Replication of mQTLs associations across contexts

Olivia Castellini-Pérez

October 30, 2023

Contents

1 Methods

1.1 Samples

Model	Cell type	N
1 & 2	All	1086
3	Neu	158
3	Mono	141
3	Tcell	159
3	Bcell	100
4 & 5	Neu	75
4 & 5	Mono	60
4 & 5	Tcell	70
4 & 5	Bcell	45

1.2 Models

(1) n=all, data=bulk, model=interaction, cell-type=x4(salas), region=cis, cell-count=predicted
(2) n=all, data=bulk, model=interaction, cell-type=x4(salas), region=cis, cell-count=observed
(3) n=all, data=cell-specific, model=main, cell-type=x4(sorted), region=cis
(4) n=overlap(~60), data=cell-specific, model=main, cell-type=x4(sorted), region=cis
(5) n=overlap(~60), data=bulk, model=interaction, cell-type=x4(the same as sorted), region=cis

2 Replication

1. Are the GoDMC SNP-CpG pairs replicating in bulk+int?
2. Are the GoDMC SNP-CpG pairs replicating in cell-specific?
3. Discovery in bulk+int replicating in cell-specific?

4. Discovery in cell-specific replicating in bulk?
5. Any pvalues from mod1 or mod2 with FDR < 0.05?

True positives: 1. Get cell-specific associations in model 3 - use FDR < 0.05 for heterogeneity test 2. Look to see if they replicate in predicted cell counts (model 1) False positives: 1. Get cell-specific associations in predicted cell counts (model 1) 2. Replication in model 3

2.1 GoDMC SNP-CpG pairs replication

3 Results

3.1 Relationship between GoDMC and cell-sorted model effects

3.1.1 Linear regression between GoDMC and m4 effects

```
##   CellType       Beta        SE       PValue
## 1      neu  0.5442842 0.1259401 3.143802e-05
## 2     mono  0.6746450 0.1298362 8.112924e-07
## 3    tcell  0.2116132 0.3391897 5.338532e-01
## 4    bcell -0.6400704 0.9553857 5.041390e-01
```

3.1.2 Overall replication rate

```
##   CellType PropTable.Var1 PropTable.Freq
## 1      neu          FALSE      0.1507937
## 2      neu           TRUE      0.8492063
## 3     mono          FALSE      0.3968254
## 4     mono           TRUE      0.6031746
## 5    tcell          FALSE      0.5396825
## 6    tcell           TRUE      0.4603175
## 7    bcell          FALSE      0.6800000
## 8    bcell           TRUE      0.3200000
```

