Supplementary information S3

Supplmentary information I

Johannes Rusch, David Strand, Tom Andersen

1 6 2021

Data preprocessing

All data from the mesocosm experiment is contained in a single Excel file which we read with the `read_xl` package. Need to set `na="N/A"` due to non-standard `NA` symbol. First line is blank and should be removed. We also convert the output from `read_excel` from `tibble` to `data.frame`.

```
library(readxl)

d <- data.frame(read_excel("PaperV_main_table_updated.xlsx",
                           sheet=1, na="N/A"))
d <- d[-1, ]
```

Column `Tank` is `NA` for all rows that are not part of the main experiment. These, and rows with `Tank = "Controll"` and `Food = "Garveyard"`are also removed, since these are data from ancillary experiments that are not analyzed here.

```
d <- subset(d, !is.na(Tank))
d <- subset(d, Tank != "Controll")
d <- subset(d, Food != "Graveyard")
```

This gives a data set with 72 rows, organized as 6 blocks of 12 observations, each with 4 treatment combinations repeated 3 times. Create new variable `Run` for identifying the 6 blocks.

```
d$Run <- rep(1:6, each=12)
```

Make a sample identifier `data.frame` `d1` containing only run, tank, and treatment levels, and shorten their names.

```
d1 <- d[, c("Run", "Tank", "Food", "Temp_hi.lo", "Density", "Replicate")]
names(d1) <- c("Run", "Tank", "Food", "Temp", "Dens", "Repl")
```

Make treatments into factor variables and relevel them.

```
d1$Run  <- factor(d1$Run)
d1$Tank <- factor(d1$Tank)
d1$Food <- factor(d1$Food == "F",  labels=c("no", "hi"))
d1$Temp <- factor(d1$Temp == "1",  labels=c("lo", "hi"))
d1$Dens <- factor(d1$Dens == "20", labels=c("lo", "hi"))
summary(d1)
```

```
##  Run     Tank    Food    Temp    Dens         Repl
##  1:12   B12:18   no:36   lo:36   lo:36   Min.   :1
##  2:12   B13:18   hi:36   hi:36   hi:36   1st Qu.:1
##  3:12   B5 :18                           Median :2
##  4:12   C13:18                           Mean   :2
##  5:12                                    3rd Qu.:3
##  6:12                                    Max.   :3
```

Need to get rid of names like `"20µLWell_Aph_A"` (R doesn't like variable names starting with a number, nor containing characters like µ)

The Excel sheet contains separate columns for each species ("Aph" and "Pac") and each filter half (A and B). We will convert each set of 4 columns from wide to long format with the `stack` command. We do this for positive droplets (`pos.drp`), total droplets (`tot.drp`), and copy number per well (`copy.well`):

```
pos.drp <- stack(d[, c("posDRP_Aph_A", "posDRP_Aph_B","posDRP_Pac_A", "posDRP_Pac_B")])
tot.drp <- stack(d[, c("totDRP_Aph_A", "totDRP_Aph_B","totDRP_Pac_A", "totDRP_Pac_B")])
copy.well <- stack(d[, c("X20µLWell_Aph_A", "X20µLWell_Aph_B","X20µLWell_Pac_A",
"X20µLWell_Pac_B")])
```

Each of these will have two columns (`ind` and `values`), where the former contains the original column names. Each of these objects have 4 * 72 = 288 rows. Since species and filters are in the same order in all three objects we need only to extract the `species` and `filter` identifiers from one of them. The identifiers are at character positions 8:10 and 12 in the `ind` column.

```
Species <- factor(substr(pos.drp$ind, 8, 10))
Filter <- factor(substr(pos.drp$ind, 12, 12))
```

We combine the 3 columns containing positive droplet, total droplets, and copy numbers into a `data.frame` `d2`and add the sample identifier `d1` into this. By the repetition rules for `data.frames` the sample identifier `d1` will be repeated 4 times such that all rows end up with 288 elements (R gives a warning that can be ignored).

```
d2 <- data.frame(d1, data.frame(Species, Filter,
                                pos.drp=pos.drp$values,
                                tot.drp=tot.drp$values,
                                copy.well=copy.well$values))
```

```
## Warning in data.frame(d1, data.frame(Species, Filter, pos.drp =
## pos.drp$values, : row names were found from a short variable and have been
## discarded
```

```
summary(d2)
```

```
##  Run     Tank    Food     Temp     Dens          Repl   Species   Filter
##  1:48   B12:72   no:144   lo:144   lo:144   Min.   :1   Aph:144   A:144
##  2:48   B13:72   hi:144   hi:144   hi:144   1st Qu.:1   Pac:144   B:144
##  3:48   B5 :72                             Median :2
##  4:48   C13:72                             Mean   :2
##  5:48                                      3rd Qu.:3
##  6:48                                      Max.   :3
##
##     pos.drp           tot.drp        copy.well
```

```
##  Min.   :    0.0   Min.   :    0   Min.   :       0
##  1st Qu.:    0.0   1st Qu.:15085   1st Qu.:       0
##  Median :   10.0   Median :17723   Median :      23
##  Mean   :  704.5   Mean   :15724   Mean   :  443032
##  3rd Qu.:  148.0   3rd Qu.:19022   3rd Qu.:     225
##  Max.   :19433.0   Max.   :21194   Max.   :20000000
##  NA's   :4         NA's   :4       NA's   :16
```

