Supplementary material 1: Code to implement bias adjustments

Rob Boyd

13 March 2023

## Introduction

In this document, I provide the code to tackle a simple biodiversity monitoring problem. The first step is to estimate mean occupancy of the plant C. vulgaris in Britain in two time-periods (1987-1999 and 2010-2019) using an unrepresentative nonprobability sample. We know the "truth" in this example, so we can assess the accuracy of various sample-based estimators. The second step is to estimate the difference between the two (i.e. the trend). Again, we know the truth, so we can assess the accuracy of the estimated trends from various estimators. Each of the estimators that we use can be viewed as an attempt to weight the sample in such a way that the distributions of "auxiliary variables" in the sample more closely resemble those in the population. We will also look at how well this has been achieved.

## Estimating per-period mean occupancy

The first step is to load the relevant packages and data. There are N rows in the data, where N is the number of land-containing 1 km grid squares in Great Britain (minus a few for which the auxiliary data are not available). There is one column per variable. heather\_true\_dist\_1987.1999 is the binary response variable (1 = occupied and 0 = unoccupied) for the first time-period; heather\_true\_dist\_2010.2019 is the same but for the second time period. sampled\_units\_1987.1999 and sampled\_units\_2010.2019 are also binary and indicate whether the grid square was sampled in each period. The next five columns are auxiliary variables, which are known for every grid square. The final two columns are estimated inclusion probabilities, which were derived using random forests and the auxiliary data.

library(raster)

## Warning: package 'raster' was built under R version 4.1.2

## Loading required package: sp

## Warning: package 'sp' was built under R version 4.1.2

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.1.2

library(survey)

## Warning: package 'survey' was built under R version 4.1.3

## Loading required package: grid

## Loading required package: Matrix

## Loading required package: survival

##   
## Attaching package: 'survey'

## The following object is masked from 'package:raster':  
##   
## cv

## The following object is masked from 'package:graphics':  
##   
## dotchart

library(rstanarm)

## Warning: package 'rstanarm' was built under R version 4.1.3

## Loading required package: Rcpp

## Warning: package 'Rcpp' was built under R version 4.1.2

## This is rstanarm version 2.21.3

## - See https://mc-stan.org/rstanarm/articles/priors for changes to default priors!

## - Default priors may change, so it's safest to specify priors, even if equivalent to the defaults.

## - For execution on a local, multicore CPU with excess RAM we recommend calling

## options(mc.cores = parallel::detectCores())

library(PracTools)

## Warning: package 'PracTools' was built under R version 4.1.3

##   
## Attaching package: 'PracTools'

## The following object is masked from 'package:survey':  
##   
## deff

library(reshape2)

## Warning: package 'reshape2' was built under R version 4.1.3

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:raster':  
##   
## intersect, select, union

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

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

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following objects are masked from 'package:rstanarm':  
##   
## compare\_models, R2

## The following object is masked from 'package:survival':  
##   
## cluster

## load population data  
pop <- read.csv("W:/PYWELL\_SHARED/Pywell Projects/BRC/Rob Boyd/NERC\_exploring\_frontiers/Data/all\_data.csv")  
  
pop <- pop[,-c(10,11,14)] # drop climate variable, which we didn't use in the end   
  
pop <- pop[complete.cases(pop), ]

Some approaches to adjusting samples, such as poststratification, require categorical axiliary variables. The auxiliary variables in pop are continuous, so they need to be discretized. We split three of the the variables into three catgories (i.e. cut points at the 33rd and 67th percentiles). This did not make sense for two variables, openAccessGB (proportion of grid square that is open access) and allPACoverage (proportion of grid square that is in a protected area), because most grid squares take the values zero or one. For these variables, we split the data into into two categories, 0 and >0, i.e. whether some of the grid square is protected or open acessess.

pop\_disc <- pop  
  
pop\_aux\_cont <- pop\_disc[,5:9]  
  
## discretize the auxiliary data in pop  
for (i in c(5,8,9)) {  
  
 q <- as.numeric(quantile(pop\_disc[,i], probs = c(0, 0.33, 0.66, 1)))  
   
 pop\_disc[,i] <- cut(pop\_disc[,i],   
 breaks = q,  
 labels = FALSE,  
 include.lowest = TRUE,  
 right = TRUE)  
   
 pop\_disc[,i] <- as.numeric(pop\_disc[,i])  
   
}  
  
## make PA and open access land coverage binary   
  
pop\_disc$openAccessGB <- ifelse(pop$openAccessGB > 0, 1, 0)  
  
pop\_disc$allPACoverage <- ifelse(pop$allPACoverage > 0, 1, 0)

There are a few data wrangling tasks to do next. We need to subset pop to create dataframes specific to each time-period and sampled grid squares only.

## pull out columns relevant to period 1  
pop\_p1 <- pop[,c(1,3,5,6,7,8,9,10)]  
  
pop\_disc\_p1 <- pop\_disc[,c(1,3,5,6,7,8,9,10)]  
  
## and period 2  
pop\_p2 <- pop[,c(2,4,5,6,7,8,9,11)]  
  
pop\_disc\_p2 <- pop\_disc[,c(2,4,5,6,7,8,9,11)]  
  
## pull out sampled rows for periods 1 and 2  
samp\_disc\_p1 <- pop\_disc\_p1[pop\_disc\_p1$sampled\_units\_1987.1999 == 1, ]  
  
samp\_disc\_p2 <- pop\_disc\_p2[pop\_disc\_p2$sampled\_units\_2010.2019 == 1, ]  
  
## pull out the auxiliary data for the whole population  
pop\_aux <- pop\_disc[,5:9]

The first step in our simple biodiversity monitoring problem is to estimate mean occupancy in each time-period. The true means are 0.317 in period one and 0.270 in period two.

