STAMPS 2018 - corncob Introduction

Bryan D Martin

Vignette Information

We thank Dr. Thea Whitman for kindly providing us with the example data set we use for this vignette. You can read more about this data in Whitman, Thea, et al. "Dynamics of microbial community composition and soil organic carbon mineralization in soil following addition of pyrogenic and fresh organic matter." *The ISME Journal* 10.12 (2016): 2918.

Introduction

Effectively modeling microbial relative abundance poses a number of statistical challenges, including:

- different sequencing depth,
- excessive zeros from unobserved taxa,
- high variability of empirical relative abundances (overdispersion),
- within-taxon correlation,
- hypothesis testing with categorical and continuous covariates.

Here, we introduce `corncob`, an individual taxon regression model that uses abundance tables and sample data. `corncob` is able to model differential abundance and differential variability, and addresses each of the challenges presented above.

Note that in order to follow along with this tutorial (but not to use `corncob`!) you will need to have `phyloseq` installed.

Install `corncob` using:

```
devtools::install_github("bryandmartin/CORNCOB")
```

To begin, we load our example data set as a `phyloseq` object.

```
library(corncob)
library(phyloseq)
library(dplyr)
data(soil_phylo)
```

If you are unfamiliar with `phyloseq`, we can view a description of the data using:

```
soil_phylo
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 7770 taxa and 119 samples ]
## sample_data() Sample Data:       [ 119 samples by 5 sample variables ]
## tax_table()   Taxonomy Table:    [ 7770 taxa by 7 taxonomic ranks ]
```

We now see that we have an OTU abundance table with 7770 OTUs and 119 samples. We can extract using `otu_table()`. Let's examine a small subset of our data in more detail.

```
otu_table(soil_phylo)[1:3, 1:3]
```

```
## OTU Table:          [3 taxa and 3 samples]
##                      taxa are rows
##         S009 S204 S112
## OTU.43   350   74  300
## OTU.2   1796 4204 1752
## OTU.187  280  709  426
```

We can also see that we have 5 sample variables. We can extract this using `sample_data()`. Let's again examine a small subset in more detail.

```
sample_data(soil_phylo)[1:3, ]
```

```
##      Plants DayAmdmt Amdmt ID Day
## S009      1       01     1  D   0
## S204      1       21     1  D   2
## S112      0       11     1  B   1
```

Our covariates are as follows:

- `Plants`: Indicator of whether plants are in the soil for this sample.
- `Amdmt`: Categoircal variable representing one of three soil additives; none, biochar, and freshbiomass, respectively.
- `ID`: Categorical variable representing different plots of soil.
- `Day`: Categorical variable representing one of three days of measurement; day 1, day 21, and day 81, respectively.

Finally, we have a taxonomy table with 7 taxonomic ranks.

```
tax_table(soil_phylo)[1:3, ]
```

```
## Taxonomy Table:     [3 taxa by 7 taxonomic ranks]:
##         Rank1         Rank2               Rank3
## OTU.43  "k__Bacteria" "p__Nitrospirae"    "c__Nitrospira"
## OTU.2   "k__Bacteria" "p__Proteobacteria" "c__Alphaproteobacteria"
## OTU.187 "k__Bacteria" "p__Acidobacteria"  "c__Acidobacteriia"
##         Rank4                 Rank5                  Rank6
## OTU.43  "o__Nitrospirales"    "f__Nitrospiraceae"    "g__Nitrospira"
## OTU.2   "o__Rhizobiales"      "f__Bradyrhizobiaceae" "g__Bradyrhizobium"
## OTU.187 "o__Acidobacteriales" "f__Koribacteraceae"   "g__"
##         Rank7
## OTU.43  "s__"
## OTU.2   "s__"
## OTU.187 "s__"
```

Fitting a Model

Now, let's set up our model.