Missing values in 4 records for pos/tot droplets (all from Run 5, Tank C13, filter B, both species) and 16 records for copy number per well (all with missing or zero total droplets)

```
subset(d2, is.na(copy.well))[, c(1:2, 6:7, 10:11)]
```

```
##     Run Tank Repl Species tot.drp copy.well
## 65    6  B12    2     Aph       0        NA
## 66    6  B12    3     Aph       0        NA
## 67    6  B13    1     Aph       0        NA
## 131   5  C13    2     Aph      NA        NA
## 132   5  C13    3     Aph      NA        NA
## 137   6  B12    2     Aph       0        NA
## 138   6  B12    3     Aph       0        NA
## 139   6  B13    1     Aph       0        NA
## 209   6  B12    2     Pac       0        NA
## 210   6  B12    3     Pac       0        NA
## 211   6  B13    1     Pac       0        NA
## 275   5  C13    2     Pac      NA        NA
## 276   5  C13    3     Pac      NA        NA
## 281   6  B12    2     Pac       0        NA
## 282   6  B12    3     Pac       0        NA
## 283   6  B13    1     Pac       0        NA
```

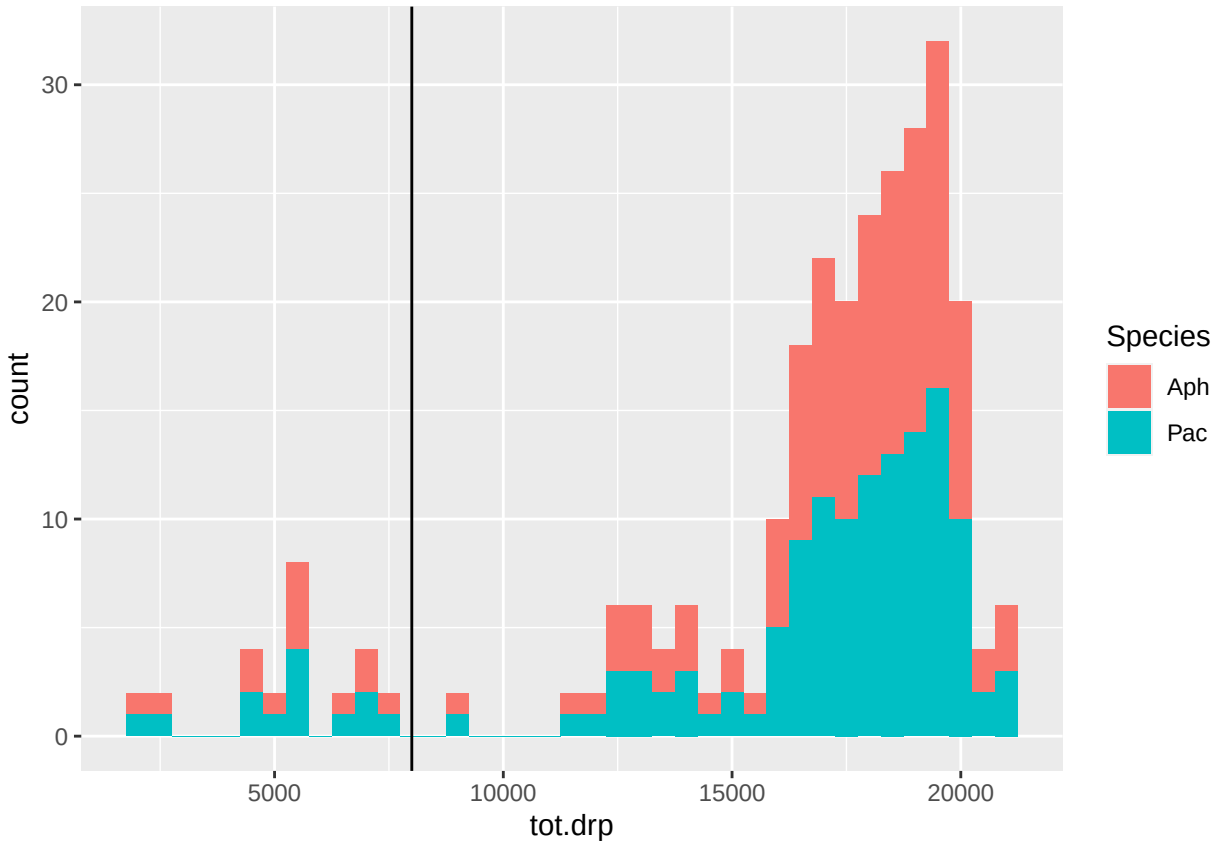
We decide to remove all rows containing `NA`s, which leaves us with 272 rows

```
d2 <- subset(d2, complete.cases(d2))
```

Plot histogram of total droplets with the 8000 droplet limit indicated (26 records out of 272)

