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 three columns are the auxiliary variables, protected area coverage (proportion per grid square) in periods one and two and avaerage elevation, 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 4.1.3

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 4.1.3

## 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

library(ppcor)

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

## Loading required package: MASS

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':  
##   
## select

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

## load population data  
pop <- read.csv("W:/PYWELL\_SHARED/Pywell Projects/BRC/Rob Boyd/NERC\_exploring\_frontiers/Data/all\_data\_varying\_PA.csv")  
  
names(pop)

## [1] "heather\_true\_dist\_1987.1999" "heather\_true\_dist\_2010.2019"   
## [3] "sampled\_units\_1987.1999" "sampled\_units\_2010.2019"   
## [5] "postcode\_density\_299\_neighbours" "openAccessGB"   
## [7] "PA\_1987" "PA\_2010"   
## [9] "road\_length\_299\_neighbours" "UKelv"   
## [11] "X\_1" "X\_3"   
## [13] "inclusionProbs\_1987.1999" "inclusionProbs\_2010.2019"   
## [15] "layer"

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 average elevation into ten categories (one per decile). This did not make sense for the proportion of grid squares that are protected, because most grid squares take the values zero or one. For this variable, we split the data into into two categories, 0 and >0, i.e. whether some of the grid square is protected.

pop\_disc <- pop  
  
pop\_aux\_cont\_p1 <- pop[,c(7,10)]  
  
pop\_aux\_cont\_p2 <- pop[,c(8,10)] # columns with relevant auxiliaries  
  
## discretize elevation  
  
q <- as.numeric(quantile(pop\_disc[,10], probs = seq(0,1,length.out = 10)))  
   
pop\_disc[,10] <- cut(pop\_disc[,10],   
 breaks = q,  
 labels = FALSE,  
 include.lowest = TRUE,  
 right = TRUE)  
   
pop\_disc[,10] <- as.numeric(pop\_disc[,10])  
  
## discretize PA coverage   
  
pop\_disc$PA\_1987 <- ifelse(pop$PA\_1987 > 0, 1, 0)  
  
pop\_disc$PA\_2010 <- ifelse(pop$PA\_2010 > 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  
## pull out columns relevant to period 1  
pop\_p1 <- pop[,c(1,3,7,10,13)]  
  
names(pop\_p1)

## [1] "heather\_true\_dist\_1987.1999" "sampled\_units\_1987.1999"   
## [3] "PA\_1987" "UKelv"   
## [5] "inclusionProbs\_1987.1999"

pop\_disc\_p1 <- pop\_disc[,c(1,3,7,10,13)]  
  
## and period 2  
pop\_p2 <- pop[,c(2,4,8,10,14)]  
  
pop\_disc\_p2 <- pop\_disc[,c(2,4,8,10,14)]  
  
## 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\_p1 <- pop\_disc[,c(7,10)]  
  
pop\_aux\_p2 <- pop\_disc[,c(8,10)]

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 adjustment we will try is poststratification.

## next we want to postratify. We use the auxiliary data from earlier  
## first, cross the covariates to get the poststrata  
cells\_p1 <- data.frame(table(pop\_aux\_p1))  
cells\_p2 <- data.frame(table(pop\_aux\_p2))  
  
## now poststratify   
ps\_design\_p1 <- survey::postStratify(design = design\_p1,  
 strata = ~UKelv + PA\_1987,  
 population = cells\_p1,  
 partial = T)  
  
ps\_design\_p2 <- survey::postStratify(design = design\_p2,  
 strata = ~UKelv + PA\_2010,  
 population = cells\_p2,  
 partial = T)  
  
## and get the weighted mean across poststrata  
ps\_samp\_mean\_p1 <- svymean(design = ps\_design\_p1,  
 x=~heather\_true\_dist\_1987.1999);ps\_samp\_mean\_p1

## mean SE  
## heather\_true\_dist\_1987.1999 0.31549 0.0014

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 0.28923 0.0011

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

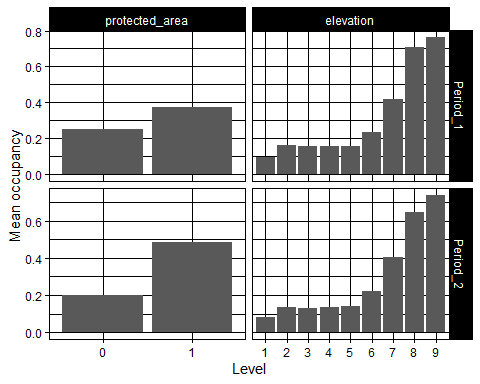
## 2.5 % 97.5 %  
## heather\_true\_dist\_1987.1999 0.312779 0.318202

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 0.2871042 0.2913508

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[,-5] # drop estimated inclusion probabilities, which aren't needed here  
  
colnames(s1) <- c("y", "R", "protected\_area", "elevation")  
  
s1$period <- "Period\_1"  
  
s2 <- samp\_disc\_p2[,-5]  
  
colnames(s2) <- c("y", "R", "protected\_area","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")

 The next set of adjustments that we will try includes superpopulation modelling, quasi-randomisation and a "doubly robust" procedure. Our superpopulation model is a linear regression; it predicts mean occupancy at non-sampled sites, and the population mean can be estimated using these predictions and the observations at sampled sites. Quasi-randomisation involves fitting a sample inclusion model, where the response variable takes the value 1 at sampled sites a 0 elsewhere. This is a matter of prediction rather than inference, because there are no missing data in the response, so we used a random forest and many predictors (in addition to the auxiliary variables). The doubly robust estimator combines the sample inclusion and superpopulation models in such a way that if either is correct, then the estimate of the population mean is unbiased.

