FMWT\_SMSCG

Rosemary Hartman

DRAFT

7/12/2019

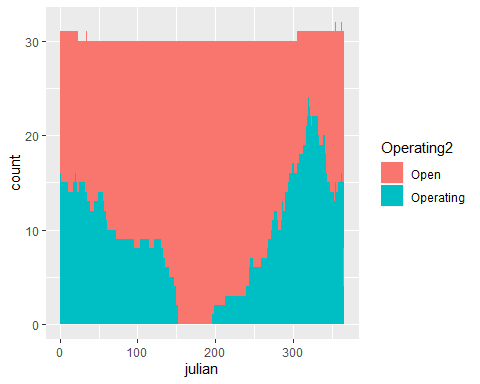
In the Delta Smelt Resiliancy strategy, there is the idea we can change the operation of the Suisun Marsh Salinity control gates to improve habitat in the marsh for Delta Smelt. I was curious if there were any trends between historical gate operations and presence of Delta Smelt in the Marsh. Therefore, I decided to compare catch of Delta Smelt in the marsh during the fall (when the gates are most frequently operated) with gate operations.

First I did some data manipulation to get the gate operation data lined up with the Delta Smelt Catch from the Fall Midwater Trawl, and I calculated CPUE.

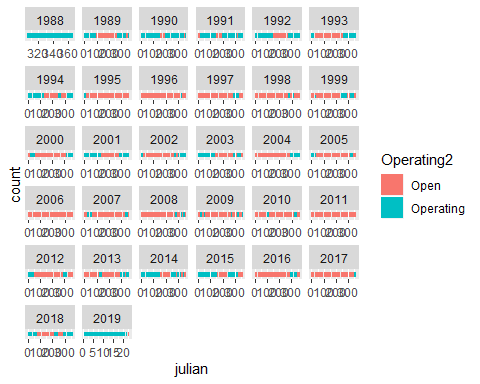
FMWT data is avaialable here: <ftp://ftp.wildlife.ca.gov/TownetFallMidwaterTrawl/FMWT%20Data/> Michael Koohafman gave me the gate operation data.

First for some quick exploritory plots of the data. Look at when the gates are usually operated and what the fish catch was like when they are or are not operated.

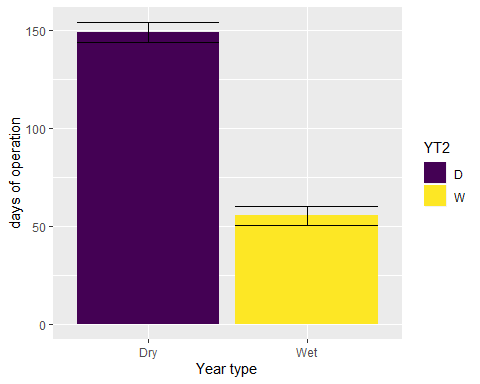
#when are the gates operated?  
ggplot(op.daily, aes(x=julian, fill = Operating2)) + geom\_bar(stat = "Count")



#look at it a different way  
 ggplot(op.daily, aes(x=julian, fill = Operating2)) + geom\_bar(stat = "Count", position = "fill") +  
 facet\_wrap(.~Year, scales = "free") + scale\_y\_discrete(breaks = NULL)

 How does gate operation relate to water year type? I’ll make a plot and run a basic model to see whether gates are operated more frequently in dry years. I’ve binned the five year types into “wet” and “dry”

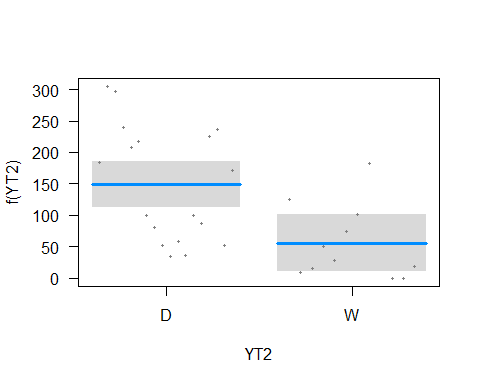
ops = filter(merge(op.daily, yrtyp), Year >1988, Year <2019)  
ops$Operating2 = as.factor(ops$Operating2)  
  
#summarize number of days per year the gates were operated, grouping by year type  
opsum = group\_by(ops, Year, YT2) %>% summarize(ops = length(Operating2[which(Operating2 == "Operating")]))  
opsmean = group\_by(opsum, YT2) %>% summarize(mean = mean(ops), se = sd(ops)/length(ops), sd = sd(ops))  
  
#plot it  
ggplot(opsmean, aes(x=YT2, y= mean, fill = YT2)) + geom\_bar(stat = "identity") +  
 geom\_errorbar(aes(ymin = mean - se, ymax = mean + se)) + ylab("days of operation") +  
 scale\_x\_discrete(name = "Year type", labels = c("Dry", "Wet"))



#It looks like they are operated more often in dry years, but I'll stats it to check.  
opmod2 = glm(ops~ YT2, data = opsum)  
summary(opmod2)

##   
## Call:  
## glm(formula = ops ~ YT2, data = opsum)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -115.17 -55.50 -15.50 66.33 155.83   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 102.33 14.92 6.859 1.88e-07 \*\*\*  
## YT2.L -66.23 21.10 -3.139 0.00397 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 6411.196)  
##   
## Null deviance: 242682 on 29 degrees of freedom  
## Residual deviance: 179514 on 28 degrees of freedom  
## AIC: 352.04  
##   
## Number of Fisher Scoring iterations: 2

