Supplementary Methods 2 for 'Uncertain competition coefficients undermine inferences about coexistence'

J. Christopher D. Terry

Contents

1 Introduction

This document is the second of two knit `.rmd` files that works through the steps of the reanalysis of *Small rainfall changes drive substantial changes in plant coexistence* by Van Dyke et al 2022 (Nature), to support a 'Matters Arising' response. Original data from the authors is available at https://doi.org/10.5281/zenodo.7083314

The original authors report that '*reduced rainfall altered the relative strength of the competition coefficients more strongly than the differences in demographic potential (Fig 3)*'. To examine whether this observation could be entirely due to the increased uncertainty in estimates α terms compared to λ terms, I tested if this result can also be observed in a null model that does not have any impact of the drought treatment.

In Section 3, I fit a seed production model in STAN using the `brms` R package. I used the same Beverton-Holt model structure $F_i \sim \lambda_i/(1 + \alpha_{ii}N_i + \alpha_{ij}N_j)$ as the original authors, with the exception that the rainfall treatments were not differentiated. Again following the original model, the predictions and seed counts were fit on a log-transformed scale assuming Gaussian error. Priors were chosen to be as uninformative as possible, but were bounded to the same extent as the original author's model. Each focal species was fit separately, and all models converged well.

From these models draws were taken from the posterior predictive distribution to generate simulated datasets of the same dimensions (e.g. number of samples per species and density combination), but without any 'true' distinction between treatments. These simulated datasets were then used in the original analysis pipeline using code from the original authors code repository. The background and justification of their methods is given in the original article.

In Section 4, I show that the fraction of draws where pairs show a change in coexistence outcome between treatments for the best-fit models (despite their being no difference between treatments) as being around 23%, but with strong differences between pairs.

In Section 5, I repeat the analysis shown in Fig.3 of the original article where a comparison is made between treatments of 1) the ratio of the demographic potential of the two species , and 2) the ratio of competition coefficients are compared for each species pair with a t-test. In the original analysis, the authors found that the difference between the treatments was larger in the competition coefficients than in the demographic potential ($p = 0.044$). Across a sample of 1000 simulated datasets, I found this same result in around half the cases, despite there being no 'true' difference between the treatments.

In Section 6, I subtract the average shifts observed in the null model from the observations, to generate a value for 'real' shifts caused by the treatment, and repeat the paired t-test examining the support for a greater shift in competition coefficient ratios over demographic potentials. This test does not find a significant difference.

In Section 7 I use the null model as a direct reference to compare the observed data against. This test also does not find a significant difference. Note that this section was written following a draft reply written by the original authors that conducted a similar analysis, but which has certain key issues. As such in this section I also demonstrate the necessity of a sufficiently large set of null model draws and carefully ordering the averaging operations within the overall analysis.

Combined with the lack of support for the treatment impacting the competition coefficients, these analyses strongly suggests that the authors' conclusion of the dominant role of changes to competition coefficients is driven by the increased uncertainty in their estimation.

2 Loading data

```
library(tidyverse)
library(brms)
library(posterior)
library(broom)
library(cowplot)

set.seed(1)

Path_to_Orig <- '../DroughtCompUncertainty/water_competition-2.0/Sedgwick_public/'

nls_boot_pairs<-read.csv(paste0(Path_to_Orig,"output/nls_boot_pairs_1000_full_model.csv"))
seed_data <- read.csv(paste0(Path_to_Orig,"data/drought_seed_production_data.csv"))

seed_data$Tr <- ifelse(seed_data$treat == "W", 1, 2)
seed_data <- seed_data %>%
  mutate( background = ifelse(is.na(background),
                              focal, background)) # where 0 background, naming after focal

seed_data$N_acwr <- ifelse(seed_data$background == "ACWR", seed_data$num_comp, 0)
seed_data$N_femi <- ifelse(seed_data$background == "FEMI", seed_data$num_comp, 0)
seed_data$N_homu <- ifelse(seed_data$background == "HOMU", seed_data$num_comp, 0)
seed_data$N_pler <- ifelse(seed_data$background == "PLER", seed_data$num_comp, 0)
seed_data$N_saco <- ifelse(seed_data$background == "SACO", seed_data$num_comp, 0)
seed_data$N_urli <- ifelse(seed_data$background == "URLI", seed_data$num_comp, 0)


spp_list <- sort(na.omit( unique(seed_data$focal)))

d = seed_data %>%
  mutate( Log_Seeds = log(num_seeds),
          Log_Comp = log(num_comp ),
          IsD = treat =='D',
          IsW = treat =='W')

total_pairs <- c("ACWR_FEMI", "ACWR_HOMU", "ACWR_PLER", "SACO_ACWR", "URLI_ACWR",
                 "HOMU_FEMI", "PLER_FEMI", "SACO_FEMI", "URLI_FEMI", "PLER_HOMU",
                 "SACO_HOMU", "URLI_HOMU","SACO_PLER", "URLI_PLER", "URLI_SACO")
```

3 Fitting generative model

3.1 Defining priors

Priors are very loose and bounds follow the same as the original NLS model. Only major difference is that lambda is fit on a log10 scale.

```
loose_priors_combined <-
  prior(normal(3, 10), nlpar = "LOG10lambda", lb = 1, ub = 5) +
  prior(normal(0, 10), nlpar = "aACWR",  lb = 0.001 )+
  prior(normal(0, 10), nlpar = "aFEMI",  lb = 0.001)+
```

```
  prior(normal(0, 10), nlpar = "aHOMU",  lb = 0.001)+
  prior(normal(0, 10), nlpar = "aPLER",  lb = 0.001)+
  prior(normal(0, 10), nlpar = "aSACO",  lb = 0.001)+
  prior(normal(0, 10), nlpar = "aURLI",  lb = 0.001)
```

3.2 Fitting generative model without treatment differences

```
model1_ACWR<-brm( bf(Log_Seeds~log((10^LOG10lambda)/(1+
                                                     aACWR*N_acwr+
                                                     aFEMI*N_femi+
                                                     aHOMU*N_homu+
                                                     aPLER*N_pler+
                                                     aSACO*N_saco+
                                                     aURLI*N_urli)),
                     LOG10lambda+aACWR+aFEMI+aHOMU+aPLER+aSACO+aURLI~1,
                     nl = TRUE),
                   data= filter(d, focal == 'ACWR'),
                   prior = loose_priors_combined)

### Refit with different data for each different species.
model1_FEMI<- update(model1_ACWR,newdata = filter(d, focal == 'FEMI') )
model1_HOMU<- update(model1_ACWR,newdata = filter(d, focal == 'HOMU') )
model1_PLER<- update(model1_ACWR,newdata = filter(d, focal == 'PLER') )
model1_SACO<- update(model1_ACWR,newdata = filter(d, focal == 'SACO') )
model1_URLI<- update(model1_ACWR,newdata = filter(d, focal == 'URLI') )

joint_model_list <- list(model1_ACWR,model1_FEMI,model1_HOMU,
                         model1_PLER,model1_SACO,model1_URLI)
save(joint_model_list,file =  'joint_model_list2')

### Gathering output summaries
bind_rows(model1_ACWR%>% summarise_draws() %>% mutate(focal = 'ACWR'),
          model1_FEMI%>% summarise_draws() %>% mutate(focal = 'FEMI'),
          model1_HOMU%>% summarise_draws() %>% mutate(focal = 'HOMU'),
          model1_PLER%>% summarise_draws() %>% mutate(focal = 'PLER'),
          model1_SACO%>% summarise_draws() %>% mutate(focal = 'SACO'),
          model1_URLI%>% summarise_draws() %>% mutate(focal = 'URLI')) -> AllJointFits

write_csv(AllJointFits, 'AllJointFits.csv')
```

3.3 Taking posteror draws

```
set.seed(1)
load(file = 'joint_model_list2')
## Draw from posterior predictive distribution
##  (includes model uncertainty and residual error)
predict_ACWR<- posterior_predict(joint_model_list[[1]], ndraws = 1000)
predict_FEMI<- posterior_predict(joint_model_list[[2]], ndraws = 1000)
predict_HOMU<- posterior_predict(joint_model_list[[3]], ndraws = 1000)
```

```
predict_PLER<- posterior_predict(joint_model_list[[4]], ndraws = 1000)
predict_SACO<- posterior_predict(joint_model_list[[5]], ndraws = 1000)
predict_URLI<- posterior_predict(joint_model_list[[6]], ndraws = 1000)
```

3.4 Function to rebuild data in original layout

```
data_ACWR<- filter(d, focal == 'ACWR')
data_FEMI<- filter(d, focal == 'FEMI')
data_HOMU<- filter(d, focal == 'HOMU')
data_PLER<- filter(d, focal == 'PLER')
data_SACO<- filter(d, focal == 'SACO')
data_URLI<- filter(d, focal == 'URLI')

Gen_simData_fromfit <- function(i){

  data_ACWR$SIM_DATA <- predict_ACWR[i,]
  data_FEMI$SIM_DATA <- predict_FEMI[i,]
  data_HOMU$SIM_DATA <- predict_HOMU[i,]
  data_PLER$SIM_DATA <- predict_PLER[i,]
  data_SACO$SIM_DATA <- predict_SACO[i,]
  data_URLI$SIM_DATA <- predict_URLI[i,]

  seed_data_WithSim <- bind_rows(data_ACWR,data_FEMI,
                                 data_HOMU, data_PLER,
                                 data_SACO,data_URLI)

  seed_data_WithSim$Simulation_i <- i
  return(seed_data_WithSim)
}
```

