Model Selection

Michelle Evans

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(ggplot2)
library(ggthemes)
library(viridis)
```

```
## Loading required package: viridisLite
```

```
library(nlme)
```

```
##
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
library(lme4)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following object is masked from 'package:tidyr':
##
##     expand

##
## Attaching package: 'lme4'

## The following object is masked from 'package:nlme':
##
##     lmList
```

```
library(MuMIn)
# install.packages("R2admb")
# install.packages("glmmADMB",
#     repos=c("http://glmmadmb.r-forge.r-project.org/repos",
#             getOption("repos")),
#     type="source")
library(glmmADMB)
```

```
## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

##
## Attaching package: 'glmmADMB'

## The following object is masked from 'package:MASS':
##
##     stepAIC

## The following object is masked from 'package:stats':
##
##     step
```

```
library(pscl)
```

```
## Warning: package 'pscl' was built under R version 3.4.2

## Classes and Methods for R developed in the
## Political Science Computational Laboratory
## Department of Political Science
## Stanford University
## Simon Jackman
## hurdle and zeroinfl functions by Achim Zeileis
```

```
library(cowplot)
```

```
##
## Attaching package: 'cowplot'

## The following object is masked from 'package:ggplot2':
##
##     ggsave
```

```
emergenceData <- readRDS("data/emergence.RData")
survival <- readRDS("data/survival.RData")
fecundity <- readRDS("data/fecundity.RData")
AedesAll <- readRDS("data/AedesIndividualSurvival.RData")
```

This document goes through model selection for each analysis section of the ms.

Survival

Aedes

Survival (or prob. of emergence) is a binary variable per individual mosquito. I use a generalized linear mixed model to explore the effect of Aedes density, Stephensi density, and temperature on emergence, with replicate as a random intercept. The data is subset to that of females (this can differ as males tend to develop more quickly and will emerge first). Predictor variables are normalized/scaled.

Reformat data into successes and failures per jar.

```r
aeSurvival <- emergenceData %>%
  filter(Species == "Aedes" & AeDens!= 0 & Sex == "Female") %>%
  group_by(Replicate, Temp, TempNum, AeDens, StDens, Ratio) %>%
  summarise(success = sum(Number, na.rm = T)) %>%
  mutate(failure = (AeDens/2) - success) %>%
  ungroup() %>%
  #rescale predictor variables
  mutate(TempScale = as.vector(scale(TempNum, center = F, scale = T))) %>%
  mutate(AeDensScale = as.vector(scale(AeDens, center = F, scale = T))) %>%
  mutate(StDensScale = as.vector(scale(StDens, center = F, scale = T))) %>%
  #drop jar that was all male
  filter(!(Replicate == "A" & Temp == "24" & Ratio == "8:24"))

#for when more than 50% were female
aeSurvival$failure[aeSurvival$failure<0] <- 0
```

Aedes model selection

```r
m0 <- glmer(cbind(success, failure) ~ 1 + (1|Replicate),
                   data = aeSurvival,
                   family = binomial(link = "logit"))
m1 <- glmer(cbind(success, failure) ~ AeDensScale + (1|Replicate),
                   data = aeSurvival,
                   family = binomial(link = "logit"))
m2 <- glmer(cbind(success, failure) ~ AeDensScale + TempNum + (1|Replicate),
                   data = aeSurvival,
                   family = binomial(link = "logit"))
m3 <- glmer(cbind(success, failure) ~ AeDensScale + poly(TempNum,2) + (1|Replicate),
                   data = aeSurvival,
                   family = binomial(link = "logit"))
m4 <- glmer(cbind(success, failure) ~ AeDensScale*poly(TempNum,2) + (1|Replicate),
                   data = aeSurvival,
                   family = binomial(link = "logit"))

m5 <- glmer(cbind(success, failure) ~ AeDensScale + poly(TempNum,2) + StDensScale + (1|Replicate),
                   data = aeSurvival,
                   family = binomial(link = "logit"))

m6 <- glmer(cbind(success, failure) ~ AeDensScale + poly(TempNum,2) * StDensScale + (1|Replicate),
                   data = aeSurvival,
                   family = binomial(link = "logit"))

m7 <- glmer(cbind(success, failure) ~ AeDensScale + poly(TempNum,2) + StDensScale + TempNum:StDensScale
                   data = aeSurvival,
                   family = binomial(link = "logit"))

m8 <- glmer(cbind(success, failure) ~ AeDensScale*poly(TempNum,2) + StDensScale*poly(TempNum,2) + (1|Re
                   data = aeSurvival,
                   family = binomial(link = "logit"))
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control
## $checkConv, : Model failed to converge with max|grad| = 0.00227615 (tol =
## 0.001, component 1)
```

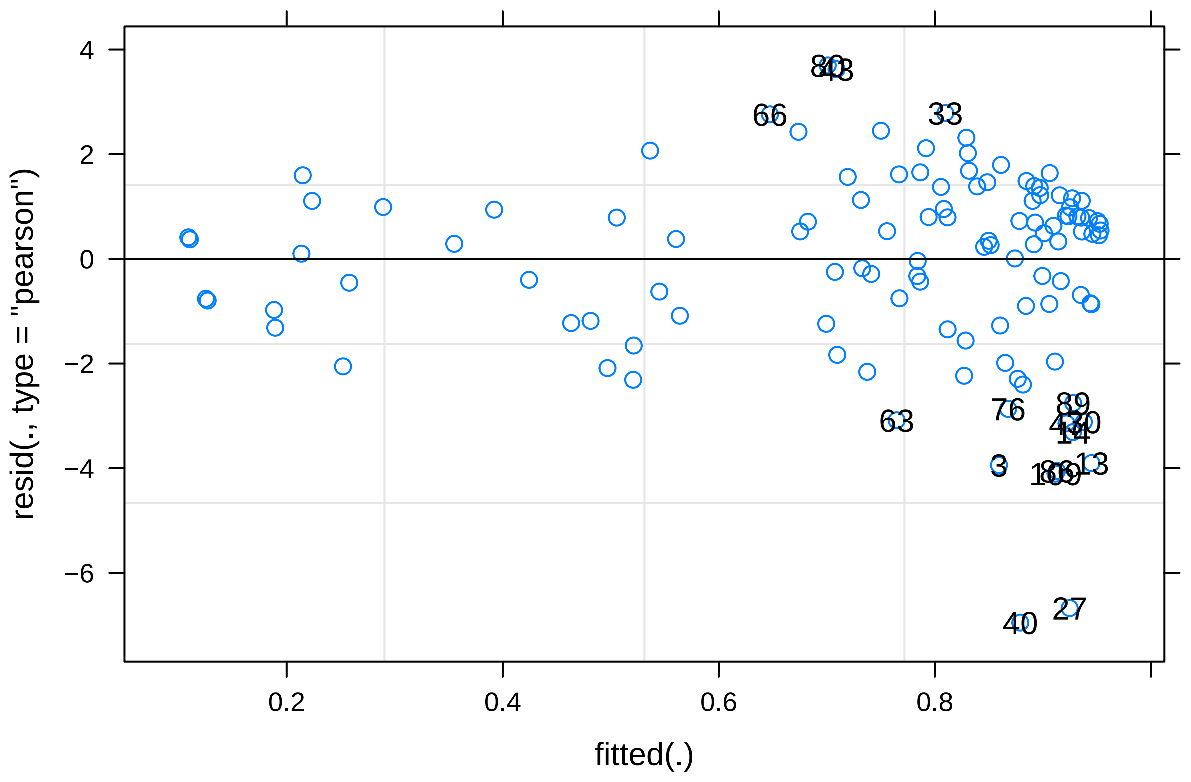
```
modelSummary <- cbind(model = c("m0","m1","m2","m3", "m4", "m5", "m6", "m7" , "m8"),
                   do.call(rbind, lapply(list(m0, m1, m2, m3, m4, m5, m6, m7, m8), broom::glance)),
                   AICc = AICc(m0, m1, m2, m3, m4, m5, m6, m7, m8),
                   AICweights = Weights(AIC(m0, m1, m2, m3, m4, m5, m6, m7, m8)))
modelSummary
```

```
##    model sigma    logLik       AIC       BIC  deviance df.residual AICc.df
## m0    m0     1 -815.5837 1635.1674 1640.6917 1365.5858         115       2
## m1    m1     1 -362.9521  731.9043  740.1908  459.1218         114       3
## m2    m2     1 -358.0774  724.1549  735.2036  449.5037         113       4
## m3    m3     1 -330.0400  670.0799  683.8908  393.0043         112       5
## m4    m4     1 -329.2971  672.5942  691.9294  391.5311         110       7
## m5    m5     1 -326.8907  665.7815  682.3545  386.5432         111       6
## m6    m6     1 -323.0674  662.1347  684.2321  378.7112         109       8
## m7    m7     1 -323.0756  660.1512  679.4864  378.7333         110       7
## m8    m8     1 -322.9522  665.9045  693.5262  378.4954         107      10
##    AICc.AICc    AICweights
## m0 1635.2726 1.267549e-212
## m1  732.1167  1.754096e-16
## m2  724.5120  8.449045e-15
## m3  670.6205  4.667015e-03
## m4  673.6217  1.327635e-03
## m5  666.5451  4.003427e-02
## m6  663.4681  2.479221e-01
## m7  661.1787  6.684025e-01
## m8  667.9800  3.764646e-02
```

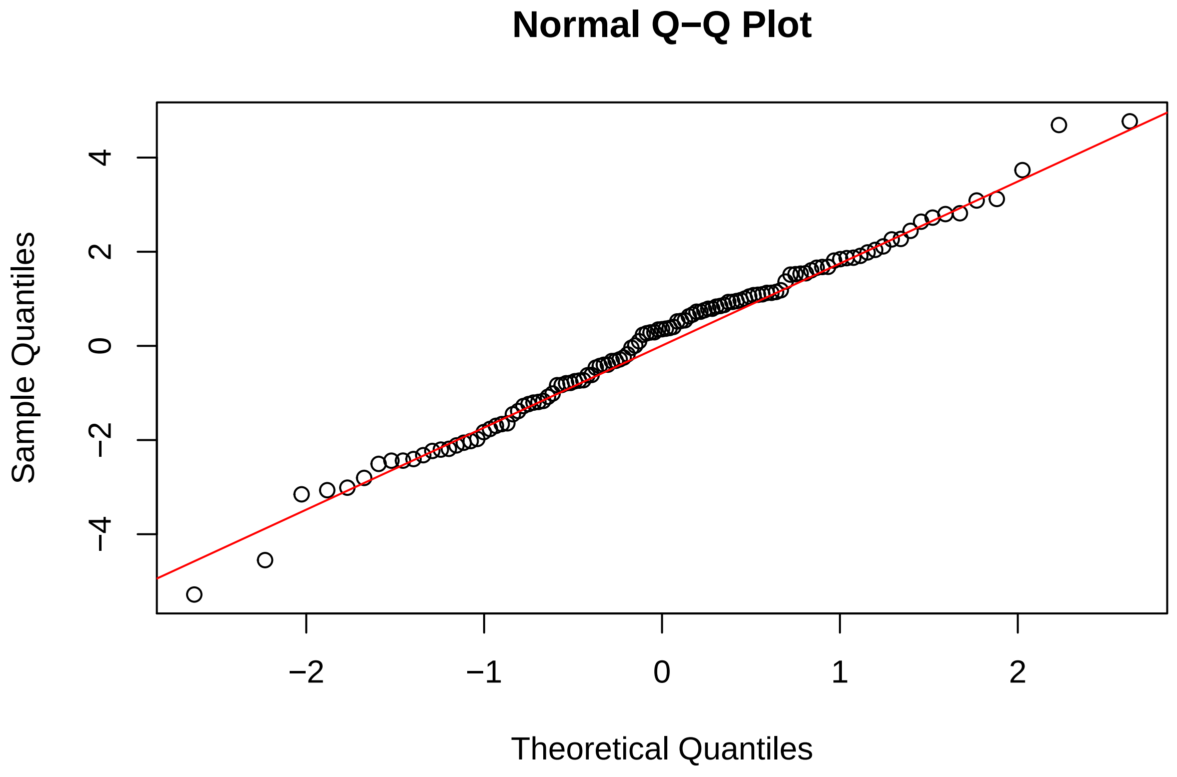
```
AedesSurvivalModel <- m7
```

Based on the above, model 7 (survival ~ AeDens + StDens + Temp^2 + StDens:Temp) fits best. Model 6 fits similary, but has an additional term of an interaction between polynomial temperature and stephensi density, and so the more parismonious model is chosen. Now we explore model residuals.

```
plot(AedesSurvivalModel, id = 0.01, idLabels=~.obs)
```



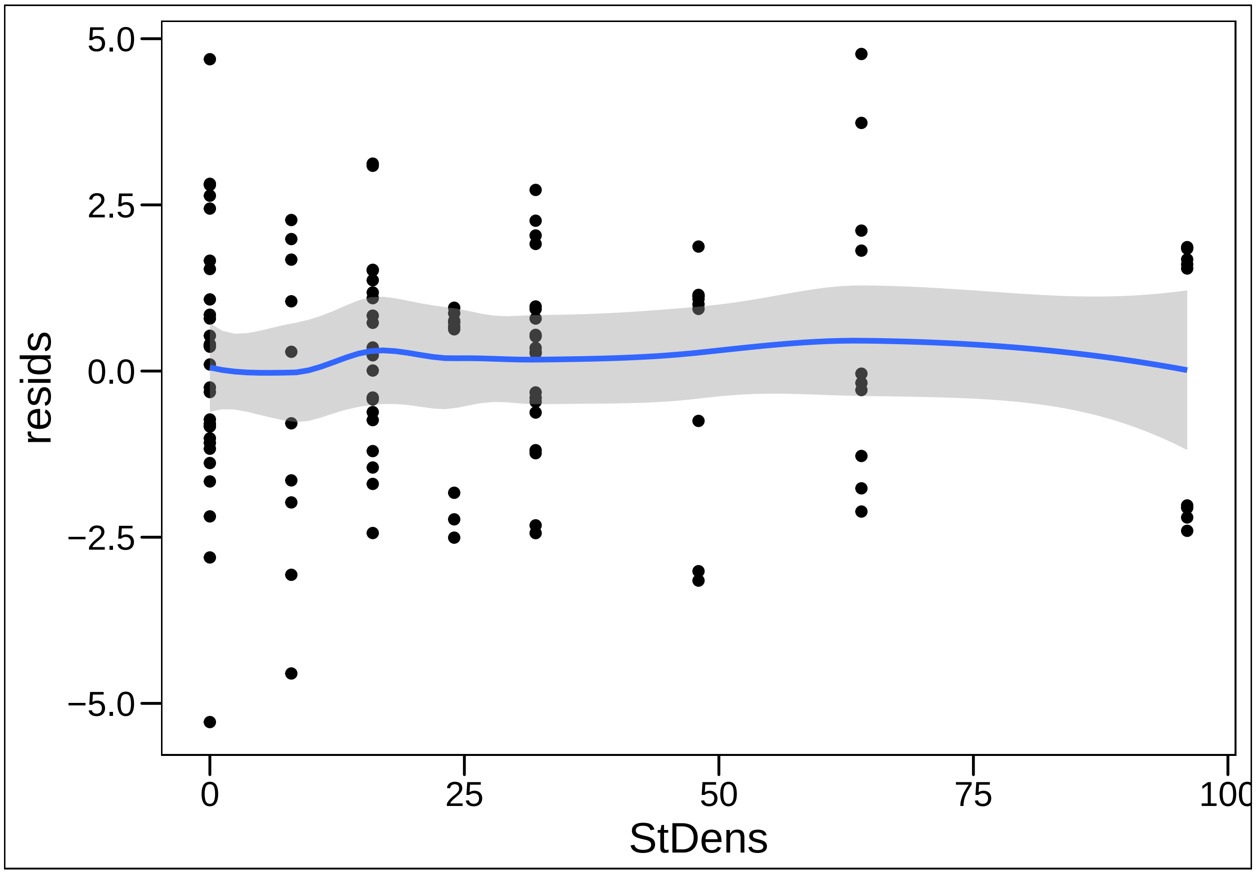
```
qqnorm(resid(AedesSurvivalModel))
qqline(resid(AedesSurvivalModel), col = "red")
```



```
aeSurvival$preds <- predict(AedesSurvivalModel)
aeSurvival$resids <- resid(AedesSurvivalModel)
```

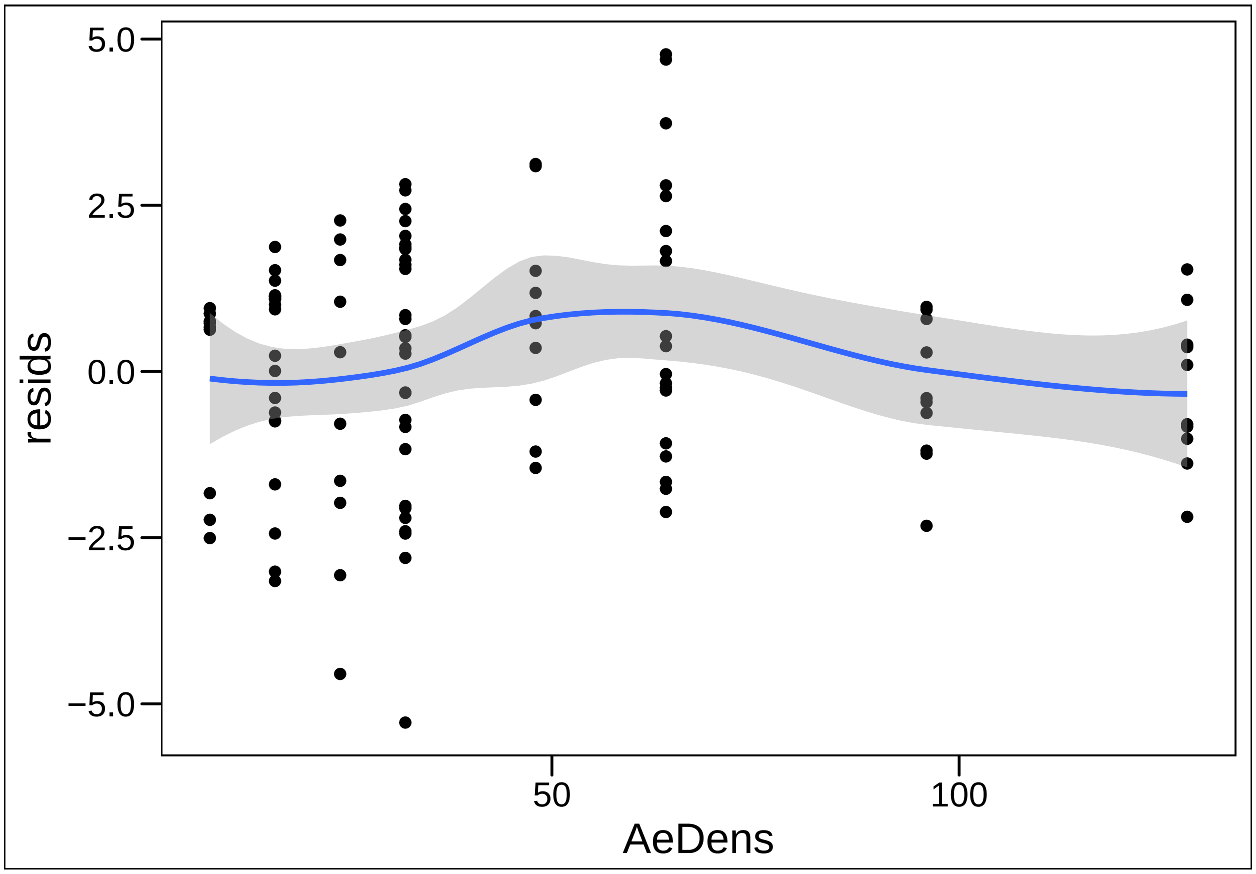
```r
ggplot(data = aeSurvival, aes(x=StDens, y = resids))+
  geom_point() +
  geom_smooth(se=T)+
  theme_base()
```

```
## `geom_smooth()` using method = 'loess'
```



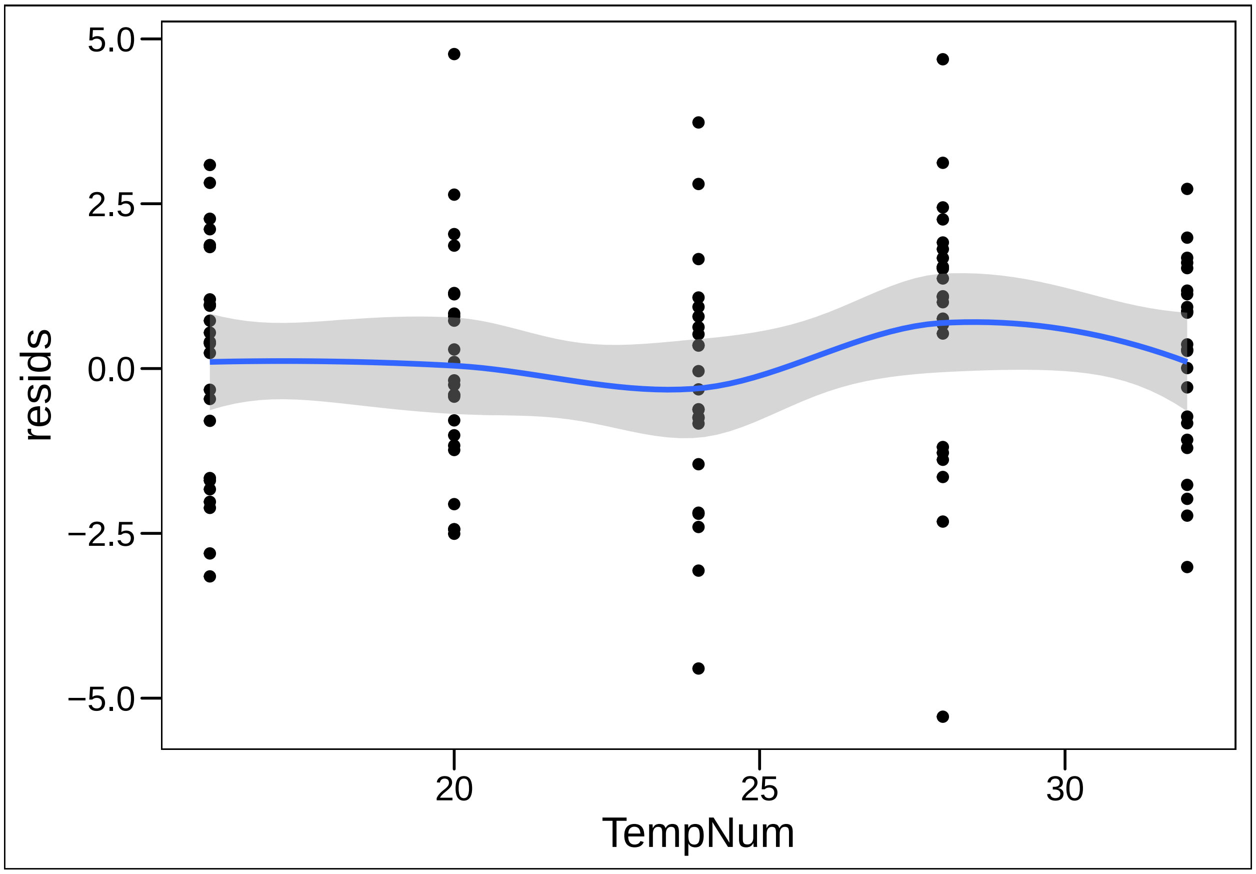
```r
ggplot(data = aeSurvival, aes(x=AeDens, y = resids))+
  geom_point() +
  geom_smooth(se=T)+
  theme_base()
```

```
## `geom_smooth()` using method = 'loess'
```



```r
ggplot(data = aeSurvival, aes(x=TempNum, y = resids))+
  geom_point() +
  geom_smooth(se=T)+
  theme_base()
```

```
## `geom_smooth()` using method = 'loess'
```



The residuals for this model are slightly heteroskedastic, but I think are fine in general.

```r
summary(AedesSurvivalModel)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula:
## cbind(success, failure) ~ AeDensScale + poly(TempNum, 2) + StDensScale +
##     TempNum:StDensScale + (1 | Replicate)
##    Data: aeSurvival
##
##      AIC      BIC   logLik deviance df.resid
##    660.2    679.5   -323.1    646.2      110
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -6.9507 -1.1850  0.3458  1.1078  3.6962
##
## Random effects:
##  Groups    Name        Variance Std.Dev.
##  Replicate (Intercept) 0.009197 0.0959
## Number of obs: 117, groups:  Replicate, 2
##
## Fixed effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)           2.78017    0.15444  18.001  < 2e-16 ***
## AeDensScale         -1.99257    0.08363 -23.825  < 2e-16 ***
## poly(TempNum, 2)1    0.09639    0.70680   0.136  0.89153
## poly(TempNum, 2)2   -3.83763    0.51334  -7.476 7.67e-14 ***
## StDensScale         -0.60399    0.29535  -2.045  0.04086 *
## StDensScale:TempNum  0.03404    0.01240   2.745  0.00605 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) ADnsSc p(TN,2)1 p(TN,2)2 StDnsS
## AeDensScale -0.796
## ply(TmN,2)1  0.044 -0.026
## ply(TmN,2)2 -0.096  0.147  0.025
## StDensScale -0.100  0.078  0.681   -0.006
## StDnsScl:TN -0.030 -0.003 -0.701   -0.004   -0.967
```

```
confint(AedesSurvivalModel)
```

```
## Computing profile confidence intervals ...
```

```
##                            2.5 %      97.5 %
## .sig01               0.000000000  0.48341491
## (Intercept)          2.445441095  3.12095464
## AeDensScale         -2.158699523 -1.83075883
## poly(TempNum, 2)1   -1.289426029  1.48235724
## poly(TempNum, 2)2   -4.848420407 -2.83549324
## StDensScale         -1.182425199 -0.02379168
## StDensScale:TempNum  0.009849299  0.05849671
```

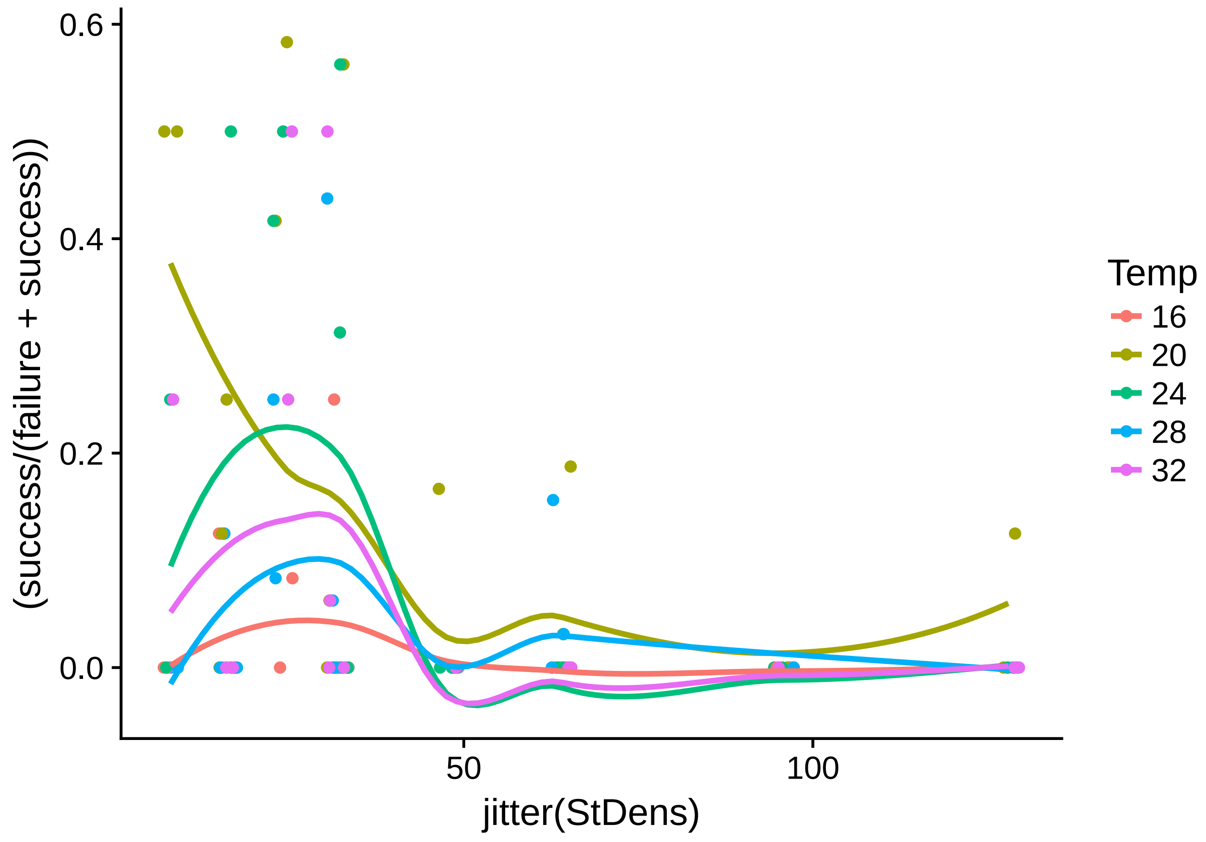
Plot predicted values in a heatmap.