Variance estimation is challenging for the doubly robust estimator. Our approach was to use bootstrapping: we created 1000 bootstrap samples by resampling the data with replacement and estimated the population mean in each time-period using each sample. The random forest used to estimate sample inclusion probabilities, which had to be fitted for each of the 1000 bootstrap samples, is quite expensive to run (in terms of time), so we conducted this portion of the analysis on a computer cluster. I present the code that was passed to the cluster below for transparency, but it is not actually implemented.

library(randomForest)  
  
covs\_df <- read.csv("/home/users/rboyd/inputs/doubly\_robust\_dat.csv")  
  
boot\_samp <- covs\_df[sample(1:nrow(covs\_df),  
 size = nrow(covs\_df),  
 replace = T), ]  
  
## fit period 1 superpopulation model   
sp\_mod\_1 <- lm(heather\_true\_dist\_1987.1999~PA\_1987+UKelv, data = boot\_samp[boot\_samp$sampled\_units\_1987.1999==1,])  
  
## predict for all i   
p\_1 <- as.numeric(predict(sp\_mod\_1, newdata = covs\_df))  
  
## get sampled indices  
sampled\_inds\_1 <- which(covs\_df$sampled\_units\_1987.1999==1)  
  
unSampled\_inds\_1 <- which(covs\_df$sampled\_units\_1987.1999==0)  
  
## get superpopulation model prediction of mean   
  
sp\_1 <- c(covs\_df$heather\_true\_dist\_1987.1999[sampled\_inds\_1],  
 p\_1[unSampled\_inds\_1])  
  
## residuals  
r\_1 <- covs\_df$heather\_true\_dist\_1987.1999[sampled\_inds\_1] - p\_1[sampled\_inds\_1]  
  
## sample inclusion model  
qr\_mod\_1 <- randomForest(x = boot\_samp[,c(1:3,5:10)],   
 y = as.factor(boot\_samp$sampled\_units\_1987.1999),  
 norm.votes = T)  
  
## predict for all i  
preds\_1 <- predict(qr\_mod\_1,  
 newdata = covs\_df,  
 type = "prob",  
 index = 2)  
  
## give 0 weights a small value  
preds\_1[preds\_1 == 0] <- 0.01  
  
## quasi-randomisation prediction   
  
qr\_1 <- sum((1/preds\_1[sampled\_inds\_1,2]) \* covs\_df$heather\_true\_dist\_1987.1999[sampled\_inds\_1]) / sum(1/(preds\_1[sampled\_inds\_1,2]))  
  
## doubly robust estimator of population mean  
DR\_1 <- 1/nrow(covs\_df) \* sum(r\_1/preds\_1[sampled\_inds\_1,2]) + 1/nrow(covs\_df) \* sum(p\_1)  
  
### period 2  
sp\_mod\_2 <- lm(heather\_true\_dist\_2010.2019~PA\_2010+UKelv, data = boot\_samp[boot\_samp$sampled\_units\_2010.2019==1,])  
  
p\_2 <- as.numeric(predict(sp\_mod\_2, newdata = covs\_df))  
  
sampled\_inds\_2 <- which(covs\_df$sampled\_units\_2010.2019==1)  
  
unSampled\_inds\_2 <- which(covs\_df$sampled\_units\_2010.2019==0)  
  
sp\_2 <- c(covs\_df$heather\_true\_dist\_2010.2019[sampled\_inds\_2],  
 p\_2[unSampled\_inds\_2])  
  
r\_2 <- covs\_df$heather\_true\_dist\_2010.2019[sampled\_inds\_2] - p\_2[sampled\_inds\_2]  
  
qr\_mod\_2 <- randomForest(x = boot\_samp[,c(1:2,4:10)],   
 y = as.factor(boot\_samp$sampled\_units\_2010.2019),  
 norm.votes = T)  
  
preds\_2 <- predict(qr\_mod\_2,  
 newdata = covs\_df,  
 type = "prob",  
 index = 2)  
  
preds\_2[preds\_2 == 0] <- 0.01  
  
qr\_2 <- sum((1/preds\_2[sampled\_inds\_2,2]) \* covs\_df$heather\_true\_dist\_2010.2019[sampled\_inds\_2]) / sum(1/(preds\_2[sampled\_inds\_2,2]))  
  
DR\_2 <- 1/nrow(covs\_df) \* sum(r\_2/preds\_2[sampled\_inds\_2,2]) + 1/nrow(covs\_df) \* sum(p\_2)  
  
write.csv(data.frame(p\_1 = DR\_1,  
 p\_2 = DR\_2,  
 sp\_1 = mean(sp\_1),  
 sp\_2 = mean(sp\_2),  
 qr\_1 = qr\_1,  
 qr\_2 = qr\_2),  
 file = paste0("/home/users/rboyd/outputs/", runif(1, 0,100), ".csv"))

Instead of running this code, let's load the outputs.

dr <- read.csv("W:/PYWELL\_SHARED/Pywell Projects/BRC/Rob Boyd/NERC\_exploring\_frontiers/Data/boot.csv")  
  
DR <- data.frame(dr\_mean\_1 = mean(dr$p\_1),  
 dr\_lower\_1 = quantile(dr$p\_1, probs = 0.025),  
 dr\_upper\_1 = quantile(dr$p\_1, probs = 0.975),  
 dr\_mean\_2 = mean(dr$p\_2),  
 dr\_lower\_2 = quantile(dr$p\_2, probs = 0.025),  
 dr\_upper\_2 = quantile(dr$p\_2, probs = 0.975),  
 sp\_mean\_1 = mean(dr$sp\_1),  
 sp\_lower\_1 = quantile(dr$sp\_1, probs = 0.025),  
 sp\_upper\_1 = quantile(dr$sp\_1, probs = 0.975),  
 sp\_mean\_2 = mean(dr$sp\_2),  
 sp\_lower\_2 = quantile(dr$sp\_2, probs = 0.025),  
 sp\_upper\_2 = quantile(dr$sp\_2, probs = 0.975),  
 qr\_mean\_1 = mean(dr$qr\_1),  
 qr\_lower\_1 = quantile(dr$qr\_1, probs = 0.025),  
 qr\_upper\_1 = quantile(dr$qr\_1, probs = 0.975),  
 qr\_mean\_2 = mean(dr$qr\_2),  
 qr\_lower\_2 = quantile(dr$qr\_2, probs = 0.025),  
 qr\_upper\_2 = quantile(dr$qr\_2, probs = 0.975)); DR

## dr\_mean\_1 dr\_lower\_1 dr\_upper\_1 dr\_mean\_2 dr\_lower\_2 dr\_upper\_2 sp\_mean\_1  
## 2.5% 0.3286131 0.3220561 0.3361233 0.2976623 0.2941639 0.3015666 0.310216  
## sp\_lower\_1 sp\_upper\_1 sp\_mean\_2 sp\_lower\_2 sp\_upper\_2 qr\_mean\_1 qr\_lower\_1  
## 2.5% 0.3081725 0.3123039 0.2872833 0.2862387 0.2884778 0.3137386 0.3022572  
## qr\_upper\_1 qr\_mean\_2 qr\_lower\_2 qr\_upper\_2  
## 2.5% 0.3277224 0.2826154 0.278257 0.2902654

Perhaps more familiar to ecologists than the above approaches is subsampling. Our approach was to draw stratified random samples of size N/10~22,900 with replacement from the original samples. The subsample mean is the estimator of the population mean. Note that I bootstrap the procedure, which is necessary because there is a random element to the downsampling, and the estimated means are sensitive to it.

