SOFTWARE TOOL ARTICLE

**microbiomeDASim: Simulating longitudinal differential abundance for microbiome data** [version 1; peer review: 1 approved, 1 approved with reservations]

Justin Williams 1,2, Hector Corrada Bravo3, Jennifer Tom1*, Joseph Nathaniel Paulson1*

1Department of Biostatistics, Genentech, Inc, South San Francisco, CA, 94080, USA
2Department of Biostatistics, University of California, Los Angeles, Los Angeles, CA, 90095, USA
3Department of Computer Science, University of Maryland, College Park, College Park, MD, 24072, USA

* Equal contributors

**Abstract**

An increasing emphasis on understanding the dynamics of microbial communities in various settings has led to the proliferation of longitudinal metagenomic sampling studies. Data from whole metagenomic shotgun sequencing and marker-gene survey studies have characteristics 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 implemented in the R Bioconductor software package microbiomeDASim available at http://bioconductor.org/packages/microbiomeDASim. microbiomeDASim allows investigators to simulate longitudinal differential abundant microbiome features with a variety of known functional forms with flexible parameters to control desired signal-to-noise ratio. We present metrics of success results on one particular method called metaSplines.

**Keywords**

Microbiome, Differential Abundance, Longitudinal, R, Bioconductor

This article is included in the Bioconductor gateway.
Introduction

Analysis of the microbiome aims to characterize the composition and functional potential of microbes in a particular ecosystem. Recent studies have shown the gut microbiome plays an important role in various diseases, from the efficacy of cancer immunotherapy to the pathogenesis of inflammatory bowel disease (IBD)\(^1\)–\(^4\). While many studies profile static community “snapshots”, microbial communities do not exist within an equilibrium\(^5\). To better understand bacterial population dynamics, many studies are expanding to longitudinal sampling and foregoing cross-sectional or single time-point explorations. With a decrease in sequencing costs, more longitudinal data will be generated for varying communities of interest. While data generation will present fewer difficulties, there remain several statistical challenges involved in analyzing these datasets.

The common approach in the marker-gene survey literature is to perform pairwise differential abundance tests between specific time points and visually confirm, sometimes using smoothing methods like splines, how differences are manifested across time\(^6\). These methods require that analysts provide one or more specific time points to test, and the statistical inferences derived from these procedures are specific to these pairwise tests. Other standard methods for longitudinal analysis test for global differences across time, sometimes using non-linear methods including splines to capture dynamic profiles across time\(^7\). Incorporating confounding sources of variability, both biological and technical is essential in high-throughput studies\(^8\) and require statistical methods capable of estimating both smooth functions and sample-specific characteristics.

Simulating marker-gene amplicon sequencing data presents a variety of challenges related to biological and technical limitations when collecting data. We present a framework for simulating data that can be used across multiple methods for estimating longitudinal differential abundance. This simulation framework allows for appropriate comparison between methods while taking into account some of the unique challenges for the marker-gene amplicon sequencing data, including the following:

1. Non-negative restriction
2. Presence of Missing Data/High Number of Zero Reads
3. Low Number of Repeated Measurements
4. Asynchronous Repeated Measures
5. Small Number of Subjects

The first two challenges described above are related to the data generating process itself while the following three represent logistical challenges often faced when collecting the data. In microbiomeDASim, we attempt to address these data generating challenges through specific simulation mechanisms described in the Microbiome adaptions section. Similarly, logistical challenges are addressed by allowing users to specify these values flexibly and investigate the corresponding effects, tailoring the simulation to an appropriate setting.

This package allows investigators to simulate longitudinal differential abundant microbiome features with a variety of known functional forms along with flexible parameters to control design aspects such as signal to noise ratio, correlation structure, and effect size. We highlight the application of a simulation design using one particular method, metaSplines\(^9\).

Methods

Distributional assumptions

Sequencing data are often non-normal. However, transformations, such as log(·) or arcsinh(·), are often applied to raw marker-gene amplicon sequencing data so that the subsequent data is approximately normally distributed. As such, we generate simulated data from a multivariate normal distribution. Using a multivariate normal is a natural choice in this setting as longitudinal correlation structure can be easily incorporated. The following methods focus on cases where the desired microbiome features following appropriate transformation are approximately normally distributed.

Assume that we have data generated from the following distribution,

\[ Y \sim N(\mu, \Sigma), \]
where

\[
Y = \begin{bmatrix}
Y_1^T \\
Y_2^T \\
\vdots \\
Y_n^T
\end{bmatrix},
\]

with \(Y_i\) representing the \(i^{th}\) individual at the \(j^{th}\) time point and each individual 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 \(N = \sum_i q_i\). While this model holds for different choices of \(q_i\), throughout this article we will assume, without loss of generality, that the number of repeated measurements is constant, i.e., \(q_i = q \forall i \in \{1, \ldots, n\}\). This means that the total number of observations simplifies to the expression \(N = nq\). Similarly, we split the total patients (\(n\)) into two groups, control (\(n_0\)) and treatment (\(n_1\)), with the first \(n_0\) patients representing the control patients and the remaining \(n-n_0\) representing the treatment patients. Subsequently we define the total number of observations in each group as \(N_0 = n_0 \cdot q\) and \(N_1 = n_1 \cdot q\) respectively.

**Mean components**

Partitioning our observations into control and treatment groups in this way allows us to define the mean vector separately for each group as \(\mu = (\mu_0, \mu_1)\) where \(\mu_0\) is an \(N_0 \times 1\) vector and \(\mu_1\) is an \(N_1 \times 1\) vector. To generate differential abundance the mean for the control group is held constant \(\mu_0\), but allow the mean vector for the treatment group to vary as a function of time \(\mu_1(t) = \mu_0 + f(t)\) for \(i = 1, \ldots, n_1\) and \(j = 1, \ldots, q\). The form of \(f(t)\) will dictate the functional form of the differential abundance. Note that if \(f(t) = 0\), then both groups have equal mean at baseline.