```
#heatmap plots
newData <- expand.grid(AeDens=seq(0,128, by=2), StDens=seq(0,128, by=2), TempNum=c(16,20,24,28,32), Repl
newData <- newData %>%
  mutate(TempScale = as.vector(scale(TempNum, center = F, scale = T))) %>%
  mutate(AeDensScale = as.vector(scale(AeDens, center = F, scale = T))) %>%
  mutate(StDensScale = as.vector(scale(StDens, center = F, scale = T)))

newData$preds <- predict(AedesSurvivalModel, type = "response", newdata = newData)
#remove extrapolation outside of measured range
newData$preds[newData$AeDens+newData$StDens>128] <- NA
#get means over replicates to plot
predicted <- newData %>%
  group_by(TempNum, AeDens, StDens) %>%
  summarise(preds = mean(preds, na.rm = T))

#plot it
aeSurvPlot <- ggplot(predicted, aes(x=StDens, y=AeDens, z=preds))+
  geom_raster(aes(fill=preds))+
  geom_contour(color="black", binwidth = 0.1)+
  theme_minimal()+
  scale_fill_viridis(name="Prop. Emerged", na.value = "gray90",
                     begin = 0, end = 1)+
  #geom_contour(color="white")+
  #scale_fill_gradient(low = "gray90", high = "gray10", na.value = "white")+
  facet_wrap(~TempNum, ncol = 5) +
  xlab("Stephensi Density") +
  ylab("Aegypti Density")
```

Stephensi

```
#stephensi
stSurvival <- emergenceData %>%
  filter(Species == "Stephensi" & StDens!= 0 & Sex == "Female") %>%
  group_by(Replicate, Temp, TempNum, AeDens, StDens, Ratio) %>%
  summarise(success = sum(Number, na.rm = T)) %>%
  mutate(failure = (StDens/2) - success) %>%
  ungroup() %>%
  #rescale predictor variables
  mutate(TempScale = as.vector(scale(TempNum, center = F, scale = T))) %>%
  mutate(AeDensScale = as.vector(scale(AeDens, center = F, scale = T))) %>%
  mutate(StDensScale = as.vector(scale(StDens, center = F, scale = T)))
#for when more than 50% were female
stSurvival$failure[stSurvival$failure<0] <- 0
```

```
ggplot(data = stSurvival, aes(y = (success/(failure+success))))+
  geom_point(aes(x = jitter(StDens), color = Temp))+
  geom_smooth(method="loess", aes(color = Temp, x = StDens), se = F)
```



Stephensi model selection

```r
m0 <- glmer(cbind(success, failure) ~ 1 + (1|Replicate),
                    data = stSurvival,
                    family = binomial(link = "logit"))

m1 <- glmer(cbind(success, failure) ~ StDensScale + (1|Replicate),
                    data = stSurvival,
                    family = binomial(link = "logit"))

m2 <- glmer(cbind(success, failure) ~ AeDensScale + (1|Replicate),
                    data = stSurvival,
                    family = binomial(link = "logit"))

m3 <- glmer(cbind(success, failure) ~ AeDensScale + StDensScale + (1|Replicate),
                    data = stSurvival,
                    family = binomial(link = "logit"))

m4 <- glmer(cbind(success, failure) ~ StDensScale + TempScale + (1|Replicate),
                    data = stSurvival,
                    family = binomial(link = "logit"))

m5 <- glmer(cbind(success, failure) ~ AeDensScale + StDensScale + TempScale + (1|Replicate),
                    data = stSurvival,
                    family = binomial(link = "logit"))

m6 <- glmer(cbind(success, failure) ~ poly(AeDensScale,2) + TempScale + (1|Replicate),
                    data = stSurvival,
                    family = binomial(link = "logit"))

m7 <- glmer(cbind(success, failure) ~ AeDensScale + TempScale + poly(StDensScale,2) + (1|Replicate),
                    data = stSurvival,
                    family = binomial(link = "logit"))

m8 <- glmer(cbind(success, failure) ~ poly(StDensScale,2) + AeDensScale + (1|Replicate),
                    data = stSurvival,
                    family = binomial(link = "logit"))
```

```r
modelSummary <- cbind(model = c("m0","m1","m2","m3", "m4", "m5", "m6", "m7" , "m8"),
                    do.call(rbind, lapply(list(m0, m1, m2, m3, m4, m5, m6, m7, m8), broom::glance)),
                    AICc = AICc(m0, m1, m2, m3, m4, m5, m6, m7,m8),
                    AICweights = Weights(AIC(m0, m1, m2, m3, m4, m5, m6, m7,m8)))
modelSummary
```

```
##    model sigma    logLik      AIC      BIC deviance df.residual AICc.df
## m0    m0     1 -297.6846 599.3692 604.9274 504.6813         117       2
## m1    m1     1 -237.9006 481.8011 490.1385 385.1724         116       3
## m2    m2     1 -256.1250 518.2499 526.5873 421.6337         116       3
## m3    m3     1 -149.5923 307.1846 318.3011 208.3711         115       4
## m4    m4     1 -237.8882 483.7763 494.8928 385.1481         115       4
## m5    m5     1 -149.5698 309.1396 323.0352 208.3260         114       5
## m6    m6     1 -225.8552 461.7105 475.6061 361.1321         114       5
## m7    m7     1 -138.5947 289.1894 305.8641 186.3157         113       6
## m8    m8     1 -138.6183 287.2366 301.1323 186.3631         114       5
##    AICc.AICc   AICweights
## m0  599.4726 1.208987e-68
## m1  482.0098 4.092674e-43
## m2  518.4586 4.980202e-51
## m3  307.5354 3.384615e-05
## m4  484.1272 1.524372e-43
## m5  309.6705 1.273459e-05
## m6  462.2415 9.432540e-39
## m7  289.9394 2.736021e-01
## m8  287.7676 7.263513e-01
```

```
StephSurvivalModel <- m8
```

For stephensi, model m8 is the best fit (survival ~ AeDens + StDens^2). Temperature is not significant in m7, which has a similar AIC as m8.

```
summary(StephSurvivalModel)
```

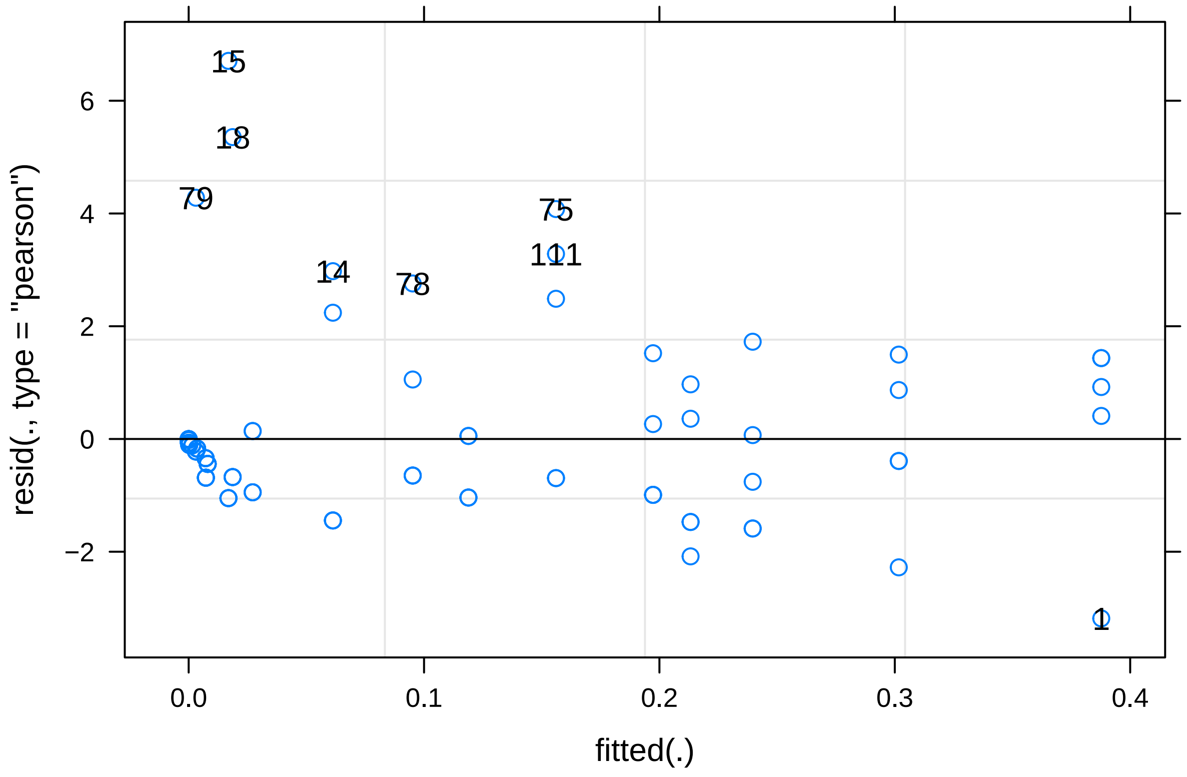
```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: cbind(success, failure) ~ poly(StDensScale, 2) + AeDensScale +
##     (1 | Replicate)
##    Data: stSurvival
##
##      AIC      BIC   logLik deviance df.resid
##    287.2    301.1   -138.6    277.2      114
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.1830 -0.6864 -0.1111 -0.0026  6.7071
##
## Random effects:
##  Groups    Name        Variance Std.Dev.
##  Replicate (Intercept) 0.202    0.4494
## Number of obs: 119, groups:  Replicate, 2
##
## Fixed effects:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -1.4190     0.3462  -4.098 4.16e-05 ***
## poly(StDensScale, 2)1 -19.6742     1.6183 -12.157  < 2e-16 ***
## poly(StDensScale, 2)2   6.9151     1.5378   4.497 6.90e-06 ***
## AeDensScale             -5.5629     0.6991  -7.957 1.76e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##              (Intr) p(SDS,2)1 p(SDS,2)2
## ply(SDS,2)1 -0.090
## ply(SDS,2)2  0.118 -0.460
## AeDensScale -0.239  0.603    -0.557
```

```
confint(StephSurvivalModel)
```

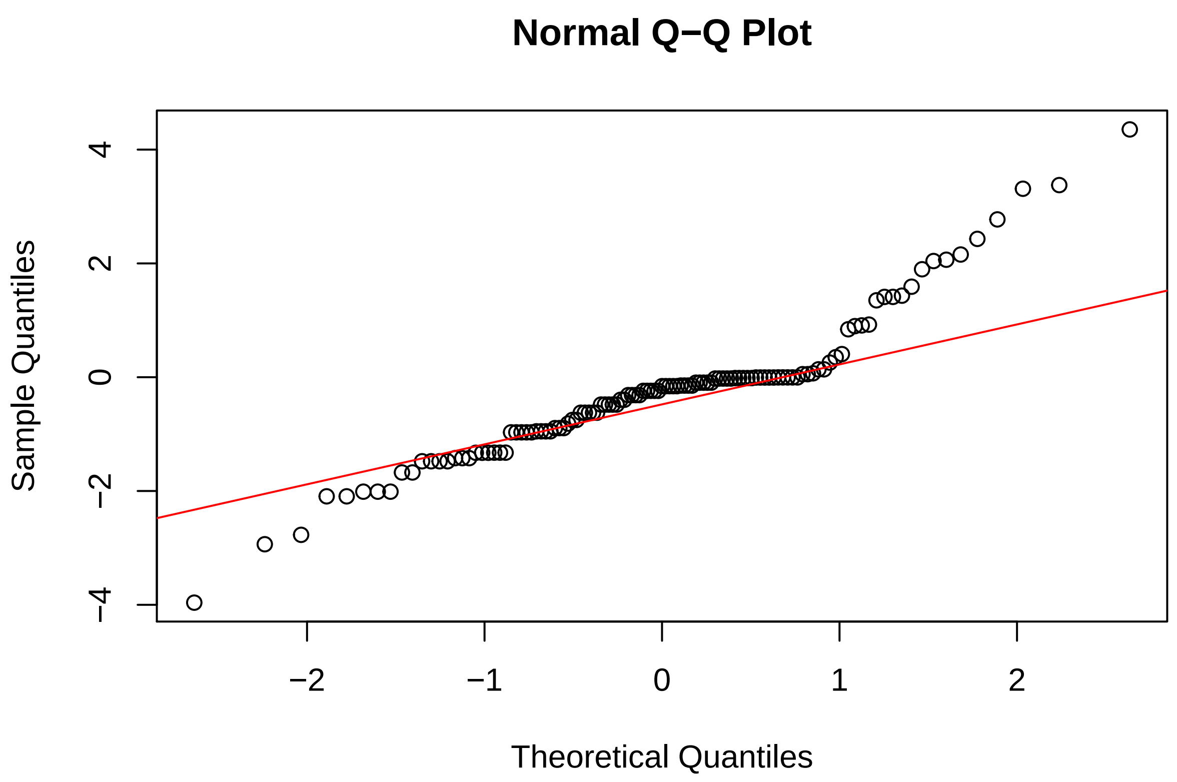
```
## Computing profile confidence intervals ...
```

```
##                             2.5 %      97.5 %
## .sig01                  0.1681237   1.9825641
## (Intercept)            -2.5813812  -0.2776183
## poly(StDensScale, 2)1 -23.0023494 -16.6240987
## poly(StDensScale, 2)2   3.9850359  10.0771246
## AeDensScale             -7.0354834  -4.2887939
```

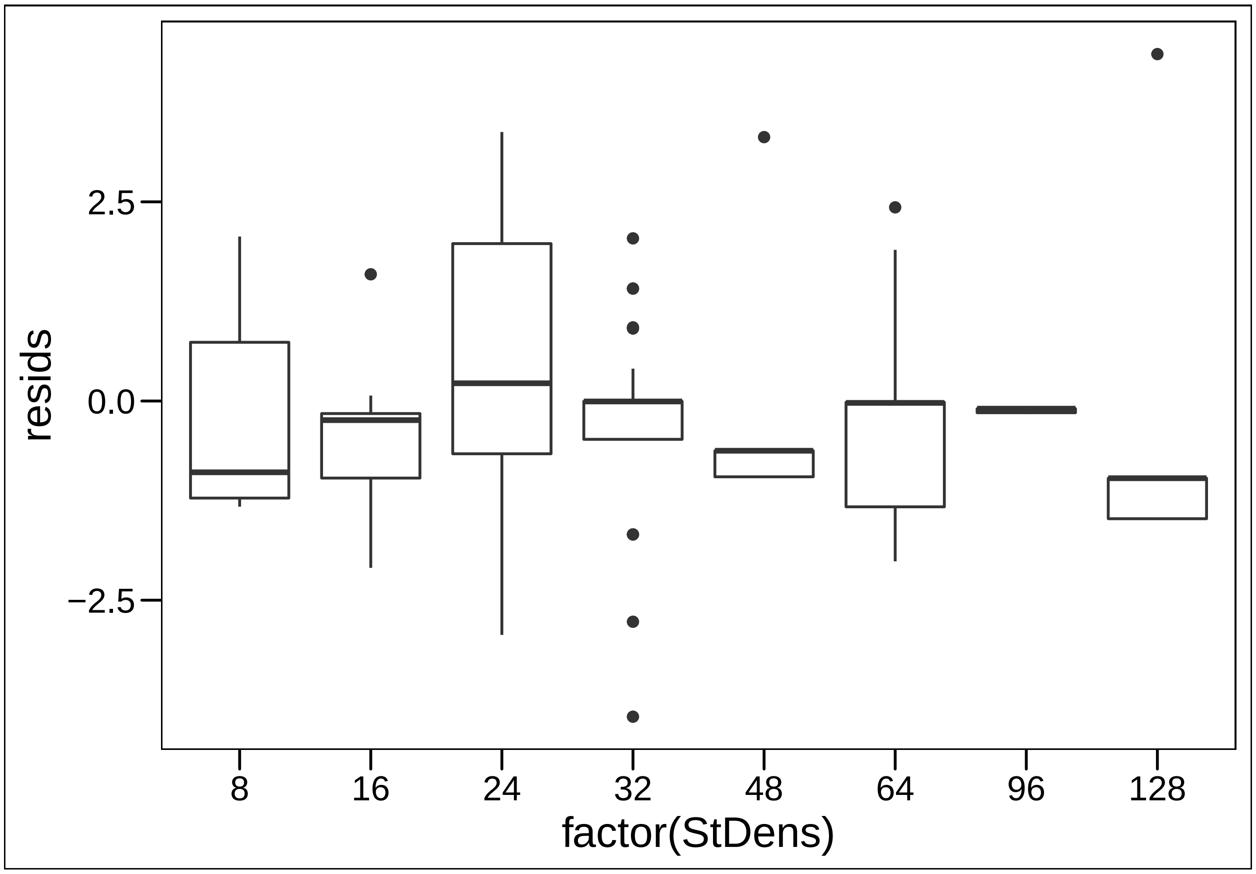
```
plot(StephSurvivalModel, id = 0.01, idLabels=~.obs)
```



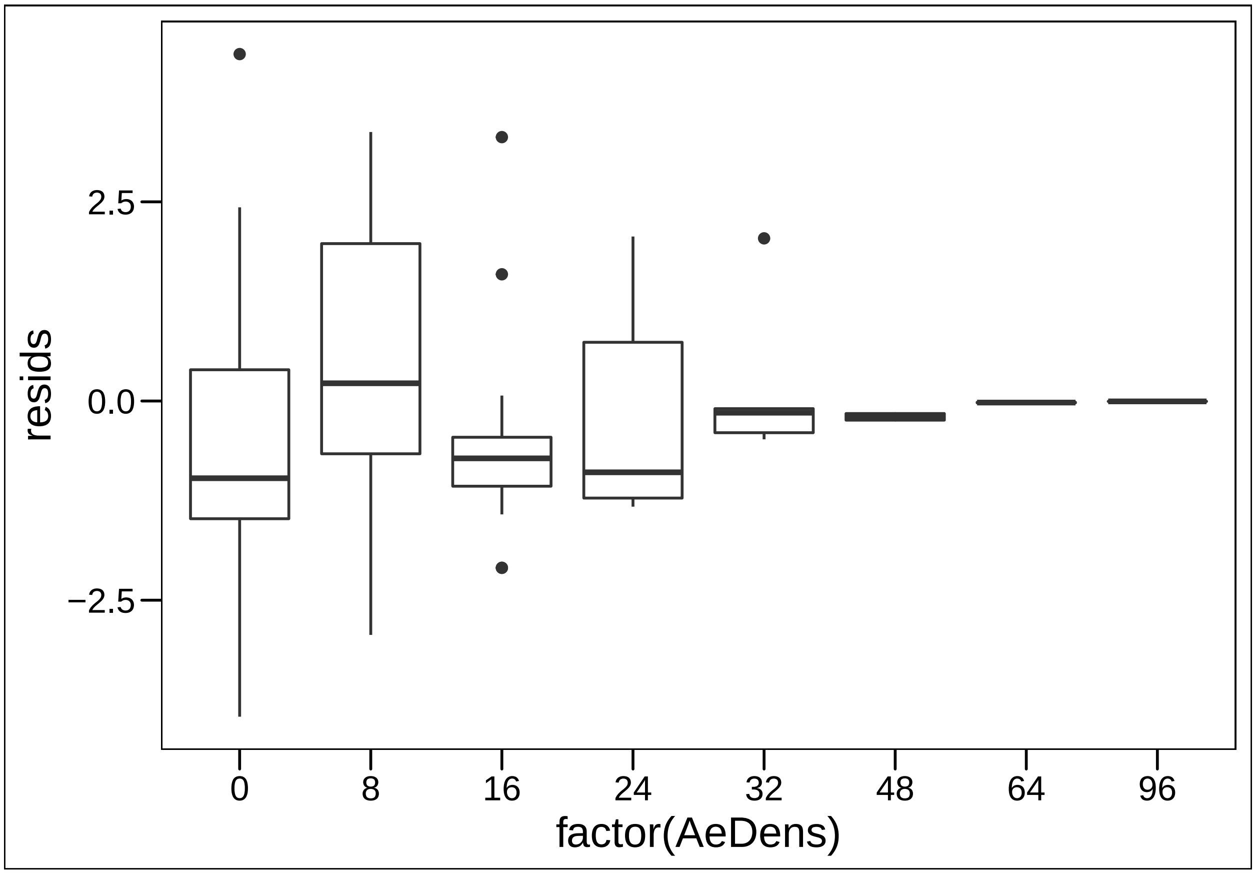
```
qqnorm(resid(StephSurvivalModel))
qqline(resid(StephSurvivalModel), col = "red")
```



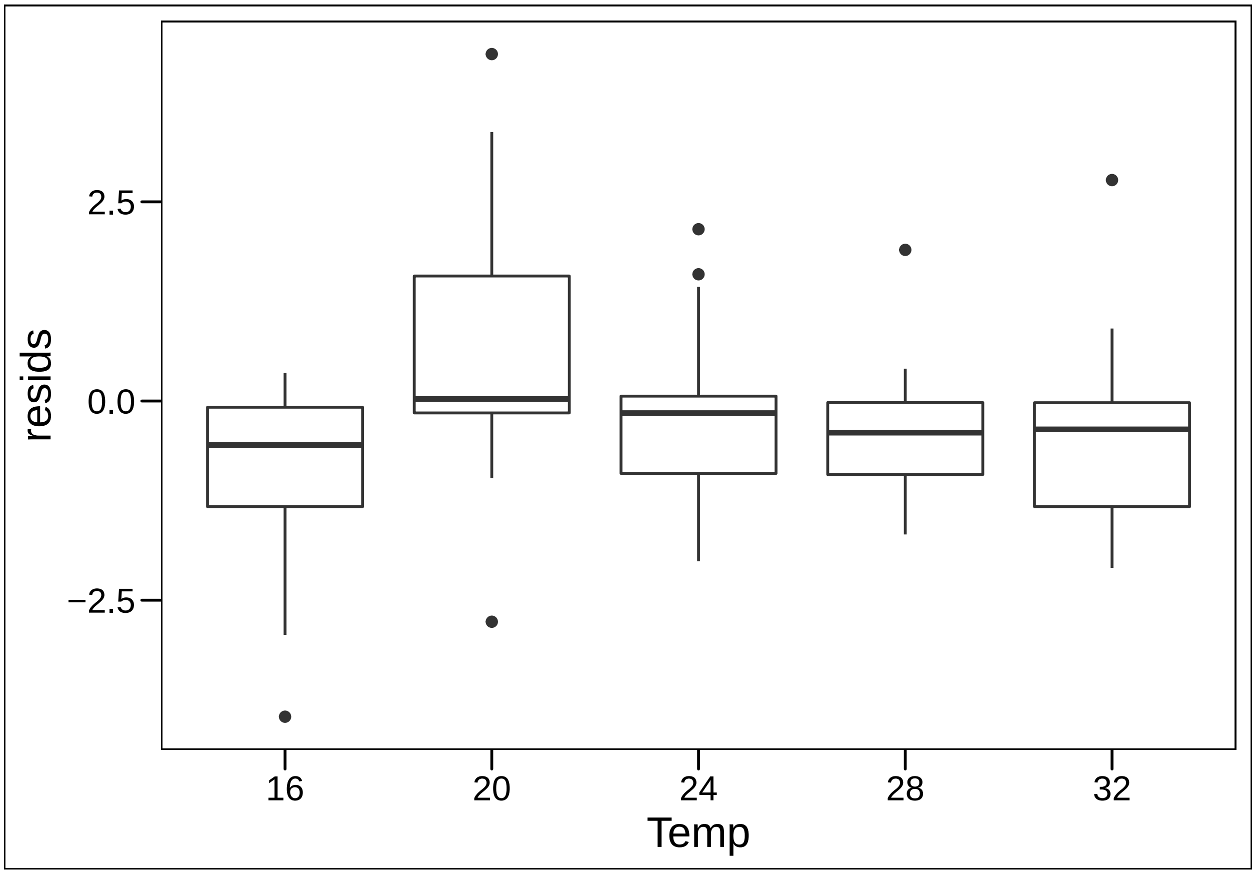
```
stSurvival$preds <- predict(StephSurvivalModel)
stSurvival$resids <- resid(StephSurvivalModel)

ggplot(data = stSurvival, aes(x=factor(StDens), y = resids))+
  geom_boxplot() +
  geom_line(aes(y=0), color="red") +
  theme_base()
```



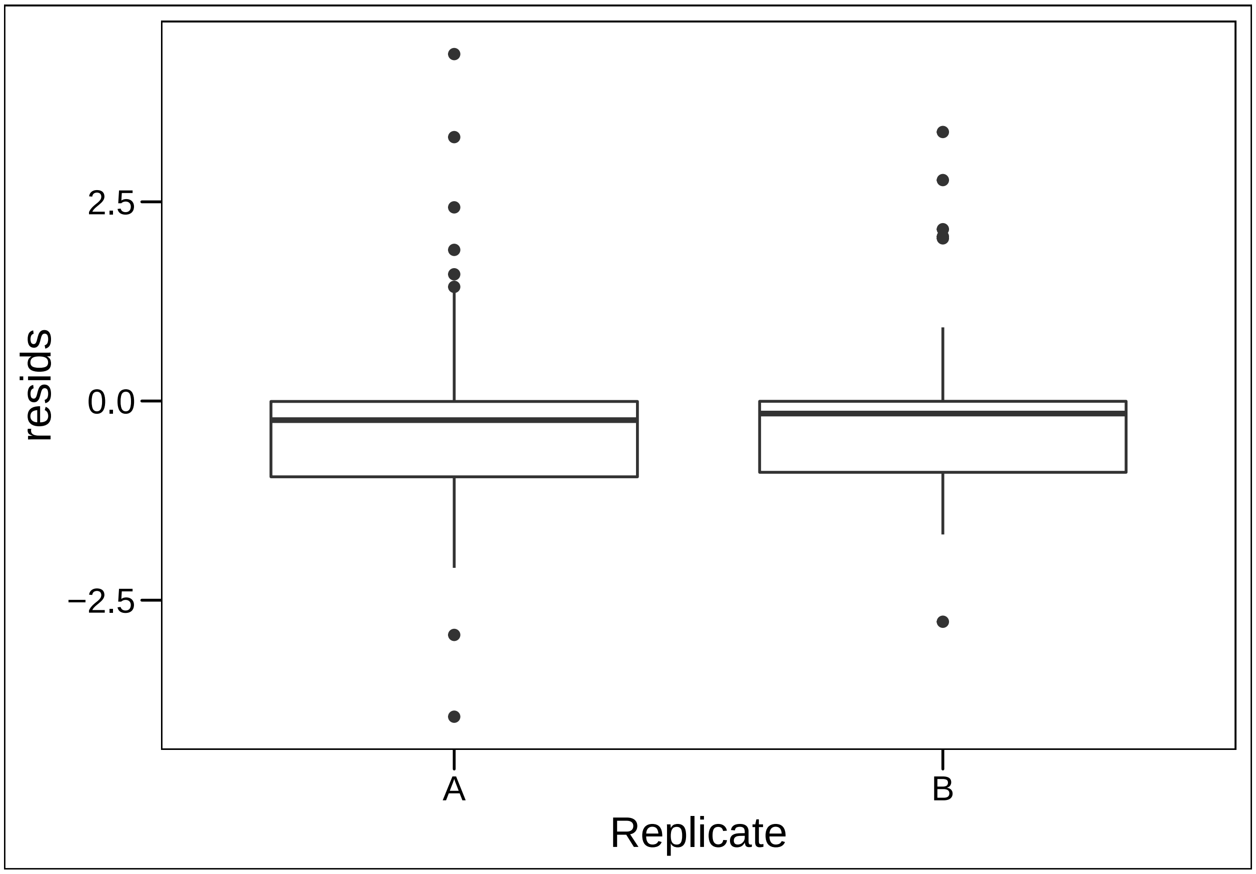
```
ggplot(data = stSurvival, aes(x=factor(AeDens), y = resids))+
  geom_boxplot() +
  theme_base()
```



```
ggplot(data = stSurvival, aes(x=Temp, y = resids))+
  geom_boxplot() +
  theme_base()
```



```
ggplot(data = stSurvival, aes(x=Replicate, y = resids))+
  geom_boxplot() +
  theme_base()
```



The residuals on this model look very bad.

Now we can predict over the response surface.

