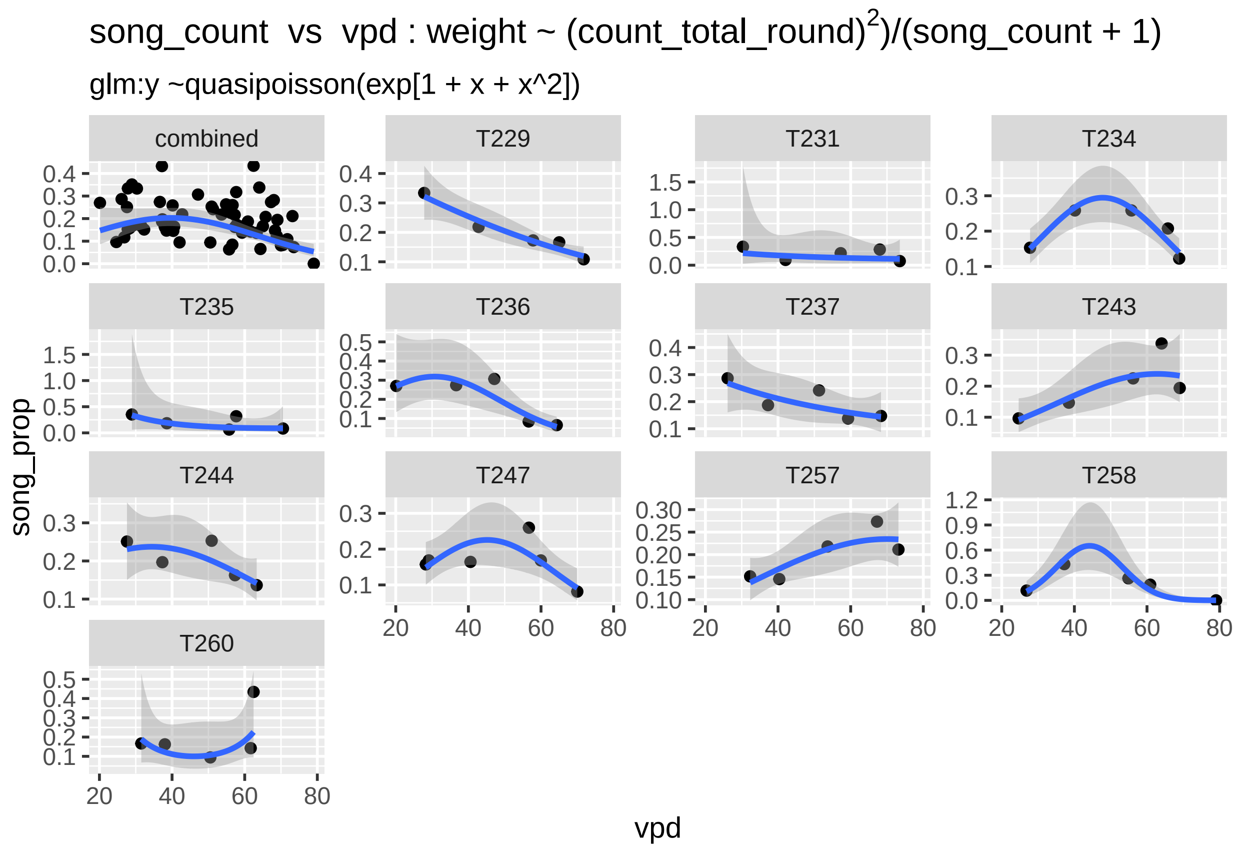
Model Fitting

Plots

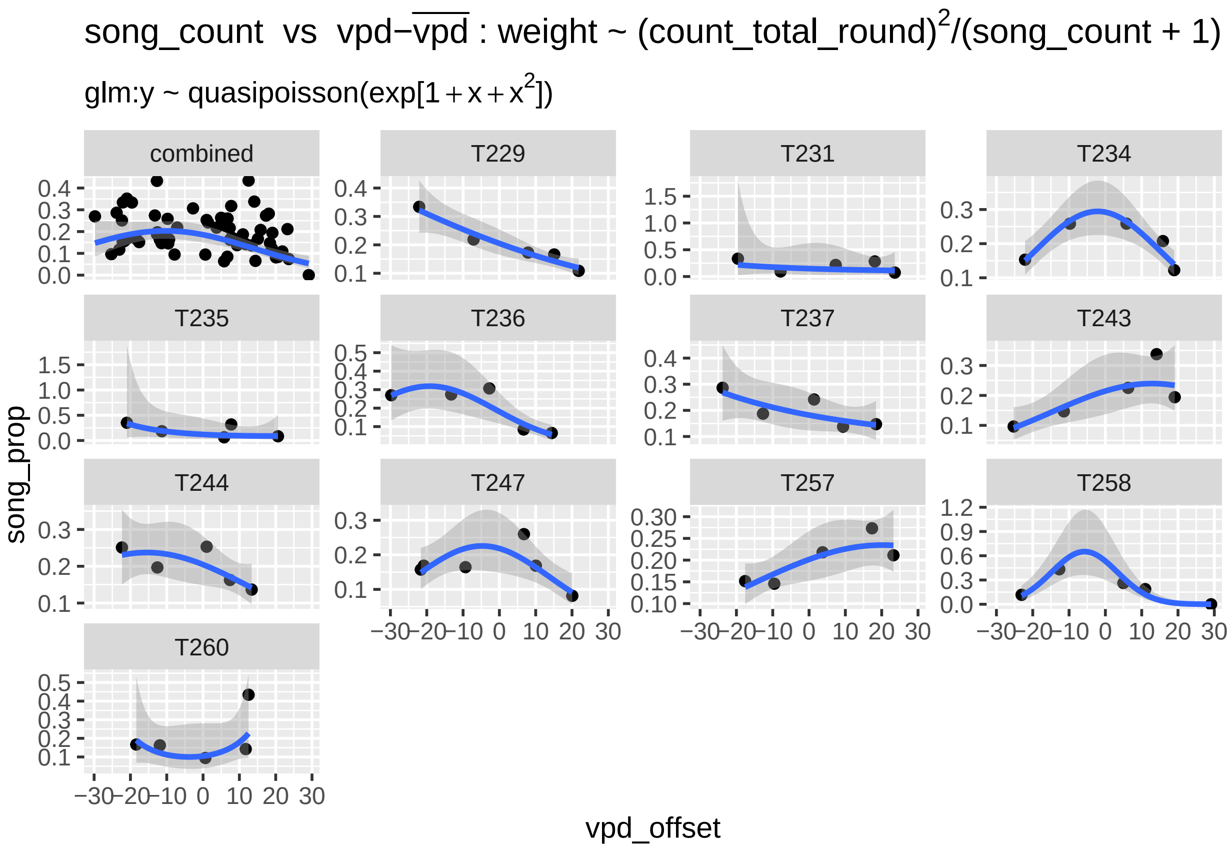
```
xlab = "vpd"


plot_tmp <- ggplot(data) +
    aes(x = vpd,
        y = song_prop) +
#    ylim(0, 0.6) +
    facet_wrap("male", scales = "free_y") +
    geom_point()

plot_glm_vpd <- plot_tmp +
    stat_smooth(method = "glm",
                method.args = list(
                    family = quasipoisson(link = "log"),
                    maxit = 100),
                se = TRUE,
                formula = y ~ 1 + x + I(x^2),
                size = 1
                ) +
    labs( title = paste( ylab, " vs ", xlab, ": unweighted"),
         subtitle = "glm:y ~quasipoisson(exp[1 + x + x^2])"
         )
last_plot()
```



```
plot_glm_vpd_weighted <- plot_tmp +
    stat_smooth(method = "glm",
                method.args = list(
                    family = quasipoisson(link = "log"),
                    maxit = 100),
                    aes(weight = prop_wt),
                se = TRUE,
                formula = y ~ 1 + x + I(x^2), size = 1,
                ) +
    labs( title = TeX(paste( ylab, " vs ", xlab, ": weight ~ (count_total_round)$^2$)/(song_count + 1)"
        subtitle = "glm:y ~quasipoisson(exp[1 + x + x^2])"
        )
last_plot()
```



```r
xlab <- "vpd-$\\bar{vpd}$"

plot_glm_vpd_weighted_centered <- ggplot(data) +
    aes(x = vpd_offset,
        y = song_prop) +
    facet_wrap("male", scales = "free_y") +
    geom_point() +
    stat_smooth(method = "glm",
                method.args = list(
                    family = quasipoisson(link = "log"),
                    maxit = 100),
                    aes(weight = prop_wt),
                se = TRUE,
                formula = y ~ 1 + x + I(x^2), size = 1,
                ) +
    labs( title = TeX(paste( ylab, " vs ", xlab, ": weight ~ (count_total_round)$^2$/(song_count + 1)"))
          xlab = TeX(xlab),
          subtitle = TeX("glm:y ~ quasipoisson($\\exp[1 + x + x^2]$)")
          )
last_plot()
```