pop\_mean\_p1 <- mean(pop\_p1$heather\_true\_dist\_1987.1999);pop\_mean\_p1

## [1] 0.3173697

pop\_mean\_p2 <- mean(pop\_p2$heather\_true\_dist\_2010.2019);pop\_mean\_p2

## [1] 0.2695963

In real life, we don't know the population means so have to estimate them. The R package survey provides functionality for estimating population parameters from samples. The first job is to create what is called a survey design object, which includes the data in the sample and information about the survey design. In our case, we don't know anything about the "survey" (or lack thereof), so the code is simple.

design\_p1 <- svydesign(ids=~0,  
 data = samp\_disc\_p1)

## Warning in svydesign.default(ids = ~0, data = samp\_disc\_p1): No weights or  
## probabilities supplied, assuming equal probability

design\_p2 <- svydesign(ids=~0,  
 data = samp\_disc\_p2)

## Warning in svydesign.default(ids = ~0, data = samp\_disc\_p2): No weights or  
## probabilities supplied, assuming equal probability

The warnings tell us that the package will assume equal sampling weights. This is not a problem, because we will adjust the weights later. Setting ids to ~0 just tells the function that we are not aware of any clustering in the data.

Using the survey designs, we can calculate the sample means for each period and the associated confidence intervals. The sample mean will act a baseline, and the aim is to improve on it by weighting.

samp\_mean\_p1 <- svymean(design = design\_p1,  
 x=~heather\_true\_dist\_1987.1999); samp\_mean\_p1

## mean SE  
## heather\_true\_dist\_1987.1999 0.25503 0.0014

samp\_mean\_p2 <- svymean(design = design\_p2,  
 x=~heather\_true\_dist\_2010.2019); samp\_mean\_p2

## mean SE  
## heather\_true\_dist\_2010.2019 0.24963 0.0012

## and their confidence intervals  
samp\_mean\_p1\_conf <- confint(object = samp\_mean\_p1,  
 level = 0.95); samp\_mean\_p1\_conf

## 2.5 % 97.5 %  
## heather\_true\_dist\_1987.1999 0.2522981 0.2577611

samp\_mean\_p2\_conf <- confint(object = samp\_mean\_p2,  
 level = 0.95); samp\_mean\_p2\_conf

## 2.5 % 97.5 %  
## heather\_true\_dist\_2010.2019 0.2473695 0.251881

The first set of weights we will try are the inverses of inclusion probabilities that were estimated using random forests. This approach is called quasi-randomisation.

weighted\_design\_p1 <- svydesign(ids=~0,  
 data = samp\_disc\_p1,  
 probs=~inclusionProbs\_1987.1999)  
  
weighted\_design\_p2 <- svydesign(ids=~0,  
 data = samp\_disc\_p2,  
 probs=~inclusionProbs\_2010.2019)  
  
## then get the weighted sample means  
weighted\_samp\_mean\_p1 <- svymean(design = weighted\_design\_p1,  
 x=~heather\_true\_dist\_1987.1999);weighted\_samp\_mean\_p1

## mean SE  
## heather\_true\_dist\_1987.1999 0.28211 0.0016

weighted\_samp\_mean\_p2 <- svymean(design = weighted\_design\_p2,  
 x=~heather\_true\_dist\_2010.2019);weighted\_samp\_mean\_p2

## mean SE  
## heather\_true\_dist\_2010.2019 0.30369 0.0014

## and their confidence intervals   
weighted\_samp\_mean\_p1\_conf <- confint(object = weighted\_samp\_mean\_p1,  
 level = 0.95);weighted\_samp\_mean\_p1\_conf

## 2.5 % 97.5 %  
## heather\_true\_dist\_1987.1999 0.2789893 0.2852357

weighted\_samp\_mean\_p2\_conf <- confint(object = weighted\_samp\_mean\_p2,  
 level = 0.95);weighted\_samp\_mean\_p2\_conf

## 2.5 % 97.5 %  
## heather\_true\_dist\_2010.2019 0.3009166 0.3064574

Next we can try poststratification.

cells <- data.frame(table(pop\_aux))  
  
str(cells)

## 'data.frame': 108 obs. of 6 variables:  
## $ postcode\_density\_299\_neighbours: Factor w/ 3 levels "1","2","3": 1 2 3 1 2 3 1 2 3 1 ...  
## $ openAccessGB : Factor w/ 2 levels "0","1": 1 1 1 2 2 2 1 1 1 2 ...  
## $ allPACoverage : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 2 2 2 ...  
## $ road\_length\_299\_neighbours : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...  
## $ UKelv : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Freq : int 111 652 417 5112 981 224 67 603 354 3266 ...

## now poststratify   
ps\_design\_p1 <- postStratify(design = design\_p1,  
 strata = samp\_disc\_p1[,3:7],  
 population = data.frame(table(pop\_aux)),  
 partial = T)  
  
ps\_design\_p2 <- postStratify(design = design\_p2,  
 strata = samp\_disc\_p2[,3:7],  
 population = data.frame(table(pop\_aux)),  
 partial = T)  
  
## and get the weighted mean across poststrata  
ps\_samp\_mean\_p1 <- svymean(design = ps\_design\_p1,  
 x=~heather\_true\_dist\_1987.1999,  
 na.rm = T);ps\_samp\_mean\_p1

## mean SE  
## heather\_true\_dist\_1987.1999 0.3358 0.0013

ps\_samp\_mean\_p2 <- svymean(design = ps\_design\_p2,  
 x=~heather\_true\_dist\_2010.2019);ps\_samp\_mean\_p2

## mean SE  
## heather\_true\_dist\_2010.2019 0.31441 0.001

## and their confidence intervals  
ps\_samp\_mean\_p1\_conf <- confint(object = ps\_samp\_mean\_p1,  
 level = 0.95);ps\_samp\_mean\_p1\_conf

