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.

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
  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        altern~1
##      <dbl>     <dbl>   <dbl>     <dbl>    <dbl>     <dbl> <chr>         <chr>
## 1   -0.196     -2.16  0.0488        14   -0.391  -0.00116 Paired t-test two.sid~
## # ... with abbreviated variable name 1: alternative
```

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))) -> 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)
```

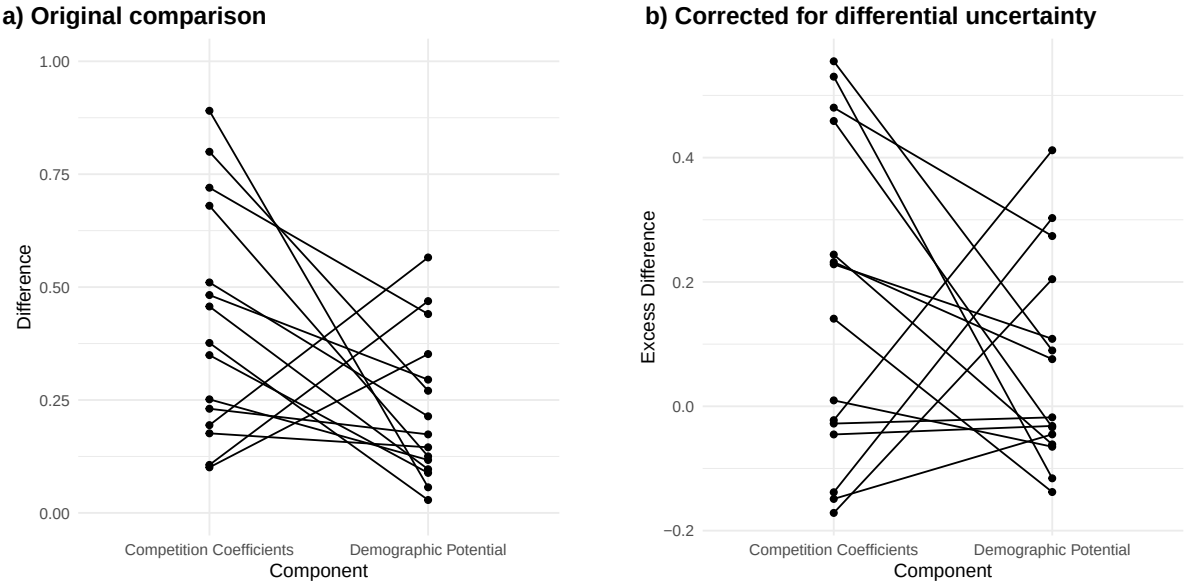
```
##
##  Paired t-test
##
## data:  WhichShiftGeater_df$n_change and WhichShiftGeater_df$a_change
## t = -2.204, df = 14, p-value = 0.04476
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -0.379794917 -0.005171194
## sample estimates:
## mean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

```r
## 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
```

```r
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 Session Information

```
sessionInfo()
```

```
## R version 4.2.2 (2022-10-31 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
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] cowplot_1.1.1   broom_1.0.1     posterior_1.3.1 brms_2.18.0
##  [5] Rcpp_1.0.9      forcats_0.5.2   stringr_1.5.0   dplyr_1.0.10
##  [9] purrr_0.3.5     readr_2.1.3     tidyr_1.2.1     tibble_3.1.8
## [13] ggplot2_3.4.0   tidyverse_1.3.2
##
## loaded via a namespace (and not attached):
##   [1] googledrive_2.0.0    colorspace_2.0-3     ellipsis_0.3.2
##   [4] markdown_1.4         base64enc_0.1-3      fs_1.5.2
##   [7] rstudioapi_0.14      farver_2.1.1         rstan_2.26.13
```

```
##  [10] bit64_4.0.5          DT_0.26              fansi_1.0.3
##  [13] mvtnorm_1.1-3        lubridate_1.9.0      xml2_1.3.3
##  [16] codetools_0.2-18     bridgesampling_1.1-2 knitr_1.41
##  [19] shinythemes_1.2.0    bayesplot_1.10.0     jsonlite_1.8.3
##  [22] dbplyr_2.2.1         shiny_1.7.3          compiler_4.2.2
##  [25] httr_1.4.4           backports_1.4.1      assertthat_0.2.1
##  [28] Matrix_1.5-1         fastmap_1.1.0        gargle_1.2.1
##  [31] cli_3.4.1            later_1.3.0          prettyunits_1.1.1
##  [34] htmltools_0.5.3      tools_4.2.2          igraph_1.3.5
##  [37] coda_0.19-4          gtable_0.3.1         glue_1.6.2
##  [40] reshape2_1.4.4       V8_4.2.2             cellranger_1.1.0
##  [43] vctrs_0.5.1          nlme_3.1-160         crosstalk_1.2.0
##  [46] tensorA_0.36.2       xfun_0.35            ps_1.7.2
##  [49] rvest_1.0.3          timechange_0.1.1     mime_0.12
##  [52] miniUI_0.1.1.1       lifecycle_1.0.3      gtools_3.9.4
##  [55] googlesheets4_1.0.1  zoo_1.8-11           scales_1.2.1
##  [58] vroom_1.6.0          colourpicker_1.2.0   hms_1.1.2
##  [61] promises_1.2.0.1     Brobdingnag_1.2-9    parallel_4.2.2
##  [64] inline_0.3.19        shinystan_2.6.0      curl_4.3.3
##  [67] yaml_2.3.6           gridExtra_2.3        StanHeaders_2.26.13
##  [70] loo_2.5.1            stringi_1.7.8        highr_0.9
##  [73] dygraphs_1.1.1.6     checkmate_2.1.0      pkgbuild_1.4.0
##  [76] rlang_1.0.6          pkgconfig_2.0.3      matrixStats_0.63.0
##  [79] distributional_0.3.1 evaluate_0.18        lattice_0.20-45
##  [82] labeling_0.4.2       rstantools_2.2.0     htmlwidgets_1.5.4
##  [85] bit_4.0.5            processx_3.8.0       tidyselect_1.2.0
##  [88] plyr_1.8.8           magrittr_2.0.3       R6_2.5.1
##  [91] generics_0.1.3       DBI_1.1.3            pillar_1.8.1
##  [94] haven_2.5.1          withr_2.5.0          xts_0.12.2
##  [97] abind_1.4-5          modelr_0.1.10        crayon_1.5.2
## [100] utf8_1.2.2           tzdb_0.3.0           rmarkdown_2.18
## [103] grid_4.2.2           readxl_1.4.1         callr_3.7.3
## [106] threejs_0.3.3        reprex_2.0.2         digest_0.6.30
## [109] xtable_1.8-4         httpuv_1.6.6         RcppParallel_5.1.5
## [112] stats4_4.2.2         munsell_0.5.0        shinyjs_2.1.0
```