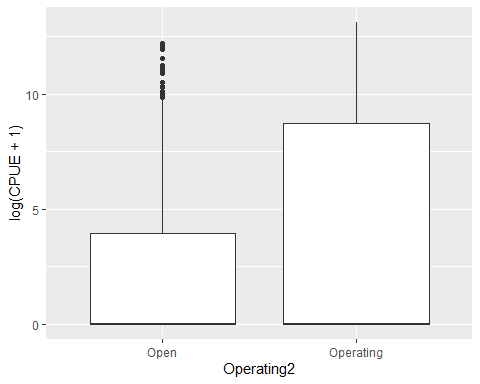
visreg(opmod2)



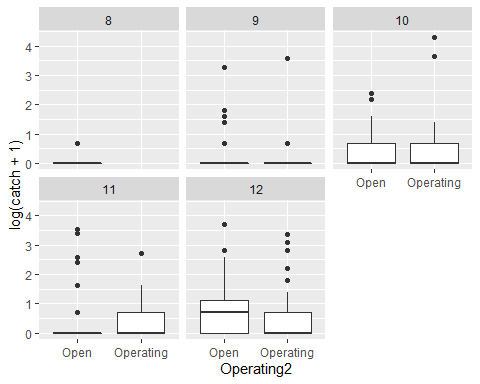
#Yup. Definitely.

Now I"ll make some basic plots of fish catch. I’ll filter it to just the time period where we have gate data (1988-2011) and just the fall, since most of the FMWT data is Sep-Dec. I didn’t plot 2012-2018, because the catch was too low

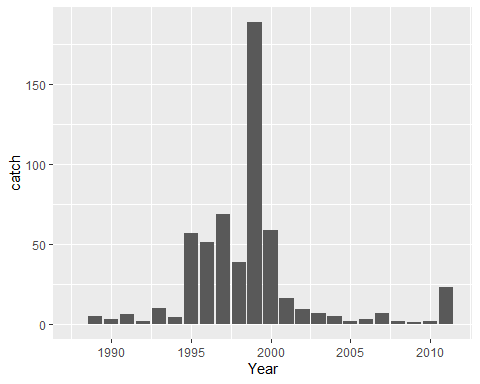
ggplot(FMWT\_DSmg2, aes(x = Operating2, y = log(CPUE+1))) + geom\_boxplot()



#seperate by month  
ggplot(FMWT\_DSmg2, aes(x = Operating2, y = log(catch+1))) + geom\_boxplot() + facet\_wrap(~month(Date))

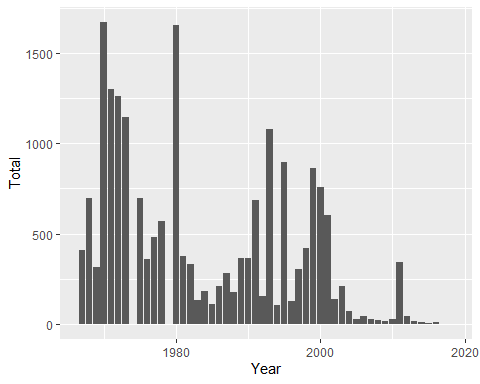


#Look at smelt catch in Montezuma Slough and the FMWT index overall - do they match up?  
ggplot(tots, aes(x=Year, y= catch)) + geom\_bar(stat = "identity")



ggplot(index, aes(x=Year, y= Total)) + geom\_bar(stat = "identity")

## Warning: Removed 2 rows containing missing values (position\_stack).



Let’s run some models to see whether there are statistically more smelt when the gates are operating. Other things are probably involved too, such as year, water year type, salinity, station, day of the year, etc. I’ll run several models and rank them with AICc to see which is best.

It took me a long time to figure out what type of model to run. Delta smelt catch data is “count data”, so theoretically it should follow a poisson distribution. However, my preliminary analysis showed it is highly overdisperssed and has WAY more zeros than a normal Poisson distribution. Therefore, after much discussion, research, statistics textbooks, and false starts, I settled on a zero-inflated negative binomial model.

I added the volume sampled as an offset, since I want count data rather than CPUE, but I still want to encorporate the volume’s potential effect on catch.

Now for some models. Some notes: - I have “scaled” all the predictor variables so that they have a mean of zero and a standard deviation of 1. This lets me compare between variables more easily. - I subset the data so that it is just dry years (below normal, dry, or critical) prior to 2011 - I binned the gate operation into “operating” or “not”, I did not make any distinction bewtween full and partial operation. - EC is the conductivity measured by CDFW’s boat, not the nearest WQ station. - SacX2 is from Hutton et al.  - If no volume of sample was available, I replaced it with the average.