Result

- We do see a concave down curve if we naively fit a quadratic function to the log transformed data.
- Adding `weight = (count_total_round^2)/(song_count + 1)` greatly improves fit

Formal Model Fits to `song_count`

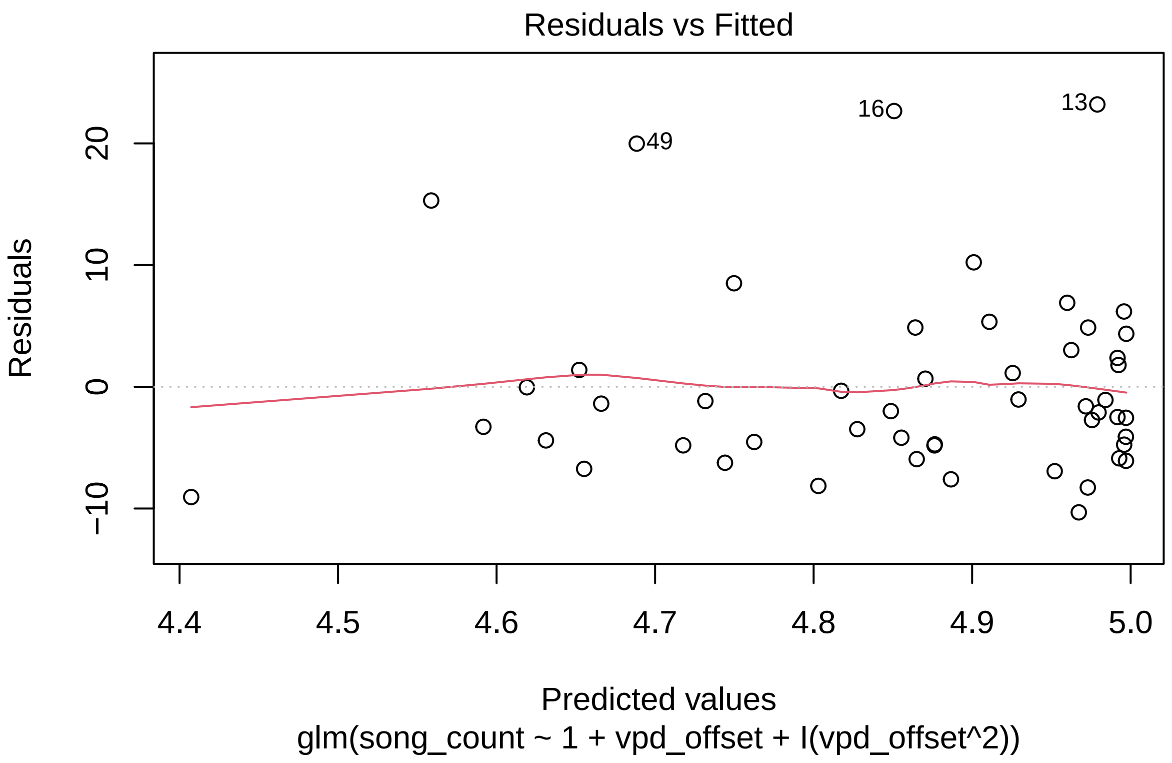
- Using midpoint of vpd

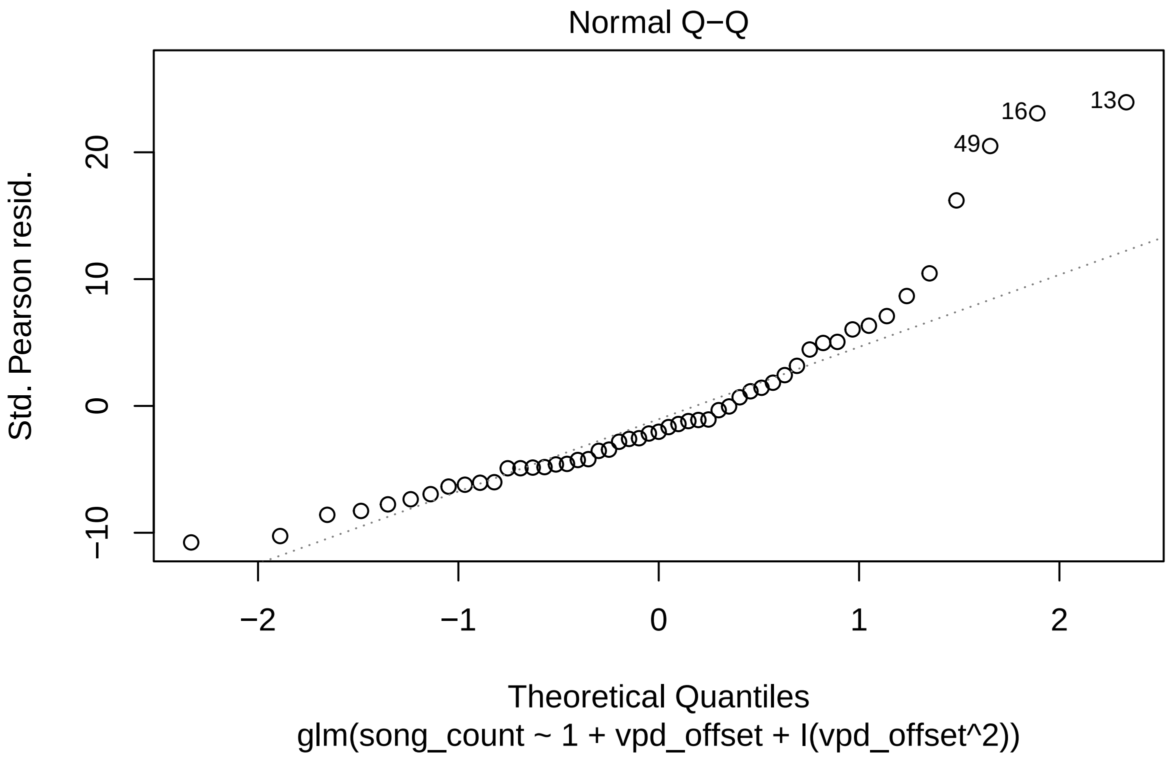
```r
## Try filtering the data a bit more
## Goal is to get good starting values
count_glm_poisson <- glm(song_count ~
            1 + vpd_offset + I(vpd_offset^2),
            data = data_ind %>% filter( !(male %in% c("T231", "T260"))),
        family = poisson(link = "log")
       )
summary(count_glm_poisson)
```

```
##
## Call:
## glm(formula = song_count ~ 1 + vpd_offset + I(vpd_offset^2),
##     family = poisson(link = "log"), data = data_ind %>% filter(!(male %in%
##         c("T231", "T260"))))
##
```

