LDM Package

Yi-Juan Hu and Glen A. Satten

March 19, 2018

1 Overview

The LDM package implements the Linear Decomposition Model (Hu and Satten 2018), which provides a single analysis path that includes distance-based ordination, global tests of any effect of the microbiome, and tests of the effects of individual OTUs (operational taxonomic units) or ASVs (amplicon sequence variants) with false discovery rate (FDR)-based correction for multiple testing. 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 gives an "omnibus" test that combines results from analyses conducted on the two scales.

The main function of the LDM package is to test association between covariates of interest and the microbiome, possibly after adjusting for confounding covariates. This can be accomplished using the core function `ldm`. 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). Note that the OTU table is converted to the frequency scale and `binary` is set to `FALSE` (i.e., not allowing calculation of presence-absence distances) when we internally call `vegdist` for its supported distances. For some guidance on presence-absence data, see the discussion section of Hu and Satten (2018). 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. 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 = 10) 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 = 10) 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 has been satisfied. Although all tests are always terminated if a pre-specified maximum number (default = 5000) of permutations have been generated, some tests may not have "stopped" and hence their results have not achieved good precision. Further, 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).

Several additional functions are included in the LDM package. The function `test.ldm.fromfile` allows restarting the LDM test from external files that have stored test statistics calculated from the observed data as well as permuted datasets. This function is also useful if we want to carry out more permutations than are needed to satisfy the sequential stopping rule in `ldm` (which can be useful as the output of a sequential stopping rule can vary across runs). The function `permanovaC` implements the PERMANOVA test and uses the same Redundancy-Analysis-based approach as LDM to control for confounding covariates. The function `adjust.data.by.covariates` produces adjusted OTU table and distance matrix after removing the effects of confounding covariates. The function `ortho.decomp` performs (approximate) orthogonal decomposition of any matrix as described by Satten at al. (2017). 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 (2018) 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 (2018) 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 (2018)
```

Samples with zero reads for *every* OTU should be manually removed from the OTU table and the metadata . OTUs with zero reads in *every* sample will be automatically removed by `ldm` before any further analysis is carried out.

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 `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 will fail in the unlikely case that the number of OTUs and samples are equal. If there are no confounders, the vertical bar and list of confounders should not appear in the formula. Thus, 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 "model". The covariates in each model are considered jointly, and in order from left to right such that the variance explained by each model is that part that remains after the previous models have been fit. If a "set" contains only a single covariate, then the parentheses are optional. Similarly, since the confounders are always treated as a single model, the parentheses around the confounders is also optional. Note that 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 models, 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 fairly complex model to that can be fit to the throat data. We first wish to test the association of the microbiome with smoking (using a model that consists of the joint effect of SmokingStatus and PackYears) and then test the association of the microbiome with Age after removing the association with smoking, both tests accounting for the confounding effects of Sex and AntibioticUse. Thus we specify this formula in the `ldm`:

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

note that we do not need to specify the data frame as part of the covariate name if we use the option `data=throat.meta` when we call `ldm`. For convenience, specifying a covariate using the syntax `dataframe$varname` overrides the option `data=data.frame` for that covariate. Finally, if the option `data=data.frame` is specified, covariates that are not in data.frame will be taken from the R global environment (as is the case for `throat.otu.tab` in this example).

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.

4.1 Fitting the LDM

We can fit the LDM to the observed data (i.e., obtaining the VE statistics 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. For the throat microbiome data, the call would be