3.5 Utility functions to extract parameters and key quantities from simulation data

Code copies as closely as possible analyses in `nls_orig_data.R`.

```
##Calculate Stabilizing niche differences
stabilizing_niche_diff_func <- function(df, species1, species2, treat) {

  aij <- with(df, alpha[focal == species1 & competitor == species2 & treatment== treat])
  aji <- with(df, alpha[focal == species2 & competitor == species1 & treatment== treat])
  ajj <- with(df, alpha[focal == species2 & competitor == species2 & treatment== treat])
  aii <- with(df, alpha[focal == species1 & competitor == species1 & treatment== treat])
  snd <- (1 - sqrt((aij * aji)/(ajj * aii)))
  return(snd)
}
#eta_i equation function
get_ni_func<- function(df, species, treat){
  lambda <- with(df, lambda[ focal == species & treatment == treat])[1]
  gi <- with( df, g[focal == species & treatment == treat ])[1]
  si <- with( df, s[focal == species & treatment == treat])[1]
```

```
  ni<- ((lambda*gi)/(1-((1-gi)*si)))
  return(ni[1])
}

#Get fitness differences ------
fitness_diff_func <- function(df, species1, species2, treat) {

  ni <- with(df, ni[focal == species1 & treatment == treat])[1]
  nj <- with(df, ni[focal == species2 & treatment == treat])[1]
  aij <- with(df, alpha[focal == species1 & competitor == species2 & treatment == treat ])
  aji <- with(df, alpha[focal == species2 & competitor == species1 & treatment == treat])
  ajj <- with(df, alpha[focal == species2 & competitor == species2 & treatment == treat])
  aii <- with(df, alpha[focal == species1 & competitor == species1 & treatment == treat])
  nn<- (nj-1)/(ni-1)
  aa<- sqrt((aij * aii)/(ajj * aji))
  FDij <- nn*aa
  return(FDij[1])
}
```

```
Calc_all_fit_sim<- function(seed_data_WithSim){

  spp_combos <- expand.grid(species = spp_list)
  out <- list()
  for( i in 1:nrow( spp_combos )){
    temp_data <-filter(seed_data_WithSim, ### Now using simulated data
                       focal == spp_combos[i,1])

    fit_test<-NULL
    try(
      ###################   nb predicting SIM_DATA (log scale predicted seeds)
      fit_test<- nls(SIM_DATA ~log(lambda[Tr]/(1+a_ACWR[Tr]*N_acwr+a_FEMI[Tr]*N_femi+
                                                 a_HOMU[Tr]*N_homu+a_PLER[Tr]*N_pler+
                                                 a_SACO[Tr]*N_saco+a_URLI[Tr]*N_urli)),
                     data=temp_data, start=list('lambda'= c(100,100),
                                                a_ACWR=c(0.1, 0.1), a_FEMI=c(0.1, 0.1),
                                                a_HOMU=c(0.1, 0.1), a_PLER=c(0.1, 0.1),
                                                a_SACO=c(0.1, 0.1),a_URLI=c(0.1, 0.1)),
                     lower = c( 1, 1, rep(.001, 12) ),
                     upper = c(10000, 10000, rep(2, 12)),
                     control = list(maxiter = 100000),
                     algorithm = 'port')

    )
    if(is.null(fit_test)){print('Converge Failed');print(temp_data);return(0)}

    df <- tidy(fit_test)

    df$focal <- spp_combos[i,1]

    df <- df %>%
      select(term , estimate, focal) %>%
      spread(term, estimate) %>%
      pivot_longer( cols = starts_with("a"),
```

```
                    names_to = c("competitor","treatment"),
                    names_prefix = "a_",
                    names_sep = 4,
                    values_to ="alpha"
    )

  df$lambda<-ifelse(df$treatment == 1, df$lambda1, df$lambda2)
  df$lambda1<-NULL
  df$lambda2<-NULL

  out[[i]] <- df
}


all_fit <- do.call(rbind.data.frame, out)
all_fit$snd <- 0

for(i in 1:nrow(all_fit)) {
  sp1 <- all_fit[i, "focal"] %>% unlist
  sp2 <- all_fit[i, "competitor"] %>% unlist
  trt <- all_fit[i , "treatment"] %>% unlist
  snd <- stabilizing_niche_diff_func(all_fit, sp1, sp2, trt)
  all_fit[i, "snd"] <- snd
}

##Get eta_i-------
#seed survival and germination data
s_g_data <- read.csv(paste0(Path_to_Orig,"data/s_g_data.csv"))
all_fit <- merge(all_fit, s_g_data, by = "focal")
all_fit$X<- NULL
all_fit$ni <- 0
for(i in 1:nrow(all_fit)) {
  sp1 <- all_fit[i, "focal"] %>% unlist
  trt <- all_fit[i , "treatment"] %>% unlist
  ni <- get_ni_func(all_fit, sp1, trt)
  all_fit[i, "ni"] <- ni
}

#add fitness difference column to data frame
all_fit$fd <- 0

for(i in 1:nrow(all_fit)) {

  sp1 <- all_fit[i, "focal"] %>% unlist
  sp2 <- all_fit[i, "competitor"] %>% unlist
  trt <- all_fit[i , "treatment"] %>% unlist
  fitdif <- fitness_diff_func(all_fit, sp1, sp2, trt)
  all_fit[i, "fd"] <- fitdif
}

all_fit$focal <- as.character(all_fit$focal)
# identifying greater fitness difference
all_fit$fd_superior <- ifelse(all_fit$fd < 1, 1/all_fit$fd, all_fit$fd)
```

```
  all_fit$fd_sup_sp <- ifelse(all_fit$fd <= 1, all_fit$focal, all_fit$competitor)

  all_fit$coexist <- ifelse((all_fit$snd > (1-1/all_fit$fd_superior)), 1, 0 )
  all_fit$sp_pair <- paste(all_fit$focal, all_fit$competitor, sep = "_")

  return(all_fit)
}
```

3.6 Taking draws from generative model

```
1:1000 %>% ## about 16 fail to converge
  map(Gen_simData_fromfit)%>%
  map(Calc_all_fit_sim) -> List_Null_Fits

save(List_Null_Fits, file='List_Null_Fits')
```

```
load('List_Null_Fits')
```

4 What is the assigned probability of a coexistence shift being 'observed' when no real effect at all?

Using data derived from the no-treatment null model, what proportion of species pairs change coexistence prediction?

```
Calc_if_shiftprediction <- function(all_fit){
  if(!is.data.frame(all_fit)){return(NULL)}

  all_fit %>%
    filter(sp_pair %in% total_pairs) %>%
    group_by(sp_pair)%>%
    summarise(ChangeCoexist = sum(coexist)==1 ) -> PairsChangeCoexist
  ## ^^^ NB this doesn't include cases where the winning species changes

  return(PairsChangeCoexist)
}

List_Null_Fits %>%
  map_df(Calc_if_shiftprediction, .id = 'Rep')-> PredictedChangesFromNull

PredictedChangesFromNull %>%
  group_by(sp_pair) %>%
  summarise(Frac_ChangeCoexist =mean(ChangeCoexist)) %>%
  arrange(Frac_ChangeCoexist)
```

```
## # A tibble: 15 x 2
##    sp_pair   Frac_ChangeCoexist
##    <chr>                  <dbl>
##  1 SACO_FEMI             0.0152
```

```
##  2 URLI_PLER             0.0254
##  3 SACO_HOMU             0.0811
##  4 PLER_HOMU             0.124
##  5 SACO_PLER             0.195
##  6 PLER_FEMI             0.208
##  7 URLI_HOMU             0.209
##  8 ACWR_PLER             0.214
##  9 URLI_FEMI             0.266
## 10 ACWR_HOMU             0.285
## 11 ACWR_FEMI             0.335
## 12 URLI_ACWR             0.347
## 13 HOMU_FEMI             0.351
## 14 SACO_ACWR             0.386
## 15 URLI_SACO             0.400
```

```
PredictedChangesFromNull %>%
  summarise(Frac_ChangeCoexist =mean(ChangeCoexist))
```

```
## # A tibble: 1 x 1
##   Frac_ChangeCoexist
##                <dbl>
## 1              0.229
```

5 Repeating original Figure 3 analysis on draws from null model

This test is to identify the probability of seeing the significantly larger effect of changes to competition coefficients over demographic potentials (a result described in the original paper) in a null model that has no treatment effects.

Code sourced where possible from `n_alpha_ratios.r` by the original authors.

```
igr_change <-function(i, all_fit) { #invasion growth rate ratios
  #Which ratio changes more in invasion growth rate inequality?
  foc <- all_fit$focal[i]
  comp <- all_fit$competitor[i]

  nj_D <- filter(all_fit,focal == comp & treatment == 2)$ni[1]
  ni_D <- filter(all_fit,focal == foc & treatment == 2)$ni[1]
  ajj_D <- filter(all_fit, focal == comp & competitor == comp & treatment == 2)$alpha
  aij_D <- filter(all_fit, focal == foc & competitor == comp & treatment == 2)$alpha
  n_ratio_D = log10((ni_D-1)/(nj_D-1))
  a_ratio_D = log10(ajj_D/aij_D)

  nj_W <- filter(all_fit,focal == comp & treatment == 1)$ni[1]
  ni_W <- filter(all_fit,focal == foc & treatment == 1)$ni[1]
  ajj_W <- filter(all_fit, focal == comp & competitor == comp & treatment == 1)$alpha
  aij_W<- filter(all_fit, focal == foc & competitor == comp & treatment == 1)$alpha
  n_ratio_W = log10((ni_W-1)/(nj_W-1))
  a_ratio_W = log10(ajj_W/aij_W)

  nc<-abs(n_ratio_W - n_ratio_D)
  ac<-abs(a_ratio_W - a_ratio_D)
```

```
  return(data.frame( 'focal' = foc,  'comp' = comp,
                     "n_ratio_DRY"=n_ratio_D,"a_ratio_DRY"= a_ratio_D,
                     "n_ratio_WET"=n_ratio_W, "a_ratio_WET"= a_ratio_W,
                     nc = nc, ac=ac))
}

Calc_t.test_ratiodiffs <- function(all_fit){
  print('i')
  if(!is.data.frame(all_fit)){return(NULL)}

  ToCalc<- which(all_fit$sp_pair %in% total_pairs) ## only calc those needed. Keeps selected pairs cons
  Comp_ratio_diffs <- map_df(ToCalc, igr_change, all_fit)

  Comp_ratio_diffs %>%
    filter(focal !=comp ) %>%
    distinct(focal, comp, .keep_all = TRUE) %>%
    mutate( sp_pairs = paste0(focal, '_', comp)) %>%
    filter( sp_pairs %in% total_pairs) %>%
    mutate( larger = ifelse(abs(ac)> abs(nc), "a", 'n')) -> data_for_test


  t.test_out <- t.test(data_for_test$nc, data_for_test$ac, paired = T)
  return(tidy(t.test_out))
}
```

5.0.1 Testing pipeline can reproduce original results

Expect a slight difference between best fit and average of bootstrap. But close enough to original paper's finding of p=0.044 to be confident it is ok.

```
d %>%
  mutate(SIM_DATA = log(num_seeds))%>%
  Calc_all_fit_sim -> OrigData_newAnalysis
Calc_t.test_ratiodiffs(all_fit = OrigData_newAnalysis)
```

```
## [1] "i"
```

```
## # A tibble: 1 x 8
##   estimate statistic p.value parameter conf.low conf.high method     alternative
##      <dbl>     <dbl>   <dbl>     <dbl>    <dbl>     <dbl> <chr>      <chr>
## 1   -0.196     -2.16  0.0488        14   -0.391  -0.00116 Paired t-~ two.sided
```