**Polynomial functional forms**

We allow \(f(t)\) to be specified using polynomial basis as

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

for a \(p\) dimensional polynomial. We restrict the allowed polynomials to be either linear, \(p=1\), quadratic, \(p=2\), or cubic, \(p=3\). For instance, to define a quadratic polynomial one would specify \(\beta = (\beta_0, \beta_1, \beta_2)\) in the following equation,

\[
f(t_j) = \beta_0 + \beta_1 t_j + \beta_2 t_j^2.
\]

Again, it is important to note that if \(\beta = \mathbf{0}\), that the treatment group is assumed to have no differentially abundant timepoints. Typically to simulate no differential abundance, a linear trend is chosen with \(\beta_0 = \beta_1 = 0\).

**Oscillating functional forms**

While polynomial functions are often natural choices for longitudinal trends, interest also lies in exploring other non-smooth, i.e., non-differentiable, types of trends. One such form we refer to as oscillating functional forms. These trends include types that transition from linearly increasing to linear decreasing at a point, or vice versa from linearly decreasing to linear increasing. One of the most well known trends of this type is the absolute value function. To allow for flexible choices in oscillating type trends, we allow for these non differentiable linearly connected trends to repeat forming what we call M and W trends. From a biological perspective we could think of these trends as representing spikes in a particular feature that may occur immediately after
a treatment dose is given, but then decays rapidly to baseline levels followed by a similar spike and decay upon repeated dosing. These functional trends are operationalized as

$$f(t_i) = \beta_i \mathbb{1}(t_i < \text{IP}_1) t_i + \left(\beta_0 + \beta_1 \text{IP}_1\right) t_i + \left(\beta_2 + \beta_3 \text{IP}_1\right) t_i \left(\text{IP}_1 \leq t_i < \text{IP}_2\right) + \left(\beta_4 + \beta_5 \text{IP}_1\right) t_i \left(\text{IP}_2 \leq t_i < \text{IP}_3\right) + \left(\beta_6 + \beta_7 \text{IP}_1\right) t_i \left(t_i \geq \text{IP}_3\right)$$

where IP$_k$ for $k = 1, 2, 3$ denotes an inflection point where the linear trend changes from increasing to decreasing or vice versa. Note that for these types of trends that the sign of $\beta_i$ determines whether the trend is initially increasing, i.e. M, ($\beta_i > 0$) or initially decreasing, i.e. W, ($\beta_i < 0$). By construction, we force the trend line to be exactly zero at IP$_k$ and by doing so the trend is specified completely as $\beta = (\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7)^T$ and $\text{IP} = (\text{IP}_1, \text{IP}_2, \text{IP}_3)^T$. An implicit restriction on the functional trend is that $\text{IP}_1 \neq t_i$. However, we can construct absolute value and inverted absolute value type trends by defining $\text{IP}_1 \in (t_i, t_j)$ and $\text{IP}_2, \text{IP}_3 > t_i$. Again, the key difference for these set of trends is that the inflection points create non-smooth trends.

Hockey stick functional forms

An additional extension to linear functional trends is the family of Hockey Stick functional forms. There are two available families of hockey stick functional forms, which are referred to as L$_{up}$ and L$_{down}$ within the package. Both of these trends are designed to create two mutually exclusive regions over the time frame specified. These two regions are defined as $\mathcal{R}_1 = (t_i, \text{IP})$ and $\mathcal{R}_2 = (\text{IP}, t_j)$ where one of the regions $\mathcal{R}_1$ or $\mathcal{R}_2$ has linear differential abundance while the other has no differential abundance and IP denotes the inflection point. In the case of the L$_{up}$ trend, $\mathcal{R}_1$ is defined as the non-differentially abundant region and $\mathcal{R}_2$ is a linearly increasing region. We can define the functional form as

$$f(t_i) = (-\beta_i \times \text{IP}) \mathbb{1}(t_i \geq \text{IP}) + \beta_i \mathbb{1}(t_i \geq \text{IP}) t_i$$

Note that with this specification that we do not specify the intercept $\beta_0$ and instead only need to specify the slope term $\beta_i$ and the appropriate point of change. We restrict the slope term to be positive, i.e., $\beta_i \in (0, \infty)$ to create the “up” trend.

Conversely, the L$_{down}$ trend assumes that $\mathcal{R}_1$ is a differentially abundant region that begins with the treatment group higher than the control group and then linearly decreases to the region $\mathcal{R}_2$ where there is no differential abundance. We define this functional form as

$$f(t_i) = \beta_i t_i \left(t_i < \frac{-\beta_0}{\beta_i}\right) + \beta_i t_i \left(t_i < \frac{-\beta_0}{\beta_i}\right) t_i$$

Note that in this case we do not specify the point of change directly, but rather it is implicitly implied by the choice of $\beta_0$ and $\beta_i$, i.e. IP = $-\beta_0/\beta_i$. To ensure that the trend in $\mathcal{R}_1$ is properly specified, we place additional restrictions on the parameters so that $\beta_0 \in (0, \infty)$ and $\beta_i \in (-\infty, 0)$ to ensure the trend is decreasing and check that the choice of $\beta_0$ and $\beta_i$ are appropriately defined so that IP $\in (t_i, t_j)$.

Example trends are shown in Figure 1 generated using the mean_trend function.

