

Exploitation Rate Creel Titration

Colin Dassow

2025-03-07

Contents

1 WDNR Creel Titration	1
1.1 Purpose	1
1.2 Current Creel Data as Baseline	1
1.3 Walleye Exploitation Rate u	2
1.4 Appendix	30

1 WDNR Creel Titration

1.1 Purpose

The main question this analysis seeks to answer:

How much less creel data could be collected, on an individual survey basis while still maintaining the stratified random sampling design, and still produce exploitation rate (u) estimates that are NOT significantly different from exploitation rates obtained using the current creel survey design?

The data informing these analyses has all been pulled from the Fisheries Management Information System (FMIS) using the `wdnr.fmdb` R package built to interface with this database. All analyses were conducted in R using R version 4.3.1 (2023-06-16 ucrt).

1.2 Current Creel Data as Baseline

DNR has amassed a large amount of inland creel information over many decades (1984-2023) across many waterbodies (n=317). Using this data and functionality of `wdnr.fmdb` it is possible to calculate effort, catch, harvest, and harvest rate for multiple species for each unique creel survey conducted. This information can be further grouped by month, season, day type, etc. to provide insight into important temporal patterns in the fisheries metrics of interest. These calculations are accomplished fairly easily using the following code:

```
# reading in DNR creel data

cserv = get_creel_surveys()
cvis = get_creel_visits()
ccou = get_creel_counts()
```

```

cint = get_creel_int_party()
cfish = get_creel_fish_data()

ceff = calc_creel_effort(creel_count_data = ccou, creel_int_data = cint)

charv = calc_creel_harvest(creel_count_data = ccou, creel_int_data = cint,
                           creel_fish_data = cfish)

charvR = calc_creel_harvest_rates(creel_fish_data = cfish)

```

1.3 Walleye Exploitation Rate u

Here I'm dealing specifically with walleye exploitation rate estimates for ceded territory lakes only. Exploitation rate is a key metric under the Voigt Decision and any proposed changes to creel surveys would need to ensure that the proper exploitation rate information can still be gathered.

What I've done here is calculate exploitation rates based on the full data sets as well as for 5 scenarios:

1. all winter creel information removed
2. only creel information from 'summer' May to August is used
3. 25% of weekday creel visits per month are removed
4. 50% of weekday creel visits per month are removed
5. 50% of weekend creel visits per month are removed.

Creel visits are removed on a month by month basis by randomly removing creel visits to the lake that day

This random removal of days per month and day type should maintain the statistical integrity of the stratified random design that is a part of the creel survey. For context, this was not how data was removed in a similar analysis done by Deroba et al. (2007) where they removed entire weeks (randomly selected) per month instead of randomly selecting days. Their analysis also employed t-tests to analyze the resulting exploitation rates in each of their reduced data sets which relies on the assumption of normality and has been discussed above.

Exploitation rate (u) was calculated as:

$$u = \frac{R}{M}$$

$$R = \left(\frac{C}{T}\right)H$$

Where R is the estimated total number of marked fish harvested for a given creel strata and M is the total number of walleye marked that spring during fyke net surveys. The estimate of the number of marked fish harvested per strata (R) is obtained by applying the proportion of marked fish observed in the creel to the total harvest estimate for that strata (H). C is the actual number of marked fish observed in a given creel strata and T is the total number of fish examined.

When calculating u for each data set I first calculated u by each month and day type then calculated an average u for that creel survey based on `survey.seq.no` to produce one exploitation rate for that survey. This was done to match the exploitation rate database which provides exploitation rates on a survey-level and not the strata-level.

Another important note, all exploitation rates = 0 were dropped before modeling for two reasons, 1) because the data is lognormally distributed (Figures 1 and 2) taking the log of 0 isn't possible and the typical approach would be to add a small number to make it non-zero but this leads to skewed data in this case that would require modeling with a different data distribution. An alternative to this would likely be to model the data with a gamma or beta distribution (see appendix for the results of this alternate modeling). Here I stuck with lognormal and dropped 0s. 2) If a creel survey produces a 0 now, chances are that as more data is removed from that survey it won't suddenly become non-zero, so the 0s are not likely to change under future creel designs. If anything more lakes may produce 0 exploitation rates if fewer samples are taken and the scenarios here show just that.

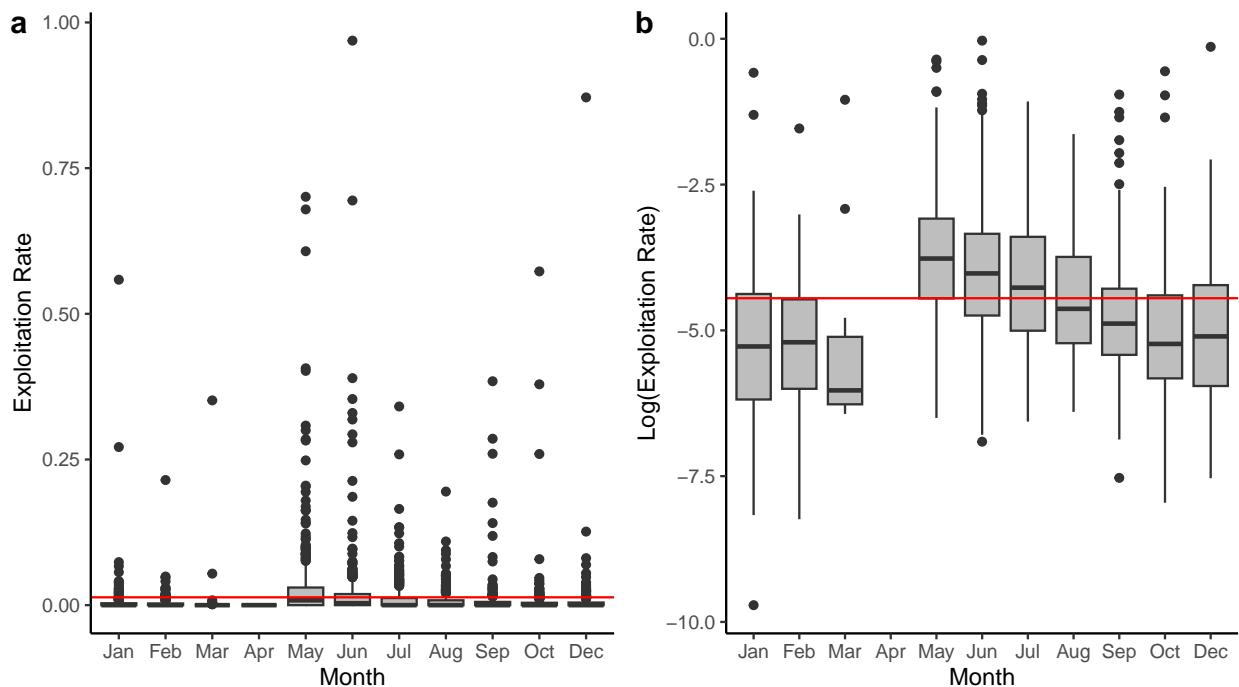


Figure 1: Distribution of exploitation rates by month (panel a) for all years of CTWI creel data. Horizontal red line is overall mean exploitation rate. Black points are outliers. Panel B is the same information presented on a log-scale. In both plots exploitation rates of 0 are omitted.

1.3.1 Bayesian Model Fitting

In order to estimate parameters for a model of the exploitation rate distribution that produced what is represented in the creel data a Bayesian modeling approach was used. This is because the exploitation rate data are not normally distributed and thus don't meet the assumptions of typical analyses based on normality. Here I've chosen to model the data with a lognormal distribution, other potential options for the type of data are the gamma and beta distributions (see Appendix for more information). A Bayesian approach also allows for the inclusion of prior information about the system and does not rely on p-values (though some Bayesian p-values are presented later on, these work a bit differently) which are arbitrary in nature and can be manipulated via high sample sizes. The Bayesian approach can be adapted to accommodate any prior distribution we feel is necessary be it lognormal, beta, or gamma.

Individual likelihoods were constructed for each reduced data set and fit using the functions in the `BayesianTools` package in R. These model fits were checked for convergence using visual inspection of trace

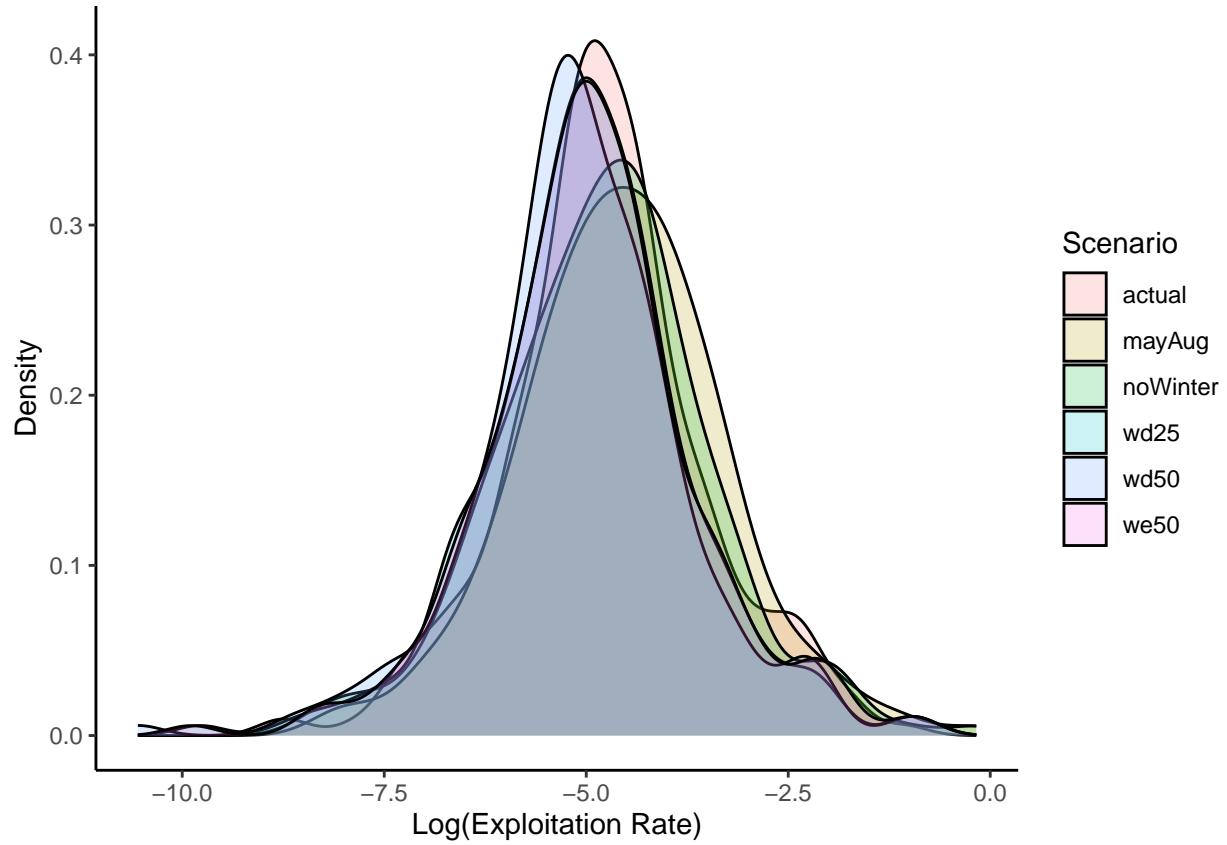


Figure 2: Distribution, on a log scale, of exploitation rates for the full data set plus 5 scenarios with reduced data. Actual = full data set, mayAug = May to August creel data only, noWinter = winter creel data removed, wd25 = 25% of weekday creel visits per month removed, wd50 = 50% of weekday creel visits per month removed, we50 = 50% of weekend creel visits per month removed.

Table 1: Table of the number of surveys with exploitation rate estimates equal to 0 for each data scenario.

Scenario	Zeros
Actual	86
No Winter	102
May-August	108
25% Weekday Reduction	110
50% Weekday Reduction	115
50% Weekend Reduction	107

plots, posterior distributions, and Gelman-Rubin Diagnostic tests to ensure that models had converged. All models were fit using Differential Evolution Markov-Chain-Monte-Carlo algorithms run for 10,000 iterations with the first 5,000 discarded as burn-in.

```

# one likelihood to estimate parms for using the 6
# different treatments creating data frame to model with u
# rates that are NA removed, these are 6% of the data and
# are from years where creel happened but no fish were FN
# marked concurrently.
modDat = ttrExp[!is.na(ttrExp$exp.rate), ]

# removing 0s since they can't be logged and if I were to
# make them a small number they would throw off the data
# and make it bimodal which would probably mean switching
# to a gamma or beta distribution to model the data. Could
# work, but I don't have time to mess with that right now.
# I'm going to operate under the assumption that if a
# survey give a 0 exploitation rate with the full survey,
# then us collecting less data isn't going to change that
# number.
zeros.actual = sum(modDat$exp.rate[modDat$treat == "actual"] ==
  0)
zeros.nw = sum(modDat$exp.rate[modDat$treat == "noWinter"] ==
  0)
zeros.ma = sum(modDat$exp.rate[modDat$treat == "mayAug"] == 0)
zeros.wd25 = sum(modDat$exp.rate[modDat$treat == "wd25"] == 0)
zeros.wd50 = sum(modDat$exp.rate[modDat$treat == "wd50"] == 0)
zeros.we50 = sum(modDat$exp.rate[modDat$treat == "we50"] == 0)

Ztab = data.frame(Scenario = c("Actual", "No Winter", "May-August",
  "25% Weekday Reduction", "50% Weekday Reduction", "50% Weekend Reduction"),
  Zeros = c(zeros.actual, zeros.nw, zeros.ma, zeros.wd25, zeros.wd50,
  zeros.we50))
kable(Ztab, format = "latex", caption = "Table of the number of surveys with exploitation rate estimate

# removing 0s here as described in the text.
modDat = modDat[modDat$exp.rate != 0, ]

#### ACTUAL ####
uLL.a = function(param) {
  alpha = param[1]
  beta = param[2]
```

```

us = rlnorm(nrow(modDat[modDat$treat == "actual", ]), meanlog = alpha,
            sdlog = beta)
ll = dlnorm(modDat$exp.rate[modDat$treat == "actual"], meanlog = mean(log(us)),
            sdlog = sd(log(us)), log = T)
return(sum(ll))
}

prior = createTruncatedNormalPrior(mean = c(mean(log(modDat$exp.rate[modDat$treat ==
  "actual"])), sd(log(modDat$exp.rate[modDat$treat == "actual"]))),
  sd = c(1, 1), lower = c(-15, 0), upper = c(15, 5))

setup.actual = createBayesianSetup(uLL.a, prior = prior)

settings = list(iterations = 10000, nrChains = 3, message = F,
  burnin = 5000)
set.seed(10)
expR.actual = runMCMC(bayesianSetup = setup.actual, sampler = "DEzs",
  settings = settings)

##### NW #####
uLL.nw = function(param) {
  alpha = param[1]
  beta = param[2]

  us = rlnorm(nrow(modDat[modDat$treat == "noWinter", ]), meanlog = alpha,
              sdlog = beta)
  ll = dlnorm(modDat$exp.rate[modDat$treat == "noWinter"],
              meanlog = mean(log(us)), sdlog = sd(log(us)), log = T)
  return(sum(ll))
}

prior = createTruncatedNormalPrior(mean = c(mean(log(modDat$exp.rate[modDat$treat ==
  "noWinter"])), sd(log(modDat$exp.rate[modDat$treat == "noWinter"]))),
  sd = c(1, 1), lower = c(-15, 0), upper = c(15, 5))

setup.nw = createBayesianSetup(uLL.nw, prior = prior)

settings = list(iterations = 10000, nrChains = 3, message = F,
  burnin = 5000)
set.seed(10)
expR.nw = runMCMC(bayesianSetup = setup.nw, sampler = "DEzs",
  settings = settings)

##### MA #####
uLL.ma = function(param) {
  alpha = param[1]
  beta = param[2]

  us = rlnorm(nrow(modDat[modDat$treat == "mayAug", ]), meanlog = alpha,
              sdlog = beta)
  ll = dlnorm(modDat$exp.rate[modDat$treat == "mayAug"], meanlog = mean(log(us)),