```
#heatmap plots
newData <- expand.grid(AeDens=seq(0,128, by=2), StDens=seq(0,128, by=2), TempNum=c(16,20,24,28,32), Repl
newData <- newData %>%
  mutate(TempScale = as.vector(scale(TempNum, center = F, scale = T))) %>%
  mutate(AeDensScale = as.vector(scale(AeDens, center = F, scale = T))) %>%
  mutate(StDensScale = as.vector(scale(StDens, center = F, scale = T)))

newData$preds <- predict(StephSurvivalModel, type = "response", newdata = newData)
#remove extrapolation outside of measured range
newData$preds[newData$AeDens+newData$StDens>128] <- NA
#get means over replicates to plot
predicted <- newData %>%
  group_by(AeDens, StDens, TempNum) %>%
  summarise(preds = mean(preds))

#plot it
stSurvPlot <- ggplot(predicted, aes(x=StDens, y=AeDens, z=preds))+
  geom_raster(aes(fill=preds))+
  geom_contour(color="gray90", binwidth=0.1)+
  theme_minimal()+
  scale_fill_viridis(name="Prop. Emerged", na.value = "gray90",
                     begin = 0, end = 1)+
  facet_wrap(~TempNum, ncol = 5) +
  xlab("Stephensi Density") +
  ylab("Aegypti Density")
```

Fecundity

Split into Two Dataframes. Change the NAs of those that didn't lay to 0.

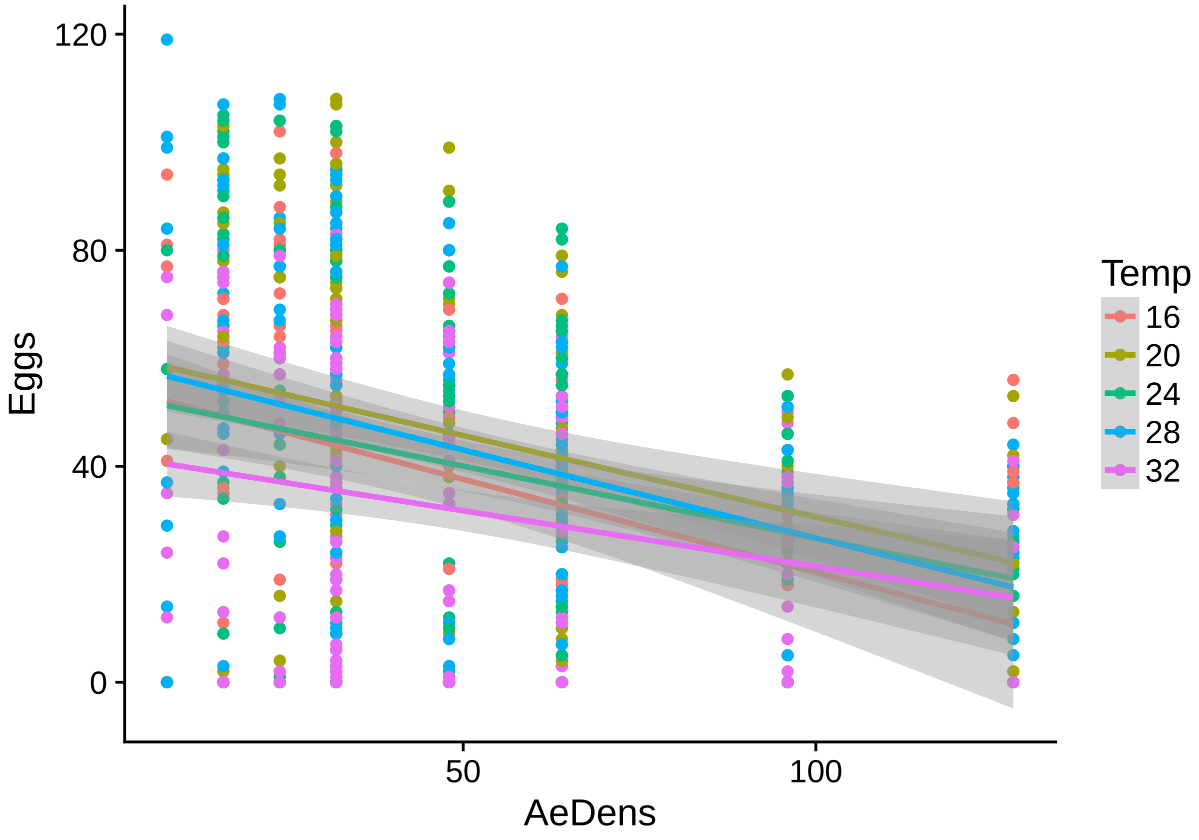
```
aeFec <- dplyr::filter(fecundity, Species == "Aedes")
aeFec$Eggs[is.na(aeFec$Eggs)] <- 0
stFec <- dplyr::filter(fecundity, Species == "Stephensi")
stFec$Eggs[is.na(stFec$Eggs)] <- 0
```

We will use a zero-inflated mixed model here from the `glmmADMB` package because we have many zeros, and this drastically improved model fit during preliminary explorations. Note that these models take a long time to run.

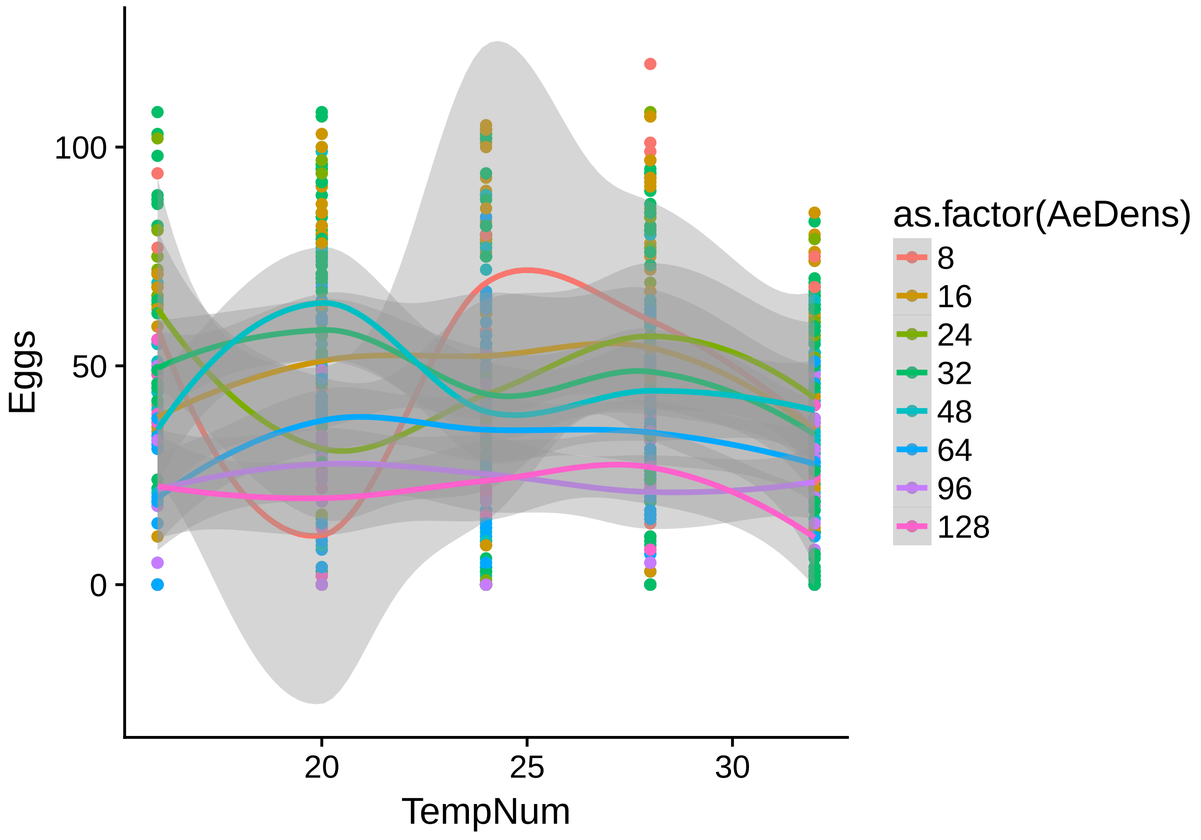
Aedes

Model Selection

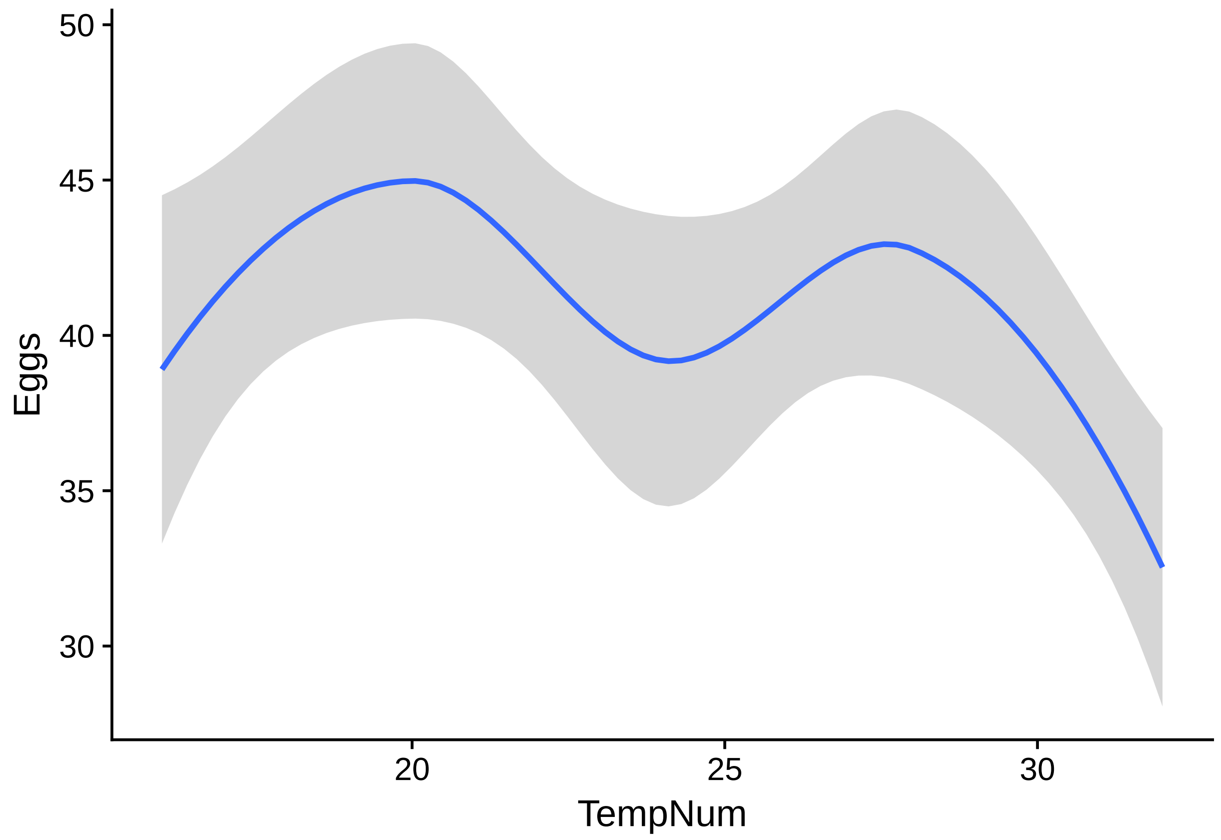
```
ggplot(data = aeFec, aes(y = Eggs, x = AeDens))+
  geom_point(aes(color = Temp))+
  geom_smooth(method = "lm", aes(color = Temp))
```



```
ggplot(data = aeFec, aes(y = Eggs, x = TempNum))+
  geom_point(aes(color = as.factor(AeDens)))+
  geom_smooth(method = "loess", aes(color = as.factor(AeDens)))
```



```
ggplot(data = aeFec, aes(y = Eggs, x = TempNum))+
  geom_smooth(method = "loess")
```



```
m0 <- glmmadmb(Eggs ~ 1 + (1|Replicate),
                   data = aeFec,
                    zeroInflation=TRUE,
                    family="nbinom")

m1 <- glmmadmb(Eggs ~ TempNum + (1|Replicate),
                   data = aeFec,
                    zeroInflation=TRUE,
                    family="nbinom")

m2 <- glmmadmb(Eggs ~ AeDens + (1|Replicate),
                   data = aeFec,
                    zeroInflation=TRUE,
                    family="nbinom")

m3 <- glmmadmb(Eggs ~ AeDens + StDens + (1|Replicate),
                   data = aeFec,
                    zeroInflation=TRUE,
                    family="nbinom")

m4 <- glmmadmb(Eggs ~ AeDens + TempNum + (1|Replicate),
                   data = aeFec,
                    zeroInflation=TRUE,
                    family="nbinom")

m5 <- glmmadmb(Eggs ~ AeDens + poly(TempNum,2) + (1|Replicate),
                  data = aeFec,
                    zeroInflation=TRUE,
                    family="nbinom")

m6 <- glmmadmb(Eggs ~ AeDens*poly(TempNum,2) + (1|Replicate),
                   data = aeFec,
                    zeroInflation=TRUE,
                    family="nbinom")

m7 <- glmmadmb(Eggs ~ AeDens*TempNum + (1|Replicate),
                   data = aeFec,
                    zeroInflation=TRUE,
                    family="nbinom")
```

```
modelSummary <- cbind(model = c("m0","m1","m2","m3", "m4", "m5", "m6", "m7"),
                    logLik = logLik(m0, m1, m2, m3, m4, m5, m6, m7),
                    AIC = AIC(m0, m1, m2, m3, m4, m5, m6, m7),
                    AICc = AICc(m0, m1, m2, m3, m4, m5, m6, m7),
                    AICweights = Weights(AIC(m0, m1, m2, m3, m4, m5, m6, m7)))
modelSummary
```

```
##    model  logLik AIC.df AIC.AIC AICc.df AICc.AICc   AICweights
## m0    m0 -3559.7      4 7127.40       4  7127.448 1.626996e-28
## m1    m1 -3559.7      5 7112.68       5  7112.753 2.557371e-25
## m2    m2 -3559.7      5 7024.64       5  7024.713 3.353026e-06
## m3    m3 -3559.7      6 7024.24       6  7024.342 4.095395e-06
## m4    m4 -3559.7      6 7006.02       6  7006.122 3.704406e-02
## m5    m5 -3559.7      7 7000.52       7  7000.656 5.794665e-01
## m6    m6 -3559.7      9 7001.44       9  7001.659 3.658077e-01
## m7    m7 -3559.7      7 7007.50       7  7007.636 1.767423e-02
```

```
AedesFecundityModel <- m5
```

The best fitting model is m5 (AeDens + TempNum^2).

```
summary(AedesFecundityModel)
```

```
##
## Call:
## glmmadmb(formula = Eggs ~ AeDens + poly(TempNum, 2) + (1 | Replicate),
##     data = aeFec, family = "nbinom", zeroInflation = TRUE)
##
## AIC: 7000.5
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         4.274073   0.071869   59.47  < 2e-16 ***
## AeDens             -0.008073   0.000699  -11.56  < 2e-16 ***
## poly(TempNum, 2)1 -3.040470   0.653640   -4.65  3.3e-06 ***
## poly(TempNum, 2)2 -1.810872   0.656650   -2.76   0.0058 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of observations: total=832, Replicate=2
## Random effect variance(s):
```

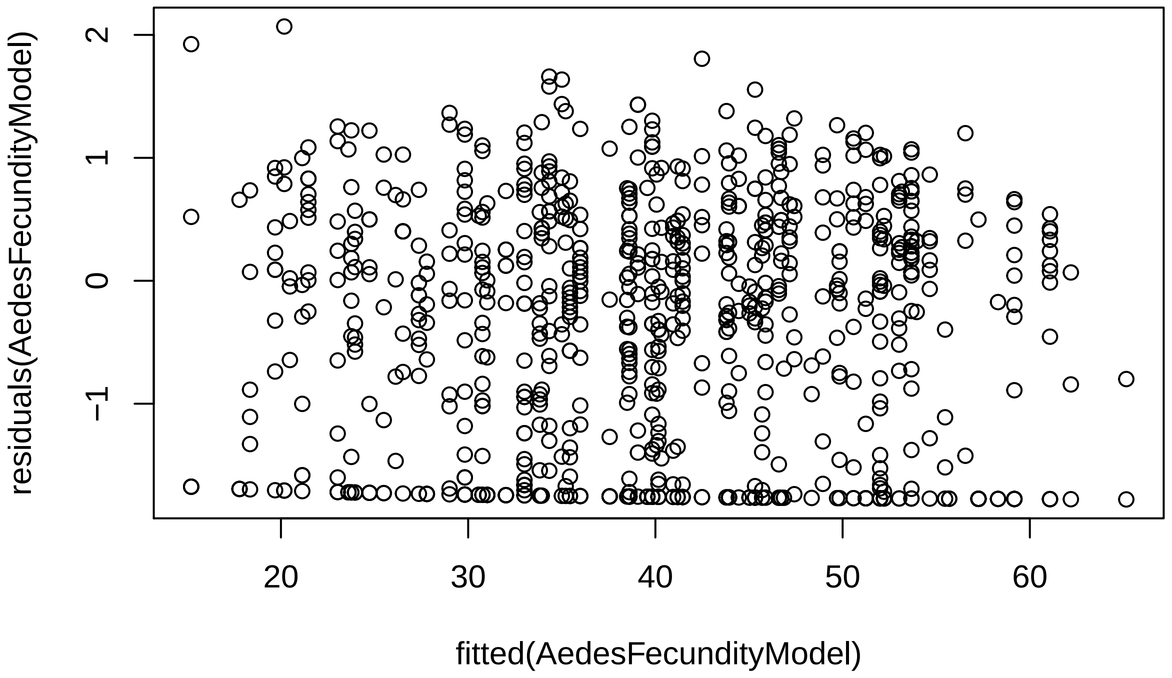
```
## Warning in .local(x, sigma, ...): 'sigma' and 'rdig' arguments are present
## for compatibility only: ignored
```

```
## Group=Replicate
##             Variance  StdDev
## (Intercept) 0.007013 0.08374
##
## Negative binomial dispersion parameter: 3.2897 (std. err.: 0.19832)
## Zero-inflation: 0.20296  (std. err.:  0.013951 )
##
## Log-likelihood: -3493.26
```

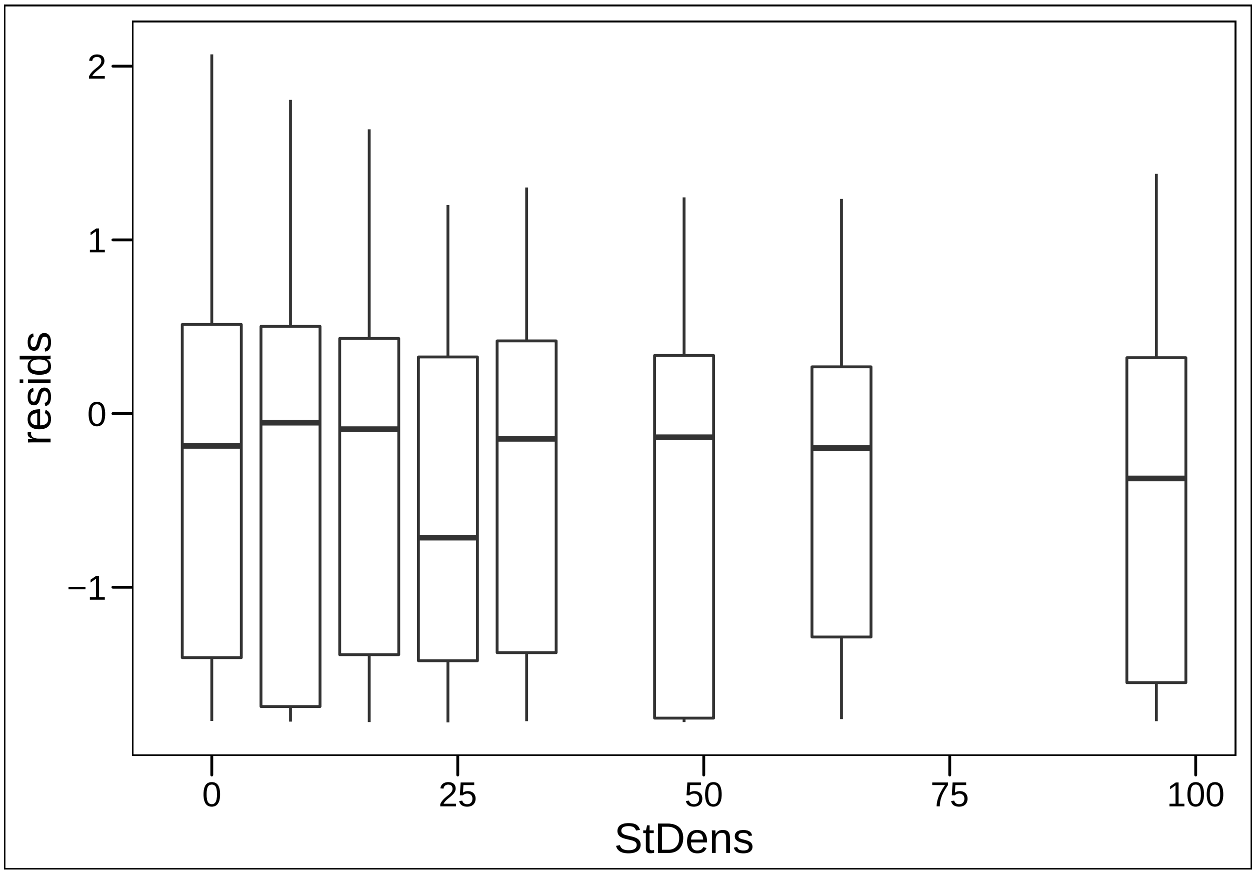
```
confint(AedesFecundityModel)
```