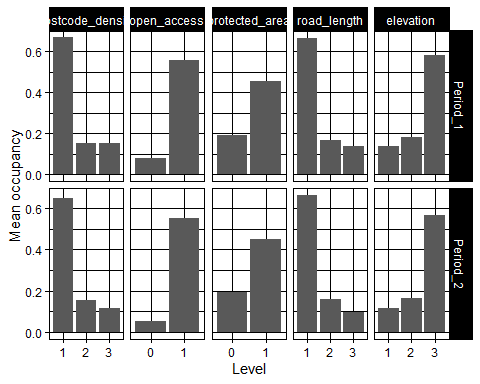
## 2.5 % 97.5 %  
## heather\_true\_dist\_1987.1999 0.3331772 0.3384183

ps\_samp\_mean\_p2\_conf <- confint(object = ps\_samp\_mean\_p2,  
 level = 0.95);ps\_samp\_mean\_p2\_conf

## 2.5 % 97.5 %  
## heather\_true\_dist\_2010.2019 0.3124262 0.316393

It is instructive to look at the variable of interest in each category of the auxiliary variables (recalling that we discretized them earlier). It is good to see that mean occupancy varies among categories, because this implies that there is something to be gained by poststratifying. If there was little difference, the adjustment from poststratifying would be minor.

s1 <- samp\_disc\_p1[,-8] # drop estimated inclusion probabilities, which aren't needed here  
  
colnames(s1) <- c("y", "R", "postcode\_density", "open\_access", "protected\_area",  
 "road\_length", "elevation")  
  
s1$period <- "Period\_1"  
  
s2 <- samp\_disc\_p2[,-8]  
  
colnames(s2) <- c("y", "R", "postcode\_density", "open\_access", "protected\_area",  
 "road\_length", "elevation")  
  
s2$period <- "Period\_2"  
  
s <- rbind(s1, s2)  
  
s <- melt(s, id = c("y", "period", "R"))  
  
ggplot(data = s, aes(x = factor(value), y = y)) +  
 geom\_bar(stat = "summary", fun = "mean") +  
 facet\_grid(period~variable, scales = "free") +  
 theme\_linedraw() +  
 labs(x = "Level", y = "Mean occupancy")



Another approach to weighting the sample mean is superpopulation modelling, which can be implemented using the function calibrate. It is better to use the continuous rather than discretized auxiliary variables for this approach, so we begin by creating two new survey designs based on those data (i.e. pop not pop\_disc). The calibrate function also requires population totals for each auxiliary variable. The intercept total is the population size and must be named "(Intercept)".

## get population totals for auxiliary variables  
aux\_tots <- c(nrow(pop\_aux\_cont), colSums(pop\_aux\_cont))  
  
names(aux\_tots)[1] <- "(Intercept)"  
  
## create new designs with the continuous rather than discretized auxiliary variables  
samp\_p1 <- pop\_p1[pop\_p1$sampled\_units\_1987.1999 == 1, ]  
  
samp\_p2 <- pop\_p2[pop\_p2$sampled\_units\_2010.2019 == 1, ]  
  
pre\_calib\_design\_p1 <- svydesign(ids=~0,  
 data = samp\_p1)

## Warning in svydesign.default(ids = ~0, data = samp\_p1): No weights or  
## probabilities supplied, assuming equal probability

pre\_calib\_design\_p2 <- svydesign(ids=~0,  
 data = samp\_p2)

## Warning in svydesign.default(ids = ~0, data = samp\_p2): No weights or  
## probabilities supplied, assuming equal probability

## now calibrate   
calib\_design\_p1 <- calibrate(design = pre\_calib\_design\_p1,  
 formula = ~ postcode\_density\_299\_neighbours +  
 openAccessGB + allPACoverage +   
 road\_length\_299\_neighbours + UKelv,  
 population = aux\_tots,  
 calfun="linear")  
  
sum(weights(calib\_design\_p1)) # check weights sum to the population size

## [1] 229584

summary(weights(calib\_design\_p1))

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -0.3964 1.7066 2.0154 2.3470 2.8445 7.6631

sp\_samp\_mean\_p1 <- svymean(~heather\_true\_dist\_1987.1999, design=calib\_design\_p1); sp\_samp\_mean\_p1

## mean SE  
## heather\_true\_dist\_1987.1999 0.34845 0.0015

calib\_design\_p2 <- calibrate(design = pre\_calib\_design\_p2,  
 formula = ~ postcode\_density\_299\_neighbours +  
 openAccessGB + allPACoverage +   
 road\_length\_299\_neighbours + UKelv,  
 population = aux\_tots,  
 calfun="linear")  
  
sum(weights(calib\_design\_p2))

## [1] 229584

summary(weights(calib\_design\_p2))

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.5078 1.2885 1.4500 1.6235 1.9524 3.6461

sp\_samp\_mean\_p2 <- svymean(~heather\_true\_dist\_2010.2019, design=calib\_design\_p2); sp\_samp\_mean\_p2

## mean SE  
## heather\_true\_dist\_2010.2019 0.32202 0.0011

## and the confidence intervals   
sp\_samp\_mean\_p1\_conf <- confint(object = sp\_samp\_mean\_p1,  
 level = 0.95); sp\_samp\_mean\_p1\_conf

## 2.5 % 97.5 %  
## heather\_true\_dist\_1987.1999 0.345594 0.3513011

sp\_samp\_mean\_p2\_conf <- confint(object = sp\_samp\_mean\_p2,  
 level = 0.95); sp\_samp\_mean\_p2\_conf

## 2.5 % 97.5 %  
## heather\_true\_dist\_2010.2019 0.3198705 0.324162

Perhaps more familiar to ecologists than the above approaches is subsampling. Our approach was to draw weighted random samples of size 500 with replacement from the original samples [note that these are different to sampling weights in 2)]. Using the postrata created above, I assign each grid square a weight equal to the proportion of the population in its stratum. The result is a subsample whose members were more likely to be from strata comprising a larger fraction of the population. The subsample mean is the estimator of the population mean. Note that I bootstrap the procedure. This is necessary because there is a random element to the downsampling, and the estimated means are sensitive to it.