Covariance components

As discussed in the Introduction, the multivariate normal is a natural choice for longitudinal simulation due to the ease with which dependency of repeated measures is specified. To encode this longitudinal dependency observations within an individual are assumed to be correlated, i.e. $\text{Cor}(Y_{ij}, Y_{ij'}) \neq 0 \forall j \neq j'$ and $i \in \{1, \ldots, n\}$, but observations between individuals are assumed independent, i.e. $\text{Cor}(Y_{ij}, Y_{i'j}) = 0 \forall i \neq i'$ and $j \in \{1, \ldots, q_j\}$. To accomplish this we define the block diagonal matrix $\Sigma$ as $\Sigma = \text{bdiag}(\Sigma_j, \ldots, \Sigma_j)$, where each $\Sigma_j$ is a $q \times q$ covariance matrix for individual $i$ and $\text{bdiag}()$ indicates that the matrix is block diagonal with all off diagonal elements not in $\Sigma_j$ equal to zero. For each individuals covariance matrix, we assume a global standard deviation parameter and correlation component $\rho$, i.e. $\Sigma_j = \sigma \Omega_j \rho$. 

Page 5 of 21
For instance, if we want to specify an autoregressive correlation structure for individual \( i \) the covariance matrix is defined as

\[
\Sigma_i = \sigma^2 \begin{bmatrix}
1 & \rho & \rho^2 & \ldots & \rho^{k-1} \\
\rho & 1 & \rho & \ldots & \rho^{k-1} \\
\rho^2 & \rho & 1 & \ldots & \rho^{k-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\rho^{k-1} & \rho^{k-1} & \ldots & 1
\end{bmatrix}
\]

In this case we are using the first order autoregressive definition and therefore will refer to this as AR(1).

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

\[
\Sigma_{i'} = \sigma^2 \begin{bmatrix}
1 & \rho & \rho & \ldots & \rho \\
\rho & 1 & \rho & \ldots & \rho \\
\rho & \rho & 1 & \ldots & \rho \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\rho & \rho & \rho & \ldots & 1
\end{bmatrix}
\]
Finally, we allow the user to specify an independent correlation structure for an individual \(i''\), which assumes that repeated observations are in fact uncorrelated and is defined as

\[
\Sigma_i = \sigma^2 \begin{bmatrix}
1 & 0 & 0 & \cdots & 0 \\
0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & : \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & \cdots & \cdots & 1
\end{bmatrix}
\]

Each of these correlation structures are referred as AR(1), compound, and independent respectively.

Microbiome adaptions

As discussed in the Introduction, simulating microbiome data presents a variety of unique challenges. In particular there are two data generating restrictions, 1. non-negative restriction and 2. presence of missing data/high number of zero reads, that must be addressed when simulating this data. In this section we will outline some of the specific adaptions of the simulation framework designed to address these issues.

1. Non-negative restriction. One of the most relevant challenges faced with microbiome data, is the restriction of the domain to non-negative values. To assure that the simulated normalized counts are non-negative, one solution is to simply replace the multivariate normal distribution with a multivariate truncated normal distribution. The new data generating distribution is now

\[
Y \sim \text{TN}(\mu, \Sigma, a 1_N),
\]

where TN indicates the multivariate truncated normal distribution and \(a\) is the left-truncation value. To impose zero truncation it is assumed that \(a = 0\). Values from the multivariate truncated normal are drawn using the package `tmvtnorm`. Note that the default method for drawing observations from this distribution is rejection sampling which proceeds by first drawing from a multivariate normal and then for all values that fall below \(a\) to reject the observed sample and re-sample. This procedure works well when the majority of the distribution falls above the truncation point, but can be computational intensive when the probability of acceptance, \(p_{acpt} = P(Y > a1_N)\), is low. In our simulation design if the value of \(\mu\) is sufficiently close to \(a\) then rejection sampling is not feasible. In the case there the \(p_{acpt} \leq 0.1\), non-negative restriction is imposed by censoring negative values and using point imputation with the truncation value \(a\) as shown below

\[
Y^* \sim N(\mu, \Sigma),
\]

\[
Y^*_y = \begin{cases} 
Y^*_y & \text{if } Y^*_y \geq 0, \\
0 & \text{if } Y^*_y < 0.
\end{cases}
\]

To remove the non-negative restriction there is an option in the function `mvnorm_sim` which can be used to turn-off the domain restriction, but by default the zero truncation is imposed. Note that an alternative option to using the multivariate truncated normal is to use the Johnson translation system which can allow samples to be drawn from a multivariate log normal distribution via an appropriate translation function. The current implementation uses only the multivariate truncated normal distribution for drawing samples via the `zero_trunc` option within the `mvnorm_sim()` and `gen_norm_microbiome()` functions.

2. Presence of missing data/high number of zero reads. The second major data generating challenge when simulating microbiome data is the presence of missing data along with a high percentage of features with zero counts. Based on technical limitations when amplifying and sequencing microbiome data, certain features may be present but remain undetected. To approximate this potential for missing features that are truly present, options within `mvnorm_sim` allow the user to specify: 1) the percent of individuals to generate missing values from (missing_pct), 2) the number of measurements per individual to assign as missing (missing_per_subject), and 3) the value to impute for missing observations (miss_val). Sample IDs are randomly chosen without replacement across all \(n\) units and for each selected ID measurements are randomly selected without replacement from \(\{t_1, \ldots, t_\}\) until the specified number of measurements per individual is achieved. For each missing measurement selected the observed value is replaced with the user specified missing value. Typically the missing value is specified as 0 or as \(\text{NA}\) with the first case representing a situation where the feature
was not included due to technical limitations and the second representing an individual whose data was not collected for a particular time point. The initial value $t_1$ cannot be assigned as missing since it is assumed that all individuals have baseline values collected.