```

```

    sdlog = sd(log(us)), log = T)
  return(sum(l1))
}

prior = createTruncatedNormalPrior(mean = c(mean(log(modDat$exp.rate[modDat$treat ==
  "mayAug"])), sd(log(modDat$exp.rate[modDat$treat == "mayAug"]))),
  sd = c(1, 1), lower = c(-15, 0), upper = c(15, 5))

setup.ma = createBayesianSetup(uLL.ma, prior = prior)

settings = list(iterations = 10000, nrChains = 3, message = F,
  burnin = 5000)
set.seed(10)
expR.ma = runMCMC(bayesianSetup = setup.ma, sampler = "DEzs",
  settings = settings)

##### WD25 #####
uLL.wd25 = function(param) {
  alpha = param[1]
  beta = param[2]

  us = rlnorm(nrow(modDat[modDat$treat == "wd25", ]), meanlog = alpha,
    sdlog = beta)
  l1 = dlnorm(modDat$exp.rate[modDat$treat == "wd25"], meanlog = mean(log(us)),
    sdlog = sd(log(us)), log = T)
  return(sum(l1))
}

prior = createTruncatedNormalPrior(mean = c(mean(log(modDat$exp.rate[modDat$treat ==
  "wd25"])), sd(log(modDat$exp.rate[modDat$treat == "wd25"]))),
  sd = c(1, 1), lower = c(-15, 0), upper = c(15, 5))

setup.wd25 = createBayesianSetup(uLL.wd25, prior = prior)

settings = list(iterations = 10000, nrChains = 3, message = F,
  burnin = 5000)
set.seed(10)
expR.25wd = runMCMC(bayesianSetup = setup.wd25, sampler = "DEzs",
  settings = settings)

##### WD50 #####
uLL.wd50 = function(param) {
  alpha = param[1]
  beta = param[2]

  us = rlnorm(nrow(modDat[modDat$treat == "wd50", ]), meanlog = alpha,
    sdlog = beta)
  l1 = dlnorm(modDat$exp.rate[modDat$treat == "wd50"], meanlog = mean(log(us)),
    sdlog = sd(log(us)), log = T)
  return(sum(l1))
}

```

```

prior = createTruncatedNormalPrior(mean = c(mean(log(modDat$exp.rate[modDat$treat ==
  "wd50"])), sd(log(modDat$exp.rate[modDat$treat == "wd50"]))),
  sd = c(1, 1), lower = c(-15, 0), upper = c(15, 5))

setup.wd50 = createBayesianSetup(uLL.wd50, prior = prior)

settings = list(iterations = 10000, nrChains = 3, message = F,
  burnin = 5000)
set.seed(10)
expR.50wd = runMCMC(bayesianSetup = setup.wd50, sampler = "DEzs",
  settings = settings)

##### WE50 #####
uLL.we50 = function(param) {
  alpha = param[1]
  beta = param[2]

  us = rlnorm(nrow(modDat[modDat$treat == "we50", ]), meanlog = alpha,
    sdlog = beta)
  ll = dlnorm(modDat$exp.rate[modDat$treat == "we50"], meanlog = mean(log(us)),
    sdlog = sd(log(us)), log = T)
  return(sum(ll))
}

prior = createTruncatedNormalPrior(mean = c(mean(log(modDat$exp.rate[modDat$treat ==
  "we50"])), sd(log(modDat$exp.rate[modDat$treat == "we50"]))),
  sd = c(1, 1), lower = c(-15, 0), upper = c(15, 5))

setup.we50 = createBayesianSetup(uLL.we50, prior = prior)

settings = list(iterations = 10000, nrChains = 3, message = F,
  burnin = 5000)
set.seed(10)
expR.50we = runMCMC(bayesianSetup = setup.we50, sampler = "DEzs",
  settings = settings)

```

1.3.2 Model Checking

Here model fit is being checked in several ways. Initially a visual inspection comparing data simulated using median parameter values from the posterior compared to the observed data for that scenario. A good model fit here is evident when the distribution of the observed and predicted data overlap nearly entirely (Figure 3).

Gelman-Rubin diagnostics were also used to assess convergence, while this doesn't assess model fit it does describe whether or not the MCMC algorithm converged on a solution or whether the parameter space has not been fully explored yet. For this test, values below 1.1 are considered 'converged' with a value of 1 being the lowest possible score. All models had Gelman-Rubin test values < 1.03.

Lastly, a Bayesian p-value was calculated for each model to further describe model fit to the data (Figure 2). Bayesian p-values are based on the same idea as a frequentist p-value : *What is the probability of observing a more extreme test statistic than the one calculated from the observed data*, but is calculated differently and easier to transparently critique. To calculate a Bayesian p-value each set of parameter values in the MCMC chain is used to parameterize a lognormal distribution from which a set of 'new' data are drawn. A test statistic is calculated for this new data and determined to either be more or less extreme than the test

statistic that is for the observed data. The proportion of these test statistics that are more extreme than the observed is then calculated. If the model has done a good job of fitting the data then the parameter values should generally generate data that looks like the observed data and the test statistic should be equally likely to be more or less extreme than the observed value. Thus, with Bayesian p-values a value of 0.5 is ideal as it means the model can generate data that matches the distribution of the observed data really well (Table 2, 3). If this p-value was very low ($< .10$) or very high (> 0.9) that would indicate that the model is not fitting the data well because it is unable to regularly reproduce the observed data.

In this analysis two test statistics have been chosen to provide alternate, but not unrelated, measures of model fit. These are the coefficient of variation and standard deviation. Any test statistic of choice could be chosen as long as it can be justified for the objectives of the analysis at hand (see the appendix for further exploration of alternative test statistics).

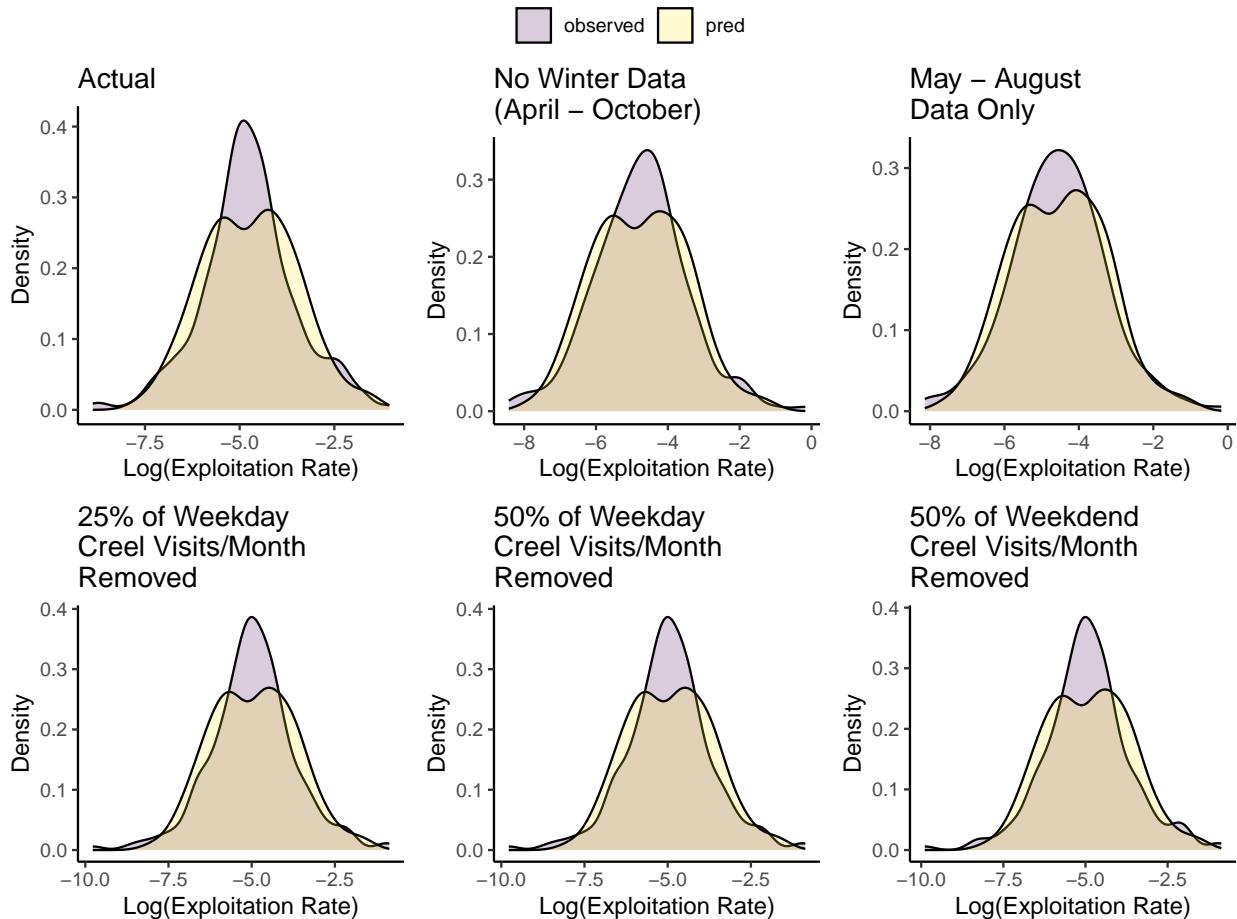


Figure 3: Distributions of the observed and model predicted data for each scenario. Model predicted data comes from normal distribution paramterized using the median parameter estimate of the model posterior.

```
gelmanDiagnostics(expR.actual) # 1.03 converged
gelmanDiagnostics(expR.nw) # 1.01 converged
gelmanDiagnostics(expR.ma) # 1.01 converged
gelmanDiagnostics(expR.25wd) # 1.02 converged
gelmanDiagnostics(expR.50wd) # 1.03 converged
gelmanDiagnostics(expR.50we) # 1.03 converged
```

Table 2: Bayesian p-values for a lognormal distributed model. Each p-value describes whether or not there is a significant difference between data generated by the fitted model and the actual data for that scenario. Two metrics are assessed here, coefficient of variation (CV) and standard deviation (SD).

Scenario	CV p-value	SD p-value
Actual	0.479	0.530
No Winter	0.467	0.539
May-August	0.459	0.546
25% weeday removal	0.464	0.531
50% weekday removal	0.475	0.525
50% weekend removal	0.481	0.523

Table 3: Bayesian p-values for the comparison between the actual data and the scenario-specific model fit to a lognormal distribution. This test is asking whether the scenario-specific model can approximate the actual data. When true this signals no effect of the data reduction for that scenario on the resulting exploitation rates. Two metrics are assessed here, coefficient of variation (CV) and standard deviation (SD).

Scenario	CV p-value	SD p-value
Actual	0.472	0.530
No Winter	0.217	0.795
May-August	0.138	0.738
25% weeday removal	0.519	0.683
50% weekday removal	0.497	0.766
50% weekend removal	0.405	0.771

1.3.3 Inference

Having established that the models are fitting the data well, some inference can be gained as to whether or not the reductions in creel effort proposed here would result in a meaningful change in the exploitation rates estimated from the reduced data.

A Bayesian p-value can be employed here too. A test between the cv and sd of the actual data and the cv and sd of the reduced data can describe whether the model of the reduced data is able to approximate the actual data or not. Successful approximations are p-values close to 0.5 with reduced fit as values increase or decrease beyond 0.5. Generally, the rule of thumb is that p-values < 0.10 or > 0.90 signal unacceptable fits. However, these cutoffs can be set based on the objectives of the study at hand and the values of the decision makers.

The results of these tests suggest that when using coefficient of variation as the variance metric to assess model fit and the < 0.10 or > 0.90 rule that the no winter and may-august data reduction models both approximate the actual data ok, but not as well as the 3 percentage reductions whose Bayesian p-values suggest they produce exploitation rate distributions that are very similar to the actual data (Table 3). When using standard deviation as the metric for comparison then all models approximate the actual data pretty well as the model fit to the actual data (Table 3). The ‘No Winter’ scenario and two 50% reduction scenarios are starting to creep higher but still not to a point where they’re different from the actual data. The reason for this discrepancy between metrics is likely that the coefficient of variation is standardized based on the mean of the distribution while the standard deviation is not and is a raw variance metric. In this case using the coefficient of variation results seems like the most pragmatic choice, it is also supported by multi-panel plot comparing the actual exploitation rates to the distributions of simulated data from each scenario (Figure 4). That would appear to be two lines of evidence pointing in the same direction.

