Conditional-eQTL-meta-analysis

Introduction

This Bioinformatics project tries to answer the question, "Is it possible to do eQTL analysis without the SuSiE Method?". Therefore, we will instead do all-but-one conditional analysis on the SIGLEC14 gene and compare those results at the end with the results from the regular methods. This analysis is done in the following 6 steps:

For steps 1-5 (Conditional eQTL analysis) we use the tool `tensorqtl`, wich is a python package. These steps were run on HPC. For step 6 (Colocalisation) we used R tools.

Required data types:

1. Phenotypes: BED format
2. Genotypes: PLINK format
3. Covariates: a tab-delimited text file (covariates x samples) or dataframe (samples x covariates), with row and column headers

Step 1: Perform normal eQTL analysis for SIGLEC14 gene, testing all common (MAF > 1%) variants in +/- 1Mb window around the promoter of the gene.

Normal eQTL analysis

A normal eQTL analysis typically involves testing for genetic variants that are associated with changes in gene expression levels (expression quantitative trait loci or eQTLs) across the genome, without any assumptions about the relationship between the genetic variant and the gene's location. So, no other variables (other variants as covariants) are used.

The term "eQTL" stands for expression quantitative trait loci, and it refers to genetic variants (usually single nucleotide polymorphisms, or SNPs) that are associated with differences in gene expression levels.

1.1. Create phenotypes file.

tensorqtl requires phenotype data be in .bed or .bed.gz format. Phenotype file must contain #Chr, start, end, TargetID columns in the mentioned order.

To create such a file, we processed and merged normalised gene expression data and covariates data from different files: chromosome number, phenotype position (start, and end is start+1) were taken from one file, and TargetIds from another file.

1.2. TensorQTL analysis.

Tensorqtl provides 2 analyses: cis-eQTL analysis and trans-eQTL analysis.

cis-eQTL analysis is used to identify genetic variants that affect the expression level of genes located nearby (usually within 1 Mb distance from the transcription start site(TSS)).

So, **cis-eQTL analysis** is the type of analysis we use for steps 1-4. Parameters used: maf_threshold=0.01, window=1000000. Besides, we run the analysis only for the SIGLEC14 (ENSG00000254415) gene.

```
eQTL_result_df = cis.map_cis(genotype_df, variant_df, phenotype_df.loc[phenotype_pos_df.index=='ENSG00000254415'],
phenotype_pos_df, maf_threshold=0.01, window=1000000)
```

Step 2: Identify the most strongly associated variant (lead variant).

Lead variant is the variant with the lowest p value.

By default, only the lead variant is displayed as output after the tensorqtl cis mapping. Thus, after the first step we get the following dataframe with the lead variant (chr19_51627384_T_A):

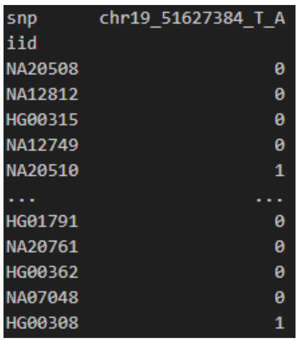
phenotype_id	num_var	beta_shape1	beta_shape2	true_df	pval_true_df	variant_id	tss_distance	ma_samples	ma_count	a
ENSG00000254415	14863	1.03784	2284.05127	397.486603	4.515546e-09	chr19_51627384_T_A	-19442	105	111	0

Step 3. Perform conditional eQTL analysis for SIGLEC14 by adding the lead variant as a covariate into the model, and

Step 4: Repeat this until no significant (p < 1e-5) associations remain.

For these steps, we need to create a covariant dataframe. Dataframe row indexes must be sample ids, and each column - variant id. The values are the genotype dosages (0, 1, or 2) for the variant in each individual (sample id).

```
cov_df = genotype_df.loc[eQTL_SIGLEC14_df['variant_id']].T
```



```
snp      chr19_51627384_T_A
iid
NA20508                   0
NA12812                   0
HG00315                   0
NA12749                   0
NA20510                   1
...                     ...
HG01791                   0
NA20761                   0
HG00362                   0
NA07048                   0
HG00308                   1
```

