LDM Package

Yi-Juan Hu and Glen A. Satten

February 6, 2019

1 Overview

The LDM package implements the Linear Decomposition Model (Hu and Satten 2019), which provides a single analysis path that includes global tests of any effect of the microbiome, tests of the effects of individual OTUs (operational taxonomic units) or ASVs (amplicon sequence variants) while accounting for multiple testing by controlling the false discovery rate (FDR), and a connection to distance-based ordination. It accommodates multiple covariates (e.g., clinical outcomes, environmental factors, treatment groups), either continuous or discrete (with ≥ 2 levels), as well as interaction terms to be tested either singly or in combination, allows for adjustment of confounding covariates, and uses permutation-based p-values that can control for clustered data (e.g., repeated measurements on the same individual). It gives results for both the frequency and arcsine-root-transformed data, and can give an "omnibus" test that combines results from analyses conducted on the two scales.

The core function of the LDM package is `ldm`, which can be used to test association between covariates of interest and the microbiome, possibly after adjusting for confounding covariates. Several additional functions are included in the LDM package. The function `permanovaFL` implements the PERMANOVA-FL test (our version of the PERMANOVA test); unlike the implementation of PERMANOVA in `adonis` and `adonis2` in the R package `vegan`, `permanovaFL` adopts the permutation scheme described by Freedman and Lane (1983). The function `adjust.data.by.covariates` produces the adjusted distance matrix and OTU table after removing the effects of covariates (e.g., confounders).

The function `ldm` uses two sequential stopping criteria. For the global test, LDM uses the stopping rule of Besag and Clifford (1991), which stops permutation when a pre-specified minimum number (default = 20) of rejections (i.e., the permutation statistic exceeded the observed test statistic) has been reached. For the OTU-specific tests, LDM uses the stopping rule of Sandve et al. (2011), which stops permutation when every OTU test has either reached the pre-specified number (default = 20) of rejections or yielded a q-value that is below the nominal FDR level (default = 0.1). As a convention, we call a test "*stopped*" if the corresponding stopping criterion for that test has been satisfied. All tests are terminated when a user-specified maximum number of permutations have been generated; if this number is "too

small", some tests may not have "stopped". Tests that require a large number of permutations are those that are close to the threshold for acceptance or rejection. The function `permanovaFL` only uses the stopping rule of Besag and Clifford (1991) as it only implements a global test. Further, unless specified otherwise by the user, the LDM automatically performs tests on two scales: the frequency scale and the arcsin-root-transformed frequency scale. The omnibus test reports the most powerful of the two scales (after properly adjusting for the comparison).

The main features of the LDM package are illustrated here using the throat microbiome data that formed the basis of the simulations in Hu and Satten (2019) and a simulated dataset with clustered samples.

2 Example dataset 1: throat microbiome data

The throat microbiome data included in this package contain 60 subjects with 28 smokers and 32 non-smokers. Microbiome data were collected from right and left nasopharynx and oropharynx region to form an OTU table with 856 OTUs. The phylogenetic tree was constructed using UPGMA on the K80 distance matrix of the OTUs. Metadata include smoking status, packs smoked per year, age, gender, and antibiotic use. For further information on these data see Hu and Satten (2019) and references therein.

3 Prepare the throat microbiome data

```
> library(LDM)
> library(GUniFrac) # for calculating UniFrac distance;
>                   # not needed unless a UniFrac distance needs to be calculated
>
> data(throat.tree)    # phylogenetic tree; only needed if a UniFrac distance is requested
> data(throat.otu.tab) # OTU table; rows are samples, columns are OTUs
> data(throat.meta)    # metadata (covariates of interest, confounding covariates, etc.)
>
> throat.meta$AntibioticUse <-
            (throat.meta$AntibioticUsePast3Months_TimeFromAntibioticUsage != "None")
            # create the antibiotic use variable used in Hu and Satten (2019)
```

The `otu.table` should have rows corresponding to samples and columns corresponding to OTUs; `ldm` will transpose `otu.table` if the number of rows is not equal to the length of the covariates but this consistency check will fail in the unlikely case that the number of OTUs and samples are equal. Note that samples with zero reads for *every* OTU, possibly representing negative controls, are analyzed as they are, but a warning message will be sent. OTUs with zero reads in *every* sample will be automatically removed by `ldm` before any further analysis is carried out.

