

# Paleo superstatistics notebook

## Getting the data

### PBDB API

To obtain the PBDB data we make use of the API in script `data/pbdb_data_get.R`, which accesses the API and cleans the data by:

- removing poorly lithified specimens
- removing collections at the basin scale
- including only fine-scale stratigraphy (below the “group” level)
- resolving taxonomy to the genus or subgenus level where available (storing genus or subgenus as `otu`)
- combining multiple records of the same OTU per collection
- importing standardized timebins from fossilworks.org (timebins are scraped with script `data/fossilworks_tbins_intervals.R`)

The data gathering script `data/pbdb_data_get.R` is shown below:

```
setwd('data')

# call to the API
show <- paste0(c('ident', 'phylo', 'lith', 'loc', 'time', 'geo', 'stratext',
                 'ecospace'),
               collapse = ',')

version <- '1.2'
base_name <- 'Animalia~Craniata'
min_ma <- 0
max_ma <- 560
timerule <- 'contain'
envtype <- 'marine'

# break-up backbone URI just so it can be nicely displayed
bbURI <- paste0('https://paleobiodb.org/data%s/occs/list.csv?',
               'base_name=%s&show=%s&limit=all&min_ma=%s&max_ma=%s&',
               'timerule=%s&envtype=%s')

# the actual call to the URI
uri <- sprintf(bbURI,
               version,
               base_name,
               show,
               min_ma,
               max_ma,
               timerule,
               envtype)

# get pbdb occurrences
x <- read.csv(uri, as.is = TRUE)

# write out raw data
write.csv(x, 'pbdb_data_raw.csv', row.names = FALSE)
```

```

# clean up

# remove unnecceary columns
c2rm <- c('record_type', 'reid_no', 'flags', 'identified_name',
         'identified_rank', 'identified_no', 'difference', 'species_name',
         'species_reso', 'lithdescript', 'lithology1', 'minor_lithology1',
         'lithology2', 'lithification2', 'minor_lithology2', 'cc', 'state',
         'county', 'latlng_basis', 'geogcomments', 'geology_comments',
         'zone', 'localsection', 'localbed', 'localorder',
         'regionalsection', 'regionalbed', 'regionalorder',
         'stratcomments')
x <- x[, !(names(x) %in% c2rm)]

# only well lithified specimens
x <- x[x$lithification1 %in% c('', 'lithified'), ]

# no basin-scale collections
x <- x[x$geogscale != 'basin', ]

# fine scale stratigraphy only
x <- x[!(x$stratscale %in% c('group', 'supergroup')), ]

# resolve taxonomy to genus or subgenus where availible
otu <- x$genus
otu[x$subgenus_name != ''] <- ifelse(x$subgenus_reso[x$subgenus_name != ''] == '',
                                   x$subgenus_name[x$subgenus_name != ''],
                                   otu[x$subgenus_name != ''])
otu[x$primary_reso != ''] <- ''
x$otu <- otu
x <- x[x$otu != '', ]

# combine multiple records of same otu per collection
x <- x[!duplicated(x[, c('collection_no', 'otu'))], ]

# standard time bins
stages <- read.csv('tbins_stages.csv', as.is = TRUE)
earlyTbin <- stages$tbin[match(x$early_interval, stages$name)]
lateTbin <- stages$tbin[match(x$late_interval, stages$name)]
lateTbin[is.na(lateTbin)] <- earlyTbin[is.na(lateTbin)]
earlyTbin[earlyTbin != lateTbin] <- NA

x$tbin <- earlyTbin
x <- x[!is.na(x$tbin), ]

# write out fully processed data
write.csv(x, 'pbdb_data.csv', row.names = FALSE)