```
> fit <- ldm(throat.otu.tab|(Sex+AntibioticUse)~(SmokingStatus+PackYears)+Age,
             data=throat.meta, dist.method="bray", n.perm.max=0)
>
> tot.Var.freq <- sum(fit$x.freq^2)
> tot.Var.tran <- sum(fit$x.tran^2)
>
> fit$VE.global.freq/tot.Var.freq
[1] 0.025822177 0.004088777
> fit$VE.global.tran/tot.Var.tran
[1] 0.03754524 0.01010072
```

The (proportion of) variance explained (VE) by the two smoking variables and age after adjusting for confounding are 0.025822177 and 0.004088777 respectively when using the frequency scale, and 0.03754524 and 0.01010072 when using the arcsine-root-transformed frequency scale. Note that we used `x.freq` and `x.tran` to calculate total variability since they have the variability due to confounders removed.

The LDM performs an approximate SVD of the OTU table; the (approximate) singular values are returned in `d.freq` and `d.tran` (frequency and arc-sin-root-transformed scale). It can be informative to make a scree plot to see how much variability is explained by model variables compared to the residual variability. The following code accomplishes this; the VE corresponding to the models is plotted in red, while residual components are plotted in black.

```
> par(mfrow=c(1,2), pty="s")
> color <- c(rep("red",length(fit$VE.global.freq)), rep("black", length(fit$d.freq)))
> plot(c(fit$VE.global.freq, sort(fit$d.freq^2, decreasing=TRUE))/tot.Var.freq,
        xlab="Component", ylab="Proportion of Total Sum of Squares (frequency scale)",
        col=color)
> plot(c(fit$VE.global.tran, sort(fit$d.tran^2, decreasing=TRUE))/tot.Var.tran,
        xlab="Component", ylab="Proportion of Total Sum of Squares (arcsin-root scale)",
        col=color)
```

The resulting plots are shown in Figure 1. We can see that although the VE by `(SmokingStatus+ PackYears)` does not rank the highest compared to other directions in the B matrix (as defined in the orthogonal decomposition), it still appears substantial. By contrast, the VE by `Age` seems minimal. 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 approximate SVD (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). Note that `ldm` will remove any OTUs for which no samples have any counts, which will change the dimension of these vectors.

Alternatively, we can obtain the OTU table after adjusting for confounders and the B matrix externally, e.g., from the output of `ldm`, and perform the orthogonal decomposition using `ortho.decomp`:

```
> od <- ortho.decomp(fit$x.freq, fit$b)
```

Then we obtain the same VEs as using `ldm`. For example, the following code produced the same figure as Figure 1 (left).

```
> plot(c(sum(od$d[1:2]^2), od$d[3]^2, sort(od$d^2, decreasing=TRUE))/tot.Var.freq,
        xlab="Component", ylab="Proportion of Total Sum of Squares (frequency scale)",
        col=color)
```

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

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

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.

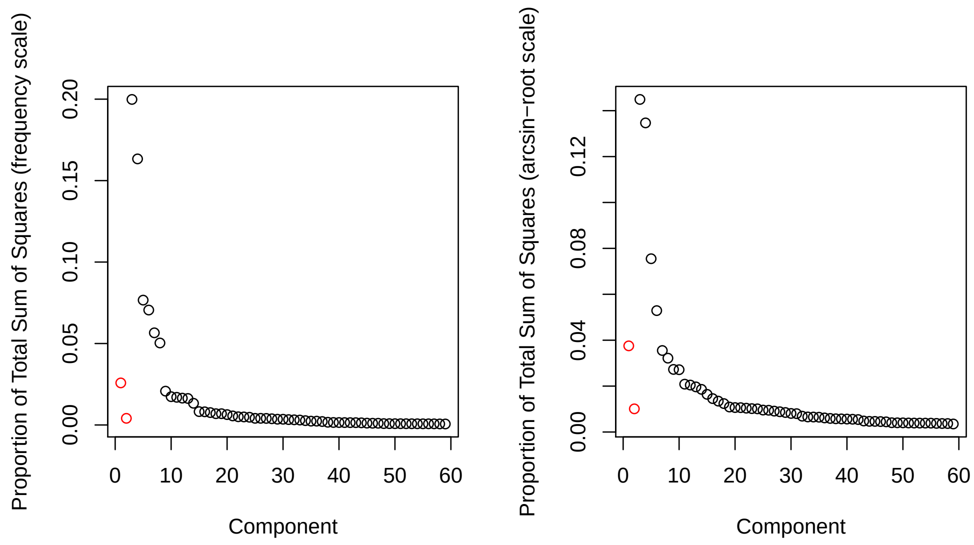

Figure 1 VEs when fitting the LDM to the throat microbiome data. The first two red circles represent VEs by `(SmokingStatus+PackYears)` and `Age`, respectively. The black circles represent VEs (in a descending order) by each column of the B matrix, usually referred to as the scree plot. Note that the VEs marked in red correspond to a single VE or a sum of multiple VEs among those marked in black.

4.2 Testing global hypotheses only

While fitting the LDM can yield interesting information, we typically need 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 usually requires a relatively small number of permutations for the algorithm to stop. We typically start with a small maximum number of permutations, e.g., `n.perm.max=10000` as below, to obtain quick, preliminary results. 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 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)+Age,
                  data=throat.meta, dist.method="bray",
                  test.global=TRUE, test.otu=FALSE,
                  n.perm.max=10000, seed=seed)
> res1.ldm$n.perm.completed
[1] 5600
```

```
> res1.ldm$global.tests.stopped
[1] TRUE
> res1.ldm$p.global.omni
[1] 0.002857143 0.555000000
> res1.ldm$seed
[1] 15597
```