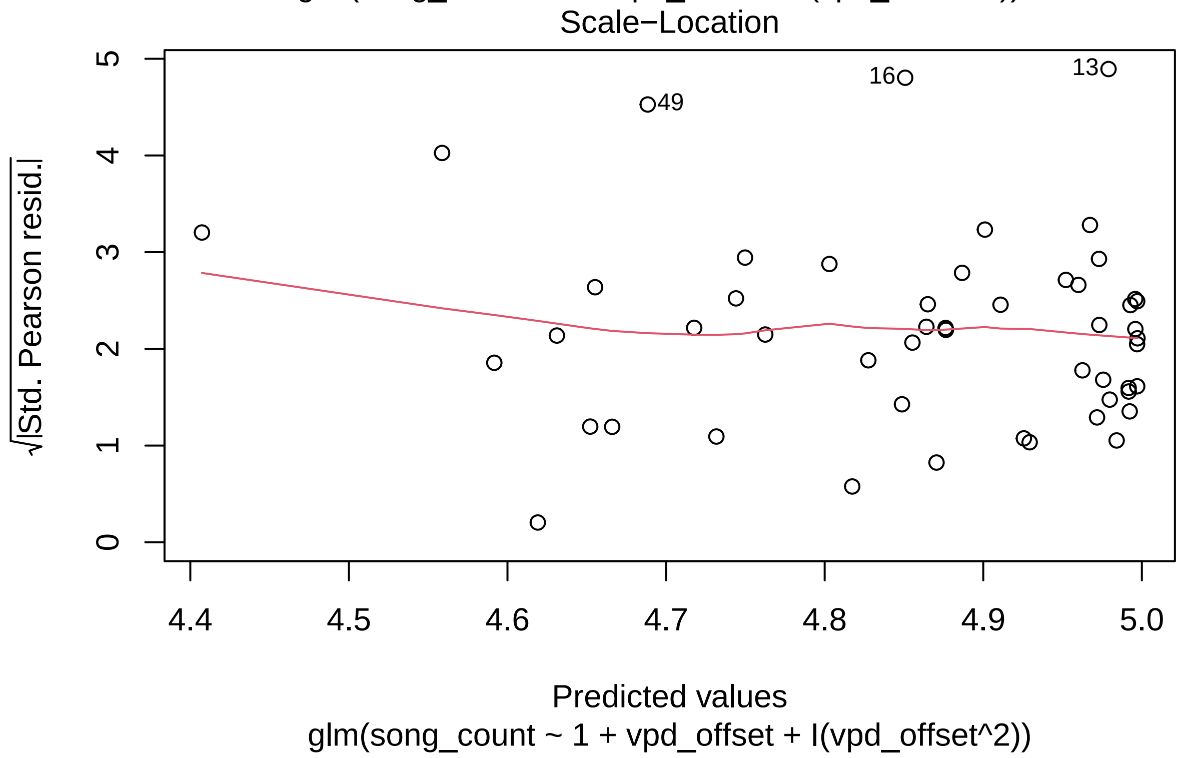
```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -12.977   -5.180   -2.063    2.601   18.783
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      4.938e+00  1.869e-02 264.270  < 2e-16 ***
## vpd_offset      -8.765e-03  8.343e-04 -10.506  < 2e-16 ***
## I(vpd_offset^2) -3.253e-04  5.943e-05  -5.474  4.4e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 2713.2  on 50  degrees of freedom
## Residual deviance: 2591.2  on 48  degrees of freedom
## AIC: 2923.8
##
## Number of Fisher Scoring iterations: 5
```

```
plot(count_glm_poisson)
```

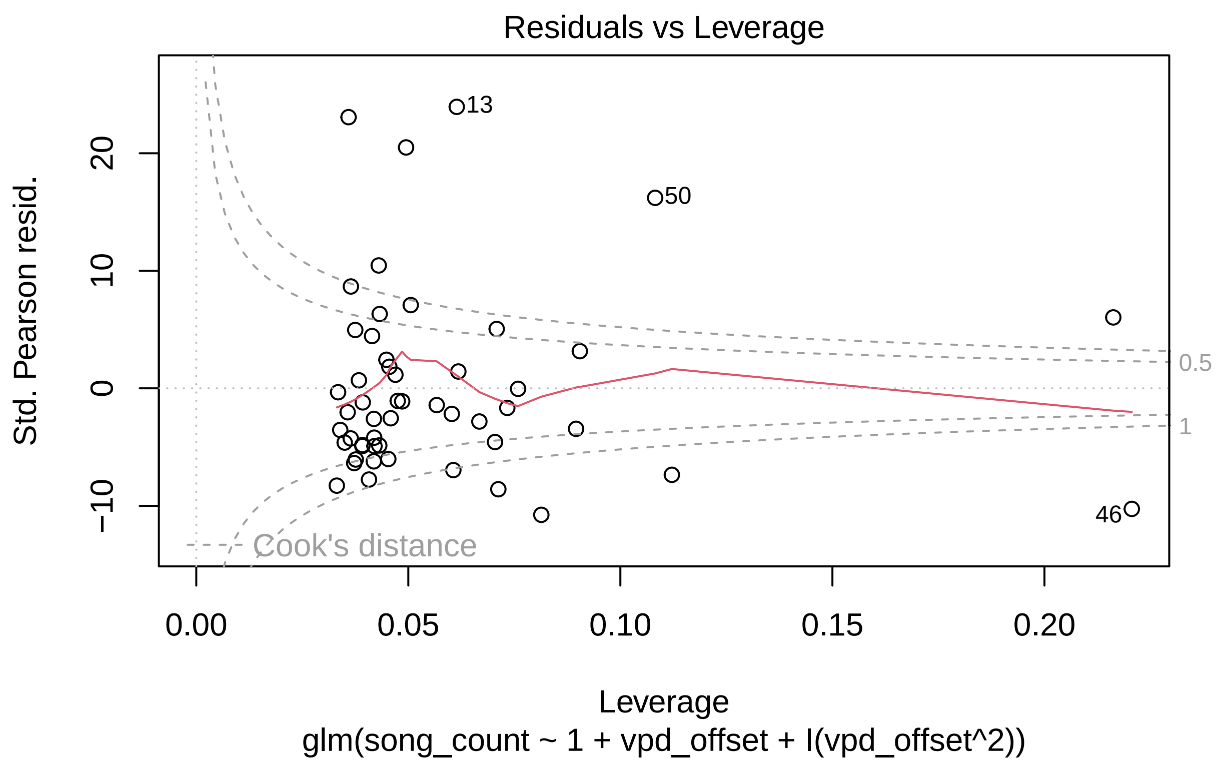




Normal Q−Q
Std. Pearson resid.
Theoretical Quantiles
glm(song_count ~ 1 + vpd_offset + I(vpd_offset^2))
16
13
49
−10
0
10
20
−2
−1
0
1
2



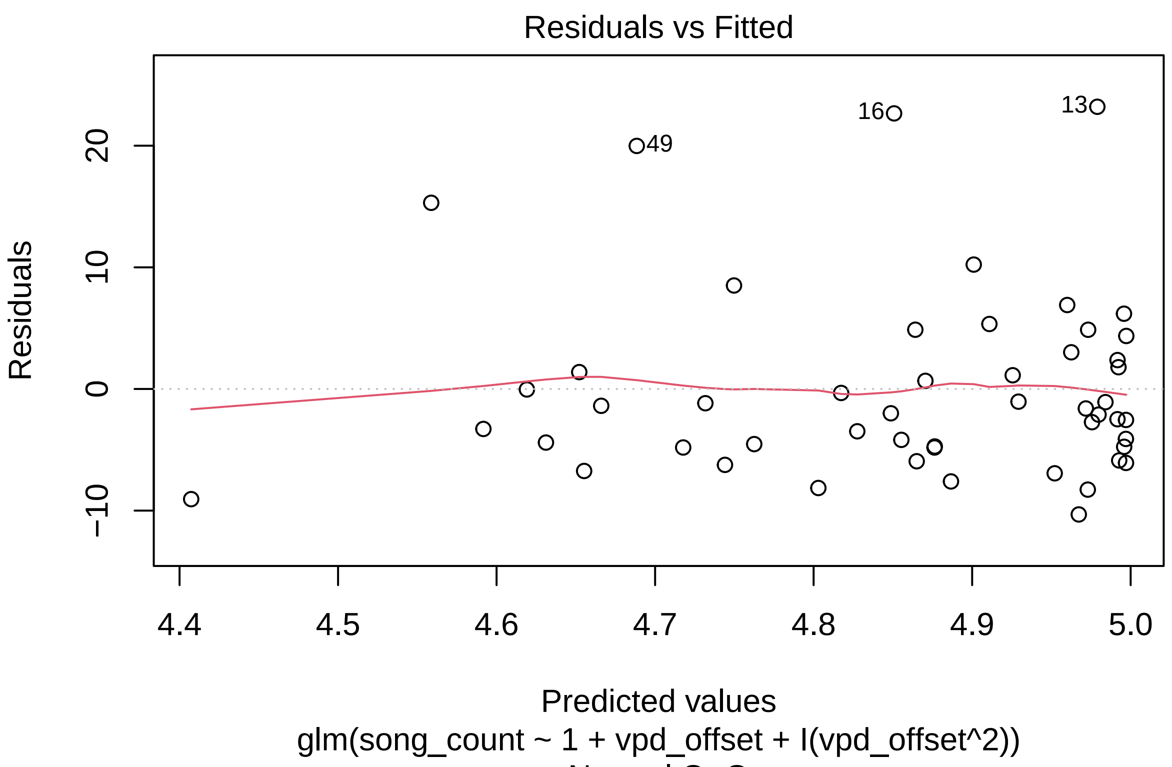
Scale−Location
√|Std. Pearson resid.|
Predicted values
glm(song_count ~ 1 + vpd_offset + I(vpd_offset^2))
16
13
49
0
1
2
3
4
5
4.4
4.5
4.6
4.7
4.8
4.9
5.0