It is *essential* that the OTU table and the metadata have the same number of observations (samples) and that these occur *in the same order* in the OTU table and all covariates. Note also

that fitting the LDM assumes no missing data; samples having a NA value for any covariate in the formula (including confounders) are removed by `ldm` before any further analysis is carried out. Similarly, if a `cluster.id` variable is specified (see section 7), samples having `cluster.id=NA` are removed.

As in the analysis presented in Hu and Satten (2019), we filtered out OTUs with presence in fewer than 5 samples, leaving 233 OTUs for analysis. Although this criterion is commonly used, its use is not necessary to run `ldm`.

```
> otu_presence = which(colSums(throat.otu.tab>0)>=5)
> throat.otu.tab = throat.otu.tab[,otu_presence]
> dim(throat.otu.tab)
[1]  60 233
```

4 Writing formulas for testing hypotheses using LDM

The core function that implemented the LDM method is `ldm`. The function `ldm` uses a formula interface that is similar to other regression models in R, but with some differences. The general form of a formula for LDM is:

```
>  otu.table | (confounders) ~ (first set of covariates) + (second set of covariates)
                                                    ... + (last set of covariates)
```

The left hand of the formula gives the OTU table and any confounders that need to be controlled for. The generic formula in the absence of confounders is

```
>  otu.table ~ (first set of covariates) + (second set of covariates)
                                                    ... + (last set of covariates)
```

On the right hand side of the formula, parentheses are used to group covariates that should be considered together as a single "submodel". The covariates in each submodel are considered jointly, and in order from left to right such that the variance explained by each submodel is that part that remains after the previous submodels have been fit. If a submodel contains only a single covariate, then the parentheses are optional. Similarly, since the confounders are always treated as a single submodel, the parentheses around the confounders is also optional. The formula

```
>  otu.table ~ (confounders) + (first set of covariates) + (second set of covariates)
                                                    ... + (last set of covariates)
```

will fit the same model as the first formula statement in this section. The only difference is that moving `(confounders)` to the right hand side of the model will produce p-values for tests of the confounders as well as the other submodels; specifying the confounders on the left hand side of the formula suppresses these tests. Suppressing the tests of the confounders can speed

up the `ldm`, both by reducing the number of test statistics calculated in each permutation and by possibly reducing the number of permutations required to satisfy all the stopping criteria.

The standard syntax for formulas in R can be used to expand single covariates or single terms into a model matrices with multiple columns. For example, choosing the first set of covariates to consist of a single factor variable with 2 levels is equivalent to specifying that the first set of covariates consist of 2 indicator variables (one for each factor level). Similarly, the formula

```
>  otu.table ~ as.factor(a) + b*c
```

will have two submodels, the first with as many columns (degrees of freedom) as `a` has levels, and the second (using the standard R syntax for interactions) with columns corresponding to `(b, c, b×c)`.

We now give an example of a formula that can be fit to the throat data. We first wish to test the association of the microbiome with smoking status and then test the association of the microbiome with pack per year after removing the association with smoking status, both tests accounting for the confounding effects of sex and antibiotic use. Thus we specify this formula in the `ldm`:

```
> throat.otu.tab | (Sex+AntibioticUse) ~ SmokingStatus + PackYears
```

The function `ldm` will assume that variable names in a model statement correspond to objects in the R global environment. There are two ways to explicitly specify the environment that variables should be taken from. The first is to specify a dataframe to use with the option `data=dataframe` when calling `ldm`; in the throat microbiome data, we could thus specify `data=throat.meta`. Alternatively, the environment can be explicitly specified as part of the variable name; for convenience, specifying a covariate using the syntax `dataframe$varname` overrides the option `data=dataframe` for that covariate. Note that the otu.table does not need to be part of `dataframe` when using the `data=dataframe` option to specify the environment for the covariates or confounders.