## concatenate the levels of the variables in each column of sub\_samp\_disc to store unique strata as "id"  
samp\_p1$id <- paste0(samp\_disc\_p1$postcode\_density\_299\_neighbours,  
 samp\_disc\_p1$openAccessGB,  
 samp\_disc\_p1$allPACoverage,  
 samp\_disc\_p1$road\_length\_299\_neighbours,  
 samp\_disc\_p1$UKelv)  
  
pop$id <- paste0(pop\_disc$postcode\_density\_299\_neighbours,  
 pop\_disc$openAccessGB,  
 pop\_disc$allPACoverage,  
 pop\_disc$road\_length\_299\_neighbours,  
 pop\_disc$UKelv)  
  
pop\_props <- data.frame(table(pop$id))  
  
pop\_props$prop <- pop\_props$Freq/sum(pop\_props$Freq)  
  
colnames(pop\_props)[1] <- "id"  
  
samp\_props\_p1 <- merge(pop\_props, samp\_p1, by = "id")  
  
sub\_samp\_means\_p1 <- lapply(1:1000,  
 function(x) {  
 subSamp <- samp\_props\_p1[sample(1:nrow(samp\_props\_p1),   
 size = 200,   
 replace = T,   
 prob = samp\_props\_p1$prop),]  
   
 data.frame(mean = mean(subSamp$heather\_true\_dist\_1987.1999),  
 id = x)  
 })  
  
sub\_samp\_means\_p1 <- do.call("rbind", sub\_samp\_means\_p1)  
  
sub\_samp\_p1\_mean <- c(mean = mean(sub\_samp\_means\_p1$mean),  
 lower = quantile(sub\_samp\_means\_p1$mean, probs = 0.025),  
 upper = quantile(sub\_samp\_means\_p1$mean, probs = 0.975)); sub\_samp\_p1\_mean

## mean lower.2.5% upper.97.5%   
## 0.25473 0.19500 0.32000

## and period 2   
  
samp\_p2$id <- paste0(samp\_disc\_p2$postcode\_density\_299\_neighbours,  
 samp\_disc\_p2$openAccessGB,  
 samp\_disc\_p2$allPACoverage,  
 samp\_disc\_p2$road\_length\_299\_neighbours,  
 samp\_disc\_p2$UKelv)  
  
head(samp\_p2)

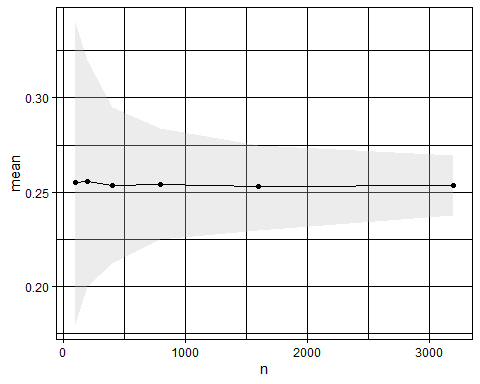
## heather\_true\_dist\_2010.2019 sampled\_units\_2010.2019  
## 1 1 1  
## 2 1 1  
## 3 1 1  
## 9 1 1  
## 12 1 1  
## 15 1 1  
## postcode\_density\_299\_neighbours openAccessGB allPACoverage  
## 1 34 0.96 1.00  
## 2 34 0.99 1.00  
## 3 34 1.00 0.00  
## 9 34 1.00 0.76  
## 12 34 1.00 0.00  
## 15 34 1.00 0.31  
## road\_length\_299\_neighbours UKelv inclusionProbs\_2010.2019 id  
## 1 46928.30 141 0.856 11112  
## 2 46928.30 115 0.770 11112  
## 3 47698.44 202 0.482 11013  
## 9 47296.35 65 0.766 11111  
## 12 46799.42 48 0.592 11011  
## 15 49603.95 25 0.916 11111

samp\_props\_p2 <- merge(pop\_props, samp\_p2, by = "id")  
  
sub\_samp\_means\_p2 <- lapply(1:1000,  
 function(x) {  
 subSamp <- samp\_props\_p2[sample(1:nrow(samp\_props\_p2),   
 size = 200,   
 replace = T,   
 prob = samp\_props\_p2$prop),]  
   
 data.frame(mean = mean(subSamp$heather\_true\_dist\_2010.2019),  
 id = x)  
 })  
  
sub\_samp\_means\_p2 <- do.call("rbind", sub\_samp\_means\_p2)  
  
sub\_samp\_p2\_mean <- c(mean = mean(sub\_samp\_means\_p2$mean),  
 lower = quantile(sub\_samp\_means\_p2$mean, probs = 0.025),  
 upper = quantile(sub\_samp\_means\_p2$mean, probs = 0.975)); sub\_samp\_p2\_mean

## mean lower.2.5% upper.97.5%   
## 0.271865 0.215000 0.330000

The decision to create subsamples of size 500 was somewhat arbitrary. Changing the size of the subsamples makes little difference to their point estimates, but, naturally, the width of the confidence intervals decrease with increasing sample size. The code below demonstrates this for period one, but the result is the same for period two.

getSens <- function(n) {  
   
 sub\_samp\_means\_p1 <- lapply(1:1000,  
 function(x) {  
 subSamp <- samp\_props\_p1[sample(1:nrow(samp\_props\_p1),   
 size = n,   
 replace = T,   
 prob = samp\_props\_p1$prop),]  
   
 data.frame(mean = mean(subSamp$heather\_true\_dist\_1987.1999),  
 id = x)  
 })  
   
 sub\_samp\_means\_p1 <- do.call("rbind", sub\_samp\_means\_p1)  
   
 sub\_samp\_p1\_mean <- data.frame(mean = mean(sub\_samp\_means\_p1$mean),  
 lower = quantile(sub\_samp\_means\_p1$mean, probs = 0.025),  
 upper = quantile(sub\_samp\_means\_p1$mean, probs = 0.975),  
 n = n)  
   
}  
  
sens <- lapply(c(100,200,400,800,1600,3200),  
 getSens)  
  
sens <- do.call("rbind", sens)  
  
ggplot(data = sens, aes(x = n, y = mean)) +  
 geom\_ribbon(aes(ymin =lower, ymax = upper), fill = "grey", alpha = 0.3) +  
 geom\_line() +  
 theme\_linedraw() +   
 geom\_point()



The next approach to estimating the populations means is Multilevel Regression and Poststratification (MRP). Strictly speaking, MRP does not produce unit level weights (i.e. a weight for every grid square), but the concept is very similar. Another feature of MRP is that it is computationally demanding.