## next let's try subsampling   
  
samp\_p1 <- pop\_p1[pop\_p1$sampled\_units\_1987.1999 == 1, ]  
  
samp\_p1$id <- paste0(samp\_disc\_p1$PA\_1987,samp\_disc\_p1$UKelv)  
  
samp\_p2 <- pop\_p2[pop\_p2$sampled\_units\_2010.2019 == 1, ]  
  
samp\_p2$id <- paste0(samp\_disc\_p2$PA\_2010,samp\_disc\_p2$UKelv)  
  
df\_1 <- cells\_p1  
  
df\_1$code <- paste0(df\_1$PA\_1987, df\_1$UKelv)  
  
df\_1$size <- round(df\_1$Freq/10)  
  
df\_2 <- cells\_p2  
  
df\_2$code <- paste0(df\_2$PA\_2010, df\_2$UKelv)  
  
df\_2$size <- round(df\_2$Freq/10)  
  
subSamp <- function(id, Y, samp, df) {  
   
 sub <- sapply(df$code,  
 function(x) sample(samp[samp$id==x, c(Y)],  
 size = df$size[df$code==x]))  
   
 sub <- do.call("c", sub)  
   
 return(mean(sub))  
   
}  
  
subs\_p1 <- sapply(1:1000,  
 subSamp,  
 Y = "heather\_true\_dist\_1987.1999",  
 samp = samp\_p1,  
 df = df\_1)  
  
sub\_samp\_p1\_mean <- c(mean = mean(subs\_p1),  
 lower = quantile(subs\_p1, probs = 0.025),  
 upper = quantile(subs\_p1, probs = 0.975));sub\_samp\_p1\_mean

## mean lower.2.5% upper.97.5%   
## 0.3154849 0.3110463 0.3199767

subs\_p2 <- sapply(1:1000,  
 subSamp,  
 Y = "heather\_true\_dist\_2010.2019",  
 samp = samp\_p2,  
 df = df\_2)  
  
sub\_samp\_p2\_mean <- c(mean = mean(subs\_p2),  
 lower = quantile(subs\_p2, probs = 0.025),  
 upper = quantile(subs\_p2, probs = 0.975));sub\_samp\_p2\_mean

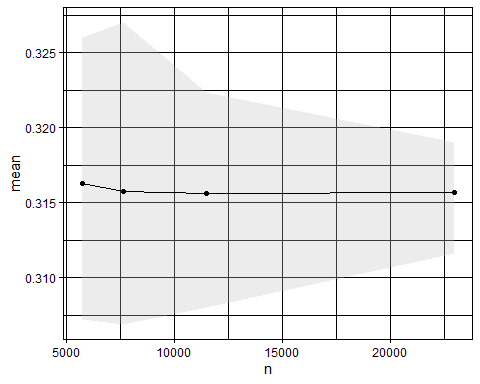
## mean lower.2.5% upper.97.5%   
## 0.2891425 0.2846729 0.2936073

The decision to create subsamples of size N/10 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(denom) {  
   
df\_1$size <- round(df\_1$Freq/denom)  
  
subs <- sapply(1:100,  
 subSamp,  
 Y = "heather\_true\_dist\_1987.1999",  
 samp = samp\_p1,  
 df = df\_1)  
  
out <- data.frame(mean = mean(subs),  
 lower = quantile(subs, probs = 0.025),  
 upper = quantile(subs, probs = 0.975),  
 n = nrow(pop)/denom)  
   
out  
}  
  
sens <- lapply(c(40,30,20,10),  
 getSens)  
  
sens <- do.call("rbind", sens)  
  
sens

## mean lower upper n  
## 2.5% 0.3163101 0.3071821 0.3260497 5739.6  
## 2.5%1 0.3157201 0.3068904 0.3270321 7652.8  
## 2.5%2 0.3155871 0.3080292 0.3223868 11479.2  
## 2.5%3 0.3157074 0.3116125 0.3190195 22958.4

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). It involes constructing a hierarchical model predicting mean occupancy in each poststratum (same strata as earlier) based on the auxiliaries. 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, so we do not actually implement it here.

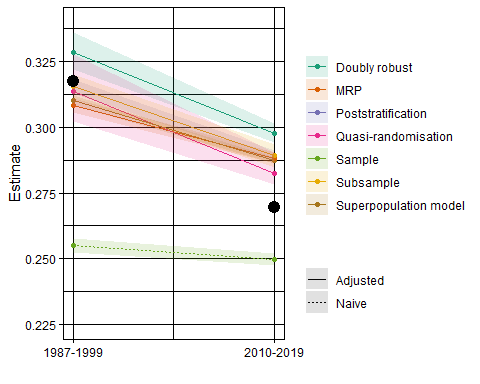
## fit model  
fit <- stan\_glmer(  
 heather\_true\_dist\_1987.1999 ~ 1 + (1 | allPACoverage) +   
 (1 | UKelv) +  
 (1 | allPACoverage:UKelv),  
 family = binomial(link = "logit"),  
 data = samp\_disc\_p1,  
 chains = 5,  
 iter = 1000,  
 cores = 5  
)  
  
posterior\_prob <- posterior\_linpred(fit, transform = T, newdata = cells)  
  
poststrat\_prob <- posterior\_prob %\*% cells$Freq / sum(cells$Freq)  
  
print(poststrat\_prob)  
  
#write.csv(poststrat\_prob, "/home/users/rboyd/outputs/mrp\_p1.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)  
  
## period 2  
fit2 <- stan\_glmer(  
 heather\_true\_dist\_2010.2019 ~ 1 + (1 | allPACoverage) +   
 (1 | UKelv) +  
 (1 | allPACoverage:UKelv),  
 family = binomial(link = "logit"),  
 data = samp\_disc\_p2,  
 chains = 5,  
 iter = 1000,  
 cores = 5  
)  
  
print(fit2)  
  
posterior\_prob\_p2 <- posterior\_linpred(fit2, transform = TRUE, newdata = cells)  
  
poststrat\_prob\_p2 <- posterior\_prob\_p2 %\*% cells$Freq / sum(cells$Freq)  
  
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))