4.1 Fitting the LDM

We can fit the LDM to the observed data (i.e., obtaining the variance explained (VE) without testing their significance) by calling `ldm` with `n.perm.max=0`. This is useful if we only want to see how much variability is explained by each set of covariates, or if we want factor loadings from OTUs. Because we would also like to compare the variance explained by covariates to those by an ordination of the data using distance, an externally-calculated distance matrix can be used, or LDM can calculate the distance internally. Current distances supported include all of those supported by `vegdist` in the `vegan` package (i.e., "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "altGower", "morisita", "horn", "mountford", "raup" , "binomial", "chao", "cao", "mahalanobis") as well as "hellinger" and "wt-unifrac" (weighted Unifrac). The default choice for distance is the Bray-Curtis distance. Note

that when we internally call `vegdist` for its supported distances without changing any of the default settings, the OTU table is converted to the frequency scale if `scale.otu.table=TRUE` and `binary` is set to `FALSE` (i.e., suppressing calculation of presence-absence distances). For some guidance on presence-absence data, see the discussion section of Hu and Satten (2019). The Hellinger distance measure ("hellinger") takes the form 0.5*E, where E is the Euclidean distance between the square-root-transformed frequency data. The weighted UniFrac distance ("wt-unifrac") is calculated by internally calling `GUniFrac` in the `GUniFrac` package. For the throat microbiome data, we choose the Bray-Curtis distance:

```
> fit <- ldm(throat.otu.tab|(Sex+AntibioticUse)~SmokingStatus+PackYears,
             data=throat.meta, dist.method="bray", n.perm.max=0)

> # frequency scale
> fit$VE.global.freq.submodels        # VE for submodels 1 and 2
[1] 0.13891274 0.02751291
> fit$VE.otu.freq.submodels[1,1:3]    # Contribution of OTUs 1-3 to VE for submodel 1
        4695         4194         5160
3.780166e-07 1.138872e-07 2.666018e-03

> fit$F.global.freq            # F statistics for VE for submodels 1 and 2
[1] 0.049056264 0.009716031
> fit$F.otu.freq[1,1:3]        # F statistics for contribution of OTUs 1-3 in submodel 1
       4695        4194        5160
0.098296641 0.000170493 0.041144505

> # arcsin-root scale
> fit$VE.global.tran.submodels       # VE for submodels 1 and 2
[1] 1.2829126 0.3691146
```

The LDM performs a full decomposition (similar to SVD) of the OTU table; the singular values are returned in `d.freq` and `d.tran` (frequency and arcsin-root-transformed scale). It can be informative to make a scree plot to see how much variability is explained by submodel variables compared to the residual variability. The following code accomplishes this; the VE corresponding to the submodels is plotted in red, while residual components are plotted in black. In the code that follows, we handle the possibility of multiple terms per submodel by plotting a single point for each submodel; the value plotted is the average VE (i.e., averaged over each term in each submodel). We could also plot the VE for each term in each submodel by plotting the appropriate values of d.freq^2.

```
> scree.VE.freq <- c(fit$VE.global.freq.submodels/fit$VE.df.submodels,
  fit$VE.global.freq.residuals)
> color <- c(rep("red", 2), rep("black", length(scree.VE.freq)-2))
> plot(scree.VE.freq/sum(scree.VE.freq), main="frequency scale",
       xlab="Component", ylab="Proportion of Total Sum of Squares", col=color)
```