## fit model  
fit <- stan\_glmer(  
 heather\_true\_dist\_1987.1999 ~ 1 + (1 | postcode\_density\_299\_neighbours) +   
 (1 | road\_length\_299\_neighbours) + (1 | allPACoverage) +   
 (1 | openAccessGB) + (1 | UKelv),  
 family = binomial(link = "logit"),  
 data = samp\_disc\_p1,  
 chains = 3,  
 iter = 5000  
)  
  
print(fit)  
  
posterior\_prob <- posterior\_linpred(fit, transform = T, newdata = cells)  
  
poststrat\_prob <- posterior\_prob %\*% cells$Freq / sum(cells$Freq)  
  
props\_p1 <- samp\_means\_per\_cell\_p1$n / samp\_means\_per\_cell\_p1$N  
  
props\_p1[is.na(props\_p1)] <- 0  
  
poststrat\_prob <- posterior\_prob %\*% props\_p1  
   
#write.csv(poststrat\_prob,  
# "W:/PYWELL\_SHARED/Pywell Projects/BRC/Rob Boyd/NERC\_exploring\_frontiers/Data/poststrat\_prob.csv",  
# row.names = F)  
  
model\_popn\_pref <- c(mean = mean(poststrat\_prob),  
 lower = quantile(poststrat\_prob, probs = 0.025),  
 upper = quantile(poststrat\_prob, probs = 0.975))  
  
round(model\_popn\_pref, 3)  
  
fit2 <- stan\_glmer(  
 heather\_true\_dist\_2010.2019 ~ 1 + (1 | postcode\_density\_299\_neighbours) +   
 (1 | road\_length\_299\_neighbours) + (1 | allPACoverage) +   
 (1 | openAccessGB) + (1 | UKelv),  
 family = binomial(link = "logit"),  
 data = samp\_disc\_p2,  
 chains = 2  
)  
  
print(fit2)  
  
posterior\_prob\_p2 <- posterior\_linpred(fit2, transform = TRUE, newdata = cells)  
  
poststrat\_prob\_p2 <- posterior\_prob\_p2 %\*% cells$Freq / sum(cells$Freq)  
  
#write.csv(poststrat\_prob\_p2,  
# "W:/PYWELL\_SHARED/Pywell Projects/BRC/Rob Boyd/NERC\_exploring\_frontiers/Data/poststrat\_prob\_p2.csv",  
# row.names = F)  
  
model\_popn\_pref\_p2 <- c(mean = mean(poststrat\_prob\_p2),  
 lower = quantile(poststrat\_prob\_p2, probs = 0.025),  
 upper = quantile(poststrat\_prob\_p2, probs = 0.975))  
  
#round(model\_popn\_pref\_p2, 3)

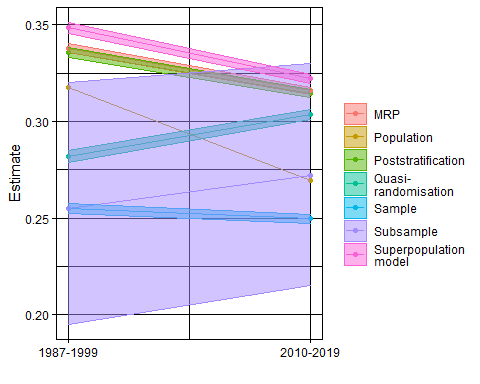
Instead we load the posterior distributions from models fitted on a computer cluster.

poststrat\_prob <- read.csv("W:/PYWELL\_SHARED/Pywell Projects/BRC/Rob Boyd/NERC\_exploring\_frontiers/Data/mrp\_p1.csv")[,1]  
  
model\_popn\_pref <- c(mean = mean(poststrat\_prob),  
 lower = quantile(poststrat\_prob, probs = 0.025),  
 upper = quantile(poststrat\_prob, probs = 0.975))  
  
poststrat\_prob\_p2 <- read.csv("W:/PYWELL\_SHARED/Pywell Projects/BRC/Rob Boyd/NERC\_exploring\_frontiers/Data/mrp\_p2.csv")[,1]  
  
model\_popn\_pref\_p2 <- c(mean = mean(poststrat\_prob\_p2),  
 lower = quantile(poststrat\_prob\_p2, probs = 0.025),  
 upper = quantile(poststrat\_prob\_p2, probs = 0.975))

We can combine the estimates of the population means in each time-period and plot them to get an idea of which methods work best.