```
##                          2.5 %       97.5 %
## (Intercept)         4.13321201  4.414933313
## AeDens             -0.00944218 -0.006703836
## poly(TempNum, 2)1 -4.32158105 -1.759359330
## poly(TempNum, 2)2 -3.09788248 -0.523861782
```

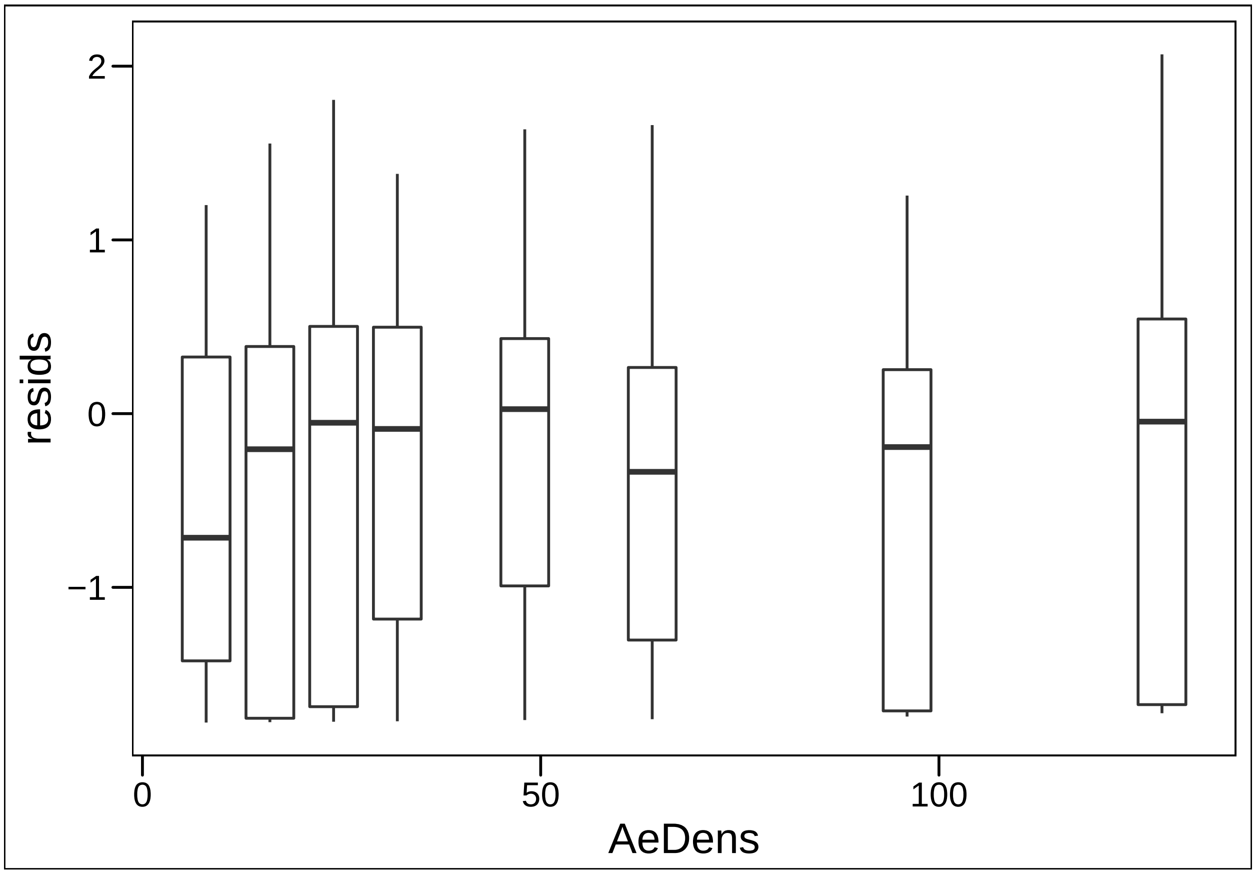
```
#plot(AedesFecundityModel, id = 0.01, idLabels=~.obs)
plot(fitted(AedesFecundityModel), residuals(AedesFecundityModel))
```



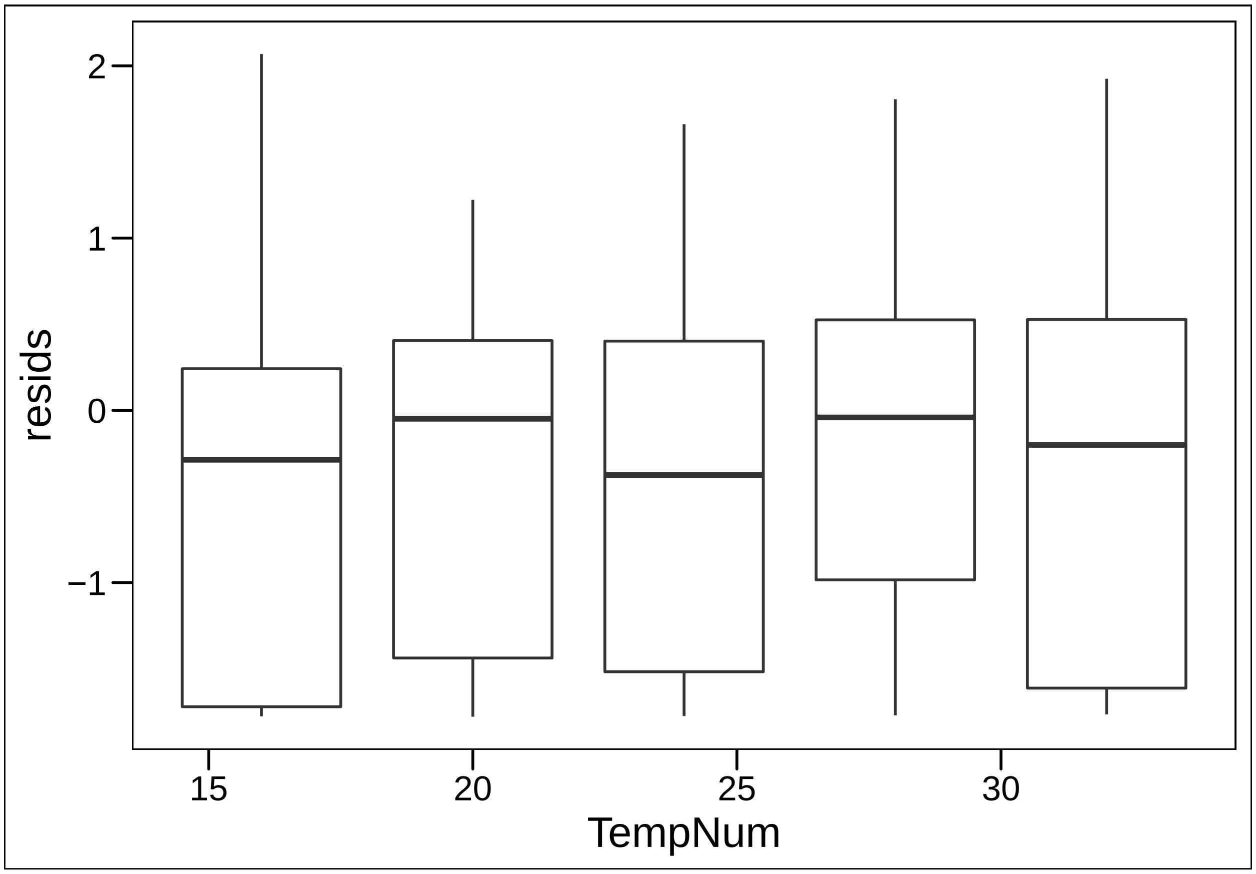
```
# qqnorm(resid(AedesFecundityModel))
# qqline(resid(AedesFecundityModel), col = "red")

aeFec$preds <- predict(AedesFecundityModel)
aeFec$resids <- resid(AedesFecundityModel)

ggplot(data = aeFec, aes(x = StDens, y = resids, group=StDens))+
  geom_boxplot() +
  theme_base()
```



```
ggplot(data = aeFec, aes(x = AeDens, y = resids, group = AeDens))+
  geom_boxplot() +
  theme_base()
```



```r
ggplot(data = aeFec, aes(x = TempNum, y = resids, group = TempNum))+
  geom_boxplot() +
  theme_base()
```



Create heatmaps of the eggs. If there are only two covariates, can do a 3D surface plot.

```
#heatmap plots
newData <- expand.grid(AeDens=seq(0,128, by=2), StDens=seq(0,128, by=2), TempNum=c(16,20,24,28,32), Rep
newData <- newData %>%
  mutate(TempScale = as.vector(scale(TempNum, center = F, scale = T))) %>%
  mutate(AeDensScale = as.vector(scale(AeDens, center = F, scale = T))) %>%
  mutate(StDensScale = as.vector(scale(StDens, center = F, scale = T)))

newData$preds <- predict(AedesFecundityModel, type = "response", newdata = newData)
#remove extrapolation outside of measured range
newData$preds[newData$AeDens+newData$StDens>128] <- NA
#get means over replicates to plot
predicted <- newData %>%
  group_by(TempNum, AeDens) %>%
  summarise(preds = mean(preds, na.rm = T)) %>%
  ungroup()

aeFecPlot<- ggplot(predicted, aes(x=TempNum, y=AeDens, z=preds))+
  geom_raster(aes(fill=preds))+
  geom_contour(color="gray90", binwidth = 10)+
  theme_minimal()+
  scale_fill_viridis(name="Number of Eggs Laid")+
  #facet_wrap(~TempNum, ncol = 5) +
  xlab("Temperature(C)") +
  ylab("Aegypti Density") +
  coord_cartesian(xlim = c(16, 32))
```

Stephensi

There are too many zeros in this for the zero-inflated model to work, so we try a hurdle. We end up dropping the randome effect of replicate because the variance was so low.

Model Selection

```
m0 <- pscl::hurdle(Eggs ~ 1,
                   data = stFec,
                   dist = "negbin")

m1 <- hurdle(Eggs ~ poly(TempNum,2),
                   data = stFec,
                   dist = "negbin")

m2 <- hurdle(Eggs ~ AeDens,
                   data = stFec,
                   dist = "negbin")

m3 <- hurdle(Eggs ~ StDens,
                   data = stFec,
                   dist = "negbin")

m4 <- hurdle(Eggs ~ TempNum,
                   data = stFec,
                   dist = "negbin")
```

```
modelSummary <- cbind(model = c("m0","m1","m2","m3", "m4"),
                   AICc = AICc(m0, m1, m2, m3, m4),
                   AICweights = Weights(AIC(m0, m1, m2, m3, m4)))
modelSummary
```

```
##    model AICc.df AICc.AICc AICweights
## m0    m0       3  329.4214 0.40656754
## m1    m1       7  331.9014 0.20123954
## m2    m2       5  333.4822 0.06617691
## m3    m3       5  330.7322 0.26173239
## m4    m4       5  333.5403 0.06428362
```

```
StephensiFecundityModel <- m0
```

The best fitting model is the null model. None of these factors affect Stephensi fecundity.

```
summary(StephensiFecundityModel)
```

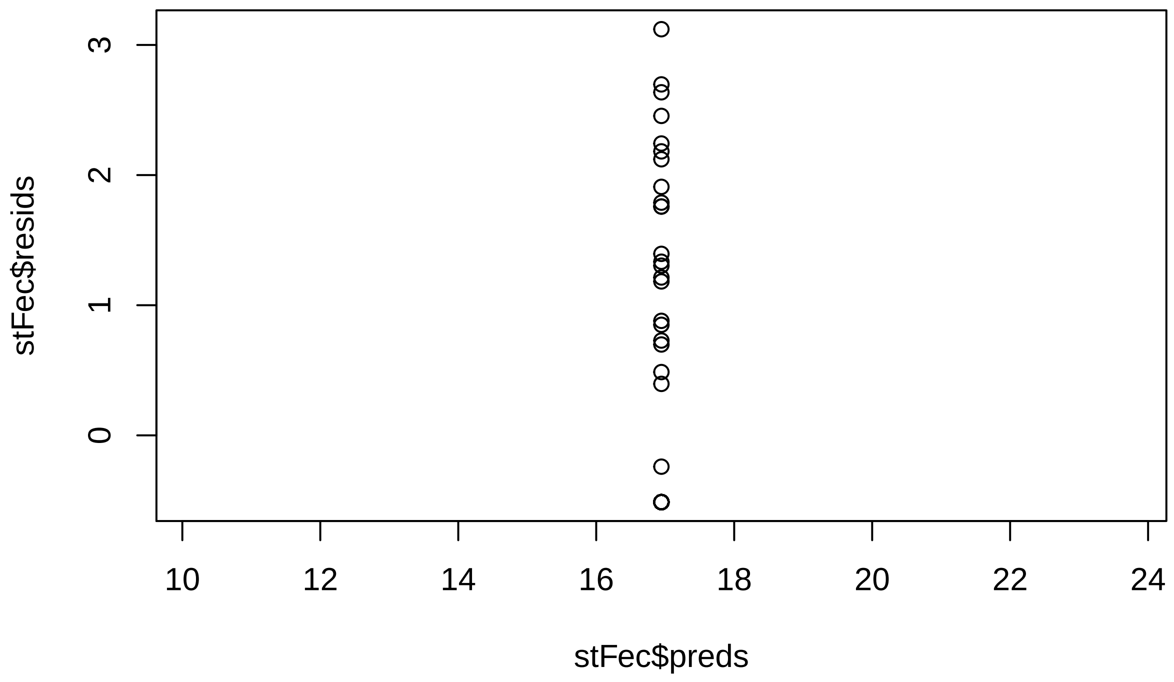
```
##
## Call:
## pscl::hurdle(formula = Eggs ~ 1, data = stFec, dist = "negbin")
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -0.5131 -0.5131 -0.5131 -0.3769  3.1208
##
## Count model coefficients (truncated negbin with log link):
```

```
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.20534    0.09614   43.74  < 2e-16 ***
## Log(theta)   1.62111    0.31725    5.11 3.22e-07 ***
## Zero hurdle model coefficients (binomial with logit link):
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.0840     0.2412  -4.494 6.99e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Theta: count = 5.0587
## Number of iterations in BFGS optimization: 18
## Log-likelihood: -161.6 on 3 Df
```

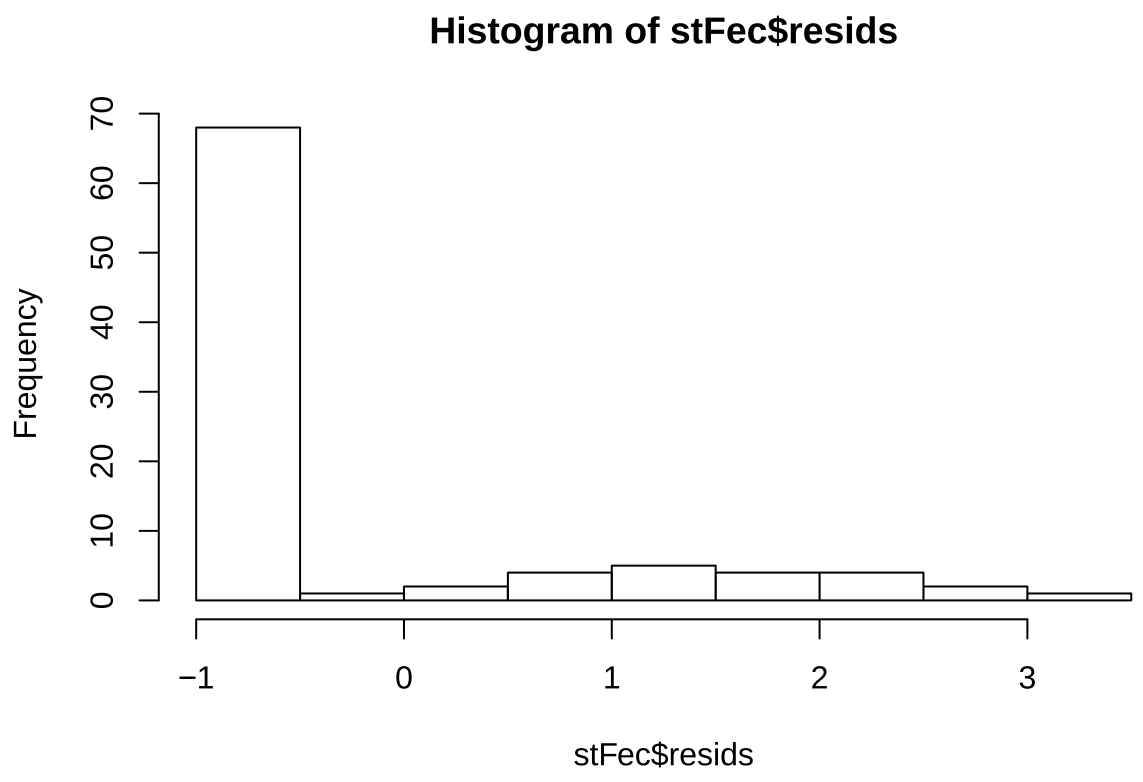
```
confint(StephensiFecundityModel)
```

```
##                          2.5 %     97.5 %
## count_(Intercept)  4.016905  4.3937752
## zero_(Intercept)  -1.556784 -0.6112429
```

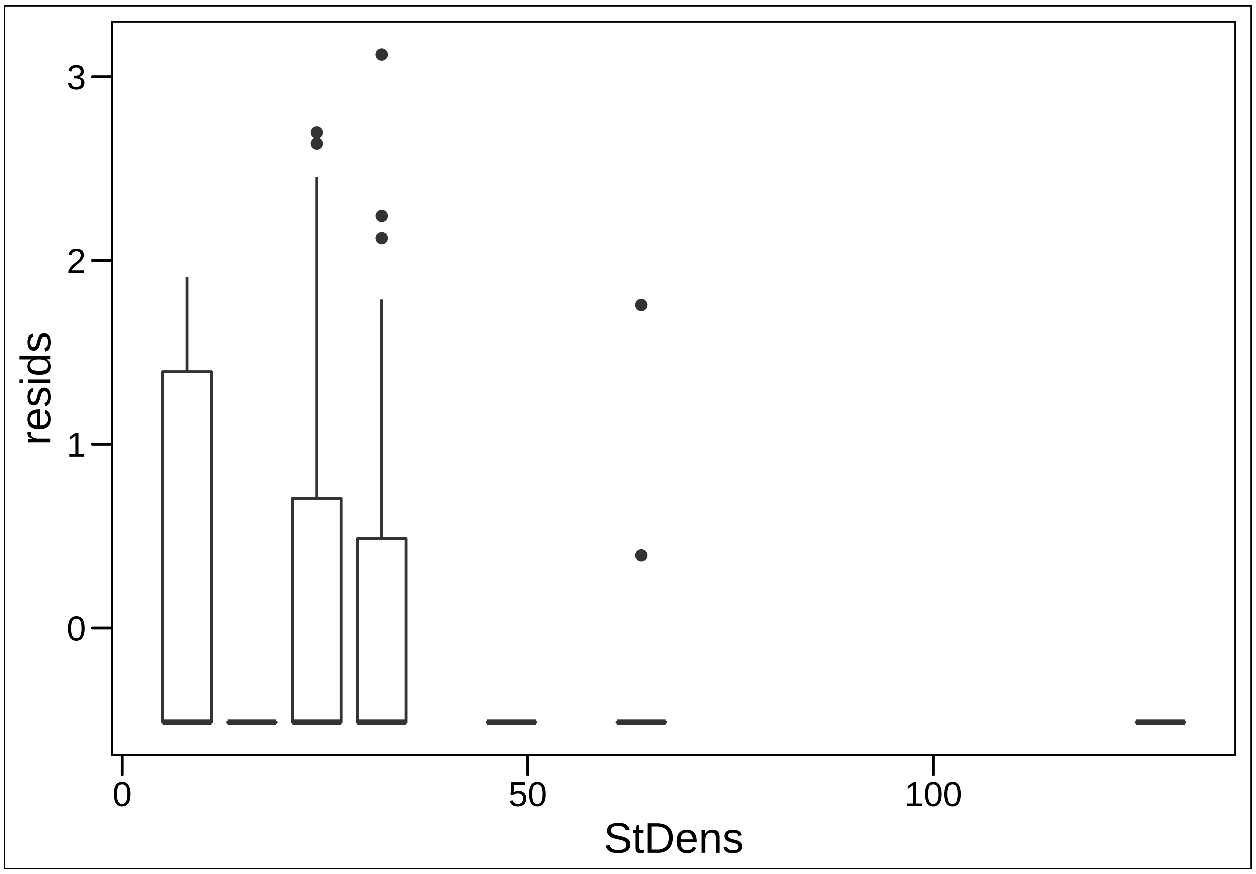
```
stFec$preds <- fitted(StephensiFecundityModel)
stFec$resids <- resid(StephensiFecundityModel)

plot(stFec$preds, stFec$resids)
```



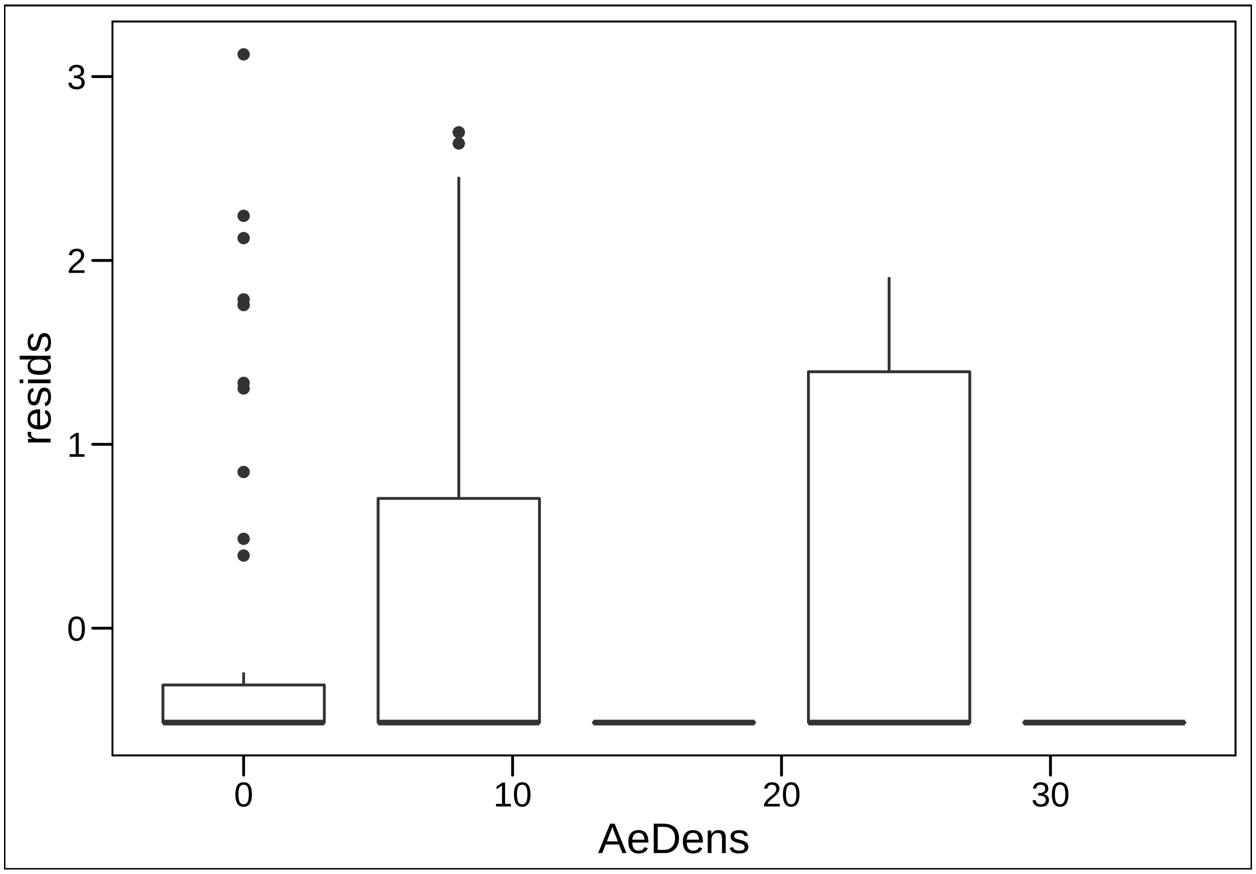
```
hist(stFec$resids)
```



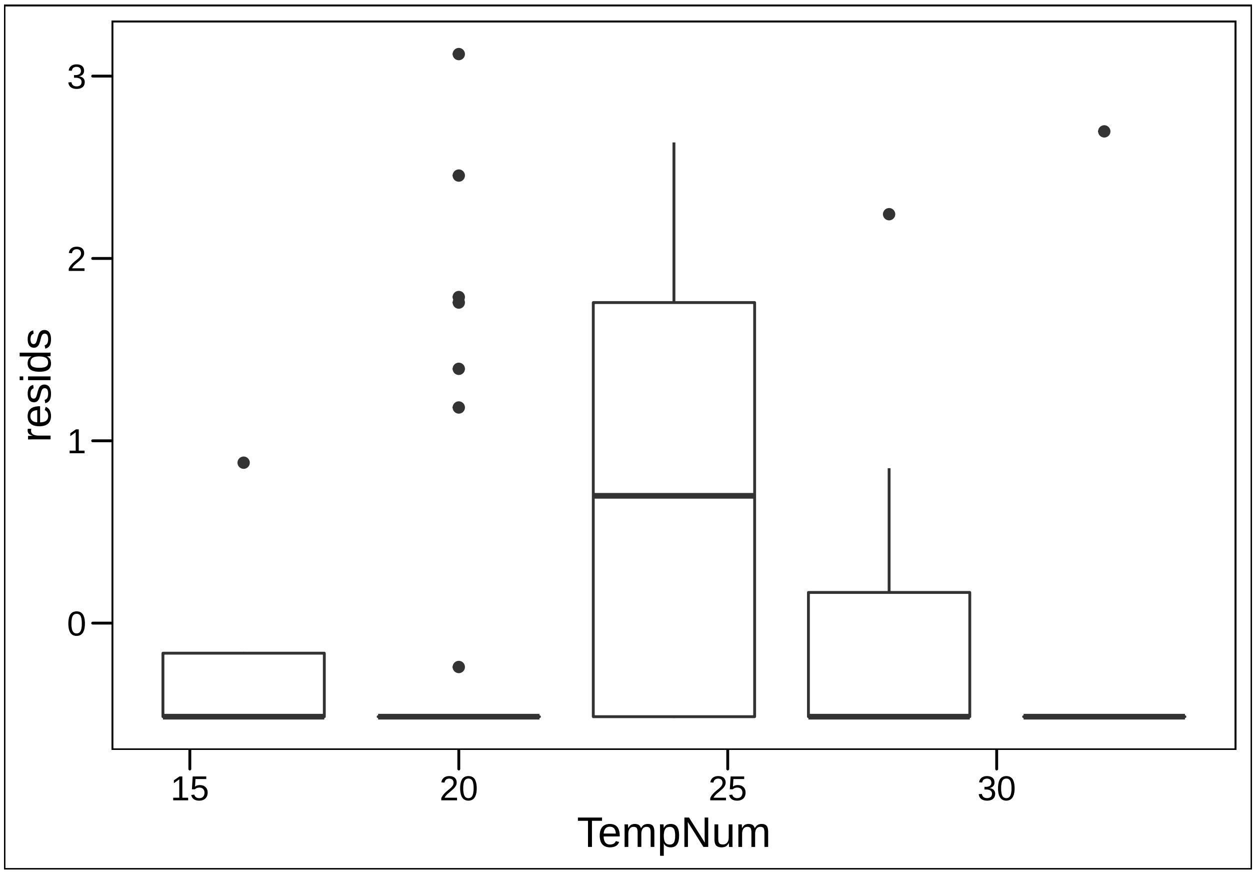
```r
ggplot(data = stFec, aes(x = StDens, y = resids, group = StDens))+
  geom_boxplot() +
  theme_base()
```



```r
ggplot(data = stFec, aes(x = AeDens, y = resids, group = AeDens))+
  geom_boxplot() +
  theme_base()
```



```r
ggplot(data = stFec, aes(x=TempNum, y = resids, group = TempNum))+
  geom_boxplot() +
  theme_base()
```



Growth Rate

Growth rate calculated via Sugihara based on raw data.

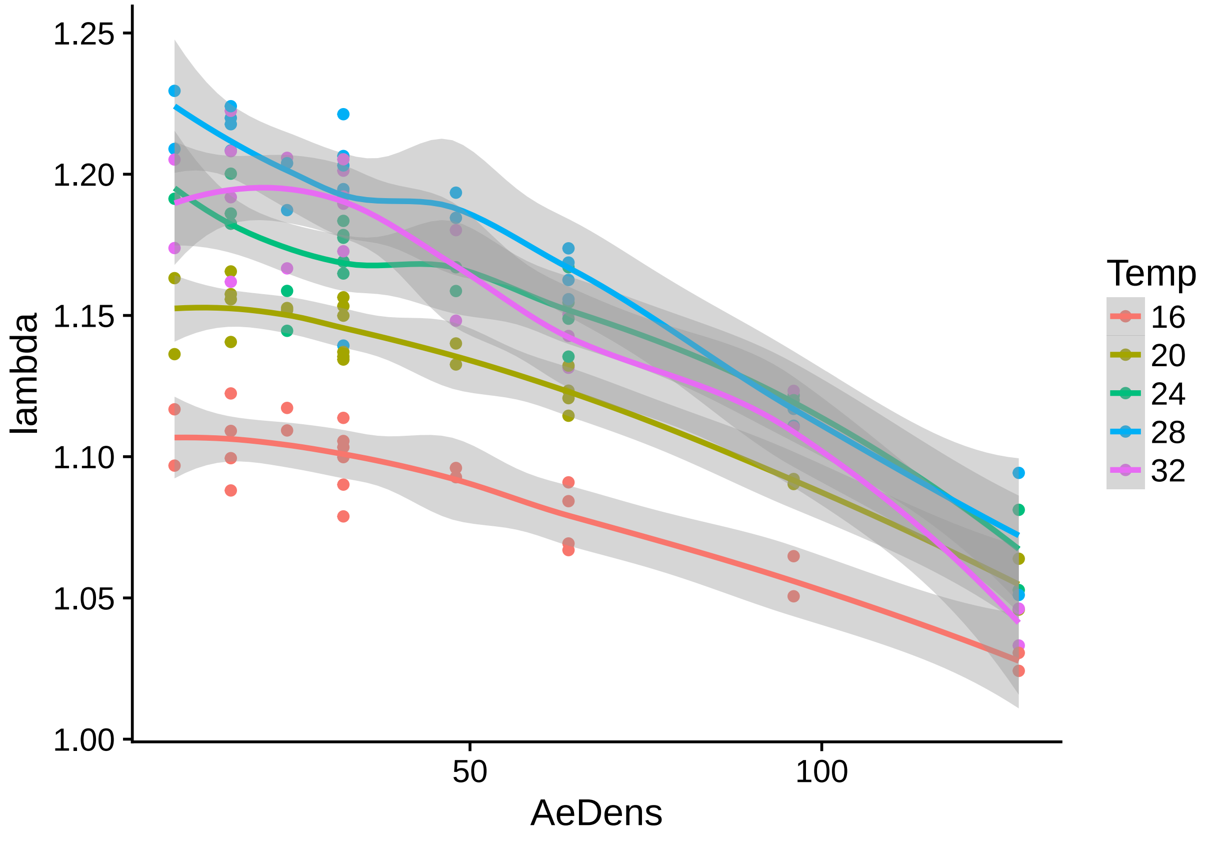
Aedes

Calculate growth rate. Using raw data except for fecundity (since this was only estimated from a subset). Fecundity is estimated using models above. In the case of stephensi, it is just the mean across all treatments, since none of our covariates were significant.

