Exploring Estimators

Derek Sonderegger

11/19/2018

## ── Attaching packages ───────────────────────────────────────── tidyverse 1.2.1 ──

## ✔ ggplot2 3.1.0 ✔ purrr 0.2.5  
## ✔ tibble 1.4.2 ✔ dplyr 0.7.8  
## ✔ tidyr 0.8.2 ✔ stringr 1.3.1  
## ✔ readr 1.1.1 ✔ forcats 0.3.0

## ── Conflicts ──────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

We have observed clusters of *C. diff.* patients. For each cluster, we let represent the number of patients in the cluster that have HAI. For this simulation, we assume that this number is the number of patients in the cluster minus 1. So we assume . Notice that can take on values of .

To create a population to sample from, we will model being random variables from a Poisson() distribution, and thus are random variables from a shifted Poisson distribution with a shift of .

Full.Data <- generate\_population(n=1000, lambda=2) #   
Full.Data %>%  
 count(Aquired) %>%  
 mutate(proportion = n/sum(n))

## # A tibble: 2 x 3  
## Aquired n proportion  
## <chr> <int> <dbl>  
## 1 Community 416 0.416  
## 2 Hospital 584 0.584

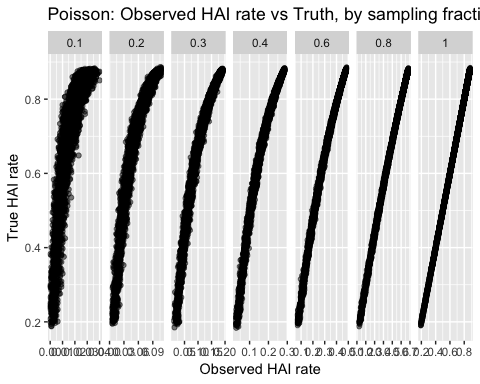
Full.Data %>%  
 count(clusterID) %>% rename(clusterSize = n) %>% # How many patients in each cluster  
 count(clusterSize) %>% rename(numClusters = n) %>% # How many clusters in each cluster size  
 mutate(numCommunity = numClusters ,  
 numHAI = numClusters \* (clusterSize-1),  
 numPatients = numCommunity + numHAI )

## # A tibble: 8 x 5  
## clusterSize numClusters numCommunity numHAI numPatients  
## <int> <int> <int> <dbl> <dbl>  
## 1 1 121 121 0 121  
## 2 2 124 124 124 248  
## 3 3 91 91 182 273  
## 4 4 52 52 156 208  
## 5 5 22 22 88 110  
## 6 6 3 3 15 18  
## 7 7 2 2 12 14  
## 8 8 1 1 7 8

# Create the Sampled Data sets  
frac\_sampled <- 1.0  
Sample.Data <- Full.Data %>%  
 sample\_frac(frac\_sampled, replace = FALSE)

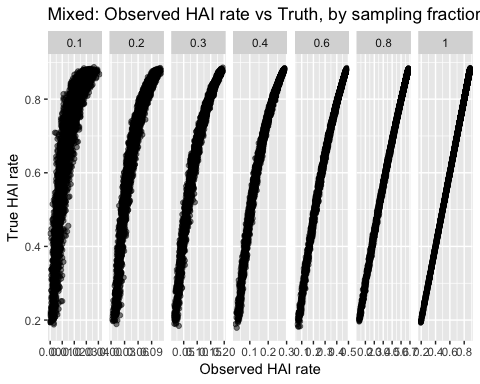
# If the population acutally is actually Poisson distributed  
# then we really need 30% sampling or more to actually get any accuracy  
n <- 1000  
N <- 50  
sample\_frac = .4  
lambda\_range=seq(.5,8, by=.1)  
  
out <- NULL  
for( sample\_frac in c(.1, .2, .3, .4, .6, .8, 1) ){  
 for( lambda in lambda\_range ){  
 for( i in 1:N ){  
 Full.Data <- generate\_population(n, lambda)  
 Sample.Data <- Full.Data %>% sample\_frac(sample\_frac, replace = FALSE)   
 True.HAI <- Full.Data %>% filter(Aquired == 'Hospital') %>% nrow(.)/nrow(Full.Data)  
 Obs.HAI <-   
 Sample.Data %>%   
 group\_by(clusterID) %>% count() %>%   
 mutate(n = n-1) %>% ungroup() %>%   
 summarise(HAI = sum(n)) %>% pull(HAI)  
 proportion <- Obs.HAI / nrow(Full.Data)  
 out <- rbind(out, data.frame(lambda=lambda, n=n, sample\_frac=sample\_frac, Obs.HAI.rate=proportion, True.HAI.rate = True.HAI))  
 }   
 }  
}  
Poisson <- out  
usethis::use\_data(Poisson)

ggplot(Poisson, aes(x=Obs.HAI.rate, y=True.HAI.rate)) +   
 geom\_point(alpha=.5) +  
 facet\_grid(. ~ sample\_frac, scales='free' ) +  
 labs(x='Observed HAI rate', y='True HAI rate') +   
 ggtitle('Poisson: Observed HAI rate vs Truth, by sampling fraction')