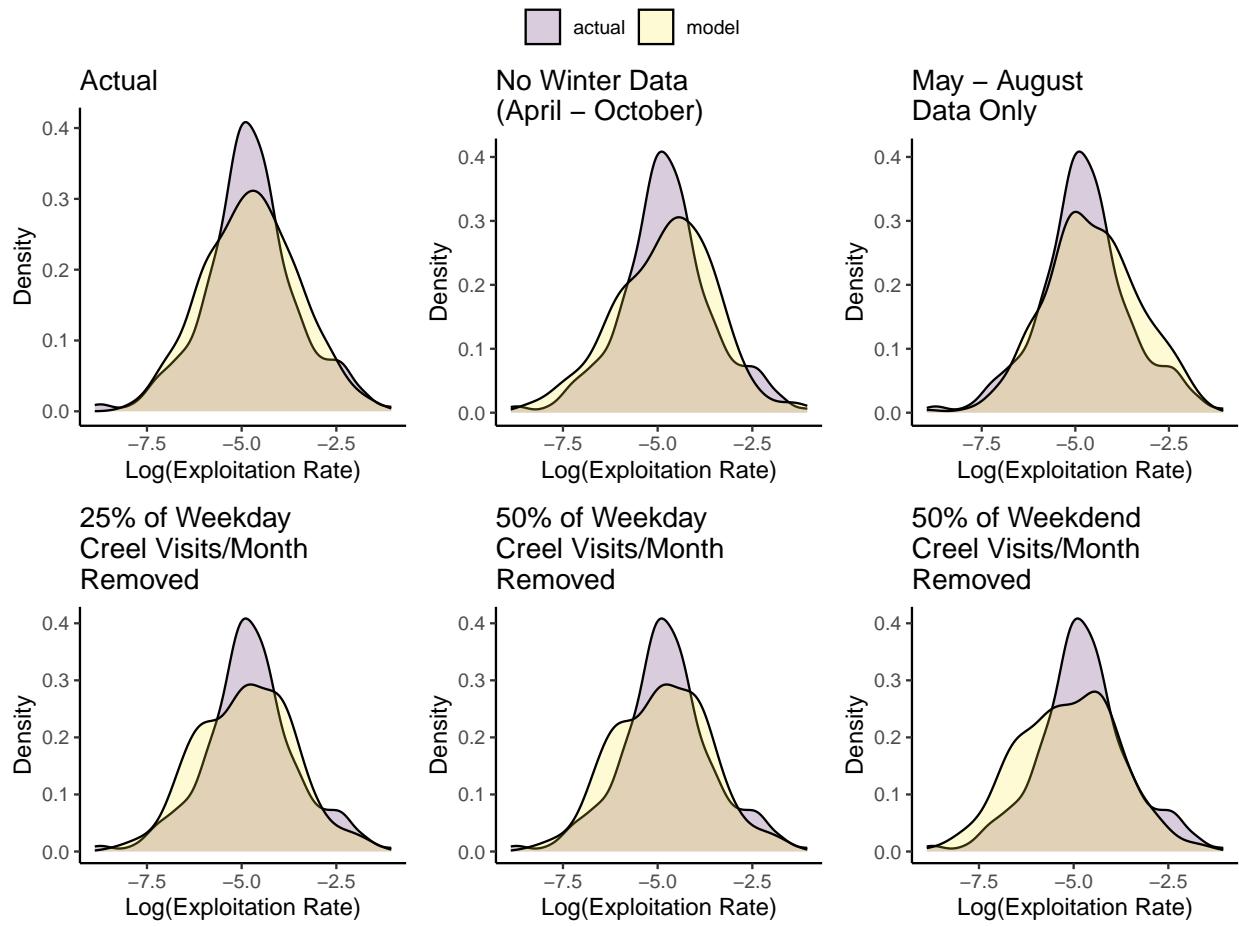


Figure 4: Comparison of the distribution of actual exploitation rates calculated from creel data and exploitation rates calculated from simulated creel data for each data reduction scenario and the resulting median parameter values for their respective model fits.

Table 4: Counts of the number of times the exploitation rate estimate for the reduced data scenario differs from the actual exploitation rate by more than 1 standard deviation. NAs occurred where there was no variance in the exploitation rate estimate and thus no standard deviation could be calculated for the actual data.

Data Reduction Scenario	u's exceeding actual	u's not exceeding actual	NAs
actual	0	303	45
mayAug	4	284	24
noWinter	3	289	25
wd25	3	300	35
wd50	3	300	35
we50	3	300	35

1.3.4 Year Specific Analyses

Instead of modeling the full population of exploitation rates as was done above, here I have done the model fitting on individual creel-years so that I can see how the removal of data for a creel-year (which is a comparatively large amount of missing data since only 16-20 lakes are creeled each year) impacts the exploitation rates estimated that year and used to set safe harvest limits. This analysis will follow the same routine as the whole-population analyses above but only consider one creel-year at a time.

I first started with some basic analyses of the data on a year by year basis to understand how the data changes from year to year and if any lake characteristics explained variation (if there is any) in exploitation rates across data reduction scenarios for specific creel-years (Figure 5).

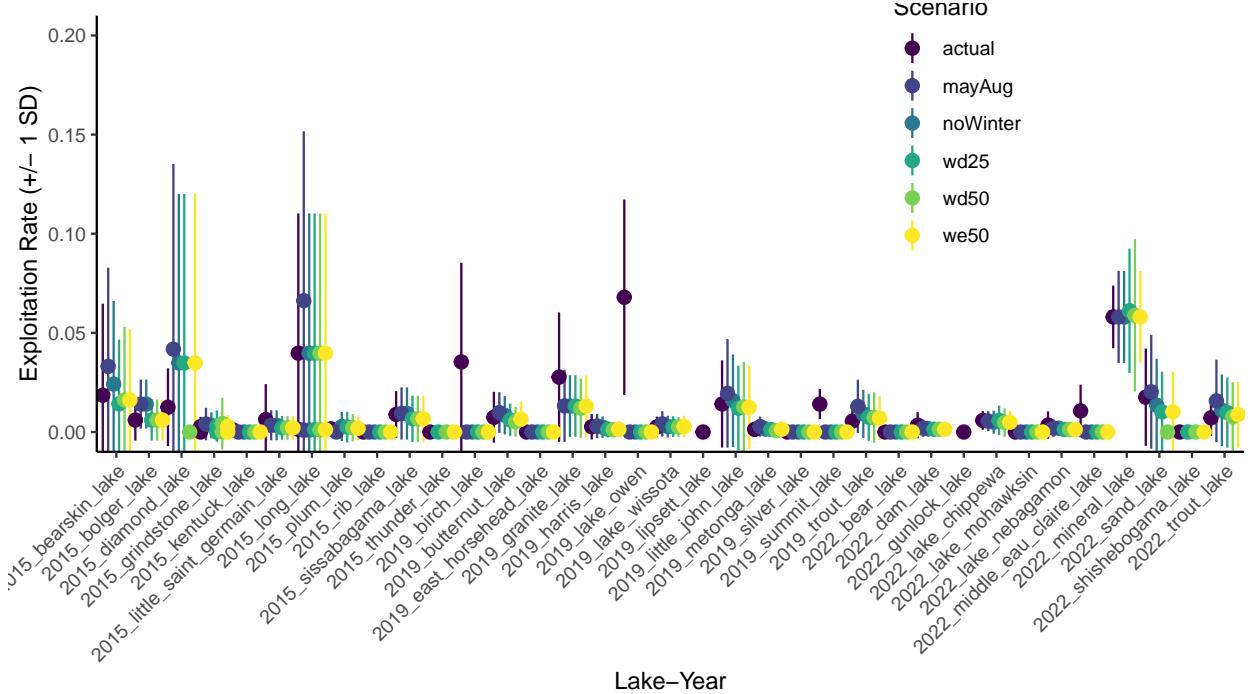


Figure 5: Comparison of the exploitation rates estimated from the reduced data for a handful of creel surveys. Most data reductions do not result in exploitation rates that are much different from the actual data (most are within 1 SD of the actual). Colors represent data reduction scenarios, points are mean exploitation rate estimates across all months of that creel survey and vertical lines represent 1 standard deviation.

After the brief analysis of lake characteristics in light of the difference between the actual exploitation rate

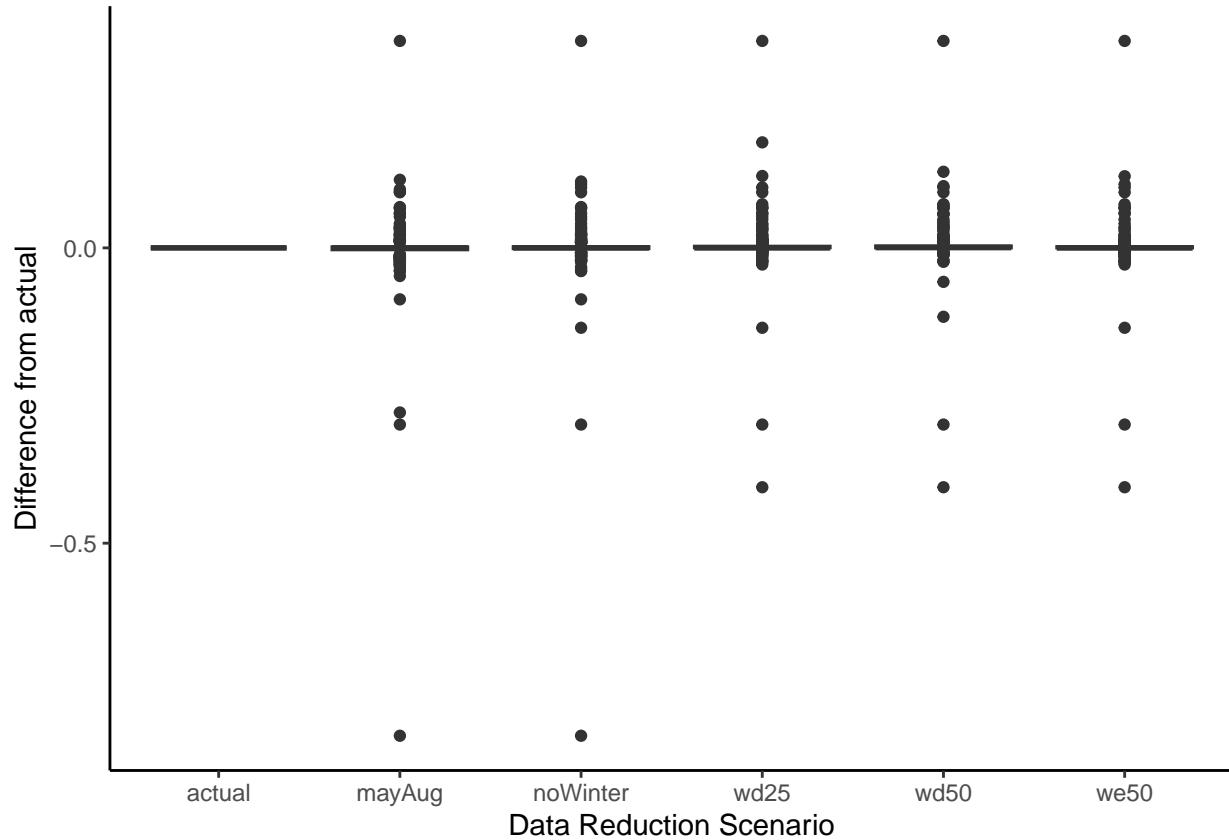


Figure 6: Boxplot of the difference between the actual exploitation rate and the data-reduced exploitation rate for each creel survey in the creel dataset. Most of the data exhibits differences very near 0, dots represent outliers ($x < |x - \text{median}| > 1.5 \times \text{IQR}$)

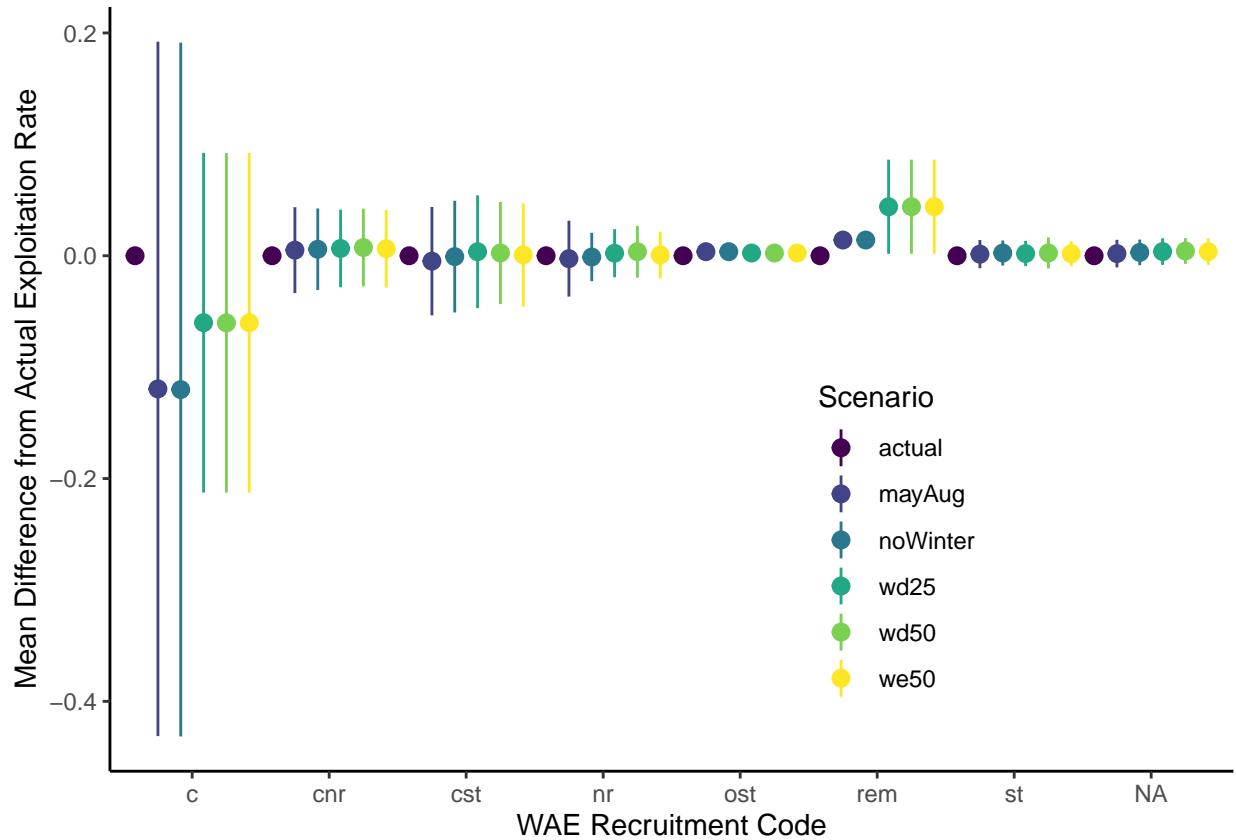


Figure 7: Comparison of the magnitude of the difference between the mean actual exploitation rate and the estimate derived from each reduced data set. Vertical lines represent 1 standard deviation. Walleye recruitment codes are along the x axis. Note the y-axis scale, many of the differences are small.