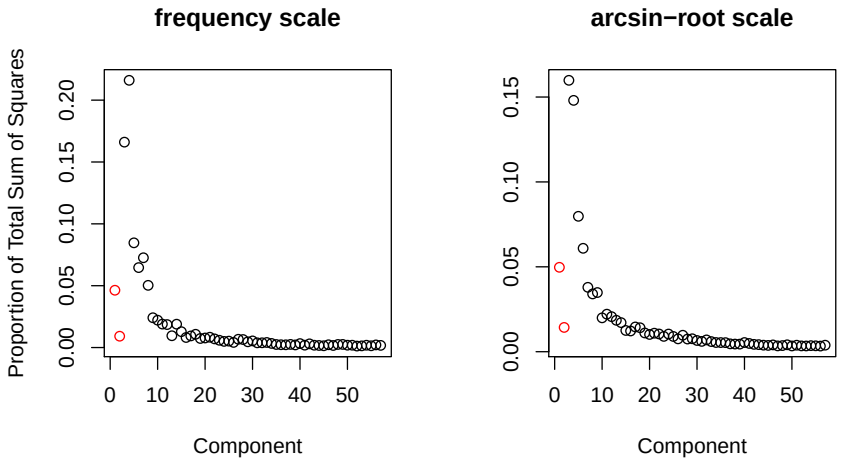

Figure 1 VEs when fitting the LDM to the throat microbiome data. The first two red circles represent VEs by `SmokingStatus` and `PackYears`, respectively. The black circles represent VEs (ordered by their Bray-Curtis eigenvalues) by residual components, usually referred to as the scree plot.

```
> scree.VE.tran <- c(fit$VE.global.tran.submodels/fit$VE.df.submodels,
  fit$VE.global.tran.residuals)
> color <- c(rep("red", 2), rep("black", length(scree.VE.tran)-2))
> plot(scree.VE.tran/sum(scree.VE.tran), main="arcsin-root scale",
       xlab="Component", ylab="", col=color)
```

The resulting plots are shown in Figure 1. We can see that although the VE by `SmokingStatus` does not rank the highest compared to other directions in the B matrix, it still appears substantial. By contrast, the VE by `PackYears` seems very small. Although these results are suggestive, we must use permutation to decide which VEs are significant. The function `ldm` also returns the right singular vectors from the decomposition (denoted `v.freq` on the frequency scale and `v.tran` on the transformed scale) which can be used to construct biplots (see Satten et al. 2017 for additional details).

Finally, some users may prefer to specify the formula statement in a separate line:

```
> form <- throat.otu.tab | (Sex + AntibioticUse) ~ SmokingStatus + PackYears
```

and then call `ldm` using

```
> fit <- ldm(formula=form, data=throat.meta, dist.method="bray", n.perm.max=0)
```

which can be convenient when calling `ldm` multiple times with different settings but the same formula.

4.2 Testing global hypotheses only

While just fitting the LDM can yield interesting information, we usually want to assess the significance of the results found in the fit. We typically start by testing only the global hypotheses by calling `ldm` with `test.global=TRUE` and `test.otu=FALSE`, which frequently takes only a few seconds to finish. The maximum number of permutations can be set through `n.perm.max`, or takes the value 5000 if the default value `n.perm.max=NULL` is used. Note that we can set the seed using an integer for `seed` if we want to reproduce the permutations that are generated; with the default value `seed=NULL`, an integer seed will be generated internally and randomly. In either case, the integer seed at the start of the permutation process will be stored in the output object. In general, a random seed should be used either by setting `seed=NULL` or generating a random seed before calling `ldm` as in this example.

```
> (seed=sample.int(100000, size=1)) # use seed=15597 to reproduce the results below
[1] 15597
> res1.ldm <- ldm(throat.otu.tab|(Sex+AntibioticUse)~SmokingStatus+PackYears,
                  data=throat.meta, test.global=TRUE, test.otu=FALSE, seed=seed)
> res1.ldm$n.perm.completed
[1] 5000
> res1.ldm$global.tests.stopped
[1] FALSE
> res1.ldm$p.global.omni
[1] 0.0042 0.7162
> res1.ldm$seed
[1] 15597
```

With 5000 permutations, the stopping criterion of obtaining at least 20 (default value) rejections has not been met by the global test. According to the omnibus test results, smoking status (after accounting for the confounders) is significantly associated with the microbiome (p-value = 0.0042) while pack per year after accounting for smoking status (and the confounders) is not (p-value = 0.7162). Note that, when the true p-value is very small, a larger number of permutations may be required to "accurately" estimate it and there is no upper bound for `n.perm.max` to guarantee the stop of the algorithm; if the value of `global.tests.stopped=FALSE`, then the reported p-value may be used as an upper bound.