5.1 Replicating analysis on simulated data

```
List_Null_Fits%>%
  map_df(Calc_t.test_ratiodiffs )-> t.test_FromSims_1000

write_csv(t.test_FromSims_1000, 't.test_FromSims_1000.csv')
# ^^ not very optimised so takes nearly an hour to run
```

```
t.test_FromSims_1000 <- read_csv('t.test_FromSims_1000.csv')
mean(t.test_FromSims_1000$p.value < 0.05)
```

```
## [1] 0.484787
```

```
t.test_FromSims_1000 %>%
  count(statistic   <0 & p.value <0.05 )
```

```
## # A tibble: 2 x 2
##   `statistic < 0 & p.value < 0.05`     n
##   <lgl>                            <int>
## 1 FALSE                              509
## 2 TRUE                               477
```

6 Using the null model to account for differential uncertainty in the two quantities.

```
ChangeInLogRatio <- function(pairwiseValues){
  ## Calculate change in log_ratio of demographic potentials log_10[(\eta_j -1)/(\eta_i -1)]

  if(!is.data.frame(pairwiseValues)){return(NULL)}
  ## Get alpha_jj
  IntraAlpha = pairwiseValues %>%
    filter(   focal== competitor  ) %>%
    select( spp_j = focal, treatment, alpha_jj = alpha)

  ## Get eta_j
  Competitor_demo = pairwiseValues %>%
    filter(   focal== competitor  ) %>%
    select(  spp_j = focal, treatment, eta_j = ni)


  Ratios = pairwiseValues %>%
    filter(   focal != competitor  ) %>%
    rename( eta_i = ni, spp_i= focal, spp_j=competitor, alpha_ij = alpha) %>%
    left_join(IntraAlpha, by = c("spp_j", "treatment")) %>%
    left_join(Competitor_demo, by = c("spp_j", "treatment")) %>%
    select(-(lambda:s), -(fd:coexist) ) %>%
    mutate( LR_demo= log10((eta_i   -1)/ ( eta_j-1)),
            LR_intc = log10(alpha_jj / alpha_ij))
  return(Ratios)
}

List_Null_Fits%>%
  map_df(ChangeInLogRatio, .id = 'rep') %>%
  group_by(rep, sp_pair)%>%
  summarise(alpha_ij_shift = abs(diff(alpha_ij)),
            alpha_jj_shift = abs(diff(alpha_jj)),
            Demo_shift = abs(diff(LR_demo)),
            Intc_shift = abs(diff(LR_intc))) -> Null_Shifts
```

```
## `summarise()` has grouped output by 'rep'. You can override using the `.groups`
## argument.
```

6.0.1 Shifts in the real data

```
d %>%
  mutate(SIM_DATA = log(num_seeds))%>%
  Calc_all_fit_sim %>%
  ChangeInLogRatio-> OrigData_RatioResults

OrigData_RatioResults %>%
  group_by(sp_pair)%>%
  summarise(alpha_ij_shift = abs(diff(alpha_ij)),
            alpha_jj_shift = abs(diff(alpha_jj)),
            Demo_shift = abs(diff(LR_demo)),
            Intc_shift = abs(diff(LR_intc)))%>%
  mutate(diff = Intc_shift-Demo_shift )-> OrigData_Shifts
```

6.1 Repeating t-test (with correction) of whether across the 15 pairs competition is larger than changes in demographic potential.

```
### Calculate mean shifts for each pair across the null:
Null_Shifts %>%
  group_by(sp_pair) %>%
  summarise(Null_Mean_Demo_Shift = mean(Demo_shift),
            Null_Mean_Intc_Shift = mean(Intc_shift)) -> Null_Pair_means

## Calculate 'excess' change by subtracting off the mean seen in the null model
OriginalData <- read_csv('spp_comp_combos.csv') ## as generated by original paper `n_alpha_ratios.R`
```

```
## Rows: 15 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): species, competitor, larger, sp_pair
## dbl (2): n_change, a_change
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
OriginalData %>%
  filter(sp_pair %in% total_pairs) %>%
  left_join(Null_Pair_means, by = "sp_pair") %>%
  mutate( Demo_NullExcess =   n_change  - Null_Mean_Demo_Shift,
          Intc_NullExcess = a_change  - Null_Mean_Intc_Shift,
          Intc_Greater = Intc_NullExcess>Demo_NullExcess) %>%
  as.data.frame() -> WhichShiftGeater_df

## Original t-test
t.test(WhichShiftGeater_df$n_change ,
       WhichShiftGeater_df$a_change , paired = TRUE)
```

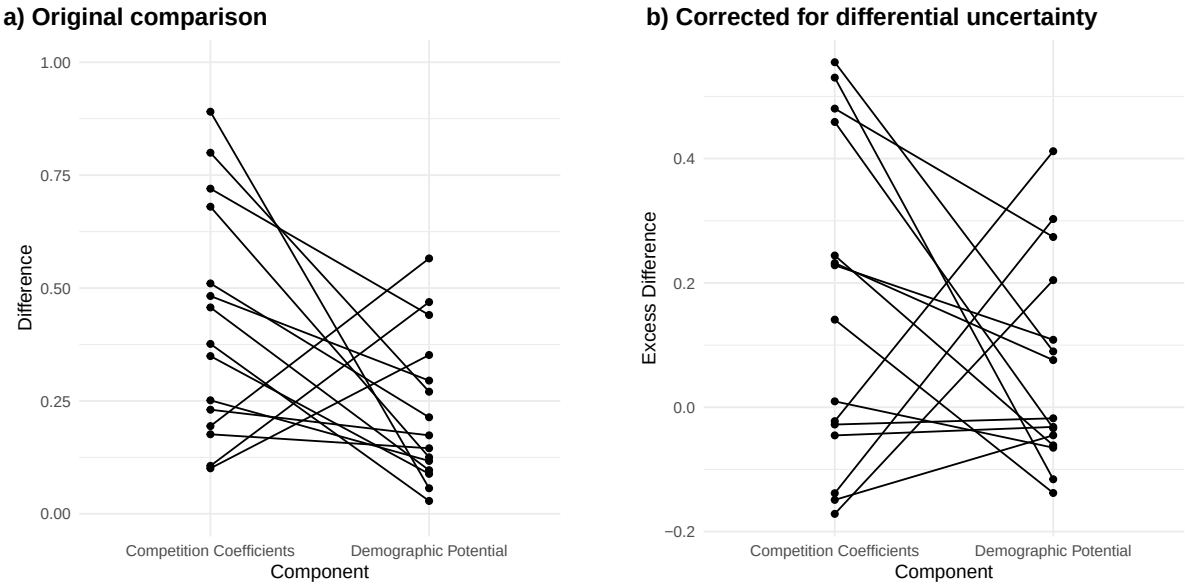
```
##
##  Paired t-test
##
## data:  WhichShiftGeater_df$n_change and WhichShiftGeater_df$a_change
## t = -2.204, df = 14, p-value = 0.04476
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -0.379794917 -0.005171194
## sample estimates:
## mean difference
##      -0.1924831
```

```
## t-test correcting for extra variation:
t.test(WhichShiftGeater_df$Demo_NullExcess,
       WhichShiftGeater_df$Intc_NullExcess, paired = TRUE )
```

```
##
##  Paired t-test
##
## data:  WhichShiftGeater_df$Demo_NullExcess and WhichShiftGeater_df$Intc_NullExcess
## t = -1.0616, df = 14, p-value = 0.3064
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -0.2750508  0.0929164
## sample estimates:
## mean difference
##     -0.09106722
```

```
WhichShiftGeater_df %>%
  select( sp_pair,
          `Demographic Potential` =n_change,
          `Competition Coefficients`=a_change) %>%
  pivot_longer(-sp_pair,
               names_to = 'Component',
               values_to = 'Difference')%>%
  ggplot( aes( y= `Difference`, x = Component))+
  # geom_boxplot(fill = 'grey')+
  geom_line(aes(group=sp_pair))+
  geom_point()+
  coord_cartesian(ylim = c(0,1))+
  theme_minimal()-> Original_ApproachPlot

WhichShiftGeater_df %>%
  select( sp_pair,
          `Demographic Potential` =Demo_NullExcess,
          `Competition Coefficients`=Intc_NullExcess) %>%
  pivot_longer(-sp_pair,
               names_to = 'Component',
               values_to = 'Excess Difference')%>%
  ggplot( aes( y= `Excess Difference`, x = Component))+
  # geom_boxplot(fill = 'grey')+
  geom_line(aes(group=sp_pair))+
  geom_point()+
```

```
  theme_minimal()-> CorrectedApproachPlot
plot_grid(Original_ApproachPlot, CorrectedApproachPlot,
          labels = c('a) Original comparison',
                     'b) Corrected for differential uncertainty' ),
          hjust = -0.1, scale= 0.9)
```



7 Using Null Model as a reference distribution

A more direct approach to testing the statistical significance of an observation compared to a null model is to examine the where on the distribution of the null model draws the data lies. This approach was developed by Van Dyke et al. in a draft reply of theirs to an earlier draft of this Matters Arising, but as I detail below included certain analysis issues. Where possible, the code in this section follows that included in the Supplementary Materials associated with the second draft of Van Dyke et al's reply to the Matters Arising (May 2023).