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/bias\_mitigation\_ms/submission\_2/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/bias\_mitigation\_ms/submission\_2/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(1,2,1,2,1,2,1,2,1,2,1,2,1,2,1,2),  
 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], DR$qr\_mean\_1, DR$qr\_mean\_2, DR$sp\_mean\_1, DR$sp\_mean\_2, model\_popn\_pref[1],model\_popn\_pref\_p2[1], sub\_samp\_p1\_mean[1], sub\_samp\_p2\_mean[1],DR$dr\_mean\_1, DR$dr\_mean\_2),  
 type = c("Population", "Population", "Sample", "Sample", "Poststratification", "Poststratification", "Quasi-randomisation", "Quasi-randomisation", "Superpopulation model", "Superpopulation model", "MRP", "MRP", "Subsample", "Subsample", "Doubly robust", "Doubly robust"),  
 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], DR$qr\_lower\_1, DR$qr\_lower\_2, DR$sp\_lower\_1, DR$sp\_lower\_2, model\_popn\_pref[2], model\_popn\_pref\_p2[2], sub\_samp\_p1\_mean[2], sub\_samp\_p2\_mean[2], DR$dr\_lower\_1, DR$dr\_lower\_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], DR$qr\_upper\_1, DR$qr\_upper\_2, DR$sp\_upper\_1, DR$sp\_upper\_2, model\_popn\_pref[3], model\_popn\_pref\_p2[3], sub\_samp\_p1\_mean[3], sub\_samp\_p2\_mean[3], DR$dr\_upper\_1, DR$dr\_upper\_2))  
  
  
y1 <- 0.3173697  
y2 <- 0.2695963  
  
plotDat$adjust <- ifelse(plotDat$type == "Sample", "Naive", "Adjusted")  
  
ggplot(data = plotDat[-c(1,2),], aes(x = p, y = est, colour = type, fill = type, linetype = adjust)) +  
 geom\_point() +  
 geom\_line() +  
 theme\_linedraw() +  
 geom\_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.15, colour = NA) +  
 labs(x = "",  
 y = "Estimate",  
 fill = "",  
 colour = "") +  
 scale\_x\_continuous(breaks = c(1,2), labels = c("1987-1999", "2010-2019")) +  
 ylim(c(0.225,0.34)) +   
 labs(linetype = "") +  
 geom\_point(data = data.frame(x = c(1, 2), y = c(0.3173697, 0.2695963)), aes(x = x, y = y), colour = "black", size = 4, inherit.aes = F) +  
 scale\_color\_brewer(palette="Dark2") +  
 scale\_fill\_brewer(palette="Dark2")



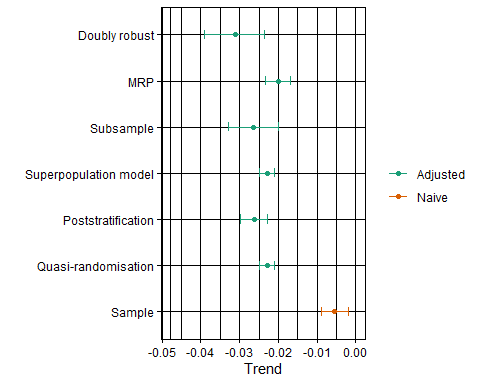
## Estimating the trend in mean occupancy

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   
  
diffqr <- dr$sp\_2 - dr$sp\_1  
  
qr\_up <- quantile(diffqr, probs = 0.975)  
  
qr\_low <- quantile(diffqr, probs = 0.025)  
  
qr\_mean <- mean(diffqr)  
  
#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  
diffsp <- dr$sp\_2 - dr$sp\_1  
  
sp\_up <- quantile(diffsp, probs = 0.975)  
  
sp\_low <- quantile(diffsp, probs = 0.025)  
  
sp\_mean <- mean(diffsp)  
  
#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  
  
## doubly robust   
  
diffDR <- dr$p\_2 - dr$p\_1  
  
dr\_up <- quantile(diffDR, probs = 0.975)  
  
dr\_low <- quantile(diffDR, probs = 0.025)  
  
dr\_mean <- mean(diffDR)  
  
## subsampling  
  
diffsSub <- subs\_p2 - subs\_p1  
  
#diffs <- diffs[,1]  
  
sub\_up <- quantile(diffsSub, probs = 0.975)  
  
sub\_low <- quantile(diffsSub, probs = 0.025)  
  
sub\_mean <- mean(diffsSub)  
  
## MRP   
  
diffs <- poststrat\_prob\_p2 - poststrat\_prob  
  
#diffs <- diffs[,1]  
  
MRP\_up <- quantile(diffs, probs = 0.975)  
  
MRP\_low <- quantile(diffs, probs = 0.025)  
  
MRP\_mean <- mean(diffs)  
  
trends$difference <- c(trends$difference[1:3], qr\_mean, sp\_mean, MRP\_mean, sub\_mean, dr\_mean)  
   
trends$lower <- c(NA, sample\_mean\_lower, ps\_mean\_lower, qr\_low, sp\_low, MRP\_low, sub\_low, dr\_low)  
  
trends$upper <- c(NA, sample\_mean\_upper, ps\_mean\_upper, qr\_up, sp\_up, MRP\_up, sub\_up, dr\_up)  
  
trends$estimator[4] <- "Quasi-randomisation"  
  
trends$estimator[5] <- "Superpopulation model"  
  
trends$estimator <- factor(trends$estimator, levels = c("Population", "Sample", "Quasi-randomisation",   
 "Poststratification", "Superpopulation model", "Subsample", "MRP",  
 "Doubly robust"))  
  
trends$adjusted <- ifelse(trends$estimator == "Sample", "Naive", "Adjusted")  
  
ggplot(data = trends[-1,], aes(x = difference, y = estimator, colour = adjusted)) +  
 geom\_point() +   
 theme\_linedraw() +  
 geom\_vline(xintercept = -0.047773364) +  
 geom\_vline(xintercept = 0, linetype = 3) +  
 labs(x = "Trend",  
 y = "") +  
 geom\_errorbar(aes(xmin = lower, xmax = upper, width = .2)) +  
 labs(colour = "") +  
 scale\_color\_brewer(palette="Dark2")



## 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 from the quasi-randomisation sample inclusion model, the superpopulation model and poststratification, because the R package survey will provide us with the weights (it already has for poststratification). I previously estimated the inclusion probabilities using the random forest for quasi-randomisation, which are in pop. For the superpopulation model, we can use the function calibrate in the survey package.