First, let's subset our samples to only include those with the `DayAmdmt` covariate equal to 11 or 21 and then collapse the samples to the phylum level.

```
soil <- soil_phylo %>%
            phyloseq::subset_samples(DayAmdmt %in% c(11,21)) %>%
            tax_glom("Rank2")
```

Let's examine the data and the taxonomy table again.

```
soil
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 40 taxa and 32 samples ]
## sample_data() Sample Data:       [ 32 samples by 5 sample variables ]
## tax_table()   Taxonomy Table:    [ 40 taxa by 7 taxonomic ranks ]
```

```
tax_table(soil)[1:5, ]
```

```
## Taxonomy Table:     [5 taxa by 7 taxonomic ranks]:
##         Rank1          Rank2                 Rank3 Rank4 Rank5 Rank6 Rank7
## OTU.19  "k__Bacteria" "p__Acidobacteria"    NA    NA    NA    NA    NA
## OTU.1   "k__Bacteria" "p__Proteobacteria"   NA    NA    NA    NA    NA
## OTU.15  "k__Bacteria" "p__Gemmatimonadetes" NA    NA    NA    NA    NA
## OTU.3   "k__Bacteria" "p__Actinobacteria"   NA    NA    NA    NA    NA
## OTU.133 "k__Bacteria" "p__[Thermi]"         NA    NA    NA    NA    NA
```

Note that collapsing the samples is not necessary, and this model can work at any taxonomic rank. However, we will later be fitting a model to every taxa. We can see that by agglomerating taxa to the phylum level, we have gone from from 7770 to 40 taxa. Thus we collapse in order to increase the speed for the purposes of this tutorial.

Now we fit our model. We will demonstrate with Proteobacteria, or OTU.1.

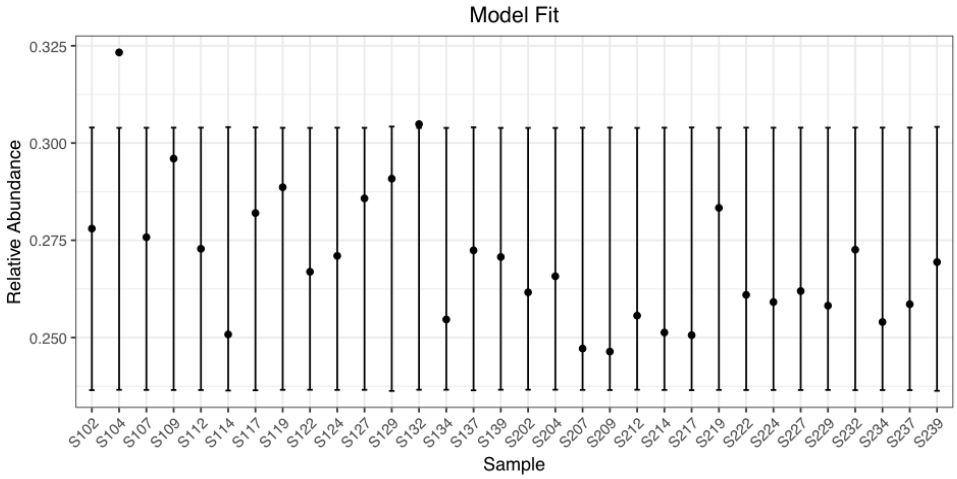
For now, we will not include any covariates, so we use `1` as our model formula responses.

```
corncob <- bbdml(formula = OTU.1 ~ 1,
             phi.formula = ~ 1,
             data = soil)
```

Interpreting a Model

First, let's plot the data with our model fit on the relative abundance scale. To do this, we simply type:

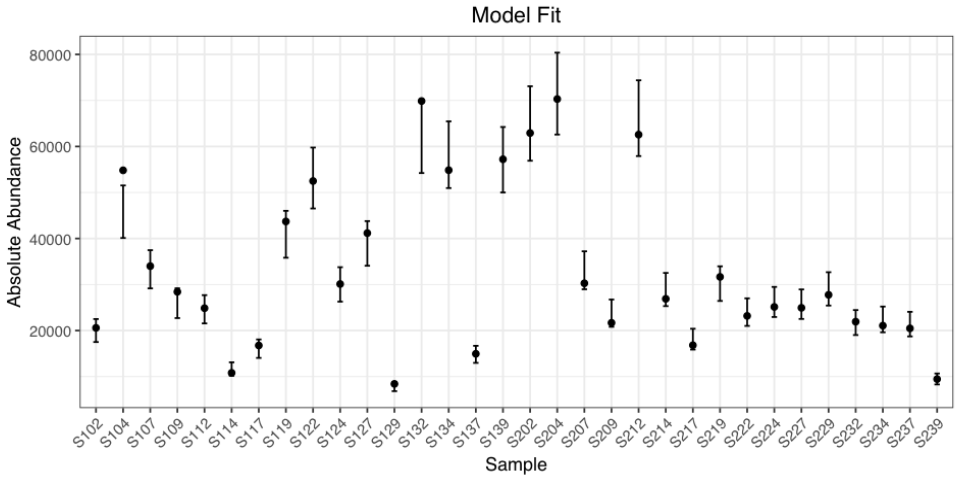
```
plot(corncob)
```



The points represent the relative abundances. The bars represent the 95% prediction intervals for the observed relative abundance by sample.

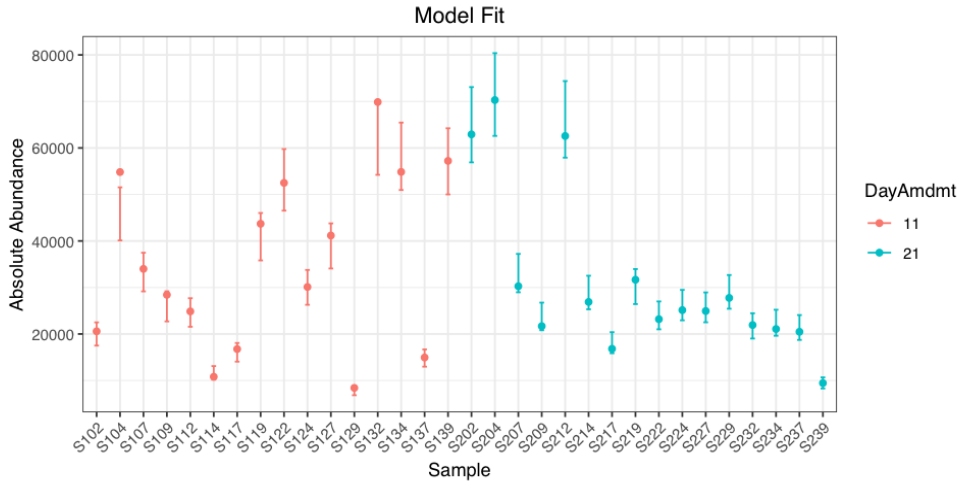
Now let's look at the same plot, but on the absolute abundance scale with 95% prediction intervals (since absolute abundance is not a parameter). To do this, we add the option `AA = TRUE` to our plotting code.

```
plot(corncob, AA = TRUE)
```