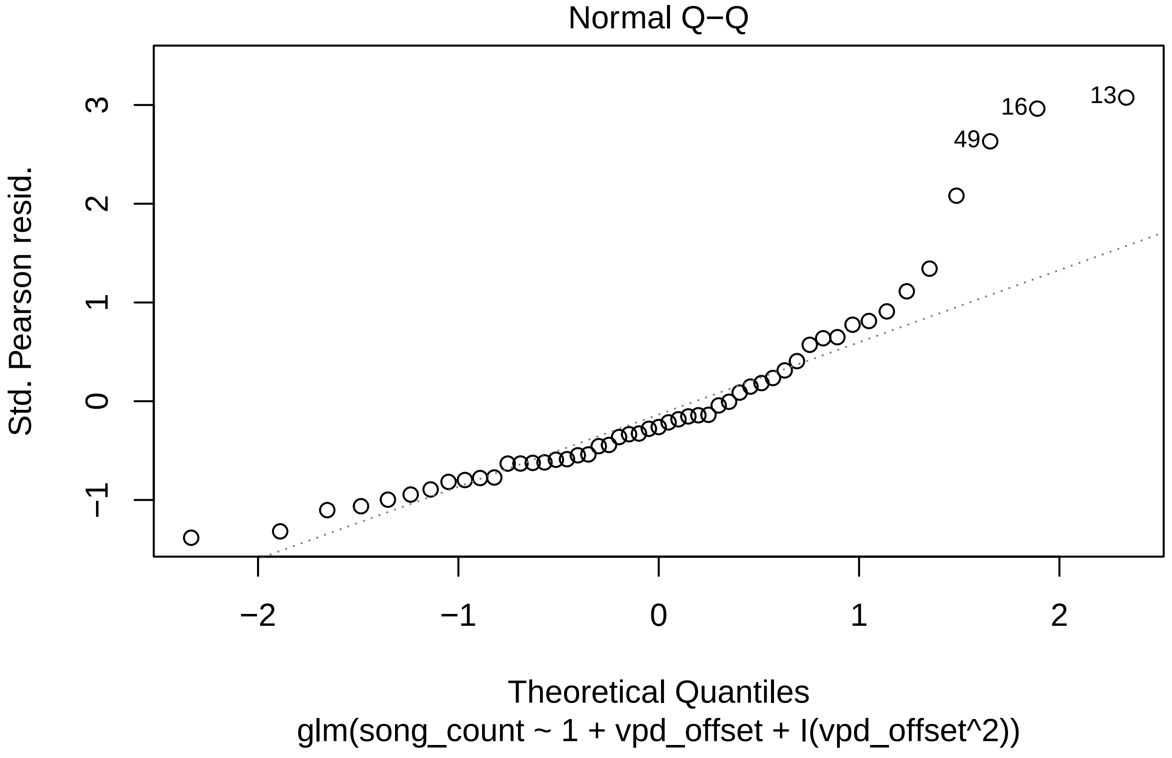


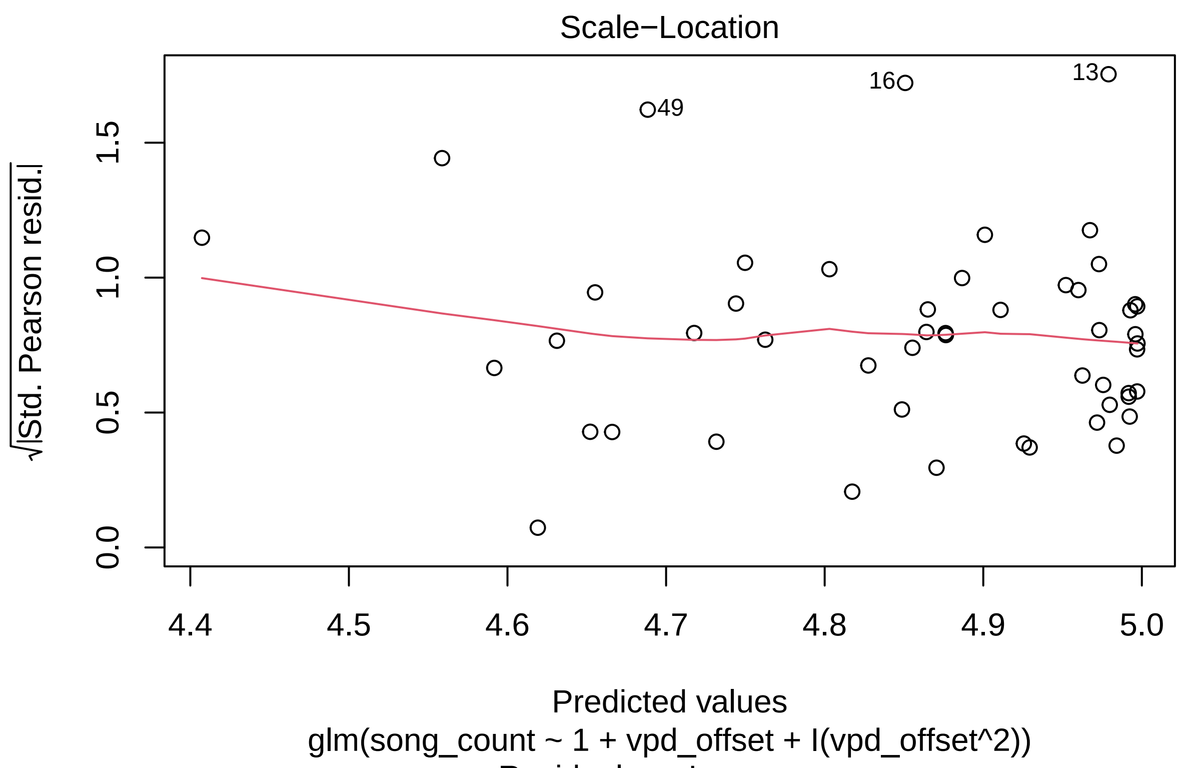
```
count_glm_qpoisson <- glm(song_count ~
            1 + vpd_offset + I(vpd_offset^2),
            data = data_ind %>% filter( !(male %in% c("T231", "T260"))),
        family = quasipoisson(link = "log")
)
summary(count_glm_qpoisson)
```

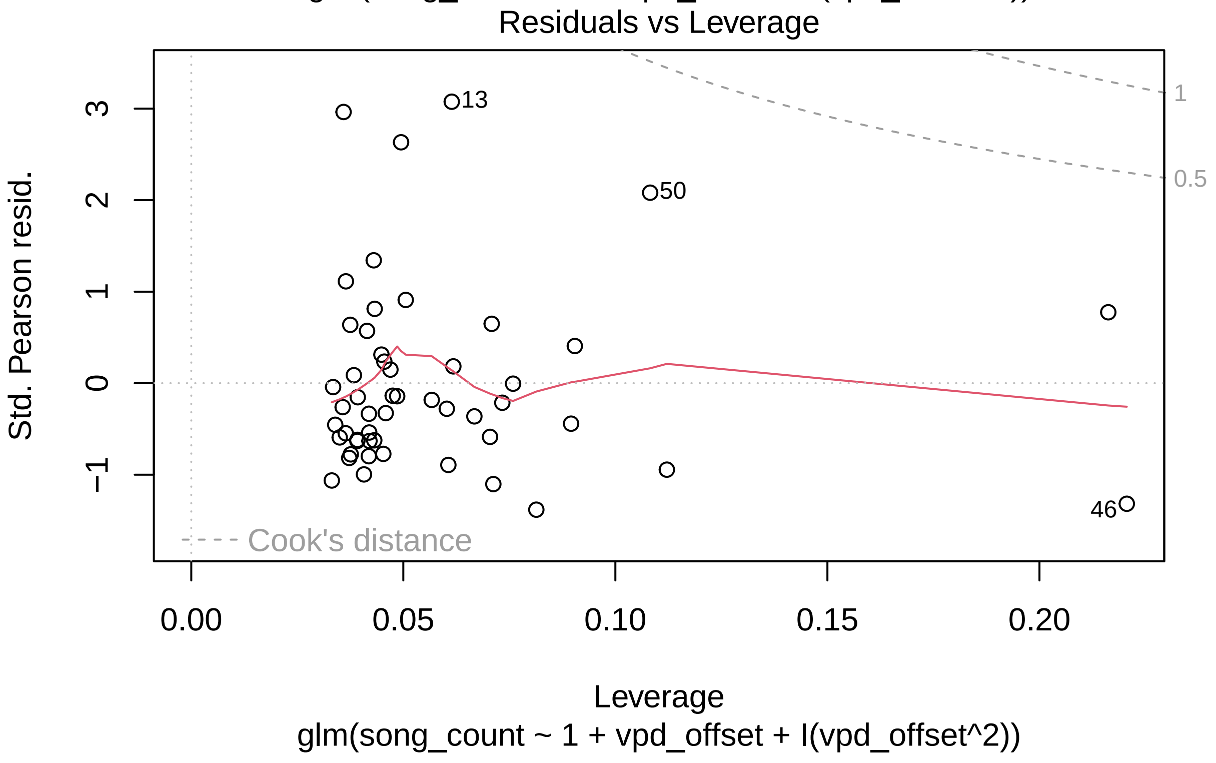
```
##
## Call:
## glm(formula = song_count ~ 1 + vpd_offset + I(vpd_offset^2),
##     family = quasipoisson(link = "log"), data = data_ind %>%
##         filter(!(male %in% c("T231", "T260"))))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -12.977   -5.180   -2.063    2.601   18.783
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.9382108  0.1454980  33.940   <2e-16 ***
## vpd_offset      -0.0087653  0.0064960  -1.349    0.184
## I(vpd_offset^2) -0.0003253  0.0004627  -0.703    0.485
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 60.6278)
##
##     Null deviance: 2713.2  on 50  degrees of freedom
## Residual deviance: 2591.2  on 48  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

