Longitudinal analysis pipeline with longdat_cont()

Time (the proxy for treatment) as a continuous variable

Introduction

This is an example of running `longdat_cont()`. Note that the time variable (proxy of treatment) here should be continuous. If the time variable is discrete, please apply `longdat_disc()` instead.

```
# Load the packages
library(LongDat)
library(tidyverse)
library(kableExtra)
```

Explaining the input data frame format

The input data frame (called master table) should have the same format as the example data "LongDat_cont_master_table". If you have metadata and feature (eg. microbiome, immunome) data stored in separate tables, you can go to the section Preparing the input data frame with make_master_table() below. The function `make_master_table()` helps you to create master table from metadata and feature tables.

Now let's have a look at the required format for the input master table. The example below is a dummy longitudinal data set with 2 time points (day 0 and 7). Here we want to see if the treatment has a significant effect on gut microbial abundance or not.

```
# Read in the data frame. LongDat_cont_master_table is already lazily loaded.
master <- LongDat_cont_master_table
master %>%
    kableExtra::kbl() %>%
    kableExtra::kable_paper(bootstrap_options = "responsive", font_size = 12) %>%
    kableExtra::scroll_box(width = "700px", height = "200px")
```

Individual	Day	sex	age	DrugA	DrugB	BacteriumA	BacteriumB	BacteriumC
1	0	0	61	0.0	10	11	4	26
1	7	0	61	0.0	10	13	2	22

Individual	Day	sex	age	DrugA	DrugB	BacteriumA	BacteriumB	BacteriumC
2	0	0	66	0.0	640	344	0	6

As you can see, the “Individual” is at the first column, and the features (dependent variables), which are gut microbial abundances in this case, are at the end of the table. Any column apart from individual, test_var (e.g. Day) and dependent variables will be taken as potential confounders (confounding with the test_var). For example, here the potential confounders are sex, age, drug A and drug B. **Please avoid using characters that don’t belong to ASCII printable characters for the column names in the input data frame.**

Preparing the input data frame with make_master_table()

If you have your input master table prepared already, you can skip this section and go to Run longdat_cont() directly. If your metadata and feature (eg. microbiome, immunome) data are stored in two tables, you can create a master table out of them easily with the function `make_master_table()`.

First, let’s take a look at an example of the metadata table. Metadata table should be a data frame whose columns consist of sample identifiers (sample_ID, unique for each sample), individual, time point and other meta data. Each row corresponds to one sample_ID.

```
# Read in the data frame. LongDat_cont_metadata_table is already lazily loaded.
metadata <- LongDat_cont_metadata_table
metadata %>%
    kableExtra::kbl() %>%
    kableExtra::kable_paper(bootstrap_options = "responsive", font_size = 12) %>%
    kableExtra::scroll_box(width = "700px", height = "200px")
```

Sample_ID	Individual	Day	sex	age	DrugA	DrugB
1_0	1	0	0	61	0.0	10
1_7	1	7	0	61	0.0	10
2_0	2	0	0	66	0.0	640
2_7	2	7	0	66	0.0	320
3_0	3	0	0	63	7.5	100
3_7	3	7	0	63	7.5	0

This example is a dummy longitudinal meatadata with 2 time points for each individual. Besides sample_ID, individual, day columns, there are also information of sex, age and drugs that individuals take. Here we want to see if the treatment has a significant effect on gut microbial abundance or not.

Then, let’s see how a feature table looks like. Feature table should be a data frame whose columns only consist of sample identifiers (sample_ID) and features (dependent variables, e.g. microbiome). Each row corresponds to one sample_ID. Please do not include any columns other than sample_ID and features in the feature table.

```
# Read in the data frame. LongDat_cont_feature_table is already lazily loaded.
feature <- LongDat_cont_feature_table
feature %>%
    kableExtra::kbl() %>%
    kableExtra::kable_paper(bootstrap_options = "responsive", font_size = 12) %>%
    kableExtra::scroll_box(width = "700px", height = "200px")
```

Sample_ID	BacteriumA	BacteriumB	BacteriumC

Sample_ID	BacteriumA	BacteriumB	BacteriumC
1_0	11	4	26
1_7	13	2	22
2_0	344	0	6
2_7	3	0	670

This example is a dummy longitudinal feature data. It stores the gut microbial abundance of each sample.

