`microbiomeDASim`: Tools for Simulationg Longitudinal Differential Abundance

Justin Williams, Hector Corrada Bravo, Jennifer Tom, Joseph Nathaniel Paulson

Modified: September 9, 2019. Compiled: August 29, 2024

Contents

1 Introduction

With an increasing emphasis on understanding the dynamics of microbial communities in various settings, longitudinal sampling studies are underway. Whole metagenomic shotgun sequencing and marker-gene survey studies have unique technical artifacts that drive novel statistical methodological development for estimating time intervals of differential abundance. In designing a study and the frequency of collection prior to a study, one may wish to model the ability to detect an effect, e.g., there may be issues with respect to cost, ease of access, etc. Additionally, while every study is unique, it is possible that in certain scenarios one statistical framework may be more appropriate than another.

Here, we present a simulation paradigm in a R software package termed `microbiomeDASim`. This package allows investigators to simulate longitudinal differential abundance for single features, `mvrnorm_sim`, or a community with multiple features, `gen_norm_microbiome`. The functions allow the user to specify a variety of known functional forms (e.g, linear, quadratic, oscillating, hockey stick) along with flexible parameters to control desired signal to noise ratio. Different longitudinal correlation structures are available including AR(1), compound, and independent to account for expected within individual correlation. Comparing estimation methods or designing a potential longidutinal investigation for microbiome sequencing data using `microbiomeDASim` provides accurate and reproducible results.

2 Installation

To install `microbiomeDASim` from *Bioconductor* use the code below:

```
if(!requireNamespace("BiocManager", quietly = TRUE)){
    install.packages("BiocManager")
}
BiocManager::install("microbiomeDASim")
```

3 Motivation

In our analyses we want to try and simulate longitudinal differential abundance for microbiome normalized counts generated from 16S rRNA sequencing in order to compare different methodologies for estimating this differential abundance. Simulating microbiome data presents a variety of different challenges based both on biological and technical challenges.

These challenges include:

- Non-negative restriction

- Presence of Missing Data/High Number of Zero Reads
- Low Number of Repeated Measurements
- Small Number of Subjects

For our initial simulation design, we are attempting to address many of these challanges by simulating data across a variety of different parameter scenarios where each element of the challenges listed above can be investigated. Since many of the methods when analyzing microbiome data involve normalization procedures and the central limit theory allows us to think about the normal distribution as an asymptotic ideal, we will treat the Multivariate Normal Distribution as our best case scenario for simulation purposes.

4 Statistical Methodology

Assume that we have data generated from the following distribution,

$$\mathbf{Y} \sim N(\mu, \mathbf{\Sigma})$$

where

$$\mathbf{Y} = \begin{pmatrix} \mathbf{Y}_1^T \\ \mathbf{Y}_2^T \\ \vdots \\ \mathbf{Y}_n^T \end{pmatrix} = \begin{pmatrix} Y_{11} \\ Y_{12} \\ \vdots \\ Y_{1q_1} \\ Y_{21} \\ \vdots \\ Y_{2q_2} \\ \vdots \\ Y_{nq_n} \end{pmatrix}$$

with Y_{ij} representing the i^{th} patient at the j^{th} timepoint where each patient has q_i repeated measurements with $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, q_i\}$. We define the total number of observations as $N = \sum_{i=1}^{n} q_i$. Therefore in our original assumption defined above we can explicitly define the dimension of our objects as

$$\mathbf{Y}_{N\times 1} \sim N_N(\mu_{N\times 1}, \mathbf{\Sigma}_{N\times N})$$

In our current simulations we choose to keep the number of repeated measurements constant, i.e., $q_i = q \; \forall \; i \in \{1, \ldots, n\}$. This means that the total number of obseravtions simplifies to the expression $N = qn$. However, we will vary the value of q across the simulations to simulate data generated

from a study with a small ($q = 3$), medium ($q = 6$), or large ($q = 12$) number of repeated observations. Currently most studies with microbiome data collected fall closest to the small case, but there are a few publically available datasets that contain a large number of repeated observations for each individual. For simplicity in the remaining description of the methodology we will assume constant number of repeated observations.