```
plot(count_glm_qpoisson)
```









```
## Clearly the data is over dispersed
## quasipoisson() doesn't seem to exist for glmer (but likely exists in nlme)

## Try using negative binomial

count_glmer_nb <-
```

```
    glmer.nb(song_count ~ vpd_offset + I(vpd_offset^2) + (vpd_offset|male),
             data = data_ind %>% filter( !(male %in% c("T231", "T260"))),
             ## control values are used by the initial optimization
             ## using a poisson glmer model, which doesn't converge
             control = glmerControl(
                 optCtrl = list(maxiter = 1E5,
                                maxfun = 2E6,
                                verbose = TRUE,
                                trace = TRUE),
                 optimizer="bobyqa"),
             ## nb.control values are used by the second optimizer
             nb.control = list(
#                 trace = TRUE,
                 maxit = 100,
                 verbose = TRUE)
             )
```

```
## Warning in (function (npt = min(n + 2L, 2L * n), rhobeg = NA, rhoend = NA, :
## unused control arguments ignored

## Warning in (function (npt = min(n + 2L, 2L * n), rhobeg = NA, rhoend = NA, :
## unused control arguments ignored

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.0579335 (tol = 0.002, component 1)

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly uniden
##  - Rescale variables?;Model is nearly unidentifiable: large eigenvalue ratio
##  - Rescale variables?

## Warning in (function (npt = min(n + 2L, 2L * n), rhobeg = NA, rhoend = NA, :
## unused control arguments ignored

## Error in pwrssUpdate(pp, resp, tol = tolPwrss, GQmat = GHrule(0L), compDev = compDev, : pwrssUpdate
```

```
## Try and fail to use `start` rgument
tmp <- glmer(song_count ~
            vpd_offset + I(vpd_offset^2) + (vpd_offset + I(vpd_offset^2)||male),
        data = data_ind,
        family = poisson(link = "log"),
        start = list(theta = 4.8, fixef = -0.3515557), #coef,
control = glmerControl(optCtrl = list(maxiter = 1E4, maxfun = 2E6), optimizer="bobyqa"),
    verbose = TRUE)
```

```
## Error in getStart(start, rho$pp, "theta"): incorrect number of theta components (!=3)
```

Result

- Model doesn't converge.
- Output includes > Model is nearly unidentifiable: very large eigenvalue > - Rescale variables?;Model is nearly unidentifiable: large eigenvalue ratio > - Rescale variables? To me this suggests we should use `song_prop` and the glmer weights function.

Formal Model Fits to `song_prop`

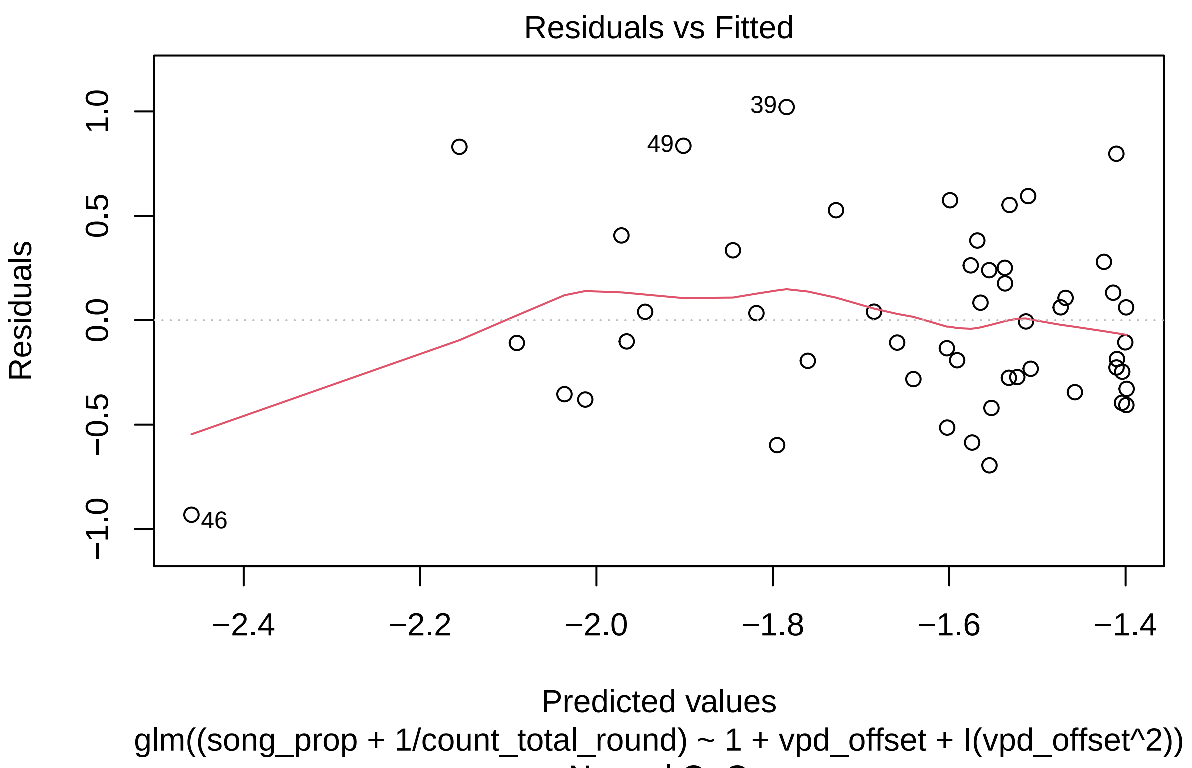
Using GLM

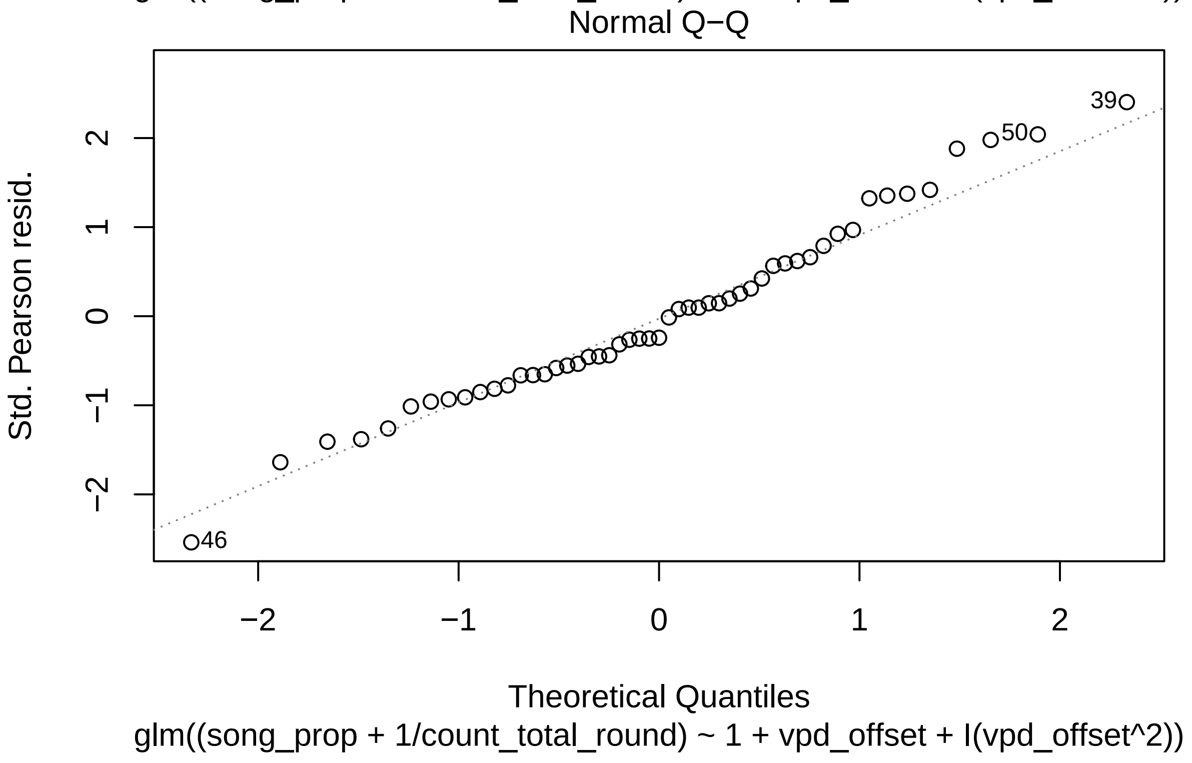
- Set `family` to "Gamma" or "Gaussian" and link = 'log'

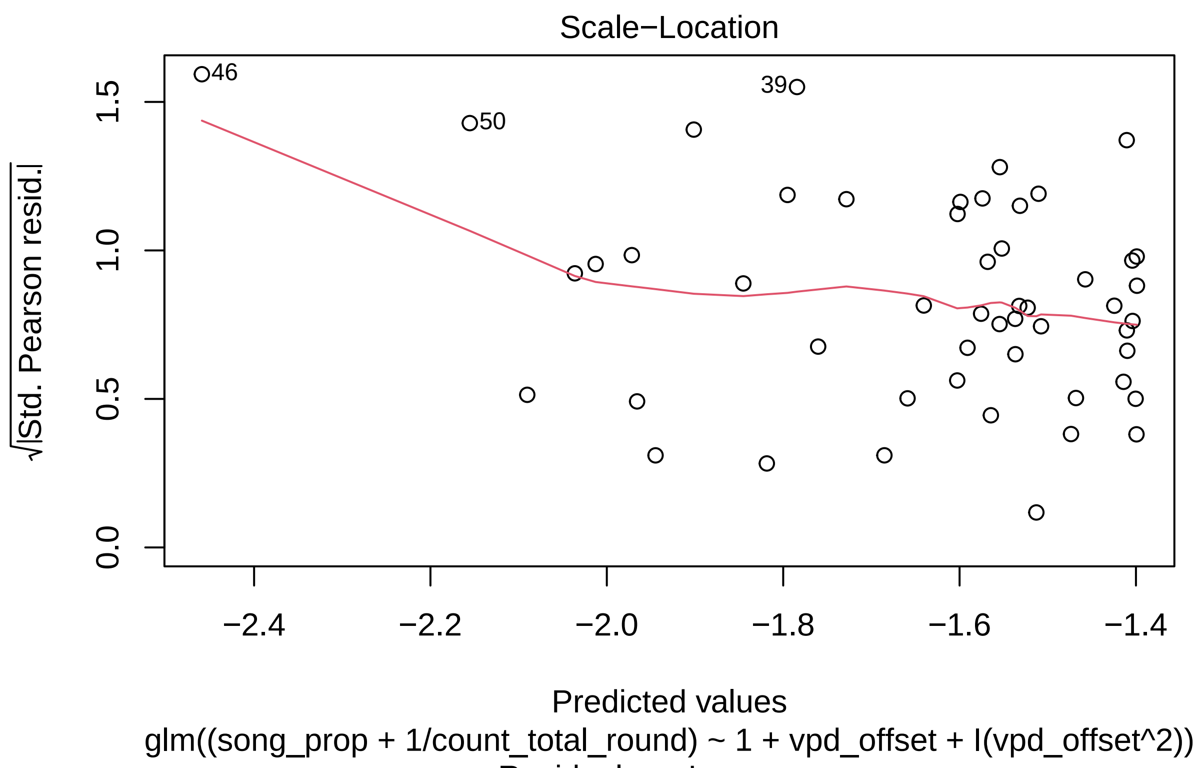
```
## Gamma has issue with 0 value, so add equivalent of 1 count to every observation (~ like a prior)
glm_gamma <- glm((song_prop + 1/count_total_round) ~
            1 + vpd_offset + I(vpd_offset^2),
            data = data_ind %>% filter( !(male %in% c("T231", "T260"))),,
        family = "Gamma"(link = 'log')
       )