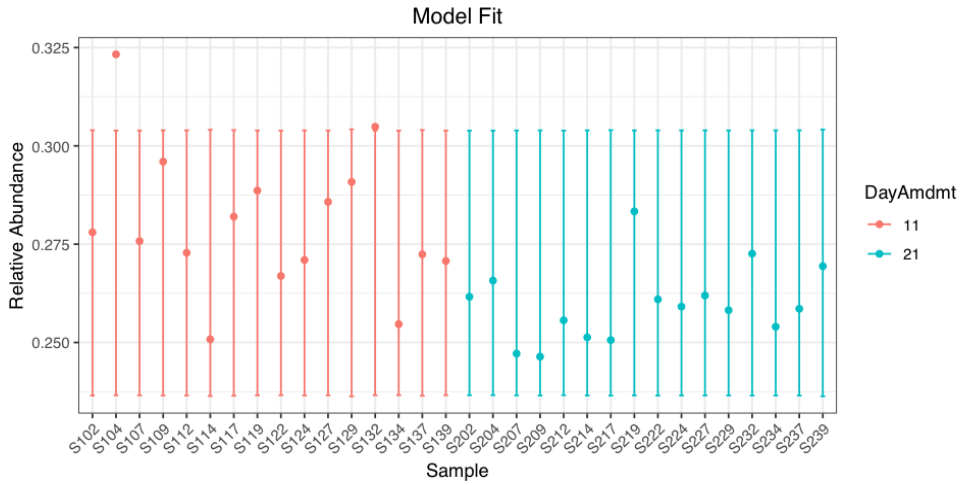


Finally, let's color the plot by the `DayAmdmt` covariate. To do this, we add the option `color = "DayAmdmt"` to our plotting code.

```
plot(corncob, AA = TRUE, color = "DayAmdmt")
```




```
plot(corncob, color = "DayAmdmt")
```



Notice that this plot also reorders our samples so that groups appear together so that they are easier to compare.

We can observe on this plot that it might be of interest to distinguish between the two groups with covariates. The average empirical relative abundance for the samples with `DayAmdmt = 21` tends to be lower and less variable than the samples with `DayAmdmt = 11`.

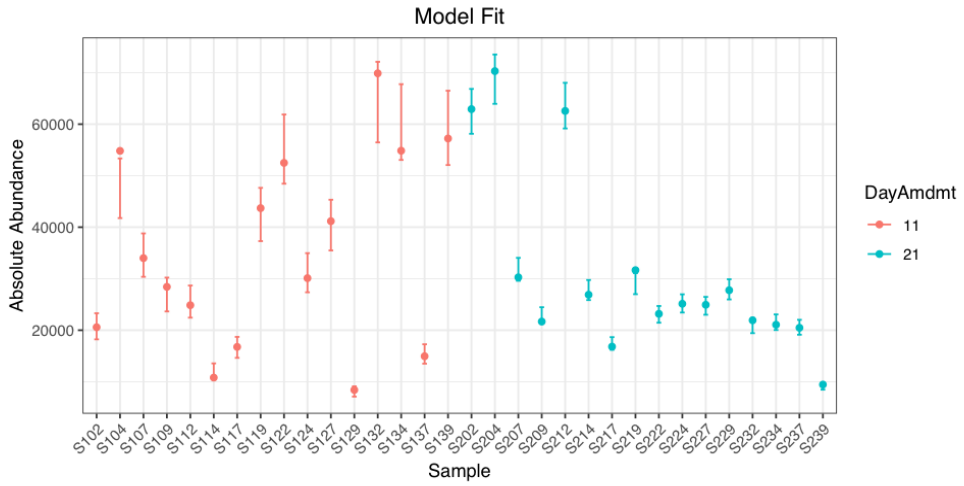
Adding covariates

Let's try modeling the expected relative abundance and the variability of the absolute abundance with `DayAmdmt` as a covariate. We do this by modifying `formula` and `phi.formula` as:

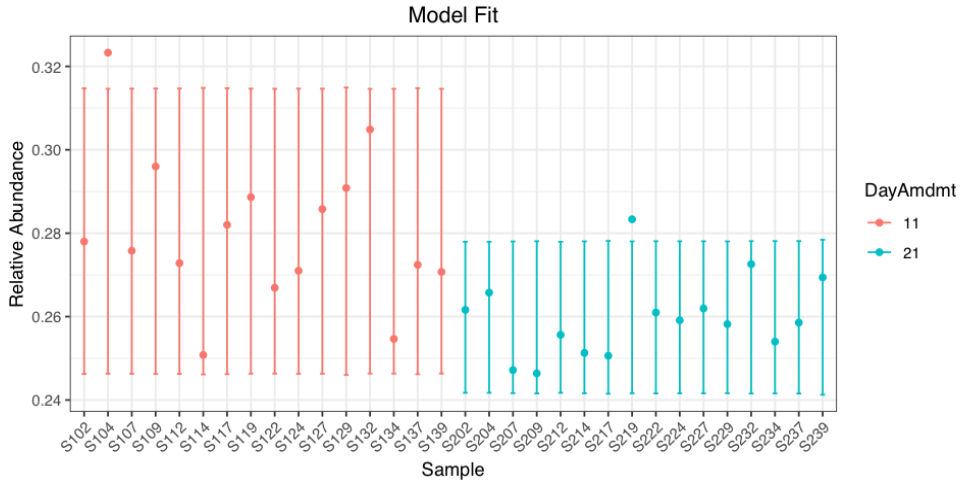
```
corncob_da <- bbdml(formula = OTU.1 ~ DayAmdmt,
             phi.formula = ~ DayAmdmt,
             data = soil)
```