#First I'll put in all the variables for a global model  
dsznb2 = zeroinfl(catch~ #catch of Delta Smelt  
 Station+ #Station on Montezuma Slough  
 Operating2+ #whether the gates are operating  
 julianscaled + #Day of the Year (scaled)  
 Indexscaled + #FMWT index (scaled)  
 SacX2 + #sacramento river X2  
 ECscaled + #Top electrical conductivity (scaled)  
 offset(Vol), #volume added as an offset  
 dist = "negbin", data = FMWT\_DSmg4a, na.action = "na.fail")  
  
#assess all possible models  
dreznb2 = dredge(dsznb2)

## Fixed terms are "count\_(Intercept)" and "zero\_(Intercept)"

## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced

## Warning in solve.default(as.matrix(fit$hessian)): system is computationally  
## singular: reciprocal condition number = 2.42208e-22 (model 3289 skipped)

## Warning in solve.default(as.matrix(fit$hessian)): system is computationally  
## singular: reciprocal condition number = 1.15231e-33 (model 5998 skipped)

## Warning in solve.default(as.matrix(fit$hessian)): system is computationally  
## singular: reciprocal condition number = 1.13191e-33 (model 6127 skipped)

## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced

## Warning in solve.default(as.matrix(fit$hessian)): system is computationally  
## singular: reciprocal condition number = 1.25441e-22 (model 14255 skipped)

## Warning in solve.default(as.matrix(fit$hessian)): system is computationally  
## singular: reciprocal condition number = 1.03427e-16 (model 14964 skipped)

## Warning in solve.default(as.matrix(fit$hessian)): system is computationally  
## singular: reciprocal condition number = 2.74569e-18 (model 15674 skipped)

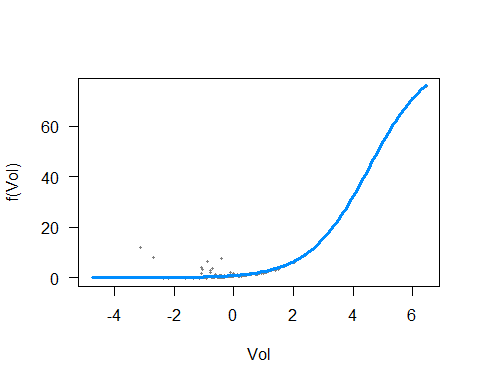
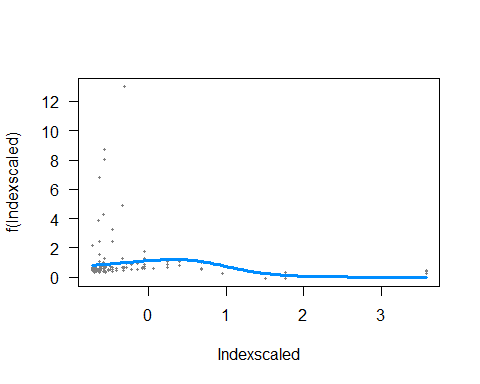
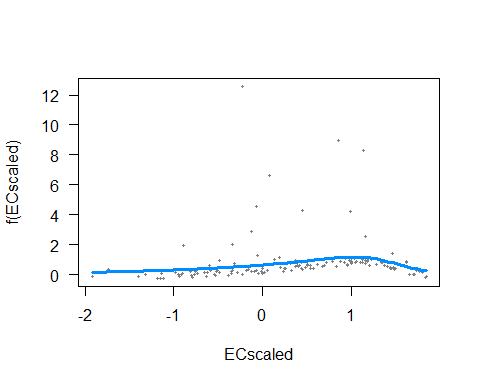
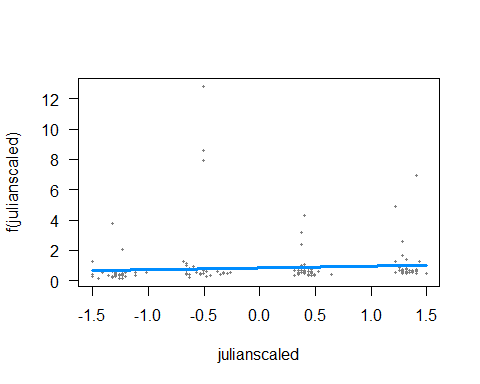
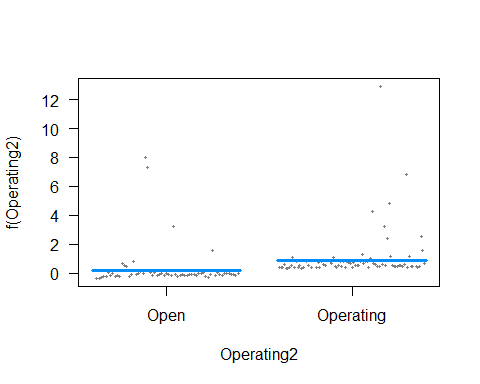
head(dreznb2)