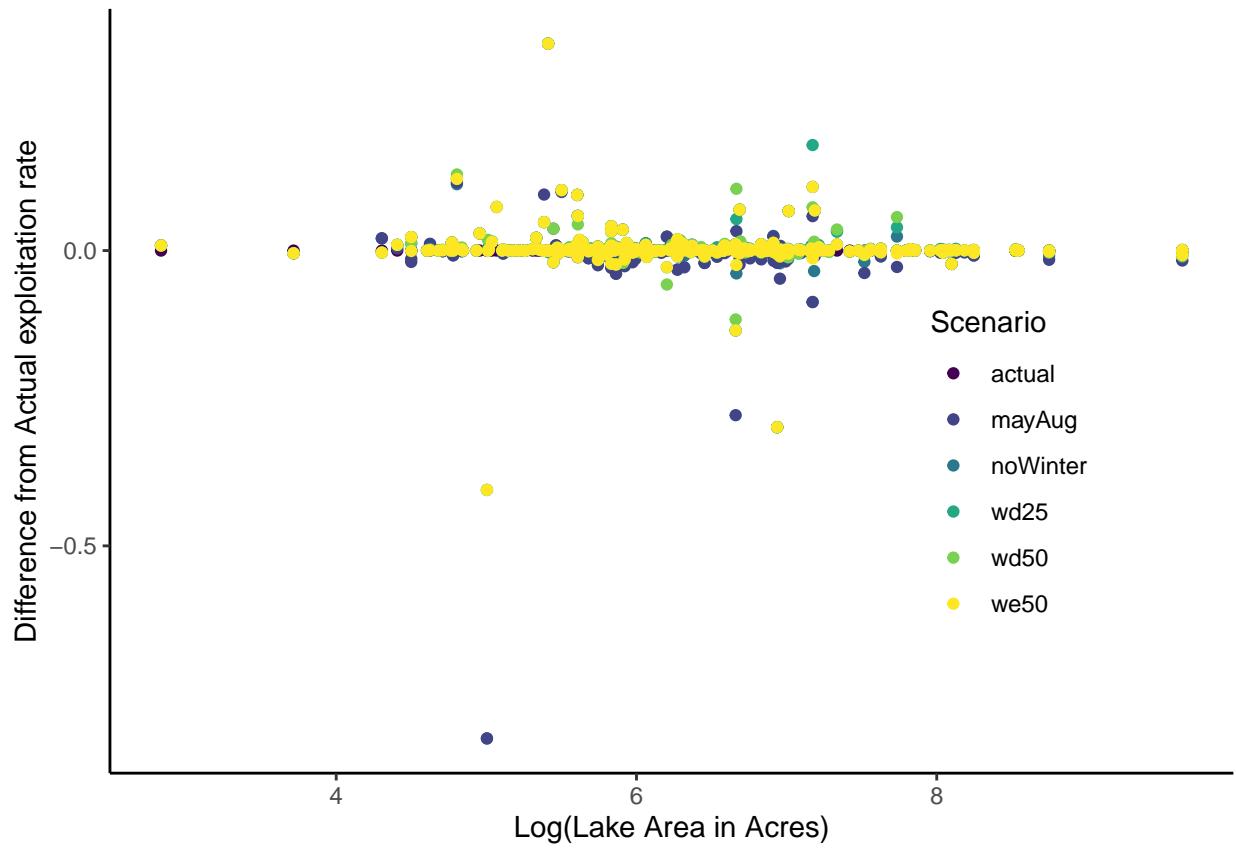


Figure 8: Comparison of the magnitude of the difference between the mean actual exploitation rate and the estimate derived from each reduced data set. Note the y-axis scale, many of the differences are small.

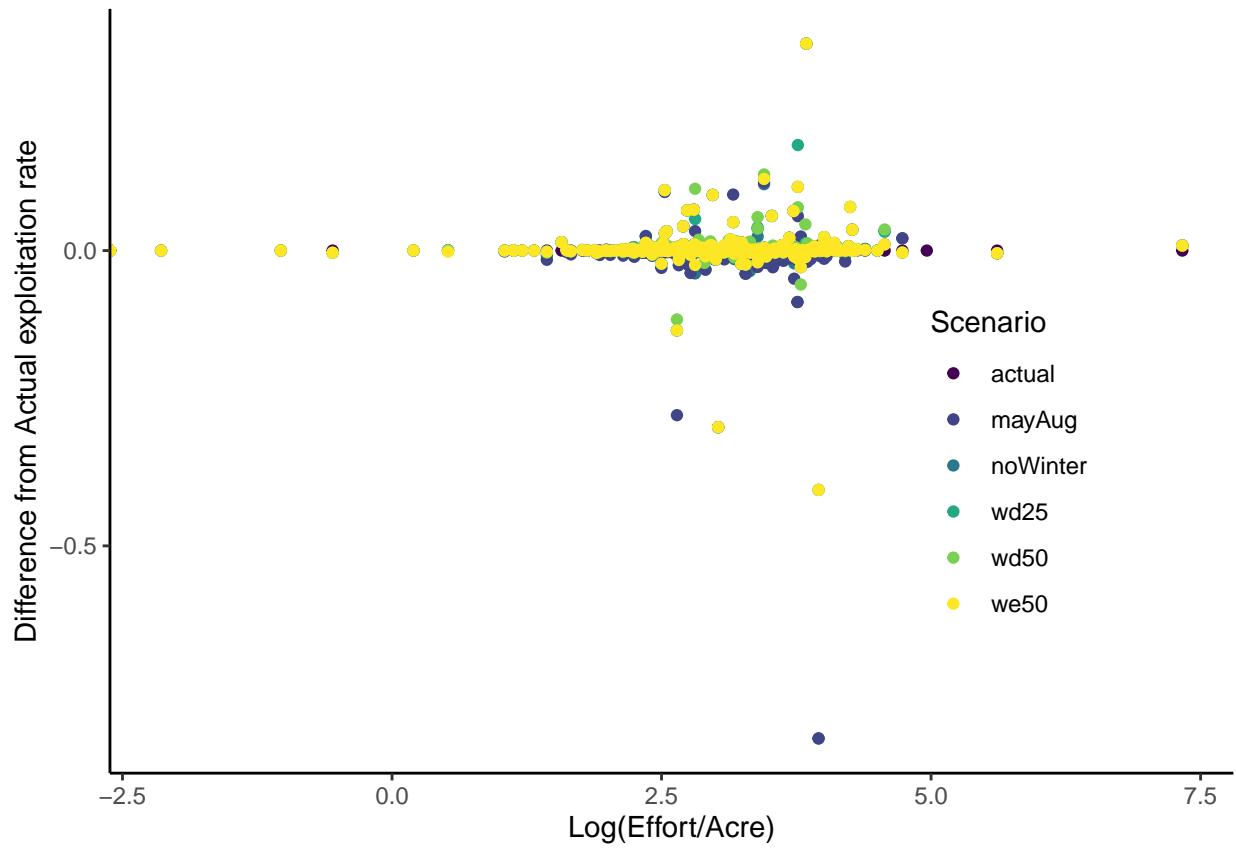


Figure 9: Comparison of the magnitude of the difference between the mean actual exploitation rate and the estimate derived from each reduced data set. Note the y-axis scale, many of the differences are small.

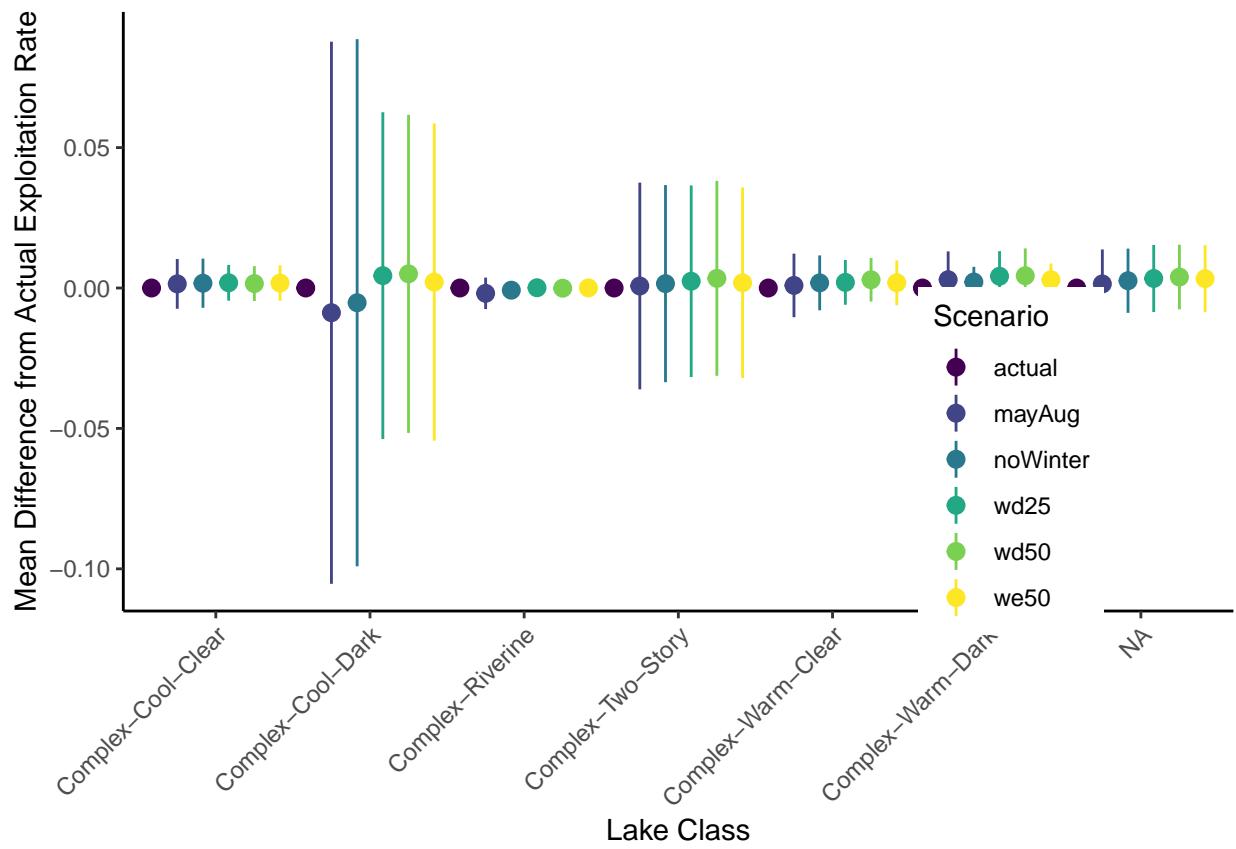


Figure 10: Comparison of the magnitude of the difference between the mean actual exploitation rate and the estimate derived from each reduced data set. Vertical lines represent 1 standard deviation. Note the y-axis scale, many of the differences are very small.

estimate and the reduced data estimate I moved on to the Bayesian model fitting process used earlier to evaluate whether the exploitation rate estimates obtained from the reduced data differed significantly from the estimates calculated using the actual creel data.

```
# MODELING EFFECTS OF DATA REDUCTION ON INDIVIDUAL YEARS

# likelihoods to fit
uLL.we50 = function(param) {
  alpha = param[1]
  beta = param[2]

  us = rlnorm(nrow(tdat[tdat$treat == "we50", ]), meanlog = alpha,
              sdlog = beta)
  ll = dlnorm(tdat$exp.rate[tdat$treat == "we50"], meanlog = mean(log(us)),
              sdlog = sd(log(us)), log = T)
  return(sum(ll))
}

uLL.wd50 = function(param) {
  alpha = param[1]
  beta = param[2]

  us = rlnorm(nrow(tdat[tdat$treat == "wd50", ]), meanlog = alpha,
              sdlog = beta)
  ll = dlnorm(tdat$exp.rate[tdat$treat == "wd50"], meanlog = mean(log(us)),
              sdlog = sd(log(us)), log = T)
  return(sum(ll))
}

uLL.wd25 = function(param) {
  alpha = param[1]
  beta = param[2]

  us = rlnorm(nrow(tdat[tdat$treat == "wd25", ]), meanlog = alpha,
              sdlog = beta)
  ll = dlnorm(tdat$exp.rate[tdat$treat == "wd25"], meanlog = mean(log(us)),
              sdlog = sd(log(us)), log = T)
  return(sum(ll))
}

uLL.ma = function(param) {
  alpha = param[1]
  beta = param[2]

  us = rlnorm(nrow(tdat[tdat$treat == "mayAug", ]), meanlog = alpha,
              sdlog = beta)
  ll = dlnorm(tdat$exp.rate[tdat$treat == "mayAug"], meanlog = mean(log(us)),
              sdlog = sd(log(us)), log = T)
  return(sum(ll))
}

uLL.nw = function(param) {
  alpha = param[1]
  beta = param[2]

  us = rlnorm(nrow(tdat[tdat$treat == "noWinter", ]), meanlog = alpha,
              sdlog = beta)
  ll = dlnorm(tdat$exp.rate[tdat$treat == "noWinter"], meanlog = mean(log(us)),
              sdlog = sd(log(us)), log = T)
  return(sum(ll))
}
```

```

    sdlog = sd(log(us)), log = T)
  return(sum(l1))
}

uLL.a = function(param) {
  alpha = param[1]
  beta = param[2]

  us = rlnorm(nrow(tdat[tdat$treat == "actual", ]), meanlog = alpha,
              sdlog = beta)
  l1 = dlnorm(tdat$exp.rate[tdat$treat == "actual"], meanlog = mean(log(us)),
              sdlog = sd(log(us)), log = T)
  return(sum(l1))
}

# removing 0s
ttrExp = ttrExp[ttrExp$exp.rate != 0, ]
loopY = sort(unique(ttrExp$year))

bpval.comp.y = data.frame(year = NA, scenario = NA, coef.var.pval = NA,
                           sd.pval = NA)

bpval.self.y = data.frame(year = NA, scenario = NA, coef.var.pval = NA,
                           sd.pval = NA)
grMetrics.y = data.frame(year = NA, scenario = NA, gr.prsf = NA,
                           gr.par1 = NA, gr.par2 = NA)

for (y in 1:length(loopY)) {
  # first get to year-specific data
  tdat = ttrExp[ttrExp$year == loopY[y], ]
  # removing 0s,
  tdat = tdat[tdat$exp.rate != 0, ]
  # Bayesian Model Fitting ACTUAL

  prior = createTruncatedNormalPrior(mean = c(mean(log(ttrExp$exp.rate[ttrExp$treat ==
    "actual"])), sd(log(ttrExp$exp.rate[ttrExp$treat == "actual"]))),
    sd = c(1, 1), lower = c(-15, 0), upper = c(15, 5))

  setup.a = createBayesianSetup(uLL.a, prior = prior)

  settings = list(iterations = 10000, nrChains = 3, message = F,
                  burnin = 5000)
  set.seed(10)
  t.a = runMCMC(bayesianSetup = setup.a, sampler = "DEzs",
                 settings = settings)
  # NW

  prior = createTruncatedNormalPrior(mean = c(mean(log(ttrExp$exp.rate[ttrExp$treat ==
    "noWinter"])), sd(log(ttrExp$exp.rate[ttrExp$treat ==
    "noWinter"]))), sd = c(1, 1), lower = c(-15, 0), upper = c(15,
    5))

  setup.nw = createBayesianSetup(uLL.nw, prior = prior)
}