7.1 Generating a larger set of null model draws

While 1000 draws was enough for estimating a mean for the subtraction approach, for this approach a larger null model draw is necessary (see section below). This requires a larger set of draws from the Bayesian model, but otherwise all steps are the same as above.

```
model1_ACWR<-brm( bf(Log_Seeds~log((10^LOG10lambda)/(1+
                                                     aACWR*N_acwr+
                                                     aFEMI*N_femi+
                                                     aHOMU*N_homu+
                                                     aPLER*N_pler+
                                                     aSACO*N_saco+
                                                     aURLI*N_urli)),
                     LOG10lambda+aACWR+aFEMI+aHOMU+aPLER+aSACO+aURLI~1,
                     nl = TRUE),
```

```
                  iter = 21000,
                  data= filter(d, focal == 'ACWR'),
                  prior = loose_priors_combined)

### Refit with different data for each different species.
model1_FEMI<- update(model1_ACWR,newdata = filter(d, focal == 'FEMI') )
model1_HOMU<- update(model1_ACWR,newdata = filter(d, focal == 'HOMU') )
model1_PLER<- update(model1_ACWR,newdata = filter(d, focal == 'PLER') )
model1_SACO<- update(model1_ACWR,newdata = filter(d, focal == 'SACO') )
model1_URLI<- update(model1_ACWR,newdata = filter(d, focal == 'URLI') )

joint_model_list <- list(model1_ACWR,model1_FEMI,model1_HOMU,
                         model1_PLER,model1_SACO,model1_URLI)
save(joint_model_list,file =  'joint_model_list2_10000')

## Taking posteror draws
set.seed(1)
load(file = 'joint_model_list2_10000')
## Draw from posterior predictive distribution (includes model uncertainty and residual error)
predict_ACWR<- posterior_predict(joint_model_list[[1]], ndraws = 10500)
predict_FEMI<- posterior_predict(joint_model_list[[2]], ndraws = 10500)
predict_HOMU<- posterior_predict(joint_model_list[[3]], ndraws = 10500)
predict_PLER<- posterior_predict(joint_model_list[[4]], ndraws = 10500)
predict_SACO<- posterior_predict(joint_model_list[[5]], ndraws = 10500)
predict_URLI<- posterior_predict(joint_model_list[[6]], ndraws = 10500)
## Taking draws from generative model
1:10500 %>%
  map(Gen_simData_fromfit)%>%
  map(Calc_all_fit_sim) -> List_Null_Fits

List_Null_Fits<- List_Null_Fits[List_Null_Fits != "0"]
#remove ones that did not converge and take 10k
List_Null_Fits10k<- List_Null_Fits[c(1:10000)]
save(List_Null_Fits10k, file='List_Null_Fits10k')
```

7.2 Comparing data to null distribution

7.2.1 Calculating key quantitites for null model set

```
load(file='List_Null_Fits10k')

List_Null_Fits10k%>%
  map_df(ChangeInLogRatio, .id = 'rep') %>%
  group_by(rep, sp_pair)%>%
  summarise(alpha_ij_shift = abs(diff(alpha_ij)), ## calculate difference betwee treatments
            alpha_jj_shift = abs(diff(alpha_jj)),
            Demo_shift = abs(diff(LR_demo)),
            Intc_shift = abs(diff(LR_intc))) -> Null_Shifts10k
```

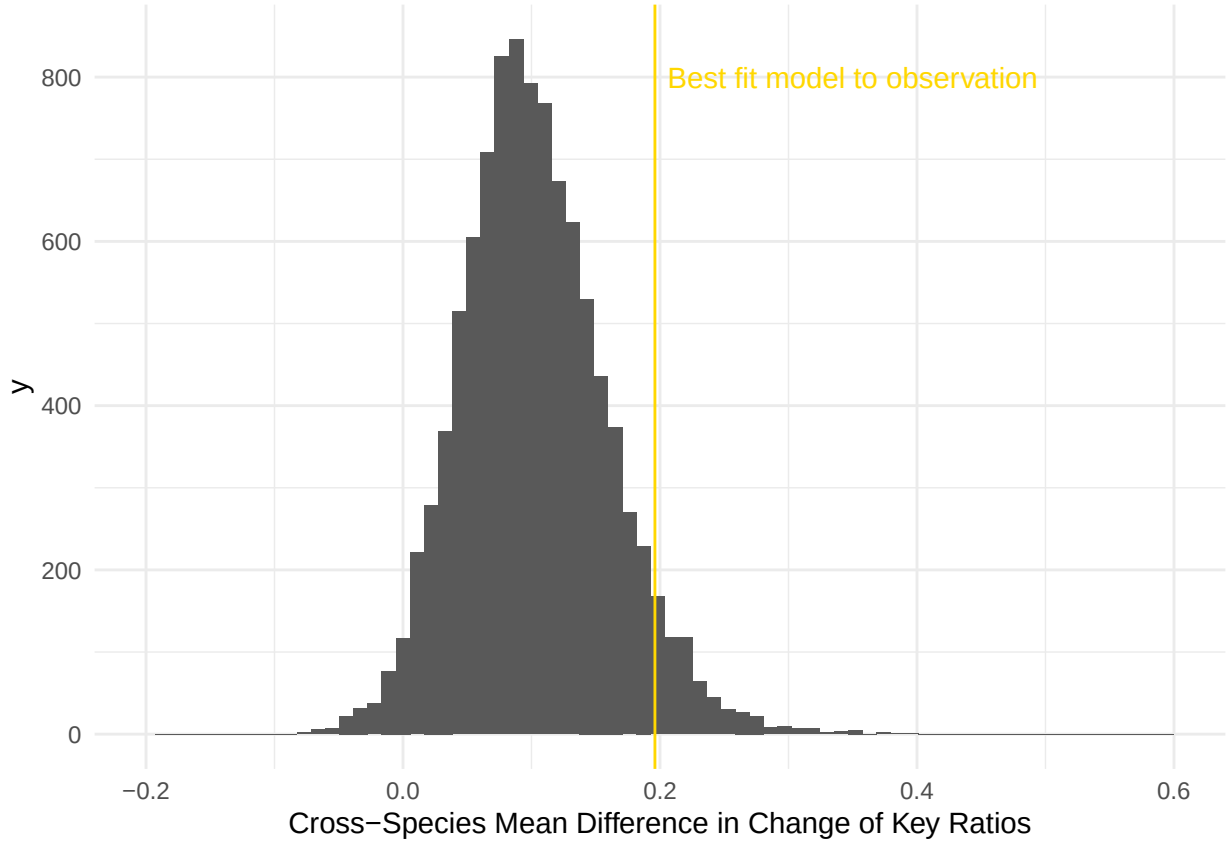
```
## `summarise()` has grouped output by 'rep'. You can override using the `.groups`
## argument.
```

```
Null_Shifts10k %>%
  group_by(sp_pair) %>%
  summarise(Null_Mean_Demo_Shift = mean(Demo_shift),
            Null_Mean_Intc_Shift = mean(Intc_shift)) -> Null10k_SpMeans

Null_Shifts10k %>%
  filter(sp_pair %in% total_pairs) %>%
  group_by(rep) %>%
  summarise(mean_a_change = mean(Intc_shift),
            mean_n_change = mean(Demo_shift),
            diff = mean(Intc_shift-Demo_shift)) -> Null10k_CrossSpeciesMeans
```

7.2.2 Comparing best-fit values to to null distribution to calculate p-value

```
OrigData_Shifts %>%
  filter(sp_pair %in% total_pairs) %>%
  mutate( DiffShifts =   Intc_shift-Demo_shift)  %>%
  summarise(MeanDemo_shift    = mean(Demo_shift),
            MeanIntc_shift    = mean(Intc_shift),
            MeanDiffShift = mean(DiffShifts)) -> BestFitDifferences

## Difference
Null10k_CrossSpeciesMeans  %>%
  ggplot( aes(x=diff))+
  geom_histogram(binwidth = 0.011)+
  geom_vline(xintercept = BestFitDifferences$MeanDiffShift , col = 'gold' )+
  theme_minimal()+
  xlab('Cross-Species Mean Difference in Change of Key Ratios')+
  annotate("text", label = "Best fit model to observation",
           x = 0.35, y = 800, col = 'gold', )+
  scale_x_continuous(limits = c(-0.2, 0.6))
```

```
## Warning: Removed 2 rows containing missing values (`geom_bar()`).
```




```
### Data Value
 BestFitDifferences$MeanDiffShift
```

```
## [1] 0.1960542
```

```
### p-value (two-tailed test of different from null)
sum(Null10k_CrossSpeciesMeans$diff >  BestFitDifferences$MeanDiffShift)*2/nrow(Null10k_CrossSpeciesMeans
```

```
## [1] 0.1166
```

7.2.3 Comparing whole bootstrap to null distribution

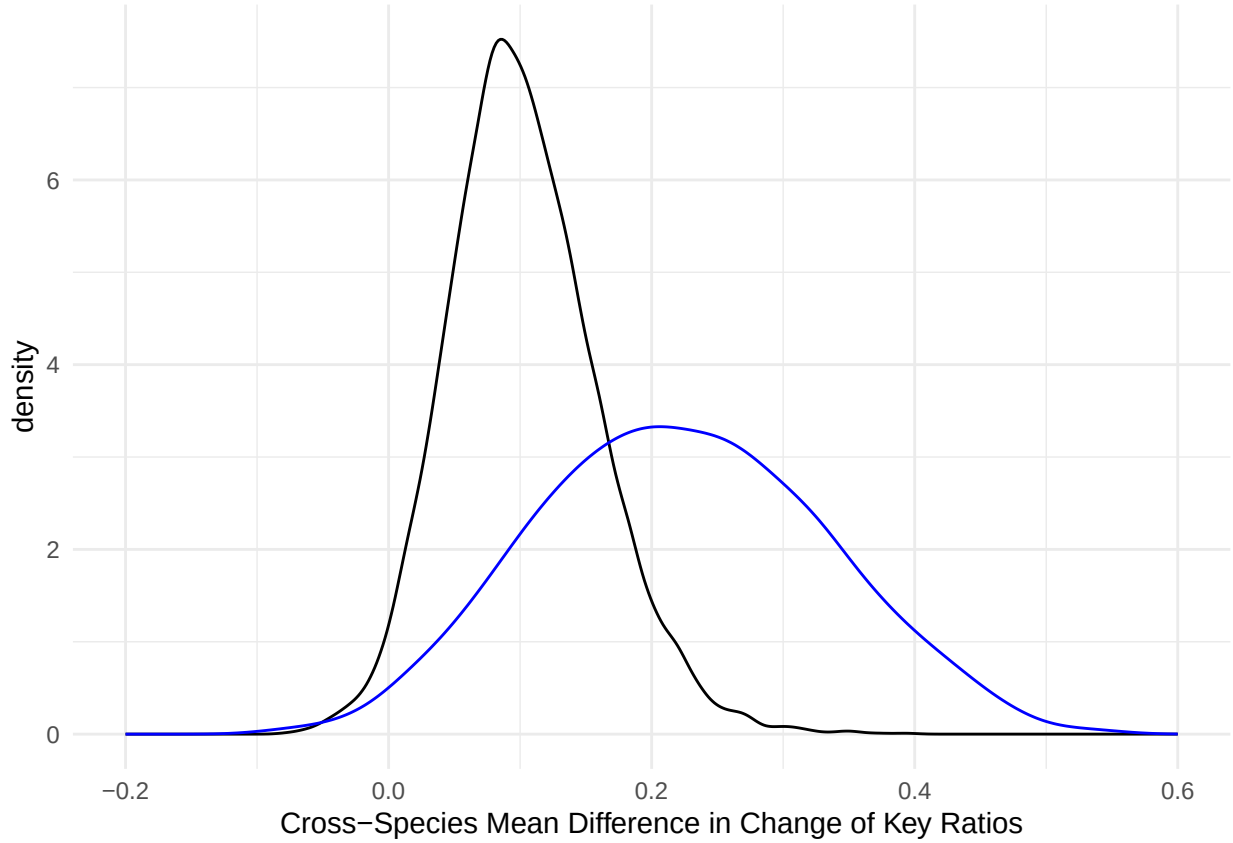
Repeating the analysis using samples from the bootstrap gives a sense of the bounds of the confidence that the final p-value can be specified given the uncertainty in the data.

```
igr_change_2 <- function(foc, comp,df_temp) {
  nj_d <- with(df_temp, ni[focal == comp & treatment == 2])[1]
  ni_d <- with(df_temp, ni[focal == foc & treatment == 2])[1]
  ajj_d <- with(df_temp, alpha[focal == comp & competitor == comp & treatment == 2])
  aij_d <- with(df_temp, alpha[focal == foc & competitor == comp & treatment == 2])
  n_ratio_d = log10((ni_d-1)/(nj_d-1))
  a_ratio_d = log10(ajj_d/aij_d)

  nj_w <- with(df_temp, ni[focal == comp & treatment == 1])[1]
```