```r
# only need female emergence
GrowthAedes <- emergenceData %>%
  filter(Species =="Aedes" & Sex == "Female") %>%
  #assume 50% female
  mutate(N0 = AeDens/2) %>%
  rename(x = Day, Ax = Number) %>%
  #don't calculate for ones that had no aedes to begin with
  filter(AeDens>0)

#add in predicted fecundity
GrowthAedes$fwx <- predict(AedesFecundityModel, type = "response", newdata = GrowthAedes)

#calculate overall per capita growth rate
GrowthAedes <- GrowthAedes %>%
  mutate(Axfwx = Ax*fwx, xAxfwx = x*Ax*fwx) %>%
  group_by(Replicate, Temp, Ratio, TempNum, AeDens, StDens) %>%
  mutate_at(c("Axfwx", "xAxfwx"), sum, na.rm=T) %>%
  mutate(D=14) %>%
  ungroup() %>%
  mutate(r=(log((1/N0)*Axfwx))/(D+(xAxfwx/Axfwx))) %>%
  group_by(Replicate, Temp, TempNum, Ratio, AeDens, StDens, Species) %>%
  #each row is now a duplicate of the same thing so only need the first one
  slice(1) %>%
  mutate(lambda=exp(r)) %>%
  ungroup() %>%
    #rescale predictor variables
  mutate(TempScale = as.vector(scale(TempNum, center = F, scale = T))) %>%
  mutate(AeDensScale = as.vector(scale(AeDens, center = F, scale = T))) %>%
  mutate(StDensScale = as.vector(scale(StDens, center = F, scale = T))) %>%
  #drop jar that was all male
  filter(!(Replicate == "A" & Temp == "24" & Ratio == "8:24")) %>%
  #drop unneccesary columns
  dplyr::select(-x, -Ax, - Sex, - N0, -fwx, - Axfwx, - xAxfwx, -D)
```

```
ggplot(data =  GrowthAedes, aes(x = AeDens, y = lambda, color = Temp)) +
  geom_point() +
  geom_smooth()
```

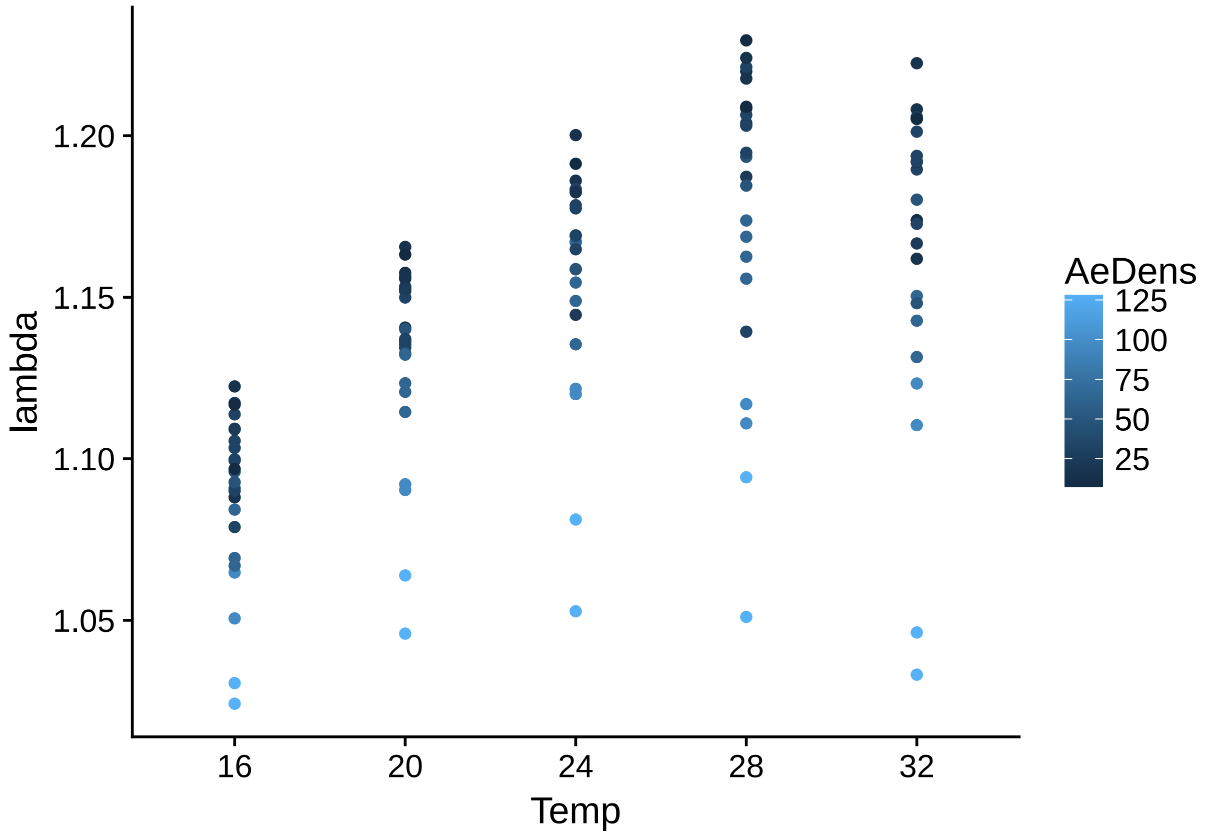
```
## `geom_smooth()` using method = 'loess'
```



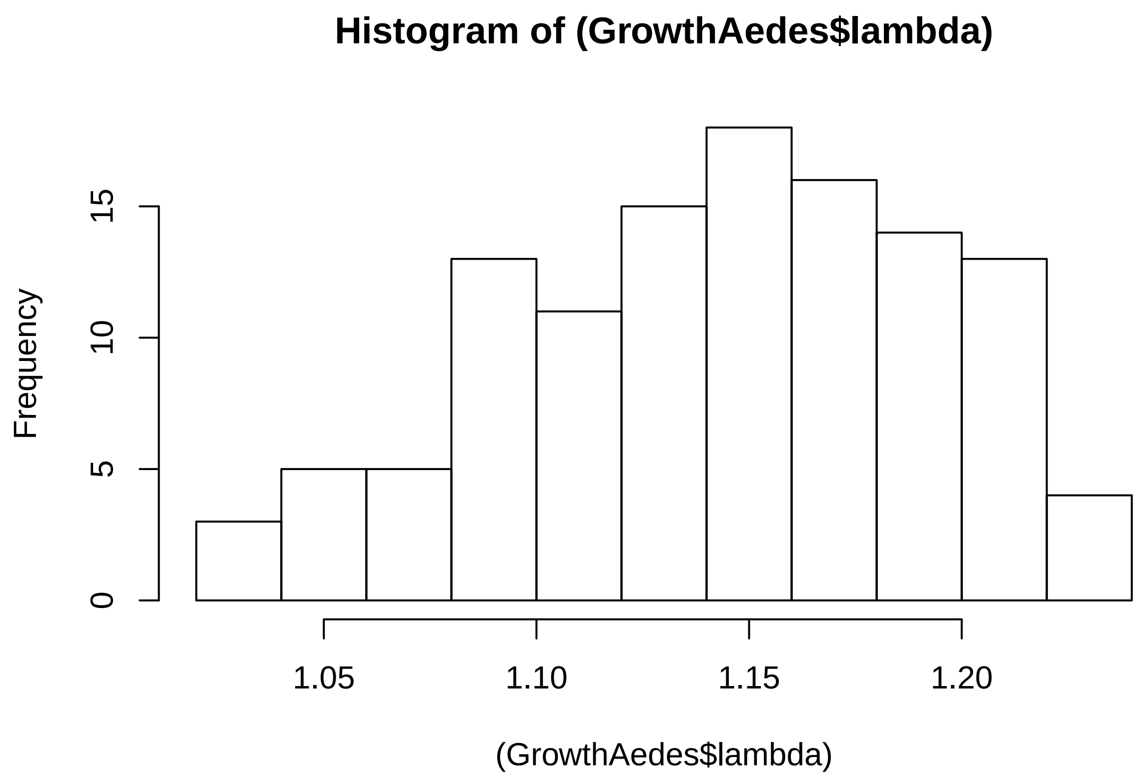
```
ggplot(data =  GrowthAedes, aes(x = Temp, y = lambda, color = AeDens)) +
  geom_point() +
```

```r
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess'
```



```r
hist((GrowthAedes$lambda))
```



Model Selection

```
#model selection
m0 <- glmer(lambda ~ 1 + (1|Replicate),
                   data = GrowthAedes,
                   family = gaussian)
```

```
## Warning in glmer(lambda ~ 1 + (1 | Replicate), data = GrowthAedes, family
## = gaussian): calling glmer() with family=gaussian (identity link) as a
## shortcut to lmer() is deprecated; please call lmer() directly
```

```
m1 <- glmer(lambda ~ poly(TempScale,2) + (1|Replicate),
                   data = GrowthAedes,
                   family = gaussian)
```

```
## Warning in glmer(lambda ~ poly(TempScale, 2) + (1 | Replicate), data =
## GrowthAedes, : calling glmer() with family=gaussian (identity link) as a
## shortcut to lmer() is deprecated; please call lmer() directly
```

```
m2 <- glmer(lambda ~ AeDensScale + (1|Replicate),
                   data = GrowthAedes,
                   family = gaussian)
```

```
## Warning in glmer(lambda ~ AeDensScale + (1 | Replicate), data =
## GrowthAedes, : calling glmer() with family=gaussian (identity link) as a
## shortcut to lmer() is deprecated; please call lmer() directly
```

```
m3 <- glmer(lambda ~ poly(AeDensScale,2)+ (1|Replicate),
                   data = GrowthAedes,
                   family = gaussian)
```

```
## Warning in glmer(lambda ~ poly(AeDensScale, 2) + (1 | Replicate), data =
## GrowthAedes, : calling glmer() with family=gaussian (identity link) as a
## shortcut to lmer() is deprecated; please call lmer() directly
```

```
m4 <- glmer(lambda ~ AeDensScale + poly(TempScale,2) + (1|Replicate),
                  data = GrowthAedes)
```

```
## Warning in glmer(lambda ~ AeDensScale + poly(TempScale, 2) + (1 |
## Replicate), : calling glmer() with family=gaussian (identity link) as a
## shortcut to lmer() is deprecated; please call lmer() directly
```

```
m5 <- glmer(lambda ~ AeDensScale + poly(TempScale,2) + StDensScale + (1|Replicate),
                  data = GrowthAedes,
                  family = gaussian)
```

```
## Warning in glmer(lambda ~ AeDensScale + poly(TempScale, 2) + StDensScale
## + : calling glmer() with family=gaussian (identity link) as a shortcut to
## lmer() is deprecated; please call lmer() directly
```

```
m6 <- glmer(lambda ~ AeDensScale*TempScale+ (1|Replicate),
                  data = GrowthAedes,
                  family = gaussian)
```

```
## Warning in glmer(lambda ~ AeDensScale * TempScale + (1 | Replicate), data
## = GrowthAedes, : calling glmer() with family=gaussian (identity link) as a
## shortcut to lmer() is deprecated; please call lmer() directly
```

```
modelSummary <- cbind(model = c("m0","m1","m2","m3", "m4", "m5", "m6"),
                  do.call(rbind, lapply(list(m0, m1, m2, m3, m4, m5, m6), broom::glance)),
                  AICc = AICc(m0, m1, m2, m3, m4, m5, m6),
                  AICweights = Weights(AIC(m0, m1, m2, m3, m4, m5, m6)))
modelSummary
```

```
##    model      sigma   logLik       AIC       BIC  deviance df.residual
## m0    m0 0.04923772 182.3091 -358.6182 -350.3317 -373.5690         114
## m1    m1 0.03827006 207.8519 -405.7037 -391.8929 -434.5699         112
## m2    m2 0.03588733 215.2306 -422.4613 -411.4126 -448.5905         113
## m3    m3 0.03560982 216.0652 -422.1303 -408.3195 -451.4288         112
## m4    m4 0.01686122 296.3931 -580.7862 -564.2131 -625.8884         111
## m5    m5 0.01674383 292.4653 -570.9306 -551.5954 -628.4225         110
## m6    m6 0.02122358 269.4684 -526.9367 -510.3637 -573.5566         111
##    AICc.df AICc.AICc   AICweights
## m0       3 -358.4058 5.671444e-49
## m1       5 -405.1632 9.510331e-39
## m2       4 -422.1041 4.140410e-35
## m3       5 -421.5898 3.509018e-35
## m4       6 -580.0226 9.928096e-01
## m5       7 -569.9031 7.190382e-03
## m6       6 -526.1731 2.011861e-12
```

```
AedesGrowthModel <- m4
```

The final model is Aedes + Temp^2.

```
summary(AedesGrowthModel)
```

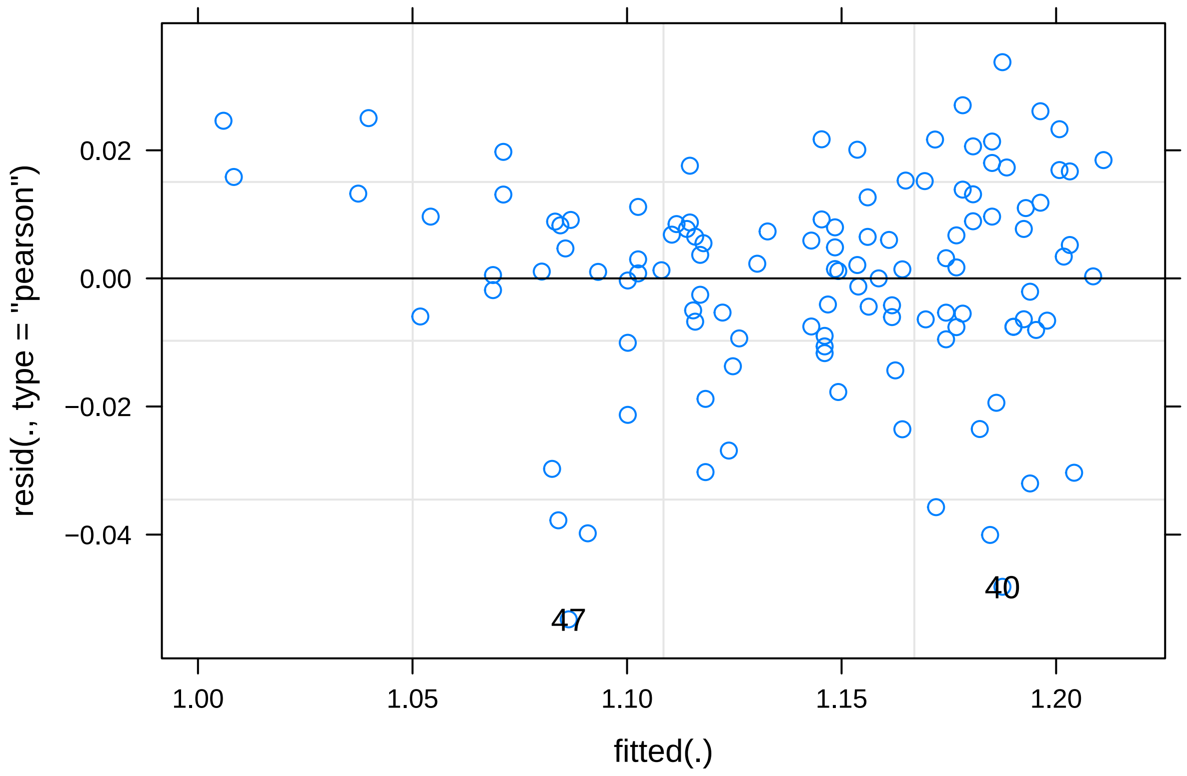
```
## Linear mixed model fit by REML ['lmerMod']
## Formula: lambda ~ AeDensScale + poly(TempScale, 2) + (1 | Replicate)
##    Data: GrowthAedes
##
```

```
## REML criterion at convergence: -592.8
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.1571 -0.4454  0.1016  0.5721  2.0021
##
## Random effects:
##  Groups    Name        Variance  Std.Dev.
##  Replicate (Intercept) 5.485e-06 0.002342
##  Residual              2.843e-04 0.016861
## Number of obs: 117, groups:  Replicate, 2
##
## Fixed effects:
##                       Estimate Std. Error t value
## (Intercept)           1.188818   0.003108   382.5
## AeDensScale          -0.057255   0.002631   -21.8
## poly(TempScale, 2)1  0.300785   0.016861    17.8
## poly(TempScale, 2)2 -0.158920   0.016867    -9.4
##
## Correlation of Fixed Effects:
##             (Intr) ADnsSc p(TS,2)1
## AeDensScale -0.682
## ply(TmS,2)1 -0.002  0.003
## ply(TmS,2)2 -0.013  0.019  0.000
```

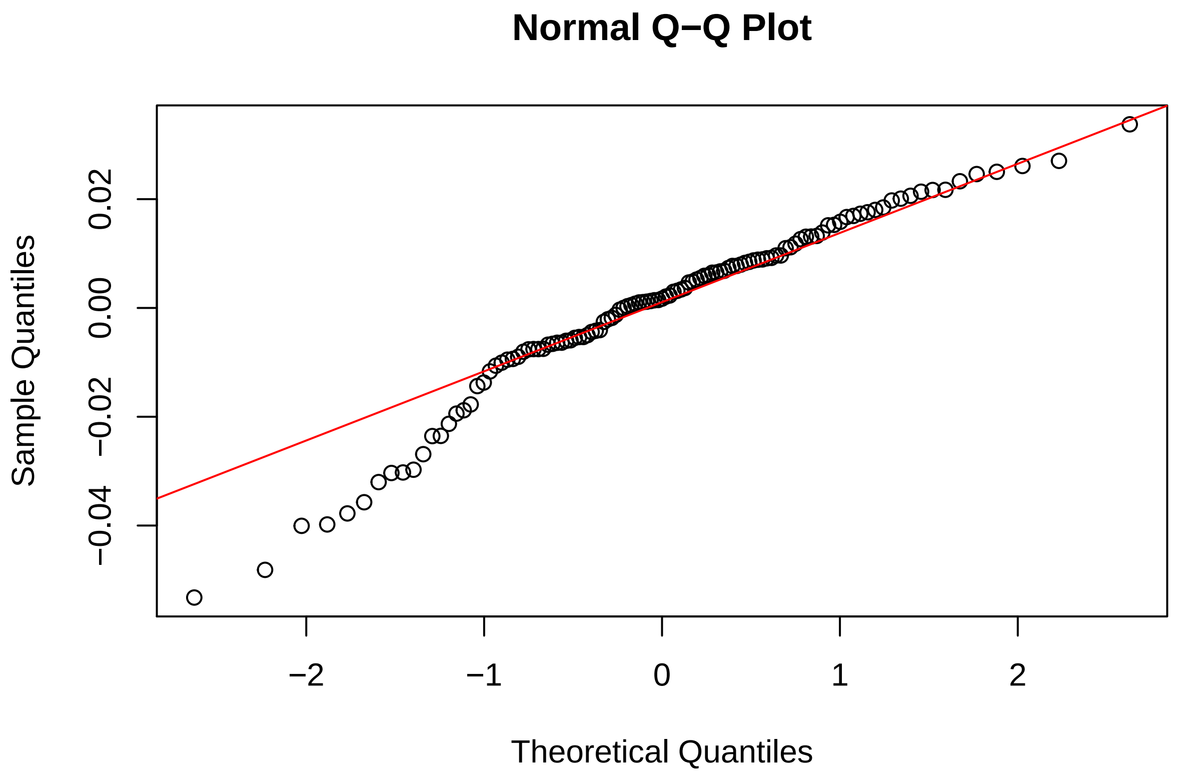
```r
confint(AedesGrowthModel)
```

```
## Computing profile confidence intervals ...