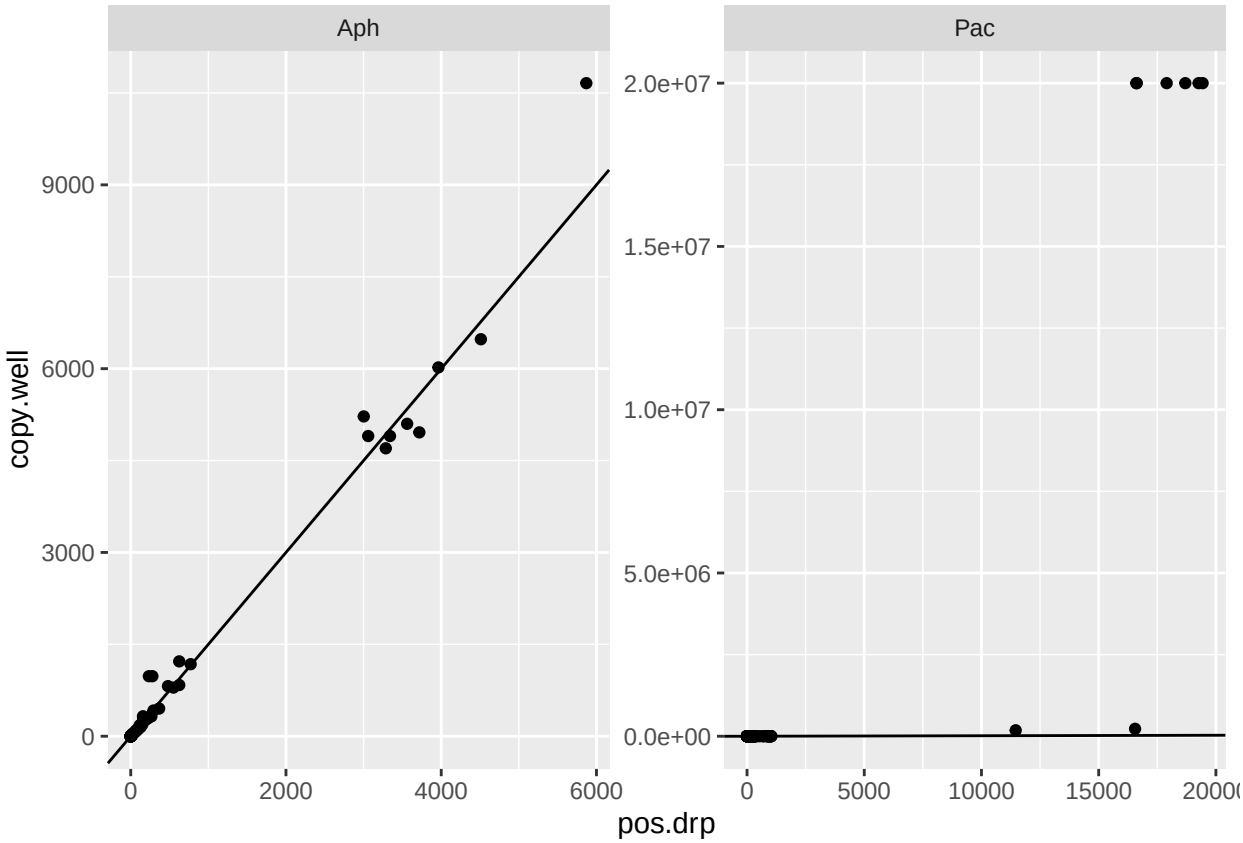
```
library(ggplot2)

ggplot(d2, aes(x=tot.drp, group=Species, fill=Species))  +
  geom_histogram(binwidth=500) +
  geom_vline(xintercept=8000)
```



Copy numbers per well is basically 1.5 times positive droplets, except for the 6 records (all *Pacifastacus*) where positive droplets is close to total droplets (i.e., saturation) and the copy number is right-censored at 20 million.

```
ggplot(d2, aes(x=pos.drp, y=copy.well, )) +
  geom_point() +
  geom_abline(intercept=0, slope=1.5) +
  facet_wrap(~Species, scales="free")
```



In the data file the A and B filter halves are kept separate, but we will pool the counts before proceeding. Due to the right censoring copy number estimates of saturates samples, we focus on positive and total droplets. For the time being, we also keep the samples with total droplet count <8000

```
d3 <- aggregate(d2[, c("pos.drp", "tot.drp")], by=d2[, c("Run", "Tank", "Food", "Temp",
"Dens", "Repl", "Species" )],  FUN= sum)
```

This gives a final data set of 138 observations, 69 for *Aphanomyces* and 69 for *Pacifastacus*.

Modelling strategy

We will fit generalized linear mixed models (GLMMs) for each species, using `pos.drp` as dependent variable. Since this variable is a count, we will fit models from the poisson and negative binomial families (both using the default log link between the fitted linear predictor and the expectation of the dependent variable). We will also use `log(tot.drp)` as an offset in all models. An `offset` term in a linear model will have its model coefficient locked to being 1, which means that we actually fit `log(pos.drp / tot.drp)`, or the logarithm of the fraction of positive droplets, but by doing it this way we can still take advantage of that `pos.drp` is a non-negative integer count.