```

## Scraping fossilworks

The script to pull Alory's time bins (data/fossilworks\_tbins\_intervals.R) is below:

```
# **script to scrape time bins from fossilworks.org**

options(stringsAsFactors = FALSE)

setwd('data')

# loop through interval info on fossilworks.org
coreURI <- 'http://fossilworks.org/bridge.pl?a=displayInterval&interval_no='

tbinfo <- lapply(1:1108, function(i) {
  print(i)
  linfo <- try(readLines(paste0(coreURI, i), n = 150))

  if('try-error' %in% class(linfo))
    linfo <- try(readLines(paste0(coreURI, i), n = 150))

  if('try-error' %in% class(linfo)) {
    thisTbin <- thisMax <- thisMin <- thisName <- NA
  } else {
    thisTbin <- gsub('^.*10 million year bin: |<br>.*$', '',
                    linfo[grepl('10 million year bin', linfo)])
    thisMax <- as.numeric(gsub('^.*Lower boundary: equal to | Ma.*|^[^0-9\\.]', '',
                              linfo[grepl('Lower boundary: equal to', linfo)]))
    thisMin <- as.numeric(gsub('^.*Upper boundary: equal to | Ma.*|^[^0-9\\.]', '',
                              linfo[grepl('Upper boundary: equal to', linfo)]))
    thisName <- gsub('^.*<p class="pageTitle">|</p>.*$', '',
                    linfo[grepl('class="pageTitle"', linfo)])
  }

  return(data.frame(name = ifelse(length(thisName) == 0, NA, thisName),
                    tbin = ifelse(length(thisTbin) == 0, NA, thisTbin),
                    ma_min = ifelse(length(thisMin) == 0, NA, thisMin),
                    ma_max = ifelse(length(thisMax) == 0, NA, thisMax)))
})

tbinfo <- do.call(rbind, tbinfo)

# clean up
# -----

tbinfo <- tbinfo[!is.na(tbinfo$name), ]

# remove 'stage' and equivilant from name
tbinfo$name <- gsub(' [[:lower:]].*', '', tbinfo$name)

# split up stages with a '/' into both names
temp <- tbinfo[grepl('/', tbinfo$name), ]
tbinfo$name <- gsub('.*/', '', tbinfo$name)
temp$name <- gsub('/.* ', ' ', temp$name)
```

```

tbinInfo <- rbind(tbinInfo, temp)

# fix random typo
tbinInfo$name[tbinInfo$name == 'Cazenovia'] <- 'Cazenovian'

# write out
write.csv(tbinInfo, 'tbins_stages.csv', row.names = FALSE)

# also write out summary of each time bin, most importantly (for plottin)
# its midpoint

tbinmid <- sapply(unique(tbinInfo$tbin[!is.na(tbinInfo$tbin)]), function(tbin) {
  tt <- unlist(tbinInfo[tbinInfo$tbin == tbin, c('ma_min', 'ma_max')])
  return(mean(range(tt, na.rm = TRUE)))
})

tbinmid <- sort(tbinmid, decreasing = TRUE)

write.csv(data.frame(tbin = names(tbinmid), ma_mid = as.numeric(tbinmid)), 'data/tbinsMid.csv',
  row.names = FALSE)

```

## Three timer and publication bias correction

Once the data have been downloaded and cleaned, we correct for incomplete and biased sampling with the script `data/pbdb_3TPub_make.R` which sources the function `R/make3TPub.R` to generate the main output: a matrix with time bins as rows, taxa (families in this case) as columns and bias-corrected richness as cells.

```

# **this script produces diversity estimates per family per time bin from PBDB data
# corrected by the '3 timers method' and for possible publication bias**

# source function to produce a matrix of time by taxon with cells
# of corrected diversity
source('R/make3TPub.R')

# load other needed functions

# source('code/sstat_comp.R')
# source('code/sstat_methods.R')
# source('code/Px_gam.R')

# load and prepare data
# -----

pbdbDat <- read.csv('data/pbdb_data.csv', as.is = TRUE)

# make column for midpoint ma
pbdbDat$ma_mid <- (pbdbDat$max_ma + pbdbDat$min_ma) / 2

# get rid of poor temporal resolution
pbdbDat <- pbdbDat[pbdbDat$tbin != '', ]

```

```

# get rid of bad taxonomy
pbdbDat <- pbdbDat[pbdbDat$family != '', ]
pbdbDat <- pbdbDat[pbdbDat$otu != '', ]

# get bin times
pbdbDat$mid_ma <- (pbdbDat$min_ma + pbdbDat$max_ma) / 2
pbdbTime <- sort(tapply(pbdbDat$mid_ma, pbdbDat$tbin, mean))
pbdbDat$tbin <- factor(pbdbDat$tbin, levels = names(pbdbTime))

# data.frame to hold publication, diversity and 3T stat
famTbinBias <- aggregate(list(div = pbdbDat$otu), list(fam = pbdbDat$family,
                                                       tbin = pbdbDat$tbin),
                         function(x) length(unique(x)))

# three timer stat and publication bias
# -----

# matrix to determine three timers and part timers (sensu alroy 2008)
mt <- matrix(0, nrow = nlevels(pbdbDat$tbin),
            ncol = nlevels(pbdbDat$otu))
diag(mt) <- -10
mt[abs(row(mt) - col(mt)) == 1] <- 1

# loop through and compute three timers and part timers
timers <- lapply(split(pbdbDat$tbin, pbdbDat$otu),
                 function(x) {
                   # browser()
                   tbins <- integer(nlevels(x))
                   tbins[as.integer(unique(x))] <- 1
                   t3 <- as.integer(mt %*% tbins == 2)
                   tp <- as.integer(mt %*% tbins == -8)

                   return(cbind(t3, tp))
                 })

# compute 3 timer stat from 3 timers and part timers
timers <- array(unlist(timers), dim = c(nrow(timers[[1]]), 2, length(timers)))
t3stat <- 1 - rowSums(timers[, 1, ]) / (rowSums(timers[, 1, ]) + rowSums(timers[, 2, ]))

# add to data.frame holding all info to be saved
famTbinBias$T3Stat <- t3stat[match(famTbinBias$tbin,
                                  levels(pbdbDat$tbin))]
famTbinBias$T3Div <- famTbinBias$div / famTbinBias$T3Stat

# record pubs per tbin
tbinPub <- tapply(pbdbDat$reference_no, pbdbDat$tbin,
                  function(x) length(unique(x)))
famTbinBias$tbinPub <- tbinPub[famTbinBias$tbin]

# calculate corrected diversity
pdf('ms/figSupp_divByPub_foo.pdf', width = 4, height = 4)