To enable the joining process of metadata and feature tables, please pay attention to the following rules.

1. The row numbers of metadata and feature tables should be the same.
2. Sample_IDs are unique for each sample (i.e. no repeated sample_ID)
3. Metadata and feature tables have the same sample_IDs. If sample_IDs don't match between the two tables, the joining process will fail.
4. As mentioned above, feature table should include only the columns of sample_ID and features.
5. Avoid using characters that don't belong to ASCII printable characters for the column names.

Now let's create a master table and take a look at the result!

```
master_created <- make_master_table(metadata_table = LongDat_cont_metadata_table,
    feature_table = LongDat_cont_feature_table, sample_ID = "Sample_ID", individual =
        "Individual")
#> [1] "Finished creating master table successfully!"

master_created %>%
    kableExtra::kbl() %>%
    kableExtra::kable_paper(bootstrap_options = "responsive", font_size = 12) %>%
    kableExtra::scroll_box(width = "700px", height = "200px")
```

Individual	Day	sex	age	DrugA	DrugB	BacteriumA	BacteriumB	BacteriumC
1	0	0	61	0.0	10	11	4	26
1	7	0	61	0.0	10	13	2	22
2	0	0	66	0.0	640	344	0	6
2	7	0	66	0.0	320	3	0	670
3	0	0	63	7.5	100	55	0	10
3	7	0	63	7.5	0	5	0	111

The table "master_created" is just the same as the table "master" or "LongDat_cont_master_table" in the previous section, with the "Individual" as the first column, and the features (dependent variables), which are gut microbial abundances in this case, are at the end of the table. Any column apart from individual, test_var (e.g. Day) and dependent variables will be taken as potential confounders (confounding with the test_var). For the details of the arguments, please read the help page of this function by using `?make_master_table`.

OK, now we're ready to run `longdat_cont()`!

Run longdat_cont()

The input is the example data frame LongDat_cont_master_table (same as "master" or "master_created" in the previous sections), and the data_type is "count" since the dependent variables (features, in this case they're gut microbial abundance) are count data. The "test_var" is the independent variable you're testing, and here we're testing "Day" (time as the proxy for treatment). The variable_col is 7 because the dependent variables start at column 7. And the fac_var mark the columns that aren't numerical. For the details of the arguments, please read the help page of this function by using `?longdat_cont`.

The run below takes less than a minute to complete. When data_type equals to “count”, please remember to set seed (as shown below) so that you’ll get reproducible randomized control test.

```
# Run longdat_cont() on LongDat_cont_master_table
set.seed(100)
test_cont <- longdat_cont(input = LongDat_cont_master_table, data_type = "count",
    test_var = "Day", variable_col = 7, fac_var = c(1, 3))
#> [1] "Start data preprocessing."
#> [1] "Finish data preprocessing."
#> [1] "Start selecting potential confounders."
#> [1] 1
#> [1] 2
#> [1] 3
#> [1] 1
#> [1] 2
#> [1] 3
#> [1] "Finished selecting potential confounders."
#> [1] "Start null model test."
#> [1] 1
#> [1] 2
#> [1] 3
#> [1] "Finish null model test."
#> [1] "Start confounding model test."
#> [1] 1
#> [1] 2
#> [1] 3
#> [1] "Finish confounding model test."
#> [1] "Start unlisting tables from confounding model result."
#> [1] "Finish unlisting tables from confounding model result."
#> [1] "Finished post-hoc correlation test."
#> [1] 1
#> [1] 2
#> [1] 3
#> [1] "Finished post-hoc correlation test."
#> [1] "Start randomized negative control model test."
#> [1] 1
#> [1] 2
#> [1] 3
#> [1] 1
#> [1] 2
#> [1] 3
#> [1] "Finish randomized negative control model test."
#> [1] "Start removing the dependent variables to be exlcuded."
#> [1] "Finish removing the dependent variables to be exlcuded."
#> [1] "Start generating result tables."
#> [1] "Finished successfully!"
```