**Implementation**
The current version of the R Bioconductor software package microbiomeDASim\(^\text{12}\) can be installed in R with the following executable code:

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

Alternatively, a development version is available from GitHub and can be accessed at the following repository `williazo/microbiomeDASim`. The developmental version may contain additional features that are being developed before they are officially introduced into the Biocondutor version. The developmental version can be installed using the following code:

```r
if(!requireNamespace("devtools", quietly = TRUE)){
  install.packages("devtools")
}
devtools::install_github("williazo/microbiomeDASim")
```

For a guided introduction into using the functions see either the package vignette for a static example of how to set up and interact with various options for simulating data or for a dynamic guide see `mvrnorm_demo.ipynb`, a Jupyter notebook on the GitHub page under the inst/script directory.

**Operation**

microbiomeDASim is compatible with major operating systems including Mac OS, Windows and Linux.

Package dependencies and system requirements are outlined in the documentation available at GitHub.

**Use cases**

**Data generating procedure**
The primary mechanism for simulating data in the microbiomeDASim package is the function `mvrnorm_sim`. Through this function, the number of subjects in each group is specified along with the necessary parameters, i.e. $\beta$, $\sigma^2$, $\rho$, and $IP$, to generate $\mu$ and $\Sigma$. Below is an example of generating differential abundance using a quadratic trend.

```r
> library(microbimeDASim)
> sim_dt <- mvrnorm_sim(n_control = 20, n_treat = 20, control_mean = 2, sigma = 1, +   num_timepoints = 6, rho = 0.7, corr_str = "compound", +   func_form = "quadratic", beta = c(0, 3, -0.5), +   missing_pct = 0, missing_per_subject = 0, +   dis_plot = TRUE)
> typeof(sim_dt)
[1] "list"
> names(sim_dt)
[1] "df" "Y" "Mu" "Sigma" "N" "miss_data" "Y_obs"
> head(sim_dt$df)
  Y ID   time group   Y_obs
1  3.499028   1    1 Control  3.499028
2  2.680805   1    2 Control  2.680805
3  2.695162   1    3 Control  2.695162
4  2.654708   1    4 Control  2.654708
5  3.529244   1    5 Control  3.529244
6  3.014870   1    6 Control  3.014870
> head(sim_dt$miss_data)
[1] miss_id
<0 rows> (or 0-length row.names)
```
The output of the simulation function is a list with 7 total objects. The main object of interest is df, which is a data.frame that contains the complete outcome, Y, IDs for each subject i = 1, … , n, the corresponding time for each observation t, a group variable indicator, and the outcome with missing data, Y_obs. Both the complete and missing data vectors are also returned as independent objects, Y and Y_obs, respectively, along with the complete mean, \( \mu_{yx} = \text{Mu} \), and covariance matrix, \( \Sigma = \text{Sigma} \). The function also includes a data.frame miss_data which lists any IDs and time points for which missing data was induced. Finally, the function also returns the total number of observations, \( N = \sum q \). The option dis_plot is used to automatically generate a time-series plot tracking each individuals trajectory along with group mean trajectories. The corresponding plot for this data is shown in Figure 2a.

One important thing to note about the example above is that we generated no missing observations as both missing_pct and missing_per_subject were set to 0. Therefore miss_data was empty. We can compare this to the case below where we induce missingness into the data.

```r
> sim_dt <- mvrnorm_sim(n_control = 20, n_treat = 20, control_mean = 2, sigma = 1,
>                         num_timepoints = 6, rho = 0.7, corr_str = "compound",
>                         func_form = "quadratic", beta = c(0, 3, -0.5),
>                         missing_pct = 0.2, missing_per_subject = 2,
>                         miss_val = 0, dis_plot = TRUE)
> head(sim_dt$miss_data[order(sim_dt$miss_data$miss_id, sim_dt$miss_data$miss_time),])
miss_id miss_time
6      10    3
5      10    5
11     14    2
12     14    6
15     16    4
16     16    5
> head(sim_dt$df[sim_dt$df$ID %in% sim_dt$miss_data$miss_id, ])
   Y ID time   group    Y_obs
55 3.461887 10    1 Control 3.461887
56 2.213105 10    2 Control 2.213105
57 1.369042 10    3 Control 0.000000
58 3.221391 10    4 Control 3.221391
59 2.053757 10    5 Control 0.000000
60 1.10175  10    6 Control 1.10175
```

Figure 2. Simulating a quadratic differential abundance trend with compound correlation structure and parameters: \( \beta = (0, 3, -0.5)^T \), \( \rho = 0.7 \), \( \sigma = 1 \), \( n_2 = n_1 = 20 \), \( q = 6 \). Missing data in Figure 2b is generated with 20% of subjects randomly selected to have missing values and for each of these subjects to have 2 non-baseline times randomly selected to be missing with the missing observations imputed as 0.
In this case we see that for $t_3$ and $t_5$ for subject 10 that our outcome with missing data, $Y_{\text{obs}}$, is now set as 0 which was specified as our missing value while the complete data has the original value before inducing missingness. The corresponding plot for this simulation with the missing data is shown in Figure 2b.

