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 availible (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:

```r
setwd('~/Dropbox/Research/paleo_supStat/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 occurences
x <- read.csv(uri, as.is = TRUE)

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

```r
# 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:

```r
options(stringsAsFactors = FALSE)

setwd('~/Dropbox/Research/paleo_supStat/data')

coreURI <- 'http://fossilworks.org/bridge.pl?a=displayInterval&interval_no='

tbinInfo <- 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>.*$', '',
                         linfo[grep('10 million year bin', linfo)])
        thisMax <- as.numeric(gsub('^.*Lower boundary: equal to | Ma.*$|[^0-9\\.]', '',
                                   linfo[grep('Lower boundary: equal to', linfo)]))
        thisMin <- as.numeric(gsub('^.*Upper boundary: equal to | Ma.*$|[^0-9\\.]', '',
                                   linfo[grep('Upper boundary: equal to', linfo)]))
        thisName <- gsub('^.*<p class="pageTitle">|</p>.*$', '',
                         linfo[grep('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)))
})

tbinInfo <- do.call(rbind, tbinInfo)


## clean up

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

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

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

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

```r
## remove time periods that do not fall completely within a 10my bin
# tbinInfo <- tbinInfo[!is.na(tbinInfo$tbin), ]

## write out
write.csv(tbinInfo, 'tbins_stages.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 `re-do.R`

```r
makePlot <- TRUE

oldwd <- setwd('~/Dropbox/Research/paleo_supStat')

# convenience function to produce a matrix of time by ord with cells
# of corrected diversity
source('code/pbdb_3t_pub.R')

# load other needed funcitons
paleoPlot <- function(...) {
    plot(..., xaxt = 'n')
    socorro::paleoAxis(1)
}
# source('~/R_functions/paleoPlot.R')

samp2site.spp <- socorro::tidy2mat
# source('~/R_functions/samp2site_spp.R')

logPlot <- function(..., log) {
    plot(..., log = log,
         xaxt = ifelse(grepl('x', log), 'n', 's'),
         yaxt = ifelse(grepl('y', log), 'n', 's'))
    s <- (1:2)[c(grepl('x', log), grepl('y', log))]

    socorro::logAxis(s)
}
# source('~/R_functions/logPlot.R')

my.ecdf <- socorro::simpECDF
# source('~/R_functions/my_ecdf.R')

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

##########  load data  ##########
setwd('data/old/pbdb_2013-05-28')

# raw occurence data
# pbdb.dat <- read.csv('marInv-occs.csv', as.is = TRUE)

pbdbNew <- read.csv('../../pbdb_data.csv', as.is = TRUE)
```

```r
## convert column names
names(pbdbNew)[names(pbdbNew) == 'tbin'] <- 'collections.10_my_bin'
names(pbdbNew)[names(pbdbNew) == 'family'] <- 'occurrences.order_name'
names(pbdbNew)[names(pbdbNew) == 'genus'] <- 'occurrences.genus_name'
names(pbdbNew)[names(pbdbNew) == 'reference_no'] <- 'collections.reference_no'
pbdbNew$ma_mid <- (pbdbNew$max_ma + pbdbNew$min_ma) / 2

## use new instead of old
pbdb.dat <- pbdbNew


## get rid of poor temporal resolution
pbdb.dat <- pbdb.dat[pbdb.dat$collections.10_my_bin != '', ]

# get rid of bad taxonomy
pbdb.dat <- pbdb.dat[pbdb.dat$occurrences.order_name != '', ]
pbdb.dat <- pbdb.dat[pbdb.dat$occurrences.order_name != 'Ammonitida', ]
pbdb.dat <- pbdb.dat[pbdb.dat$occurrences.genus_name != '', ]

# get bin times
pbdb.time <- sort(tapply(pbdb.dat$ma_mid, pbdb.dat$collections.10_my_bin, mean))
pbdb.dat$collections.10_my_bin <- factor(pbdb.dat$collections.10_my_bin,
                                         levels = names(pbdb.time))


# data.frame of publication, diversity and 3T stat
ord.tbin.bias <- aggregate(list(div=pbdb.dat$occurrences.genus_name),
                           list(ord=pbdb.dat$occurrences.order_name,
                                tbin=pbdb.dat$collections.10_my_bin),
                           function(x) length(unique(x)))


## three timer stat

## matrix to determine three timers and part timers (sensu alroy 2008)
mt <- matrix(0, nrow = nlevels(pbdb.dat$collections.10_my_bin),
             ncol = nlevels(pbdb.dat$collections.10_my_bin))
diag(mt) <- -10
mt[abs(row(mt) - col(mt)) == 1] <- 1

## loop through and compute three timers and part timers
timers <- lapply(split(pbdb.dat$collections.10_my_bin, pbdb.dat$occurrences.genus_name),
                 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))
                 })

timers <- array(unlist(timers), dim = c(nrow(timers[[1]]), 2, length(timers)))
```

```r
t3stat <- 1 - rowSums(timers[, 1, ]) / (rowSums(timers[, 1, ]) + rowSums(timers[, 2, ]))

ord.tbin.bias$T3.stat <- t3stat[match(ord.tbin.bias$tbin,
                                      levels(pbdb.dat$collections.10_my_bin))]
ord.tbin.bias$T3.div <- ord.tbin.bias$div/ord.tbin.bias$T3.stat

# record pubs per tbin
tbin.pub <- tapply(pbdb.dat$collections.reference_no,pbdb.dat$collections.10_my_bin,
                   function(x) length(unique(x)))
ord.tbin.bias$tbin.pub <- tbin.pub[ord.tbin.bias$tbin]


# calculate corrected diversity

pbdb.ord.div <- with(ord.tbin.bias,
                     pbdb.3t.pub(div, T3.stat, tbin.pub, ord, tbin, pbdb.time,
                                 min.pub=10, plotit=makePlot))

# corrected flux
pbdb.ord.flux <- apply(pbdb.ord.div,2,function(x) {
    raw.flux <- diff(c(0,x))
    return(raw.flux[raw.flux != 0])
})

# sstat analysis
pbdb.sstat.ord.cor <- sstat.comp(pbdb.ord.flux,minN=10,plotit=makePlot)
```