Having such a dataframe, we run the same cis-mapping tensorqtl method with maf_threshold=0.01, window=1000000 parameters and also add the covariant dataframe as a parameter: covariates_df=cov_df.

```
eQTL_result_df = cis.map_cis(genotype_df, variant_df, phenotype_df.loc[phenotype_pos_df.index=='ENSG00000254415'],
                                     phenotype_pos_df, covariates_df=cov_df, maf_threshold=0.01, window=1000000)
```

After getting results, we repeat the process: we take the most strongly associated variant, add it to the covariants dataframe, run the conditional eQTL analysis. We stop when the p-value becomes >= 1e5 (that means that no significant associations remain).

```
    while eQTL_SIGLEC14_df['pval_nominal'].iloc[0] < 1e-5:
        # Performing conditional eQTL analysis
        eQTL_result_df = cis.map_cis(genotype_df, variant_df, phenotype_df.loc[phenotype_pos_df.index=='ENSG00000254415'],
                                     phenotype_pos_df, covariates_df=cov_df, maf_threshold=0.01, window=1000000)

        # Extracting results for the SIGLEC14 gene
        eQTL_SIGLEC14_df = eQTL_result_df[eQTL_result_df.index=='ENSG00000254415']

        # Extracting covariates and updating the covariates dataframe
        cov_df_temp = genotype_df.loc[eQTL_SIGLEC14_df['variant_id']].T
        cov_df = pd.merge(cov_df, cov_df_temp, left_index=True, right_index=True)
```

In this step, we found 2 more variants with a significant association: chr19_51612691_C_G and chr19_50966950_G_GT.

Step 5: Identify all-but-one conditionally indepedent summary statistics.

All-but-one conditional independent eQTL analysis is used to evaluate the independence of each variant's association with gene expression after conditioning on the other variants in the haplotype (a group of correlated variants).

In the previous steps we identified 3 variants with significant association (top 3 lead variants): chr19_51627384_T_A, chr19_51612691_C_G, chr19_50966950_G_GT.

In this step, conditioning (conditional eQTL analysis) has to be performed three times:

1. for Signal 1 condition on Signals 2 and 3 (add both lead SNPs as covariates).
2. for Signal 2, condition on Signals 1 and 3.
3. for Signal 3, condition on Signals 1 and 2.

, where signal 1 is chr19_51627384_T_A, signal 2 is chr19_51612691_C_G, and signal 3 is chr19_50966950_G_GT.

All-but-one conditional eQTL analysis is similar to that we used in the Step 3. The only difference is that we use different combinations of variants (1+2, 1+3, 2+3) as described above.

In addition, there is a difference of what we need to get in the result. In previous step we needed to identify variants with the strongest association among all of them. Now, we need to know all variants despite the association significance (all 14863 variants for the SIGLEC14 gene).

cis_map function used in the Step 3 prints out only the lead variant, and there is no option to show all variants. Thus, in this step we have to utilize map_trans method. Since that it's behaviour is different, we were forced to perform extra work:

```
    # Extract 2 and 3 variants
    signals_2_3_df = covariants_df[['chr19_51612691_C_G', 'chr19_50966950_G_GT']]

    # Run trans eQTL mapping for the SIGLEC14 gene only
    # No p_val threshold to obtain results for all variants
    eQTL_result_df = trans.map_trans(genotype_df, phenotype_df.loc[phenotype_pos_df.index=='ENSG00000254415'],
                                     covariates_df=signals_2_3_df, pval_threshold=1, maf_threshold=0.01)

    # Extrant all non-cis regions for the SIGLEC14 gene
    trans_eQTL_result_df = trans.filter_cis(eQTL_result_df,
                                            phenotype_pos_df.loc[phenotype_pos_df.index=='ENSG00000254415'].T.to_dict(),
                                            variant_df, window=1000000)
    # Remove all non-cis regions from the mapping result
    # to get only cis mappings for the SIGLEC14 gene
    merged_df = eQTL_result_df.merge(trans_eQTL_result_df, indicator=True, how='left')
    cis_result_df = merged_df[merged_df['_merge'] == 'left_only'].drop(columns='_merge')
```