Without loss of generality we can split the total patients (n) into two groups, control (n_0) vs (n_1), with the first n_0 patients representing the control patients and the remaining $n-n_0$ representing the treatment. Partitioning our observations in this way allows us to also partition the mean vector as $\mu = (\mu_0, \mu_1)$. In our current simulation shown below we assume that the mean for the control group is constant $\mu_0 \mathbf{1}_{n_0 \times 1}$, but we allow our mean vector for the treatment group to vary depending on time $\mu_{1ij}(t) = \mu_0 + f(t_j)$ for $i = 1, \ldots, n_1$ and $j = 1, \ldots, q$. In our simulation we will choose a parametric form for the function $f(t_j)$ primarly from the polynomial family where

$$f(t_j) = \beta_0 + \beta_1 t_j + \beta_2 t_j^2 + \cdots + \beta_p t_j^p$$

for a p dimensional polynomial. For instance, to define a linear polynomial we would have

$$f(t_j) = \beta_0 + \beta_1 t_j$$

where $\beta_1 > 0$ for an increasing linear trend and $\beta_1 < 0$ for a decreasing linear trend. For a list of available functional forms currently implemented and the expected form of the input see documentation for 'microbiomeDASim::mean_trend'.

For the covariance matrix, we want to encode our longitudinal dependencies so that observations within an individual are correlated, i.e., $\text{Cor}(Y_{ij}, Y_{ij'}) \neq 0$, but that observations between individuals are independent, i.e., $\text{Cor}(Y_{ij}, Y_{i'j}) = 0 \ \forall i \neq i'$ and j. To accomplish this we define the matrix $\mathbf{\Sigma}_{N \times N}$ as $\mathbf{\Sigma} = \text{bdiag}(\mathbf{\Sigma}_1, \ldots, \mathbf{\Sigma}_n)$, where each $\mathbf{\Sigma}_i$ is a $q \times q$ matrix a specific longitudinal structure.

For instance if we want to specify an autoregressive correlation structure for individual i we could define their covariance matrix as

$$\mathbf{\Sigma}_i = \sigma^2 \begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{|1-q|} \\ \rho & 1 & \rho & \cdots & \rho^{|2-q|} \\ \rho^2 & \rho & 1 & \cdots & \\ \vdots & & & \ddots & \vdots \\ \rho^{|q-1|} & \rho^{|q-2|} & \ldots & \ldots & 1 \end{bmatrix}$$

In this case we are using the first order autoregressive definition and therefore will refer to this as "ar1".

Alternatively, for the compound correlation structure for an individual i' we could define the covariance matrix as

$$\mathbf{\Sigma}_{i'} = \sigma^2 \begin{bmatrix} 1 & \rho & \rho & \cdots & \rho \\ \rho & 1 & \rho & \cdots & \rho \\ \rho & \rho & 1 & \cdots & \\ \vdots & & & \ddots & \vdots \\ \rho & \rho & \cdots & \cdots & 1 \end{bmatrix}$$

Finally, the independent correlation structure for an individual i'' would be defined as

$$\mathbf{\Sigma}_{i''} = \sigma^2 \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & 1 \end{bmatrix}$$

4.1 Trivial Example

We can construct a trivial example where $n = 2$ ($n_0 = 1$ and $n_1 = 1$) and $q = 2$, an ar1 correlation structure where $\rho = 0.8$, $\sigma = 1$, a linear functional form with $\beta = (0, 1)^T$, control mean is constant at $\mu_0 = 2$ and $t = (1, 2)$.

$$Y \sim \mathrm{N}_4 \left(\begin{pmatrix} \mu_0 \\ \mu_0 \\ \mu_0 + \beta_0 + \beta_1 \times t_1 \\ \mu_0 + \beta_0 + \beta_1 \times t_2 \end{pmatrix}, \sigma^2 \begin{pmatrix} 1 & \rho & 0 & 0 \\ \rho & 1 & 0 & 0 \\ 0 & 0 & 1 & \rho \\ 0 & 0 & \rho & 1 \end{pmatrix} \right) = \mathrm{N}_4 \left(\begin{pmatrix} 2 \\ 2 \\ 3 \\ 4 \end{pmatrix}, \begin{pmatrix} 1 & 0.8 & 0 & 0 \\ 0.8 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.8 \\ 0 & 0 & 0.8 & 1 \end{pmatrix} \right)$$