## Global model call: zeroinfl(formula = catch ~ Station + Operating2 + julianscaled +   
## Indexscaled + SacX2 + ECscaled + offset(Vol), data = FMWT\_DSmg4a,   
## na.action = "na.fail", dist = "negbin")  
## ---  
## Model selection table   
## cnt\_(Int) cnt\_ECs cnt\_Ind cnt\_jln cnt\_Op2 cnt\_SX2 zer\_(Int) zer\_ECs  
## 1936 0.4612 -0.3451 0.2863 0.5272 + -77.36 -61.90  
## 10128 0.4609 -0.3445 0.2859 0.5267 + -44.72 -37.09  
## 9999 -0.1048 0.3337 0.5692 + -545.30   
## 4000 -0.8255 -0.4096 0.2866 0.6010 + 0.015530 -214.20 -49.28  
## 12192 -0.3411 -0.3710 0.2731 0.5404 + 0.009781 -103.30 -29.08  
## 3871 -2.2570 0.1499 0.3565 + 0.032090 -60.46   
## zer\_Ind zer\_jln zer\_Op2 zer\_SX2 zer\_off(Vol) df logLik AICc  
## 1936 -266.500 52.780 + 11 -105.001 233.9  
## 10128 -159.900 32.350 + + 11 -105.031 234.0  
## 9999 -1095.000 204.200 + + 9 -107.720 234.7  
## 4000 -72.610 85.710 + 3.1500 13 -104.225 237.1  
## 12192 -76.050 26.330 + 1.1890 + 13 -104.681 238.0  
## 3871 -2.671 3.416 + 0.7382 11 -107.404 238.7  
## delta weight  
## 1936 0.00 0.327  
## 10128 0.06 0.318  
## 9999 0.81 0.218  
## 4000 3.21 0.066  
## 12192 4.12 0.042  
## 3871 4.81 0.030  
## Models ranked by AICc(x)

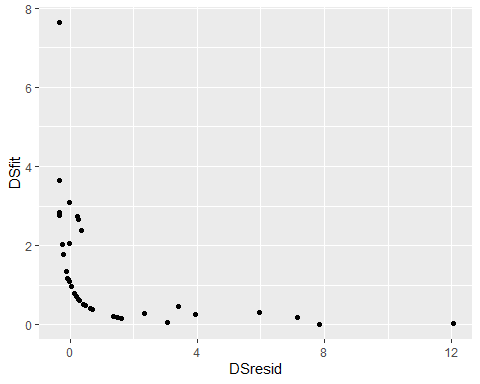
#this was the best model  
dsznb2best = zeroinfl(catch~ Operating2+julianscaled +   
 ECscaled + Indexscaled+ offset(Vol),  
 dist = "negbin", data = FMWT\_DSmg4a, na.action = "na.fail")  
summary(dsznb2best)

##   
## Call:  
## zeroinfl(formula = catch ~ Operating2 + julianscaled + ECscaled +   
## Indexscaled + offset(Vol), data = FMWT\_DSmg4a, na.action = "na.fail",   
## dist = "negbin")  
##   
## Pearson residuals:  
## Min 1Q Median 3Q Max   
## -0.54488 -0.40002 -0.28894 -0.08953 12.04291   
##   
## Count model coefficients (negbin with log link):  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.7766 0.5525 -3.215 0.0013 \*\*  
## Operating2Operating 1.6324 0.8035 2.032 0.0422 \*   
## julianscaled 0.1278 0.3576 0.357 0.7208   
## ECscaled 0.7481 0.3531 2.118 0.0341 \*   
## Indexscaled 0.5386 0.2304 2.338 0.0194 \*   
## Log(theta) -1.1137 0.3386 -3.289 0.0010 \*\*  
##   
## Zero-inflation model coefficients (binomial with logit link):  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -30.7804 150.8560 -0.204 0.83832   
## Operating2Operating 26.2194 150.8019 0.174 0.86197   
## julianscaled -0.9168 0.7731 -1.186 0.23568   
## ECscaled 4.8677 1.7897 2.720 0.00653 \*\*  
## Indexscaled 3.3218 1.5272 2.175 0.02962 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
##   
## Theta = 0.3283   
## Number of iterations in BFGS optimization: 71   
## Log-likelihood: -126.8 on 11 Df

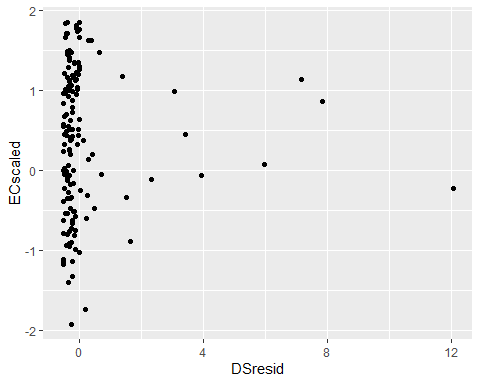
#I can make some conditional plots to see the effect of each variable by itself.  
visreg(dsznb2best)

 So that model looks OK, but we should check the residuals to make sure there is no pattern. According to Zuur et al. 2009, I should plot the pearson residuals versus the fitted values and versus the predictor variables and see no pattern.

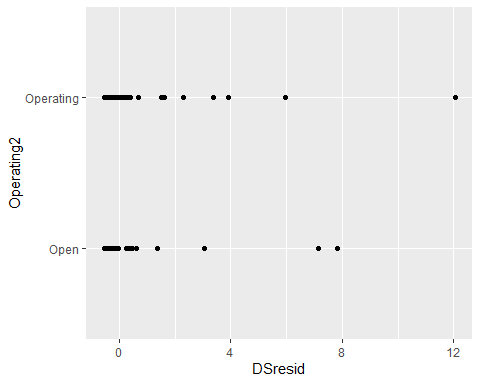
tests = data.frame(DSresid = residuals(dsznb2best, type = "pearson"),   
 DSfit = dsznb2best$fitted.values,  
 ops = dsznb2best$model["Operating2"],  
 EC = dsznb2best$model["ECscaled"],  
 Ind = dsznb2best$model["Indexscaled"],  
 day = dsznb2best$model["julianscaled"],  
 catch = FMWT\_DSmg4a$catch)  
  
  
ggplot(data = filter(tests, catch !=0), aes(x=DSresid, y = DSfit)) + geom\_point()