At first, we perform trans eQTL analysis that covers all variants (within and without 1Mb window). At second, to remove non-cis associations, we identify them, and remove from the eQTL analysis result. In this way, we obtain a dataframe with the following structure (only first variant as a example):

variant_id	phenotype_id	pval	b	b_se	af
chr19_50646870_G_A	ENSG00000254415	0.824844196111419	-0.01832097	0.08273168	0.16741572

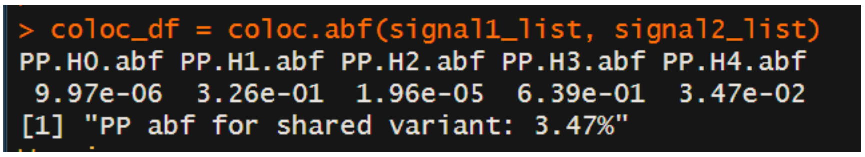
In total, in this step we gain 3 dataframes with the structure above, each has 14863 elements. These dataframes are used in the colocalization in the Step 6.

Step 6: Colocalisation

This step is done in R with help of the package "coloc". Here we try to find out, if our data actually correlates and colocolized with the data produced by the SuSiE method. Therefore we do the following:

First we have to transform each of our 3 dataframes (Further referred as signal 1 - 3) in a specific format, so that the colocalisation Methods can read them. After that we can run the colocalisation with the coloc.abf() Method on signal 1 and signal 2. This was done like this:

```
#Run coloc.abf

signal1_list = list(beta = signal1$b,

                    varbeta = signal1$b_se^2,

                    N = rep(445, length(signal1$b)),

                    MAF = pmin(signal1$af, 1-signal1$af),

                    snp = signal1$variant_id,

                    type = "quant")


coloc_df = coloc.abf(signal1_list, signal2_list)

labf_df = dplyr::transmute(coloc_df$results, variant = snp, labf_variable1 = lABF.df1, labf_variable2 = lABF.df2) %>%

  dplyr::as_tibble()
```

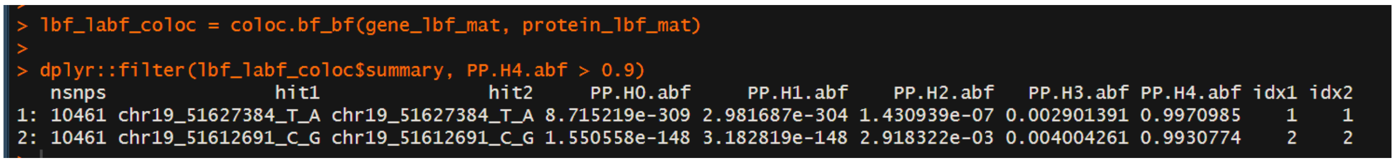


```
> coloc_df = coloc.abf(signal1_list, signal2_list)
PP.H0.abf PP.H1.abf PP.H2.abf PP.H3.abf PP.H4.abf
 9.97e-06  3.26e-01  1.96e-05  6.39e-01  3.47e-02
[1] "PP abf for shared variant: 3.47%"
```

To follow up we do another colocalisation between our results from the first colocalisation and the original Siglec14 data, to compare the LBFs from SuSiE with the LABFs of our data:

```
#Make protein lbf matrix

protein_lbf_mat = as.matrix(dplyr::select(siglec14_df, lbf_variable1:lbf_variable10))

row.names(protein_lbf_mat) = siglec14_df$variant

protein_lbf_mat = t(protein_lbf_mat)


#Make gene labf variable matrix

gene_lbf_mat = as.matrix(dplyr::select(labf_df, labf_variable1:labf_variable2))

row.names(gene_lbf_mat) = labf_df$variant

gene_lbf_mat = t(gene_lbf_mat)


#Perform colocalisation
```

