I tried simple replacement of those cells with “3”, two different multiple imputation methods, and left-censored Poisson regression based on survival methods. I tested those methods on a single two-way simulated cross-tab of the counts of three different types of animals in four different regions, with two suppressed cells.

An obvious piece of unfinished business is to extend the comparison I made to many different randomly generated tables. I have now done this, and I’ve also extended out the methods I’m comparing to cover three different multiple imputation methods and four different simple substitutions (with 0, 1, 3 and 5).

Because my tables are generated by a data generating process under my control, I know the true values of coefficients for the fully saturated Poisson family generalized linear model that is being used to analyse each table with each imputation method. So I can compare the true coefficients with those for each method, including using the full observed data (as though you had access to it, perhaps because you work for the agency that suppressed the data in the first instance). Note – if a fully saturated Poisson family generalized linear model sounds complicated, you can remember that it is the Chi square test for independence of variables in a cross tab which is usually one of the first two or three tools taught in a basic statistics methods class.

**Some surprising results**

It’s fair to say the results surprised me.

* The best of the nine methods was simply to replace all values below six with the number 5
* Four of the methods delivered better results on average in uncovering the true data generating process than did *fitting the model to the real, unsuppressed data*
* My multiple imputation that provides numbers from zero to five with probabilities based on a truncated Poisson distribution is the best of the fancy methods, but only very slightly better than the default imputation from the mice package, which allows imputed values to be greater than 5 even though we know the reality wasn’t that big
* left-censored Poisson regression with survival analysis methods still performs not particularly well.

My simulations were of tables from two to ten columns and two to ten rows. All the true coefficients for effect sizes including interactions were from a uniform distribution between -1 and 1. The intercept was drawn from a uniform distribution from 1 to 3. The intercept was excluded from the calculation of the mean square error of coefficients, because it’s usually not of substantive interest and because it is on a different scale to the other coefficients.

Looking at the relationship of error in coefficients to the size of the table and to the proportion of cells that were suppressed for being below six, we get these charts:

There are some interesting findings here too:

* all methods perform worse as the proportion of cells under six increases, but for the poorer performing methods this is a stronger effect
* all methods perform worse with more coefficents to estimate, but this is strongest for the method using the unsuppressed data
* the unsuppressed data is clearly bimodal, with a cluster of poorly performing models well separated from an otherwise good performance.

**Unusual behaviour of model with the model with full unsuppressed data**

This last point about bimodality of performance of the model with the full unsuppressed data is particularly interesting. It wasn’t obvious in my first boxplot, because while that’s a nice simple comparison graphic it will usually hide bi-modality in a distribution. Here’s density plots instead:

It’s worth having a closer look at this. Unlike the other methods, using the full data normally performs exceptionally well (as would be expected) but sometimes extremely badly. Is there any pattern? Let’s look at a plot of the mean squared error across the size of the cross tab and the proportion of cells that were suppressed, just for the method that uses the full data despite suppression:

There’s a definite pattern that the more data were below 6, the worse this model does, but there are still occasions even when half the data are below 6 the model does ok (small reddish dots).

Luckily I had written the function that compares the models so it can be reproducible and provide the full set of coefficients. I had a look at one run that was particularly poor performing for the full data model, number 125. Here is the actual data for run 125:

| **animals** | **region** | **count** | **expected** |
| --- | --- | --- | --- |
| a | A | 0 | 4.9508590 |
| b | A | 5 | 3.7161291 |
| c | A | 6 | 12.5528917 |
| d | A | 17 | 12.6124215 |
| e | A | 2 | 5.2896170 |
| f | A | 1 | 3.5390988 |
| g | A | 10 | 6.7345340 |
| h | A | 6 | 6.2255400 |
| i | A | 3 | 1.8735223 |
| a | B | 7 | 8.3839561 |
| b | B | 5 | 2.8352860 |
| c | B | 20 | 23.8147302 |
| d | B | 21 | 18.3545554 |
| e | B | 6 | 7.6222396 |
| f | B | 14 | 14.2599204 |
| g | B | 18 | 16.0890494 |
| h | B | 17 | 17.6865488 |
| i | B | 2 | 2.4653642 |
| a | C | 5 | 3.7365829 |
| b | C | 1 | 4.4282008 |
| c | C | 15 | 13.6843526 |
| d | C | 6 | 6.3213678 |
| e | C | 7 | 4.6640287 |
| f | C | 4 | 1.1177665 |
| g | C | 9 | 8.5946791 |
| h | C | 1 | 2.2103981 |
| i | C | 0 | 0.7286345 |