With 5600 permutations, the stopping criterion of obtaining at least 10 (default value) rejections was met by the global test for both sets of covariates. According to the omnibus test results, smoking (after accounting for the confounders) is significantly associated with the microbiome (p-value = 0.002857143) while Age after accounting for smoking (and the confounders) is not (p-value = 0.555). 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 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; with the default values, $L_{\min} = 10$ and $\alpha = 0.1$, this upper bound amounts to $100J$, which is 85600 for this example dataset. Thus we set `n.perm.max=85600`.

```
> (seed=sample.int(100000, size=1))   # use seed=50509 to reproduce the results below
[1] 50509
> res2.ldm <- ldm(throat.otu.tab|(Sex+AntibioticUse)~(SmokingStatus+PackYears)+Age,
                  data=throat.meta, dist.method="bray",
                  test.global=TRUE, test.otu=TRUE, fdr.nominal=0.1,
                  n.perm.max=85600, seed=seed)
> res2.ldm$n.perm.completed
[1] 29700
> res2.ldm$global.tests.stopped
[1] TRUE
> res2.ldm$otu.tests.stopped
[1] TRUE
> res2.ldm$p.global.omni
[1] 0.004857143 0.566285714
> res2.ldm$seed
[1] 50509
> w1 = which(res2.ldm$q.otu.omni[1,] < 0.1)
> (n.otu.omni.var1 = length(w1))
[1] 3
> (otu.omni.var1 = colnames(res2.ldm$q.otu.omni)[w1])
[1] "4363" "689"  "4817"
> w2 = which(res2.ldm$q.otu.omni[2,] < 0.1)
> (n.otu.omni.var1 = length(w2))
[1] 0
```

We can see that with 29700 permutations, the stopping criteria for both the global and OTU-specific tests have been met. The global p-values for smoking and Age are 0.004857143 and 0.566285714, respectively. We detected 3 OTUs (with OTU IDs "4363", "689", and "4817") that contribute to the smoking-microbiome association and no OTU for Age, which results are coherent with the significance of the corresponding global tests. Recall that, if any OTUs have no reads for any sample, they are dropped from the analysis (i.e., no p-values are produced for these OTUs).

In this example we luckily met both stopping criteria by a relatively small number of permutations (i.e., 29700). In the worst case, the OTU-specific tests may require up to $J \times L_{\min} \times \alpha^{-1}$ permutations, which will take a long time to run on a single machine. Since each permutation represents a separate analysis, the work of generating permutation can easily be parallelized, as described in the next subsection.