Let's also plot this data on both the absolute abudance relative abundance scale.

```
plot(corncob_da, color = "DayAmdmt", AA = TRUE)
```




```
plot(corncob_da, color = "DayAmdmt")
```



Visually, the model with covariates seems to provide a much better fit to the data, but how can we compare

the two models statistically?

Model Selection

Let's use a likelihood ratio test to select our final model for this taxon. We want to test the null hypothesis that the likelihood of the model with covariates is equal to the likelihood of the model without covariates. To do this test, we use:

```
lrtest(corncob, corncob_da)
```

```
## [1] 6.892183e-05
```

We obtain a p-value much smaller than a cut-off of 0.05. Therefore we conclude that there is a statistically significant difference in the likelihood of the two models. Thus, we probably want to use the model with covariates for this taxon.

Parameter Interpretation

Now that we have chosen our model, let's interpret our model output. To see a summary of the model, type:

```
summary(corncob_da)
```

```
##
## Call:
## bbdml(formula = OTU.1 ~ DayAmdmt, phi.formula = ~DayAmdmt, data = soil)
##
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## mu.(Intercept)  -0.94299    0.02165 -43.550  < 2e-16 ***
## mu.DayAmdmt21   -0.10415    0.02481  -4.198 0.000246 ***
## phi.(Intercept) -6.49947    0.35563 -18.276  < 2e-16 ***
## phi.DayAmdmt21  -1.22849    0.50767  -2.420 0.022268 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Log-likelihood: -276.02
```

This output will look familiar if you have done regression analysis in R in the past. Covariates associated with the expected relative abundance are preceded by `mu.`; covariates associated with the variance of the absolute abundances are preceded by `phi.`.

From this model summary, we can see that the `mu.DayAmdmt21` coefficient is negative and statistically significant. This suggests that this taxon is differentially-abundant across `DayAmdmt`, and that samples with `DayAmdmt = 21` are expected to have a lower relative abundance. This matches what we saw from the observed abundances.

We can also see that the `phi.DayAmdmt21` coefficient is negative and statistically significant. This suggests that this taxon is differentially-variable across `DayAmdmt`, and that samples with `DayAmdmt = 21` are expected to have a lower variability. This matches what we saw from the observed abundances.

Analysis for Multiple Taxa

What if we want to test all the taxa in our data to see if they are differentially-abundant or differentially-variable? We use the `differentialTest` function. It will perform the above tests on all taxa, and it will control the false discovery rate to account for multiple comparisons.

Next, we use the `differentialTest` command. We specify the covariates of our model using `formula` and `phi.formula` as before, except we no longer include the response term because we are testing multiple taxa. We also specify which covariates we want to test for by removing them in the `formula_null` and `phi.formula_null` arguments.

The difference between the formulas and the null version of the formulas will be the variables that are tested. In this case, as when we examined the single taxon, we will be testing the coefficients of `DayAmdmt` for both the expected relative abundance and the overdispersion.

We set `fdr_cutoff` to be our controlled false discovery rate.

