Compute LM associations functions

# Input parameters:

|  |  |
| --- | --- |
| DF (metadata) | A screenshot of text  Description automatically generated |
| Metaphlan\_df (CAGS) |  |

1. fudge\_factor <- min((metaphlan\_df %>% select(-1, -2) %>% unlist())[which((metaphlan\_df %>% select(-1, -2) %>% unlist()) > 0)])

* The fudge factor adjusts the metaphlan data so that the lowest number isn’t a zero (because you can’t run a regression on zero)
* (I couldn’t see a visible change between the input and after fudge, I’m sure there was I just didn’t see it and was too lazy to look)

1. for (j in seq\_along(metaphlan\_df %>% select(-1, -2))) {

* Iterate through the columns of metaphlan DF (excepting the first two columns because those are dataset name and sample\_id)
* J is the index of the feature (CAG) (column)

1. regression\_df=left\_join(as.data.frame(x) %>% mutate\_if(is.factor, as.character), metaphlan\_df %>% mutate(dataset\_name=str\_replace(dataset\_name, "gene\_families\_", "")) %>% select(c(1,2, (j + 2))),by = c("dataset\_name", "sampleID")) %>% select(-dataset\_name) %>% mutate\_if(is.character, as.factor) %>% drop\_na()

* This is one of the most important lines of the code! It joins the DF and the one column (feature) of the Metaphlan DF on a left join (making sure to convert factors to characters), replacing any instance of “gene\_families” with “”, and replacing by dataset name and Sample ID). Once it’s merged, it removes the dataset name and drops anything that is NA

1. return(tryCatch(tidy(stats::lm(as.formula(str\_c("I(log10(`", new\_feature\_name, "` + ", toString(fudge\_factor), ")) ~ .")), data = regression\_df))

* This calculates the linear regression for the regression table that you’ve made! Tidy() summarizes it into a nice table, lm() is the linear regression function), new\_feature\_name is the column name of the CAG).
* To use the lm function in here, here is an example of the format:
  + lm\_result = lm(territory.area~bite, data=data)
  + lm\_result = lm(**dependent (response**)~ **independent (explanatory)**, data = data)
  + Here in the code above, we have many independent variables (all the columns), so we just use “.” To indicate all the columns are explanatory variables
* In our study, the independent variable is the presence of T2D or not, and the dependent variable is the CAG
* The single output of this line of code returns a table like the one below:

|  |
| --- |
| * term estimate std.error statistic p.value * <chr> <dbl> <dbl> <dbl> <dbl> * 1 (Intercept) -3.98 0.495 -8.04 6.98e-10 * 2 study\_condition 0.499 0.342 1.46 1.52e- 1 * 3 age 0.173 0.262 0.662 5.12e- 1 * 4 age\_categorysenior -0.220 0.523 -0.421 6.76e- 1 * 5 gendermale 0.320 0.367 0.872 3.88e- 1 * 6 number\_reads 0.0530 0.165 0.321 7.50e- 1 * 7 BMI 0.0363 0.165 0.221 8.26e- 1 * 8 shannon\_diversity\_idx -0.459 0.197 -2.33 2.49e- 2 * 9 genus\_count 0.981 0.212 4.63 3.82e- 5 |

* Std. Error is Residual Standard Error (see below) divided by the square root of the sum of the square of that particular x variable.
* Statistic: Estimate divided by Std. Error
* If P is low, reject H0!

## Output of this function

* A screenshot of a cell phone

  Description automatically generatedThe output of this function is a table that looks like:
* On the first row, third column (feature\_1), the linked dataset is the one shown above as the output of the regression.
* The column “model\_selected\_df” is the curated metadata for each dataset