Identification of microbial reference frames and feature reduction using Pairbal

Hayden Brochu

2024-02-26

This is a vignette that shows an example of how to apply the microbial feature reduction (Pairbal) and reference frame methods described in the study "Pre-challenge gut microbial signature predicts RhCMV/SIV vaccine efficacy in rhesus macaques." To run this analysis yourself, you will need to have the Pairbal R function script (Pairbal.R) in your working directory as well as the Supplementary Data (AdditionalFile3_SupplementaryData.xlsx). Please note that Pairbal.R is a Windows/Mac friendly version of the code that also works in Linux. Pairbal_linux.R is a Linux-only version that uses the parallel and pbmcapply packages for multi-threading capabilities. For large datasets, it is recommended that users perform the reference frame MCMC analysis in Linux with multi-threading.

Setup and data ingestion

Load Pairbal and reference frame functions, then load Supplementary Data which are used throughout this vignette.

```
# Install packages if needed
# !! Pairbal only needs tidyverse
.cran_packages <- c("readxl", "tidyverse")
.inst <- .cran_packages %in% installed.packages()
if(any(!.inst)) install.packages(.cran_packages[!.inst], repos='http://cran.us.r-project.org')

# Load packages
suppressPackageStartupMessages(sapply(.cran_packages, require, character.only = T))
```

```
##    readxl tidyverse
##      TRUE      TRUE
```

```
# Load Pairbal and reference frame functions
source("Pairbal.R") # loads tidyverse package on the fly

# Load Supplementary Data
sdata_file <- "AdditionalFile3_SupplementaryData.xlsx"
sheets <- excel_sheets(sdata_file)
sdata <- lapply(sheets, function(sheet) read_xlsx(sdata_file, sheet)) %>% setNames(sheets)
```

Part 1: identification of microbial reference frames

First, we must identify a microbial reference frame for our dataset. In this case, we perform a shallow MCMC search using the aggregated count data in the Supplementary Data ("counts_agg_raw" sheet). Please note

the parameterization ('mcmc_iterations', 'min.cnt', 'min.pct', 'smin', and 'smax'). The MCMC iterations ('mcmc_iterations') must be reasonably large to identify highly stable reference frames (if they exist). In the study we run five MCMC experiments each with 500 iterations, yielding 2,500 potential reference frames. Since we are running single-threaded here in Windows, we only run a single experiment with 100 iterations (~5 minutes to run).

The 'min.cnt' and 'min.pct' parameters are critical for determining the candidate ASVs for the reference frame, where 'min.cnt' is the minimum read count for an ASV to be considered detected in a sample, and 'min.pct' is the minimum % of samples an ASV must be detected in to be considered for the reference frame. Here, we use 5 and 1 (100%) as used in the study. The 'smin' and 'smax' parameters are the minimum and maximum fraction of candidate ASVs that are selected in the MCMC initial step, respectively. We set smin=0.1 and smax=0.9 as in the study.

```
# Parameters for reference frame MCMC search
mcmc_iterations <- 100 # Number of MCMC iterations to run
smin <- 0.1 # Minimum fraction of candidate ASVs to initially select in each MCMC iteration
smax <- 0.9 # Maximum fraction of candidate ASVs to initially select in each MCMC iteration
min.cnt <- 5 # Minimum count for an ASV to be considered detected in a sample
min.pct <- 1 # Minimum % of samples an ASV must be detected in (100%)
seed <- 147 # For reproducibility (default seed is 100)
P <- 1 # pseudocount

# Grab counts from the Supplementary Data
cnts <- sdata$counts_agg_filtered %>% column_to_rownames("ASV")
dim(cnts)
```

```
## [1] 1434   72
```

```
# Run MCMC search
outfile <- "invariantsets.rds"
if (!file.exists(outfile)) {
  timestamp()
  # Pre-compute log-ratio covariance matrix
  message("Computing log-ratio covariance matrix")
  lrvars <- computeLRcovar(cnts, P)
  timestamp()
  message("Running reference frame search")
    InvariantSetMC(cnts, lrvars = lrvars, min.pct = min.pct, min.cnt = min.cnt, smin = smin,
                   smax = smax, mc.it = mcmc_iterations, p = P, seed = seed) %>%
    saveRDS(outfile)
    timestamp()
}
minima.all <- readRDS(outfile)
```