In this example, we also set `inits` argument. This is not necessary, but it will speed up our computation because the algorithm does not need to identify initializations for the parameters for each taxon.

```
set.seed(1)
fullAnalysis <- differentialTest(formula = ~ DayAmdmt,
                                 phi.formula = ~ DayAmdmt,
                                 formula_null = ~ 1,
                                 phi.formula_null = ~ 1,
                                 data = soil,
                                 fdr_cutoff = 0.05,
                                 inits = rbind(rep(.01, 4)))
```

We can see a list of differentially-abundant taxa using:

```
fullAnalysis$DA
```

```
##  [1] "OTU.1"    "OTU.15"   "OTU.8"    "OTU.69"   "OTU.6"    "OTU.59"
##  [7] "OTU.695"  "OTU.628"  "OTU.703"  "OTU.908"  "OTU.1570" "OTU.417"
## [13] "OTU.584"  "OTU.1672"
```

In this case, we identified 14 taxa that are differentially-abundant across `DayAmdmt` (out of the 40 taxa tested).

We can see a list of differentially-variable taxa using:

```
fullAnalysis$DV
```

```
## [1] "OTU.19"  "OTU.69"  "OTU.628" "OTU.703"
```

In this case, we identified 4 taxa that are differentially-variable across `DayAmdmt` (out of the 40 taxa tested).

We can examine a subset of the p-values of our tests using:

```
fullAnalysis$p[1:5,]
```

```
##                   DA          DV Warning
## OTU.19  5.866863e-01 0.001329089       0
## OTU.1   2.464226e-04 0.022268143       0
## OTU.15  6.502895e-08 0.914386057       0
## OTU.3   4.018381e-01 0.078358590       0
## OTU.133 1.143277e-01 0.823248631       0
```

where `DA` is the p-value associated with the test for differential abundance, `DV` is the p-value associated with the test for differential variance, and `warning` is a flag to let us know if the test wasn't able to complete for that OTU so that we may investigate further.

We can examine a subset of the p-values after controlling for the false discovery rate using:

```
fullAnalysis$p_fdr[1:5,]
```

```
##                   DA          DV
## OTU.19  8.090259e-01 0.008639077
## OTU.1   1.922096e-03 0.078950690
## OTU.15  1.690753e-06 0.977015238
## OTU.3   6.441430e-01 0.210757586
## OTU.133 2.918950e-01 0.957158223
```

where the columns are defined similarly as before, but the values are now adjusted to control the false discovery rate at 0.05.

Finally, we can see a list of any taxa for which we were not able to fit a model using:

```
fullAnalysis$warning
```

```
## [1] "OTU.4206"
```

In this case, we weren't able to fit `OTU.4206` automatically. It's worthwhile to investigate the OTU individually if this is the case. First let's check what phylum this represents.

```
tax_table(soil)["OTU.4206"]
```

```
## Taxonomy Table:     [1 taxa by 7 taxonomic ranks]:
##          Rank1         Rank2     Rank3 Rank4 Rank5 Rank6 Rank7
## OTU.4206 "k__Bacteria" "p__GN04" NA    NA    NA    NA    NA
```

It may be that the model is overparameterized because there aren't enough observations, or it may just be that the initializations were invalid for that taxa and it needs to be re-evaluated with new initializations.

Let's first try examining the data.

```
otu_table(soil)["OTU.4206"]
```

```
## OTU Table:          [1 taxa and 32 samples]
##                      taxa are rows
##          S204 S112 S134 S207 S202 S139 S122 S212 S117 S104 S214 S109 S217
## OTU.4206    0    0    0    0    0    0    0    0    0    0    0    0    0
##          S229 S132 S209 S227 S107 S237 S224 S127 S137 S114 S124 S119 S219
## OTU.4206    0    0    0    0    0    0    0    0    0    0    0    0    0
##          S232 S129 S102 S234 S222 S239
## OTU.4206    0    0    1    0    0    0
```

We see that the observed counts of OTU is zero in all samples except for `S102`, where we observed a single count. Let's try fitting the model individually by letting the model select the initializations automatically.

```
check_GN04 <- bbdml(formula = OTU.4206 ~ DayAmdmt,
                phi.formula = ~ DayAmdmt,
                data = soil)
```

While the model fits, we should be skeptical of **any** statisical model fit on a single observed count!

`corncob` is stable, but new. If you notice any issues, please log them on Github to help us help you!