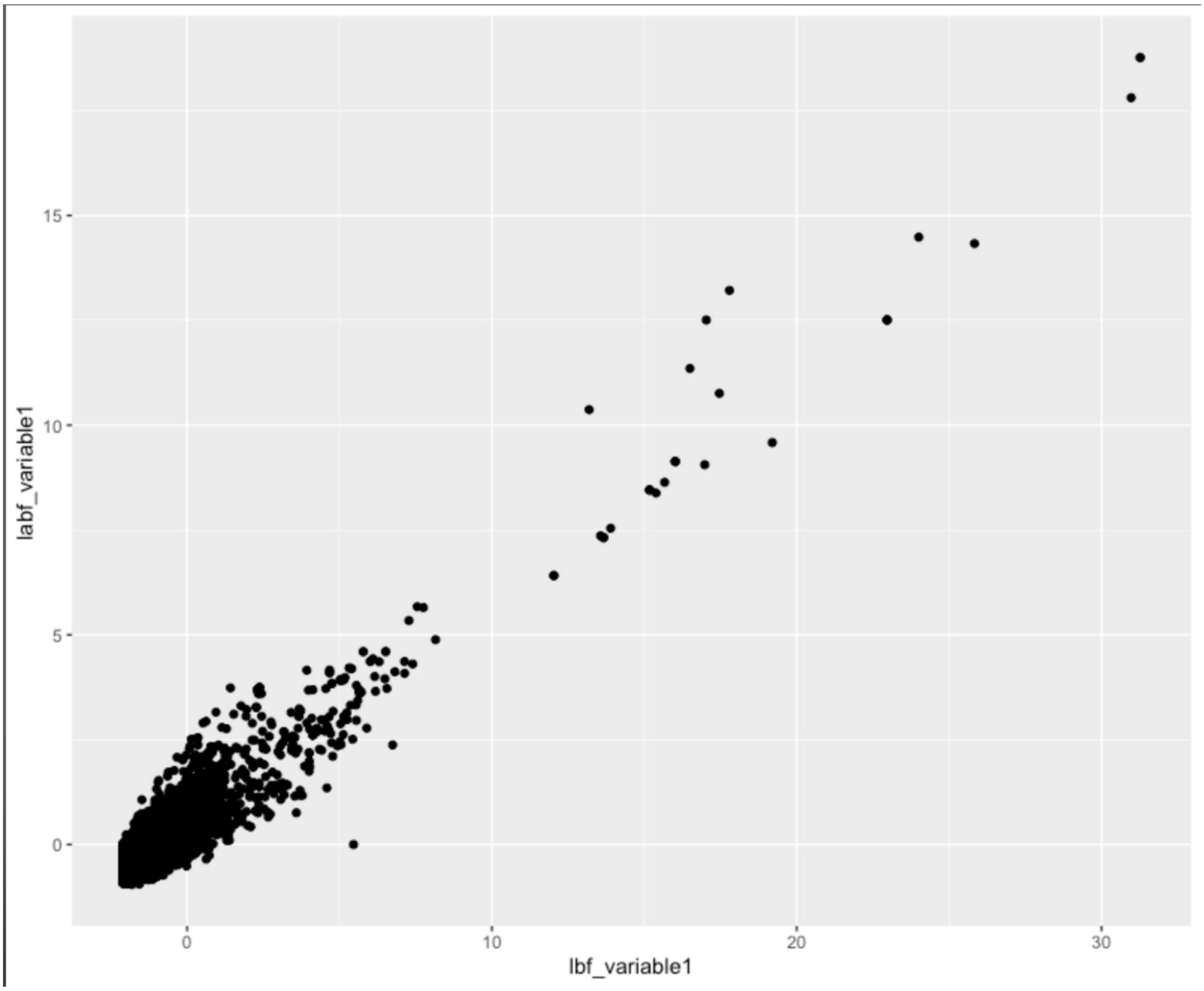
```
lbf_labf_coloc = coloc.bf_bf(gene_lbf_mat, protein_lbf_mat)

dplyr::filter(lbf_labf_coloc$summary, PP.H4.abf > 0.9)
```