Since we expect the data to have an issue with excess zeros beyond what can be modeled by the poisson distribution (so-called over-dispersion), we also fit models based on the negative binomial distribution since this distribution can represent over-dispersed situations better than the poisson distribution. We also fit models with zero-inflated versions of these distributions, which can represent zero-inflated situations at the expense one additional model parameter which represents the probability that an observation is zero. We compare different candidate models with the Akaike Information Criterion (AIC), and choose the one with the lowest AIC.

The data set has a hierarchical structure in the sense that there are different sets of treatment combinations in the 4 tanks (mesocosms) in each of the 6 runs, and that 3 replicate samples were taken from each tank/run combination. These samples will be pseudoreplicated in the sense that they will be less independent than samples from different tank/run combinations. In all models we represent this hierarchical structure as a random effect of `Tank` nested within `Run` (`+ (1 | Run / Tank)`). All GLMM models are fitted with the `glmmTMB` package.

Aphanomyces models

We start out by making a subset of only the *Aphanomyces* observations which we will use in all models

```
d.aph <- subset(d3, Species == "Aph")
```

We investigate 10 models: the first 4 based on the poisson distribution and the next 4 based on the negative binomial distribution. Each set of 4 models include versions with additive (`~ Food + Temp + Dens`; models 1, 3, 5, 7) and interactive (`~ Food * Temp * Dens`; models 2, 4, 6, 8) fixed effects of the treatments. Models 3, 4, 7, 8 also include a zero-inflation term (`ziformula = ~1`) which is assumed independent o treatments. Finally there are models 9 and 10, which are intermediates between model 5 and 6 by having only interaction terms between `Food` and `Dens` (9) or between `Temp` and `Dens`.

```
library(glmmTMB)

aph.01 <- glmmTMB(pos.drp ~ Food + Temp + Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.aph, family=poisson(link="log"))
aph.02 <- glmmTMB(pos.drp ~ Food * Temp * Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.aph, family=poisson(link="log"))
aph.03 <- glmmTMB(pos.drp ~ Food + Temp + Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.aph, family=poisson(link="log"), ziformula = ~1)
aph.04 <- glmmTMB(pos.drp ~ Food * Temp * Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.aph, family=poisson(link="log"), ziformula = ~1)
aph.05 <- glmmTMB(pos.drp ~ Food + Temp + Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.aph, family=nbinom1(link="log"))
aph.06 <- glmmTMB(pos.drp ~ Food * Temp * Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.aph, family=nbinom1(link="log"))
aph.07 <- glmmTMB(pos.drp ~ Food + Temp + Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.aph, family=nbinom1(link="log"), ziformula = ~1)
```

```
## Warning in fitTMB(TMBStruc): Model convergence problem; non-positive-definite
## Hessian matrix. See vignette('troubleshooting')
```

```
aph.08 <- glmmTMB(pos.drp ~ Food * Temp * Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.aph, family=nbinom1(link="log"), ziformula = ~1)
aph.09 <- glmmTMB(pos.drp ~ Food * Temp + Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.aph, family=nbinom1(link="log"))
aph.10 <- glmmTMB(pos.drp ~ Food + Temp * Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.aph, family=nbinom1(link="log"))
```

The NB model without interactions but with zero-inflation (`aph.07`) had convergence problems - maybe because the random effect of Run alone becomes very small. All the other converged with no problems.

```
AIC(aph.01, aph.02, aph.03, aph.04, aph.05, aph.06, aph.07, aph.09, aph.10)
```

```
##        df       AIC
## aph.01  6 1310.8998
## aph.02 10 1307.5190
## aph.03  7 1233.5312
## aph.04 11 1231.0687
## aph.05  7  561.9310
## aph.06 11  556.0795
## aph.07  8        NA
## aph.09  8  563.9265
## aph.10  8  550.8192
```

The AIC ranking was generally in favor of the negative binomial models (5:10) which had substantially lower AIC than the poisson-based ones (1:4). The zero-inflation adjustment had a stronger effect on the poisson models, while it appears that the negative binomial distribution by itself covers the over-dispersion sufficiently well such there is little advantage of the zero-inflation adjustment for this distribution. Among the negative binomial models there is a small advantage of the interactive model compared to the additive, but this contrast becomes even stronger when the fixed effect interactions are simplified to only include temperature and density (i.e., model `aph.10`)