Now let's investigate the reference frames from our MCMC search and choose the optimal reference frame found.

```
# Let's retrieve the reference frame with the smallest log-ratio variance
ref_frame <- lapply(minima.all, function(m) tibble(ASVs = paste0(m$ASVs, collapse = ";"), var = m$var)) %>%
  bind_rows %>% arrange(var) %>% dplyr::slice(1)
ref_asvs <- strsplit(ref_frame$ASVs, split = ";")[[1]]
show(ref_asvs)
```

```
##  [1] "ASV132"  "ASV196"  "ASV159"  "ASV49"   "ASV560"  "ASV15"   "ASV98"
```

```
##  [8] "ASV609"  "ASV408"  "ASV297"  "ASV116"  "ASV465"  "ASV58"   "ASV469"
## [15] "ASV642"  "ASV532"  "ASV106"  "ASV350"  "ASV62"   "ASV119"  "ASV601"
## [22] "ASV610"  "ASV140"  "ASV60"   "ASV43"   "ASV189"  "ASV548"  "ASV101"
## [29] "ASV451"  "ASV334"  "ASV4"    "ASV356"  "ASV395"  "ASV13"   "ASV66"
## [36] "ASV595"  "ASV67"   "ASV1251" "ASV186"  "ASV85"   "ASV170"  "ASV16"
## [43] "ASV205"  "ASV220"  "ASV71"   "ASV256"  "ASV10"   "ASV96"   "ASV18"
## [50] "ASV17"   "ASV599"  "ASV416"  "ASV793"  "ASV244"  "ASV52"   "ASV68"
## [57] "ASV210"  "ASV305"  "ASV21"   "ASV413"  "ASV48"   "ASV455"  "ASV188"
## [64] "ASV859"  "ASV351"  "ASV260"  "ASV384"  "ASV744"  "ASV171"  "ASV202"
## [71] "ASV453"  "ASV699"  "ASV54"   "ASV174"  "ASV80"   "ASV46"   "ASV41"
## [78] "ASV214"  "ASV293"  "ASV515"  "ASV1164" "ASV464"  "ASV316"  "ASV456"
## [85] "ASV262"  "ASV558"  "ASV968"  "ASV689"  "ASV692"
```

```
show(ref_frame$var)
```

```
## [1] 0.0008945651
```

```
# Compute reference frame standardized ASV counts
cnts_ref <- computeInvariantBal(cnts, invariantASVs = ref_asvs, p = P)
```

```
## Zero values detected. Adding user-defined pseudocount of 1
```

```
## Numerator ASVs not specified. Computing balance values for all non-invariant ASVs
```

```
# Double check the reference frame variance
var(rowMeans(cnts_ref))
```

```
## [1] 0.0008945651
```

```
# Compare this reference frame to the one identified in the study
ref_asvs_study <- sdata$ReferenceFrame %>% pull(ASV)
intersect(ref_asvs, ref_asvs_study) %>% length
```

```
## [1] 50
```

```
# Check the study reference frame variance
cnts_ref_study <- computeInvariantBal(cnts, invariantASVs = ref_asvs_study, p = P)
```

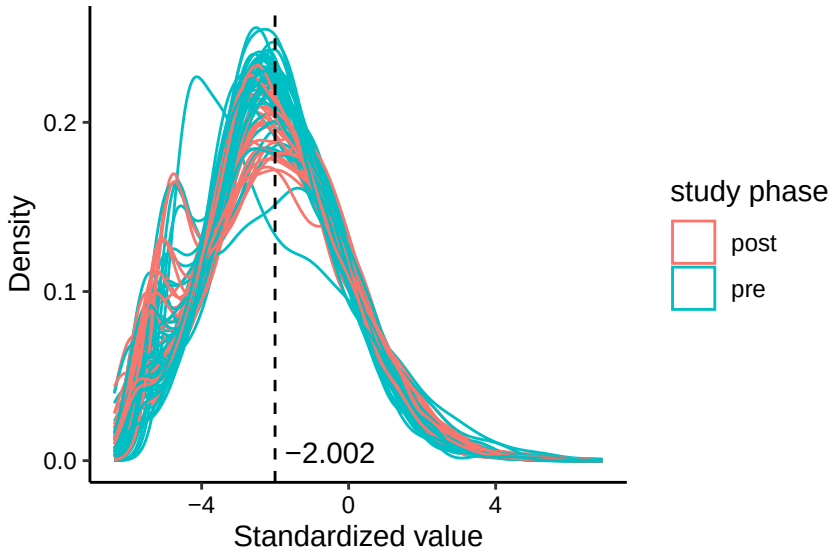
```
## Zero values detected. Adding user-defined pseudocount of 1
## Numerator ASVs not specified. Computing balance values for all non-invariant ASVs
```

```
var(rowMeans(cnts_ref_study))
```

```
## [1] 0.0007709808
```

This reference frame has 72 ASVs, 50 of which are shared with the 69 ASV reference frame identified in the study. Since we used fewer total MCMC iterations (100 vs 2,500), the reference frame variance happens to be higher here (0.000895) than in the study (0.000771). These numbers don't mean much, and even though our variance here is a bit higher, this 72 ASV reference frame is still suitable. Let's confirm the centering of the sample reference standardized count data.

```r
# Density plot of sample counts
centered_value <- mean(rowMeans(cnts_ref))
rownames_to_column(cnts_ref, "sampleid") %>%
  mutate(`study phase` = sub("^.*_", "", sampleid)) %>% arrange(`study phase`) %>%
  pivot_longer(starts_with("ASV"), names_to = "ASV", values_to = "cnt") %>%
  ggplot(aes(x = cnt, color = `study phase`, group = sampleid)) + geom_density() +
  theme(panel.background = element_blank(),
        axis.line = element_line(color = "black")) +
    theme(axis.text = element_text(color = "black"),
          axis.title = element_text(color = "black")) +
    theme(legend.key = element_blank(), legend.position = "right") +
  geom_vline(xintercept = centered_value, linetype = "dashed") +
  xlab("Standardized value") + ylab("Density") +
  annotate("text", x = centered_value + 1.5, y = 0.005,
           label = as.character(round(centered_value, 3)))
```