As mentioned in the Distributional assumptions section, data are generally generated one feature at a time. However, we may want to simultaneously create data with similar patterns across a number of features with certain features experiencing differential abundance while others have no differential abundance patterns. To do this we can use the function `gen_norm_microbiome` which lets users specify the number of total features to simulate, `features`, and the number of total features to be differentially abundant, `diff_abun_features`. In the example below 10 total features are generated with 4 features having longitudinal differential abundance with an L_down hockey stick type trend.

```r
> bug_gen <- gen_norm_microbiome(features=10, diff_abun_features=4,
+ n_control=20, n_treat=20, control_mean=2, sigma=1,
+ num_timepoints=7, rho=0.7, corr_str="compound",
+ func_form="L_down", beta=c(2, -0.5),
+ missing_pct=0.2, missing_per_subject=2,
+ miss_val=0)
```

Simulating Diff Bugs
```
|++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed = 08s
```

Simulating No-Diff Bugs
```
|++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed = 11s
```

```r
> head(bug_gen$bug_feat)  
ID time group Sample_ID
1 1 1 Control Sample_1
2 1 2 Control Sample_2
3 1 3 Control Sample_3
4 1 4 Control Sample_4
5 1 5 Control Sample_5
6 1 6 Control Sample_6
```

```r
> bug_gen$Y[, 1:5]
Sample_1 Sample_2 Sample_3 Sample_4 Sample_5
Diff_Bug1 1.940647 1.080137 1.969695 2.030142 1.650714
Diff_Bug2 3.795988 3.217864 2.947941 3.800852 3.413415
Diff_Bug3 1.471484 1.861395 2.095946 3.281902 2.148684
Diff_Bug4 2.383222 2.409076 3.511735 1.861285 3.322820
NoDiffBug1 1.952906 2.232935 1.716124 2.432606 1.669670
NoDiffBug2 2.087367 2.354907 2.541538 3.323987 2.258404
NoDiffBug3 3.011910 3.862437 3.047146 3.585548 3.687133
NoDiffBug4 1.060059 1.118622 1.578225 1.669679 1.578786
NoDiffBug5 1.375593 1.251305 0.495152 1.796081
NoDiffBug6 1.555397 1.448880 1.601438 1.715037 0.904486
```

There are two objects returned in this function, `bug_feat` and `Y`. The object `bug_feat` contains all of the sample specific information including Subject ID, timepoint $t_j$, an indicator for group assignment and the Sample_ID which ranges from Sample_1 up to Sample_N. The other object `Y` is the typical OTU table with rows corresponding to features and column to samples that are commonly used for analysis in packages such as `metagenomeSeq`\textsuperscript{13,14}.

**Longitudinal differential abundance estimation**

Next, we want to use our simulation design to test some of the available methods to estimate longitudinal differential abundance. We will examine properties of the estimation method available in the `metagenomeSeq`\textsuperscript{14} package to fit a Gaussian smoothing spline ANOVA (SS-ANOVA) model\textsuperscript{15,16} referred to here after as the `metaSplines` method. We start by generating our simulated data. In this example we will fix parameters so that we have $q = 10$ repeated measurements on each individual with $n_0 = n_1 = 30$ individuals per arm.
> # generating the simulated data
> out_sim <- mvrnorm_sim(n_control = 30, n_treat = 30, control_mean = 2, sigma = 1,
+                       num_timepoints = 10, rho = 0.8, corr_str = "compound",
+                       func_form = "L_up", beta = 0.5, missing_pct = 1,
+                       missing_per_subject = 2, IP = 5)

> # capturing the true mean values for the specified functional form
> true_mean <- mean_trend(timepoints = 1:10, form = "L_up", beta = 0.5, IP = 5)

After generating the simulated data, we can now create an MRexperiment object needed to fit the model. Note
that you can fit either the outcome with the complete data or the outcome with the imputed missing data. In this
case we use the complete data.

> # extracting the sample information
> p_dat <- out_sim$df[, -grep("Y", names(out_sim$df))]
> row.names(p_dat) <- paste0("Sample_", seq_len(nrow(out_sim$df)))
>
> # MRexperiment object with the non-missing counts
> mvrnorm_meta <- AnnotatedDataFrame(p_dat)
> MR_mvrnorm <- newMRexperiment(count = t(out_sim$Y), phenoData = mvrnorm_meta)
> MR_mvrnorm
MRexperiment (storageMode: environment)
   assayData: 1 features, 600 samples
      element names: counts
   protocolData: none
   phenoData
      sampleNames: Sample_1 Sample_2 ... Sample_600 (600 total)
      varLabels: ID time group
      varMetadata: labelDescription
   featureData: none
   experimentData: use ‘experimentData(object)’
   Annotation:
>
> # fitting the metaSplines model with random intercept
> metasplines_mod <- fitTimeSeries(obj = MR_mvrnorm, formula = abundance ~
+                                 time*class,
+                                 id = "ID", time = "time", class = "group",
+                                 feature = 1, norm = FALSE, log = FALSE, B = 1000,
+                                 random = ~ 1|id)

Loading required namespace: gss

[1] 100
[1] 200
[1] 300
[1] 400
[1] 500
[1] 600
[1] 700
[1] 800
[1] 900
[1] 1000

Now we can display the estimated interval of differential abundance

> metasplines_mod$timeIntervals
   Interval start Interval end     Area       p.value
[1,]              6           10 6.457622   0.000999001
Then we can compare the estimated trend $\hat{f}(t)$ to the truth $f(t)$ as shown in Figure 3. We observe that the metaSplines estimate falls closely to the true functional form. Further, the confidence intervals for the functional form completely contain the true trend reflecting that the variability in estimation is accurately reflected.

**Evaluating estimation procedures**

In the example for metaSplines above we looked at performance using a visual inspection for a single choice of parameter values. Using our simulation framework we can expand our investigation of performance. By knowing the true underlying functional form we can quantify how accurate a particular estimation method captures the truth as a function of sample size per group, number of repeated observations, signal-to-noise strength, type of functional form etc. In order to use the simulated data to compare different longitudinal methods for estimating differential abundance we need to define performance metrics that quantify how accurate an estimate is to the truth. We propose four different performance metrics that can be used when comparing methods.