```

```

pbdbFamDiv <- with(famTbinBias,
                  make3TPub(div, T3Stat, tbinPub, fam, tbin, pbdbTime,
                           minPub = 10, plotit = TRUE))
dev.off()

# write out corrected diversity
write.csv(pbdbFamDiv, 'data/pbdb_3TPub-corrected.csv')

```

Here is the guts of the make3TPub function

```

#' @description function to produce a matrix of time by taxa with cells of corrected diversity
#' @param rawDiv the raw diversity of each taxon in each time interval
#' @param t3stat the 3 timer stat for each diversity record
#' @param pub the number of publications associated with each diversity record
#' @param taxa the taxon names for each diversity record
#' @param tbin the time interval of each diversity record
#' @param tbinTime times associated with each `tbin`
#' @param minPub minimum number of publications for inclusion in regression analysis
#' @param plotit logical, should plot of taxon richness versus number of publications be made
#' @return a matrix with rows corresponding to time intervals and columns to the given taxa
#' each cell in the matrix represents corrected taxon richness

make3TPub <- function(rawDiv, t3stat, pub, taxa, tbin, tbinTime,
                     minPub = 10, plotit = FALSE) {
  # put data together so can be universally manipulated
  x <- data.frame(rawDiv = rawDiv, t3stat = t3stat, pub = pub, taxa = taxa, tbin = tbin)
  x$tbin <- as.character(x$tbin)
  x$taxa <- as.character(x$taxa)

  x <- x[!is.na(t3stat) & pub >= minPub, ]

  tbinTime <- tbinTime[names(tbinTime) %in% x$tbin]

  # 3-timer correction
  t3cor <- x$rawDiv/x$t3stat

  # publication correction
  logPub <- log(x$pub)
  pubLM <- lm(log(t3cor)~logPub)
  pbdbPubLM <- pubLM # save regression to global env

  pubResid <- exp(pubLM$residuals)

  # plot so you can verify cutoff etc.
  if(plotit) {
    plot(log(x$pub), log(t3cor),
         xlab = 'log(Number of publications)',
         ylab = 'log(3T-corrected number of genera)')
    abline(pubLM, col = 'red')
  }

  tbinTaxa <- socorro::tidy2mat(x$tbin, x$taxa, pubResid)

```

```

    return(tbinTaxa[names(sort(tbinTime, decreasing = TRUE)), ])
}

```

## Super statistical analysis of 3TPub-corrected PDB data

Once data have been bias-corrected we can complete their super-statistical analysis. We do that in the script `analysis/pbdb_sstat.R` shown here:

```

# **script to run super stat analysis on PBDB data and make plots**

# source needed functions
R.utils::sourceDirectory('R', modifiedOnly = FALSE)
library(socorro) # for plotting
library(viridis) # colors for plotting

# load and prepare data
# -----

pbdbFamDiv <- read.csv('data/pbdb_3TPub-corrected.csv', row.names = 1)

# coarsen to higher taxonomic groupings

# tbin midpoints
tbinMid <- read.csv('data/tbinsMid.csv', as.is = TRUE)
tbinNames <- tbinMid$tbin
tbinMid <- as.numeric(tbinMid[, 2])
names(tbinMid) <- tbinNames

tbinMid <- tbinMid[rownames(pbdbFamDiv)]

# super stat analysis for families
# -----

# corrected flux
pbdbFamFlux <- apply(pbdbFamDiv, 2, function(x) {
  flux <- diff(c(0, x))
  return(flux[flux != 0])
})

# make sstat object
sstatPBDBord3TP <- sstatComp(pbdbFamFlux, minN = 10, plotit = FALSE)

# likelihood CI for family-level sstat analysis
sstatPBDBord3TPCI <- bootMLE.sstat(sstatPBDBord3TP, B = 1000, useAll = FALSE)

# plot it
pdf('ms/fig_Px.pdf', width = 4.25, height = 4)

par(mar = c(3, 3, 0, 0) + 0.5, mgp = c(2, 0.5, 0), cex.lab = 1.4)
plot(sstatPBDBord3TP, xlim = c(1e-04, 4e+01), ylim = c(8e-05, 1),
     xaxt = 'n', yaxt = 'n',
     panel.first = quote(mlePoly(sstatPBDBord3TPCI$sstat, PPx.gam,