summary(glm_gamma)
```

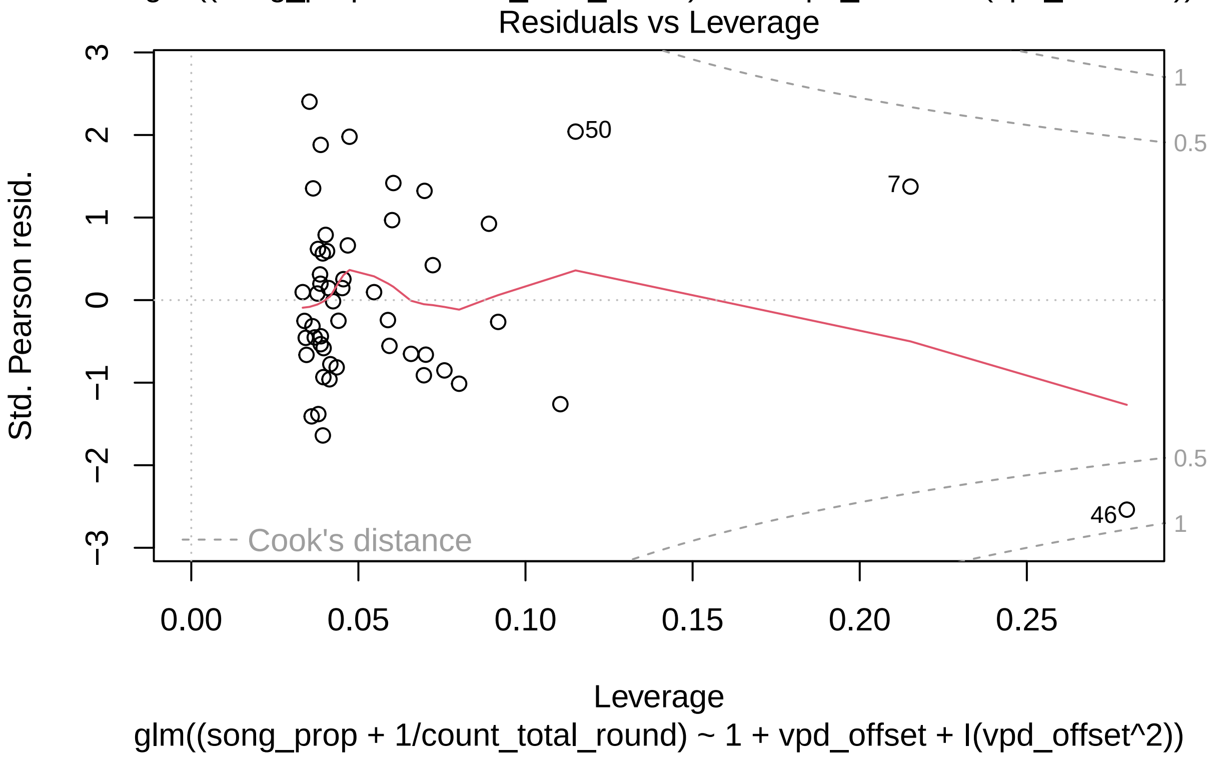
```
##
## Call:
## glm(formula = (song_prop + 1/count_total_round) ~ 1 + vpd_offset +
##     I(vpd_offset^2), family = Gamma(link = "log"), data = data_ind %>%
##     filter(!(male %in% c("T231", "T260"))))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8713  -0.3100  -0.1051   0.2376   0.7967
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -1.4545609  0.0929918 -15.642  < 2e-16 ***
## vpd_offset      -0.0128992  0.0038993  -3.308  0.00179 **
## I(vpd_offset^2) -0.0007423  0.0002805  -2.647  0.01096 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 0.1870542)
##
##     Null deviance: 13.874  on 50  degrees of freedom
## Residual deviance: 11.682  on 48  degrees of freedom
## AIC: -99.484
##
## Number of Fisher Scoring iterations: 6
```

