Supplementary Methods 2

Chris Terry

Contents

1 Approach Summary

This document is a knit `.rmd` file 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.

I first 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 orginal 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. Briefly, the absolute value of the difference 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 100 simulated datasets, I found this same result in 54 cases, despite there being no 'true' difference between the treatments. Combined with the lack of support for the treatment impacting the competition coefficients, this strongly suggests that the authors' conclusion is driven by the increased uncertainty in competition coefficients driving larger apparent changes.

2 Loading data

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

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),# where 0 background, naming after focal
                              focal, background))

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')
```

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)+(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_list')

### 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_list')
## 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)
```

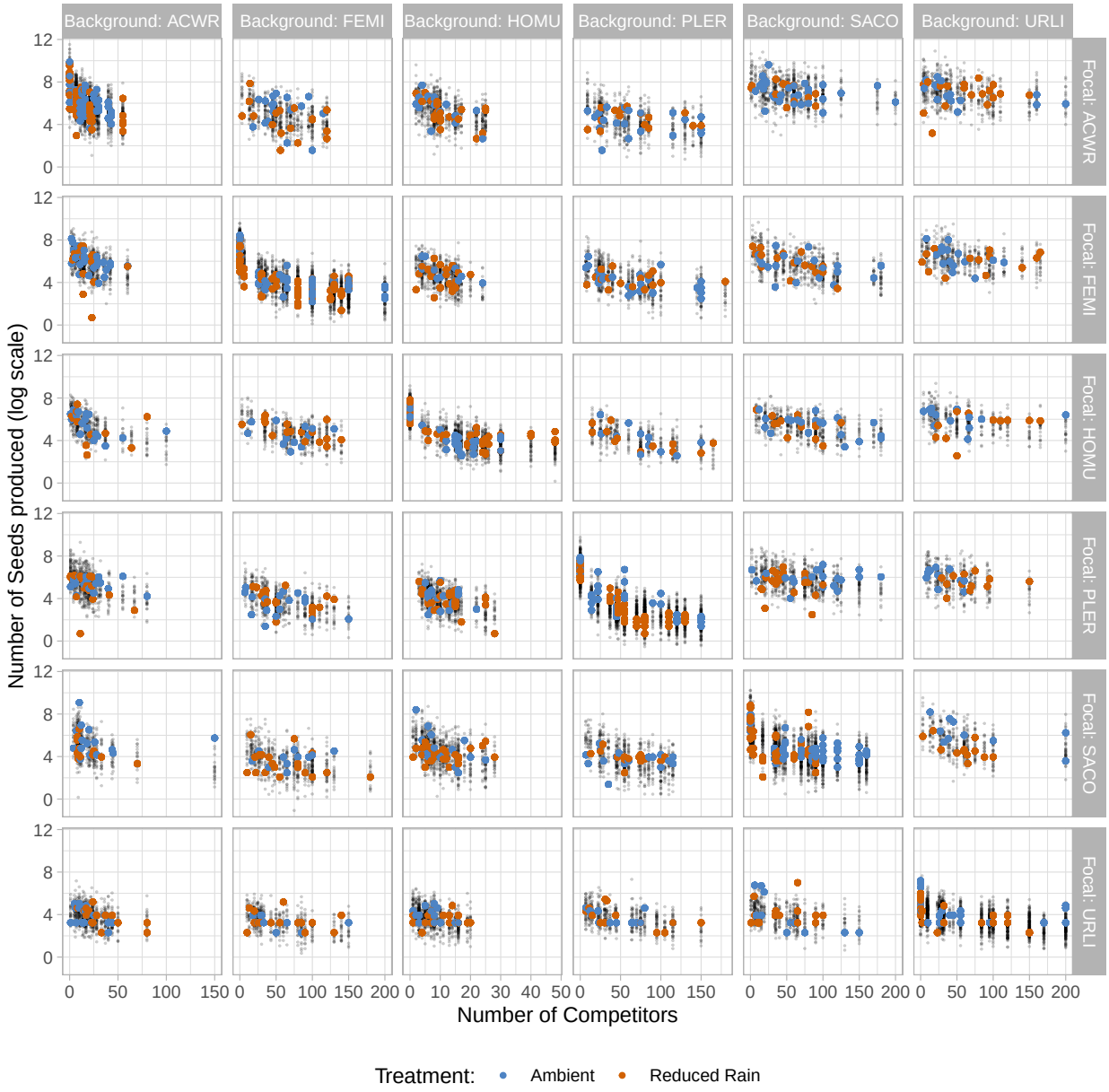
3.4 Function to rebuild data in original layout

```
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)
}
```