```

```

settings = list(iterations = 10000, nrChains = 3, message = F,
                burnin = 5000)
set.seed(10)
t.nw = runMCMC(bayesianSetup = setup.nw, sampler = "DEzs",
                settings = settings)
# MA

prior = createTruncatedNormalPrior(mean = c(mean(log(ttrExp$exp.rate[ttrExp$treat ==
    "mayAug"])), sd(log(ttrExp$exp.rate[ttrExp$treat == "mayAug"]))),
    sd = c(1, 1), lower = c(-15, 0), upper = c(15, 5))

setup.ma = createBayesianSetup(uLL.ma, prior = prior)

settings = list(iterations = 10000, nrChains = 3, message = F,
                burnin = 5000)
set.seed(10)
t.ma = runMCMC(bayesianSetup = setup.ma, sampler = "DEzs",
                settings = settings)
# WD.25

prior = createTruncatedNormalPrior(mean = c(mean(log(ttrExp$exp.rate[ttrExp$treat ==
    "wd25"])), sd(log(ttrExp$exp.rate[ttrExp$treat == "wd25"]))),
    sd = c(1, 1), lower = c(-15, 0), upper = c(15, 5))

setup.wd25 = createBayesianSetup(uLL.wd25, prior = prior)

settings = list(iterations = 10000, nrChains = 3, message = F,
                burnin = 5000)
set.seed(10)
t.25wd = runMCMC(bayesianSetup = setup.wd25, sampler = "DEzs",
                settings = settings)
# WD.50

prior = createTruncatedNormalPrior(mean = c(mean(log(ttrExp$exp.rate[ttrExp$treat ==
    "wd50"])), sd(log(ttrExp$exp.rate[ttrExp$treat == "wd50"]))),
    sd = c(1, 1), lower = c(-15, 0), upper = c(15, 5))

setup.wd50 = createBayesianSetup(uLL.wd50, prior = prior)

settings = list(iterations = 10000, nrChains = 3, message = F,
                burnin = 5000)
set.seed(10)
t.50wd = runMCMC(bayesianSetup = setup.wd50, sampler = "DEzs",
                settings = settings)

# WE.50

prior = createTruncatedNormalPrior(mean = c(mean(log(ttrExp$exp.rate[ttrExp$treat ==
    "we50"])), sd(log(ttrExp$exp.rate[ttrExp$treat == "we50"]))),
    sd = c(1, 1), lower = c(-15, 0), upper = c(15, 5))

setup.we50 = createBayesianSetup(uLL.we50, prior = prior)

```

```

settings = list(iterations = 10000, nrChains = 3, message = F,
                burnin = 5000)
set.seed(10)
t.50we = runMCMC(bayesianSetup = setup.we50, sampler = "DEzs",
                  settings = settings)

## GR Diagnostics

gr.a = gelmanDiagnostics(t.a)
gr.ma = gelmanDiagnostics(t.ma)
gr.nw = gelmanDiagnostics(t.nw)
gr.wd25 = gelmanDiagnostics(t.25wd)
gr.wd50 = gelmanDiagnostics(t.50wd)
gr.we50 = gelmanDiagnostics(t.50we)

## BAYESIAN P-VALUE CALCS

pars.a = getSample(t.a)
pars.nw = getSample(t.nw)
pars.ma = getSample(t.ma)
pars.wd25 = getSample(t.25wd)
pars.wd50 = getSample(t.50wd)
pars.we50 = getSample(t.50we)

##### Pb ACTUAL #####
pval.actual = data.frame(alpha = rep(NA, nrow(pars.a)), beta = NA,
                           cv = NA, sd = NA)
set.seed(10)
for (i in 1:nrow(pars.a)) {
  tempdat = rlnorm(n = length(tdat$exp.rate[tdat$treat ==
    "actual"]), meanlog = pars.a[i, 1], sdlog = pars.a[i,
    2])
  pval.actual$alpha[i] = pars.a[i, 1]
  pval.actual$beta[i] = pars.a[i, 2]
  pval.actual$cv[i] = sd(log(tempdat))/mean(log(tempdat))
  pval.actual$sd[i] = sd(log(tempdat))
}
# now calculate the number of times the cv or sd
# exceeds that of the real data

pval.actual$cvExceed = 0
pval.actual$sdExceed = 0

actual.cv = sd(log(tdat$exp.rate[tdat$treat == "actual"]))/mean(log(tdat$exp.rate[tdat$treat ==
  "actual"]))
actual.sd = sd(log(tdat$exp.rate[tdat$treat == "actual"]))

pval.actual$cvExceed[pval.actual$cv > actual.cv] = 1
pval.actual$sdExceed[pval.actual$sd > actual.sd] = 1

##### Pb NO WINTER #####
pval.noWinter = data.frame(alpha = rep(NA, nrow(pars.nw)),

```

```

    beta = NA, cv = NA, sd = NA)
set.seed(10)
for (i in 1:nrow(pars.nw)) {
  tempdat = rlnorm(n = length(tdat$exp.rate[tdat$treat ==
  "noWinter"]), meanlog = pars.nw[i, 1], sdlog = pars.nw[i,
  2])
  pval.noWinter$alpha[i] = pars.nw[i, 1]
  pval.noWinter$beta[i] = pars.nw[i, 2]
  pval.noWinter$cv[i] = sd(log(tempdat))/mean(log(tempdat))
  pval.noWinter$sd[i] = sd(log(tempdat))
}

# now calculate the number of times the cv or sd
# exceeds that of the real data

pval.noWinter$cvExceed = 0
pval.noWinter$sdExceed = 0

noWinter.cv = sd(log(tdat$exp.rate[tdat$treat == "noWinter"]))/mean(log(tdat$exp.rate[tdat$treat ==
  "noWinter"]))
noWinter.sd = sd(log(tdat$exp.rate[tdat$treat == "noWinter"]))

pval.noWinter$cvExceed[pval.noWinter$cv > noWinter.cv] = 1
pval.noWinter$sdExceed[pval.noWinter$sd > noWinter.sd] = 1

##### Pb MAYAUGUST #####
pval.mayAug = data.frame(alpha = rep(NA, nrow(pars.ma)),
  beta = NA, cv = NA, sd = NA)
set.seed(10)
for (i in 1:nrow(pars.ma)) {
  tempdat = rlnorm(n = length(tdat$exp.rate[tdat$treat ==
  "mayAug"]), meanlog = pars.ma[i, 1], sdlog = pars.ma[i,
  2])
  pval.mayAug$alpha[i] = pars.ma[i, 1]
  pval.mayAug$beta[i] = pars.ma[i, 2]
  pval.mayAug$cv[i] = sd(log(tempdat))/mean(log(tempdat))
  pval.mayAug$sd[i] = sd(log(tempdat))
}

# now calculate the number of times the cv or sd
# exceeds that of the real data

pval.mayAug$cvExceed = 0
pval.mayAug$sdExceed = 0

mayAug.cv = sd(log(tdat$exp.rate[tdat$treat == "mayAug"]))/mean(log(tdat$exp.rate[tdat$treat ==
  "mayAug"]))
mayAug.sd = sd(log(tdat$exp.rate[tdat$treat == "mayAug"]))

pval.mayAug$cvExceed[pval.mayAug$cv > mayAug.cv] = 1
pval.mayAug$sdExceed[pval.mayAug$sd > mayAug.sd] = 1

##### Pb WD25 #####

```

```

pval.wd25 = data.frame(alpha = rep(NA, nrow(pars.wd25)),
  beta = NA, cv = NA, sd = NA)
set.seed(10)
for (i in 1:nrow(pars.wd25)) {
  tempdat = rlnorm(n = length(tdat$exp.rate[tdat$treat ==
    "wd25"])), meanlog = pars.wd25[i, 1], sdlog = pars.wd25[i,
    2])
  pval.wd25$alpha[i] = pars.wd25[i, 1]
  pval.wd25$beta[i] = pars.wd25[i, 2]
  pval.wd25$cv[i] = sd(log(tempdat))/mean(log(tempdat))
  pval.wd25$sd[i] = sd(log(tempdat))
}

# now calculate the number of times the cv or sd
# exceeds that of the real data

pval.wd25$cvExceed = 0
pval.wd25$sdExceed = 0

wd25.cv = sd(log(tdat$exp.rate[tdat$treat == "wd25"]))/mean(log(tdat$exp.rate[tdat$treat ==
  "wd25"]))
wd25.sd = sd(log(tdat$exp.rate[tdat$treat == "wd25"]))

pval.wd25$cvExceed[pval.wd25$cv > wd25.cv] = 1
pval.wd25$sdExceed[pval.wd25$sd > wd25.sd] = 1

##### Pb WD50 #####
pval.wd50 = data.frame(alpha = rep(NA, nrow(pars.wd50)),
  beta = NA, cv = NA, sd = NA)
set.seed(10)
for (i in 1:nrow(pars.wd50)) {
  tempdat = rlnorm(n = length(tdat$exp.rate[tdat$treat ==
    "wd50"])), meanlog = pars.wd50[i, 1], sdlog = pars.wd50[i,
    2])
  pval.wd50$alpha[i] = pars.wd50[i, 1]
  pval.wd50$beta[i] = pars.wd50[i, 2]
  pval.wd50$cv[i] = sd(log(tempdat))/mean(log(tempdat))
  pval.wd50$sd[i] = sd(log(tempdat))
}

# now calculate the number of times the cv or sd
# exceeds that of the real data

pval.wd50$cvExceed = 0
pval.wd50$sdExceed = 0

wd50.cv = sd(log(tdat$exp.rate[tdat$treat == "wd50"]))/mean(log(tdat$exp.rate[tdat$treat ==
  "wd50"]))
wd50.sd = sd(log(tdat$exp.rate[tdat$treat == "wd50"]))

pval.wd50$cvExceed[pval.wd50$cv > wd50.cv] = 1
pval.wd50$sdExceed[pval.wd50$sd > wd50.sd] = 1

```

```

##### Pb WE50 #####
pval.we50 = data.frame(alpha = rep(NA, nrow(pars.we50)),
  beta = NA, cv = NA, sd = NA)
set.seed(10)
for (i in 1:nrow(pars.we50)) {
  tempdat = rlnorm(n = length(tdat$exp.rate[tdat$treat ==
    "we50"])), meanlog = pars.we50[i, 1], sdlog = pars.we50[i,
    2])
  pval.we50$alpha[i] = pars.we50[i, 1]
  pval.we50$beta[i] = pars.we50[i, 2]
  pval.we50$cv[i] = sd(log(tempdat))/mean(log(tempdat))
  pval.we50$sd[i] = sd(log(tempdat))
}

# now calculate the number of times the cv or sd
# exceeds that of the real data

pval.we50$cvExceed = 0
pval.we50$sdExceed = 0

we50.cv = sd(log(tdat$exp.rate[tdat$treat == "we50"]))/mean(log(tdat$exp.rate[tdat$treat ==
  "we50"]))
we50.sd = sd(log(tdat$exp.rate[tdat$treat == "we50"]))

pval.we50$cvExceed[pval.we50$cv > we50.cv] = 1
pval.we50$sdExceed[pval.we50$sd > we50.sd] = 1

# df to hold pvals for model comparison to self, a way
# of knowing the model fit the data well
t.pself = data.frame(year = rep(loopY[y], 6), scenario = c("Actual",
  "No Winter", "May-August", "25% weeday removal", "50% weekday removal",
  "50% weekend removal"), coef.var.pval = NA, sd.pval = NA)
# adding self comparison pvals
t.pself$coef.var.pval = c(sum(pval.actual$cvExceed)/nrow(pval.actual),
  sum(pval.noWinter$cvExceed)/nrow(pval.noWinter), sum(pval.mayAug$cvExceed)/nrow(pval.mayAug),
  sum(pval.wd25$cvExceed)/nrow(pval.wd25), sum(pval.wd50$cvExceed)/nrow(pval.wd50),
  sum(pval.we50$cvExceed)/nrow(pval.we50))
t.pself$sd.pval = c(sum(pval.actual$sdExceed)/nrow(pval.actual),
  sum(pval.noWinter$sdExceed)/nrow(pval.noWinter), sum(pval.mayAug$sdExceed)/nrow(pval.mayAug),
  sum(pval.wd25$sdExceed)/nrow(pval.wd25), sum(pval.wd50$sdExceed)/nrow(pval.wd50),
  sum(pval.we50$sdExceed)/nrow(pval.we50))
bpval.self.y = rbind(bpval.self.y, t.pself)
## p-value tests comparing the scenarios to actual to
## show that some scenarios approximate the actual
## quite well.

t.pcomp = data.frame(year = rep(loopY[y], 6), scenario = c("Actual",
  "No Winter", "May-August", "25% weeday removal", "50% weekday removal",
  "50% weekend removal"), coef.var.pval = NA, sd.pval = NA)

pval.actual$cvComp = 0
pval.actual$sdComp = 0
pval.actual$cvComp[pval.actual$cv > actual.cv] = 1