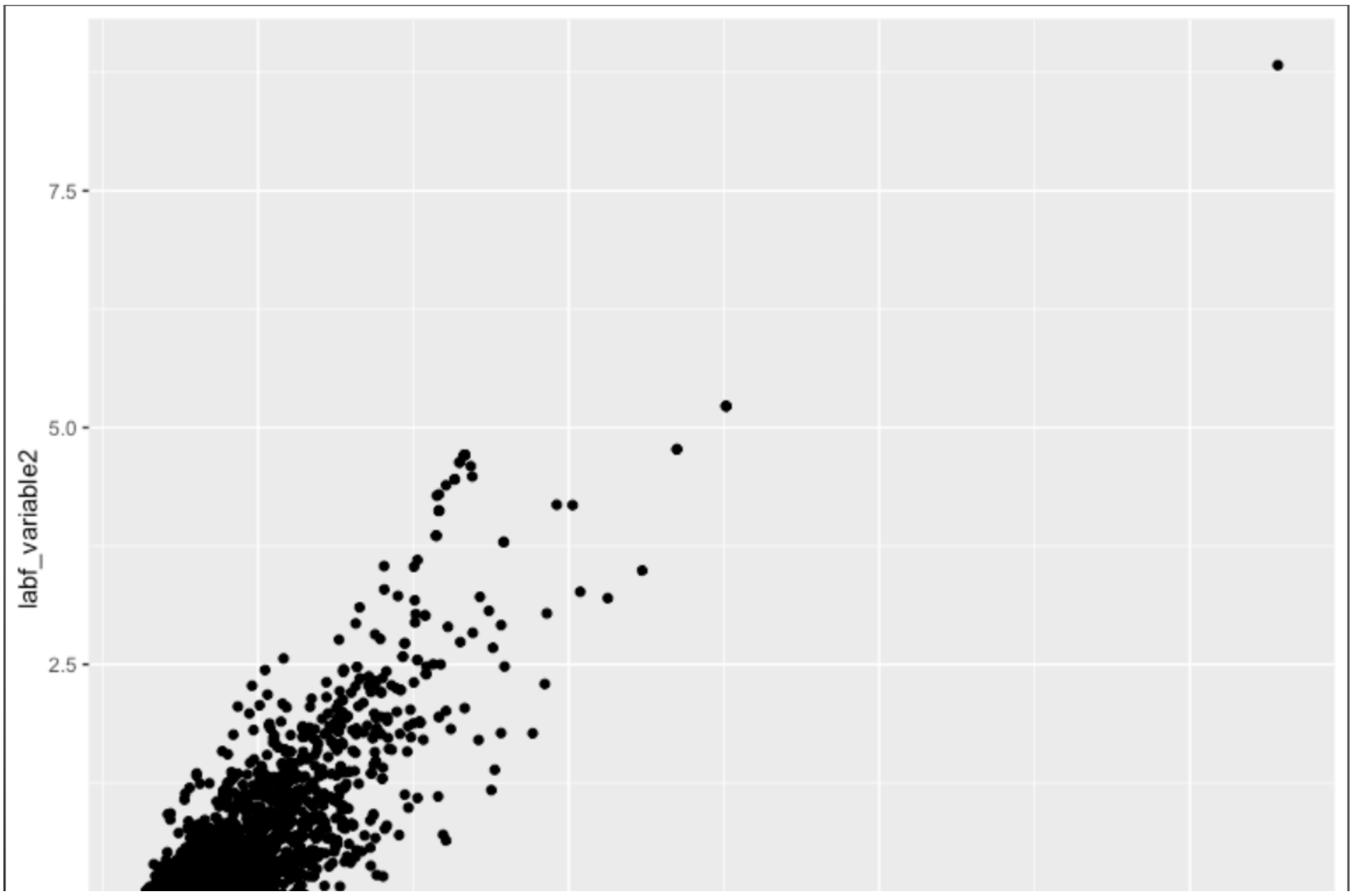


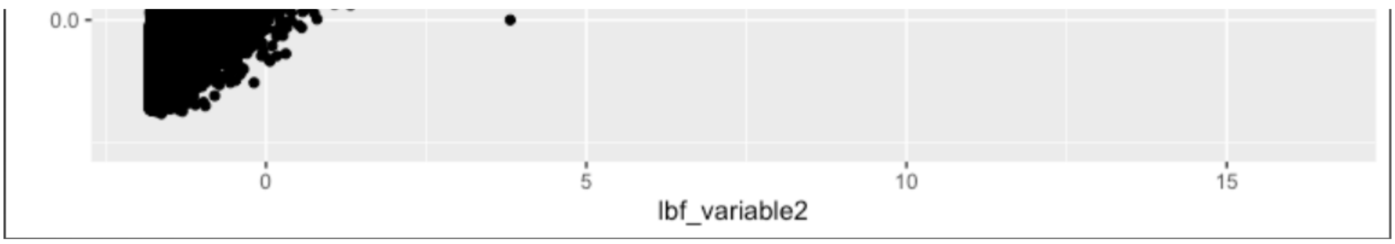
```
> lbf_labf_coloc = coloc.bf_bf(gene_lbf_mat, protein_lbf_mat)
>
> dplyr::filter(lbf_labf_coloc$summary, PP.H4.abf > 0.9)
   nsnps               hit1               hit2     PP.H0.abf     PP.H1.abf    PP.H2.abf   PP.H3.abf PP.H4.abf idx1 idx2
1: 10461 chr19_51627384_T_A chr19_51627384_T_A 8.715219e-309 2.981687e-304 1.430939e-07 0.002901391 0.9970985    1    1
2: 10461 chr19_51612691_C_G chr19_51612691_C_G 1.550558e-148 3.182819e-148 2.918322e-03 0.004004261 0.9930774    2    2
```

And finally we can do scatterplots to further analyse our results and see if they correlate. First we compare the SuSiE LBFs vs. our calculated LABFs:

```
#Visualise LABF vs LBF

full_df = dplyr::left_join(siglec14_df, labf_df, by = "variant")

ggplot(full_df, aes(x = lbf_variable1, y = labf_variable1)) + geom_point()

ggplot(full_df, aes(x = lbf_variable2, y = labf_variable2)) + geom_point()
```