… and here are the coefficients from all the models, compared to the true coefficients of the underlying data generating process:

So that’s interesting. It’s revealing that the full data model performs badly in a similar way to the absurb “replace all suppressed cells with zero” model. What’s going on here is that the observed data has an unusual number of very low values (eg a-A-0, f-A-1, b-C-1). The low values lead to a very unstable estimate of our Poisson regression model, and coefficients leap up to +25 and -25 (when the real values are between -1 and 1). This happens in enough runs that these wildly underperforming models totally mess up the record of the full data model.

What we’re seeing here is that this sort of model – a Poisson family GLM fit to a cross tab – performs erratically (and potentially delivers ridiculous results) when lots of counts are low. Which of course has been known since the early days of the Chi square test and the warning not to use it when expected counts are less than five.

A probable solution is clear – some kind of shrinkage of coefficient estimates, either through regularization if we’re a frequentist or a more sensible prior if we’re a Bayesian. Then I suspect the full data model would perform quite adequately.

**Code**

Here’s the R code for today’s exercise, all in one chunk:

library(tidyverse)

library(mice)

library(VGAM)

library(Cairo)

library(parallel)

#------------------ imputation functions----------

#' Imputation function for suppressed data for use with mice - Poisson-based

#'

#' @param y vector to be imputed

#' @param ry logical vector of observed (TRUE) and missing (FALSE) values of y

#' @param x design matrix. Ignored but is probably needed for mice to use.

#' @param wy vector that is TRUE where imputations are created for y. Not sure when this is different

#' to just !ry (which is the default).

mice.impute.desuppress <- function (y, ry, x, wy = NULL, max\_value = 5, ...) {

# during dev:

# y <- data$censored\_count\_num; ry <- !is.na(y)

if (is.null(wy)){

wy <- !ry

}

# What are the relative chances of getting values from 0 to the maximum allowed value,

# if we have a Poisson distribution which we have estimated the mean of via trimmed mean?

# (this is very approximate but probably better than just giving equal chances to 0:5)

probs <- dpois(0:max\_value, mean(y[ry], tr = 0.2))

return(sample(0:max\_value, sum(wy), prob = probs, replace = TRUE))

}

#' Imputation function for suppressed data for use with mice - simple

#'

mice.impute.uniform <- function (y, ry, x, wy = NULL, max\_value = 5, ...) {

if (is.null(wy)){

wy <- !ry

}

return(sample(0:max\_value, sum(wy), replace = TRUE))

}

#--------------main function---------------

# This function generates data and compares coefficients

coefs\_comp <- function(run, output = c("summary", "all")){

set.seed(run)

output <- match.arg(output)

data <- expand.grid(animals = letters[1:sample(2:10, 1)],

region = LETTERS[1:sample(2:10, 1)],

count = 1)

n <- nrow(data)

mm <- model.matrix(count ~ animals \* region, data = data)

true\_coefs <- c(runif(1, 1, 3), runif(n - 1, -1, 1))

data <- data %>%

mutate(expected = exp(mm %\*% true\_coefs),

count = rpois(n, lambda = exp(mm %\*% true\_coefs)),

censored\_count = ifelse(count < 6, "<6", count),

censored\_count\_num = suppressWarnings(as.numeric(censored\_count)),

count\_replaced\_5 = ifelse(count < 6, 5, count),

count\_replaced\_3 = ifelse(count < 6, 3, count),

count\_replaced\_1 = ifelse(count < 6, 1, count),

count\_replaced\_0 = ifelse(count < 6, 0, count))

data$z <- pmax(6, data$count)

data$lcensored <- is.na(data$censored\_count\_num )

prop\_suppressed <- mean(data$lcensored)

data$which\_complete <- as.numeric(!data$lcensored)

if(prop\_suppressed > 0.5){

return(NULL)