4.3 Testing both global and OTU-specific hypotheses

Finding that the global p-value for smoking status is significant encourages us to test whether we can identify individual OTUs that contribute to the global association by calling `ldm` with `test.otu=TRUE`. For the OTU-specific tests, there exists an upper bound for the number of permutations for the algorithm to stop, which is $J \times L_{\min} \times \alpha^{-1}$, where J is the number of OTUs, $L_{\min}$ is the pre-specified minimum number of rejections (i.e., the permutation statistic exceeded the observed test statistic), and α is the nominal FDR. For example,

when $L_{\min} = 100$ and $\alpha = 0.1$, the default upper bound is $1000J$, which is 856000 for this example dataset. If the default value `n.perm.max=NULL` is used then the maximum number of permutations for testing OTUs is determined in this way, while the maximum number of permutations for the global test remains 5000. Conversely, if a numeric value for `n.perm.max` is specified, this value is used for both the global tests and the OTU-level tests.

```
> (seed=sample.int(100000, size=1))   # use seed=67817 to reproduce the results below
[1] 67817
> res2.ldm <- ldm(throat.otu.tab|(Sex+AntibioticUse)~SmokingStatus+PackYears,
                  data=throat.meta, test.global=TRUE, test.otu=TRUE,
                  n.rej.stop=100, fdr.nominal=0.1, seed=seed)
> res2.ldm$n.perm.completed
[1] 91100
> res2.ldm$global.tests.stopped
[1] FALSE
> res2.ldm$otu.tests.stopped
[1] TRUE
> res2.ldm$p.global.omni
[1] 0.0009998 0.7092000
> res2.ldm$seed
[1] 67817
> w1 = which(res2.ldm$q.otu.omni[1,] < 0.1)
> (n.otu.omni.var1 = length(w1))
[1] 5
> (otu.omni.var1 = colnames(res2.ldm$q.otu.omni)[w1])
[1] "2434" "4363" "1490" "4703" "3538"
> w2 = which(res2.ldm$q.otu.omni[2,] < 0.1)
> (n.otu.omni.var1 = length(w2))
[1] 0
```

We can see that the global test was terminated after 5000 permutations without meeting the early stopping criteria, and the OTU tests stopped (met the early stopping criteria) after 91100 permutations. By identifying those OTUs for which the q-value is less than the nominal FDR = 0.1, we find five OTUs (with OTU IDs "2434" "4363" "1490" "4703" "3538") that contribute to the association with smoking status and no OTU for pack per year. These results are consistent with the pattern of significance in the corresponding global tests.

5 Testing association hypotheses using PERMANOVA-FL

The `permanovaFL` function implementing PERMANOVA-FL (our version of the PERMANOVA test) allows adjustment of confounding covariates, control of clustered data, and multiple sets of covariates to be tested in the way that the sets are entered sequentially and the

variance explained by each set is that part that remains after the previous sets have been fit. Like the PERMANOVA test implemented elsewhere (e.g., the `adonis` or `adonis2` functions in the R package `vegan`), our `permanovaFL` only allows testing of global hypotheses. Here we use the default `n.perm.max=NULL`, i.e., 5000 for the maximum number of permutations.

```
> (seed=sample.int(100000, size=1)) # use seed=82955 to reproduce the results below
[1] 82955
> res.permanova <- permanovaFL(throat.otu.tab|(Sex+AntibioticUse)~SmokingStatus+PackYears,
                               data=throat.meta, dist.method="bray", seed=seed)
> res.permanova
$F.statistics
[1] 2.9354949 0.8020703

$p.permanova
[1] 0.00259948 0.63780000

$n.perm.completed
[1] 5000

$permanova.stopped
[1] FALSE

$seed
[1] 82955
```