If you have completed running the function successfully, you’ll see the message “Finished successfully!” at the end. The results are stored in list format.

Results

The major output from `longdat_cont()` include a result table and a confounder table. If you have count data (data_type equals to “count”), then there are chances that you get a third table “randomized control table”. For more details about the “randomized control table”, please read the help page of this function by using `?longdat_cont`.

Result table

Let's have a look at the result table first.

```
# The first dataframe in the list is the result table
result_table <- test_cont[[1]]
result_table %>%
    kableExtra::kbl() %>%
    kableExtra::kable_paper(bootstrap_options = "responsive", font_size = 12, position =
        "center") %>%
    kableExtra::scroll_box(width = "700px")
```

Feature	Prevalence_percentage	Mean_abundance	Signal	Effect	EffectSize	Null_time_model_q	Post-hoc_q
BacteriumA	90	39.50	OK_nc	Decreased	-0.7809864	0.0000011	0.0000482
BacteriumB	45	34.75	NS	NS	-0.1328821	0.4496104	0.5765105
BacteriumC	90	84.00	OK_nc	Enriched	0.7112976	0.0012725	0.0004375

The second and third columns show the prevalence and mean abundance of each feature According to the "Signal" column, treatment is a significant predictor for BacteriumA and BacteriumC as they show "OK_nc" (which represents OK and no confounder), meaning that the abundance of BacteriumA and BacteriumC alter significantly through time (proxy of treatment), and that there is no potential confounder. If there is confounding effect in the result, please see the confounder table to find out what the confounders are. As for BacteriumB, time (proxy of treatment) has no effect on its abundance.

The following column "Effect" describes the trend of dependent variables change along time. Here we can tell that BacteriumA and BacteriumC have decreasing and increasing patterns, respectively. From the next column "EffectSize", we know that the effect sizes are -0.78 and 0.71, respectively. The important and the most relevant information for users ends here, which are listed from the first column to "EffectSize".

Then the following columns contain the details of model test p values ("Null_time_model_q"), the post-hoc test p values (Post.hoc_q). For more detailed information of the columns in the result table, please refer to the help page by using `?longdat_cont`.

The explanation of each type of "Signal" is listed below.

Signal	Meaning	Explanation
NS	Non-significant	There's no effect of time.
OK_nc	OK and no confounder	There's an effect of time and there's no potential confounder.
OK_d	OK but doubtful	There's an effect of time and there's no potential confounder, however the confidence interval of the test_var estimate in the model test includes zero, and thus it is doubtful. Please check the raw data (e.g. plot feature against time) to confirm if there is real effect of time.
OK_sd	OK and strictly deconfounded	There are potential confounders, however there's an effect of time and it is independent of those of confounders.
AD	Ambiguously deconfounded	There are potential confounders, and it isn't possible to conclude whether the effect is resulted from time or confounders.
C	Confounded	There's an effect of time, but it can be reduced to the confounding effects.

Confounder table

Next, let's take a look at the confounder table.

```
# The second dataframe in the list is the confounder table
confound_table <- test_cont[[2]]
confound_table %>%
    kableExtra::kbl() %>%
    kableExtra::kable_paper(bootstrap_options = "responsive", font_size = 12, position =
        "center") %>%
    kableExtra::scroll_box(width = "700px")
```

Feature	Confounder1	Confounding_type1	Effect_size1

The columns of this confounder table are grouped every three columns. "Confounder1" is the name of the confounder, while "Confounding_type1" is the confounding type of confounder1, and "Effect_size1" is the effect size of the dependent variable values between different levels of confounder1. If there are more than one confounders, they will be listed along the rows of each dependent variable. Since there is no confounding effect found in this example (according to the result table), the confounder table is a blank. If you'd like to see a result with confounders, please read the vignette of `longdat_disc()`.