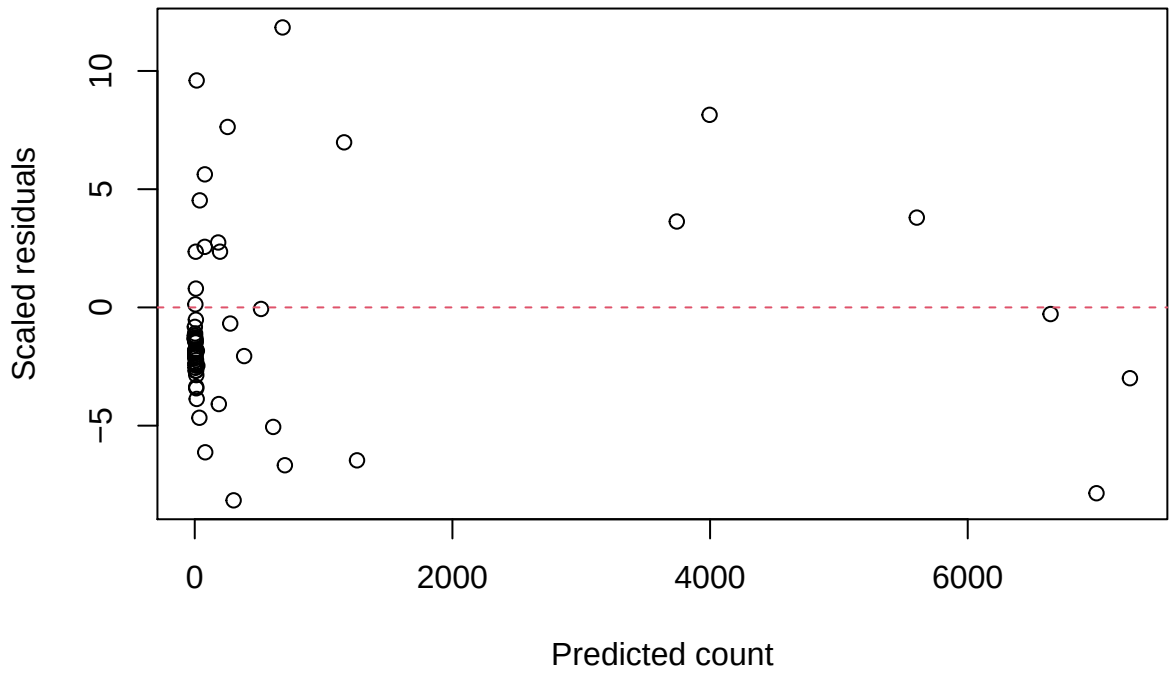
```
summary(aph.10)
```

```
##  Family: nbinom1  ( log )
## Formula:
## pos.drp ~ Food + Temp * Dens + offset(log(tot.drp)) + (1 | Run/Tank)
## Data: d.aph
##
##      AIC      BIC   logLik deviance df.resid
##    550.8    568.7   -267.4    534.8       61
##
## Random effects:
##
## Conditional model:
##  Groups   Name        Variance Std.Dev.
##  Tank:Run (Intercept) 0.9642   0.9819
##  Run      (Intercept) 0.1607   0.4009
## Number of obs: 69, groups:  Tank:Run, 24; Run, 6
##
## Overdispersion parameter for nbinom1 family ():   32
##
## Conditional model:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -7.05191    0.57392 -12.287  < 2e-16 ***
## Foodhi         0.01047    0.48101   0.022   0.9826
## Temphi        -1.96556    0.77878  -2.524   0.0116 *
## Denshi         3.77849    0.62023   6.092 1.11e-09 ***
## Temphi:Denshi -4.01625    1.00688  -3.989 6.64e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Model `aph.10` had 6-fold higher unexplained variance between tanks in the same run than between runs, which indicates that there were systematic differences between runs were small. The model basically says that

positive droplet count decreased with temperature and increased with density, but only when temperature was low. There appear to not be any pattern in scaled model residuals when plotted against fitted values.

```
plot(exp(predict(aph.10)), residuals(aph.10) / sqrt(exp(predict(aph.10)))), xlab="Predicted
count", ylab="" abline(h=0, lty=2, col=2) Scaled residuals")
```



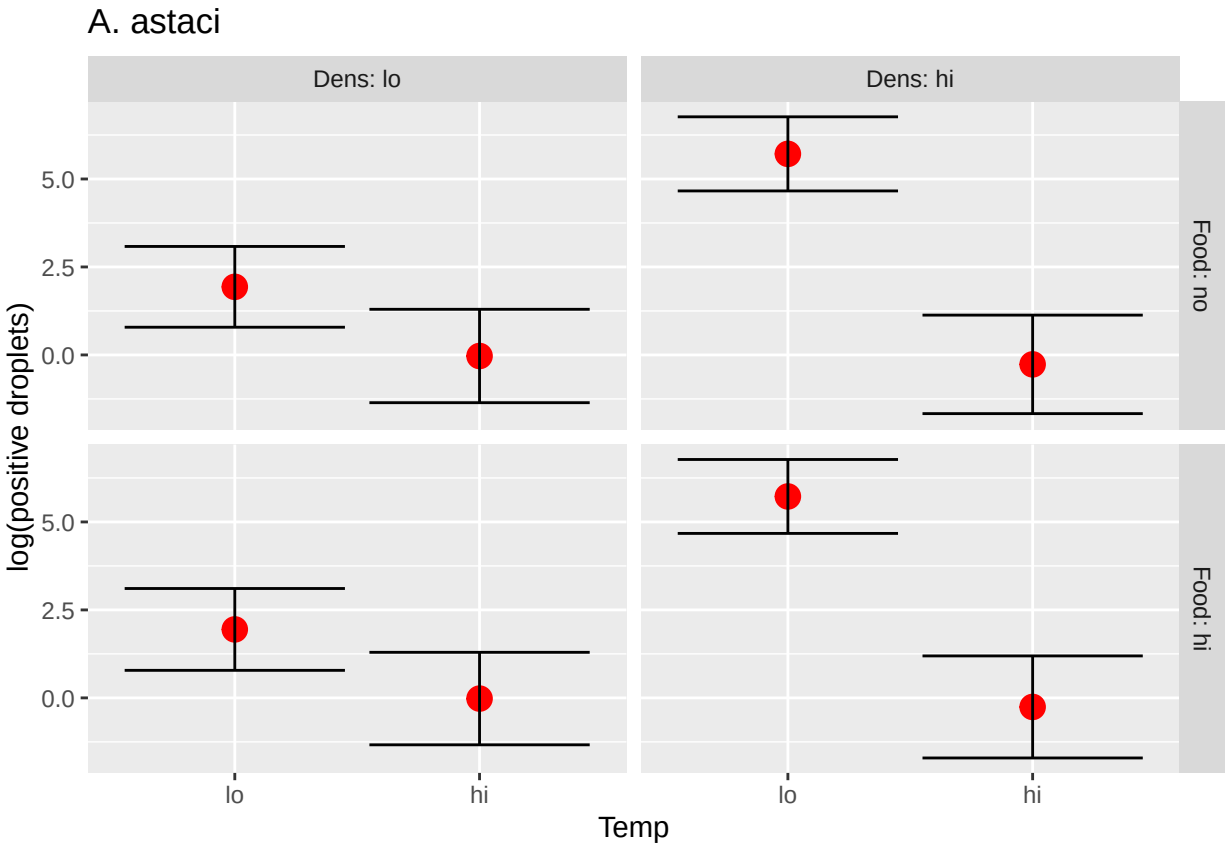
We make graphical representation of model `aph.10` to get a better overview of the fixed effect predictions. We first make a new data frame `aph.pred` containing all possible combinations of the 3 treatment factors (`Food, Temp, Dens`). Since we are only interested in the fixed effects, we leave the random effects (`Run, Tank`) at their nominal first levels. We also set total droplets to a fixed value (8000) for this prediction.