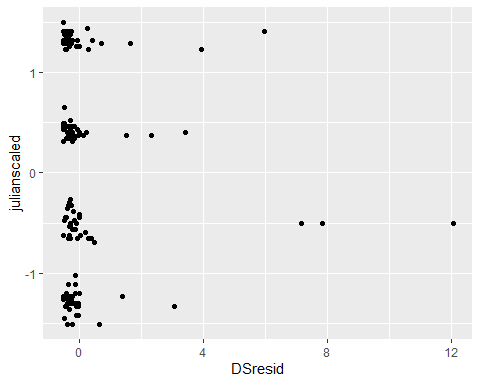
ggplot(data = tests, aes(x=DSresid, y = ECscaled)) + geom\_point()



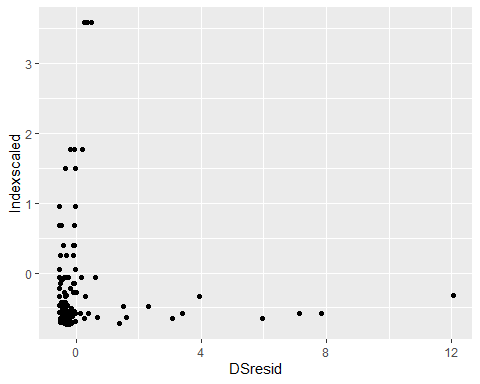
ggplot(data = tests, aes(x=DSresid, y = Operating2)) + geom\_point()



ggplot(data = tests, aes(x=DSresid, y = julianscaled)) + geom\_point()

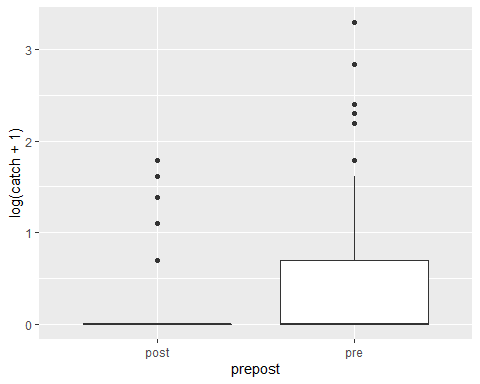


ggplot(data = tests, aes(x=DSresid, y = Indexscaled)) + geom\_point()

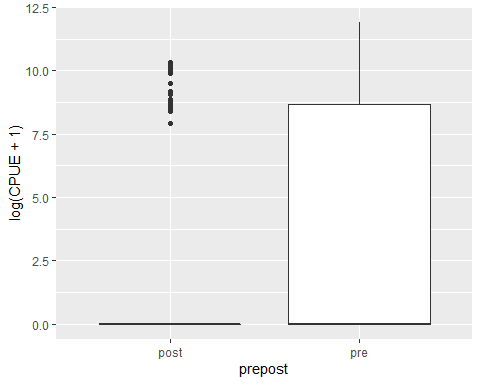
 If we’ve been operating the gates whenever it gets salty in the marsh, It would be ideal to see what would have happened to smelt catch in a dry year in the marsh where we didn’t operate the gates.

Fortunately, we have dtata from FMWT before the gates went into operation.

###########################################################################################  
#can we compare pre-gate data to post-gate data?  
  
#first make a new variable for pre-gate versus post-gate  
FMWT\_DSmg4ax$prepost = NA  
FMWT\_DSmg4ax$prepost[which(FMWT\_DSmg4ax$Year <1988)] = "pre"  
FMWT\_DSmg4ax$prepost[which(is.na(FMWT\_DSmg4ax$prepost))] = "post"  
  
#Take out the couple of places were we didn't have X2 values  
FMWT\_DSmg4ax = filter(FMWT\_DSmg4ax,!is.na(SacX2))  
  
#quick plot of smelt abundacne pre versus post  
ggplot(FMWT\_DSmg4ax, aes(x=prepost, y = log(catch+1))) + geom\_boxplot()



ggplot(FMWT\_DSmg4ax, aes(x=prepost, y = log(CPUE+1))) + geom\_boxplot()



#run a model with all the possible variables  
dsznb3 = zeroinfl(catch~Station + prepost + julianscaled +   
 ECscaled +SacX2 + Indexscaled + offset(Vol), dist = "negbin",   
 data = FMWT\_DSmg4ax,na.action = "na.fail")  
  
#What does this one look like?  
summary(dsznb3)