3.4.1 Plotting distribution of 20 posterior draws compared to raw data

```
1:20 %>%
  map_df(Gen_simData_fromfit) %>%
  rename(Focal = focal, Background = background) %>%
  arrange(num_comp)%>%
  ggplot(aes( x =num_comp)) +
  geom_point(aes(y = SIM_DATA), size = 0.1, alpha = 0.2)+
  geom_point(aes(y = Log_Seeds, col = treat ), size = 1)+
  facet_grid(Focal~Background, scales = 'free_x',
```

```
            labeller= label_both)+
  theme_light()+
  xlab('Number of Competitors')+
  ylab('Number of Seeds produced (log scale)')+
  scale_color_manual(values=c('W' ="#4E84C4", 'D' = "#D16103"),
                     name = "Treatment:",
                     labels = c("Ambient", "Reduced Rain"))+
  theme(legend.position = 'bottom')
```



4 Extracting parameters and key quantities from simulation data

4.1 Utility functions

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)
}
```

4.2 Testing analysis routes match

Values won't match exactly as 'best-fit' compared to median of bootstrap, but are sufficiently close to be confident in approach.

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

OrigData_newAnalysis %>%
  filter(focal =='PLER')%>%
  arrange(competitor, treatment) %>%
  select(sp_pair, treatment, alpha, lambda, fd, snd)
```

```
##      sp_pair treatment       alpha   lambda        fd         snd
## 1  PLER_ACWR         1 0.079307655 627.2254 0.5567532  0.36733966
## 2  PLER_ACWR         2 0.242143410 675.3974 1.3871430  0.57030226
## 3  PLER_FEMI         1 0.415987286 627.2254 2.1455654 -0.08348895
## 4  PLER_FEMI         2 0.250875885 675.3974 2.5373152  0.62119187
## 5  PLER_HOMU         1 1.006005298 627.2254 1.6303761  0.35145111
## 6  PLER_HOMU         2 1.262002358 675.3974 3.4312363  0.31731278
## 7  PLER_PLER         1 0.403801854 627.2254 1.0000000  0.00000000
## 8  PLER_PLER         2 1.048769293 675.3974 1.0000000  0.00000000
```

```
## 9  PLER_SACO         1 0.007182081 627.2254 0.1885914  0.77422276
## 10 PLER_SACO         2 0.039221058 675.3974 0.5728420  0.78316291
## 11 PLER_URLI         1 0.021114686 627.2254 0.6126184  0.66799605
## 12 PLER_URLI         2 0.024119961 675.3974 0.6815086  0.85236745
```

```
nls_boot_pairs %>%
  filter(focal =='PLER')%>%
  arrange(competitor, treatment) %>%
  select(sp_pair,treatment,  alpha, lambda, fd, snd)
```

```
##      sp_pair treatment       alpha   lambda        fd         snd
## 1  PLER_ACWR         1 0.084830786 637.7883 0.5666341  0.37253136
## 2  PLER_ACWR         2 0.239371678 683.6565 1.4074441  0.56836872
## 3  PLER_FEMI         1 0.429967755 637.7883 2.1720534 -0.07698861
## 4  PLER_FEMI         2 0.253485730 683.6565 2.5318429  0.62361206
## 5  PLER_HOMU         1 1.022167042 637.7883 1.6157786  0.35698539
## 6  PLER_HOMU         2 1.274064322 683.6565 3.4526655  0.31314343
## 7  PLER_PLER         1 0.416149539 637.7883 1.0000000  0.00000000
## 8  PLER_PLER         2 1.058661899 683.6565 1.0000000  0.00000000
## 9  PLER_SACO         1 0.007441751 637.7883 0.1879160  0.76986908
## 10 PLER_SACO         2 0.039975503 683.6565 0.5794450  0.78109243
## 11 PLER_URLI         1 0.021908605 637.7883 0.5736173  0.67908586
## 12 PLER_URLI         2 0.024383190 683.6565 0.6771512  0.85345809
```

5 Repeating Analysis

5.1 Utility functions

Code largely sourced from `n_alpha_ratios.r`

```
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){
  if(all_fit==0){return(NULL)}

  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")

  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))
}
```

5.2 Testing pipeline can reproduce original results

Again, expect a slight difference between best fit and average of bootstrap. But close enough to original paper's finding of p=0.044

```
Calc_t.test_ratiodiffs(OrigData_newAnalysis)
```