4.4 Testing both the global and OTU-specific hypotheses through parallel computing

For larger datasets it is preferable to use a high performance computing cluster or cloud computing environment, so that several instances of ldm can be run simultaneously, each with a different values of `seed`. For example, if we wish to base inference on 90000 permutations, we can run 9 separate instances of `ldm`, each having a different seed, and each generating 10000 permutations (which would be sufficient to meet the stopping criterion for OTU-specific tests in the analysis in section 4.3). This simple parallel computing is facilitated by saving the results of each `ldm` permutation in an external file (`outfile.prefix`). Once these output files have been generated, running `test.ldm.fromfile` will read the files and then summarize the results.

To save the results from each permutation, we specify the directory (path) where we want the files stored and a filename prefix, by adding `outfile.prefix="path/prefix"` to the argument list when calling `ldm`. Note that the directory specified by path must already exist; if this is not the case, it can be created from R using `dir.create("path")`. Also note that if path consists only of a single subdirectory name, this subdirectory will be created in the current working directory, whatever that is.

For example, suppose we want `ldm` to create intermediate files in a subdirectory of the current working directory to be called `tmp_files`; further suppose we want each of the intermediate files to start with the prefix `smk_age` (using different filename prefixes allows storing the files from different analyses in the same subdirectory). Finally, suppose we want to run four instances of `ldm`, each computing 10000 permutation, and each using a seed that we select randomly from the integers 1 to 100000. Then we would set `outfile.prefix="tmp_files/smk_age"` and, for the first instance of `ldm`, execute the following commands in R:

```
> dir.create("tmp_files") # create the directory for external files
                            to store intermediate statistics
> (seed.seq=sample.int(100000, size=4, replace=FALSE))  # use seed.seq=c(81426, 92877,
                                                  # 14748, 74980) to reproduce the results below
[1] 81426 92877 14748 74980
> res3.tmp <- ldm(throat.otu.tab|(Sex+AntibioticUse)~(SmokingStatus+PackYears)+Age,
                  data=throat.meta, dist.method="bray",
                  test.global=TRUE, test.otu=TRUE,
                  outfile.prefix="tmp_files/smk_age",
                  n.perm.max=10000, seed=seed.seq[1])
```

Subsequent instances of `ldm` can be run with `seed=seed.seq[2]`, `seed=seed.seq[3]` etc. The seed used to generate the set of permutations is postpended to the output filename. Thus, for example, if seed.seq[1]=81426, when `ldm` had finished the subdirectory `tmp_files` would have a file named `smk_age_global.freq.prem_seed81426.txt` with the permutation results for the global test on the frequency scale. There is also a file `smk_age_seed.txt` with the values of the seeds used (useful if `seed=NULL` was used to invoke each instance of `ldm`).

The results of one or more sets of permutations can be combined using the function `test.ldm.fromfile`. As before, the directory path and filename prefix are specified by specifying a variable with the path/prefix format; in this case the variable is `infile.prefix`. The input variable `n.perm.available` should be set to a (possibly rough) estimate of the total number of permutations to be combined, to help in memory management when reading files that may be quite large. Here, after 4 runs with `n.perm.max=10000` were performed, we set `n.perm.available=40000`. To analyze the four sets of files generated as described above, we use the R commands

```
> res3.ldm <- test.ldm.fromfile(infile.prefix="tmp_files/smk_age", n.perm.available=40000,
                                test.global=TRUE, test.otu=TRUE, fdr.nominal=0.1)
> res3.ldm$n.perm.completed
[1] 40000
> res3.ldm$global.tests.stopped
[1] TRUE
> res3.ldm$otu.tests.stopped
[1] TRUE
> res3.ldm$p.global.omni
       V1        V2
0.00265 0.55900
> w1 = which(res3.ldm$q.otu.omni[1,] < 0.1)
> (n.otu.omni.var1 = length(w1))
```

```
[1] 3
> (otu.omni.var1 = colnames(res3.ldm$q.otu.omni)[w1])
[1] "4363" "689" "4817"
> w2 = which(res3.ldm$q.otu.omni[2,] < 0.1)
> (n.otu.omni.var1 = length(w2))
[1] 0
```