## design objects with quasi-randomisation weights  
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)  
  
## superpopulation model designs  
## first, calculate sums of the auxiliary variables  
aux\_tots\_p1 <- c(nrow(pop\_aux\_cont\_p1), colSums(pop\_aux\_cont\_p1))  
  
names(aux\_tots\_p1)[1] <- "(Intercept)"  
  
aux\_tots\_p2 <- c(nrow(pop\_aux\_cont\_p2), colSums(pop\_aux\_cont\_p2))  
  
names(aux\_tots\_p2)[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 = ~ PA\_1987 + UKelv,  
 population = aux\_tots\_p1,  
 calfun="linear")  
  
sum(weights(calib\_design\_p1))

## [1] 229584

summary(weights(calib\_design\_p1))

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -0.1788 1.8684 2.1461 2.3470 2.5834 8.7167

calib\_design\_p2 <- calibrate(design = pre\_calib\_design\_p2,  
 formula = ~ PA\_2010 + UKelv,  
 population = aux\_tots\_p2,  
 calfun="linear")  
  
sum(weights(calib\_design\_p2))

## [1] 229584

summary(weights(calib\_design\_p2))

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.058 1.392 1.525 1.624 1.742 4.187

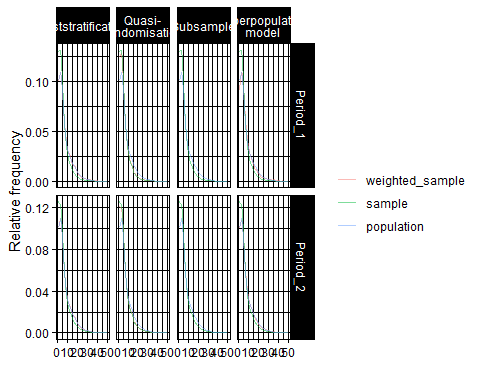
I previously stored the estimated sample inclusion probabilities from the quasi-randomisation procedure, so they are already in pop.

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, df) {  
   
 sub <- lapply(1:iter,  
 function(x) {  
  
 subSamp <- lapply(df$code,  
 function(z) {  
   
 y <- dat[dat$id==z, ]  
   
 y[sample(1:nrow(y), size = df$size[df$code==z], replace = T),]  
   
 }  
 )  
  
 subSamp <- do.call("rbind", subSamp)  
  
 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\_p1,  
 breaks = 50,   
 var = "UKelv",  
 period = "Period\_1",  
 iter = 100,  
 bins = cut(pop$UKelv, breaks = 50),  
 df = df\_1)  
  
  
# elevation  
relF\_p2\_elev <- getRelFreqs(dat = samp\_p2,  
 breaks = 50,   
 var = "UKelv",  
 period = "Period\_2",  
 iter = 100,  
 bins = cut(pop$UKelv, breaks = 50),  
 df\_2)

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.

## function to make relative frequency plots. It'll do it for most estimators, but we have to  
## pass the relative frequencies from the subsamples to the function  
  
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(alpha = 0.5) +  
 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

 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)))  
  
}  
  
## elevation in period 1  
  
auxImprovement(dat = p\_elev,  
 period = "Period\_1",  
 estimator = "Poststratification")

## mae\_samp mae\_est  
## 1 0.004377116 0.0005500176

## elevation in period 2  
  
auxImprovement(dat = p\_elev,  
 period = "Period\_2",  
 estimator = "Poststratification")

## mae\_samp mae\_est  
## 1 0.003082943 0.0003152803

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.

## Data defect correlations

We will now consider the representativeness of the samples and how the adjustment methods affect it in a slightly different way. A sample is unrepresentative if there is a correlation between a binary variable, taking the value 1 if a population unit is in the sample and 0 otherwise, and the variable of interest (i.e. if the variable of interest differs between sampled and non-sampled sites). This is the data defect correlation, or ddc. In our samples, the ddc's were -0.115 in period 1 and -0.057 in period two.

## period 1   
  
cor(pop$heather\_true\_dist\_1987.1999, pop$sampled\_units\_1987.1999)

## [1] -0.1154003

## period 2  
  
cor(pop$heather\_true\_dist\_2010.2019, pop$sampled\_units\_2010.2019)

## [1] -0.05699561

The partial correlations between occupancy and sample inclusion, which are conditional on the auxiliary variables, are lower in magnitude than the original ddc's (-0.018 and 0.035). This tells us that the samples are more representative after controlling for the auxiliaires. However, the ddc's are still appreciably non-zero, which means that the samples remain unrepresentative, albeit less so than before. See the values in row one and column two below.

## period 1  
  
pcor(data.frame(pop$heather\_true\_dist\_1987.1999,  
 pop$sampled\_units\_1987.1999,  
 pop$PA\_1987,  
 pop$UKelv))$estimate

## pop.heather\_true\_dist\_1987.1999  
## pop.heather\_true\_dist\_1987.1999 1.00000000  
## pop.sampled\_units\_1987.1999 -0.01837654  
## pop.PA\_1987 0.01116507  
## pop.UKelv 0.44972615  
## pop.sampled\_units\_1987.1999 pop.PA\_1987  
## pop.heather\_true\_dist\_1987.1999 -0.01837654 0.01116507  
## pop.sampled\_units\_1987.1999 1.00000000 0.05599042  
## pop.PA\_1987 0.05599042 1.00000000  
## pop.UKelv -0.18772386 0.06410836  
## pop.UKelv  
## pop.heather\_true\_dist\_1987.1999 0.44972615  
## pop.sampled\_units\_1987.1999 -0.18772386  
## pop.PA\_1987 0.06410836  
## pop.UKelv 1.00000000

## period 2  
  
pcor(data.frame(pop$heather\_true\_dist\_2010.2019,  
 pop$sampled\_units\_2010.2019,  
 pop$PA\_2010,  
 pop$UKelv))$estimate

## pop.heather\_true\_dist\_2010.2019  
## pop.heather\_true\_dist\_2010.2019 1.00000000  
## pop.sampled\_units\_2010.2019 0.03481063  
## pop.PA\_2010 0.14007138  
## pop.UKelv 0.35135854  
## pop.sampled\_units\_2010.2019 pop.PA\_2010  
## pop.heather\_true\_dist\_2010.2019 0.03481063 0.14007138  
## pop.sampled\_units\_2010.2019 1.00000000 0.01549302  
## pop.PA\_2010 0.01549302 1.00000000  
## pop.UKelv -0.22523191 0.19675714  
## pop.UKelv  
## pop.heather\_true\_dist\_2010.2019 0.3513585  
## pop.sampled\_units\_2010.2019 -0.2252319  
## pop.PA\_2010 0.1967571  
## pop.UKelv 1.0000000

## Testing whether the data satisfy the assumptions of a "NMAR" method

The adjustment methods that we have considered all rest on the bold assumption that non-sampled sites are "Missing At Random" from the dataset; that is, sample inclusion is independent of occupancy given the auxiliaries. We have just shown that this assumption does not hold, because the partial correlations between sample inclusion and occupancy are non-zero, so it would be useful to try some methods that do not assume MAR.