plotDat <- data.frame(p = c(1,2,1,2,1,2,1,2,1,2,1,2,1,2),  
 est = c(pop\_mean\_p1[1], pop\_mean\_p2[1], samp\_mean\_p1[1], samp\_mean\_p2[1], ps\_samp\_mean\_p1[1], ps\_samp\_mean\_p2[1], weighted\_samp\_mean\_p1[1], weighted\_samp\_mean\_p2[1], sp\_samp\_mean\_p1[1], sp\_samp\_mean\_p2[1], model\_popn\_pref[1],model\_popn\_pref\_p2[1], sub\_samp\_p1\_mean[1], sub\_samp\_p2\_mean[1]),  
 type = c("Population", "Population", "Sample", "Sample", "Poststratification", "Poststratification", "Quasi-  
randomisation", "Quasi-  
randomisation", "Superpopulation  
model", "Superpopulation  
model", "MRP", "MRP", "Subsample", "Subsample"),  
 lower = c(pop\_mean\_p1[1], pop\_mean\_p2[1], samp\_mean\_p1\_conf[1], samp\_mean\_p2\_conf[1], ps\_samp\_mean\_p1\_conf[1], ps\_samp\_mean\_p2\_conf[1], weighted\_samp\_mean\_p1\_conf[1], weighted\_samp\_mean\_p2\_conf[1], sp\_samp\_mean\_p1\_conf[1], sp\_samp\_mean\_p2\_conf[1], model\_popn\_pref[2], model\_popn\_pref\_p2[2], sub\_samp\_p1\_mean[2], sub\_samp\_p2\_mean[2]),  
 upper = c(pop\_mean\_p1[2], pop\_mean\_p2[2], samp\_mean\_p1\_conf[2], samp\_mean\_p2\_conf[2], ps\_samp\_mean\_p1\_conf[2], ps\_samp\_mean\_p2\_conf[2], weighted\_samp\_mean\_p1\_conf[2], weighted\_samp\_mean\_p2\_conf[2], sp\_samp\_mean\_p1\_conf[2], sp\_samp\_mean\_p2\_conf[2], model\_popn\_pref[3], model\_popn\_pref\_p2[3], sub\_samp\_p1\_mean[3], sub\_samp\_p2\_mean[3]))  
  
print(  
ggplot(data = plotDat, aes(x = p, y = est, colour = type, fill = type)) +  
 geom\_point() +  
 geom\_line() +  
 theme\_linedraw() +  
 geom\_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.5) +  
 labs(x = "",  
 y = "Estimate",  
 fill = "",  
 colour = "") +  
 scale\_x\_continuous(breaks = c(1,2), labels = c("1987-1999", "2010-2019"))  
)

## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning  
## -Inf



## Estimating the trend in mean occupancy

The second part of our simple biodiversity monitoring problem is to estimate the difference in mean occupancy between the two time-periods (i.e. the trend). It is simple to obtain point estimates of the trends from each method.

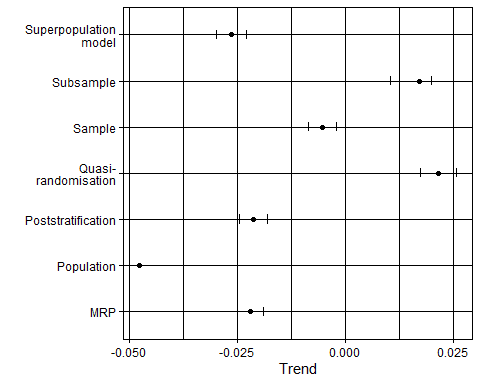
trends <- lapply(unique(plotDat$type),  
 function(x) {  
 data.frame(difference = plotDat$est[plotDat$p == 2 & plotDat$type == x] - plotDat$est[plotDat$p == 1 & plotDat$type == x],  
 estimator = x)  
 })  
  
trends <- do.call("rbind", trends)

The standard errors and confidence intervals are more complicated. The standard error of a difference in means is the square root of the sum of sampling variances of the two means. We obtained the sampling variances by squaring the standard errors provided by the survey package. MRP is different because the 95% credible interval of its trend can be extracted directly from the posterior distribution of the difference.

## sample mean   
sample\_mean\_se <- sqrt(SE(ps\_samp\_mean\_p1)^2 + SE(ps\_samp\_mean\_p2)^2)  
  
sample\_mean\_upper <- (samp\_mean\_p2 - samp\_mean\_p1) + 1.96 \* sample\_mean\_se  
  
sample\_mean\_lower <- (samp\_mean\_p2 - samp\_mean\_p1) - 1.96 \* sample\_mean\_se  
  
## quasi-randomisation   
weighted\_mean\_se <- sqrt(SE(weighted\_samp\_mean\_p1)^2 + SE(weighted\_samp\_mean\_p2)^2)  
  
weighted\_mean\_upper <- (weighted\_samp\_mean\_p2 - weighted\_samp\_mean\_p1) + 1.96 \* weighted\_mean\_se  
  
weighted\_mean\_lower <- (weighted\_samp\_mean\_p2 - weighted\_samp\_mean\_p1) - 1.96 \* weighted\_mean\_se  
  
## poststratification  
ps\_mean\_se <- sqrt(SE(ps\_samp\_mean\_p1)^2 + SE(ps\_samp\_mean\_p2)^2)  
  
ps\_mean\_upper <- (ps\_samp\_mean\_p2 - ps\_samp\_mean\_p1) + 1.96 \* ps\_mean\_se  
  
ps\_mean\_lower <- (ps\_samp\_mean\_p2 - ps\_samp\_mean\_p1) - 1.96 \* ps\_mean\_se  
  
## superpopulation  
sp\_mean\_se <- sqrt(SE(sp\_samp\_mean\_p1)^2 + SE(sp\_samp\_mean\_p2)^2)  
  
sp\_mean\_upper <- (sp\_samp\_mean\_p2 - sp\_samp\_mean\_p1) + 1.96 \* sp\_mean\_se  
  
sp\_mean\_lower <- (sp\_samp\_mean\_p2 - sp\_samp\_mean\_p1) - 1.96 \* sp\_mean\_se  
  
## subsampling  
  
diffsSub <- sub\_samp\_p2\_mean - sub\_samp\_p1\_mean  
  
#diffs <- diffs[,1]  
  
sub\_up <- quantile(diffsSub, probs = 0.975)  
  
sub\_low <- quantile(diffsSub, probs = 0.025)  
  
## MRP   
  
diffs <- poststrat\_prob\_p2 - poststrat\_prob  
  
#diffs <- diffs[,1]  
  
MRP\_up <- quantile(diffs, probs = 0.975)  
  
MRP\_low <- quantile(diffs, probs = 0.025)  
  
head(trends)