For those with limited computational resources, it is possible to run a relatively small number of simulations on as many cores or processors as are available, and then check if the stopping criterion has been met by running `test.ldm.fromfiles`; if more permutations are required, `ldm` can be run again with a new seed. In the throat example here, even though the analysis could in theory have required 90000 permutations, the stopping criteria were met after 40000 permutations, so there is no need to run the 5 additional jobs we might have started if we had simply run all 9 jobs at once. In this example, using the omnibus test we find the global p-values 0.00265 for smoking and 0.559 for age; we find 3 OTUs detected that are differentially abundant between smokers and nonsmokers, which are the same as those obtained in Section 4.3, but it may not always be this case due to the stochastic nature of the permutation-based test. Note that, when any of the true global p-values is very small, `global.tests.stopped` may not take value `TRUE` even beyond the upper bound for the OTU-specific tests; in this case, the observed p-value can be used as an upper bound.

5 Testing association hypotheses using PERMANOVA

Our implementation of PERMANOVA in the `permanovaC` function allows adjustment of confounding covariates and control of clustered data. As in `ldm`, `permanovaC` allows 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, our implementation `permanovaC` only allows testing of global hypotheses. Here we use the default `n.perm.max=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 <- permanovaC(throat.otu.tab|(Sex+AntibioticUse)~
                                     (SmokingStatus+PackYears)+Age,
                              data=throat.meta, dist.method="bray", seed=seed)
> res.permanova
$F.statistics
[1] 1.9107575 0.8784135