##   
## Call:  
## zeroinfl(formula = catch ~ Station + prepost + julianscaled + ECscaled +   
## SacX2 + Indexscaled + offset(Vol), data = FMWT\_DSmg4ax, na.action = "na.fail",   
## dist = "negbin")  
##   
## Pearson residuals:  
## Min 1Q Median 3Q Max   
## -0.6423 -0.4006 -0.3112 -0.1222 9.3358   
##   
## Count model coefficients (negbin with log link):  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -3.78982 2.68250 -1.413 0.15772   
## Station606 0.32219 0.56135 0.574 0.56600   
## Station608 -1.83967 0.75997 -2.421 0.01549 \*   
## prepostpre 1.57707 0.45166 3.492 0.00048 \*\*\*  
## julianscaled 0.32988 0.22290 1.480 0.13889   
## ECscaled -0.65021 0.41948 -1.550 0.12114   
## SacX2 0.04720 0.03335 1.415 0.15702   
## Indexscaled 0.45324 0.18077 2.507 0.01217 \*   
## Log(theta) -0.51473 0.32859 -1.566 0.11723   
##   
## Zero-inflation model coefficients (binomial with logit link):  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -10.492352 7.891055 -1.330 0.1836   
## Station606 -0.786989 0.777632 -1.012 0.3115   
## Station608 -3.434051 1.848376 -1.858 0.0632 .  
## prepostpre 1.890547 0.832407 2.271 0.0231 \*  
## julianscaled -0.418751 0.388013 -1.079 0.2805   
## ECscaled -0.188484 0.917624 -0.205 0.8373   
## SacX2 0.117036 0.095564 1.225 0.2207   
## Indexscaled 0.004679 0.331108 0.014 0.9887   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
##   
## Theta = 0.5977   
## Number of iterations in BFGS optimization: 34   
## Log-likelihood: -218.1 on 17 Df

#Go through all the possible models  
drepp = dredge(dsznb3)

## Fixed terms are "count\_(Intercept)" and "zero\_(Intercept)"

## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced

## Warning in solve.default(as.matrix(fit$hessian)): system is computationally  
## singular: reciprocal condition number = 3.85341e-33 (model 4515 skipped)

## Warning in sqrt(diag(vcov(model, ...))): NaNs produced

## Warning in solve.default(as.matrix(fit$hessian)): system is computationally  
## singular: reciprocal condition number = 3.96363e-20 (model 4644 skipped)

## Warning in solve.default(as.matrix(fit$hessian)): system is computationally  
## singular: reciprocal condition number = 3.9923e-33 (model 4902 skipped)

## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced

## Warning in solve.default(as.matrix(fit$hessian)): system is computationally  
## singular: reciprocal condition number = 1.29626e-22 (model 6450 skipped)

## Warning in solve.default(as.matrix(fit$hessian)): system is computationally  
## singular: reciprocal condition number = 1.11892e-36 (model 6579 skipped)

## Warning in solve.default(as.matrix(fit$hessian)): system is computationally  
## singular: reciprocal condition number = 4.69234e-23 (model 6966 skipped)

## Warning in solve.default(as.matrix(fit$hessian)): system is computationally  
## singular: reciprocal condition number = 4.03413e-22 (model 7095 skipped)

## Warning in sqrt(diag(vcov(model, ...))): NaNs produced

## Warning in solve.default(as.matrix(fit$hessian)): system is computationally  
## singular: reciprocal condition number = 1.93258e-16 (model 7869 skipped)

## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced  
  
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced

## Warning in solve.default(as.matrix(fit$hessian)): system is computationally  
## singular: reciprocal condition number = 9.115e-37 (model 15287 skipped)

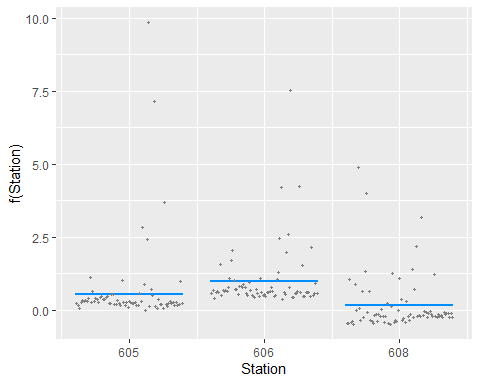
## Warning in sqrt(diag(vcov(model, ...))): NaNs produced

head(drepp)

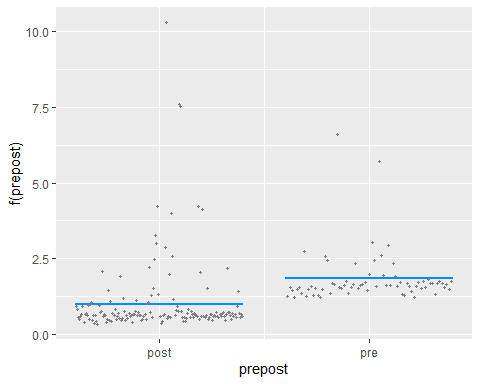
## Global model call: zeroinfl(formula = catch ~ Station + prepost + julianscaled +   
## ECscaled + SacX2 + Indexscaled + offset(Vol), data = FMWT\_DSmg4ax,   
## na.action = "na.fail", dist = "negbin")  
## ---  
## Model selection table   
## cnt\_(Int) cnt\_ECs cnt\_Ind cnt\_jln cnt\_prp cnt\_SX2 cnt\_Stt zer\_(Int)  
## 8128 -3.19300 -0.6357 0.5839 0.2554 + 0.02957 + -704.6  
## 904 -0.03467 -0.8002 0.4179 0.3426 -280.7  
## 2968 2.40400 -0.6485 0.3783 0.3787 -0.02877 180.2  
## 5032 -0.52790 -0.7887 0.4089 0.3406 + -136.9  
## 16320 -3.03200 -0.6511 0.5997 0.2928 + 0.02813 + -502.7  
## 7999 0.34170 0.5557 0.4048 + -0.01761 + -1013.0  
## zer\_ECs zer\_Ind zer\_jln zer\_prp zer\_SX2 zer\_Stt zer\_off(Vol) df  
## 8128 -26.44 13.540 -24.29 + 7.644 + 17  
## 904 -104.00 -396.700 -60.17 9  
## 2968 -26.13 -117.400 -15.12 -3.148 11  
## 5032 -56.98 -209.700 -14.91 + 13  
## 16320 -20.12 9.889 -15.74 + 5.480 + + 17  
## 7999 40.520 -22.15 + 9.733 + 15  
## logLik AICc delta weight  
## 8128 -197.143 431.2 0.00 0.303  
## 904 -206.423 431.7 0.46 0.241  
## 2968 -204.388 432.0 0.79 0.204  
## 5032 -202.361 432.4 1.22 0.165  
## 16320 -198.766 434.5 3.25 0.060  
## 7999 -201.889 436.1 4.84 0.027  
## Models ranked by AICc(x)