```
aph.pred <- data.frame(expand.grid(Food=levels(d.aph$Food), Temp=levels(d.aph$Temp),
Dens=levels(d.aph$Dens)), tot.drp= 8000, Run=levels(d.aph$ Run)[ 1], Tank=levels(d.aph$
Tank)[1])
```

We then generate predictions for model `aph.10` for this data set. We set `re.form=NA` to get only fixed effect predictions, and set `se.fit=TRUE` so that we can construct approximate confidence limits for the predictions.

```
log.aph <- predict(aph.10, newdata=aph.pred, re.form=NA, se.fit=TRUE)
aph.pred <- cbind(aph.pred, log.aph)
```

```
ggplot(aph.pred, aes(x=Temp, y=fit)) +
  geom_point(size=4, color="red") +
  geom_errorbar(aes(ymin=(fit-2*se.fit), ymax=(fit+2*se.fit))) +
  facet_grid(Food ~ Dens, labeller=label_both) +
  labs(y="log(positive droplets)") +
  ggtitle('A. astaci')
```



The approximate 95% confidence limits for positive droplet counts are generally overlapping for all treatment factor combinations, except when temperature is low (10C) and density is high (20 individuals per tank). The presence or absence of food seems to have only negligible effects on the detection of *Aphanomyces*.

Pacifastacus models

We start out by making a subset of only the *Pacifastacus* observations which we will use in all models

```
d.pac <- subset(d3, Species == "Pac")
```

We investigate the same 10 candidate models as for *Aphanomyces*:

```
pac.01 <- glmmTMB(pos.drp ~ Food + Temp + Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.pac, family=poisson(link="log"))
pac.02 <- glmmTMB(pos.drp ~ Food * Temp * Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.pac, family=poisson(link="log"))
pac.03 <- glmmTMB(pos.drp ~ Food + Temp + Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.pac, family=poisson(link="log"), ziformula = ~1)
pac.04 <- glmmTMB(pos.drp ~ Food * Temp * Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.pac, family=poisson(link="log"), ziformula = ~1)
pac.05 <- glmmTMB(pos.drp ~ Food + Temp + Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.pac, family=nbinom1(link="log"))
pac.06 <- glmmTMB(pos.drp ~ Food * Temp * Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.pac, family=nbinom1(link="log"))
pac.07 <- glmmTMB(pos.drp ~ Food + Temp + Dens + offset(log(tot.drp)) + (1 | Run / Tank),
```

```
                  data=d.pac, family=nbinom1(link="log"), ziformula = ~1)
pac.08 <- glmmTMB(pos.drp ~ Food * Temp * Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.pac, family=nbinom1(link="log"), ziformula = ~1)
pac.09 <- glmmTMB(pos.drp ~ Food * Temp + Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.pac, family=nbinom1(link="log"))
pac.10 <- glmmTMB(pos.drp ~ Food + Temp * Dens + offset(log(tot.drp)) + (1 | Run / Tank),
                  data=d.pac, family=nbinom1(link="log"))
```

All models converged without any error or warnings

```
AIC(pac.01, pac.02, pac.03, pac.04, pac.05, pac.06, pac.07, pac.08, pac.09, pac.10)
```