```
plot(glm_gamma)
```


Residuals vs Fitted
Residuals
1.0
0.5
0.0
−0.5
−1.0
39
49
46
−2.4
−2.2
−2.0
−1.8
−1.6
−1.4
Predicted values
glm((song_prop + 1/count_total_round) ~ 1 + vpd_offset + I(vpd_offset^2))


Normal Q−Q
Std. Pearson resid.
2
1
0
−1
−2
39
50
46
−2
−1
0
1
2
Theoretical Quantiles
glm((song_prop + 1/count_total_round) ~ 1 + vpd_offset + I(vpd_offset^2))

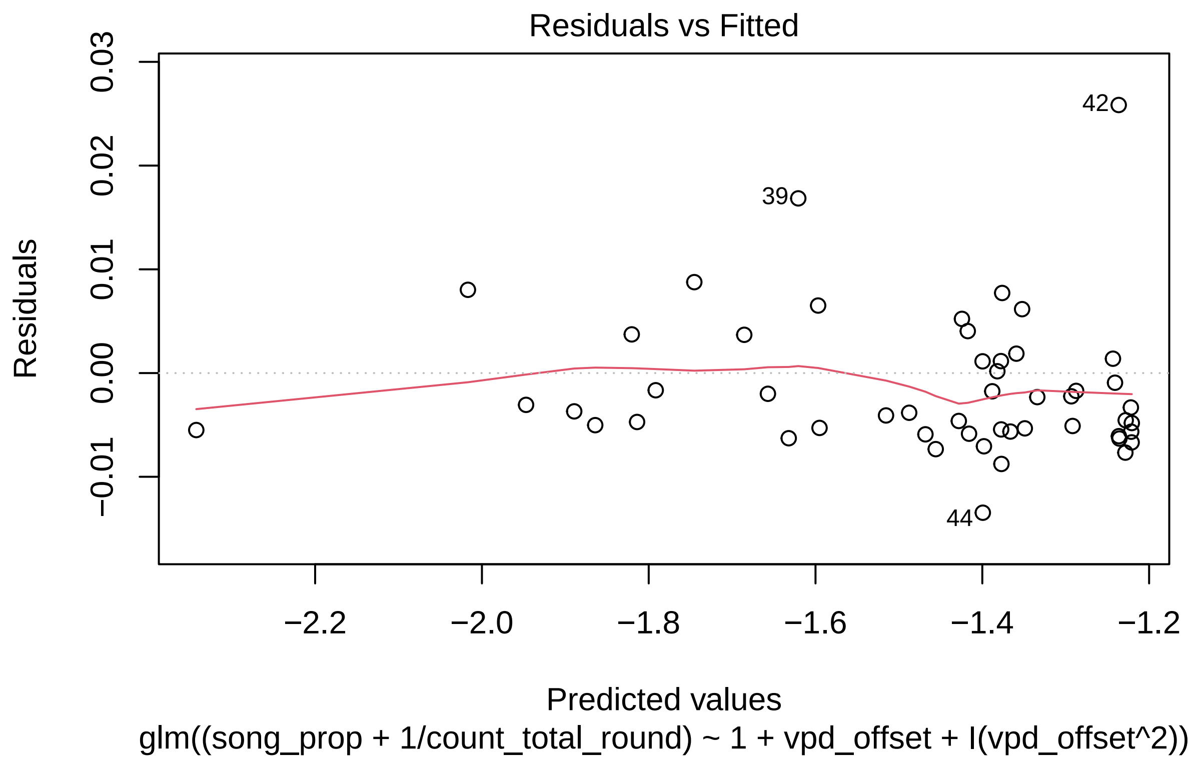


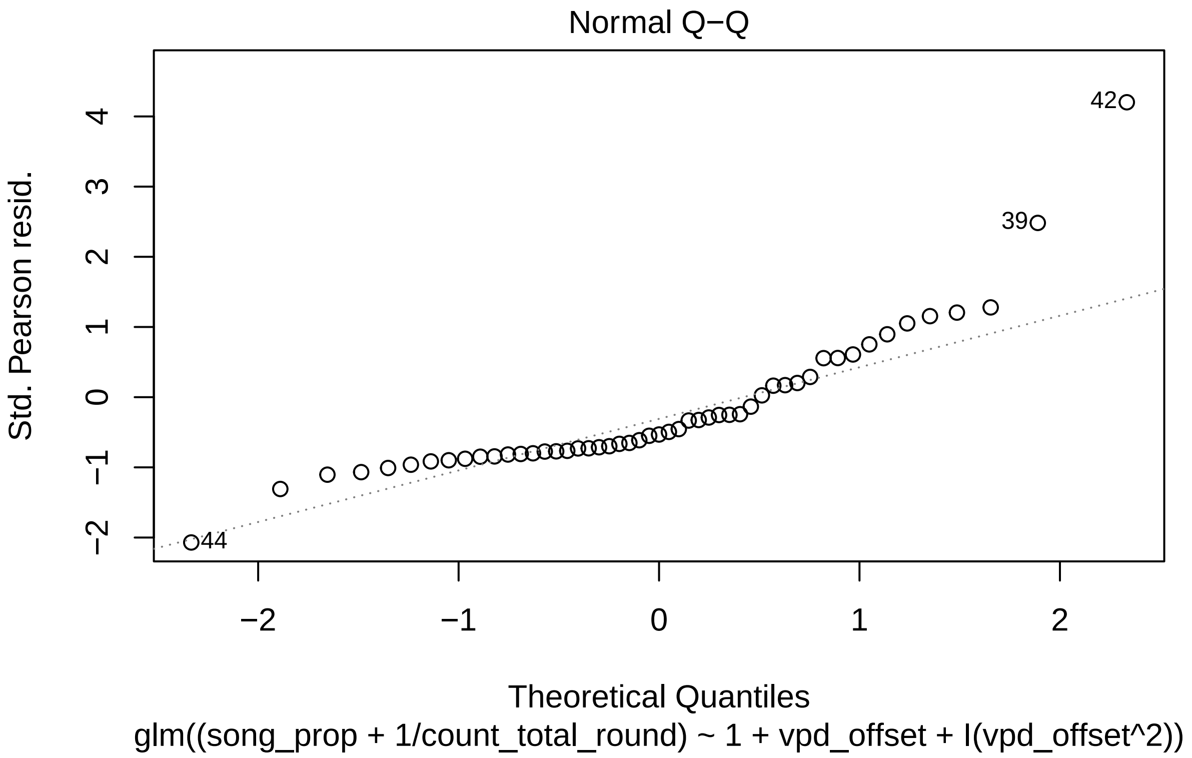


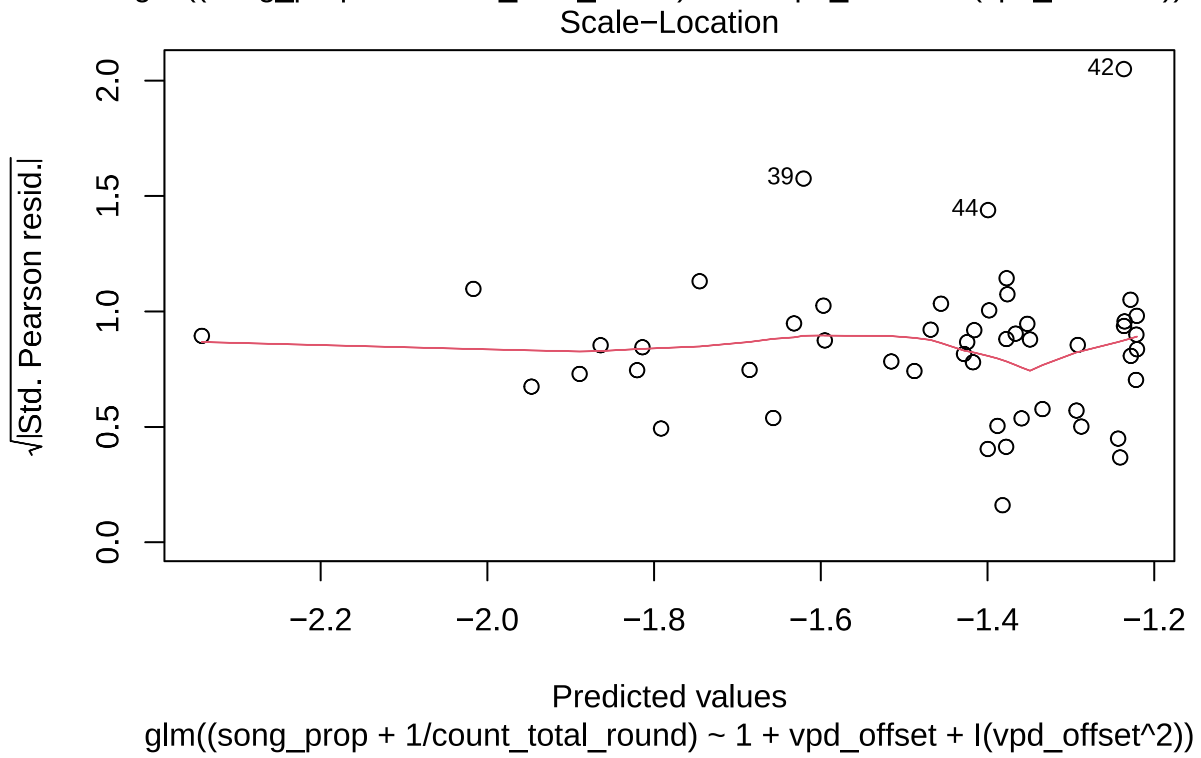
```
glm_gamma_weighted <- update(glm_gamma,  weights = 1/prop_wt)
summary(glm_gamma_weighted)
```

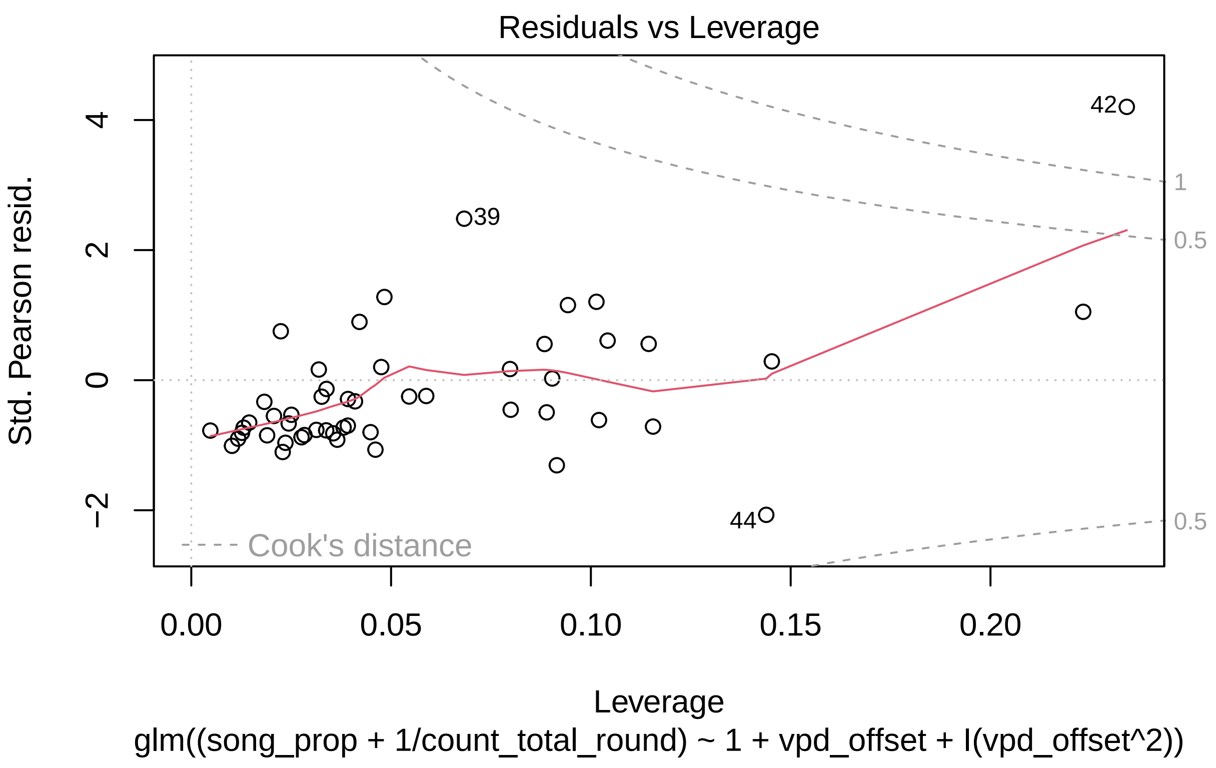
```
##
## Call:
## glm(formula = (song_prop + 1/count_total_round) ~ 1 + vpd_offset +
##     I(vpd_offset^2), family = Gamma(link = "log"), data = data_ind %>%
##     filter(!(male %in% c("T231", "T260"))), weights = 1/prop_wt)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.016756  -0.006781  -0.004198   0.001246   0.022417
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -1.2736693  0.0734354 -17.344  < 2e-16 ***
## vpd_offset      -0.0131931  0.0037598  -3.509 0.000988 ***
## I(vpd_offset^2) -0.0008080  0.0002733  -2.956 0.004816 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 4.937678e-05)
##
##     Null deviance: 0.0033205  on 50  degrees of freedom
## Residual deviance: 0.0026252  on 48  degrees of freedom
## AIC: 7.9547
##
## Number of Fisher Scoring iterations: 4
```