1. Sensitivity/Specificity $\in [0, 1]$
2. Cosine Similarity $\frac{\hat{f}(t) \cdot f(t)}{\|\hat{f}(t)\| \|f(t)\|} \in [-1, 1]$
3. Euclidean Distance $\|\hat{f}(t) - f(t)\| \in [0, \infty]$
4. Normalized Euclidean Distance $\frac{\|\hat{f}(t) - f(t)\|}{\|f(t)\|} \in [0, 2]$

To ensure robustness, for each set of parameter values simulated multiple repetitions, $B$, are required. Sensitivity is defined as the number of repetitions where any differential abundance at any value $t_j \in \{t_1, \ldots, t_q\}$ is
detected over the total number of repetitions given that the functional form had some true differential abundance over time, i.e. \( f(t_j) \neq 0 \forall t \in \{t_1, \ldots, t_q\} \). Likewise, specificity is defined as the number of repetitions where no differential abundance was detected across all timepoints over the total number of repetitions given that the functional form had no true differential abundance over time, i.e., \( f(t_j) = 0 \forall t \). The other remaining metrics are continuous values that look to compare how closely the estimated mean trend is to the true trend at a set of points \( t_j \in \{t_1, \ldots, t_q\} \). Cosine similarity is comparable across different lengths of \( t \), but is not particularly discriminant especially near the boundaries around \(-1\) and \(1\). The Euclidean distance quantifies how far apart each point is but the length of \( t \) is highly influential. Therefore, to make the Euclidean distance comparable across different lengths of repeated observations we can use the normalized Euclidean distance which first transforms the estimated and true functional form into unit vectors and then calculates the distance between these unit vectors.

**Sensitivity and specificity results**

Using these performance metrics we simulated data across a range of different parameters settings and then estimated the functional form of the trend using the metaSplines procedure described earlier for a total of 100 repetitions for each parameter setting. Below we show the performance results for a simulation where the functional form was fixed as \( \text{L} \_\text{up} \) with an AR(1) correlation structure, \( \rho = 0.7 \), and varied the sample size per group, standard deviation, and timepoints from small, medium, and large respectively. The corresponding sensitivity and specificity results are shown in Figure 4a and Figure 4b.

Looking at Figure 4a, in general the sensitivity decreases as \( \sigma \) increases for a fixed sample size and \( q \). For example when \( n_0 = n_1 = 10 \) and \( q = 6 \) the estimation procedure is perfectly sensitive (100\%) when \( \sigma = 1 \) but has lower sensitivity (42\%) when \( \sigma = 4 \). Also as the sample sizes increases for a fixed \( q \) and \( \sigma \), sensitivity generally increases. Likewise, as the number of repeated observations increase, i.e. \( q \) increases, the sensitivity increases quite dramatically. This figure suggests that 6 repeated measurements is sufficiently large to detect differential abundance for strong (\( \sigma = 1 \)) or medium (\( \sigma = 2 \)) signals regardless of the sample size per group. On the other hand, we can look at the specificity in Figure 4b to see that these trends are no longer monotonic. In general we note that as \( q \) increases the specificity decreases and that as \( \sigma \) increases the specificity tends to increase. However, the trend for sample size is more nuanced and may variable due to the number of repetitions that were estimable. Using the metaSplines method there were cases with small sample size and repeated observations that the method returned no estimate.

The sensitivity results shown above were for a single choice of functional form, but this is another potential parameter of interest to test. We ran a similar set of parameter combinations for 7 other functional forms...
shown in Table 1 below to compare the sensitivity as a function of the type of trend. In this table we can see that the non-differential trends, Oscillating, and variable trends, Hockey Stick, had lower average sensitivity while the linear and quadratic trends tended to perform the best.

Continuous performance results
The continuous performance metrics for the cosine similarity, Euclidean distance and normalized Euclidean distance are shown in Figure 5 for the L_up trend with AR(1), $\rho = 0.7$. From this figure we see similar trends as the sensitivity results. Starting from the left most panel we see that the cosine similarity is highest when $\sigma$ is small, $q$, $n_0$, $n_1$ are large. The spread of cosine similarity scores when $q = 12$ are very tightly clustered around 1 while the spread of values when $q = 3$ or $q = 6$ is larger. The center plot illustrates that using raw Euclidean distances with a small number of repeated measurements tend to have smaller distances, but this trend is not seen with normalized Euclidean distance in the last panel. Within each value of $q$ in this middle panel there is a consistent trend that as the sample size per group increases the distance generally decreases. Finally moving to the last panel we have the normalized Euclidean distance, which can now be used to compare across different repeated measurement panels. We see a similar trend to the cosine similarity where the distance decreases, meaning better performance, for small $\sigma$ and large $q$ and $n_0 = n_1$.

Similar to the sensitivity performance metrics shown in Table 1, we can also compare the average value of the continuous performance metrics based on functional form. This is shown in Table 2. Similar trends appear in this table with the linear trends having the highest average cosine similarity scores and lowest average normalized Euclidean distance and non-differential trends performing worse.