```

```

col = hsv(alpha = 0.25), border = NA)))
logAxis(1:2, expLab = TRUE)

dev.off()

# plot p_k(x/b) and f(beta) for families
# -----

# idea for normality test: sample 1 from each order and do ks test on that subsampled set

# highlight these families

# make CDF for all scale family-level fluctuations
pAll <- lapply(sstatPBDBord3TP$raw.pk,
              function(x) simpECDF(scale(x)[, 1], complement = TRUE))
pAll <- do.call(rbind, pAll)

# function to help with individual trajectory plotting
trajLines <- function(t, x, ...) {
  x[x == 0] <- NA
  alive <- range(which(!is.na(x)))

  x[min(alive) - 1] <- 0
  x[max(alive) + 1] <- 0

  t <- t[!is.na(x)]
  x <- x[!is.na(x)]

  lines(t, x, ...)
}

# the actual plotting

layout(matrix(c(1, 2, 1, 3), nrow = 2))

par(oma = c(0, 3, 0, 0) + 0.25, mar = c(4, 0, 0, 0) + 0.25,
    mgp = c(2, 0.5, 0), cex.lab = 1.4)

# highlight individual trajectories
lo <- pbdbFamDiv[, 'Tainoceratidae']
mi <- pbdbFamDiv[, 'Lophospiridae']
hi <- pbdbFamDiv[, 'Chonetidae']
cols <- hsv(h = c(0.7, 0.45, 0.12), s = c(0.7, 1, 1), v = c(0.8, 0.8, 1))
names(cols) <- c('hi', 'mi', 'lo')

plot(1, xlim = c(540, 0), xaxt = 'n', xaxs = 'i', xlab = '',
     ylim = c(0, max(lo, mi, hi, na.rm = TRUE)), type = 'n')

trajLines(tbinMid, lo, col = cols['lo'])
trajLines(tbinMid, mi, col = cols['mi'])

```



```

trajLines(tbinMid, hi, col = cols['hi'])

paleoAxis(1)
mtext('Millions of years ago', side = 1, line = 3.5)

par(mar = c(3, 0, 1, 0) + 0.25)
plot(pAll, xlim = c(-4, 4), col = 'gray', ylim = c(0, 1.025),
     xlab = 'Scaled fluctuations', ylab = 'Cumulative density')
curve(pnorm(x, lower.tail = FALSE), lwd = 2, add = TRUE)

plot(simpECDF(sstatPBDBord3TP$beta, complement = TRUE), ylim = c(0, 1.025),
     log = 'x', xaxt = 'n', yaxt = 'n',
     xlab = expression(beta), col = 'gray')
logAxis(1, expLab = TRUE)
curve(pgamma(x, sstatPBDBord3TP$gam.par[1], sstatPBDBord3TP$gam.par[2],
            lower.tail = FALSE),
     col = 'black', lwd = 2, add = TRUE)

# low = Tainoceratidae
# mid = Lophospiridae
# hi = Chonetidae

low <- which(sstatPBDBord3TP$beta < 2^-(2 - 0.25) & sstatPBDBord3TP$beta > 2^-(2 + 0.25))
mid <- which(sstatPBDBord3TP$beta < 2^(0.5 + 0.25) & sstatPBDBord3TP$beta > 2^(0.5 - 0.25))
hi <- which(sstatPBDBord3TP$beta < 2^(3 + 0.25) & sstatPBDBord3TP$beta > 2^(3 - 0.25))

sort(colSums(pbdbFamDiv[, pbdbFamDiv[nrow(pbdbFamDiv), ]] > 0))

```

## To-do

- remove all deprecated files and folders...do it in one fell swoop so we can roll back if needed ever
- make the 3TPub script seapparate and have it save (made script, haven't tried running):
  1. the plot of diversity through time (maybe not?)
- make a separate script for the plotting of the sstat analysis including:
  1. first it needs to make the corrected diversity fluctuations
  2. the main sstat style CDF
    - need to remove the raw data from the plot
    - need to make sure 95% CI still works
  3. the f(beta) plot
  4. the p\_k(x) plot
  5. example trajectories
- do KS analysis with Families
- run clean functions on sepkoski
- manuscript text
  - add more recent citations
  - add more thorough explanation and justification of correction approach
  - respond to other reviewer comments