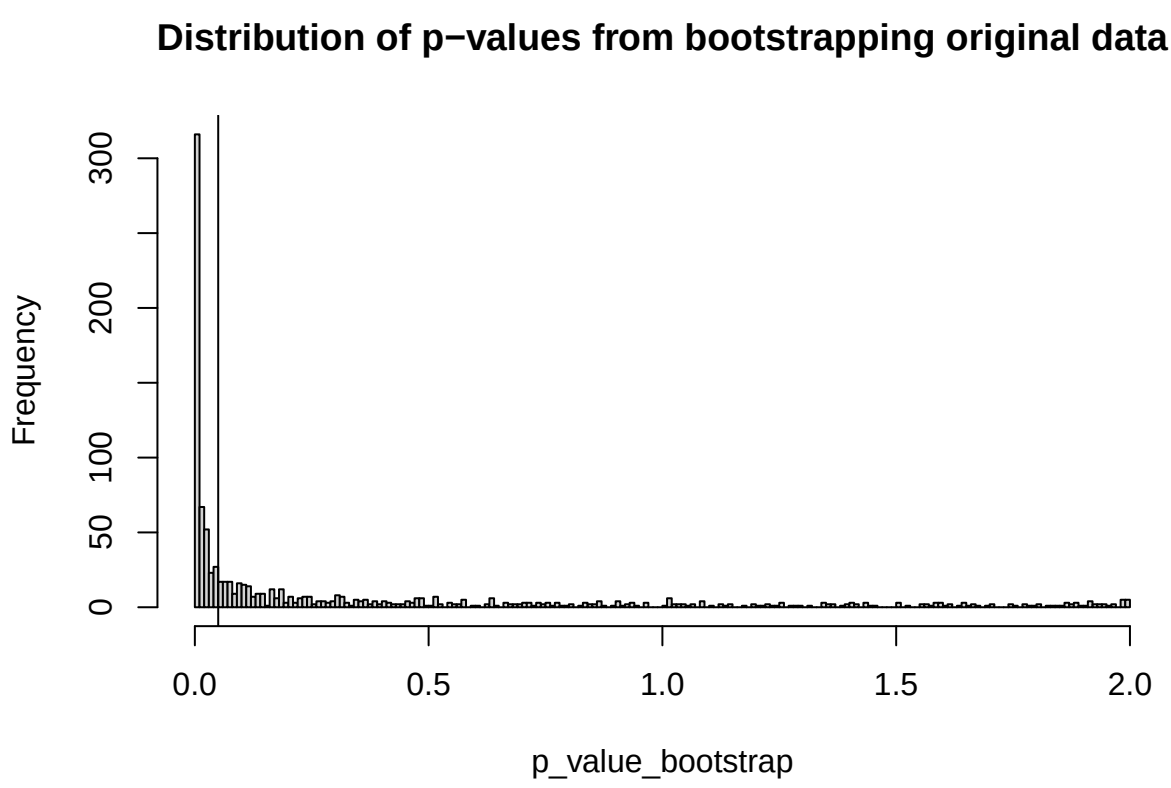
```
  ni_w <- with(df_temp, ni[focal == foc & treatment == 1])[1]
  ajj_w <- with(df_temp, alpha[focal == comp & competitor == comp & treatment == 1])
  aij_w <- with(df_temp, alpha[focal == foc & competitor == comp & treatment == 1])
  n_ratio_w = log10((ni_w-1)/(nj_w-1))
  a_ratio_w = log10(ajj_w/aij_w)

  nc<-n_ratio_w - n_ratio_d ## taking away the absolute-ing from here
  ac<-a_ratio_w - a_ratio_d
  return(c(nc, ac))
}

CalcIGR_From_Bootstrap <-  function(BootstrapId){
  df_temp = final_output_nls_boot %>%filter(id == BootstrapId)  ## pull out each bootstrap fit
  spp_list <- sort(na.omit( unique(seed_data$focal)))
  comp_labels <- sort( na.omit( unique(seed_data$background) ))
  spp_comp_combos2 <- expand.grid(species = spp_list,
                                  competitor = comp_labels) # make dataframe of species pairs
  spp_comp_combos2$n_change <- 0
  spp_comp_combos2$a_change <- 0
  spp_comp_combos2$diff <- 0
  for(i in 1:nrow(spp_comp_combos2)){
    ii <- spp_comp_combos2$species[i]
    jj <- spp_comp_combos2$competitor[i]
    IGR_change_both <- igr_change_2(ii, jj,df_temp)
    spp_comp_combos2$n_change[i] <- IGR_change_both[1]
    spp_comp_combos2$a_change[i] <- IGR_change_both[2]
  }
  spp_comp_combos2$sp_pair <- paste(spp_comp_combos2$species,
                                    spp_comp_combos2$competitor, sep = "_")
  spp_comp_combos2$BootstrapId = BootstrapId
  return(spp_comp_combos2)
}

final_output_nls_boot<-read.csv(paste0(Path_to_Orig,"output/final_output_nls_boot_1000.csv"))

map_df(unique( final_output_nls_boot$id),
       CalcIGR_From_Bootstrap ) %>%
  as_tibble() %>%
  filter(sp_pair %in% total_pairs) %>%  ## Just do each competitive pair once, but NB not symmetric
  mutate(diff= abs(a_change)- abs(n_change))-> Species_level_bootstrapIGR

### Bootstrap of cross-species averages
Species_level_bootstrapIGR %>%
  group_by(BootstrapId) %>%
  summarise(CrossSpecies_diff_direct=mean(diff)) -> BootStrap_CrossSpecies
```

```
## Difference
Null10k_CrossSpeciesMeans %>%
  ggplot()+
  geom_density( aes(x=diff))+
  geom_density( aes(x = CrossSpecies_diff_direct),
                col = 'blue', data =BootStrap_CrossSpecies )+
  theme_minimal()+
```

```
  xlab('Cross-Species Mean Difference in Change of Key Ratios')+
  scale_x_continuous(limits = c(-0.2, 0.6))
```




```
p_value_bootstrap <- rep(NA, 1000)

for( bs in 1:1000){
### Assuming a 2-sided test, need less than 0.025 of the null to be greater
### than data to identify significant difference in direction described
  p_value_bootstrap[bs] <- sum(Null10k_CrossSpeciesMeans$diff > BootStrap_CrossSpecies$CrossSpecies_dif
}

hist(p_value_bootstrap, breaks =200,
     main = 'Distribution of p-values from bootstrapping original data')
## NB p-values above 1 suggest support reverse hypothesis, that lambda difference was significantly lar
abline( v = 0.05)
```




```
quantile(p_value_bootstrap, probs = seq(from=0, to = 1, by = 0.1))
```

```
##      0%     10%     20%     30%     40%     50%     60%     70%     80%     90%
## 0.00000 0.00060 0.00320 0.00868 0.02260 0.05810 0.13320 0.30350 0.63608 1.25426
##    100%
## 2.00000
```

```
#  ^^^ P-value is very uncertain to resampling of the data,
```

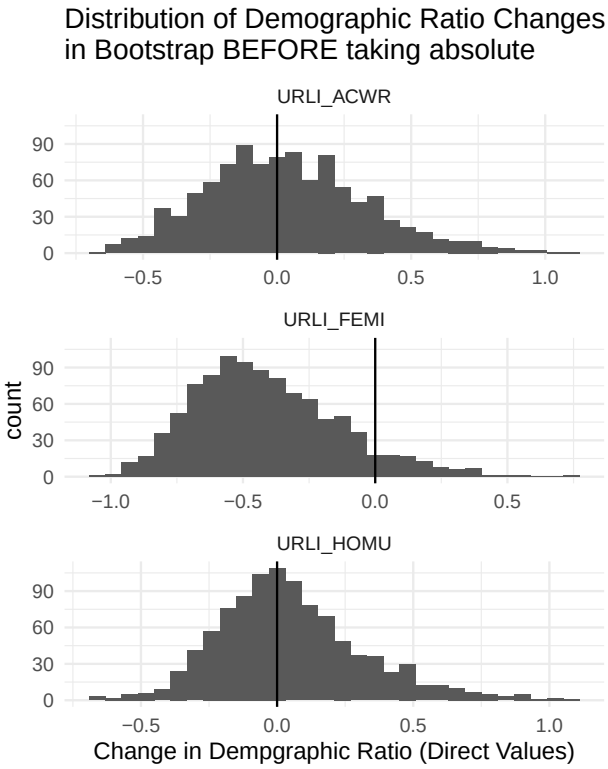
7.2.4 Challenges with averages

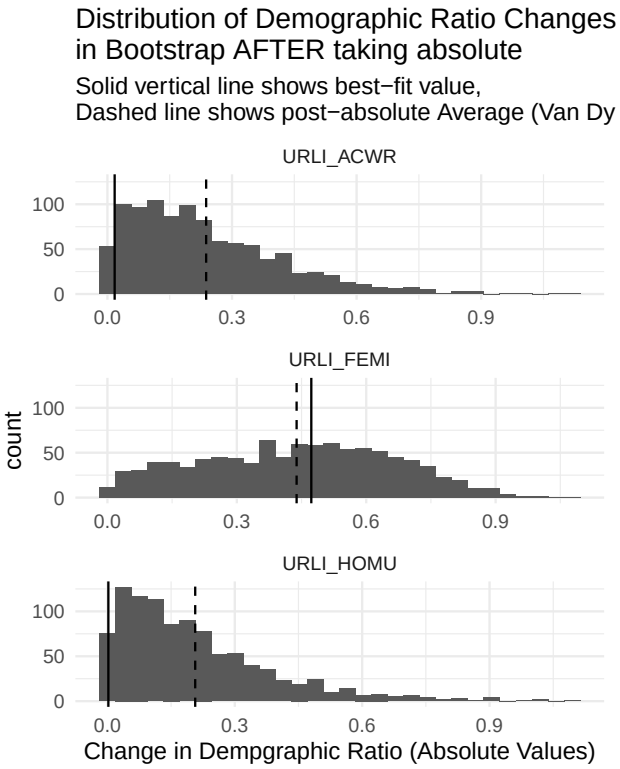
In their draft reply (dated May 2023) Van Dyke et al. present an analysis similar to the above, but taking a mean of the differences reported in the bootstrap sample. However, rather than taking an average of the final result from each bootstrap, they average of the absolute change of each of the demographic and competition ratio responses. This average is then used to calculate the difference in these quantities. I illustrate in this subsection how because the direction of ratio responses is uncertain, this gives a quite distinct result to using best-fit value.