## Warning in optwrap(optimizer, par = start, fn = function(x)
## dd(mkpar(npar1, : convergence code 3 from bobyqa: bobyqa -- a trust region
## step failed to reduce q

## Warning in optwrap(optimizer, par = thopt, fn = mkdevfun(rho, 0L), lower
## = fitted@lower): convergence code 3 from bobyqa: bobyqa -- a trust region
## step failed to reduce q

## Warning in optwrap(optimizer, par = thopt, fn = mkdevfun(rho, 0L), lower
## = fitted@lower): convergence code 3 from bobyqa: bobyqa -- a trust region
## step failed to reduce q

## Warning in optwrap(optimizer, par = thopt, fn = mkdevfun(rho, 0L), lower
## = fitted@lower): convergence code 3 from bobyqa: bobyqa -- a trust region
## step failed to reduce q

## Warning in optwrap(optimizer, par = thopt, fn = mkdevfun(rho, 0L), lower
## = fitted@lower): convergence code 3 from bobyqa: bobyqa -- a trust region
## step failed to reduce q

## Warning in optwrap(optimizer, par = thopt, fn = mkdevfun(rho, 0L), lower
## = fitted@lower): convergence code 3 from bobyqa: bobyqa -- a trust region
## step failed to reduce q

## Warning in optwrap(optimizer, par = thopt, fn = mkdevfun(rho, 0L), lower
## = fitted@lower): convergence code 3 from bobyqa: bobyqa -- a trust region
## step failed to reduce q
## Warning in optwrap(optimizer, par = thopt, fn = mkdevfun(rho, 0L), lower
## = fitted@lower): convergence code 3 from bobyqa: bobyqa -- a trust region
## step failed to reduce q

## Warning in optwrap(optimizer, par = thopt, fn = mkdevfun(rho, 0L), lower
## = fitted@lower): convergence code 3 from bobyqa: bobyqa -- a trust region
## step failed to reduce q

## Warning in optwrap(optimizer, par = thopt, fn = mkdevfun(rho, 0L), lower
## = fitted@lower): convergence code 3 from bobyqa: bobyqa -- a trust region
## step failed to reduce q

## Warning in optwrap(optimizer, par = thopt, fn = mkdevfun(rho, 0L), lower
## = fitted@lower): convergence code 3 from bobyqa: bobyqa -- a trust region
## step failed to reduce q

## Warning in optwrap(optimizer, par = thopt, fn = mkdevfun(rho, 0L), lower
## = fitted@lower): convergence code 3 from bobyqa: bobyqa -- a trust region
## step failed to reduce q

## Warning in optwrap(optimizer, par = thopt, fn = mkdevfun(rho, 0L), lower
## = fitted@lower): convergence code 3 from bobyqa: bobyqa -- a trust region
## step failed to reduce q

##                           2.5 %       97.5 %
## .sig01               0.00000000  0.009266837
## .sigma               0.01470086  0.019035151
## (Intercept)          1.18264086  1.195039674
## AeDensScale         -0.06234742 -0.052089784
## poly(TempScale, 2)1  0.26798708  0.333737367
## poly(TempScale, 2)2 -0.19146150 -0.125699949
```

```
plot(AedesGrowthModel, id = 0.01, idLabels=~.obs)
```

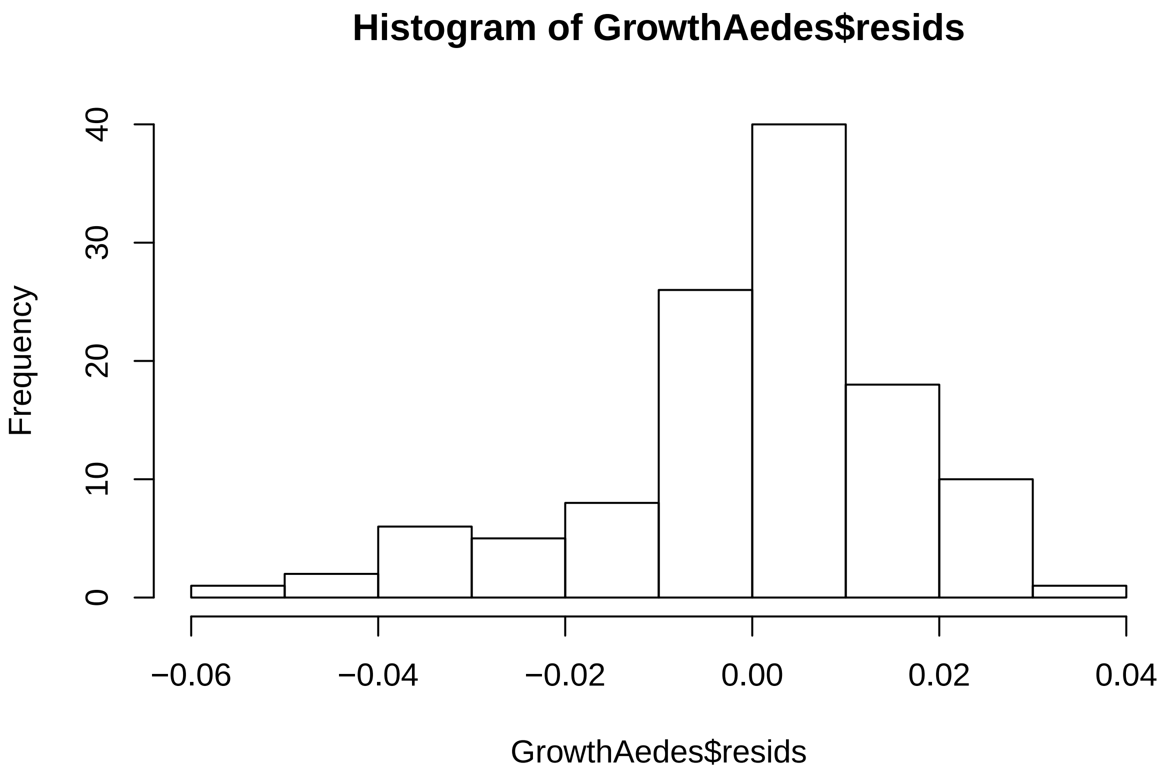


```
qqnorm(resid(AedesGrowthModel))
qqline(resid(AedesGrowthModel), col = "red")
```

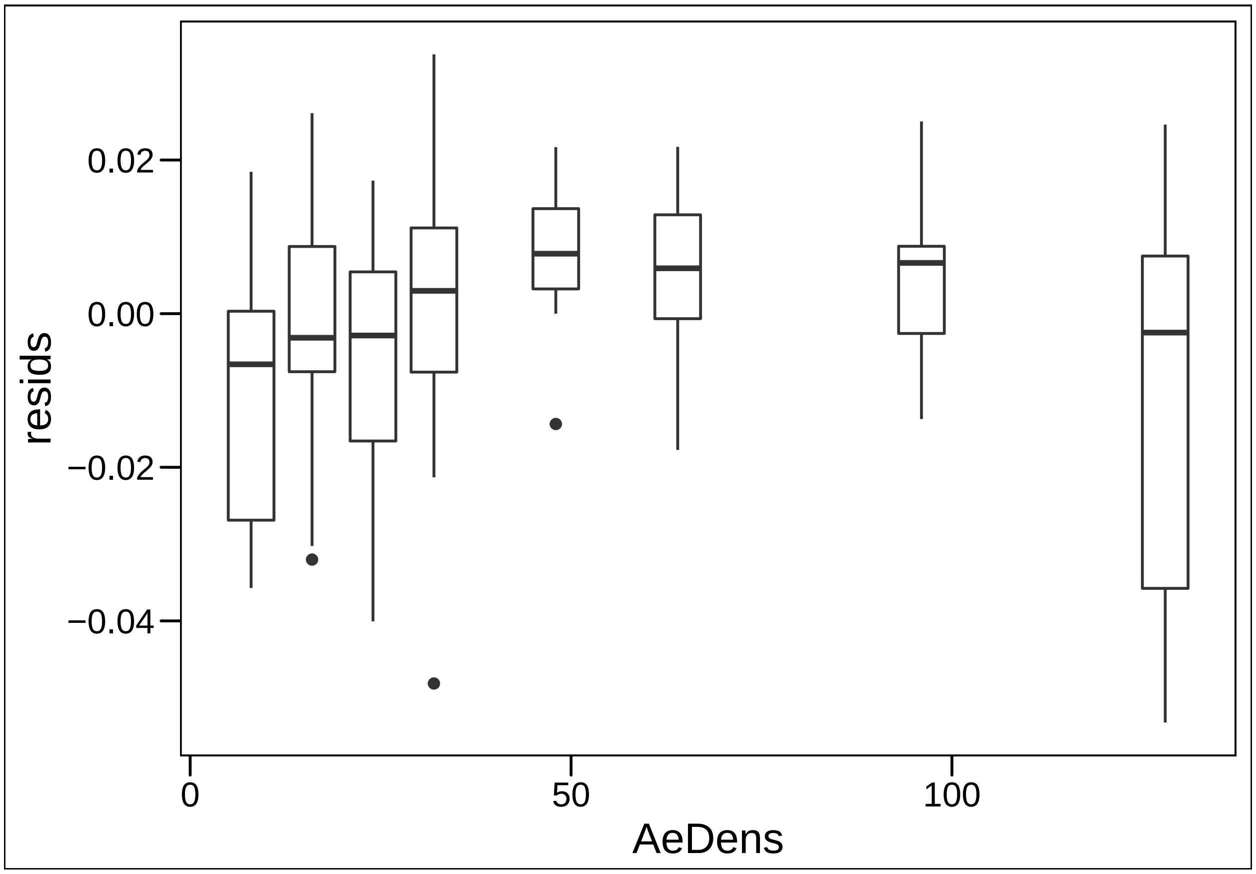


```
GrowthAedes$preds <- predict(AedesGrowthModel)
GrowthAedes$resids <- resid(AedesGrowthModel)
```

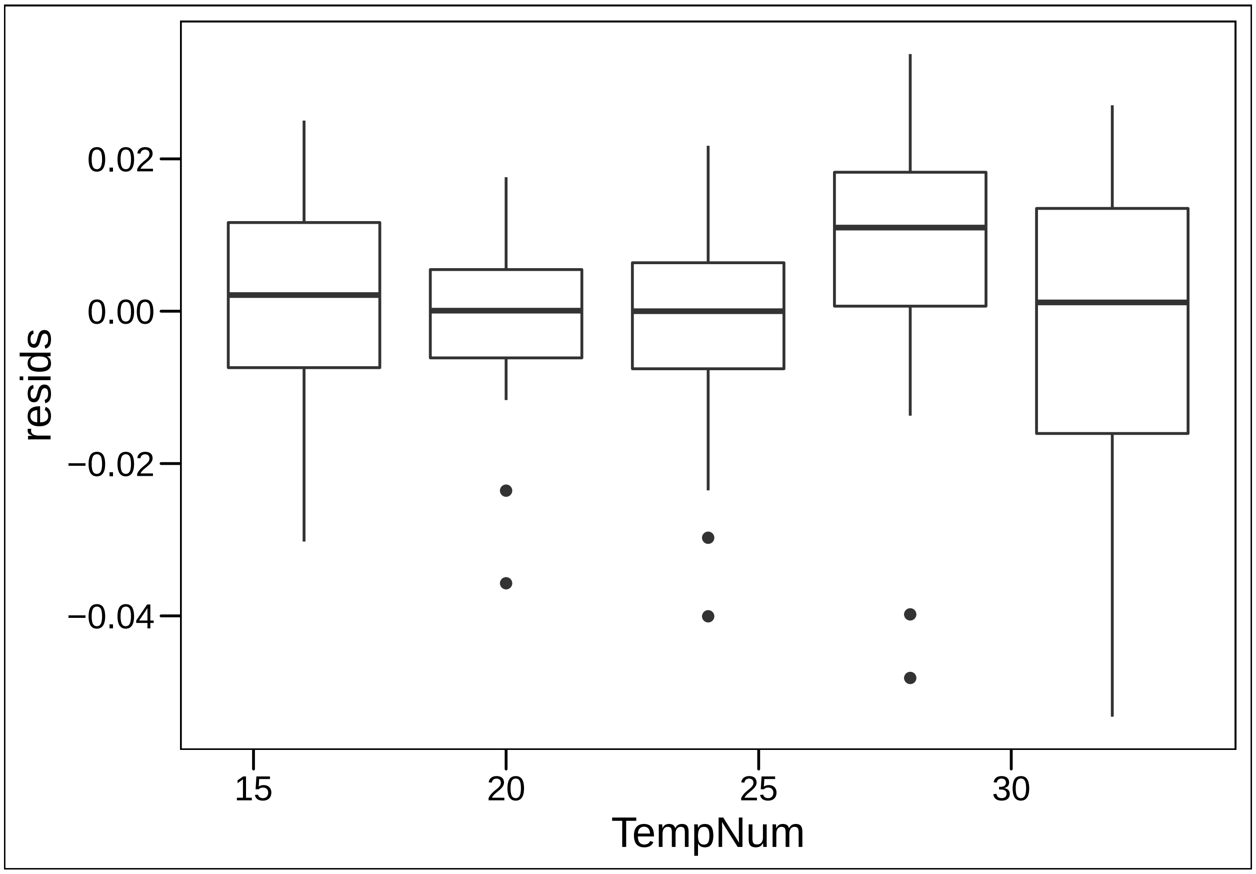
```r
hist(GrowthAedes$resids)
```



```r
ggplot(data = GrowthAedes, aes(x=AeDens, y = resids, group = AeDens))+
  geom_boxplot() +
  theme_base()
```



```
ggplot(data = GrowthAedes, aes(x=TempNum, y = resids, group = TempNum))+
  geom_boxplot() +
  theme_base()
```



Create heatmaps of the growth. Ditto 3D surface above:

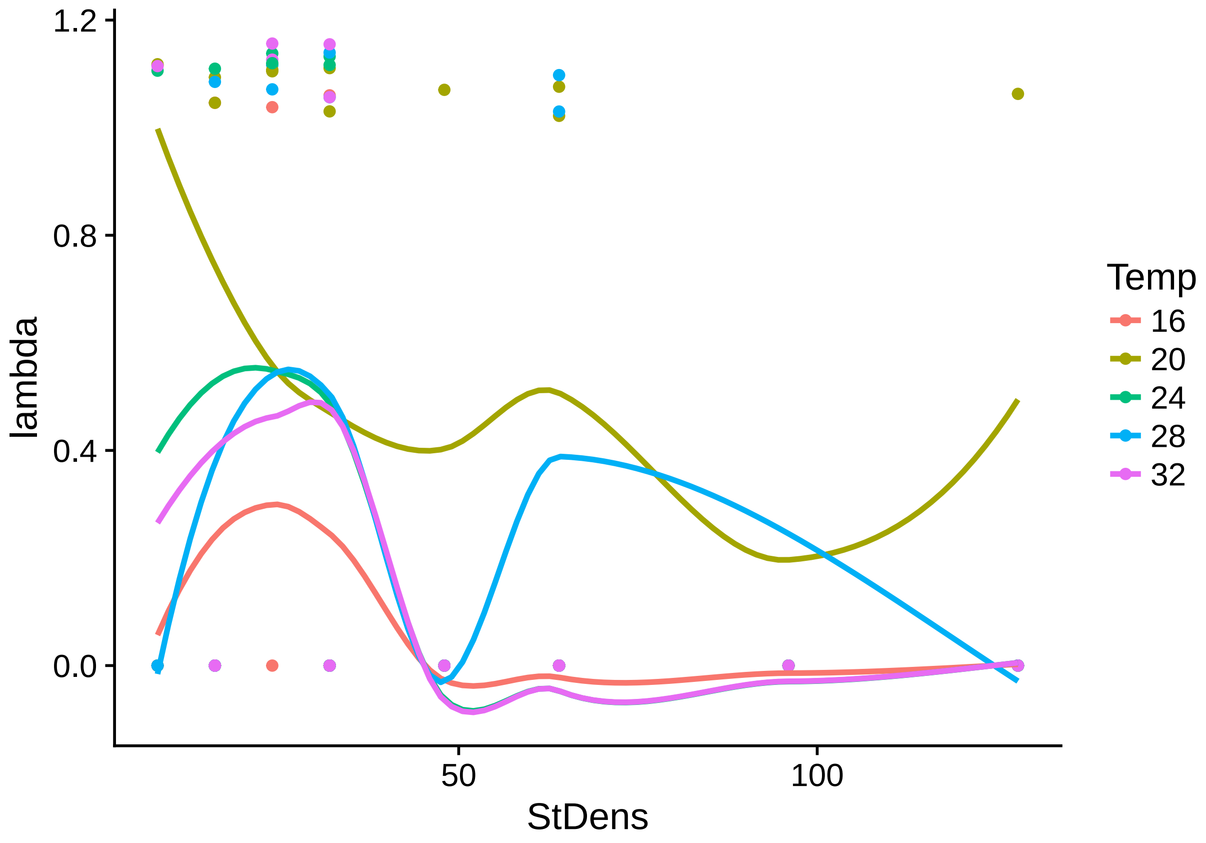
```r
#heatmap plots
newData <- expand.grid(AeDens=seq(0,128, by=2), StDens=seq(0,128, by=2), TempNum=c(16,20,24,28,32), Rep
newData <- newData %>%
  mutate(TempScale = as.vector(scale(TempNum, center = F, scale = T))) %>%
  mutate(AeDensScale = as.vector(scale(AeDens, center = F, scale = T))) %>%
  mutate(StDensScale = as.vector(scale(StDens, center = F, scale = T)))

newData$preds <- predict(AedesGrowthModel, type = "response", newdata = newData)
#remove extrapolation outside of measured range
newData$preds[newData$AeDens+newData$StDens>128] <- NA
#get means over replicates to plot
predicted <- newData %>%
  group_by(TempNum, AeDens, StDens) %>%
  summarise(preds = mean(preds, na.rm=T)) %>%
  ungroup()

aeGrowthPreds <- predicted

#plot it
aeGrowthPlot <- ggplot(aeGrowthPreds, aes(x=StDens, y=AeDens, z=preds))+
  geom_raster(aes(fill=preds))+
  geom_contour(color="black", binwidth = 0.1)+
  theme_minimal()+
  scale_fill_viridis(name="Growth Rate", na.value = "gray90",
                     limits = c(0,1.25))+
  facet_wrap(~TempNum, ncol = 5) +
  xlab("Stephensi Density") +
  ylab("Aegypti Density")
```

Stephensi

```
# only need female emergence
GrowthStephensi <- emergenceData %>%
  filter(Species =="Stephensi" & Sex == "Female") %>%
  #assume 50% female
  mutate(N0 = StDens/2) %>%
  rename(x = Day, Ax = Number)  %>%
  filter(StDens>0)

#add in predicted fecundity based on overall mean since model had no significance
GrowthStephensi$fwx <- mean(fecundity$Eggs[fecundity$Species == "Stephensi"],
                            na.rm = T)

#calculate overall per capita growth rate
GrowthStephensi <- GrowthStephensi %>%
  mutate(Axfwx = Ax*fwx, xAxfwx = x*Ax*fwx) %>%
  group_by(Replicate, Temp, Ratio, TempNum, AeDens, StDens) %>%
  mutate_at(c("Axfwx", "xAxfwx"), sum, na.rm=T) %>%
  mutate(D=14) %>%
  ungroup() %>%
  mutate(r=(log((1/N0)*Axfwx))/(D+(xAxfwx/Axfwx))) %>%
  group_by(Replicate, Temp, TempNum, Ratio, AeDens, StDens, Species) %>%
  #each row is now a duplicate of the same thing so only need the first one
  slice(1) %>%
  mutate(lambda=exp(r)) %>%
  ungroup() %>%
    #rescale predictor variables
  mutate(TempScale = as.vector(scale(TempNum, center = F, scale = T))) %>%
  mutate(AeDensScale = as.vector(scale(AeDens, center = F, scale = T))) %>%
  mutate(StDensScale = as.vector(scale(StDens, center = F, scale = T))) %>%
  #drop unneccesary columns
  dplyr::select(-x, -Ax, - Sex, - N0, -fwx, - Axfwx, - xAxfwx, -D)

# NAs are truly zeros
GrowthStephensi$lambda[is.na(GrowthStephensi$lambda)] <- 0
```

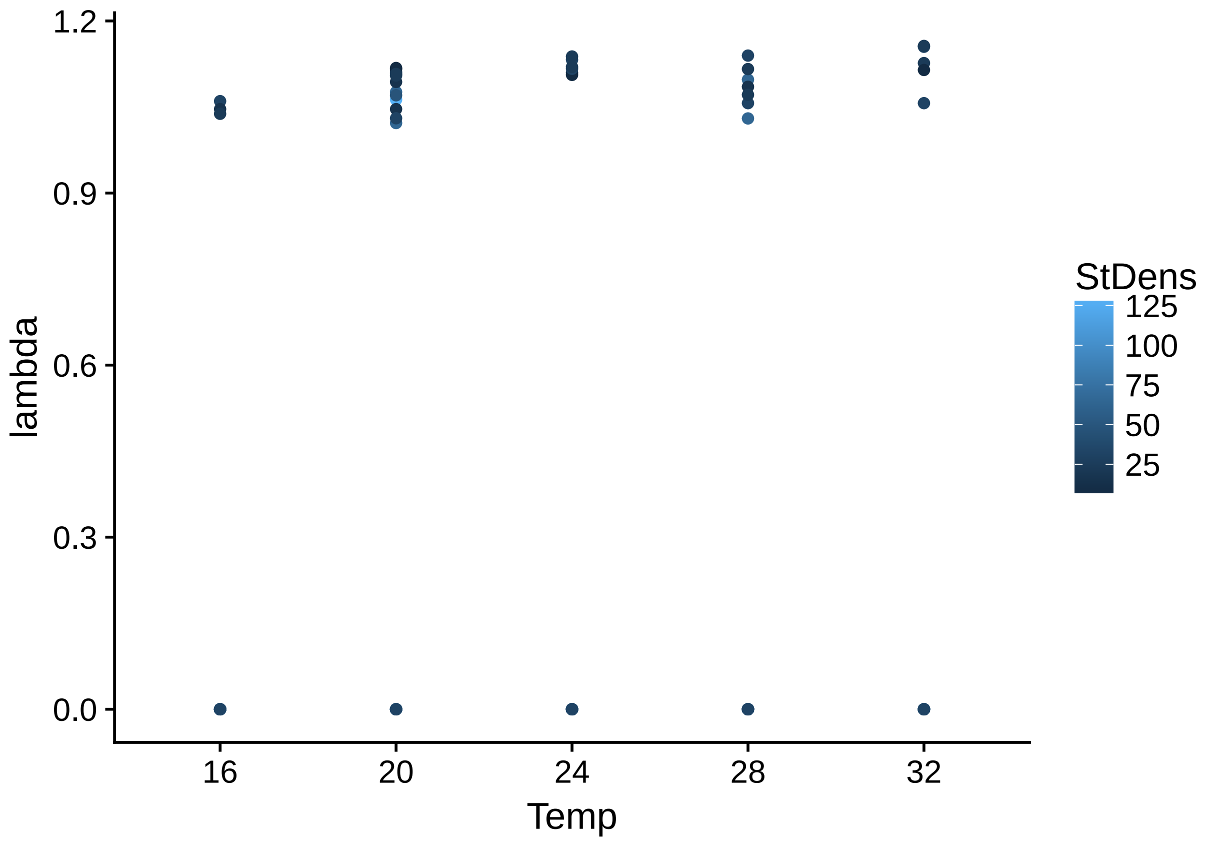
```
ggplot(data =  GrowthStephensi, aes(x = StDens, y = lambda, color = Temp)) +
  geom_point() +
  geom_smooth(se = F)
```

```
## `geom_smooth()` using method = 'loess'
```

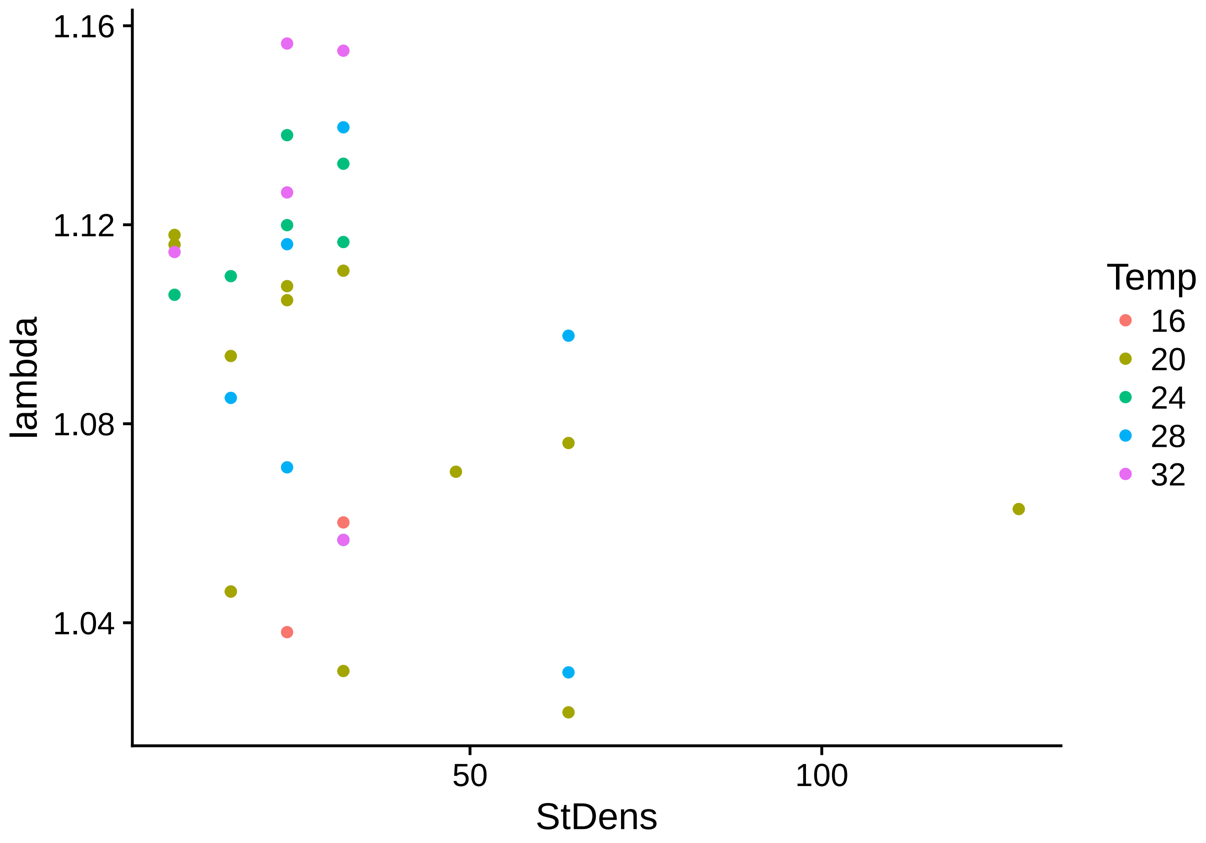


```r
ggplot(data =  GrowthStephensi, aes(x = Temp, y = lambda, color = StDens)) +
  geom_point() +
  geom_smooth(se = F)
```

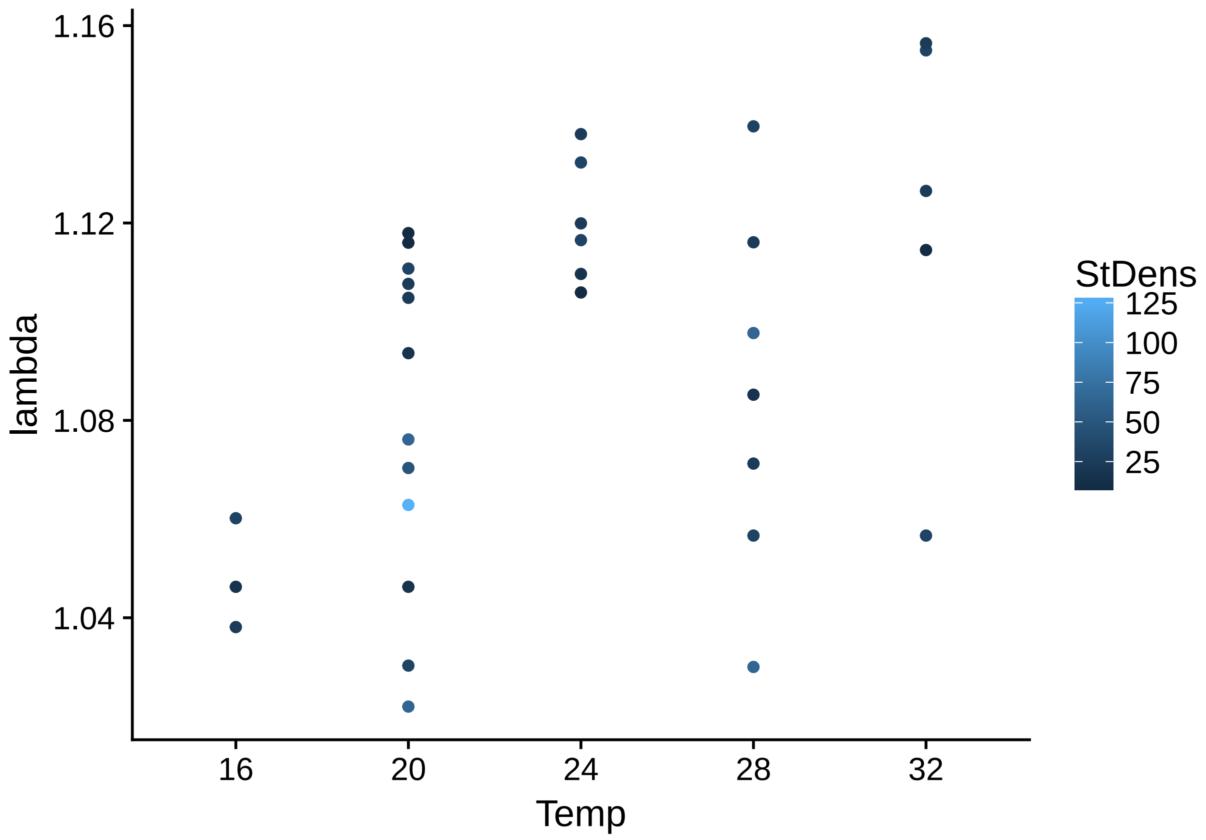
```
## `geom_smooth()` using method = 'loess'
```