One example is the method of Tchetgen Tchetgen and Wirth (2017). The authors show that it is possible to reduce bias by including variables that are i) predictive of sample inclusion given the other auxiliaires; ii) independent of the variable of interest given the other auxiliaires; and iii) independent of the selection bias, which they define as the mean of the variable of interest in the sample minus the mean of the variable of interest at non-sampled sites.

We tested three variables to see if they met this criteria: 1) postcode density in each grid square and its nearest 299 neighbours; 2) the proportion of each grid square that is openly accessible to the public; and 3) the length of roads in each grid square and its nearest 299 neighbours. Postcode density met assumptions i and ii, but not assumption iii. We tested i and ii by looking at the partial correlations between sample inclusion, occupancy, the auxiliaries and variables 1-3 (one-by-one). We tested iii by regressing occupancy on protected area coverage, elevation, sample inclusion, postcode density and the interaction between postcode density and sample inclusion. There was strong evidence for the interaction effect, which tells us that the effect of postcode density on occupancy differs between sampled and non-sampled units, violating assumption iii.

## Assumption i and ii are testable using partial correlations  
  
## period 1  
  
# test road length  
pcor(data.frame(pop$heather\_true\_dist\_1987.1999[pop$sampled\_units\_1987.1999==1],   
 pop$PA\_1987[pop$sampled\_units\_1987.1999==1],   
 pop$UKelv[pop$sampled\_units\_1987.1999==1],   
 pop$road\_length\_299\_neighbours[pop$sampled\_units\_1987.1999==1]))$estimate

## pop.heather\_true\_dist\_1987.1999.pop.sampled\_units\_1987.1999....  
## pop.heather\_true\_dist\_1987.1999.pop.sampled\_units\_1987.1999.... 1.0000000  
## pop.PA\_1987.pop.sampled\_units\_1987.1999....1. 0.0338259  
## pop.UKelv.pop.sampled\_units\_1987.1999....1. 0.3011449  
## pop.road\_length\_299\_neighbours.pop.sampled\_units\_1987.1999.... -0.3259708  
## pop.PA\_1987.pop.sampled\_units\_1987.1999....1.  
## pop.heather\_true\_dist\_1987.1999.pop.sampled\_units\_1987.1999.... 0.033825903  
## pop.PA\_1987.pop.sampled\_units\_1987.1999....1. 1.000000000  
## pop.UKelv.pop.sampled\_units\_1987.1999....1. 0.079165228  
## pop.road\_length\_299\_neighbours.pop.sampled\_units\_1987.1999.... 0.004641944  
## pop.UKelv.pop.sampled\_units\_1987.1999....1.  
## pop.heather\_true\_dist\_1987.1999.pop.sampled\_units\_1987.1999.... 0.30114494  
## pop.PA\_1987.pop.sampled\_units\_1987.1999....1. 0.07916523  
## pop.UKelv.pop.sampled\_units\_1987.1999....1. 1.00000000  
## pop.road\_length\_299\_neighbours.pop.sampled\_units\_1987.1999.... -0.23498398  
## pop.road\_length\_299\_neighbours.pop.sampled\_units\_1987.1999....  
## pop.heather\_true\_dist\_1987.1999.pop.sampled\_units\_1987.1999.... -0.325970789  
## pop.PA\_1987.pop.sampled\_units\_1987.1999....1. 0.004641944  
## pop.UKelv.pop.sampled\_units\_1987.1999....1. -0.234983975  
## pop.road\_length\_299\_neighbours.pop.sampled\_units\_1987.1999.... 1.000000000

# test open access land  
pcor(data.frame(pop$heather\_true\_dist\_1987.1999[pop$sampled\_units\_1987.1999==1],  
 pop$PA\_1987[pop$sampled\_units\_1987.1999==1],   
 pop$UKelv[pop$sampled\_units\_1987.1999==1],   
 pop$openAccessGB[pop$sampled\_units\_1987.1999==1]))$estimate

## pop.heather\_true\_dist\_1987.1999.pop.sampled\_units\_1987.1999....  
## pop.heather\_true\_dist\_1987.1999.pop.sampled\_units\_1987.1999.... 1.00000000  
## pop.PA\_1987.pop.sampled\_units\_1987.1999....1. 0.01956481  
## pop.UKelv.pop.sampled\_units\_1987.1999....1. 0.23495707  
## pop.openAccessGB.pop.sampled\_units\_1987.1999....1. 0.41700144  
## pop.PA\_1987.pop.sampled\_units\_1987.1999....1.  
## pop.heather\_true\_dist\_1987.1999.pop.sampled\_units\_1987.1999.... 0.01956481  
## pop.PA\_1987.pop.sampled\_units\_1987.1999....1. 1.00000000  
## pop.UKelv.pop.sampled\_units\_1987.1999....1. 0.06868757  
## pop.openAccessGB.pop.sampled\_units\_1987.1999....1. 0.02755345  
## pop.UKelv.pop.sampled\_units\_1987.1999....1.  
## pop.heather\_true\_dist\_1987.1999.pop.sampled\_units\_1987.1999.... 0.23495707  
## pop.PA\_1987.pop.sampled\_units\_1987.1999....1. 0.06868757  
## pop.UKelv.pop.sampled\_units\_1987.1999....1. 1.00000000  
## pop.openAccessGB.pop.sampled\_units\_1987.1999....1. 0.29317822  
## pop.openAccessGB.pop.sampled\_units\_1987.1999....1.  
## pop.heather\_true\_dist\_1987.1999.pop.sampled\_units\_1987.1999.... 0.41700144  
## pop.PA\_1987.pop.sampled\_units\_1987.1999....1. 0.02755345  
## pop.UKelv.pop.sampled\_units\_1987.1999....1. 0.29317822  
## pop.openAccessGB.pop.sampled\_units\_1987.1999....1. 1.00000000