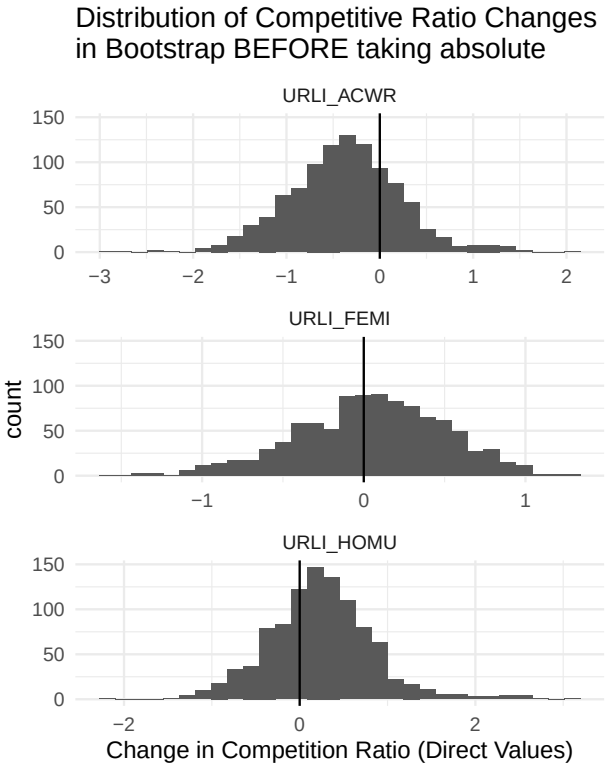
```
Species_level_bootstrapIGR %>%
  group_by(sp_pair) %>%
  summarise(BS_Mean_demo_change = mean(n_change ), ## average change of the pair
            BS_Mean_comp_change = mean(a_change),  ## average change of the pair
            BS_Mean_ABSdemo_change = mean(abs(n_change)), ## VD Approach (average absolute change)
            BS_Mean_ABScomp_change = mean(abs(a_change)),  ## VD Approach (average absolute change)
```

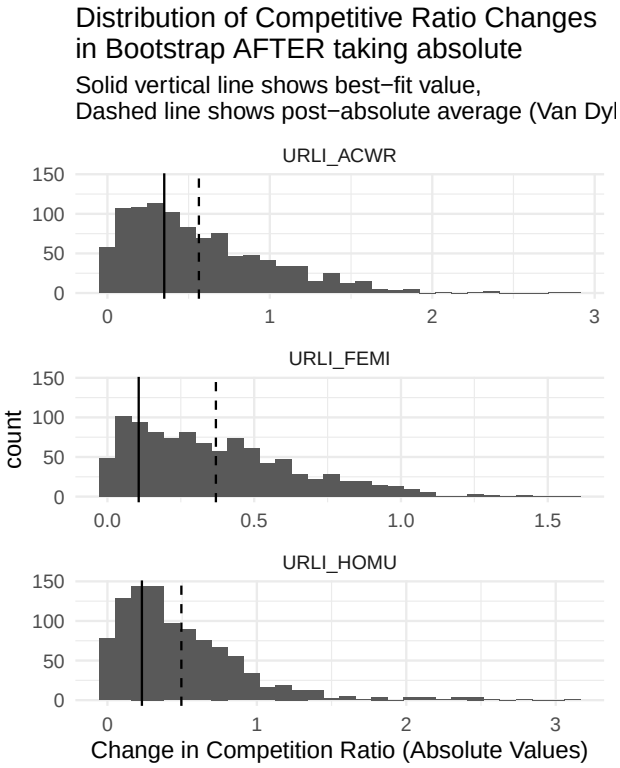
```
            BS_Mean_Diff_abs = mean(abs(a_change))-mean(abs(n_change)), ## VD Approach
            BS_Mean_diff     =   abs(mean(a_change))- abs(mean(n_change))) %>% ### avoiding averaging i
  ungroup()-> MeanBootstrap_SpLevel_Ratios

## Just taking three particularly problematic  pairs
Pairs_to_plot = c("URLI_HOMU", "URLI_FEMI","URLI_ACWR" )

Species_level_bootstrapIGR %>%
  filter( sp_pair %in% Pairs_to_plot)%>%
  ggplot( aes( x = n_change))+
  geom_histogram()+
  geom_vline(xintercept = 0)+
  facet_wrap(~sp_pair, scales = 'free_x', nrow = 3)+
  ggtitle('Distribution of Demographic Ratio Changes\nin Bootstrap BEFORE taking absolute')+
  theme_minimal()+
  xlab('Change in Dempgraphic Ratio (Direct Values)') -> ProblemID1

Species_level_bootstrapIGR %>%
  filter( sp_pair %in% Pairs_to_plot)%>%
  ggplot( aes( x = abs(n_change)))+
  geom_histogram()+
  geom_vline(data = filter( MeanBootstrap_SpLevel_Ratios, sp_pair %in% Pairs_to_plot),
             aes(xintercept = BS_Mean_ABSdemo_change ),
             linetype = 'dashed')+
  geom_vline(data =filter( OrigData_Shifts, sp_pair %in% Pairs_to_plot),
             aes(xintercept = Demo_shift  ))+
  facet_wrap(~sp_pair, scales = 'free_x', nrow = 3)+
  ggtitle('Distribution of Demographic Ratio Changes\nin Bootstrap AFTER taking absolute',
          'Solid vertical line shows best-fit value, \nDashed line shows post-absolute Average (Van Dyk
  theme_minimal()+
  xlab('Change in Dempgraphic Ratio (Absolute Values)') -> ProblemID2


### Repeating with Competition Change

Species_level_bootstrapIGR %>%
  filter( sp_pair %in% Pairs_to_plot)%>%
  ggplot( aes( x = a_change    ))+
  geom_histogram()+
  geom_vline(xintercept=0)+
  facet_wrap(~sp_pair, scales = 'free_x', nrow = 3)+
  ggtitle('Distribution of Competitive Ratio Changes\nin Bootstrap BEFORE taking absolute')+
  theme_minimal()+
  xlab('Change in Competition Ratio (Direct Values)') -> ProblemID3

Species_level_bootstrapIGR %>%
  filter( sp_pair %in% Pairs_to_plot)%>%
  ggplot( aes( x = abs(a_change    )))+
  geom_histogram()+
  geom_vline(data =  filter( MeanBootstrap_SpLevel_Ratios, sp_pair %in% Pairs_to_plot),
             aes(xintercept = BS_Mean_ABScomp_change  ),
             linetype = 'dashed')+
  geom_vline(data =filter( OrigData_Shifts, sp_pair %in% Pairs_to_plot),
```

```
            aes(xintercept = Intc_shift  ))+
  facet_wrap(~sp_pair, scales = 'free_x', nrow = 3)+
  ggtitle('Distribution of Competitive Ratio Changes\nin Bootstrap AFTER taking absolute',
          'Solid vertical line shows best-fit value, \nDashed line shows post-absolute average (Van Dyk
  theme_minimal()+
  xlab('Change in Competition Ratio (Absolute Values)') -> ProblemID4

plot_grid(ProblemID1, ProblemID2, ProblemID3, ProblemID4)
```









For reference, the key functions used in Van Dyke et al's draft reply are below:

```
### Van Dyke Method: (with issues highlights)
# igr_change_VD <- function(foc, comp) {
```