Result interpretation

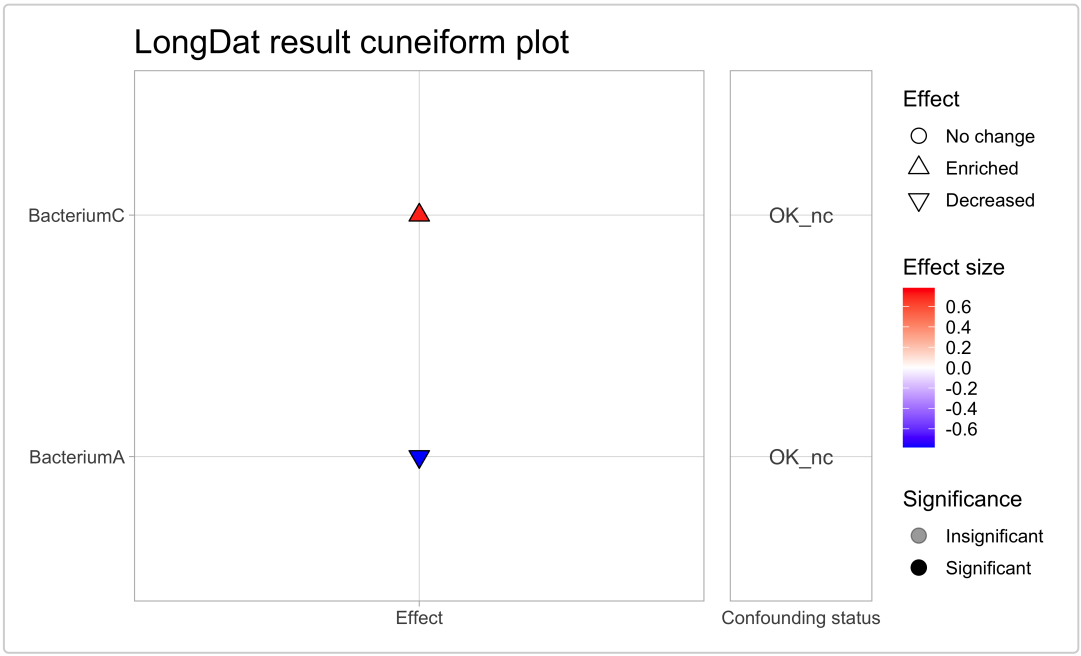
From the result above, we see that the treatment induces significant changes on the abundance of BacteriumA and BacteriumC, while causing no alteration in that of BacteriumB.

Plotting the result

Finally, we can plot the result with the function `cuneiform_plot()`. The required input is a result table from `longdat_cont()` (or any table with the same format as a result table does).

```
test_plot <- cuneiform_plot(result_table = test_cont[[1]], title_size = 15)
#> [1] "Finished plotting successfully!"
```

```
test_plot
```



Here we can see the result clearly from the cuneiform plot. It shows the features whose signals are not “NS”. The left panel displays the effects in each time interval. Red represents positive effect size while blue describes negative one (colors can be customized by users). Signficant signals are indicated by solid shapes, whereas insignificant signals are denoted by transparent ones. The right panel displays the confounding status of each feature, and users can remove it by specifying `confound_panel = FALSE`. For more details of the arguments, please read the help page of this function by using `?cuneiform_plot`.

Wrap-up

This tutorial ends here! If you have any further questions and can’t find the answers in the vignettes or help pages, please contact the author (Chia-Yu.Chen@mdc-berlin.de).