```
##        df       AIC
## pac.01  6 1325.0925
## pac.02 10 1320.2159
## pac.03  7 1327.0925
## pac.04 11 1322.2159
## pac.05  7  800.2030
## pac.06 11  793.8492
## pac.07  8  802.2030
## pac.08 12  795.8492
## pac.09  8  801.9270
## pac.10  8  800.7599
```

As with the *Aphanomyces* models, the negative binomial models for *Pacifastacus* also performed substantially better than the poisson ones. Also here, the zero-inflated versions did not any rank better than the ones without zero-inflation adjustment. The AIC ranking for *Pacifastacus* was in favor of the model with full three-way interactions, based on the negative binomial distribution without zero-inflation adjustment (model `pac.06`).

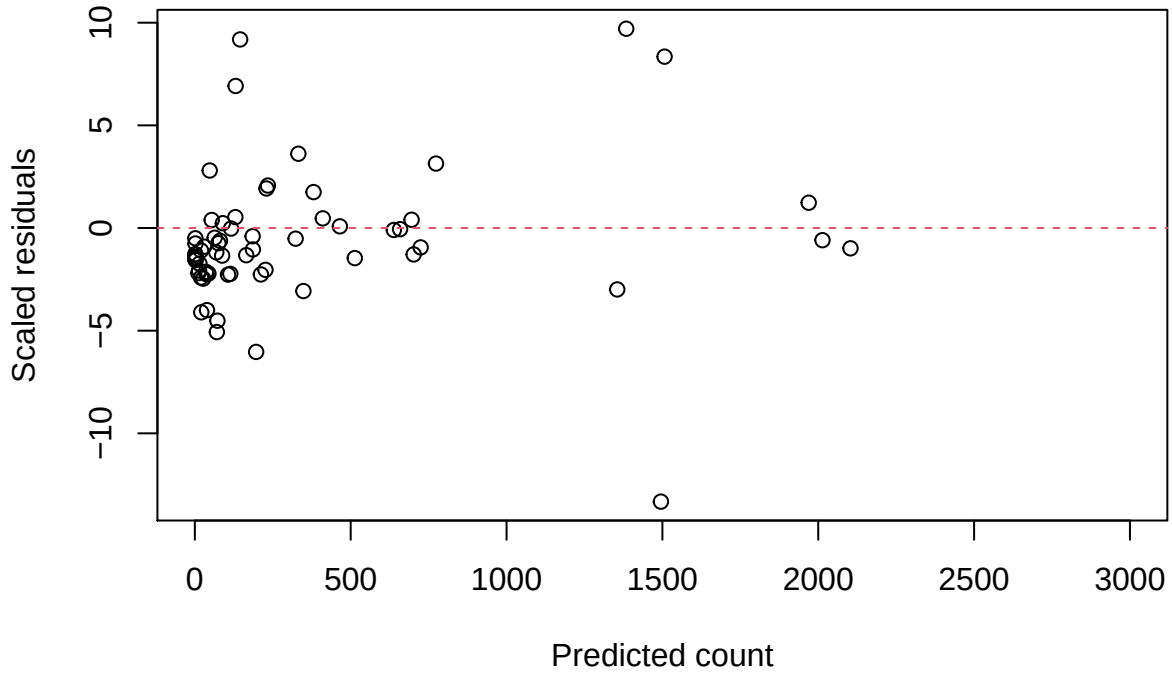
```
summary(pac.06)
```

```
##  Family: nbinom1  ( log )
## Formula:
## pos.drp ~ Food * Temp * Dens + offset(log(tot.drp)) + (1 | Run/Tank)
## Data: d.pac
##
##      AIC      BIC   logLik deviance df.resid
##    793.8    818.4   -385.9    771.8       58
##
## Random effects:
##
## Conditional model:
##  Groups   Name        Variance Std.Dev.
##  Tank:Run (Intercept) 0.9642   0.9819
##  Run      (Intercept) 2.6413   1.6252
## Number of obs: 69, groups:  Tank:Run, 24; Run, 6
##
## Overdispersion parameter for nbinom1 family (): 17.6
##
## Conditional model:
##                      Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)          -5.70455    1.10807  -5.148 2.63e-07 ***
## Foodhi                0.11484    0.83133   0.138 0.890131
## Temphi                0.02276    1.58406   0.014 0.988535
## Denshi                4.63937    0.81820   5.670 1.43e-08 ***
## Foodhi:Temphi        -1.86133    1.22909  -1.514 0.129927
## Foodhi:Denshi        -5.05231    1.16708  -4.329 1.50e-05 ***
## Temphi:Denshi        -3.98672    1.18891  -3.353 0.000799 ***
## Foodhi:Temphi:Denshi  4.85374    1.71612   2.828 0.004679 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In contrast to the *Aphanomyces* model, the chosen *Pacifastacus* model (`pac.06`) had more random effect variance attributed to between run than between tanks within runs. Due to the full three-way interaction structure it is harder to interpret the fixed effects of treatments directly from the model coefficients, but general message is that positive droplet count decreased with temperature and increased with density, but also played a role in interaction with both density and temperature (this will be clearer in the fixed effect visualization below). There appear to not be any pattern in scaled model residuals when plotted against fitted values.

```
plot(exp(predict(pac.06)), residuals(pac.06) / sqrt(exp(predict(pac.06))),
xlab="Predicted count", ylab="Scaled residuals", xlim=c(0, 3000))
abline(h=0, lty=2, col=2)
```