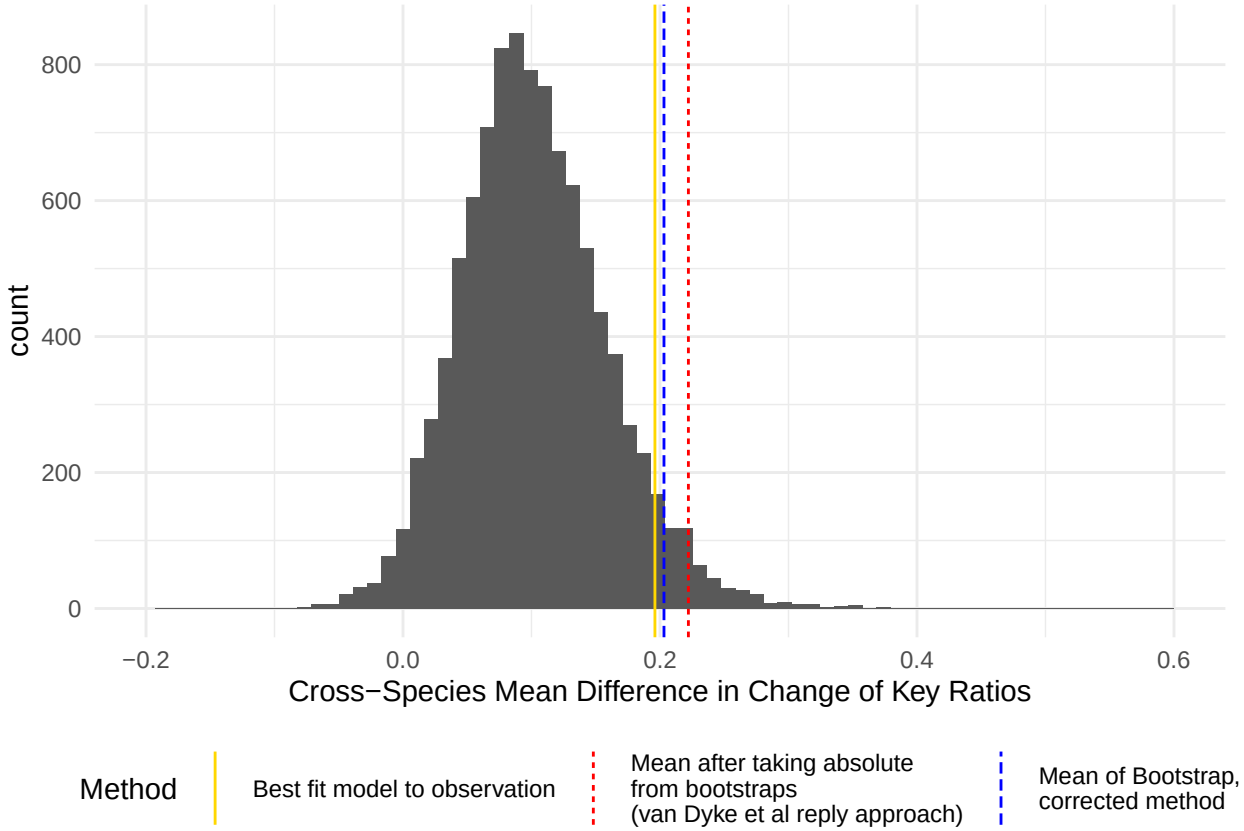
```
#   nj_d <- with(df_temp, ni[focal == comp & treatment == 2])[1]
#   ni_d <- with(df_temp, ni[focal == foc & treatment == 2])[1]
#   ajj_d <- with(df_temp, alpha[focal == comp & competitor == comp & treatment == 2])
#   aij_d <- with(df_temp, alpha[focal == foc & competitor == comp & treatment == 2])
#   n_ratio_d = log10((ni_d-1)/(nj_d-1))
#   a_ratio_d = log10(ajj_d/aij_d)
#   nj_w <- with(df_temp, ni[focal == comp & treatment == 1])[1]
#   ni_w <- with(df_temp, ni[focal == foc & treatment == 1])[1]
#   ajj_w <- with(df_temp, alpha[focal == comp & competitor == comp & treatment == 1])
#   aij_w <- with(df_temp, alpha[focal == foc & competitor == comp & treatment == 1])
#   n_ratio_w = log10((ni_w-1)/(nj_w-1))
#   a_ratio_w = log10(ajj_w/aij_w)
#   nc<-abs(n_ratio_w - n_ratio_d)                    ### <<<<Taking absolute of differences here
#   ac<-abs(a_ratio_w - a_ratio_d)                    ### <<<<Taking absolute of differences here
#   return(c(nc, ac))
# }

# for(aa in 1:nrow(n_a_change_df_real)){
#   boot_id <- n_a_change_df_real[aa, "boot"]
#   df_temp <- final_output_nls_boot %>% filter(id == boot_id)
#   #Create data frame
#   spp_list <- sort(na.omit( unique(seed_data$focal)))
#   comp_labels <- sort( na.omit( unique(seed_data$background) ))
#   spp_comp_combos2 <- expand.grid(species = spp_list, competitor = comp_labels)
#   spp_comp_combos2$n_change <- 0
#   spp_comp_combos2$a_change <- 0
#   spp_comp_combos2$diff <- 0
#   for(i in 1:nrow(spp_comp_combos2)){
#     ii <- spp_comp_combos2[i, "species"] %>% unlist
#     jj <- spp_comp_combos2[i, "competitor"] %>% unlist
#     spp_comp_combos2[i, "n_change"] <- igr_change_VD(ii, jj)[1]
#     spp_comp_combos2[i, "a_change"] <- igr_change_VD(ii, jj)[2]
#     spp_comp_combos2[i, "diff"] <- igr_change_VD(ii, jj)[2]-igr_change_VD(ii, jj)[1]
#   }
#   spp_comp_combos2$sp_pair <- paste(spp_comp_combos2$species,
#                                     spp_comp_combos2$competitor, sep = "_")
#   spp_comp_combos2 <-spp_comp_combos2 %>% filter(sp_pair %in% total_pairs)

#   n_a_change_df_real[aa, "mean_n_change"] <- mean(spp_comp_combos2$n_change)
#   n_a_change_df_real[aa, "mean_a_change"] <- mean(spp_comp_combos2$a_change)
###                       ^^^^^^^^^^^^^^^^^Taking average of differences here

#   n_a_change_df_real[aa, "diff"] <- mean(spp_comp_combos2$diff)
#   n_a_change_df_real[aa, "pval"] <- t.test(spp_comp_combos2$n_change,
#                                            spp_comp_combos2$a_change, paired = T)[3]
# }
```

7.2.5 Impact of averaging on final results

This effect of averaging over-uncertain directions leads to an inflated mean difference in the key quantity (the cross-species average difference in the key ratios).

```r
### Taking cross species means:
MeanBootstrap_SpLevel_Ratios %>%
  summarise(Cross_Species_Diff_abs=
              mean(BS_Mean_Diff_abs), ## VD Approach
            Cross_Species_Diff_direct =
              mean(BS_Mean_diff))%>%  ## Updated Approach
  as.list->  CrossSpeciesMeans

DataValues <- data.frame(Method = c("Mean after taking absolute\nfrom bootstraps\n(van Dyke et al reply
                                    "Mean of Bootstrap,\ncorrected method",
                                    "Best fit model to observation"),
                         Value = c( CrossSpeciesMeans$Cross_Species_Diff_abs,
                                    CrossSpeciesMeans$Cross_Species_Diff_direct,
                                    BestFitDifferences$MeanDiffShift ))


Null10k_CrossSpeciesMeans %>%
  ggplot( aes(x=diff))+
  geom_histogram(binwidth = 0.011)+
  geom_vline(aes(col = Method,
                 linetype =Method ,
                 xintercept = Value),
             data = DataValues)+
  scale_colour_manual(values =c('gold', 'red', 'blue'))+
  theme_minimal()+
  theme(legend.position = 'bottom')+
  xlab('Cross-Species Mean Difference in Change of Key Ratios')+
  scale_x_continuous(limits = c(-0.2, 0.6))
```

```
## Warning: Removed 2 rows containing missing values (‘geom_bar()‘).
```




```
### Calculating p-values
DataValues$pvalue = NA
for(i in 1:3){
  DataValues$pvalue[i] <- sum(Null10k_CrossSpeciesMeans$diff > DataValues$Value[i])*2/10000
}

knitr::kable(DataValues, format = 'latex')
```

Method	Value	pvalue
Mean after taking absolute from bootstraps (van Dyke et al reply approach)	0.2220924	0.0528
Mean of Bootstrap, corrected method	0.2030975	0.0954
Best fit model to observation	0.1960542	0.1166

7.2.6 Illustration of the need to take sufficient null model draws

In their draft reply, Van Dyke et al. used 1000 null models draws and found a p-value of 0.036.

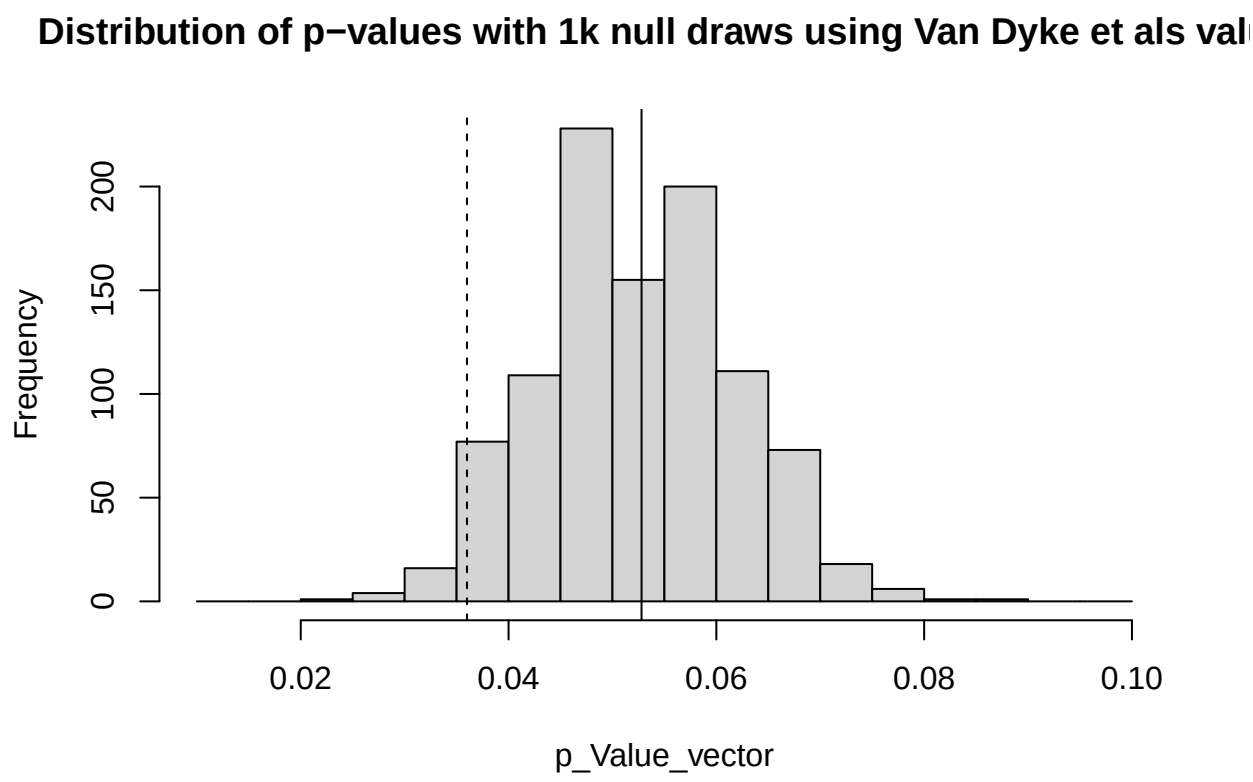
Here I show that 1000 draws is sufficient to characterise the mean, but insufficient to characterise the p-value reliably by taking subsamples of the 10k null model set.

```
## Taking 100 different draws of 1000 from the null model to assess
set.seed(1)
p_Value_vector <- rep(NA, 1000)
for( draw in 1:1000){
  SELECTION =   sample(1:10000, size = 1000, replace = FALSE)
```

```
  NULL_Selection = Null10k_CrossSpeciesMeans$diff[SELECTION]
  p_Value_vector[draw] <- ( sum(NULL_Selection> DataValues$Value[1]  )*2/1000)
}

mean(p_Value_vector<0.05)*100 ## Percent of cases where p-value was significant
```

```
## [1] 36.4
```

```
hist(p_Value_vector, breaks =seq(0.01, 0.1, by= 0.005),
     main = 'Distribution of p-values with 1k null draws using Van Dyke et als value')
abline(v = 0.036, lty=2)
abline(v = DataValues$pvalue[1])
```