$p.permanova
[1] 0.01686341 0.61382799

$n.perm.completed
```

```
[1] 593

$permanova.stopped
[1] TRUE

$seed
[1] 82955
```

6 Ordination using distances adjusted for confounding

A feature of the LDM is that controlling for confounding covariates can be achieved by (1) removing the effect of the confounders from the OTU table using Redundancy Analysis, and (2) projecting the confounding directions off of the distance matrix. The function `adjust.data.by.covariates` produces such adjusted OTU table and distance matrix, without fitting and 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.

7 Example dataset 2: simulated data with clustered samples

The core function `ldm` also accommodates data with clustered samples. Clusters in the data are identified by `cluster.id`. There are two ways to generate permutations for clustered data

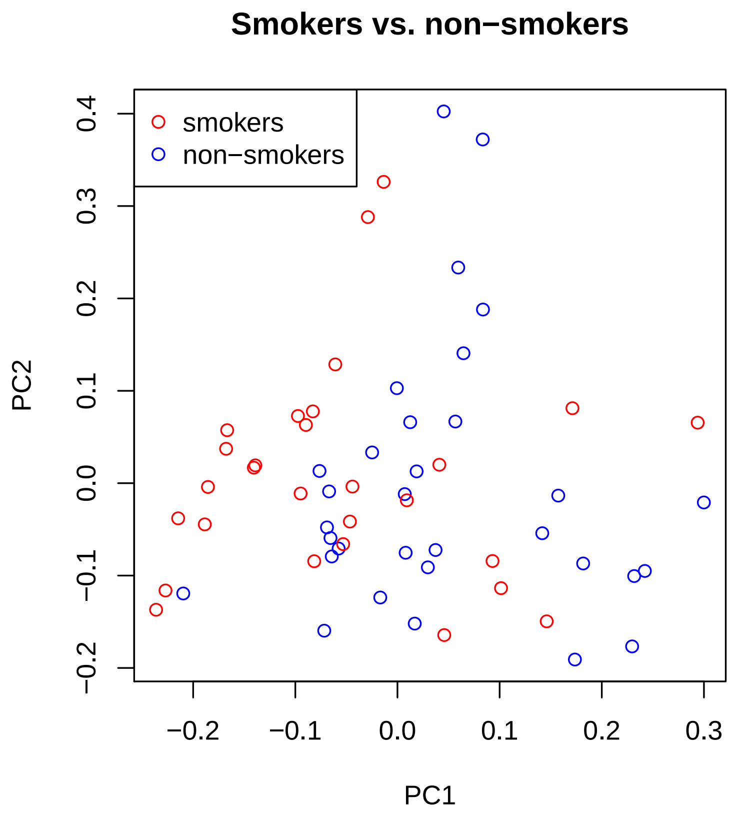

Figure 2 Ordination plot from analysis of the throat microbiome data after removing the effects of gender and antibiotic use. The red and blue circles represent smokers and nonsmokers, respectively.

depending on the study design. If all covariate values are the same for each cluster member and the goal is to generate permutations for which members of the same cluster are assigned the same values for the covariates in each permutation, then set `permute.between.clusters=TRUE`. For example, if some observations have been replicated, then these observations should always be assigned the same covariates in each permutation, and `permute.between.clusters=TRUE` should be used. If covariate values differ within clusters and the goal is to generation permutations for which members of the same cluster randomly switch values for the covariates in each permutation, then set `permute.within.clusters=TRUE`. For example, if data are pairs of observations from the same person before and after treatment, then a test of the effect of treatment effect should use `permute.within.clusters=TRUE`.

To illustrate the use of clustered sampling in `ldm`, we simulated a dataset based on the throat data. It consists of data from 40 individuals, of whom 20 contributed 2 observations and 20 contributed 3 observations (hence a total of 100 observations). Eight of the individuals contributing 2 observations and 12 of those contributing 3 observations were cases ($Y = 1$), and the remaining ones were controls ($Y = 0$), yielding a total of 52 observations from cases and 48 from controls. The OTU table contains 791 OTUs after excluding OTUs having no reads in any observation. Metadata include the case-control status (Y), two confounding covariates (C_1 and C_2), and a unique individual identifier (ID). See Hu and Satten (2018) 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)
```

To test the association of the microbiome with the case-control status Y while controlling for the confounders $C1$ and $C2$, 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` and `permute.between.clusters=TRUE` 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 | (C1+C2) ~ Y, data=sim.meta, dist.method="bray",
                  cluster.id=ID, permute.between.clusters=TRUE,
                  test.global=TRUE, test.otu=TRUE, fdr.nominal=0.1,
                  n.perm.max=5000, seed=seed)
> res4.ldm$n.perm.completed
[1] 5000
> res4.ldm$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] 45
> (otu.omni.var1 = colnames(res4.ldm$q.otu.omni)[w1])
 [1] "53"  "69"  "77"  "83"  "98"  "100" "101" "103" "109" "113" "120" "127" "148" "153"
[15] "162" "167" "174" "185" "199" "202" "248" "251" "280" "285" "319" "352" "370" "375"
[29] "407" "424" "546" "551" "571" "572" "611" "619" "665" "714" "716" "739" "744" "747"
[43] "767" "777" "787"
```

With 5000 permutations, the OTU-specific tests have met the stopping criterion and yielded 45 significant OTUs; although the global test has not met the stopping criterion of obtaining at least 10 rejections, we know the p-value is going to be very small such as 0.00019996 and there is no need to add more permutations to make it more precise.

8 References

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

Hu YJ, Satten GA. Testing hypotheses about microbiome using an ordination-based linear decomposition model. *bioRXiv.* 2018;doi.org/10.1101/229831.

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

Satten GA, Tyx RE, Rivera AJ, Stanfill S. Restoring the Duality between Principal Components of a Distance Matrix and Linear Combinations of Predictors, with Application to Studies of the Microbiome. *PloS One.* 2017;12(1):e0168131.