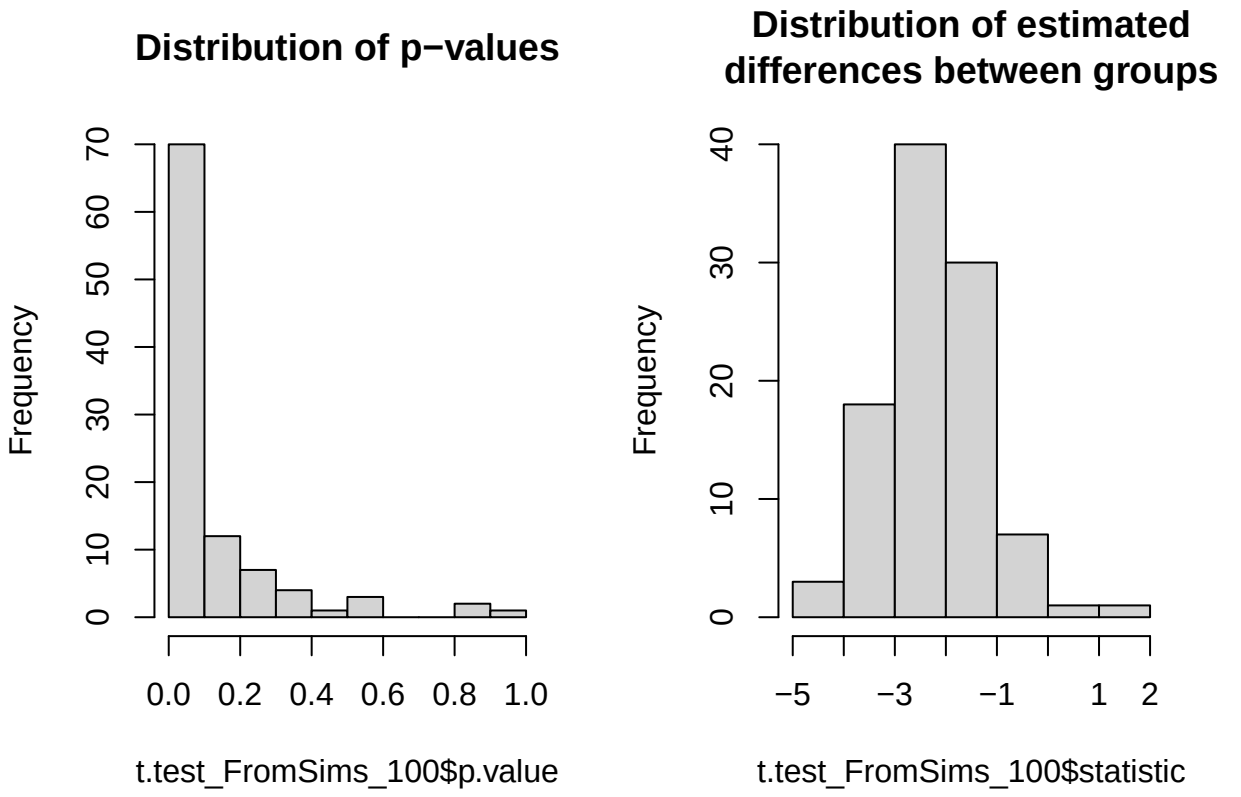
```
## Warning in if (all_fit == 0) {: the condition has length > 1 and only the first
## element will be used

## # 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  0.0489        14   -0.391  -0.00113 Paired t-test two.sid~
## # ... with abbreviated variable name 1: alternative
```

5.3 Replicating analysis on simulated data

```
1:110 %>% # make slightly too many, so can discard those few that fail to converge
  map(Gen_simData_fromfit)%>%
  map(Calc_all_fit_sim)%>%
```

```
  map_df(Calc_t.test_ratiodiffs )%>%
  slice(1:100)-> t.test_FromSims_100

write_csv(t.test_FromSims_100, 't.test_FromSims_100.csv')
# ^^ not very optimised so takes a few minutes to run
```

```
t.test_FromSims_100 <- read_csv('t.test_FromSims_100.csv')

par(mfrow= c(1,2))
hist(t.test_FromSims_100$p.value , main = 'Distribution of p-values')
hist(t.test_FromSims_100$statistic , main = 'Distribution of estimated\ndifferences between groups' )
```




```
mean(t.test_FromSims_100$p.value < 0.05)
```

```
## [1] 0.54
```

```
t.test_FromSims_100 %>%
  count(statistic   <0 & p.value <0.05 )
```

```
## # A tibble: 2 x 2
##   `statistic < 0 & p.value < 0.05`     n
##   <lgl>                            <int>
## 1 FALSE                               46
## 2 TRUE                                54
```