```

```

pval.actual$sdComp[pval.actual$sd > actual.sd] = 1

pval.noWinter$cvComp = 0
pval.noWinter$sdComp = 0
pval.noWinter$cvComp[pval.noWinter$cv > actual.cv] = 1
pval.noWinter$sdComp[pval.noWinter$sd > actual.sd] = 1

pval.mayAug$cvComp = 0
pval.mayAug$sdComp = 0
pval.mayAug$cvComp[pval.mayAug$cv > actual.cv] = 1
pval.mayAug$sdComp[pval.mayAug$sd > actual.sd] = 1

pval.wd25$cvComp = 0
pval.wd25$sdComp = 0
pval.wd25$cvComp[pval.wd25$cv > actual.cv] = 1
pval.wd25$sdComp[pval.wd25$sd > actual.sd] = 1

pval.wd50$cvComp = 0
pval.wd50$sdComp = 0
pval.wd50$cvComp[pval.wd50$cv > actual.cv] = 1
pval.wd50$sdComp[pval.wd50$sd > actual.sd] = 1

pval.we50$cvComp = 0
pval.we50$sdComp = 0
pval.we50$cvComp[pval.we50$cv > actual.cv] = 1
pval.we50$sdComp[pval.we50$sd > actual.sd] = 1

t.pcomp$coef.var.pval = c(sum(pval.actual$cvComp)/nrow(pval.actual),
  sum(pval.noWinter$cvComp)/nrow(pval.noWinter), sum(pval.mayAug$cvComp)/nrow(pval.mayAug),
  sum(pval.wd25$cvComp)/nrow(pval.wd25), sum(pval.wd50$cvComp)/nrow(pval.wd50),
  sum(pval.we50$cvComp)/nrow(pval.we50))

t.pcomp$sd.pval = c(sum(pval.actual$sdComp)/nrow(pval.actual),
  sum(pval.noWinter$sdComp)/nrow(pval.noWinter), sum(pval.mayAug$sdComp)/nrow(pval.mayAug),
  sum(pval.wd25$sdComp)/nrow(pval.wd25), sum(pval.wd50$sdComp)/nrow(pval.wd50),
  sum(pval.we50$sdComp)/nrow(pval.we50))

bpval.comp.y = rbind(bpval.comp.y, t.pcomp)

# adding GR results to the output dataframe
t.gr = data.frame(year = rep(loopY[y], 6), scenario = c("Actual",
  "No Winter", "May-August", "25% weeday removal", "50% weekday removal",
  "50% weekend removal"), gr.prsf = c(gr.a[[2]], gr.nw[[2]],
  gr.ma[[2]], gr.wd25[[2]], gr.wd50[[2]], gr.we50[[2]]),
  gr.par1 = c(gr.a[[1]][1, 1], gr.nw[[1]][1, 1], gr.ma[[1]][1,
  1], gr.wd25[[1]][1, 1], gr.wd50[[1]][1, 1], gr.we50[[1]][1,
  1]), gr.par2 = c(gr.a[[1]][2, 1], gr.nw[[1]][2, 1],
  gr.ma[[1]][2, 1], gr.wd25[[1]][2, 1], gr.wd50[[1]][2,
  1], gr.we50[[1]][2, 1]))
grMetrics.y = rbind(grMetrics.y, t.gr)
}

```

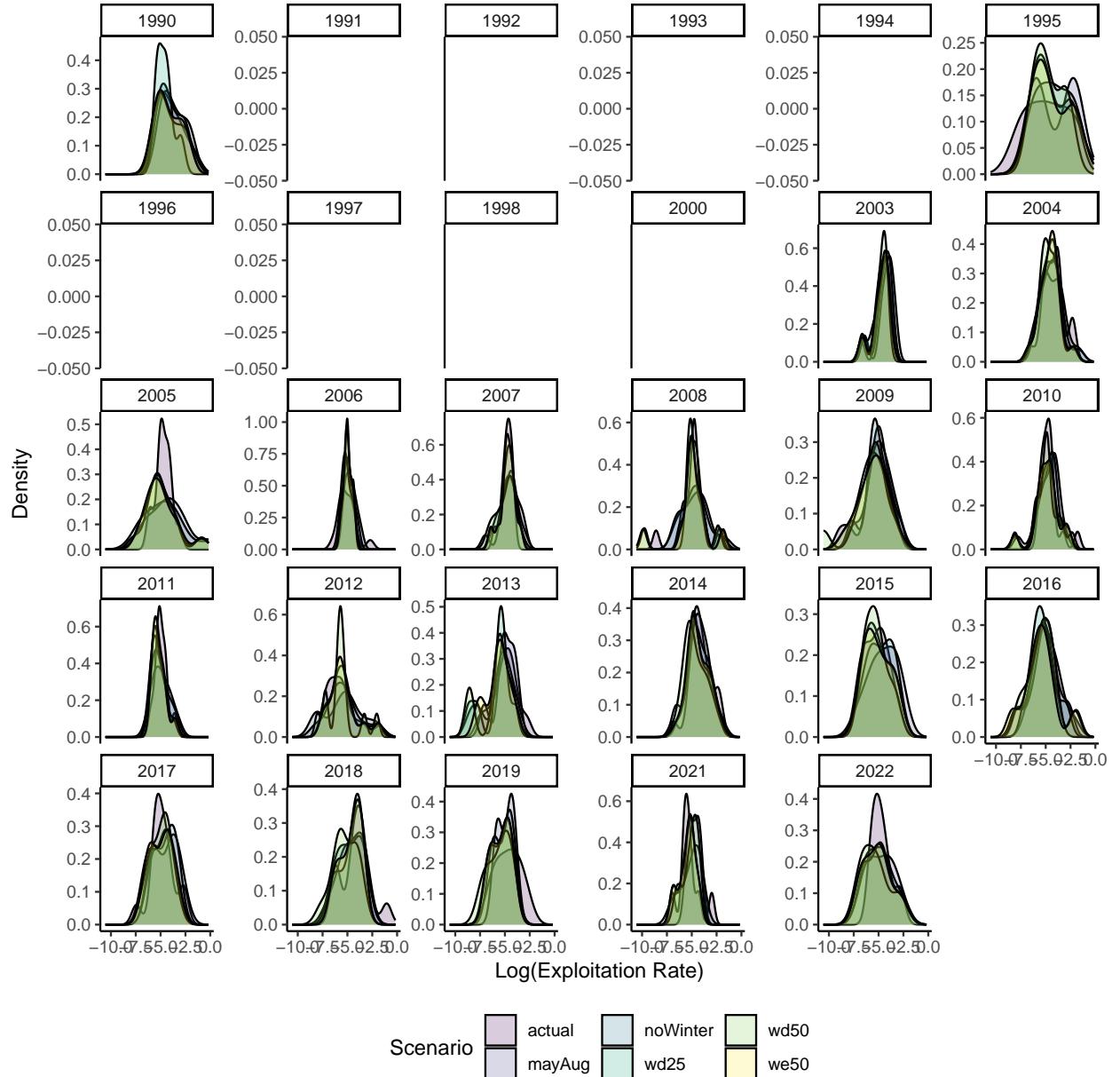


Figure 11: Distribution of exploitation rates across years in the creel data set. Different data reduction scenarios are noted with varying colors.

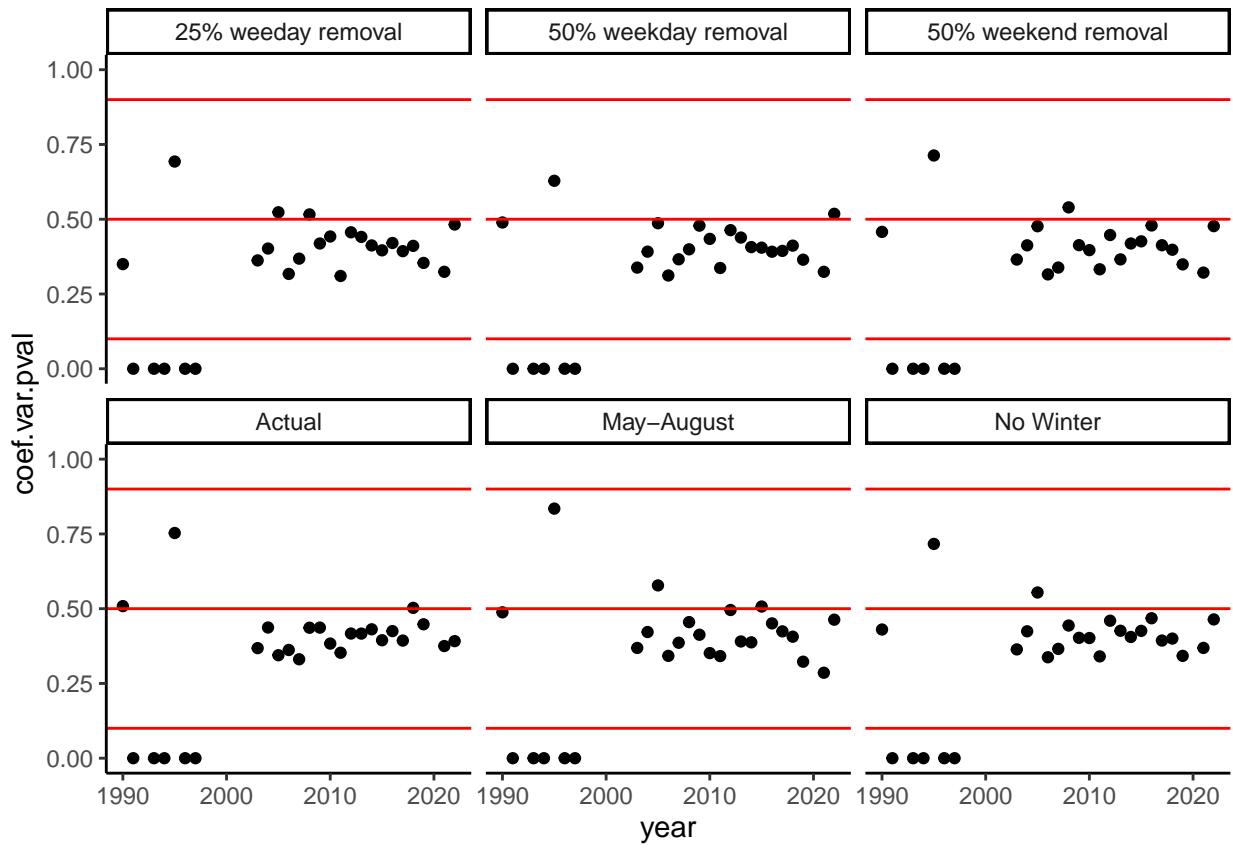
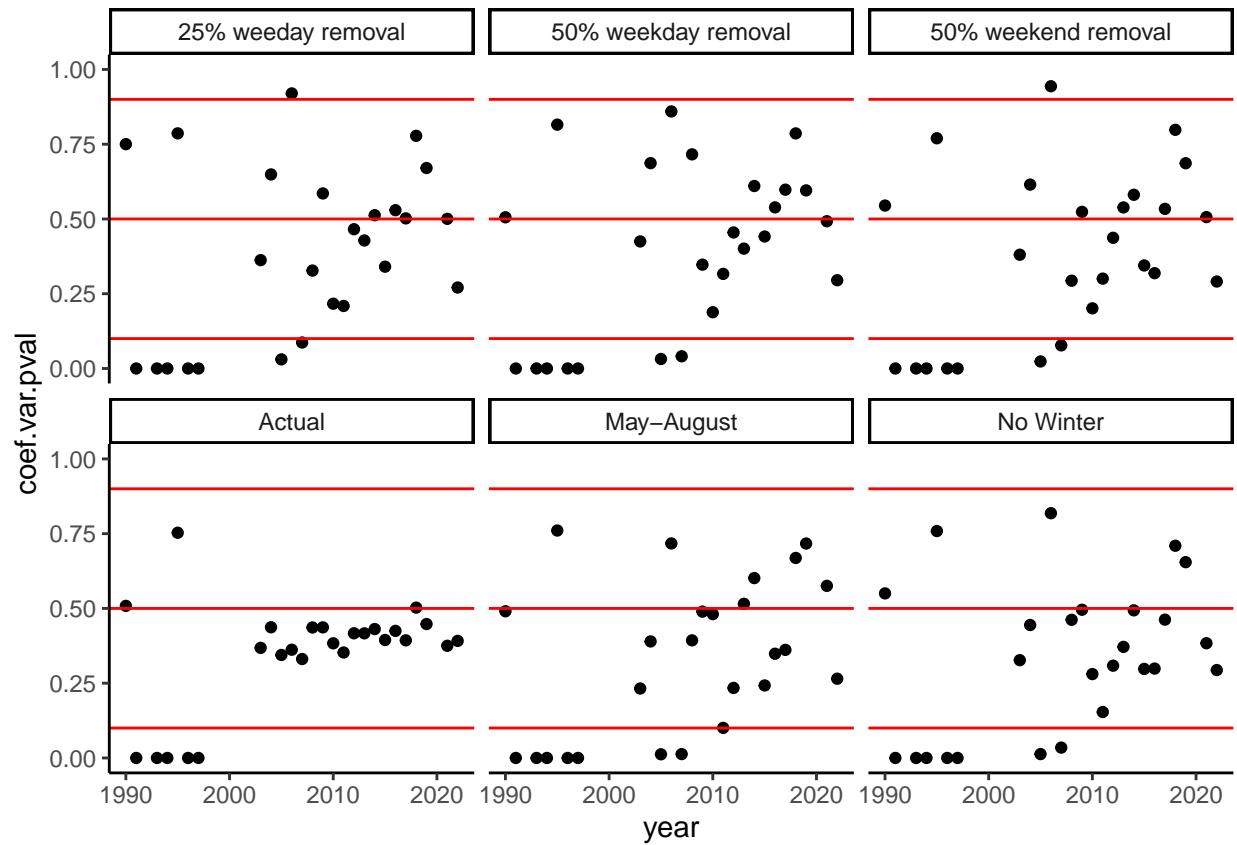
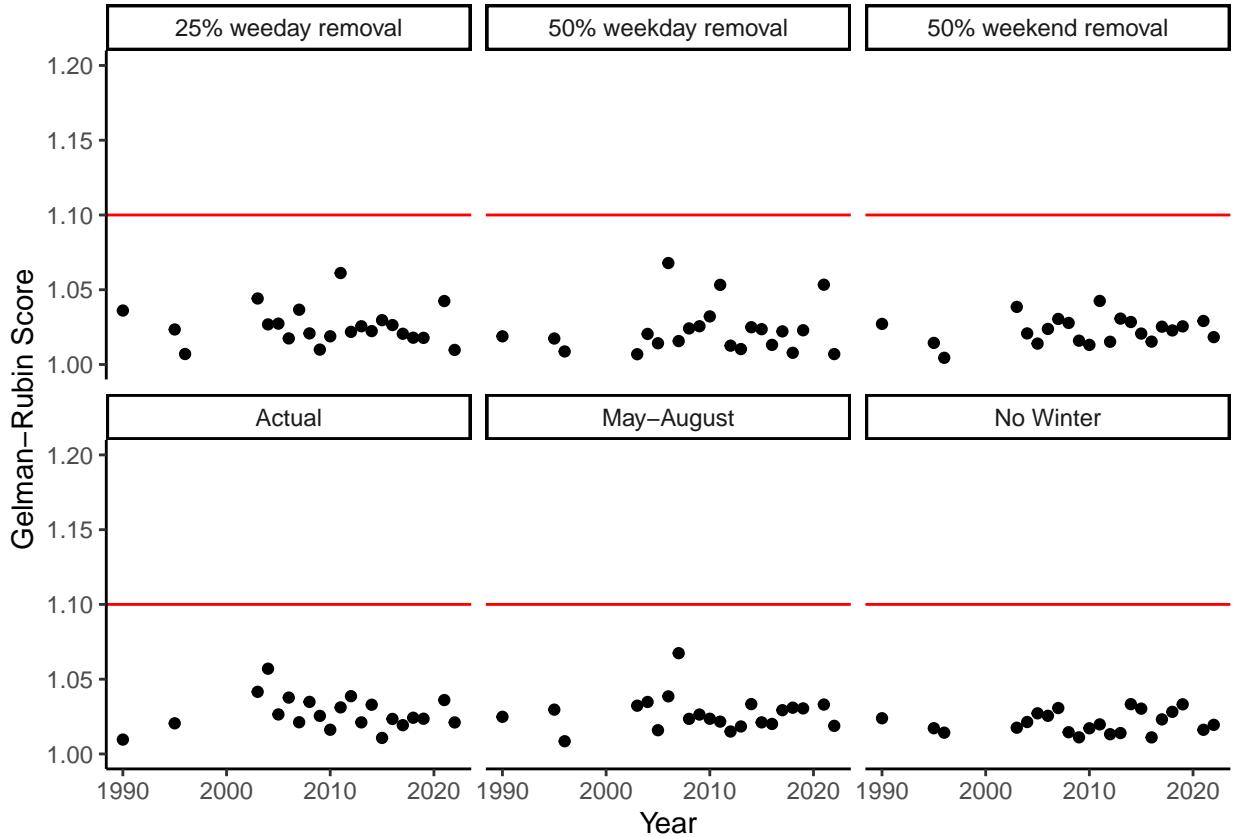


Figure 12: Comparison of coefficient of variations for bayesian p-values comparing the actual data to the simulated data for the creel year and data reduction scenario.