} else {

#--------------straightforward methods-----------

# with the full unsuppressed data, as though you were the ABS:

mod0 <- glm(count ~ animals \* region, data = data, family = "poisson")

# with replacing the suppressed data with 3 or 5:

mod1 <- glm(count\_replaced\_3 ~ animals \* region, data = data, family = "poisson")

mod2 <- glm(count\_replaced\_5 ~ animals \* region, data = data, family = "poisson")

mod2a <- glm(count\_replaced\_1 ~ animals \* region, data = data, family = "poisson")

mod2b <- glm(count\_replaced\_0 ~ animals \* region, data = data, family = "poisson")

#------------with censored poisson regression-------------------------

mod3 <- vglm(SurvS4(z, which\_complete, type = "left") ~ animals \* region,

family = cens.poisson, data = data)

#------------with multiple imputation

# number of sets of imputations to make

m <- 20

# See https://github.com/stefvanbuuren/mice/issues/150 for why remove.collinear = FALSE.

data\_imp1 <- mice(data[ , c("censored\_count\_num", "animals", "region")],

method = "desuppress", print = FALSE, m = m,

remove.collinear = FALSE)

data\_imp2 <- mice(data[ , c("censored\_count\_num", "animals", "region")],

method = "uniform", print = FALSE, m = m,

remove.collinear = FALSE)

# default mice method:

data\_imp3 <- mice(data[ , c("censored\_count\_num", "animals", "region")],

print = FALSE, m = m,

remove.collinear = FALSE)

mod\_mice1 <- with(data\_imp1, glm(censored\_count\_num ~ animals \* region, family = "poisson"))

coef\_mice1 <- pool(mod\_mice1)$pooled$estimate

mod\_mice2 <- with(data\_imp2, glm(censored\_count\_num ~ animals \* region, family = "poisson"))

coef\_mice2 <- pool(mod\_mice2)$pooled$estimate

mod\_mice3 <- with(data\_imp3, glm(censored\_count\_num ~ animals \* region, family = "poisson"))

coef\_mice3 <- pool(mod\_mice3)$pooled$estimate

#---------------results----------------------

# comparison data

d <- data\_frame(underlying = true\_coefs,

`Using full data including suppressed values (not usually possible!)` = coef(mod0),

`Replacing suppressed values with 0` = coef(mod2b),

`Replacing suppressed values with 1` = coef(mod2a),

`Replacing suppressed values with 3` = coef(mod1),

`Replacing suppressed values with 5` = coef(mod2),

`Left-censored survival-based method` = coef(mod3),

`MICE with Poisson proportional probabilities` = coef\_mice1,

`MICE with uniform probabilities` = coef\_mice2,

`MICE with default imputation` = coef\_mice3,

labels = names(coef(mod0))) %>%

mutate(labels = gsub("animals", "", labels),

labels = gsub("region", "", labels)) %>%

gather(method, value, -underlying, -labels) %>%

mutate(value = ifelse(is.na(value), 0, value))

# summary data:

d2 <- d %>%

# the intercept is on a different scale to all the other coefficients

filter(labels != "(Intercept)") %>%

mutate(square\_error = (value - underlying) ^ 2) %>%

group\_by(method) %>%

summarise(mse = mean(square\_error),

trmse = mean(square\_error, tr = 0.2)) %>%

ungroup() %>%

mutate(method = fct\_reorder(method, mse)) %>%

arrange(trmse) %>%

mutate(method = fct\_reorder(method, trmse)) %>%

mutate(run = run,

prop\_suppressed = prop\_suppressed,

n = n)

if(output == "summary"){

return(d2)

}

if(output == "all"){

return(list(

all\_coefficients = d,

performance\_summary = d2,

data = data

))