## difference estimator  
## 1 -0.047773364 Population  
## 2 -0.005404438 Sample  
## 3 -0.021388099 Poststratification  
## 4 0.021574504 Quasi-\nrandomisation  
## 5 -0.026431269 Superpopulation\nmodel  
## 6 -0.022050646 MRP

trends$lower <- c(NA, sample\_mean\_lower, ps\_mean\_lower, weighted\_mean\_lower, sp\_mean\_lower, MRP\_low, sub\_low)  
  
trends$upper <- c(NA, sample\_mean\_upper, ps\_mean\_upper, weighted\_mean\_upper, sp\_mean\_upper, MRP\_up, sub\_up)  
  
ggplot(data = trends, aes(x = difference, y = estimator)) +  
 geom\_point() +   
 theme\_linedraw() +  
 geom\_vline(xintercept = 0) +  
 labs(x = "Trend",  
 y = "") +  
 geom\_errorbar(aes(xmin = lower, xmax = upper, width = .2))



## Visualizing the effects of weighting on the distributions of auxiliary variables

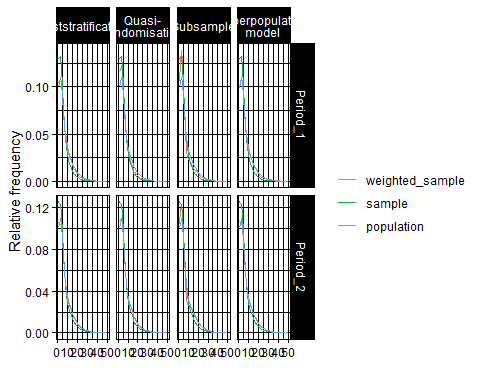
So far we have seen that weighting generally improves the accuracy of the estimates of mean occupancy in each period and the difference between the two. To see how it is doing this, it is instructive to look at the distributions of the auxiliaries in the sample, the weighted sample and the population. If the distributions in the weighted sample are closer to those in the original sample to those in the population, then weighting has been successful.

It is simple to obtain the weighted distributions for the superpopulation model, poststratification and quasi-randomisation, because we have the weights. When I implemented subsampling, however, I did not explicitly calculate weights. Instead, I wrote a simple function to extract the relative frequency distributions of the auxiliaries in the subsample.

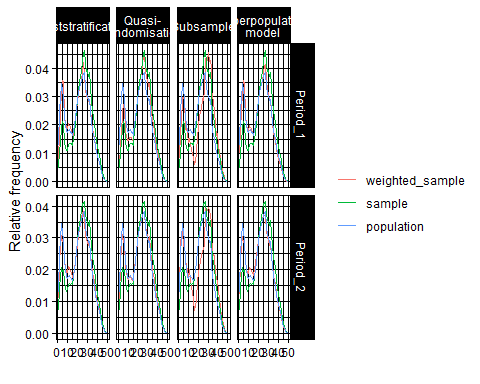
getRelFreqs <- function(dat, breaks, var, period, iter, bins) {  
   
 sub <- lapply(1:iter,  
 function(x) {  
   
 subSamp <- dat[sample(1:nrow(dat),   
 size = 500,   
 replace = F,   
 prob = dat$prop),]  
  
 z <- data.frame(val = subSamp[,var],  
 j = x)  
  
 ints <- findInterval(z$val, sort(unique(as.numeric(c(lower = as.numeric( sub("\\((.+),.\*", "\\1", bins) ),  
 upper = as.numeric( sub("[^,]\*,([^]]\*)\\]", "\\1", bins) ))))))  
  
 bin\_counts <- table(ints)  
  
 bin\_counts\_vec <- rep(0, 50)  
   
 for (bin in 1:50) {  
 count <- bin\_counts[as.character(bin)]  
 if (!is.na(count)) {  
 bin\_counts\_vec[bin] <- count  
 }  
 }  
  
 bin\_rel\_freqs <- bin\_counts\_vec / sum(bin\_counts\_vec)  
  
 })  
   
 sub <- do.call("cbind", sub)  
   
 sub <- rowMeans(sub)  
   
 data.frame(bin = 1:50,  
 var = 1,  
 id = period,  
 weightType = "Subsample",   
 variable = "weighted\_sample",   
 value = sub)  
   
}  
  
## period one  
  
# elevation  
relF\_p1\_elev <- getRelFreqs(dat = samp\_props\_p1,  
 breaks = 50,   
 var = "UKelv",  
 period = "Period\_1",  
 iter = 100,  
 bins = cut(pop$UKelv, breaks = 50))  
  
# road length   
relF\_p1\_road <- getRelFreqs(dat = samp\_props\_p1,  
 breaks = 50,   
 var = "road\_length\_299\_neighbours",  
 period = "Period\_1",  
 iter = 100,  
 bins = cut(pop$road\_length\_299\_neighbours, breaks = 50))  
  
## and period two   
  
# elevation  
relF\_p2\_elev <- getRelFreqs(dat = samp\_props\_p2,  
 breaks = 50,   
 var = "UKelv",  
 period = "Period\_2",  
 iter = 100,  
 bins = cut(pop$UKelv, breaks = 50))  
  
#road length  
relF\_p2\_road <- getRelFreqs(dat = samp\_props\_p2,  
 breaks = 50,   
 var = "road\_length\_299\_neighbours",  
 period = "Period\_2",  
 iter = 100,  
 bins = cut(pop$road\_length\_299\_neighbours, breaks = 50))

I have also written a function, relFreqPlot, that uses the weights calculated earlier to produce relative frequency plots the auxiliaries in the weighted samples and compares these to the distributions in the unadjusted samples and population. The function requires weights, which we do not have for the subsampling estimator. However, it does accept relative frequencies for a specified variable, which we created for the subsampling estimator using getRelFreqs earlier.