#It liked the full model, except for the volume offset.   
  
visreg(dsznb3, gg = T)

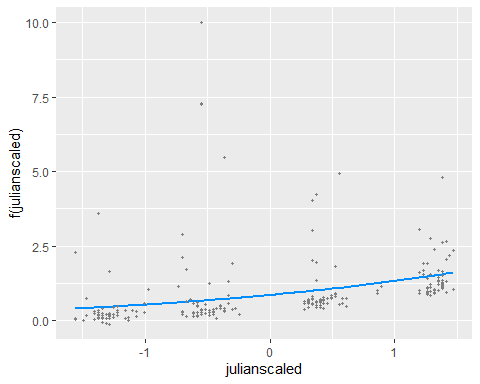
## [[1]]



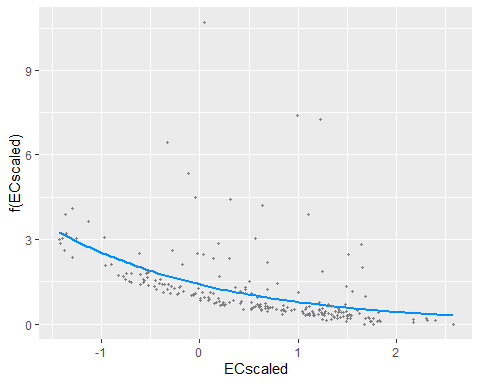
##   
## [[2]]



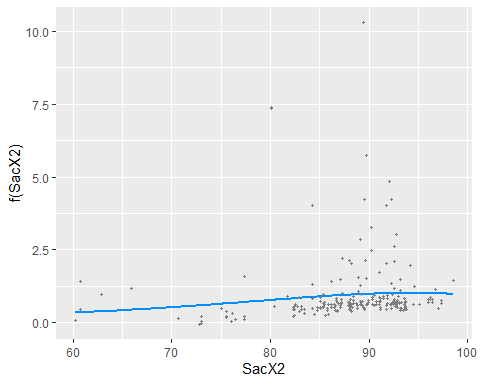
##   
## [[3]]



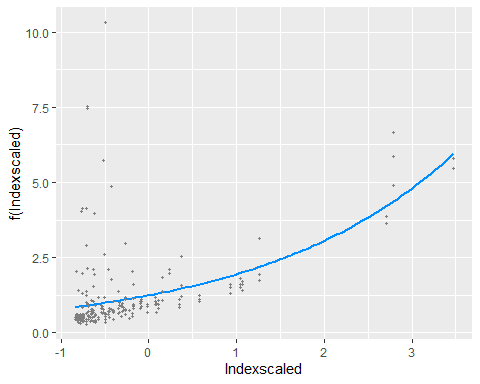
##   
## [[4]]



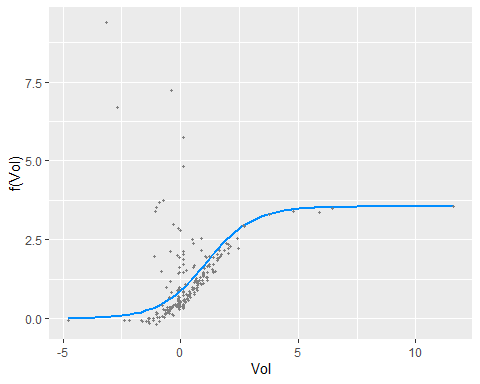
##   
## [[5]]