6 Ordination using distances adjusted for covariates

Part of adjusting for covariates in the LDM is to (1) remove the effect of the covariates from the OTU table using projection, and (2) projecting off the corresponding directions from the distance matrix. The function `adjust.data.by.covariates` produces this adjusted OTU table and distance matrix, without fitting or testing the LDM. The adjusted distance matrix can be used to perform ordination in which the effects of confounding covariates are removed. As an example, we perform distance-based ordination and visualize whether the samples from the throat data are clustered by smoking status (i.e., smokers vs. non-smokers) after removing the confounding effects from gender and antibiotic use.

```
> adj.data <- adjust.data.by.covariates(formula=~Sex+AntibioticUse, data=throat.meta,
                                        otu.table=throat.otu.tab, dist.method="bray")
> PCs <- eigen(adj.data$adj.dist, symmetric=TRUE)
```

Now we are ready to generate the ordination plot.

```
> color = rep("blue", length(SmokingStatus))
> w = which(SmokingStatus=="Smoker")
```

```
> color[w] = "red"
> plot(PCs$vectors[,1], PCs$vectors[,2], xlab="PC1", ylab="PC2",
       col=color, main="Smokers vs. non-smokers")
> legend(x="topleft", legend=c("smokers","non-smokers"), pch=c(21,21),
        col=c("red","blue"), lty=0)
```

The resulting plot is shown in Figure 2. By removing confounding directions from the distance matrix, any clustering pattern by smoking status is not due to correlation between smoking and gender and/or antibiotic use.

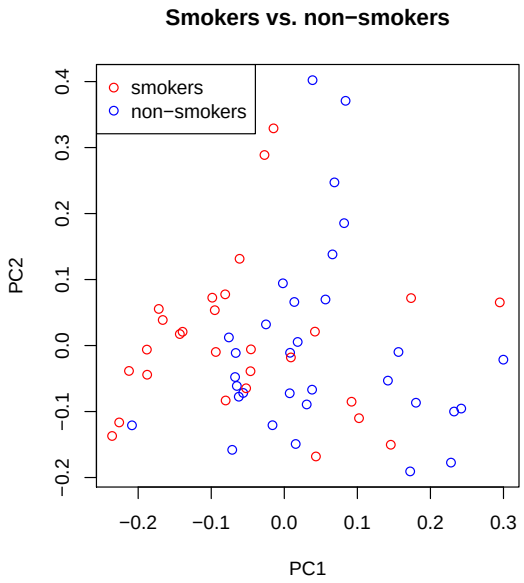

Figure 2 Ordination plot from analysis of the throat microbiome data after removing the effects of gender and antibiotic use. The red and blue circles represent smokers and nonsmokers, respectively.

7 Complex (Restricted) Permutations

The functions `ldm` and `permanovaFL` accommodate a variety of complex (i.e., restricted) permutations through the `permute` package. For unclustered data, `perm.within.type` governs the type of permutation. The default value, `permute.within.type="free"` performs unrestricted permutations, equivalent to using `sample(n.obs, replace=FALSE)`. Other allowed values are "series", "grid" and "none". If `perm.within.type="grid"` is specified, then values of `perm.within.nrow` and `perm.within.ncol` must be specified. For more information on the effect of choosing `perm.within.type` to a value other than "free", see the documentation for the `permute` package. If desired, permutations can also be restricted to groups of observations by specifying `strata=strata.id`, which restricts permutations to those that preserve the value of `strata.id` for each observation.

Clustered data may also be analyzed using `ldm` and `permanovaFL`. Clusters in the data are identified by having a common value of `cluster.id`. There are two variables that control the type of permutations possible with clustered data: `perm.between.type` and `perm.within.type`. The allowable values of `perm.within.type` are as described in the previous paragraph, while the allowable values of `perm.between.type` are "free", "none" and "series". To permute only *within* clusters, choose `perm.between.type="none"` and `perm.within.type="free"`. To permute only *across* clusters, choose `perm.between.type="free"` and `perm.within.type="none"`. To permute *both within and between* clusters, set both to "free". Note that if `perm.between.type` has a value other than "none" then all clusters must have the same size. Permutations of clustered data may be further restricted to preserve the value of `strata.id` by specifying `strata=strata.id`. Note that the value of `strata.id` must be the same for each member of a single cluster. If there are only a few observed cluster sizes, permuting within strata defined by cluster size is a way to analyze unbalanced, clustered data.