We observe similar centering of the count data (-2.002) as in the study (**Figure 5**, -2.05). This short example of reference frame identification shows that deeper MCMC searches do not guarantee significantly better reference frames. It is up to the user to decide how many MCMC iterations is needed to produce a reference frame that sufficiently centers the count data. As we explore in the study (**SFigure 2**), deeper MCMC searches may offer opportunities to evaluate aspects of core, shared microbes in the dataset.

Part 2: Pairbal feature reduction

The Pairbal method aims to enrich signal dataset through dimensional reduction and is designed to be used in conjunction with cross-validation machine learning approaches. The approach taken in the study uses 5 iterations of 15-fold cross-validation (CV). Each CV fold is analyzed as follows: 1. Optionally, identify reference frame using training data count matrix, yielding reference standardized count matrix. Otherwise, use a different log-ratio standardization approach, e.g. centered or isometric log-ratios. 2. Dimensionally reduce training data count matrix using Pairbal. Do not include reference ASVs in pairbal search, if applicable. 3. Build random forests using dimensionally reduced, reference standardized training data 4. Predict vaccination group or protection outcome in the test data

Below is an example of Pairbal feature reduction using only pre-challenge animals in vaccination groups O and S, where we are interested in protection outcome.

```
# Grab O and S sampleids from the metadata
meta <- sdata$metadata_agg %>%
  filter(vaccination_group %in% c("O", "S") & study_phase == "pre-challenge") %>%
  column_to_rownames("sampleid")

# Compute all ASV pairwise balances
balances <- computePWB(cnts[,rownames(meta)])
```

```
## Zero values detected and pseudocount not provided. Defaulting to pseudocount of 1.
```

```
nrow(balances) # 1,027,461 pairwise balances
```

```
## [1] 1027461
```

```
# Filter these pairwise balances using wilcoxon-rank sum tests
balances_filt <- filterPWB(balances, meta, bivar = "challenge_outcome",
                           group.oi = "Protected", min.abs.lfc = 1,
                           p.cutoff = 0.05, test.type = "wilcox")
```

```
## Contrast successfully parsed: 17 Protected vs. 13 NotProtected
```

```
## 103534 of 1027461 balances remain after applying minimum fold-change cutoff of 1
```

```
## 29106 of 103534 balances remain after applying p-value cutoff of 0.05
```

```
# Greedily prune these pairwise balances, retaining the most frequent ASVs
# which are likely driving the changes in the pairwise balances
balances_top <- getTopFeatures(balances_filt)
nrow(balances_top) # 277 ASVs
```

```
## [1] 277
```

In this example, we reduce the data set from 1,434 ASVs down to just 277 ASVs.

Concluding remarks

Reference frames are an alternative log-ratio standardization approach to centered and isometric log-ratios. Additionally, they can provide insights into core microbial communities that might be shared across samples regardless of phenotype or other covariates. Pairbal is a second compositional analysis technique that when used in conjunction with CV machine learning analyses, may augment model performance. As a feature reduction approach, Pairbal might be especially useful for data sets with significantly more ASVs than samples. For now, Pairbal can only be used with binary covariates.

```
sessionInfo()
```

```
## R version 4.3.2 (2023-10-31 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 11 x64 (build 22621)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=English_United States.utf8
## [2] LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.utf8
##
## time zone: America/Los_Angeles
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] lubridate_1.9.3 forcats_1.0.0   stringr_1.5.1   dplyr_1.1.4
##  [5] purrr_1.0.2     readr_2.1.5     tidyr_1.3.1     tibble_3.2.1
##  [9] ggplot2_3.5.0   tidyverse_2.0.0 readxl_1.4.3
##
## loaded via a namespace (and not attached):
##  [1] gtable_0.3.4      compiler_4.3.2    tidyselect_1.2.0  scales_1.3.0
##  [5] yaml_2.3.8        fastmap_1.1.1     R6_2.5.1          labeling_0.4.3
##  [9] generics_0.1.3    knitr_1.45        munsell_0.5.0     pillar_1.9.0
## [13] tzdb_0.4.0        rlang_1.1.3       utf8_1.2.4        stringi_1.8.3
## [17] xfun_0.42         timechange_0.3.0  cli_3.6.2         withr_3.0.0
## [21] magrittr_2.0.3    digest_0.6.34     grid_4.3.2        rstudioapi_0.15.0
## [25] hms_1.1.3         lifecycle_1.0.4   vctrs_0.6.5       evaluate_0.23
## [29] glue_1.7.0        farver_2.1.1      cellranger_1.1.0  fansi_1.0.6
## [33] colorspace_2.1-0  rmarkdown_2.25    tools_4.3.2       pkgconfig_2.0.3
## [37] htmltools_0.5.7
```