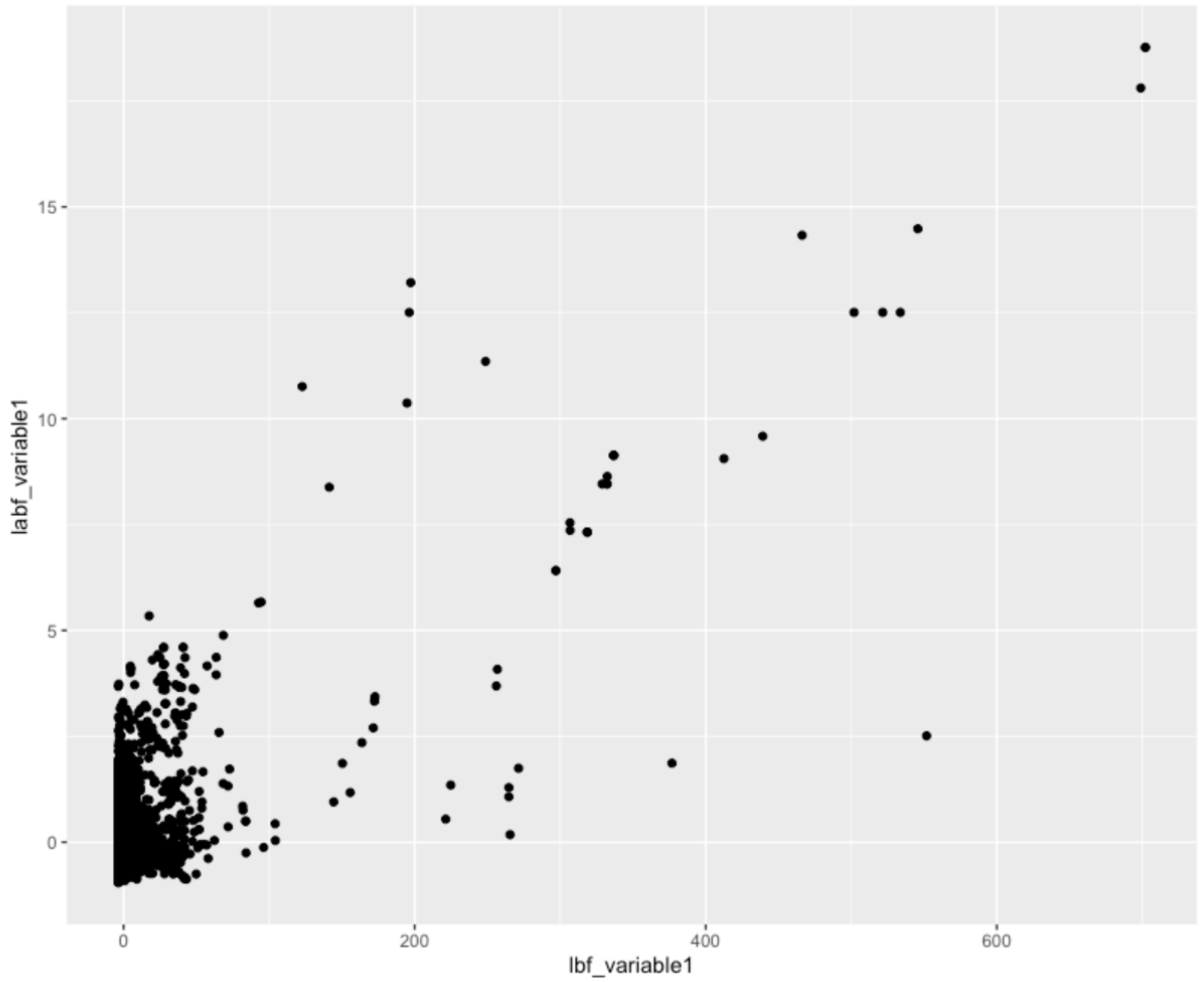

labf_variable1
lbf_variable1
0
5
10
15
0
10
20
30

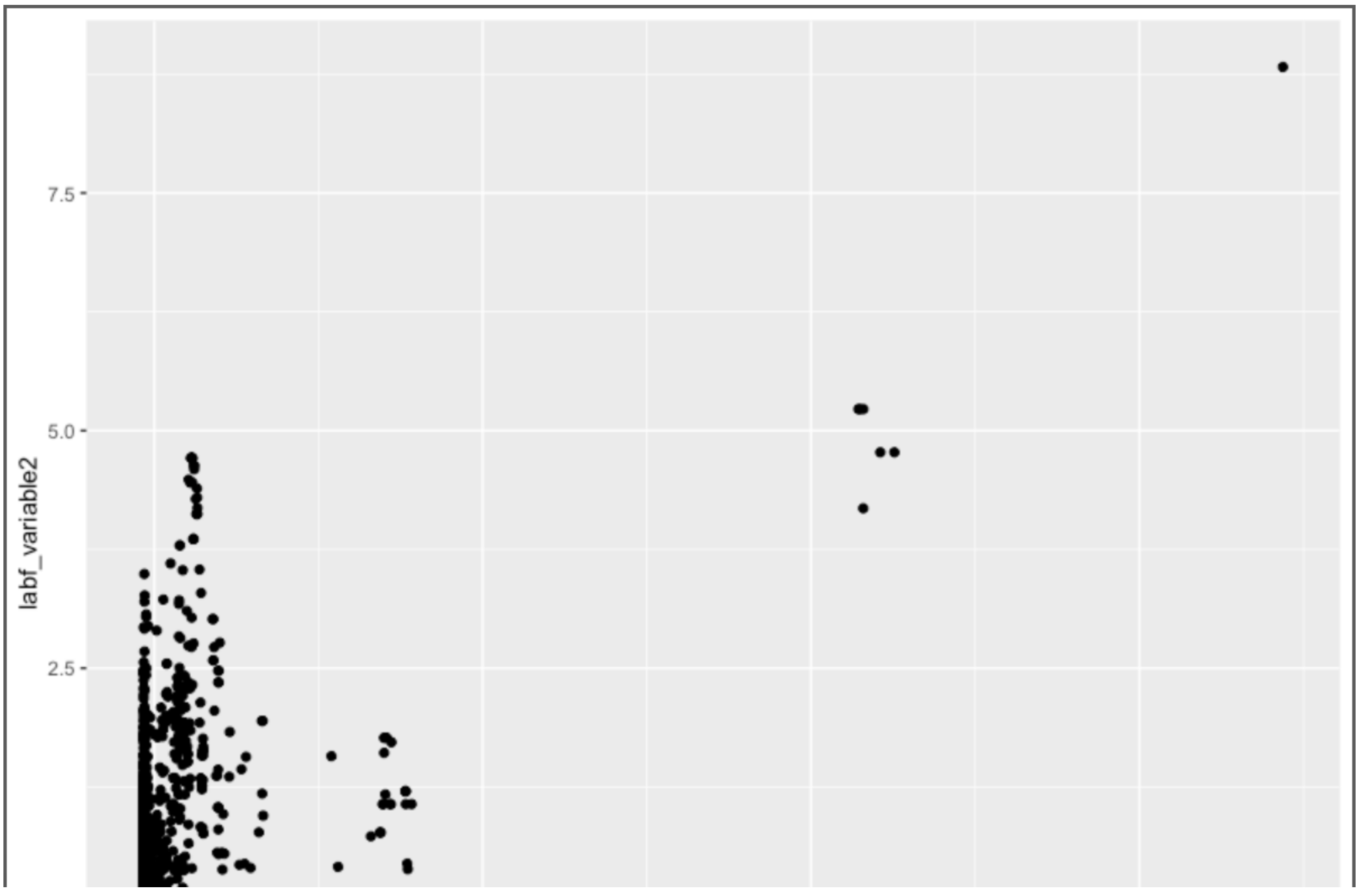

labf_variable2
2.5
5.0
7.5

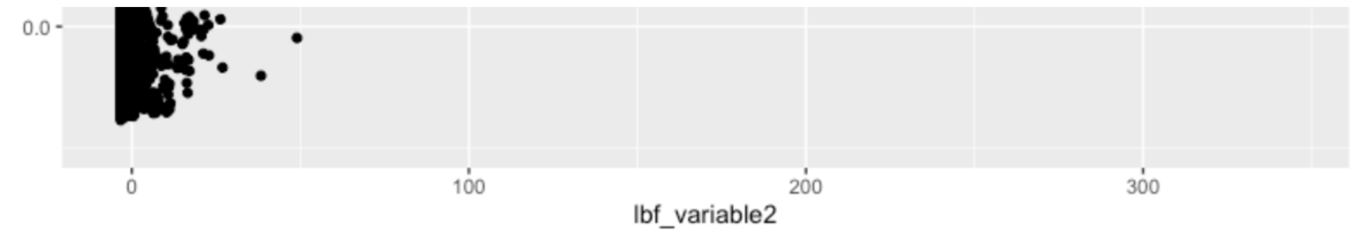


Then we compare the Protein LBFs vs. our calculated LABFs:

```
#Import gene expression lbfs

geuvadis_lbf_data = readr::read_tsv("R/Bioinformatics_project_data/step6/QTD000110.lbf_variable.txt.gz")

siglec14_eqtl_lbf = dplyr::filter(geuvadis_lbf_data, molecular_trait_id == "ENSG00000254415")


#Visualise gene expression LBF vs LABF

full_df2 = dplyr::left_join(siglec14_eqtl_lbf, labf_df, by = "variant")

ggplot(full_df2, aes(x = lbf_variable1, y = labf_variable1)) + geom_point()