```
plot(glm_gamma_weighted)
```




Normal Q−Q
Std. Pearson resid.
Theoretical Quantiles
glm((song_prop + 1/count_total_round) ~ 1 + vpd_offset + I(vpd_offset^2))
42
39
44
4
3
2
1
0
−1
−2
−2
−1
0
1
2


Scale−Location
√|Std. Pearson resid.|
Predicted values
glm((song_prop + 1/count_total_round) ~ 1 + vpd_offset + I(vpd_offset^2))
42
39
44
2.0
1.5
1.0
0.5
0.0
−2.2
−2.0
−1.8
−1.6
−1.4
−1.2



```
anova(glm_gamma, glm_gamma_weighted)
```

```
## Analysis of Deviance Table
##
## Model 1: (song_prop + 1/count_total_round) ~ 1 + vpd_offset + I(vpd_offset^2)
## Model 2: (song_prop + 1/count_total_round) ~ 1 + vpd_offset + I(vpd_offset^2)
##   Resid. Df Resid. Dev Df Deviance
## 1        48    11.6823
## 2        48     0.0026  0    11.68
```

```
## Even filtered data doesn't behave well
glmer_gamma_filtered <- glmer(
    (song_prop + 1/count_total_round) ~
            vpd_offset + I(vpd_offset^2) + (vpd_offset||male),
        data = data_ind %>% filter( !(male %in% c("T231", "T260"))),
        family = Gamma(link = "log"),
        control = glmerControl(optCtrl = list(maxiter = 1E4, maxfun = 2E6), optimizer="bobyqa"),
        )
```

```
## Warning in (function (npt = min(n + 2L, 2L * n), rhobeg = NA, rhoend = NA, :
## unused control arguments ignored
```

```
## Warning in (function (npt = min(n + 2L, 2L * n), rhobeg = NA, rhoend = NA, :
## unused control arguments ignored
```

```
## boundary (singular) fit: see help('isSingular')
```

```r
## Wrong model since data is not discrete, gives 400+ pages of warnings
#tmp <- glmer(song_prop ~
#            vpd_offset + I(vpd_offset^2) + (vpd_offset + I(vpd_offset^2)||male),
#        data = data_ind,
#        family = poisson(link = "log"),
#        weights = prop_wt,
#        control = glmerControl(optCtrl = list(maxiter = 1E4, maxfun = 2E6), optimizer="bobyqa"),
#        verbose = TRUE)
#
#summary(tmp)
```

Result

Analyze with rTPC

Analyze with other packages

According to Padfield et al. (2021)

> However, [the rTPC] pipeline does not accommodate non-independent (related) replicates, and clustered or stratified sampling (possibly with missing values). In such situations, nonlinear mixed effects model fitting (e.g. using the nlme r package; Oddi et al., 2019) or Bayesian approaches (e.g. using the brms r package; Bürkner, 2017) would be more appropriate. Nevertheless, for fitting massive TPC datasets to multiple mathematical models, rTPC offers a simple, reliable and reproducible computational pipeline with robust methods for calculation of model uncertainty, requiring minimal statistical and computational expertise, and suitable for a wide range of applications.

Thus, we can't incorporate random effects.

`stan`

This is a good excuse to learn how to use `stan`

End

```r
knitr::knit_exit()
```