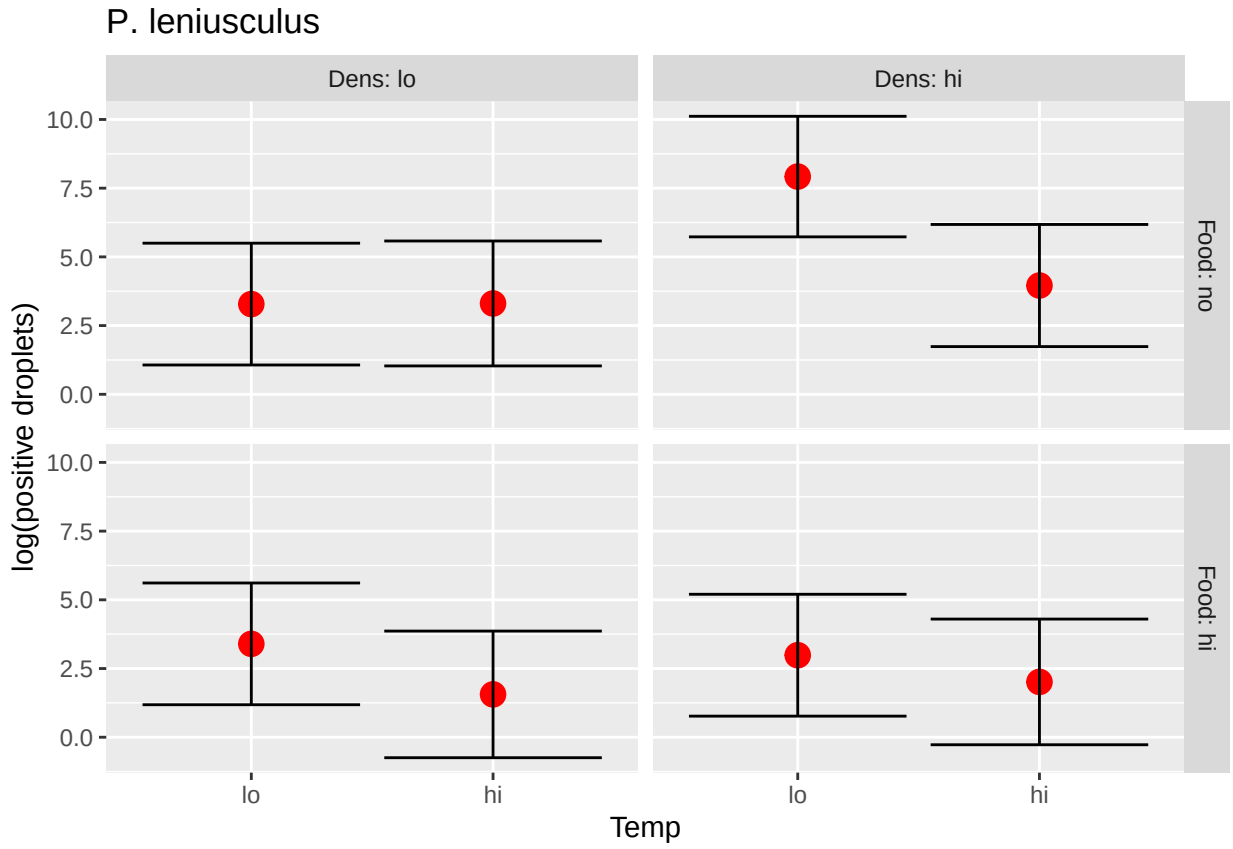
As with the *Aphanomyces* model, we first make a new data frame `pac.pred` containing all possible combinations of the 3 treatment factors (`Food, Temp, Dens`). Since we are only interested in the fixed effects, we leave the random effects (`Run, Tank`) at their nominal first levels. We also set total droplets to a fixed value (8000) for this prediction.

```
pac.pred <- data.frame(expand.grid(Food=levels(d.pac$Food), Temp=levels(d.pac$Temp),
Dens=levels(d.pac$Dens)), tot.drp=8000, Run=levels(d.pac$Run)[1], Tank=levels(d.pac$Tank)[1])
```

We then generate predictions for model `pac.06` for this data set. We set `re.form=NA` to get only fixed effect predictions, and set `se.fit=TRUE` so that we can construct approximate confidence limits for the predictions. Finally, we merge `pac.pred` with the model predictions and their standard errors into a single data frame.

```
log.pac <- predict(pac.06, newdata=pac.pred, re.form=NA, se.fit=TRUE)
pac.pred <- cbind(pac.pred, log.pac)
```

```
ggplot(pac.pred, aes(x=Temp, y=fit)) +
  geom_point(size=4, color="red") +
  ggtitle('P. leniusculus')+
  geom_errorbar(aes(ymin=(fit-2*se.fit), ymax=(fit+2*se.fit))) +
  facet_grid(Food ~ Dens, labeller=label_both) +
  labs(y="log(positive droplets)")
```



The 95% confidence limits for the fixed effects of treatments in the *Pacifastacus* model are generally overlapping, but least so between the low temperature, high density, no food treatment and all other treatment combinations.