ggplot(full_df2, aes(x = lbf_variable2, y = labf_variable2)) + geom_point()

ggplot(full_df2, aes(x = lbf_variable1, y = labf_variable2)) + geom_point()
```


labf_variable1
lbf_variable1
15
10
5
0
0
200
400
600


labf_variable2
7.5
5.0
2.5



Results

As one can see in the results from step 6, the correlation is pretty high. Both scatterplots (SIGLEC14 LBFs vs. our calculated LABFs and Protein LBFs vs. our calculated LABFs) nearly look identical and we also found a clear evidence of colocalisation with the PP.h4 values very close to 1.

These results show that our lead question can actually be answered with yes, and it is possible to replace the SuSiE method with all-but-one conditional analysis.

Discussion

This found obviously needs to be analysed more, but from this example alone it looks quite promising, so that maybe one day this method could be a standard at analyzing data, when the SuSiE method is not possible (e.g. in meta-analysis).

The main struggle turned out to be is the tensorqtl tool. We figured out that it is impossible to use it in Windows platforms. Moreover, documentation is quite short and not fully user-friednly. We would recommend to use tensorqtl to a person with some basic understanding of programming.

Author contribution

Yuliia Siur - eQTL analysis (steps 1-5)

Yannis Fynn Meyer - Colocalization (step 6)

Both - preliminary data analysis, presentation