Users may also use the `how` function in the `permute` package to manually specify a restricted permutation. This is accomplished by setting `how=how.var` where `how.var` is an object of class `how` as produced by the `permute` package. Any permutation scheme allowed by `permute` can be specified in this way.

8 Example dataset 2: simulated data with clustered samples

To illustrate the use of clustered sampling in `ldm`, we simulated a dataset based on the throat data. It consists of data from 50 individuals, each of whom contributed 2 samples. Fifty individuals were cases ($Y = 1$), and the remaining ones were controls ($Y = 0$). Metadata include the case-control status (Y), a confounding covariate (X), and a unique individual identifier (ID). See Hu and Satten (2019) for more details in generating this dataset. We first prepare this dataset:

```
> library(LDM)
> data(sim.otu.tab) # zero columns have been excluded
> data(sim.meta)
>
> otu_presence = which(colSums(sim.otu.tab>0)>=5)    #optionally remove OTUs that occur
> sim.otu.tab = sim.otu.tab[,otu_presence]           #in fewer than 5 individuals
> dim(sim.otu.tab)
[1] 100 593
```

The OTU table contains 593 OTUs after excluding OTUs having less than 5 presence in the sample. To test the association of the microbiome with the case-control status Y while controlling for the confounder X, while also accounting for the clustering structure (so that the case-control status is always the same for all observations from the same individual), we

specify `cluster.id=ID`, `perm.between.type="free"`, and `perm.within.type="none"` and use the R code:

```
> (seed=sample.int(100000, size=1)) # use seed=34794 to reproduce the results below
[1] 34794
> res4.ldm <- ldm(sim.otu.tab | X ~ Y, data=sim.meta,
                  cluster.id=ID, perm.between.type="free", perm.within.type="none",
                  test.global=TRUE, test.otu=TRUE, fdr.nominal=0.1, seed=seed)
> res4.ldm$n.perm.completed
[1] 10400
> res4.ldm$global.tests.stopped
[1] FALSE
> res4.ldm$otu.tests.stopped
[1] TRUE
> res4.ldm$p.global.omni
[1] 0.00019996
> res4.ldm$seed
[1] 34794
> w1 = which(res4.ldm$q.otu.omni[1,] < 0.1)
> (n.otu.omni.var1 = length(w1))
[1] 21
> (otu.omni.var1 = colnames(res4.ldm$q.otu.omni)[w1])
 [1] "4194" "1583" "3067" "5522" "5443" "2122" "2334" "2245" "444"  "661"  "2213" "330"
[13] "2434" "3619" "4036" "922"  "3954" "3108" "799"  "3957" "4599"
```

With 10400 permutations, the OTU-specific tests have met the stopping criterion and examination of the q-values detects 21 OTUs at the FDR = 0.1 level. The global test has not met the stopping criterion of obtaining at least 20 rejections with 5000 permutations; however, we know the p-value is going to be very small around 0.00019996 and there is no need to add more permutations to make it more precise.

9 References

Besag J, Clifford P. Sequential Monte Carlo p-values. *Biometrika*. 1991;78(2):301–304.

Hu YJ, Satten GA. Testing hypotheses about the microbiome using the linear decomposition model. *bioRXiv*. 2019;doi.org/10.1101/229831.

Sandve GK, Ferkingstad E, Nygard S. Sequential Monte Carlo multiple testing. *Bioinformatics*. 2011;27(23):3235–3241.

Freedman D, Lane D. A nonstochastic interpretation of reported significance levels. *Journal of Business & Economic Statistics*. 1983;1(4):292–298.