6 Session Information

```
sessionInfo()
```

```
## R version 4.1.1 (2021-08-10)
## 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.1252
## [2] LC_CTYPE=English_United Kingdom.1252
## [3] LC_MONETARY=English_United Kingdom.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United Kingdom.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] broom_1.0.1     posterior_1.3.1 brms_2.18.0     Rcpp_1.0.9
##  [5] forcats_0.5.2   stringr_1.4.1   dplyr_1.0.10    purrr_0.3.5
##  [9] readr_2.1.3     tidyr_1.2.1     tibble_3.1.8    ggplot2_3.3.6
## [13] tidyverse_1.3.2
##
## loaded via a namespace (and not attached):
##   [1] TH.data_1.1-1        googledrive_2.0.0    colorspace_2.0-3
##   [4] ellipsis_0.3.2       ggridges_0.5.4       estimability_1.4.1
##   [7] markdown_1.3         base64enc_0.1-3      fs_1.5.2
##  [10] rstudioapi_0.14      farver_2.1.1         rstan_2.21.7
##  [13] bit64_4.0.5          DT_0.26              fansi_1.0.3
##  [16] mvtnorm_1.1-3        lubridate_1.8.0      xml2_1.3.3
##  [19] splines_4.1.1        codetools_0.2-18     bridgesampling_1.1-2
##  [22] knitr_1.40           shinythemes_1.2.0    bayesplot_1.9.0
##  [25] jsonlite_1.8.3       dbplyr_2.2.1         shiny_1.7.3
##  [28] compiler_4.1.1       httr_1.4.4           emmeans_1.8.2
##  [31] backports_1.4.1      assertthat_0.2.1     Matrix_1.5-1
##  [34] fastmap_1.1.0        gargle_1.2.1         cli_3.4.1
##  [37] later_1.3.0          prettyunits_1.1.1    htmltools_0.5.3
##  [40] tools_4.1.1          igraph_1.3.5         coda_0.19-4
##  [43] gtable_0.3.1         glue_1.6.2           reshape2_1.4.4
##  [46] cellranger_1.1.0     vctrs_0.5.0          nlme_3.1-152
##  [49] crosstalk_1.2.0      tensorA_0.36.2       xfun_0.34
##  [52] ps_1.7.2             rvest_1.0.3          mime_0.12
##  [55] miniUI_0.1.1.1       lifecycle_1.0.3      gtools_3.9.3
##  [58] googlesheets4_1.0.1  MASS_7.3-54          zoo_1.8-11
##  [61] scales_1.2.1         vroom_1.6.0          colourpicker_1.2.0
##  [64] hms_1.1.2            promises_1.2.0.1     Brobdingnag_1.2-9
##  [67] sandwich_3.0-2       parallel_4.1.1       inline_0.3.19
##  [70] shinystan_2.6.0      yaml_2.3.6           gridExtra_2.3
##  [73] StanHeaders_2.21.0-7 loo_2.5.1            stringi_1.7.8
```

```
##  [76] highr_0.9            dygraphs_1.1.1.6     checkmate_2.1.0
##  [79] pkgbuild_1.3.1       rlang_1.0.6          pkgconfig_2.0.3
##  [82] matrixStats_0.62.0   distributional_0.3.1 evaluate_0.17
##  [85] lattice_0.20-44      labeling_0.4.2       rstantools_2.2.0
##  [88] htmlwidgets_1.5.4    bit_4.0.4            processx_3.8.0
##  [91] tidyselect_1.2.0     plyr_1.8.7           magrittr_2.0.3
##  [94] R6_2.5.1             generics_0.1.3       multcomp_1.4-20
##  [97] DBI_1.1.3            pillar_1.8.1         haven_2.5.1
## [100] withr_2.5.0          xts_0.12.2           survival_3.2-11
## [103] abind_1.4-5          modelr_0.1.9         crayon_1.5.2
## [106] utf8_1.2.2           tzdb_0.3.0           rmarkdown_2.17
## [109] grid_4.1.1           readxl_1.4.1         callr_3.7.3
## [112] threejs_0.3.3        reprex_2.0.2         digest_0.6.30
## [115] xtable_1.8-4         httpuv_1.6.6         RcppParallel_5.1.5
## [118] stats4_4.1.1         munsell_0.5.0        shinyjs_2.1.0
```