<table>
<thead>
<tr>
<th>Functional Form</th>
<th>Sensitivity</th>
<th>Total Repetitions</th>
<th>Non-Missing Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Increasing</td>
<td>1.00</td>
<td>2700</td>
<td>2686</td>
</tr>
<tr>
<td>Linear Decreasing</td>
<td>0.97</td>
<td>2700</td>
<td>2634</td>
</tr>
<tr>
<td>Quadratic: Concave Up</td>
<td>0.91</td>
<td>2700</td>
<td>2154</td>
</tr>
<tr>
<td>Quadratic: Concave Down</td>
<td>0.95</td>
<td>2700</td>
<td>2600</td>
</tr>
<tr>
<td>Oscillating 1</td>
<td>0.96</td>
<td>2700</td>
<td>2614</td>
</tr>
<tr>
<td>Oscillating 2</td>
<td>0.84</td>
<td>2700</td>
<td>2501</td>
</tr>
<tr>
<td>Hockey Stick 1</td>
<td>0.78</td>
<td>2700</td>
<td>2261</td>
</tr>
<tr>
<td>Hockey Stick 2</td>
<td>0.77</td>
<td>2700</td>
<td>2280</td>
</tr>
</tbody>
</table>

Figure 5. Estimated values of the normalized Euclidean distance based on 100 repetitions for an L_up Hockey Stick trend with AR(1) correlation structure, $\rho = 0.7$, simulated across multiple settings varying repeated measurements $q$, sample size per group, $n_0$ and $n_1$, and $\sigma$. Note that the red dashed line serves as a reference point at 0.5 and the green dot in each panel represents the mean value across the 100 repetitions.
Table 2. Average continuous performance metrics from metaSplines method for data simulated from each respective functional form for a total of 100 repetitions across 27 different parameter settings fixing the correlation structure to be AR(1) with $\rho = 0.7$. Parameter values used: $\sigma \in \{1, 2, 4\}$, $n_0 = n_1 \in \{10, 20, 50\}$, $q \in \{3, 6, 12\}$. Note that the Total Non-Missing Observations is less than the Total Observations.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Increasing</td>
<td>2700</td>
<td>2686</td>
<td>0.99</td>
<td>1.26</td>
<td>0.07</td>
</tr>
<tr>
<td>Linear Decreasing</td>
<td>2700</td>
<td>2634</td>
<td>0.98</td>
<td>1.27</td>
<td>0.09</td>
</tr>
<tr>
<td>Quadratic: Concave Up</td>
<td>2700</td>
<td>2154</td>
<td>0.94</td>
<td>1.60</td>
<td>0.23</td>
</tr>
<tr>
<td>Quadratic: Concave Down</td>
<td>2700</td>
<td>2600</td>
<td>0.97</td>
<td>1.55</td>
<td>0.15</td>
</tr>
<tr>
<td>Oscillating 1</td>
<td>2700</td>
<td>2614</td>
<td>0.97</td>
<td>1.69</td>
<td>0.14</td>
</tr>
<tr>
<td>Oscillating 2</td>
<td>2700</td>
<td>2501</td>
<td>0.88</td>
<td>1.71</td>
<td>0.35</td>
</tr>
<tr>
<td>Hockey Stick 1</td>
<td>2700</td>
<td>2261</td>
<td>0.84</td>
<td>1.35</td>
<td>0.40</td>
</tr>
<tr>
<td>Hockey Stick 2</td>
<td>2700</td>
<td>2280</td>
<td>0.84</td>
<td>1.38</td>
<td>0.38</td>
</tr>
</tbody>
</table>

Conclusions

With an increasing emphasis on understanding the dynamics of microbial communities in various settings, longitudinal sampling studies are underway. There remain many statistical challenges when dealing with longitudinal data collected from marker-gene amplicon sequencing. In order to validate and compare methods of estimation for longitudinal differential abundance a unified simulation framework is needed. With the microbiomeDASim package the tools are now available to simulate various functional forms for longitudinal differential abundance with added flexibility to control important factors such as the number of repeated measurements per subject, the number of subjects per group, etc. We have shown the benefit of these simulation tools using the metaSplines estimation procedure to compare the performance across a wide range of different parameter settings. In this manner the microbiomeDASim helps meet an important need in the research community to help compare existing methods as well as validate potentially novel methods.

Data availability

All data shown from the Use Cases section were simulated and can be generated using source code shown above.

Software availability

microbiomeDASim is available at: http://bioconductor.org/packages/microbiomeDASim.

Source code available from: https://github.com/williazo/microbiomeDASim

Archived source code at time of publication: https://doi.org/10.5281/zenodo.345856312.

License: MIT.

Author contributions

JW performed analyses, implemented software and wrote first draft of article. HCB contributed to analysis and article review. JT and JNP oversaw analyses and designed experiment.

Acknowledgments

Authors would like to acknowledge Jane Fridlyland and Christina Rabe for helpful discussions and support.
References


Open Peer Review

Current Peer Review Status: ? ✓

Version 1

Reviewer Report 06 November 2019

https://doi.org/10.5256/f1000research.22722.r55802

© 2019 Sankaran K. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Kris Sankaran
Montreal Institute for Learning Algorithms (MILA), Montreal, QC, Canada

Contributions

The authors have developed an R package to simulate longitudinal microbiome time course data, especially where there are differences in trajectories between treatment and control groups. This can be used to address,

1. Experimental design: Simulations can guide power analysis, to see whether a proposed study will be well-powered, as a function of assumptions on the generating mechanisms.

2. Methods comparisons: The effectiveness of different methods will depend on the structure of the data, and simulations provide ground truth from which to make assessments.

They simulate data one species at a time. Both treatment and control groups are assumed to have Gaussian data, truncated below at 0 to reflect transformed counts. Control data are assumed to be drawn from some common mean, but with specified correlation structure over time. Treatment data are assumed to have a mean that deviates from the control according to some function f(), but have the same correlation structure. The authors provide an interface for simulating a few patterns of f() that are believed to be common in real data (e.g., oscillating, quadratic, and linear shapes).

The authors share code to display simulated data. They also describe a study evaluating the power of a particular method, 'metaSplines', as simulation parameters are changed.