# If the population acutally is Poisson distributed mixed with logNormal  
n <- 1000  
N <- 50  
sample\_frac = .4  
lambda <- 2  
lambda\_range=seq(.5,8, by=.1)  
  
out <- NULL  
for( sample\_frac in c(.1, .2, .3, .4, .6, .8, 1) ){  
 for( lambda in lambda\_range ){  
 for( i in 1:N ){  
 Full.Data <- generate\_population(n, lambda, mixed=TRUE, mix.proportion = .01)  
 Sample.Data <- Full.Data %>% sample\_frac(sample\_frac, replace = FALSE)   
 True.HAI <- Full.Data %>% filter(Aquired == 'Hospital') %>% nrow(.)/nrow(Full.Data)  
 Obs.HAI <-   
 Sample.Data %>%   
 group\_by(clusterID) %>% count() %>%   
 mutate(n = n-1) %>% ungroup() %>%   
 summarise(HAI = sum(n)) %>% pull(HAI)  
 proportion <- Obs.HAI / nrow(Full.Data)  
 out <- rbind(out, data.frame(lambda=lambda, n=n, sample\_frac=sample\_frac, Obs.HAI.rate=proportion, True.HAI.rate = True.HAI))  
 }   
 }  
}  
Mixed <- out  
usethis::use\_data(Mixed)

ggplot(Mixed, aes(x=Obs.HAI.rate, y=True.HAI.rate)) +   
 geom\_point(alpha=.5) +  
 facet\_grid(. ~ sample\_frac, scales='free' ) +  
 labs(x='Observed HAI rate', y='True HAI rate') +   
 ggtitle('Mixed: Observed HAI rate vs Truth, by sampling fraction')



# # Can we use the Conway-Maxwell Poisson distribution?  
# df <- Sample.Data %>%  
# group\_by(clusterID) %>% count() %>% rename(clusterSize = n)   
# library(glmmTMB)  
# model <- glmmTMB(clusterSize ~ 1, data=df, family=truncated\_compois)  
# confint(model)[1,] %>% exp()  
# # This fails as the sampling fraction gets small < 0.4. So unless I  
# # can figure out how to specify the weighting fraction. Basically  
# # we must use that we know the sampling fraction.

# Full.Data <- generate\_population(n=1000, lambda=2) #   
# Full.Data %>%  
# count(Aquired) %>%  
# mutate(proportion = n/sum(n))  
#   
# Full.Data %>%  
# count(clusterID) %>% rename(clusterSize = n) %>% # How many patients in each cluster  
# count(clusterSize) %>% rename(numClusters = n) %>% # How many clusters in each cluster size  
# mutate(numCommunity = numClusters ,  
# numHAI = numClusters \* (clusterSize-1),  
# numPatients = numCommunity + numHAI )

# I created a Poisson distribution that truncates and, via weighting,  
# we also account for the sampling schemes.  
  
# Create the Sampled Data sets  
# Full.Data <- generate\_population(n=2000, lambda=2) #   
# frac\_sampled <- .2  
# Sample.Data <- Full.Data %>%  
# sample\_frac(frac\_sampled, replace = FALSE) %>%  
# arrange(clusterID)  
# df <- Sample.Data %>%  
# group\_by(clusterID) %>% count() %>% rename(clusterSize = n) %>%  
# mutate(HAI = clusterSize - 1)  
#   
# temp <- Vectorize(function(lambda){  
# out <- dtruncated\_weighted\_poisson(  
# x=df$clusterSize, lambda=lambda,  
# N = nrow(Full.Data), n=nrow(Sample.Data), log=TRUE )  
# # out <- dweighted\_poisson(  
# # x=df$HAI, lambda=lambda,  
# # N = nrow(Full.Data), n=nrow(Sample.Data), log=TRUE )  
#   
# out <- sum(out)   
# return(out)  
# })  
#   
# data.frame( lambda = seq(0.1, 4, by=.1) ) %>%  
# mutate( loglikelihood = temp(lambda) ) %>%  
# ggplot( aes(x=lambda, y=loglikelihood) ) +  
# geom\_line() + geom\_point()  
#   
# optimize(temp, c(0,4), maximum = TRUE )