# test postcode density  
pcor(data.frame(pop$heather\_true\_dist\_1987.1999[pop$sampled\_units\_1987.1999==1],  
 pop$PA\_1987[pop$sampled\_units\_1987.1999==1],  
 pop$UKelv[pop$sampled\_units\_1987.1999==1],  
 pop$postcode\_density\_299\_neighbours[pop$sampled\_units\_1987.1999==1]))$estimate

## pop.heather\_true\_dist\_1987.1999.pop.sampled\_units\_1987.1999....  
## pop.heather\_true\_dist\_1987.1999.pop.sampled\_units\_1987.1999.... 1.00000000  
## pop.PA\_1987.pop.sampled\_units\_1987.1999....1. 0.03286590  
## pop.UKelv.pop.sampled\_units\_1987.1999....1. 0.39518078  
## pop.postcode\_density\_299\_neighbours.pop.sampled\_units\_1987.1999.... -0.07467107  
## pop.PA\_1987.pop.sampled\_units\_1987.1999....1.  
## pop.heather\_true\_dist\_1987.1999.pop.sampled\_units\_1987.1999.... 0.03286590  
## pop.PA\_1987.pop.sampled\_units\_1987.1999....1. 1.00000000  
## pop.UKelv.pop.sampled\_units\_1987.1999....1. 0.07719260  
## pop.postcode\_density\_299\_neighbours.pop.sampled\_units\_1987.1999.... -0.01626062  
## pop.UKelv.pop.sampled\_units\_1987.1999....1.  
## pop.heather\_true\_dist\_1987.1999.pop.sampled\_units\_1987.1999.... 0.3951808  
## pop.PA\_1987.pop.sampled\_units\_1987.1999....1. 0.0771926  
## pop.UKelv.pop.sampled\_units\_1987.1999....1. 1.0000000  
## pop.postcode\_density\_299\_neighbours.pop.sampled\_units\_1987.1999.... -0.1419408  
## pop.postcode\_density\_299\_neighbours.pop.sampled\_units\_1987.1999....  
## pop.heather\_true\_dist\_1987.1999.pop.sampled\_units\_1987.1999.... -0.07467107  
## pop.PA\_1987.pop.sampled\_units\_1987.1999....1. -0.01626062  
## pop.UKelv.pop.sampled\_units\_1987.1999....1. -0.14194079  
## pop.postcode\_density\_299\_neighbours.pop.sampled\_units\_1987.1999.... 1.00000000

## period 2  
  
#test road length  
pcor(data.frame(pop$heather\_true\_dist\_2010.2019[pop$sampled\_units\_2010.2019==1],   
 pop$PA\_2010[pop$sampled\_units\_2010.2019==1],   
 pop$UKelv[pop$sampled\_units\_2010.2019==1],   
 pop$road\_length\_299\_neighbours[pop$sampled\_units\_2010.2019==1]))$estimate

## pop.heather\_true\_dist\_2010.2019.pop.sampled\_units\_2010.2019....  
## pop.heather\_true\_dist\_2010.2019.pop.sampled\_units\_2010.2019.... 1.0000000  
## pop.PA\_2010.pop.sampled\_units\_2010.2019....1. 0.1347205  
## pop.UKelv.pop.sampled\_units\_2010.2019....1. 0.2543776  
## pop.road\_length\_299\_neighbours.pop.sampled\_units\_2010.2019.... -0.3710888  
## pop.PA\_2010.pop.sampled\_units\_2010.2019....1.  
## pop.heather\_true\_dist\_2010.2019.pop.sampled\_units\_2010.2019.... 0.13472049  
## pop.PA\_2010.pop.sampled\_units\_2010.2019....1. 1.00000000  
## pop.UKelv.pop.sampled\_units\_2010.2019....1. 0.18818131  
## pop.road\_length\_299\_neighbours.pop.sampled\_units\_2010.2019.... -0.06938316  
## pop.UKelv.pop.sampled\_units\_2010.2019....1.  
## pop.heather\_true\_dist\_2010.2019.pop.sampled\_units\_2010.2019.... 0.2543776  
## pop.PA\_2010.pop.sampled\_units\_2010.2019....1. 0.1881813  
## pop.UKelv.pop.sampled\_units\_2010.2019....1. 1.0000000  
## pop.road\_length\_299\_neighbours.pop.sampled\_units\_2010.2019.... -0.2453909  
## pop.road\_length\_299\_neighbours.pop.sampled\_units\_2010.2019....  
## pop.heather\_true\_dist\_2010.2019.pop.sampled\_units\_2010.2019.... -0.37108880  
## pop.PA\_2010.pop.sampled\_units\_2010.2019....1. -0.06938316  
## pop.UKelv.pop.sampled\_units\_2010.2019....1. -0.24539093  
## pop.road\_length\_299\_neighbours.pop.sampled\_units\_2010.2019.... 1.00000000

# test open access land  
pcor(data.frame(pop$heather\_true\_dist\_2010.2019[pop$sampled\_units\_2010.2019==1],  
 pop$PA\_2010[pop$sampled\_units\_2010.2019==1],   
 pop$UKelv[pop$sampled\_units\_2010.2019==1],   
 pop$openAccessGB[pop$sampled\_units\_2010.2019==1]))$estimate

## pop.heather\_true\_dist\_2010.2019.pop.sampled\_units\_2010.2019....  
## pop.heather\_true\_dist\_2010.2019.pop.sampled\_units\_2010.2019.... 1.0000000  
## pop.PA\_2010.pop.sampled\_units\_2010.2019....1. 0.1203897  
## pop.UKelv.pop.sampled\_units\_2010.2019....1. 0.2077046  
## pop.openAccessGB.pop.sampled\_units\_2010.2019....1. 0.4293818  
## pop.PA\_2010.pop.sampled\_units\_2010.2019....1.  
## pop.heather\_true\_dist\_2010.2019.pop.sampled\_units\_2010.2019.... 0.12038975  
## pop.PA\_2010.pop.sampled\_units\_2010.2019....1. 1.00000000  
## pop.UKelv.pop.sampled\_units\_2010.2019....1. 0.17854224  
## pop.openAccessGB.pop.sampled\_units\_2010.2019....1. 0.08271617  
## pop.UKelv.pop.sampled\_units\_2010.2019....1.  
## pop.heather\_true\_dist\_2010.2019.pop.sampled\_units\_2010.2019.... 0.2077046  
## pop.PA\_2010.pop.sampled\_units\_2010.2019....1. 0.1785422  
## pop.UKelv.pop.sampled\_units\_2010.2019....1. 1.0000000  
## pop.openAccessGB.pop.sampled\_units\_2010.2019....1. 0.2889538  
## pop.openAccessGB.pop.sampled\_units\_2010.2019....1.  
## pop.heather\_true\_dist\_2010.2019.pop.sampled\_units\_2010.2019.... 0.42938184  
## pop.PA\_2010.pop.sampled\_units\_2010.2019....1. 0.08271617  
## pop.UKelv.pop.sampled\_units\_2010.2019....1. 0.28895381  
## pop.openAccessGB.pop.sampled\_units\_2010.2019....1. 1.00000000