Evaluation

Strengths:

- I like the idea of formalizing simulation-based power analysis. In the microbiome setting, simulations make more sense than theory, but have two issues (1) they are potentially labor-intensive and (2) they can be ad hoc, and never published. By preparing a package, the
authors lower the barrier to entry to / introduce a more formal standard for this work, hopefully enabling simulation-based power analysis in the field.

- The paper is generally technically sound, and reads well. Code is available publicly, is clearly documented, and written in a professional style.

Weaknesses:
- The simulated data are never properly evaluated -- this is my reason for the "partly" response in my report. Of course, any simulation is only an approximation of reality, but it would be nice to know along which dimensions the approximation is close, and along which it is poor. This would also set the stage for studying whether the conclusions that you're aiming for (study design or methods choices) are substantially affected by / robust to these deviations in real data. Something in the spirit of graphical inference could be quite interesting here.¹

Missed Opportunities:
- The 'metaSplines' analysis ends somewhat abruptly, because it's not clear what actual conclusions would be drawn from it. I think it would be interesting if you compared another method against it, because you'd be getting at something like the relative efficiency of the approaches (you could also measure their robustness to particular assumptions).

- The functional forms seem somewhat restrictive, though I see their value for people who don't want to spend time writing code. Could you define some kind of interface that makes it easier for people to specify classes of alternatives? E.g., maybe you could let people draw functions interactively, or use as input some examples of microbiome series they see in real data.

Discussion
- I have trouble believing in any kind of i.i.d. assumption across species. First, the scale of abundance across species tends to differ by orders of magnitude. Second, many species exhibit very similar behavior.

- Among the controls, couldn't some species also vary over time, because of factors in that individual that change which are not specifically treatment?

- Setting missing data to 0 is generally bad practice, because then you can't distinguish true zeros from missingness. You should either do proper missing data imputation, or recommend methods that explicitly model the missign values / don't require measurements at equal timepoints.

- The different correlation structures you propose reflect an equispaced sampling design. It wouldn't be too hard to change the correlation structure to allow for unevenly spaced sampling, and it would address your point (4, "Asynchronous repeated measures").

- Could you create an interactive notebook? E.g., using binder: https://mybinder.org/v2/gh/krisrs1128/microbiome_dasim_example/master. This would make it easier for people (esp. nonexperts) to get acquainted with your work, without having to install jupyter etc.

- For dosage effects, I'd find a (reversed) sawtooth or wavelet-style spike more believable than an oscillating function. But again, this is related to the point of letting people choose their own alternatives.
Minor Comments
• The caption in Figure 5 seems deprecated.
• I don't think you ever defined "OTU".
• The library load should say "microbiome" not "microbime".
• There are still a few typos here and there (e.g., "differential abundant" features and "metrics of success results"), so I recommend another careful read.

References

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Partly

**Competing Interests:** No competing interests were disclosed.

**Reviewer Expertise:** statistics

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Reviewer Report 05 November 2019
https://doi.org/10.5256/f1000research.22722.r55801

© 2019 Lahti L. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.
This manuscript introduces a new method for simulating longitudinal differential abundance for microbiome data. The method is implemented as an R/Bioc package. The proposed package allows the user to simulate longitudinal microbiome data based on various assumptions, and allows the tuning of key design aspects such as signal-to-noise ratio, correlation structure, effect size and zero inflation. One of the available methods is validated with benchmarking comparisons.

The manuscript is technically sound and written in a fluent and easily understandable English. Experiments and statistical analyses have been conducted rigorously. The source code and experiments are openly available via Github but I have not tried to replicate the analysis.

Realistic simulations are valuable for study design, and help to address questions about sample size, density of time points, experimental costs, etc. The work provides pragmatic solutions to a topical problem in microbiome bioinformatics.

**Major comments:**

1. The simulator provides versatile options to tune signal shape, correlations, and noise. However, I am left wondering how well the simulations correspond to real microbiome data. In particular, it is not clear nor validated how the time series shape and correlation structures correspond to known processes in microbial ecology, such as neutral process, competition models (such as generalized Lotka-Volterra), compositionally aware naive models (Dirichlet-Multinomial), mean-reversing processes (Ornstein-Uhlenbeck). All of these have ecological interpretations and have been visible in recent microbiome time series literature. These models are motivated by known ecological processes, rather than technical modifications on the signal shape; it would be relevant to know how large impact the chosen modeling assumptions might have on the results. Can we expect that the proposed simulator will yield qualitative similar conclusions, even if the connection to ecological mechanisms might be weak?

2. The proposed model does not (explicitly) account for heteroskedasticity or overdispersion, and its performance has not been demonstrated with recently popular models of differential abundance, such as DESeq2. It could be true that longitudinal testing of differential abundance requires different methodology. But longitudinal simulators can be also used to simulate cross-sectional data, which is always a snap-shot of longitudinal data. I wonder if the simulator would perform well with standard methods for cross-sectional data; or if it can be shown to yield similar overall distributions. This could provide some additional support for the simulations as the feasibility of the modeling assumptions and their impact on the conclusions remains open.

**Minor comments:**

1. Other simulators for microbiome data and time series are available. One that I am aware of is the seqtime package (https://github.com/hallucigenia-sparsa/seqtime), although that is only available as an R package (and not formally published), but there may be other recent simulators. I did not find other simulation works being cited, it would be good to check if other simulators can be identified in the recent literature, and how they relate to this work.
2. Lack of integration with phyloseq is a weakness, as this class structure is now very popular among the microbiome R users, and many tools build directly on that class structure. It would be useful addition to the package if the simulations could be made available in a phyloseq format.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Partly

**Competing Interests:** No competing interests were disclosed.

**Reviewer Expertise:** Microbiome bioinformatics.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.