relFreqPlot <- function(pop,  
 R,  
 x,  
 weights,  
 breaks,  
 RNames,  
 WNames,  
 addVarByRelFreq = FALSE,  
 varByRelFreq) {  
  
 dat <- lapply(1:length(R),  
 function(y) {  
  
 stats <- lapply(1:length(weights),  
 function(z) {  
   
 pop$bin <- cut(pop[,x], breaks = breaks, labels = FALSE)  
   
 samp <- pop[pop[R[y]]==1,]  
  
 samp$weights = weights[[z]][[y]]  
  
 weightedFreq <- lapply(unique(pop$bin),  
 function(x) {  
 data.frame(weighted\_sample = sum(samp$weights[samp$bin==x]) / sum(samp$weights),  
 sample = nrow(samp[samp$bin== x,]) / nrow(samp),  
 population = nrow(pop[pop$bin == x,]) / nrow(pop),  
 bin = x,  
 var = z,  
 id = RNames[y],  
 weightType = WNames[z])  
 })  
   
 weightedFreq <- do.call("rbind", weightedFreq)  
   
 melt(weightedFreq, id = c("bin", "var", "id", "weightType"))  
   
 })  
   
 if (length(weights) > 1 | length(R) > 1) stats <- do.call("rbind", stats)  
   
 })  
  
   
 if (length(weights) > 1 | length(R) > 1) dat <- do.call("rbind", dat)  
   
 if (addVarByRelFreq == TRUE) {  
   
 for (i in 1:length(R)) {  
   
 dfWSamp <- varByRelFreq[[i]]  
   
 dfSamp <- dat[dat$weightType == WNames[2] & dat$variable == "sample" & dat$id == dfWSamp$id,]  
   
 dfSamp$weightType <- dfWSamp$weightType  
   
 dfPop <- dat[dat$weightType == WNames[2] & dat$variable == "population" & dat$id == dfWSamp$id,]  
   
 dfPop$weightType <- dfWSamp$weightType  
   
 dat <- rbind(dat, dfWSamp, dfSamp, dfPop)  
  
 }  
   
   
 }   
  
 p <- ggplot(data=dat,aes(y = value, x = bin, colour = variable)) +  
 geom\_line() +  
 theme\_linedraw() +  
 labs(colour = "",  
 x = "",  
 y = "Relative frequency")  
  
 if (length(weights) > 1 | length(R) > 1) p <- p + facet\_grid(id~weightType,   
 scales = "free\_y")  
   
 return(list(plot = p, data = dat))  
   
}  
  
p\_elev <- relFreqPlot(pop = pop,  
 x = c("UKelv"),  
 R = c("sampled\_units\_1987.1999", "sampled\_units\_2010.2019"),  
 RNames = c("Period\_1", "Period\_2"),  
 weights = list(list(p1 = 1/calib\_design\_p1$prob,  
 p2 = 1/calib\_design\_p2$prob),  
 list(p1 = 1/ps\_design\_p1$prob,  
 p2 = 1/ps\_design\_p2$prob),  
 list(p1 = 1/weighted\_design\_p1$prob,  
 p2 = 1/weighted\_design\_p2$prob)),  
 WNames = c("Superpopulation  
model", "Poststratification", "Quasi-  
randomisation"),  
 breaks = 50,  
 addVarByRelFreq = TRUE,  
 varByRelFreq = list(relF\_p1\_elev,relF\_p2\_elev))  
  
p\_elev$plot



p\_road <- relFreqPlot(pop = pop,  
 x = c("road\_length\_299\_neighbours"),  
 R = c("sampled\_units\_1987.1999", "sampled\_units\_2010.2019"),  
 RNames = c("Period\_1", "Period\_2"),  
 weights = list(list(p1 = 1/calib\_design\_p1$prob,  
 p2 = 1/calib\_design\_p2$prob),  
 list(p1 = 1/ps\_design\_p1$prob,  
 p2 = 1/ps\_design\_p2$prob),  
 list(p1 = 1/weighted\_design\_p1$prob,  
 p2 = 1/weighted\_design\_p2$prob)),  
 WNames = c("Superpopulation  
model", "Poststratification", "Quasi-  
randomisation"),  
 breaks = 50,  
 addVarByRelFreq = TRUE,  
 varByRelFreq = list(relF\_p1\_road,relF\_p2\_road))  
  
p\_road$plot



A visual comparison is fine, but it is better to do it formally. I created another function, auxImprovement, that assesses the deviations of the sample and weighted samples' relative frequency distributions from those in the population. The test statistic is the mean absolute error across all bins in the frequency distributions.

auxImprovement <- function(dat, period, estimator) {  
   
 samp <- dat$data$value[dat$data$variable=="sample" & dat$data$id == period & dat$data$weightType == estimator]  
   
 est <- dat$data$value[dat$data$variable=="weighted\_sample" & dat$data$id == period & dat$data$weightType == estimator]  
   
 pop <- dat$data$value[dat$data$variable=="population" & dat$data$id == period & dat$data$weightType == estimator]  
   
 data.frame(mae\_samp = mean(abs(pop-samp)),  
 mae\_est = mean(abs(pop-est)))  
  
}  
  
## road length in period one  
  
auxImprovement(dat = p\_road,  
 period = "Period\_1",  
 estimator = "Subsample")

## mae\_samp mae\_est  
## 1 0.005297292 0.007669324

## road length in period two  
  
auxImprovement(dat = p\_road,  
 period = "Period\_1",  
 estimator = "Subsample")

## mae\_samp mae\_est  
## 1 0.005297292 0.007669324

## elevation in period one  
  
auxImprovement(dat = p\_elev,  
 period = "Period\_1",  
 estimator = "Subsample")

## mae\_samp mae\_est  
## 1 0.004377116 0.008811633

## elevation in period two  
  
auxImprovement(dat = p\_elev,  
 period = "Period\_2",  
 estimator = "Subsample")

## mae\_samp mae\_est  
## 1 0.003082943 0.007984822

Note that we have not looked at the distributions of three of the five auxiliary variables or of the weighted samples produced by MRP. The shapes of the distributions of the other auxiliaries make it difficult to assess the effects of weighting. For our implementation of MRP, it is not clear how to obtain grid-square-level weights or relative frequencies.