The results of these creel-year specific analyses suggest that even on an individual creel year level, the estimated exploitation rate from the data-reduced scenario generally is not significantly different from the exploitation rate calculated from the full data set (Figure ??). Compared to the same analysis for the full population of creel data all at once instead of on a year-by-year basis, the estimated exploitation rates from the reduced data here don't match the actual quite as well but the majority of years still fall within the typical bounds of acceptance for this test. All models do appear to fit well and converge (Figure ??).

There did not appear to be any obvious characteristics of the lakes themselves or their walleye populations that correlated with the magnitude of the difference between the actual estimated exploitation rate and the reduced-data estimation. Walleye recruitment code (Figure 7), angler effort density (Figure 9), lake size (Figure 8), and lake class (Figure 10) were all examined and showed no clear trend. This may be in large part due to the relatively small differences between each estimate of exploitation rate and the reduced-data estimate that could make it hard to see an effects, which would further point towards there being minimal effect of data reductions on exploitation rate estimate even at the individual creel-year level.

1.3.5 Important Caveats

Here are the important caveats for this analysis. These are things that I would expect another researcher to raise as potential weak points of this work. Numbers 1 and 2 can be addressed by me in consultation with other researchers and the scientific literature. Numbers 3 and 4 are probably better informed by the legal context of the situation. These don't have 'right' answers in the way that the other two do. Here the 'right' answer will be values-based and likely depends on what the group, or legal system, decides.

1. How to deal with exploitation rate estimates equal to 0. I have outlined my approach to this above and the rationale behind it, but that doesn't mean someone else might not pick at that and say it's a flaw. One way to address this would probably be to model the data using a beta or gamma distribution instead of a lognormal distribution. This would allow the 0s to be included.

2. Accuracy measure. In order to compare the exploitation rate estimates from the reduced data to the actual data I used 1 standard deviation as my measure. This means that a reduced-data exploitation rate estimate that is within 1 SD of the actual data exploitation rate estimate was considered to be ‘the same’. There may be better metrics to use here, and different people may have different opinions about the best metric to use. My rationale for using this metric has been outlined above.
3. Bayesian p-value metrics. when calculating the Bayesian p-value to understand if the data from the reduced scenarios resembled that of the actual data I used the coefficient of variation and standard deviation. In reality any variance metric could be used to compare the two data sets. I chose these two because I felt that were intuitive enough for anyone to understand and defensible. But that may be others that should be explored and the results of the p-value tests may differ based on the metric used.
4. Related to the p-value metric are the cutoffs to use to decide whether there is a meaningful difference between the actual exploitation rate and the data reduced rate. The rule of thumb for this test is values <0.1 or >0.9 indicate a significant difference between the actual data estimate and the reduced data estimate. In reality it is up to the user to decide what those cutoffs should be based on the context of their problem. Whether or not we consider the reduced data exploitation rates similar or different from the actual can obviously be influenced by the cutoff numbers we chose.

1.4 Appendix

1.4.1 Alternative modeling distributions

Instead of the lognormal distribution there are two other probability distributions that could be used to model this data based on the characteristics of the data and the data characteristics assumed by each of the distributions.

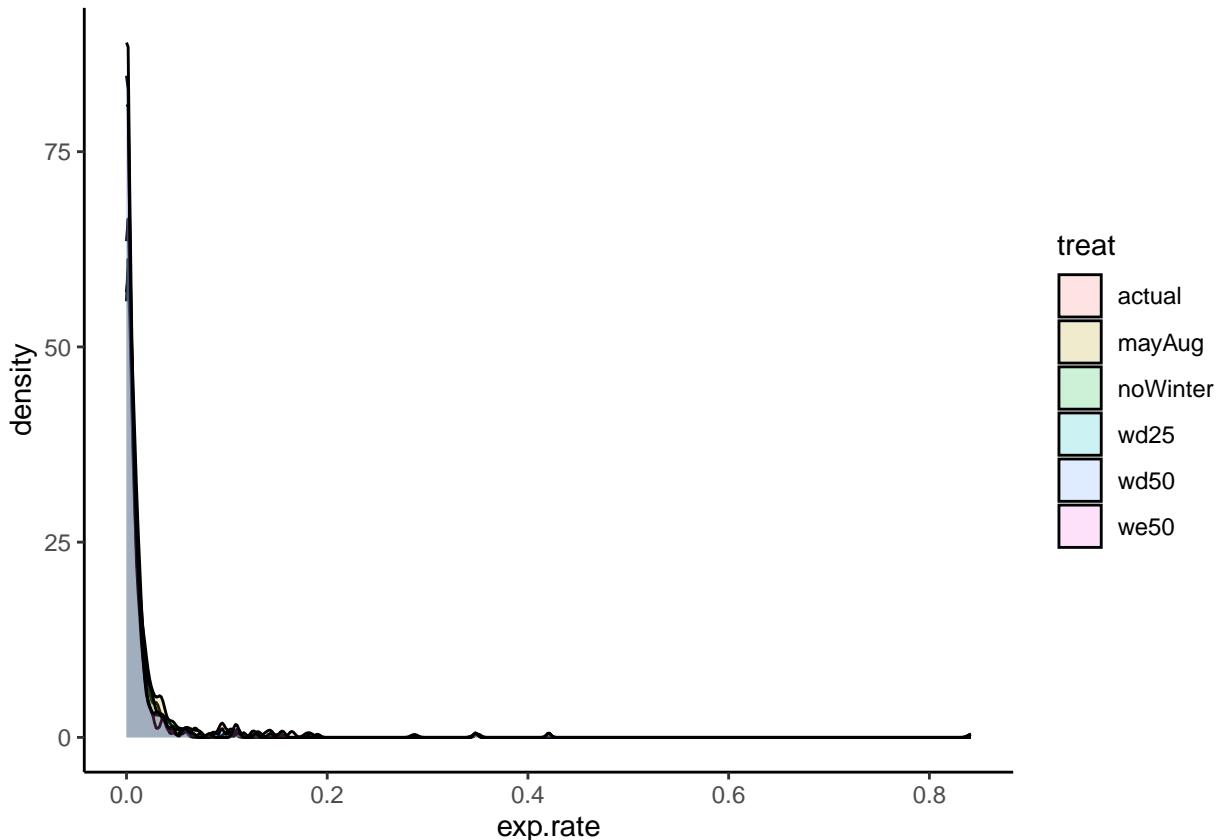
Table of data distributions and their definitions:

Distribution	Definition
Lognormal	Continuously distributed quantities with nonnegative values. Random variables with the property that their logs are normally distributed.
Gamma	Any continuous quantity that is nonnegative. Continuous version of a Poisson distribution.
Beta	Continuous random variable that can take on values between 0 and 1. Any random variable that can be expressed as a proportion.

The following analyses will compare the ability of models using the lognormal and beta distributions to fit the full data and each of the reduced data set. The gamma distribution is not included in this analysis. A cursory look at the ability of gamma distributed models showed no real difference from beta and lognormal and the definition of the beta better represents the data we’re dealing with than the gamma definition so further analyses with this family of models was not pursued (Figure 13). The purpose of the comparison between lognormal and beta family models is not to see which model provided the most convenient answer but to compare their ability to fit the data. All comparisons presented in this analysis are comparing a model’s ability to fit the data **within** each dataset and **not** across data sets which is what is necessary to gain inference on the effect of any data reductions.

Table 6: Bayesian p-values for a beta distributed model. Each p-value describes whether or not there is a significant difference between data generated by the fitted model and the actual data for that scenario. Two metrics are assessed here, coefficient of variation (CV) and standard deviation (SD).

Scenario	CV p-value	SD p-value
Actual	0.000	0.001
No Winter	0.001	0.528
May-August	0.001	0.739
25% weeday removal	0.000	0.000
50% weekday removal	0.000	0.006
50% weekend removal	0.000	0.007



Gelman-Rubin convergence diagnostics indicate all models have converged with the exception of the 25% weekday reduction scenario. This model did not converge and the results from it should not be trusted. I've been through the code a few times and can't find any errors so I think this lack of convergence is legitimate.

```
### MODEL CHECKNG ####

gelmanDiagnostics(expR.actual.beta, plot = T) # 1.05 converged
gelmanDiagnostics(expR.nw.beta, plot = T) # 1.02 converged
gelmanDiagnostics(expR.ma.beta, plot = T) # 1.03 converged
gelmanDiagnostics(expR.25wd.beta, plot = T) # 25.8 NOT converged
gelmanDiagnostics(expR.50wd.beta, plot = T) # 1.02 converged
gelmanDiagnostics(expR.50we.beta, plot = T) # 1.02 converged
```

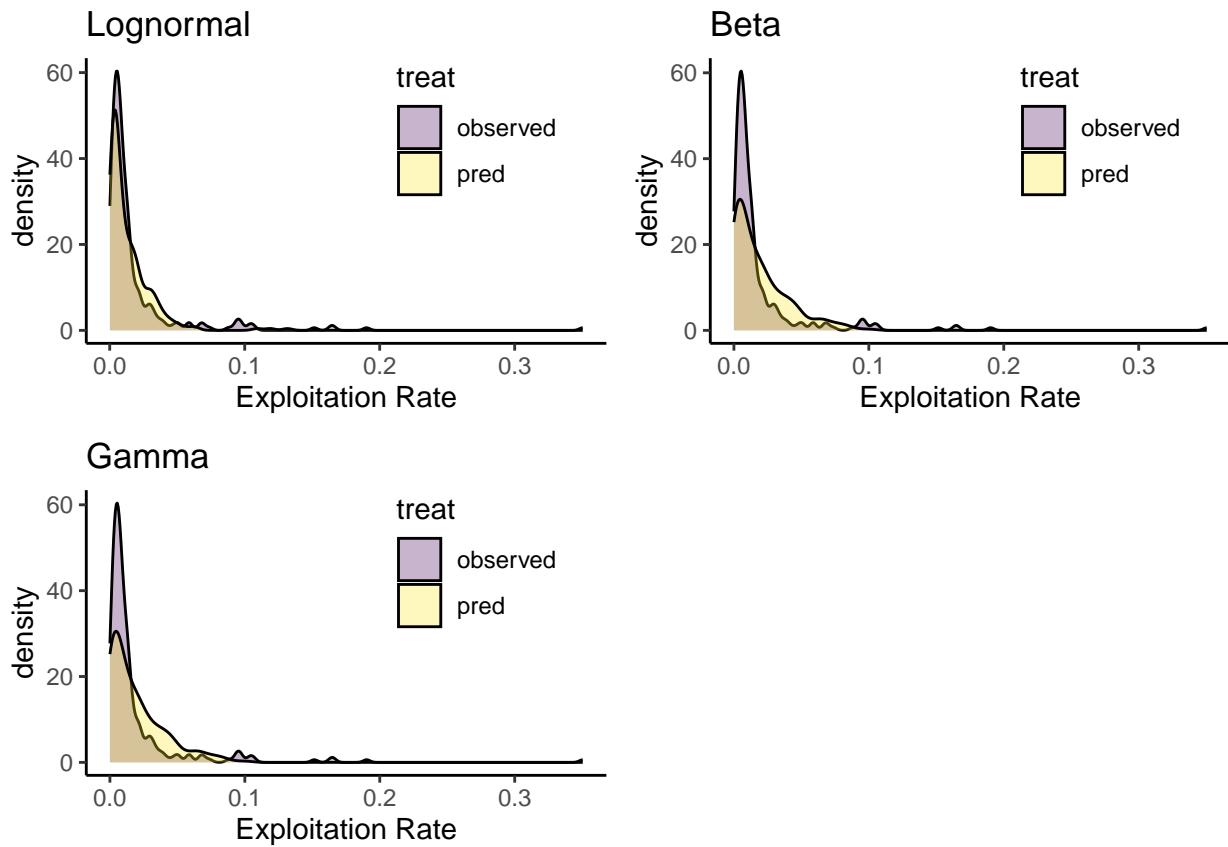


Figure 13: Comparison of model fits for lognormal, gamma, and beta family of models. This rough look at the models suggests there are no obvious differences between the choice of modeling distribution. This will be evaluated statistically using bayesian p values later on. Based on what I see here I've dropped the gamma family of models from further analysis since the definition of gamma data is not really different from the lognormal data and beta better fits the data description than gamma.