}

}

}

#--------------------do the runs----------------

# It was useful to do this in a single-threaded way during dev to locate problems such

# as with run 357, but when that was sorted it was better to dso the experiments with

# parallel processing.

# set up cluster with number of cores minus one

cores <- detectCores()

cluster <- makePSOCKcluster(max(1, cores - 1))

# set up the functionality on the cluster

clusterEvalQ(cluster, {

library(dplyr)

library(tidyr)

library(forcats)

library(mice)

library(VGAM)

})

clusterExport(cluster, c("coefs\_comp", "mice.impute.desuppress", "mice.impute.uniform"))

# Generate data for a sequence of seeds and compare the coefficient from the various methods

results <- parLapply(cluster, 1:1000, coefs\_comp)

stopCluster(cluster)

#--------------------present results-------------------

results\_df <- do.call("rbind", results) %>%

mutate(method = fct\_reorder(str\_wrap(method, 27), mse))

summary(results\_df)

results\_df %>%

ggplot(aes(x = method, y = mse)) +

geom\_boxplot() +

scale\_y\_log10() +

coord\_flip() +

labs(y = "Mean squared error of coefficients, compared to true data generating process",

x = "") +

ggtitle("Fitting a Poisson GLM to a cross tab of counts",

"Comparison of methods of dealing with suppressed counts under 6")

results\_df %>%

ggplot(aes(x = prop\_suppressed, y = mse)) +

facet\_wrap(~method) +

geom\_point(size = 0.3) +

scale\_y\_log10() +

geom\_smooth(method = "lm") +

labs(y = "Mean squared error of coefficients,\ncompared to true data generating process",

x = "Proportion of cells in original data that are under 6") +

ggtitle("Fitting a Poisson GLM to a cross tab of counts",

"Comparison of methods of dealing with suppressed counts under 6")

results\_df %>%

ggplot(aes(x = n, y = mse)) +

facet\_wrap(~method) +

geom\_point(size = 0.3) +

scale\_y\_log10() +

geom\_smooth(method = "lm") +

labs(y = "Mean squared error of coefficients,\ncompared to true data generating process",

x = "Number of coefficients to estimate") +

ggtitle("Fitting a Poisson GLM to a cross tab of counts",

"Comparison of methods of dealing with suppressed counts under 6")

#--------------further investigation----------

results\_df %>%

ggplot(aes(x = mse)) +

facet\_wrap(~method) +

geom\_density(fill = "steelblue", alpha = 0.5) +

scale\_x\_log10(label = comma, breaks = c(0.1, 0.3, 1, 3, 10, 100)) +

ggtitle("Bimodal distribution of performance when using the real data") +

theme(panel.grid.minor = element\_blank())

# which are the really poor performing methods when using the full data?

full\_data\_method <- results\_df %>%

filter(grepl("^Using full data", method )) %>%

arrange(desc(mse))

full\_data\_method %>%

ggplot(aes(x = prop\_suppressed, y = n, colour = mse > 3, size = mse)) +

geom\_point() +

labs(x = "Proportion of data that were suppressed as <6",

y = "Number of cells in table",

size = "Mean squared error of coefficients:",

colour = "Unusually large:") +

ggtitle("Performance of a model with the true underlying data",

"General success and occasional complete failure to pick the underlying data generating process")

full\_data\_method

# some really bad runs are 125, 721, 506, lets look at just one:

run125 <- coefs\_comp(125, output = "all")

run125$data %>%

select(animals, region, count, expected) %>%

knitr::kable()

run125$all\_coefficients %>%

mutate(method = str\_wrap(method, 30)) %>%

ggplot(aes(x = underlying, y = value, label = labels)) +

geom\_abline(slope = 1, intercept = 0) +

geom\_text() +

facet\_wrap(~method) +

ggtitle("Example of a particularly poorly performing model with the real data",

"Model run 125;\nMany zeroes in the data mean the GLM estimates are highly unstable") +

labs(x = "True value of underlying coefficients",

y = "Estimate from various types of model")

# lots of 25s:

glm(count ~ animals \* region, data = run125$data, family = "poisson")