```
ggplot(data =  GrowthStephensi[GrowthStephensi$lambda>0,],
       aes(x = StDens, y = lambda, color = Temp)) +
  geom_point()
```



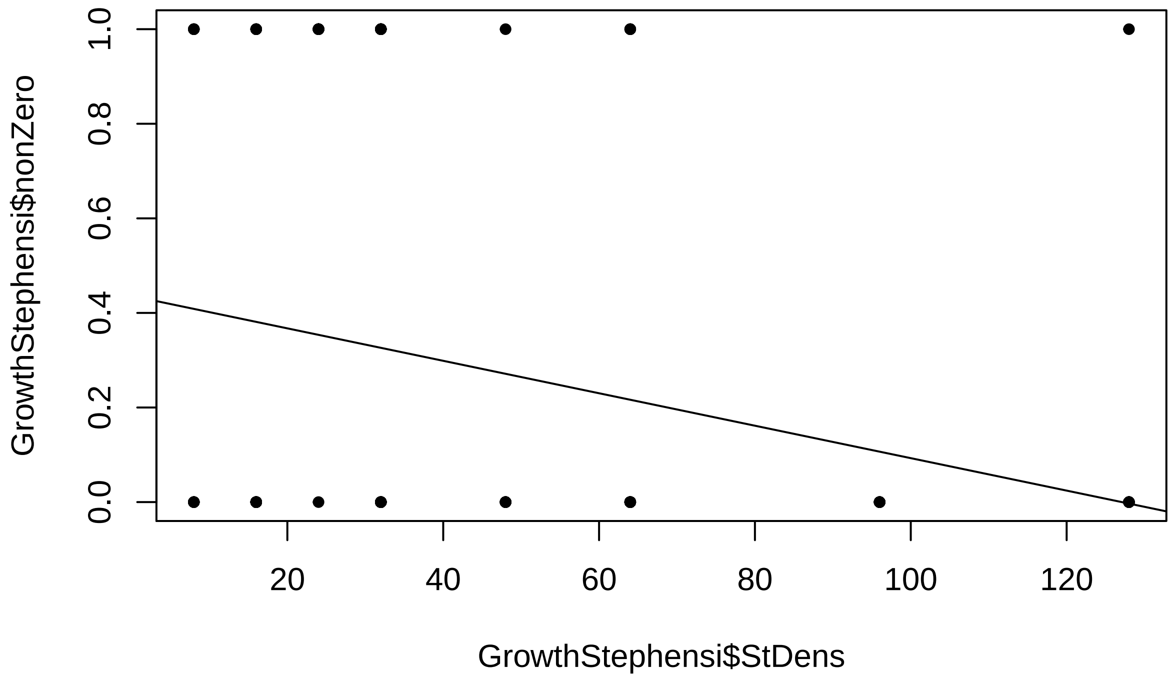
```
ggplot(data =  GrowthStephensi[GrowthStephensi$lambda>0,],
       aes(x = Temp, y = lambda, color = StDens)) +
  geom_point()
```



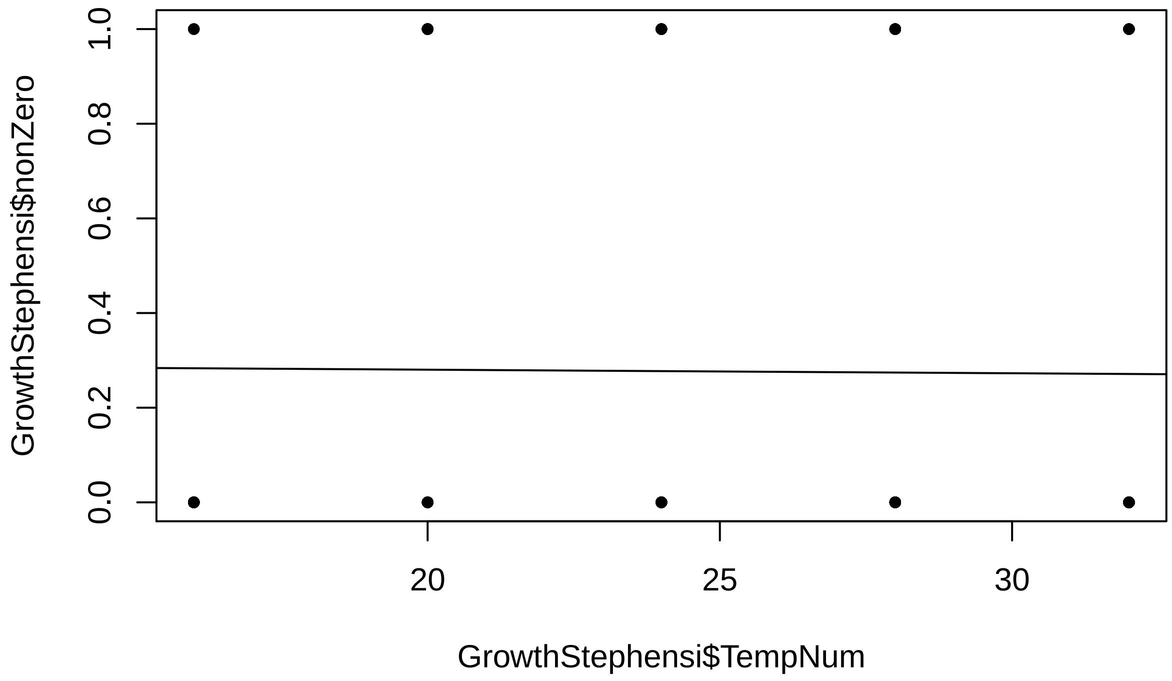
Model Selection

Impliment a hurdle model.

```r
#create nonzero variable
GrowthStephensi$nonZero <- ifelse(GrowthStephensi$lambda>0,1,0)

plot(GrowthStephensi$StDens, GrowthStephensi$nonZero, pch=20)
abline(lm(nonZero ~ StDens, data = GrowthStephensi))
```



```
plot(GrowthStephensi$TempNum, GrowthStephensi$nonZero, pch=20)
abline(lm(nonZero ~ TempNum, data = GrowthStephensi)) #temperature has no effect zero
```



Model selection for zero vs. nonzero growth

```
m0 <- glmer(nonZero ~ 1 + (1|Replicate),
            data = GrowthStephensi,
            family = binomial(link = logit))

m1 <- glmer(nonZero ~ TempScale + (1|Replicate),
            data = GrowthStephensi,
            family = binomial(link = logit))

m2 <- glmer(nonZero ~ AeDensScale + (1|Replicate),
            data = GrowthStephensi,
            family = binomial(link = logit))

m3 <- glmer(nonZero ~ StDensScale + (1|Replicate),
            data = GrowthStephensi,
            family = binomial(link = logit))

m4 <- glmer(nonZero ~ AeDensScale + StDensScale + (1|Replicate),
            data = GrowthStephensi,
            family = binomial(link =logit))

m5 <- glmer(nonZero ~ AeDensScale * StDensScale + (1|Replicate),
            data = GrowthStephensi,
            family = binomial(link = logit))
```

```
modelSummary <- cbind(model = c("m0","m1","m2","m3", "m4", "m5"),
                   do.call(rbind, lapply(list(m0, m1, m2, m3, m4, m5), broom::glance)),
                   AICc = AICc(m0, m1, m2, m3, m4, m5),
                   AICweights = Weights(AIC(m0, m1, m2, m3, m4, m5)))
modelSummary
```

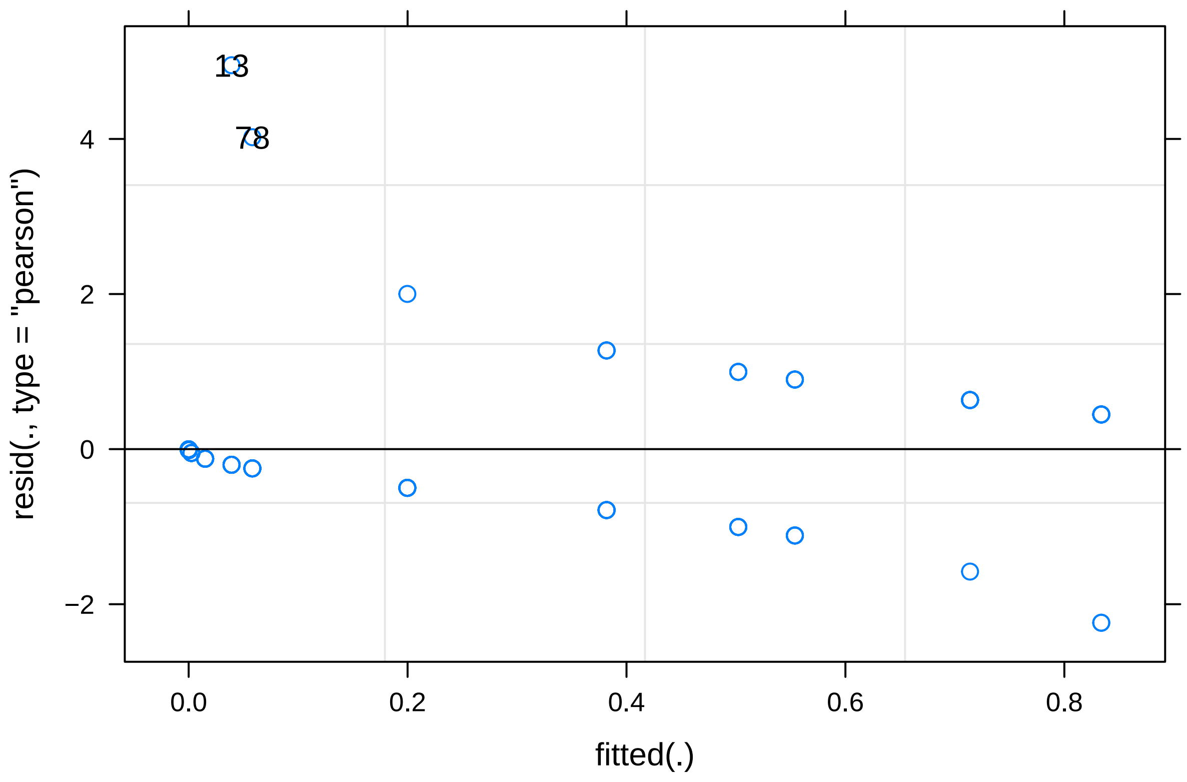
```
##    model sigma    logLik       AIC      BIC  deviance df.residual AICc.df
## m0    m0     1 -70.25708 144.51416 150.0724 140.51416         117       2
## m1    m1     1 -70.25163 146.50325 154.8406 140.50325         116       3
## m2    m2     1 -54.88365 115.76730 124.1047 109.76730         116       3
## m3    m3     1 -65.40196 136.80393 145.1413 130.80393         116       3
## m4    m4     1 -40.76960  89.53920 100.6557  81.53920         115       4
## m5    m5     1 -40.70169  91.40338 105.2990  81.40338         114       5
##    AICc.AICc   AICweights
## m0 144.61761 8.282498e-13
## m1 146.71195 3.063620e-13
## m2 115.97599 1.446982e-06
## m3 137.01262 3.912163e-11
## m4  89.89008 7.174979e-01
## m5  91.93436 2.825007e-01
```

```
StephensiGrowthModelBin <- m4
```

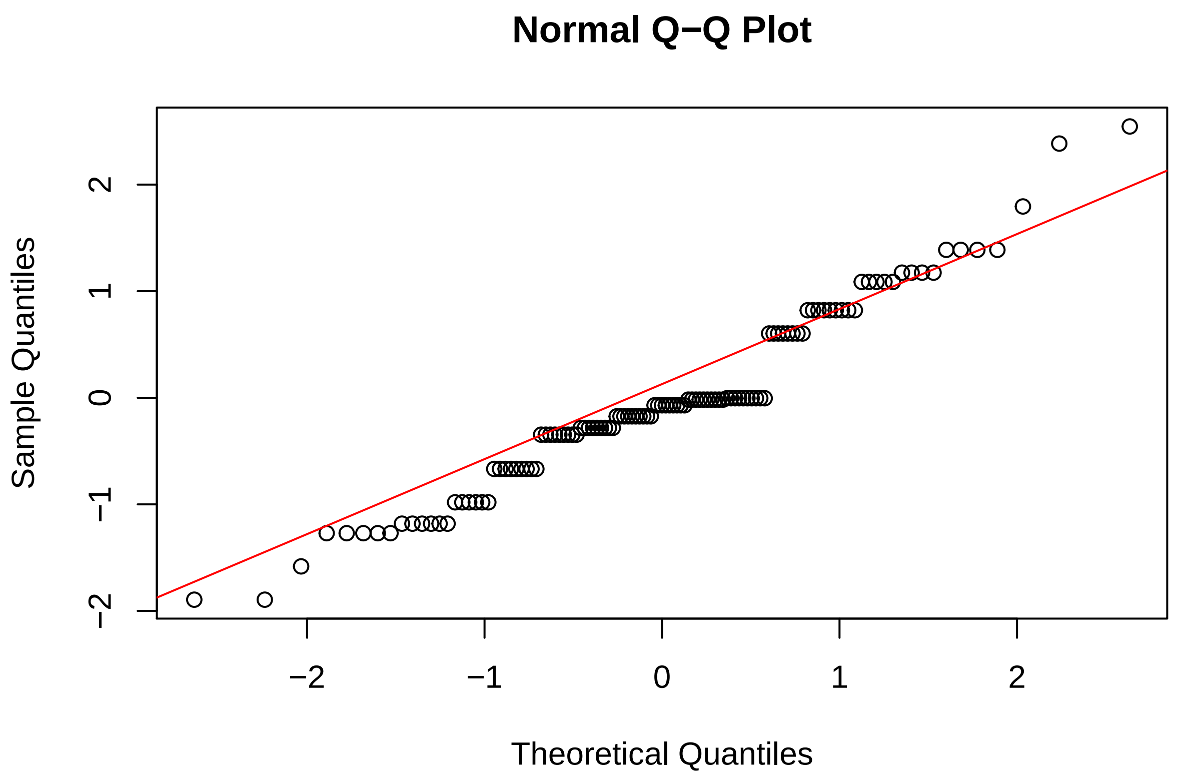
Whether or not growth is zero (basically mosquitoes emerge), is dependent on density (Aedes and Stephensi), not temperature.

Inspect residuals, although this is more important for final hurdle model.

```
plot(StephensiGrowthModelBin, id = 0.01, idLabels=~.obs)
```

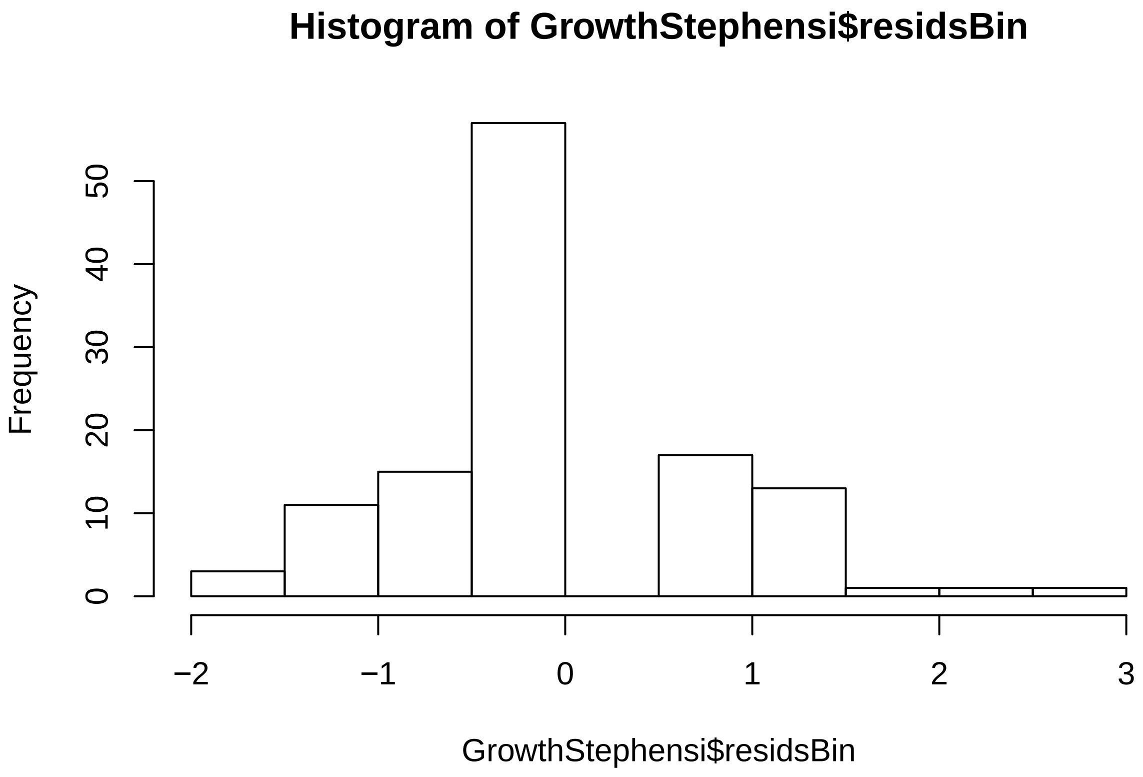


```
qqnorm(resid(StephensiGrowthModelBin))
qqline(resid(StephensiGrowthModelBin), col = "red")
```

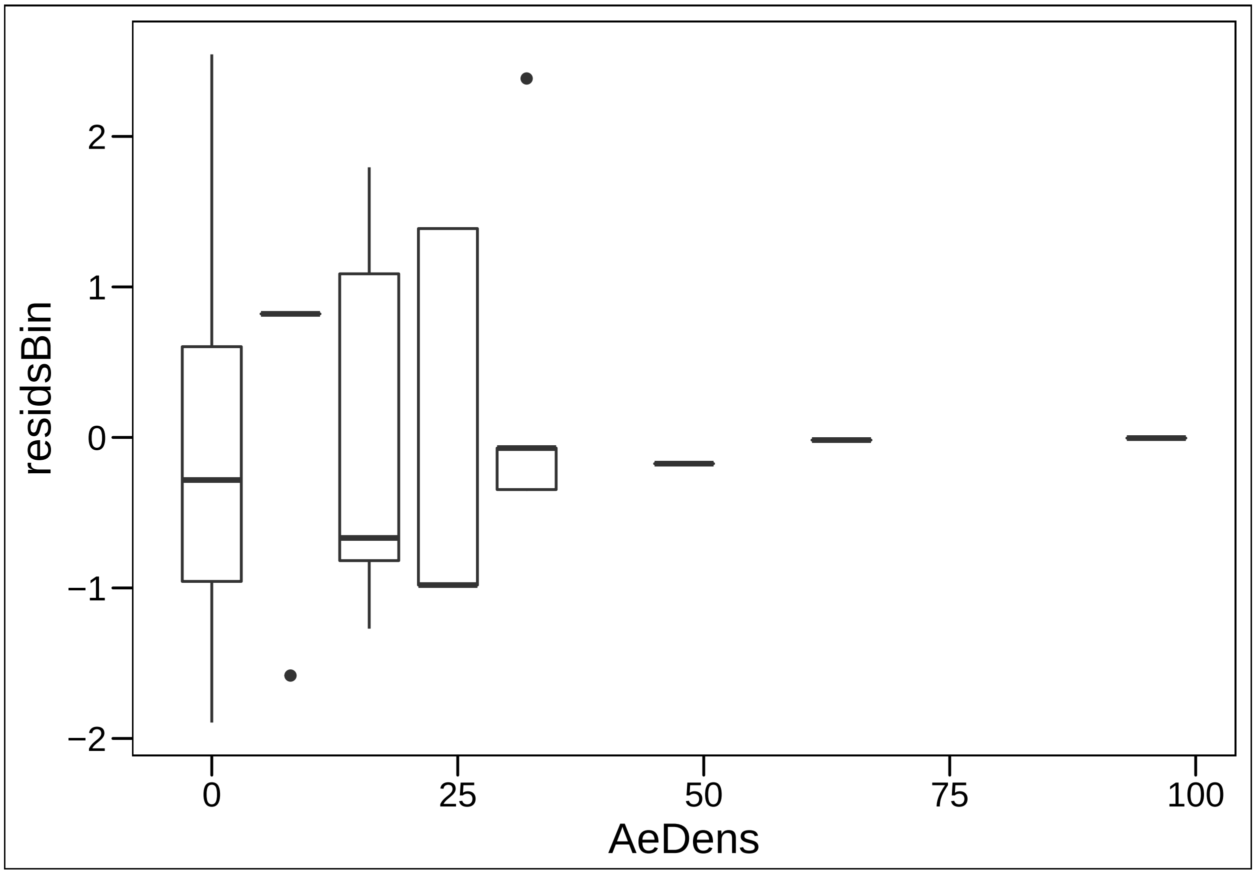


```
GrowthStephensi$predsBin <- predict(StephensiGrowthModelBin)
GrowthStephensi$residsBin <- resid(StephensiGrowthModelBin)
```

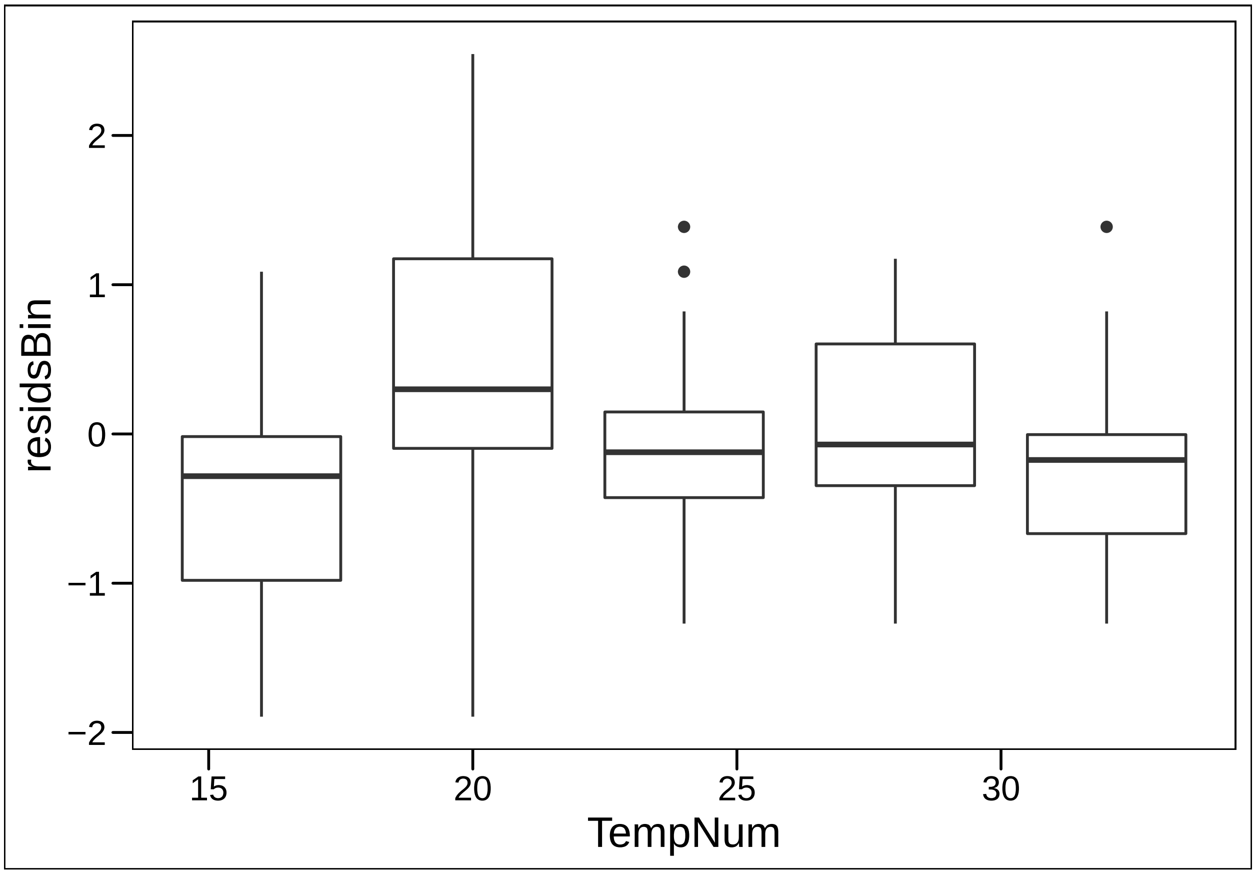
```r
hist(GrowthStephensi$residsBin)
```



```r
ggplot(data = GrowthStephensi, aes(x=AeDens, y = residsBin, group = AeDens))+
  geom_boxplot() +
  theme_base()
```



```
ggplot(data = GrowthStephensi, aes(x=TempNum, y = residsBin, group = TempNum))+
  geom_boxplot() +
  theme_base()
```



Second part of hurdle model for non-zeros

```
m0 <- glmer(lambda ~ 1 + (1|Replicate),
            data = subset(GrowthStephensi, nonZero==1),
            family = gaussian)
```

```
## Warning in glmer(lambda ~ 1 + (1 | Replicate), data =
## subset(GrowthStephensi, : calling glmer() with family=gaussian (identity
## link) as a shortcut to lmer() is deprecated; please call lmer() directly
```

```
m1 <- lmerTest::lmer(lambda ~ TempScale + (1|Replicate),
            data = subset(GrowthStephensi, nonZero==1))

m2 <- glmer(lambda ~ AeDensScale + (1|Replicate),
            data = subset(GrowthStephensi, nonZero==1),
            family = gaussian)
```

```
## Warning in glmer(lambda ~ AeDensScale + (1 | Replicate), data =
## subset(GrowthStephensi, : calling glmer() with family=gaussian (identity
## link) as a shortcut to lmer() is deprecated; please call lmer() directly
```

```
m3 <- glmer(lambda ~ StDensScale + (1|Replicate),
            data = subset(GrowthStephensi, nonZero==1),
            family = gaussian)
```

```
## Warning in glmer(lambda ~ StDensScale + (1 | Replicate), data =
## subset(GrowthStephensi, : calling glmer() with family=gaussian (identity
## link) as a shortcut to lmer() is deprecated; please call lmer() directly
```

```
m4 <- glmer(lambda ~ StDensScale + TempScale + (1|Replicate),
            data = subset(GrowthStephensi, nonZero==1),
            family = gaussian)
```

```
## Warning in glmer(lambda ~ StDensScale + TempScale + (1 | Replicate), data
## = subset(GrowthStephensi, : calling glmer() with family=gaussian (identity
## link) as a shortcut to lmer() is deprecated; please call lmer() directly
```

```
m5 <-  glmer(lambda ~ StDensScale * TempScale + (1|Replicate),
            data = subset(GrowthStephensi, nonZero==1),
            family = gaussian)
```

```
## Warning in glmer(lambda ~ StDensScale * TempScale + (1 | Replicate), data
## = subset(GrowthStephensi, : calling glmer() with family=gaussian (identity
## link) as a shortcut to lmer() is deprecated; please call lmer() directly
```