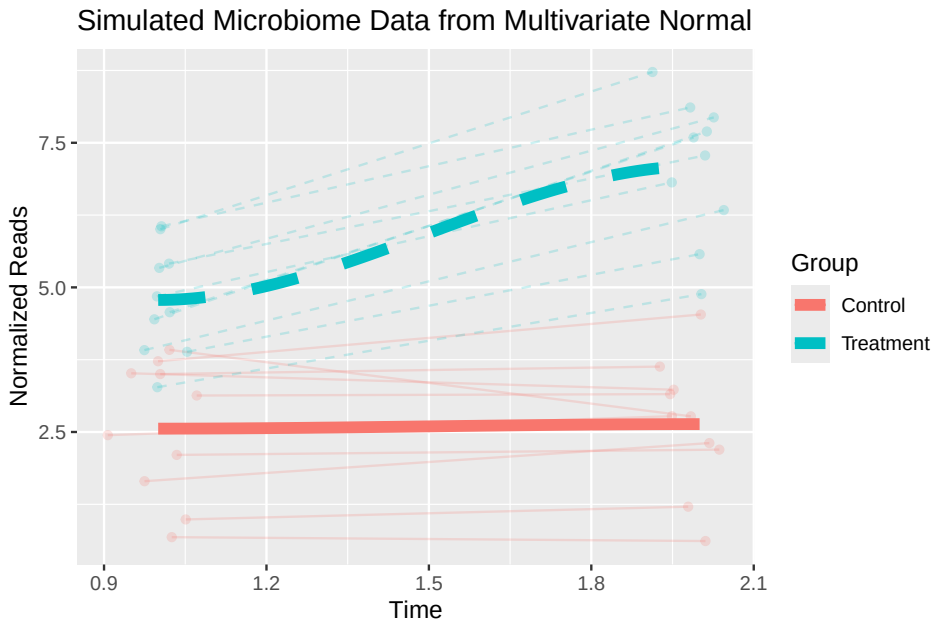
Next, we can now look at sample data generated for our trivial example,

```
require(microbiomeDASim)

## Loading required package:  microbiomeDASim

triv_ex <- mvrnorm_sim(n_control=10, n_treat=10, control_mean=2,
                       sigma=1, num_timepoints=2, t_interval=c(1, 2), rho=0.8,
                       corr_str="ar1", func_form="linear",
                       beta= c(1, 2), missing_pct=0,
                       missing_per_subject=0, dis_plot=TRUE)

## `geom_smooth()` using formula = 'y ~ x'
```




```
head(triv_ex$df)

##           Y ID time   group     Y_obs
## 1 3.7042950  1    1 Control 3.7042950
## 2 4.5416504  1    2 Control 4.5416504
## 3 1.6048665  2    1 Control 1.6048665
## 4 2.3096459  2    2 Control 2.3096459
## 5 0.6608956  3    1 Control 0.6608956
## 6 0.6545231  3    2 Control 0.6545231

triv_ex$Sigma[seq_len(2), seq_len(2)]

## 2 x 2 sparse Matrix of class "dsCMatrix"
##
## [1,] 1.0 0.8
## [2,] 0.8 1.0
```

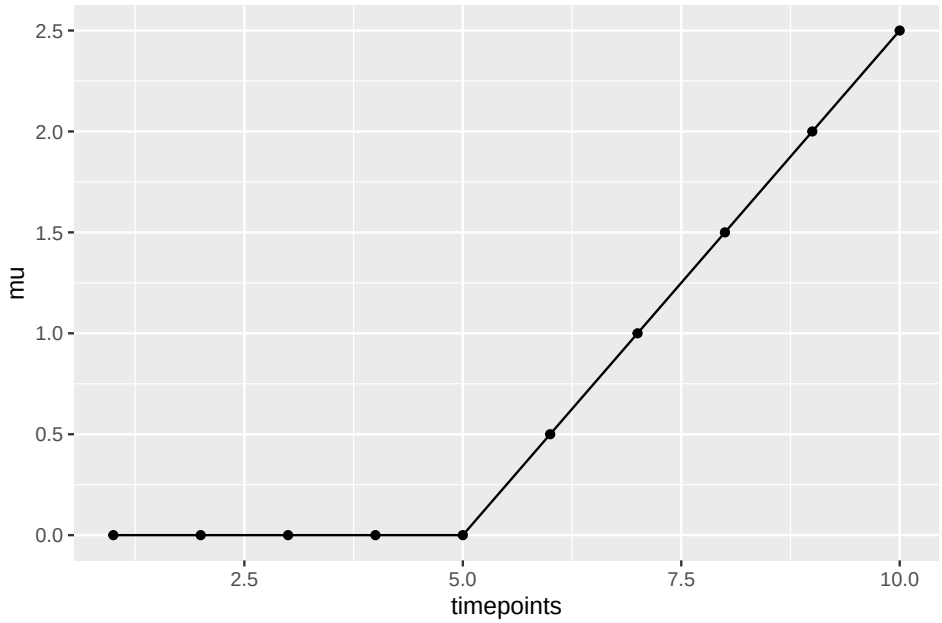
Note that there are a variety of flexible choices for the the functional form of the trend:

- Linear
- Quadratic
- Cubic
- M/W (Oscillating Trends)

- L_up/L_down (Hockey Stick Trends)

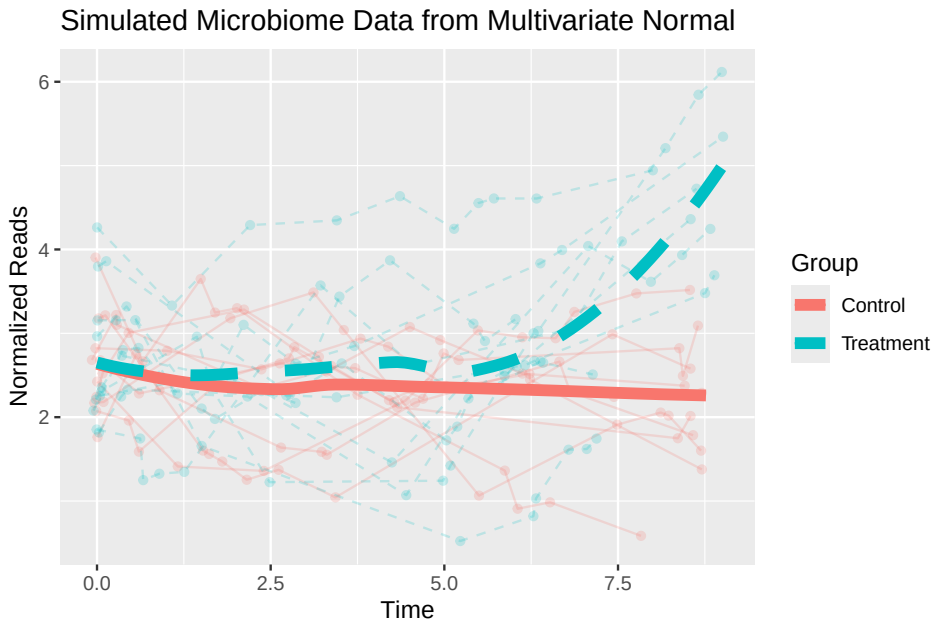
Below we show an example using a Hockey Stick trend where we graph the true functional form, and then simulate data with this trend.

```
true_mean <- mean_trend(timepoints=1:10, form="L_up", beta=0.5, IP=5,
                        plot_trend=TRUE)
```




```
hockey_sim <- mvrnorm_sim(n_control=10, n_treat=10, control_mean=2, sigma=1,
                          num_timepoints=10, t_interval=c(0, 9), rho=0.8,
                          corr_str="ar1", func_form="L_up", beta= 0.5, IP=5,
                          missing_pct=0, missing_per_subject=0, dis_plot=TRUE,
                          asynch_time=TRUE)

## `geom_smooth()` using formula = 'y ~ x'
```



The `mvrnorm_sim` method generates a single feature with a specified longitudinal differential abundance pattern. However, we may want to simulate a microboime environment with multiple features where certain features have differential abundance while others do not.

To address this we can use the function `gen_microbiome`, which specifies thenumber of features and the number of differentiall abundant features. All features selected to have differential abundance will have the same type of functional form.

```
bug_gen <- gen_norm_microbiome(features=6, diff_abun_features=3, n_control=30,
                               n_treat=20, control_mean=2, sigma=2,
                               num_timepoints=4, t_interval=c(0, 3),
                               rho=0.9, corr_str="compound",
                               func_form="M", beta=c(4, 3), IP=c(2, 3.3, 6),
                               missing_pct=0.2, missing_per_subject=2,
                               miss_val=0)

## Simulating Diff Bugs
## Simulating No-Diff Bugs

head(bug_gen$bug_feat)

##           ID time   group Sample_ID
## Sample_1  1    0 Control  Sample_1
## Sample_2  1    1 Control  Sample_2
## Sample_3  1    2 Control  Sample_3
## Sample_4  1    3 Control  Sample_4
```

```
## Sample_5  2    0 Control  Sample_5
## Sample_6  2    1 Control  Sample_6

bug_gen$Y[, 1:5]

##             Sample_1 Sample_2 Sample_3 Sample_4  Sample_5
## Diff_Bug1   2.159108 2.308005 2.600628 0.781598 1.6654818
## Diff_Bug2   3.628834 2.681153 2.260422 2.640305 0.0000000
## Diff_Bug3   2.895703 3.002676 2.621965 3.113923 0.0000000
## NoDiffBug_1 4.093643 5.386791 5.793327 4.721187 0.9717779
## NoDiffBug_2 3.620989 2.590141 3.230859 2.109404 0.3580812
## NoDiffBug_3 0.000000 0.000000 0.000000 0.000000 1.8677757

names(bug_gen)

## [1] "Y"        "bug_feat"
```