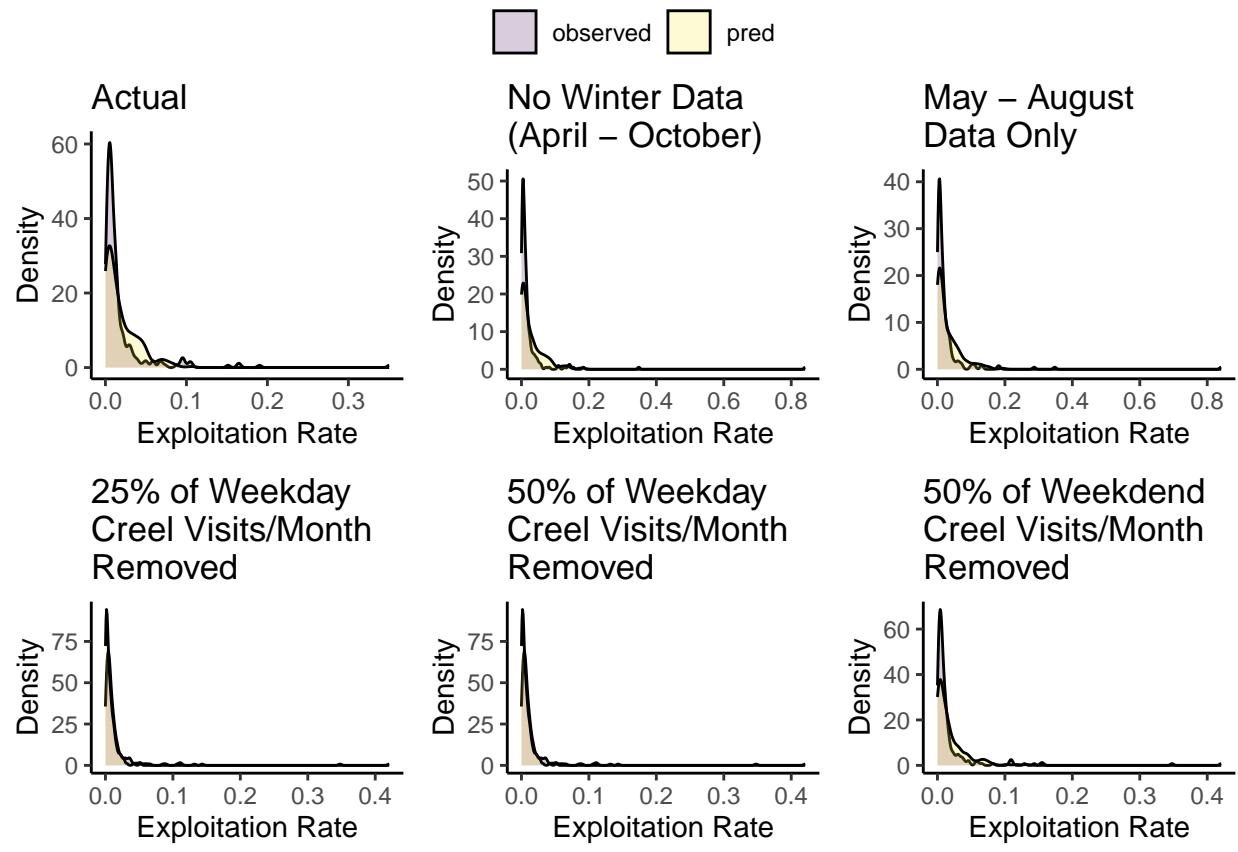


Figure 14: Visualization of the model fit to the data for each scenario using a beta distribution instead of a lognormal distribution.

The Bayesian p-values in the table presented here suggests the beta models do a poor job of reproducing the data they were fit to (Table 6). This inference comes from the lack of p-values between 0.1 - 0.9 for the CV metric and the majority of p-values being outside the 0.1-0.9 range for the SD metric too. The abundance of significant p-values across model scenarios and metrics points to the inability of models fit to the creel data using a beta distribution to be able to reproduce that data. Because each model is unable to reproduce the data it was fit to, we know that this family of models is not a useful family for fitting these data. This means that the beta distribution not likely to be appropriate for making comparisons between the actual data and the reduced data to understand whether or not creel effort reductions result in significantly different exploitation rates or not.

1.4.2 P-value sensitivity analysis

Another important area of uncertainty in this analysis is the choice of metric to calculate Bayesian p-values for. In the main text of this document I've chosen the coefficient of variation and standard deviation as two metrics to assess. However any metric could be chosen as long as the choice can be justified in the context of the data and question at hand.

In this analysis I've chosen 3 additional test statistics to compare alongside the CV and SD. The test statistics used are summarized in Table ???. Here I've fit the models using the lognormal distribution as above and then calculated Bayesian p-values for each of the 5 test statistics and 6 data scenarios. I've also done a separate test using the χ^2 goodness of fit test to see how often the modeled data is significantly different from the observed data for each of the parameter values in the Markov-Chain Monte Carlo output from the model fitting algorithm.

1.4.2.1 Interpreting each statistic Some more detail on what I'm looking for in each statistic and what the values mean that are used to calculate each Bayesian p-value. It's important to remember that the way the sampling algorithm works in this Bayesian framework the frequency of values in the MCMC output is relative to their support in the data. In other words, parameter values that produce better models fits show up in the chain more often. This is critical to understanding the Bayesian p-values here because each test statistic is calculated for simulated data based on the MCMC output so each test statistic value should appear with a frequency equal to the support provided by the data too. For example, the medians calculated from data sets produced by each parameter set in the MCMC output should produce medians that have frequencies equal to how well those parameters fit the data. Thus medians well below the median of the observed data should be pretty rare (and likewise for medians well above) because the parameter set from the MCMC chain that produces data with said median should be relatively infrequent in the MCMC output. Medians close to the median of the observed data should be more common, if the model is fitting the data well, because the parameters generating those means are more common in the MCMC output due to the better fit to the data they provide. This results in medians that are often close to the median of the observed data and by random chance should be just as likely to be above as below the median of the observed data. This is why Bayesian p-values close to 0.5 signify a good model fit, because about 50% of the time the test statistic is more extreme than the value for the observed data. If this values is extremely high or low then is signals that our model is not adequately representing the observed data.

1.4.2.1.1 Coefficient of Variation Straightforward, the CV is calculated for a data set simulated from each set of parameters in the MCMC chain from the model fit. Each dataset is simulated by drawing from a random lognormal distribution parameterized according to the values at a given place in the MCMC chain. The CV value from each of these simulated datasets is then compared to the CV for the observed data for a given scenario (Actual, No Winter, May-August, etc.). Whether the CV for the simulated data is greater than or less than the CV of the observed data is recorded. **If the model is able to reproduce the observed data well then the CV of the simulated data should be greater than the CV of the observed data about 50% of the time and less than the observed CV about 50% of the time too.**

1.4.2.1.2 Standard Deviation Another straightforward calculation. SD is calculated for a data set drawn from a random lognormal distribution parameterized using the values in the MCMC chain. This is repeated for each set of parameter values in the MCMC chain to create as many simulated data sets as there are iterations in the MCMC chain. The SD of each of these simulated data sets is compared to the SD for the observed data for that scenario (Actual, No Winter, May - August, etc.). Whether the SD of the simulated data is greater than or less than the SD of the observed data is recorded. **If the model is able to reproduce the observed data well then the SD of the simulated data should be greater than the SD of the observed data about 50% of the time and less than the SD of the observed data about 50% of the time too.**

1.4.2.1.3 Median Simple descriptive statistic here. The median is calculated for each simulated data set produced by drawing values from a random lognormal distribution parameterized using the values in the MCMC output. The number of times the median of the simulated data set exceeds the median of the observed data is calculated and that proportion of times the test statistic exceeds the value for the observed data is the Bayesian p-value. **If the model is adequately representing the data then the medians generated from the simulated data sets should exceed the median of the observed data roughly 50% of the time.**

1.4.2.1.4 Kurtosis This metric essentially measures the shape of a distribution, specifically the tails. Kurtosis is calculated for the simulated data for each set of parameter values in the MCMC output and compared to the kurtosis of the observed data for the given scenario (Actual, No Winter, May - August, etc.). When the kurtosis value of the simulated data is greater than the kurtosis of the observed data this signals more frequent extreme values in the data. In other words the tails of the distribution of the simulated data are thicker than the tails of the distribution of the observed data. The opposite is true when the kurtosis of the simulated data is less than the kurtosis of the observed data. **A well-fitting model will produce data with kurtosis values similar to the the observed data's kurtosis. The kurtosis of the simulated data will exceed that of the observed data roughly 50% of the time. If the kurtosis of the simulated data is more frequently less than the observed data this would signal that the model is not doing well as representing the rare values in the tails of the data distribution. The opposite is true if kurtosis values for the simulated data tend to be larger than the observed data.**

1.4.2.1.5 Chi Square Test (χ^2) This test is performed outside the Bayesian p-value framework. Instead, each simulated data set arising from a set of parameters in the MCMC output is compared to the probability distribution of the observed data to ask whether the simulated data arises from the same distribution as the observed data. Again, because of the way the parameter space is sampled in MCMC algorithms the values that produce better model fits will be more common which means that in the χ^2 test the null hypothesis that the data comes from the supplied probability distribution should be accepted more often than it's rejected. **If the null hypothesis is rejected the majority of the time that would indicate that the model is not adequately representing the data if it can't produce simulated data that would appear to be from the probability distribution of the observed data used to fit the model in the first place.**

1.4.2.1.6 Fisher 'F' Statistic The F statistic is the ratio of variances between the simulated and observed data sets. For this test I'm looking to see if the F statistic for the simulated:observed comparison is greater than or less than 1 (the ratio of the variance for the observed:observed comparison). **The closer the F statistic is to 1 for the simulated:observed comparison the better job the model is doing at adequately representing the data. So the number of times the F statistic exceeds 1 should, if the model is fitting the data well, be about 50% of the time. If the Bayesian p-value is much higher than 0.5 that would indicate that the variance of the simulated data is often greater than the variance of the observed data. The opposite then is true when the p-value is smaller than 0.5**

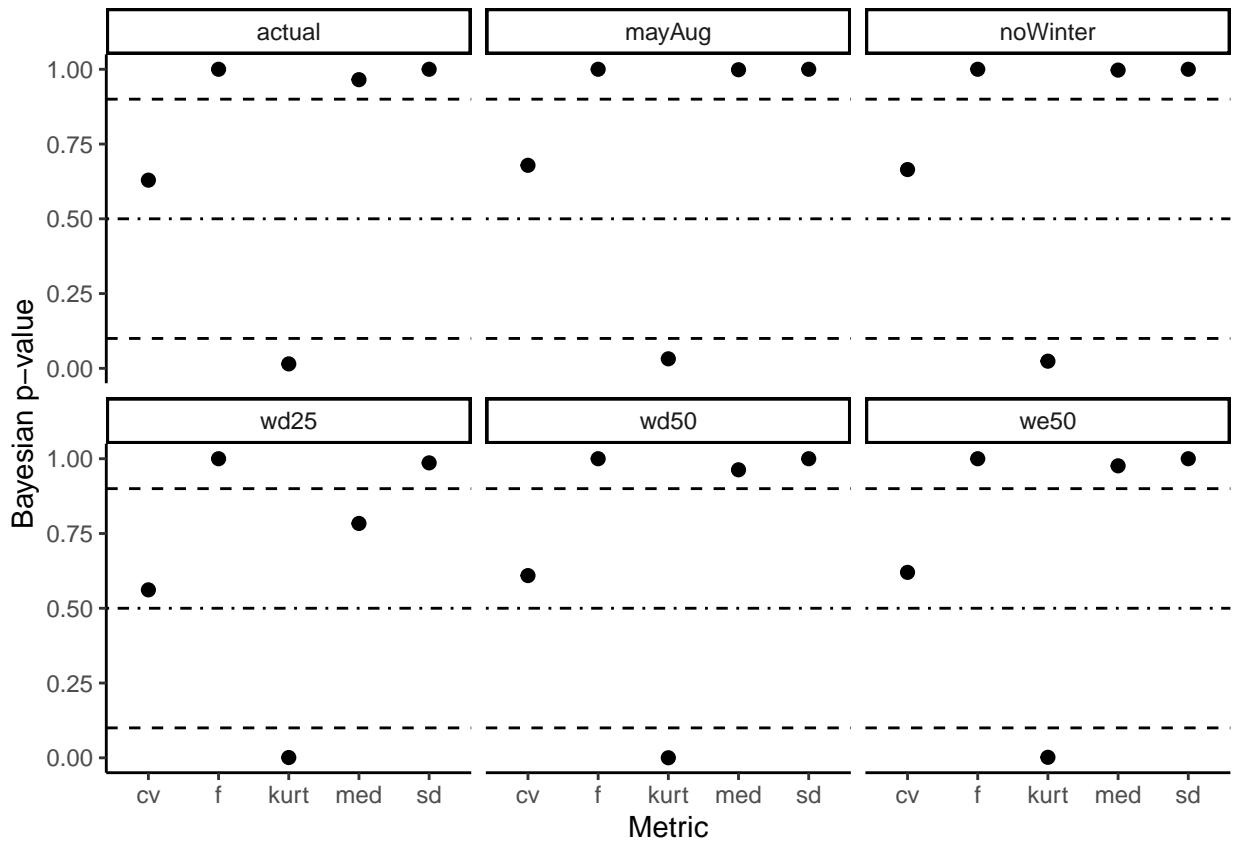


Figure 15: Comparison of multiple metrics for calculating Bayesian p-values. Each panel is a different data scenario. The comparisons in each panel are comparing the model simulated data to the observed data for that same scenario, NOT comparing model simulated data from a reduction scenario to the actual data. This comparison to self is to understand whether the model is adequately representing the data and whether the choice of statistic influences the answer to that question. Values close to 0.5 are ideal, values more extreme than 0.1 or 0.9 are considered evidence for a poor model fit (i.e. significant difference between model simulated data and the observed data).

1.4.3

Figure 15 shows the p-values resulting from each metric for each data scenario. Aside from kurtosis, the p-values are quite similar across metrics within a specific data scenario. This suggests that while kurtosis may not be a good choice here, the choice between f statistic, cv, sd, or median should have minimal impact on the inference drawn from these models. Digging into kurtosis a bit more, it's not immediately clear why so many of the simulated data sets have thinner tails than observed data. This may have to do with the random sampling function `rlnorm()` which samples randomly from a distribution informed by the MCMC output parameters (as the mean and standard deviation respectively). It may be that the variance term in the function is a tad low and that's preventing some of the outliers from being chosen and creating those thicker tails that are seen in the observed data.

Finally, the choice of 0.1 and 0.9 as the cutoff values for significance here do not seem to influence the outcomes. A narrower range could be chosen and, unless it were extremely narrow (e.g. 0.4-0.6), the p-values here would all still indicate our models are accurately representing the observed data for all measures except kurtosis.