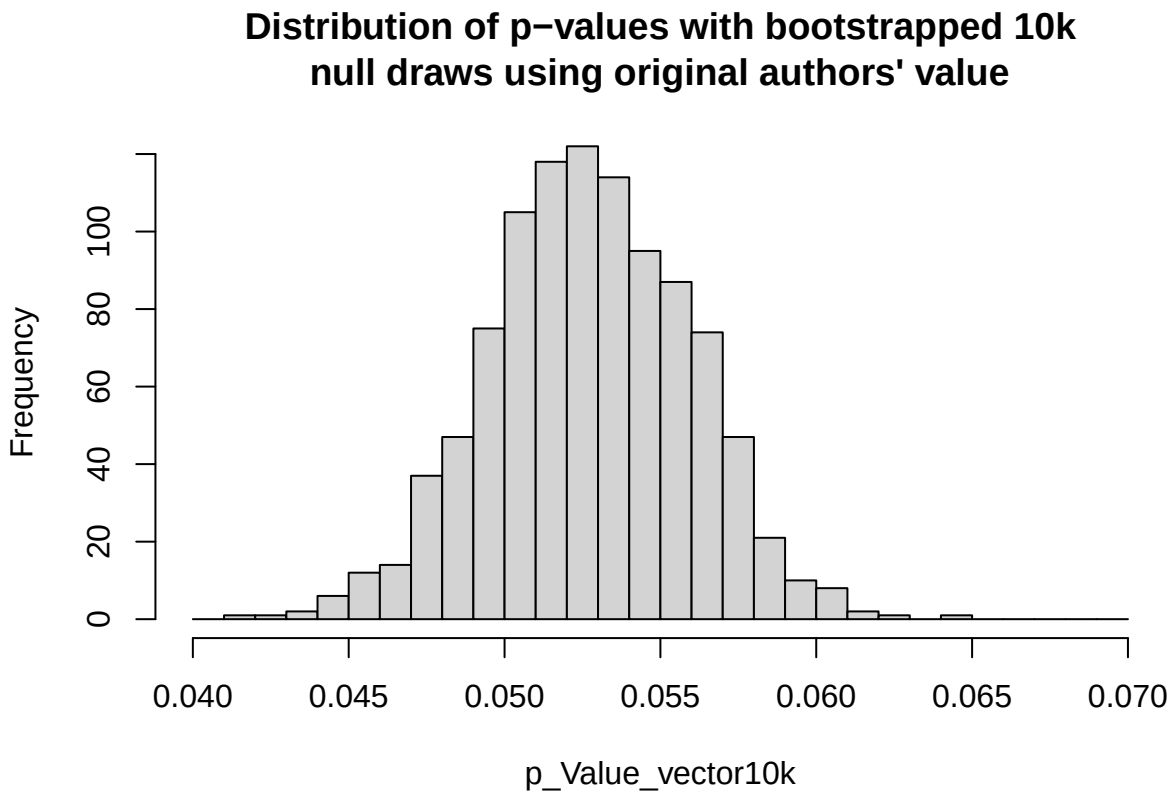



```
### 95% of
quantile2(p_Value_vector)
```

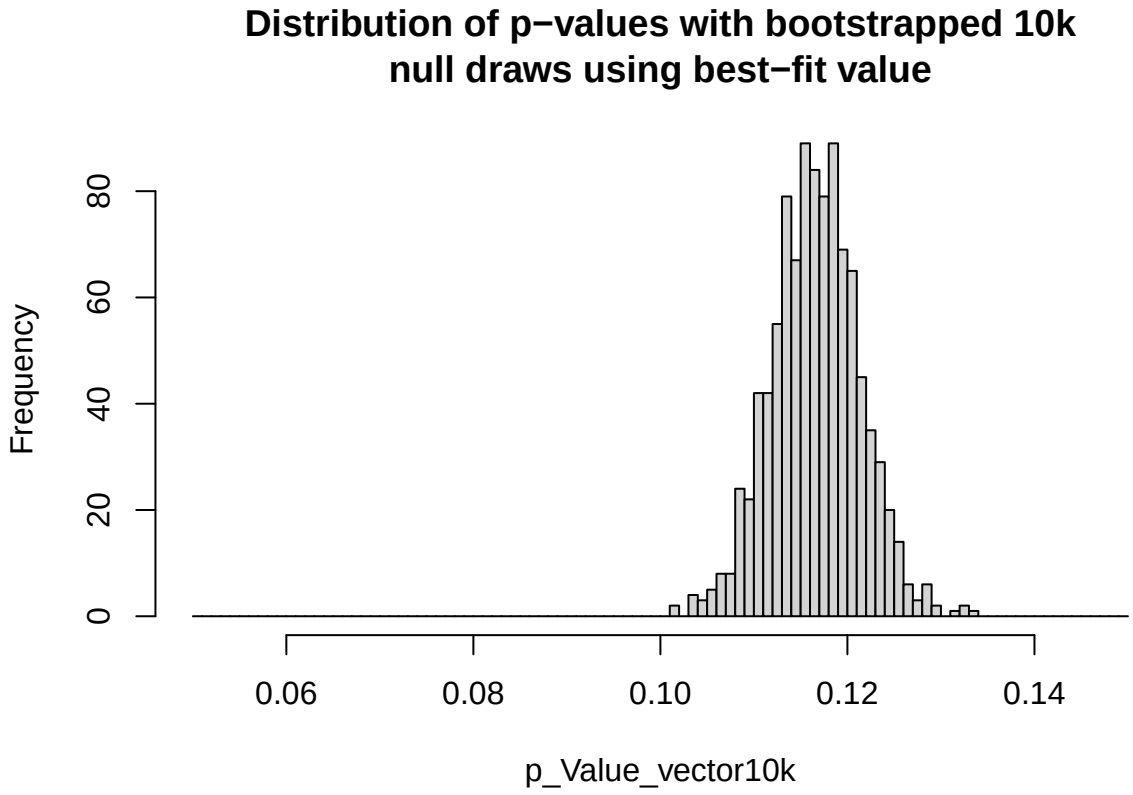
```
##    q5   q95
## 0.038 0.070
```

```
## Even 10k draws is still a little uncertain about the significance
## Note that this is using the inflated value of the data observation.
set.seed(1)
p_Value_vector10k <- rep(NA, 1000)
```

```r
for( draw in 1:1000){
  SELECTION =   sample(1:10000, size = 10000, replace = TRUE) ## resampling
  NULL_Selection = Null10k_CrossSpeciesMeans$diff[SELECTION]
  p_Value_vector10k[draw] <- ( sum(NULL_Selection> DataValues$Value[1]  )*2/10000)
}

hist(p_Value_vector10k, breaks =seq(0.04, 0.07, by= 0.001),
     main = "Distribution of p-values with bootstrapped 10k\nnull draws using original authors' value")
```




```r
## Bootstrapping the 10k null models and the best-fit data observation
## suggests strong confidence non-significant result is not dependent on sample size.

set.seed(1)
p_Value_vector10k <- rep(NA, 1000)
for( draw in 1:1000){
  SELECTION =   sample(1:10000, size = 10000, replace = TRUE) ## resampling
  NULL_Selection = Null10k_CrossSpeciesMeans$diff[SELECTION]
  p_Value_vector10k[draw] <- ( sum(NULL_Selection> DataValues$Value[3]  )*2/10000)
}

hist(p_Value_vector10k, breaks =seq(0.05, 0.15, by= 0.001),
     main = 'Distribution of p-values with bootstrapped 10k\nnull draws using best-fit value')
```



8 Session Information

```
sessionInfo()
```

```
## R version 4.3.0 (2023-04-21 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19045)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=English_United Kingdom.utf8
## [2] LC_CTYPE=English_United Kingdom.utf8
## [3] LC_MONETARY=English_United Kingdom.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United Kingdom.utf8
##
## time zone: Europe/London
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
```

```
##
## other attached packages:
##  [1] cowplot_1.1.1   broom_1.0.4     posterior_1.4.1 brms_2.19.0
##  [5] Rcpp_1.0.10     lubridate_1.9.2 forcats_1.0.0   stringr_1.5.0
##  [9] dplyr_1.1.2     purrr_1.0.1     readr_2.1.4     tidyr_1.3.0
## [13] tibble_3.2.1    ggplot2_3.4.2   tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] gridExtra_2.3        inline_0.3.19        rlang_1.1.1
##  [4] magrittr_2.0.3       matrixStats_1.0.0    compiler_4.3.0
##  [7] loo_2.6.0            callr_3.7.3          vctrs_0.6.2
## [10] reshape2_1.4.4       pkgconfig_2.0.3      crayon_1.5.2
## [13] fastmap_1.1.1        backports_1.4.1      ellipsis_0.3.2
## [16] labeling_0.4.2       utf8_1.2.3           threejs_0.3.3
## [19] promises_1.2.0.1     rmarkdown_2.22       markdown_1.7
## [22] tzdb_0.4.0           ps_1.7.5             bit_4.0.5
## [25] xfun_0.39            jsonlite_1.8.5       highr_0.10
## [28] later_1.3.1          parallel_4.3.0       prettyunits_1.1.1
## [31] R6_2.5.1             dygraphs_1.1.1.6     stringi_1.7.12
## [34] StanHeaders_2.26.26  rstan_2.26.22        knitr_1.43
## [37] zoo_1.8-12           base64enc_0.1-3      bayesplot_1.10.0
## [40] httpuv_1.6.11        Matrix_1.5-4         igraph_1.4.3
## [43] timechange_0.2.0     tidyselect_1.2.0     rstudioapi_0.14
## [46] abind_1.4-5          yaml_2.3.7           codetools_0.2-19
## [49] miniUI_0.1.1.1       curl_5.0.1           processx_3.8.1
## [52] pkgbuild_1.4.0       lattice_0.21-8       plyr_1.8.8
## [55] shiny_1.7.4          withr_2.5.0          bridgesampling_1.1-2
## [58] coda_0.19-4          evaluate_0.21        RcppParallel_5.1.7
## [61] xts_0.13.1           pillar_1.9.0         tensorA_0.36.2
## [64] checkmate_2.2.0      DT_0.28              stats4_4.3.0
## [67] shinyjs_2.1.0        distributional_0.3.2 generics_0.1.3
## [70] vroom_1.6.3          hms_1.1.3            rstantools_2.3.1
## [73] munsell_0.5.0        scales_1.2.1         gtools_3.9.4
## [76] xtable_1.8-4         glue_1.6.2           tools_4.3.0
## [79] shinystan_2.6.0      colourpicker_1.2.0   mvtnorm_1.2-1
## [82] grid_4.3.0           crosstalk_1.2.0      colorspace_2.1-0
## [85] nlme_3.1-162         cli_3.6.1            fansi_1.0.4
## [88] Brobdingnag_1.2-9    V8_4.3.0             gtable_0.3.3
## [91] digest_0.6.31        htmlwidgets_1.6.2    farver_2.1.1
## [94] htmltools_0.5.5      lifecycle_1.0.3      mime_0.12
## [97] bit64_4.0.5          shinythemes_1.2.0
```