```
m6 <- glmer(lambda ~ StDensScale + poly(TempScale,2) + (1|Replicate),
            data = subset(GrowthStephensi, nonZero==1),
            family = gaussian)
```

```
## Warning in glmer(lambda ~ StDensScale + poly(TempScale, 2) + (1 |
## Replicate), : calling glmer() with family=gaussian (identity link) as a
## shortcut to lmer() is deprecated; please call lmer() directly
```

```
modelSummary <- cbind(model = c("m0","m1","m2","m3", "m4", "m5", "m6"),
                      do.call(rbind, lapply(list(m0, m1, m2, m3, m4, m5, m6), broom::glance)),
                      AICc = AICc(m0, m1, m2, m3, m4, m5, m6),
                      AICweights = Weights(AIC(m0, m1, m2, m3, m4, m5, m6)))
```

```
## Warning in deviance.merMod(x): deviance() is deprecated for REML fits;
## use REMLcrit for the REML criterion or deviance(.,REML=FALSE) for deviance
## calculated at the REML fit
```

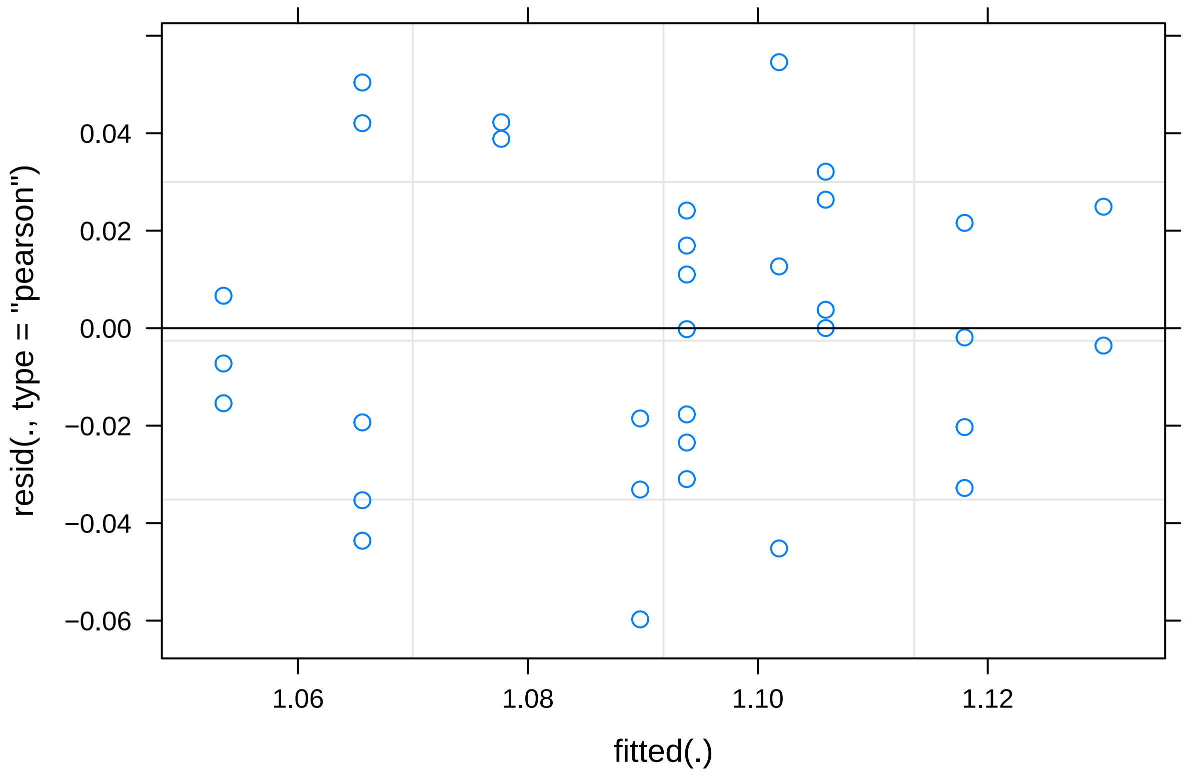
```
modelSummary
```

```
##    model      sigma   logLik       AIC        BIC  deviance df.residual
## m0    m0 0.03405524 59.94293 -113.8859 -109.39634 -126.2077          30
## m1    m1 0.03092073 60.80450 -113.6090 -107.62297 -121.6090          29
## m2    m2 0.03461804 57.21009 -106.4202 -100.43416 -126.2387          29
## m3    m3 0.03075010 60.10393 -112.2079 -106.22184 -133.1257          29
## m4    m4 0.02778850 60.90507 -111.8101 -104.32760 -140.7422          28
## m5    m5 0.02800588 59.68014 -107.3603  -98.38123 -141.4462          27
## m6    m6 0.02822665 58.53774 -105.0755  -96.09643 -141.0352          27
##    AICc.df AICc.AICc  AICweights
## m0       3 -113.0583 0.366100165
## m1       4 -112.1804 0.318771044
## m2       4 -104.9916 0.008758874
## m3       4 -110.7793 0.158207410
## m4       5 -109.5879 0.129676280
## m5       6 -104.1295 0.014014773
## m6       6 -101.8447 0.004471453
```

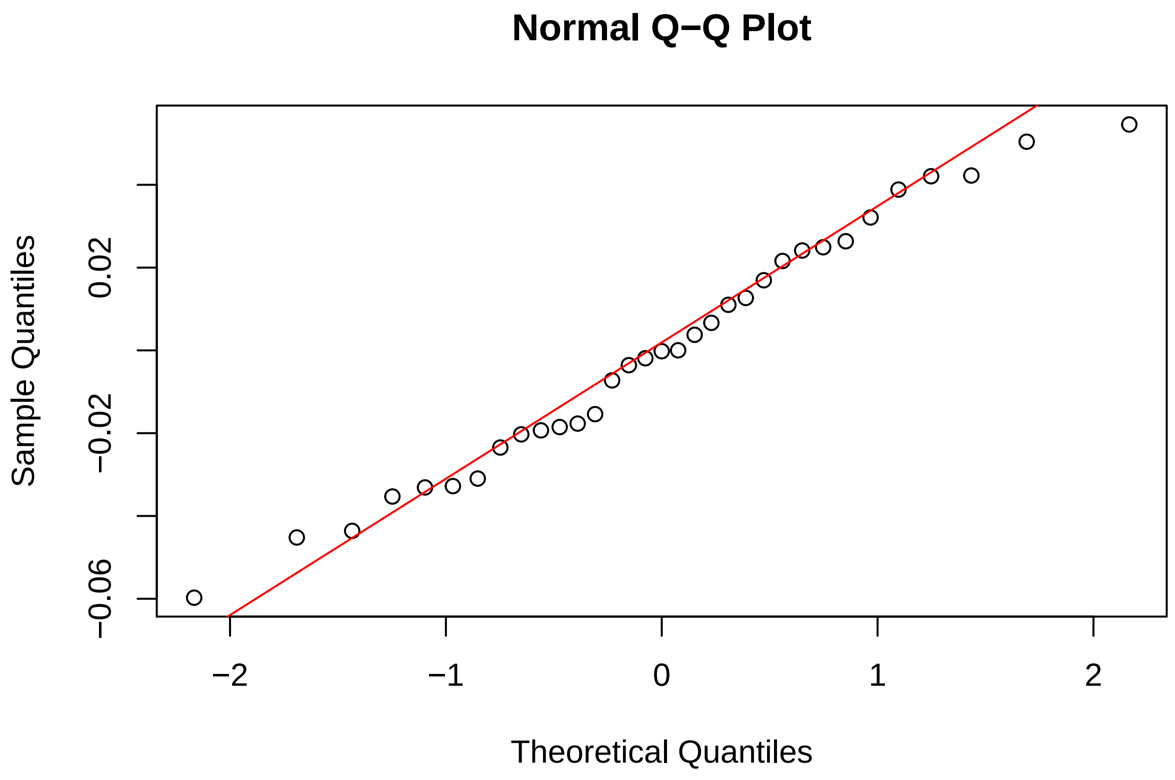
```
StephensiGrowthModel2 <- m1
```

Best Model is temperature (slightly positive).

```
plot(StephensiGrowthModel2, id = 0.01, idLabels=~.obs)
```



```r
qqnorm(resid(StephensiGrowthModel2))
qqline(resid(StephensiGrowthModel2), col = "red")
```



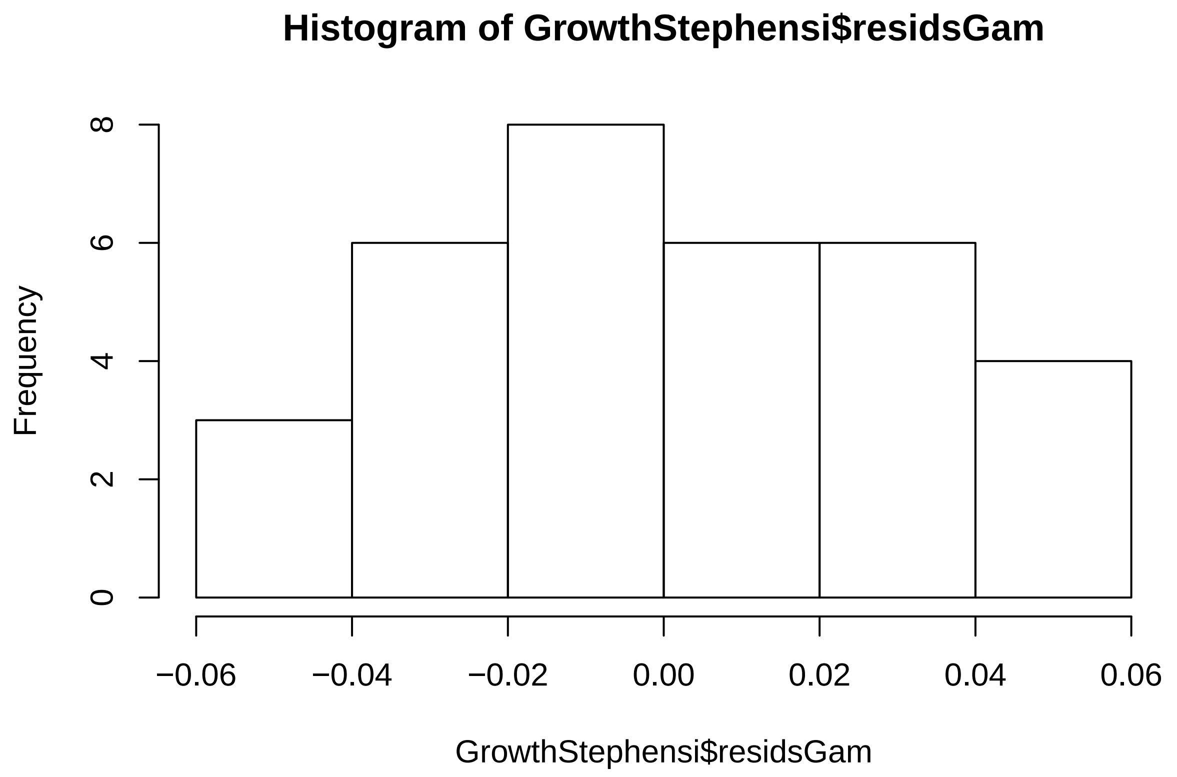
```
GrowthStephensi$predsGam[GrowthStephensi$nonZero==1] <- predict(StephensiGrowthModel2)
```

```
## Warning: Unknown or uninitialised column: 'predsGam'.
```

```
GrowthStephensi$residsGam[GrowthStephensi$nonZero==1] <- resid(StephensiGrowthModel2)
```

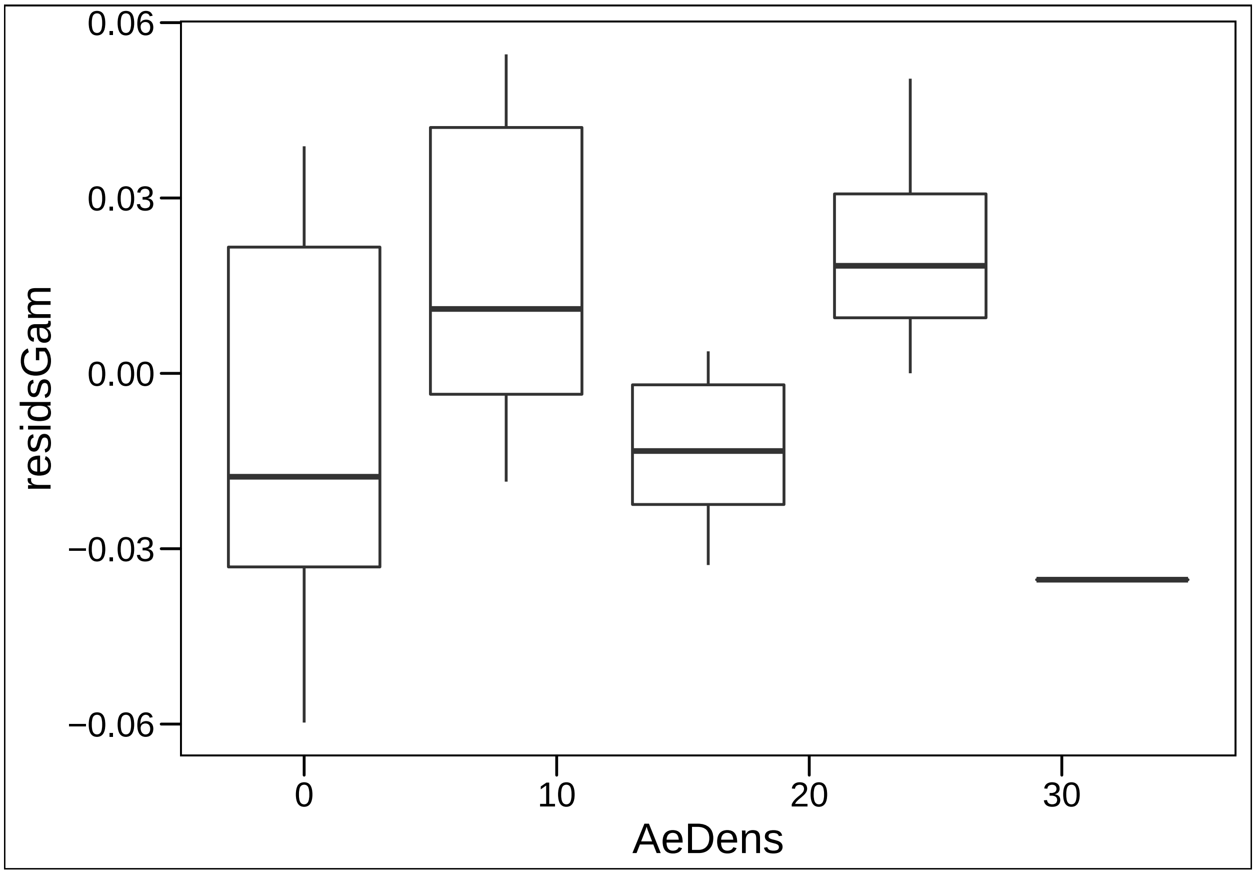
```
## Warning: Unknown or uninitialised column: 'residsGam'.
```

```
hist(GrowthStephensi$residsGam)
```



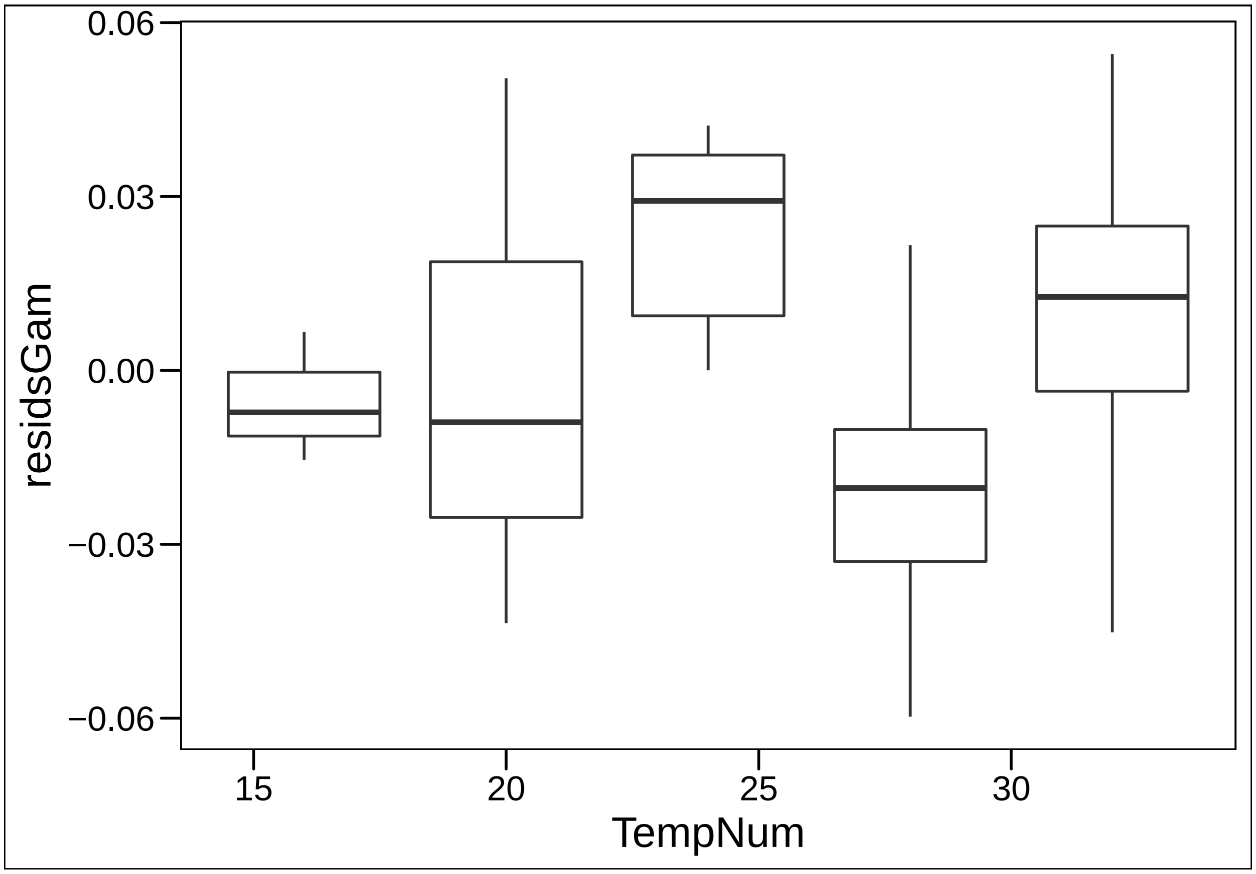
```
ggplot(data = GrowthStephensi, aes(x=AeDens, y = residsGam, group = AeDens))+
  geom_boxplot() +
  theme_base()
```

```
## Warning: Removed 86 rows containing non-finite values (stat_boxplot).
```



```
ggplot(data = GrowthStephensi, aes(x=TempNum, y = residsGam, group = TempNum))+
  geom_boxplot() +
  theme_base()
```

```
## Warning: Removed 86 rows containing non-finite values (stat_boxplot).
```



```r
summary(StephensiGrowthModel2)
```

```
## Linear mixed model fit by REML t-tests use Satterthwaite approximations
##   to degrees of freedom [lmerMod]
## Formula: lambda ~ TempScale + (1 | Replicate)
##    Data: subset(GrowthStephensi, nonZero == 1)
##
## REML criterion at convergence: -121.6
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.93201 -0.65619 -0.00646  0.78043  1.76496
##
## Random effects:
##  Groups    Name        Variance  Std.Dev.
##  Replicate (Intercept) 0.0004499 0.02121
##  Residual              0.0009561 0.03092
## Number of obs: 33, groups:  Replicate, 2
##
## Fixed effects:
##             Estimate Std. Error       df t value Pr(>|t|)
## (Intercept)  1.01929    0.03042 10.73300  33.512 3.29e-12 ***
## TempScale    0.07472    0.02684 30.03800   2.783  0.00921 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Correlation of Fixed Effects:
##           (Intr)
## TempScale -0.852
```

```
confint(StephensiGrowthModel2)
```

```
## Computing profile confidence intervals ...

##                  2.5 %     97.5 %
## .sig01      0.00000000 0.06626058
## .sigma      0.02417192 0.03990503
## (Intercept) 0.96259638 1.07576942
## TempScale   0.02205714 0.12898464
```

```
unHurdle <- function(mod1, mod2, newDF){
  #' Function that creates predictions based on binomial/guassian hurdle model

  #' @param mod1 binomial model
  #' @param mod2 second model, gaussian in this case
  #' @param newDF new data frame to predict over, must have same columns as original
  phi_zero <- predict(mod1, newdata = newDF, type = "response")
  mod2Preds <- predict(mod2, newdata = newDF, type = "response")
  #calculate probability of zero in second model to subtract from first
  phi_count <- pnorm(q = 0,
                     mean = mod2Preds,
                     sd = sigma(mod2),
                     log.p = F)
  phi <- phi_zero - phi_count
  predsAll <- phi * mod2Preds
  return(predsAll)
}

predsAll <- unHurdle(mod1 = StephensiGrowthModelBin, mod2 = StephensiGrowthModel2, newDF = GrowthStephe

GrowthStephensi$predsFinal <- predsAll
```

```
#heatmap plots
newData <- expand.grid(AeDens=seq(0,128, by=2), StDens=seq(0,128, by=2), TempNum=c(16,20,24,28,32), Rep
newData <- newData %>%
  mutate(TempScale = as.vector(scale(TempNum, center = F, scale = T))) %>%
  mutate(AeDensScale = as.vector(scale(AeDens, center = F, scale = T))) %>%
  mutate(StDensScale = as.vector(scale(StDens, center = F, scale = T)))

newData$preds <- unHurdle(mod1 = StephensiGrowthModelBin, mod2 = StephensiGrowthModel2, newDF = newData
#get means over replicates to plot
predicted <- newData %>%
  group_by(TempNum, AeDens, StDens) %>%
  summarise(preds = mean(preds))
#remove extrapolation outside of measured range
predicted$preds[predicted$AeDens+predicted$StDens>128] <- NA

stGrowthPreds <- predicted

#plot it
stGrowthPlot <- ggplot(stGrowthPreds, aes(x=StDens, y=AeDens,
                                          z=preds))+
  geom_raster(aes(fill=preds))+
  geom_contour(color="gray90", binwidth = 0.1)+
  theme_minimal()+
  scale_fill_viridis(name="Growth Rate", na.value = "gray90",
                     limits = c(0,1.25))+
  facet_wrap(~TempNum, ncol = 5) +
  xlab("Stephensi Density") +
  ylab("Aegypti Density")
```

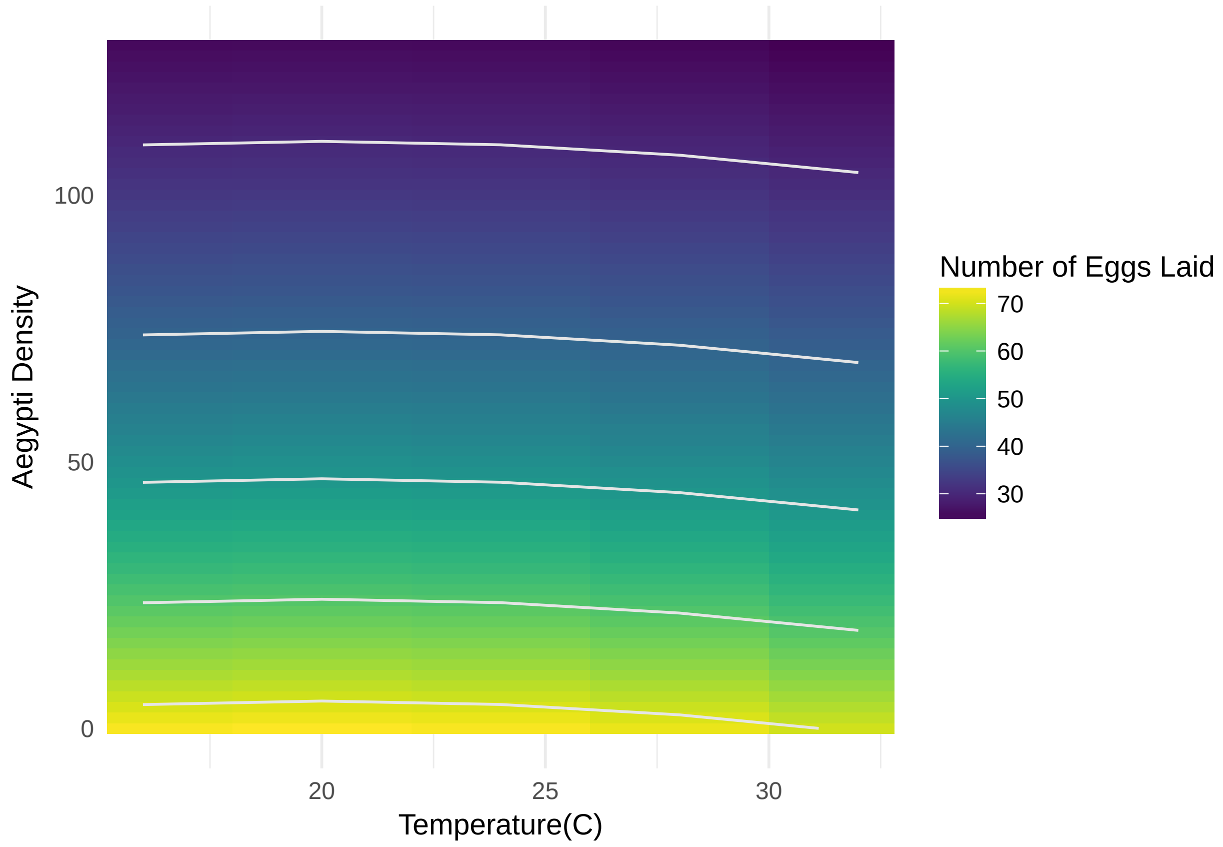
Save All Results

```
#models
save(AedesSurvivalModel, AedesFecundityModel, AedesGrowthModel,
     StephSurvivalModel, StephensiFecundityModel, StephensiGrowthModelBin,
     StephensiGrowthModel2, file = "code/finalModels.RData")
```

```
survivalPlot <- plot_grid(aeSurvPlot,
          stSurvPlot,
          nrow = 2,
          labels=c("A", "B"))
```

```
## Warning: Removed 10400 rows containing non-finite values (stat_contour).

## Warning: Removed 10400 rows containing non-finite values (stat_contour).
```

```
aeFecPlot
```



```
growthPlot <- plot_grid(aeGrowthPlot, stGrowthPlot, nrow = 2, labels = c("A", "B"))
```

```
## Warning: Removed 10400 rows containing non-finite values (stat_contour).

## Warning: Removed 10400 rows containing non-finite values (stat_contour).
```

```
save(survivalPlot, aeFecPlot, growthPlot, file = "code/figures/finalFigures.RData")
```