Note that we now have two objects returned in this function.

- `Y` is our observed feature matrix with rows representing features and columns indicating our repeated samples
- `bug_feat` identifies the subject ID, time, group (Control vs. Treatment) and corresponding Sample ID from the columns of `Y`.

5 Session Info

```
sessionInfo()

## R version 4.4.1 (2024-06-14)
## Platform: x86_64-pc-linux-gnu
## Running under: Ubuntu 24.04 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.26.so;  LAPA
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
```

```
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## time zone: Etc/UTC
## tzcode source: system (glibc)
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods
## [7] base
##
## other attached packages:
## [1] microbiomeDASim_1.19.0 knitr_1.48
##
## loaded via a namespace (and not attached):
##  [1] ade4_1.7-22             farver_2.1.2
##  [3] Biostrings_2.73.1       bitops_1.0-8
##  [5] phyloseq_1.49.0         digest_0.6.37
##  [7] lifecycle_1.0.4         cluster_2.1.6
##  [9] survival_3.7-0          statmod_1.5.0
## [11] magrittr_2.0.3          compiler_4.4.1
## [13] rlang_1.1.4             tools_4.4.1
## [15] igraph_2.0.3            utf8_1.2.4
## [17] data.table_1.16.0       labeling_0.4.3
## [19] metagenomeSeq_1.47.0    plyr_1.8.9
## [21] RColorBrewer_1.1-3      KernSmooth_2.23-24
## [23] withr_3.0.1             BiocGenerics_0.51.0
## [25] sys_3.4.2               grid_4.4.1
## [27] stats4_4.4.1            fansi_1.0.6
## [29] caTools_1.18.2          multtest_2.61.0
## [31] biomformat_1.33.0       colorspace_2.1-1
## [33] Rhdf5lib_1.27.0         ggplot2_3.5.1
## [35] scales_1.3.0            gtools_3.9.5
## [37] iterators_1.0.14        MASS_7.3-61
## [39] tmvtnorm_1.6            cli_3.6.3
## [41] mvtnorm_1.2-6           vegan_2.6-8
## [43] crayon_1.5.3            httr_1.4.7
## [45] reshape2_1.4.4          pbapply_1.7-2
## [47] ape_5.8                 rhdf5_2.49.0
## [49] stringr_1.5.1           zlibbioc_1.51.1
## [51] splines_4.4.1           parallel_4.4.1
## [53] XVector_0.45.0          matrixStats_1.3.0
## [55] vctrs_0.6.5             sandwich_3.1-0
## [57] glmnet_4.1-8            Matrix_1.7-0
## [59] jsonlite_1.8.8          IRanges_2.39.2
```

```
## [61] S4Vectors_0.43.2        maketools_1.3.0
## [63] locfit_1.5-9.10         foreach_1.5.2
## [65] limma_3.61.9            glue_1.7.0
## [67] codetools_0.2-20        gmm_1.8
## [69] stringi_1.8.4           shape_1.4.6.1
## [71] gtable_0.3.5            GenomeInfoDb_1.41.1
## [73] UCSC.utils_1.1.0        munsell_0.5.1
## [75] tibble_3.2.1            pillar_1.9.0
## [77] gplots_3.1.3.1          rhdf5filters_1.17.0
## [79] GenomeInfoDbData_1.2.12 R6_2.5.1
## [81] evaluate_0.24.0         lattice_0.22-6
## [83] Biobase_2.65.1          highr_0.11
## [85] Rcpp_1.0.13             Wrench_1.23.0
## [87] permute_0.9-7           nlme_3.1-166
## [89] mgcv_1.9-1              xfun_0.47
## [91] zoo_1.8-12              buildtools_1.0.0
## [93] pkgconfig_2.0.3
```