##   
## [[6]]



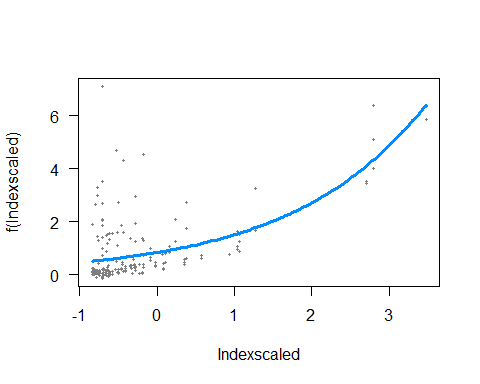
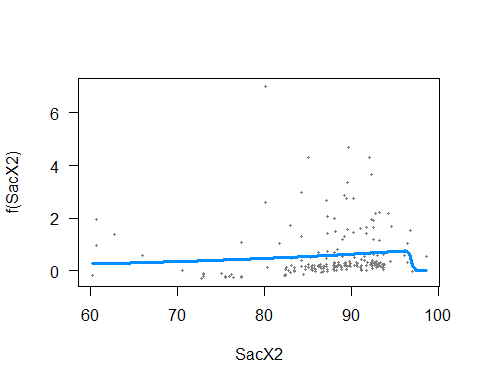
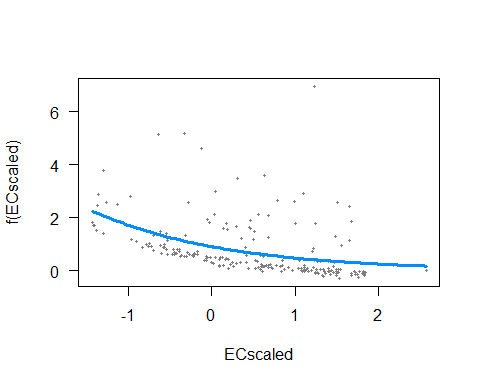
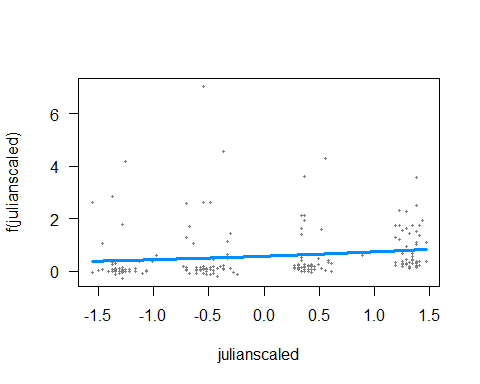
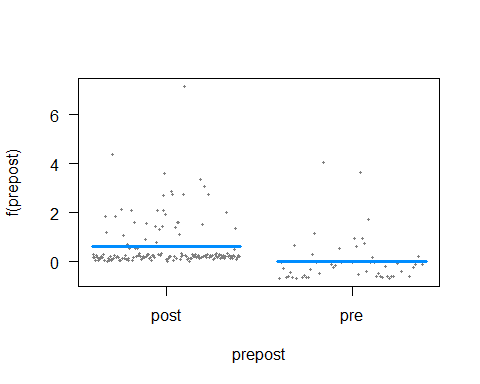
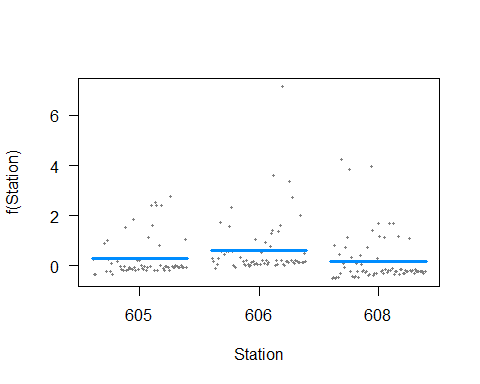
##   
## [[7]]



#Try it again without the volume offset  
dsznb4 = zeroinfl(catch~Station + prepost + julianscaled +   
 ECscaled +SacX2 + Indexscaled, dist = "negbin",   
 data = FMWT\_DSmg4ax,na.action = "na.fail")  
summary(dsznb4)

##   
## Call:  
## zeroinfl(formula = catch ~ Station + prepost + julianscaled + ECscaled +   
## SacX2 + Indexscaled, data = FMWT\_DSmg4ax, na.action = "na.fail",   
## dist = "negbin")  
##   
## Pearson residuals:  
## Min 1Q Median 3Q Max   
## -0.6999888 -0.4725685 -0.3731483 -0.0000309 6.5382520   
##   
## Count model coefficients (negbin with log link):  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -3.20326 2.31539 -1.383 0.166523   
## Station606 0.70640 0.38942 1.814 0.069684 .   
## Station608 -0.43927 0.56725 -0.774 0.438700   
## prepostpre 1.34934 0.36202 3.727 0.000194 \*\*\*  
## julianscaled 0.25699 0.16876 1.523 0.127801   
## ECscaled -0.63606 0.32113 -1.981 0.047626 \*   
## SacX2 0.02970 0.02864 1.037 0.299642   
## Indexscaled 0.58457 0.17444 3.351 0.000805 \*\*\*  
## Log(theta) -0.56142 0.26165 -2.146 0.031897 \*   
##   
## Zero-inflation model coefficients (binomial with logit link):  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -647.853 395.939 -1.636 0.1018   
## Station606 -5.078 5.090 -0.998 0.3185   
## Station608 -60.814 35.804 -1.699 0.0894 .  
## prepostpre 59.548 38.351 1.553 0.1205   
## julianscaled -22.253 13.937 -1.597 0.1103   
## ECscaled -24.391 14.473 -1.685 0.0919 .  
## SacX2 7.027 4.270 1.646 0.0998 .  
## Indexscaled 12.415 8.556 1.451 0.1468   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
##   
## Theta = 0.5704   
## Number of iterations in BFGS optimization: 460   
## Log-likelihood: -197.3 on 17 Df

visreg(dsznb4)

 OK, That’s wierd the pre/post coefficient reversed.I think we’ll stick with the offset. Anyway, It doesn’t look like this is a particularly useful avenue for evaluating the gates, since Delta Smelt were a lot more common everywhere in the 70s and early 80s than they were in the 90s and 2000s