# test postcode density  
pcor(data.frame(pop$heather\_true\_dist\_2010.2019[pop$sampled\_units\_2010.2019==1],  
 pop$PA\_2010[pop$sampled\_units\_2010.2019==1],  
 pop$UKelv[pop$sampled\_units\_2010.2019==1],  
 pop$postcode\_density\_299\_neighbours[pop$sampled\_units\_2010.2019==1]))$estimate

## pop.heather\_true\_dist\_2010.2019.pop.sampled\_units\_2010.2019....  
## pop.heather\_true\_dist\_2010.2019.pop.sampled\_units\_2010.2019.... 1.0000000  
## pop.PA\_2010.pop.sampled\_units\_2010.2019....1. 0.1720517  
## pop.UKelv.pop.sampled\_units\_2010.2019....1. 0.3675228  
## pop.postcode\_density\_299\_neighbours.pop.sampled\_units\_2010.2019.... -0.0827533  
## pop.PA\_2010.pop.sampled\_units\_2010.2019....1.  
## pop.heather\_true\_dist\_2010.2019.pop.sampled\_units\_2010.2019.... 0.172051691  
## pop.PA\_2010.pop.sampled\_units\_2010.2019....1. 1.000000000  
## pop.UKelv.pop.sampled\_units\_2010.2019....1. 0.209247458  
## pop.postcode\_density\_299\_neighbours.pop.sampled\_units\_2010.2019.... -0.006937587  
## pop.UKelv.pop.sampled\_units\_2010.2019....1.  
## pop.heather\_true\_dist\_2010.2019.pop.sampled\_units\_2010.2019.... 0.3675228  
## pop.PA\_2010.pop.sampled\_units\_2010.2019....1. 0.2092475  
## pop.UKelv.pop.sampled\_units\_2010.2019....1. 1.0000000  
## pop.postcode\_density\_299\_neighbours.pop.sampled\_units\_2010.2019.... -0.1366479  
## pop.postcode\_density\_299\_neighbours.pop.sampled\_units\_2010.2019....  
## pop.heather\_true\_dist\_2010.2019.pop.sampled\_units\_2010.2019.... -0.082753302  
## pop.PA\_2010.pop.sampled\_units\_2010.2019....1. -0.006937587  
## pop.UKelv.pop.sampled\_units\_2010.2019....1. -0.136647914  
## pop.postcode\_density\_299\_neighbours.pop.sampled\_units\_2010.2019.... 1.000000000

## test assumption iii  
  
summary(lm(heather\_true\_dist\_1987.1999~PA\_1987 + UKelv + sampled\_units\_1987.1999 \* postcode\_density\_299\_neighbours, data = pop))

##   
## Call:  
## lm(formula = heather\_true\_dist\_1987.1999 ~ PA\_1987 + UKelv +   
## sampled\_units\_1987.1999 \* postcode\_density\_299\_neighbours,   
## data = pop)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.7205 -0.2505 -0.1516 0.3563 1.2972   
##   
## Coefficients:  
## Estimate Std. Error  
## (Intercept) 1.288e-01 1.700e-03  
## PA\_1987 5.306e-02 1.061e-02  
## UKelv 1.313e-03 5.794e-06  
## sampled\_units\_1987.1999 -2.163e-02 1.959e-03  
## postcode\_density\_299\_neighbours -8.298e-06 1.919e-07  
## sampled\_units\_1987.1999:postcode\_density\_299\_neighbours 4.608e-06 2.423e-07  
## t value Pr(>|t|)   
## (Intercept) 75.764 < 2e-16 \*\*\*  
## PA\_1987 4.999 5.75e-07 \*\*\*  
## UKelv 226.680 < 2e-16 \*\*\*  
## sampled\_units\_1987.1999 -11.039 < 2e-16 \*\*\*  
## postcode\_density\_299\_neighbours -43.247 < 2e-16 \*\*\*  
## sampled\_units\_1987.1999:postcode\_density\_299\_neighbours 19.016 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4104 on 229578 degrees of freedom  
## Multiple R-squared: 0.2227, Adjusted R-squared: 0.2226   
## F-statistic: 1.315e+04 on 5 and 229578 DF, p-value: < 2.2e-16

summary(lm(heather\_true\_dist\_2010.2019~PA\_2010 + UKelv + sampled\_units\_2010.2019 \* postcode\_density\_299\_neighbours, data = pop))

##   
## Call:  
## lm(formula = heather\_true\_dist\_2010.2019 ~ PA\_2010 + UKelv +   
## sampled\_units\_2010.2019 \* postcode\_density\_299\_neighbours,   
## data = pop)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.5206 -0.2277 -0.1454 0.2882 1.5439   
##   
## Coefficients:  
## Estimate Std. Error  
## (Intercept) 1.078e-01 2.023e-03  
## PA\_2010 2.432e-01 3.679e-03  
## UKelv 9.641e-04 5.908e-06  
## sampled\_units\_2010.2019 1.363e-02 1.982e-03  
## postcode\_density\_299\_neighbours -1.763e-05 4.145e-07  
## sampled\_units\_2010.2019:postcode\_density\_299\_neighbours 1.315e-05 4.284e-07  
## t value Pr(>|t|)   
## (Intercept) 53.305 < 2e-16 \*\*\*  
## PA\_2010 66.122 < 2e-16 \*\*\*  
## UKelv 163.179 < 2e-16 \*\*\*  
## sampled\_units\_2010.2019 6.877 6.11e-12 \*\*\*  
## postcode\_density\_299\_neighbours -42.531 < 2e-16 \*\*\*  
## sampled\_units\_2010.2019:postcode\_density\_299\_neighbours 30.683 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4012 on 229578 degrees of freedom  
## Multiple R-squared: 0.1826, Adjusted R-squared: 0.1826   
## F-statistic: 1.026e+04 on 5 and 229578 DF, p-value: < 2.2e-16