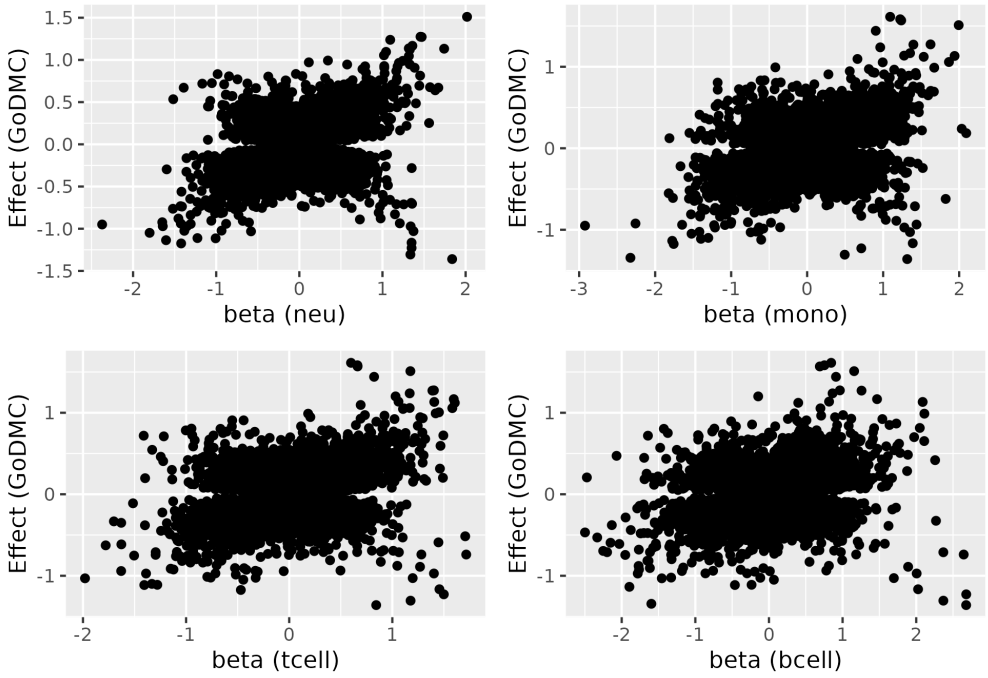


Figure 1: GoDMC effect versus model 4 effect

3.1.3 Replication rate (observed vs expected)

	CellType	Model	Repl.nsnp	Repl.metric	Repl.datum	Repl.value
1	neu	Replication Rate	126	Sign	Expected	119.931402
2	neu	Replication Rate	126	Sign	Observed	83.000000
3	neu	Replication Rate	126	P-value	Expected	10.333876
4	neu	Replication Rate	126	P-value	Observed	34.000000
5	mono	Replication Rate	126	Sign	Expected	113.701772
6	mono	Replication Rate	126	Sign	Observed	89.000000
7	mono	Replication Rate	126	P-value	Expected	2.839605
8	mono	Replication Rate	126	P-value	Observed	18.000000
9	tcell	Replication Rate	126	Sign	Expected	111.289722
10	tcell	Replication Rate	126	Sign	Observed	72.000000
11	tcell	Replication Rate	126	P-value	Expected	14.214326
12	tcell	Replication Rate	126	P-value	Observed	18.000000
13	bcell	Replication Rate	125	Sign	Expected	101.659835
14	bcell	Replication Rate	125	Sign	Observed	74.000000
15	bcell	Replication Rate	125	P-value	Expected	4.672418
16	bcell	Replication Rate	125	P-value	Observed	7.000000

3.1.4 Plots

3.2 Heterogeneity test for each cell in model 3

3.2.1 Cell-specific associations for netrophils model 3

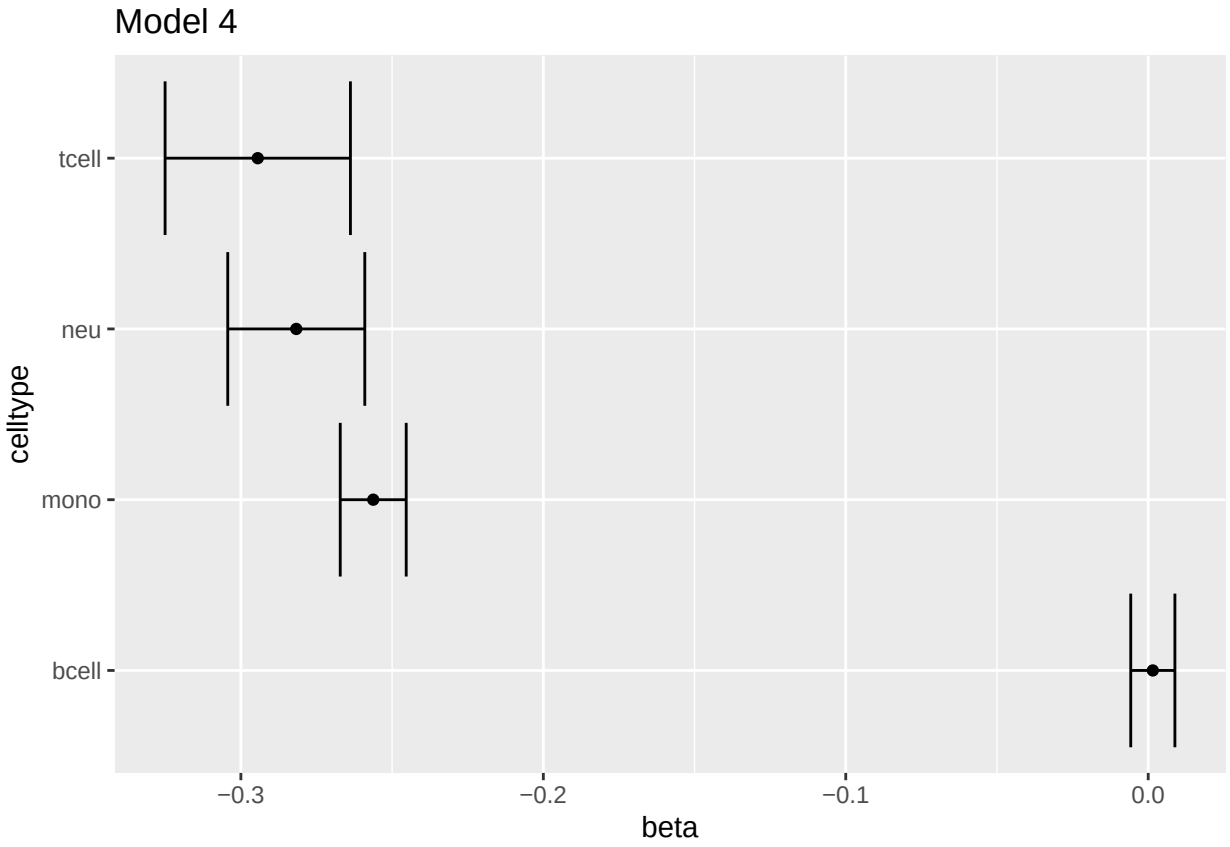
```
## # A tibble: 23,973 x 7
## # Groups:   code [23,973]
```

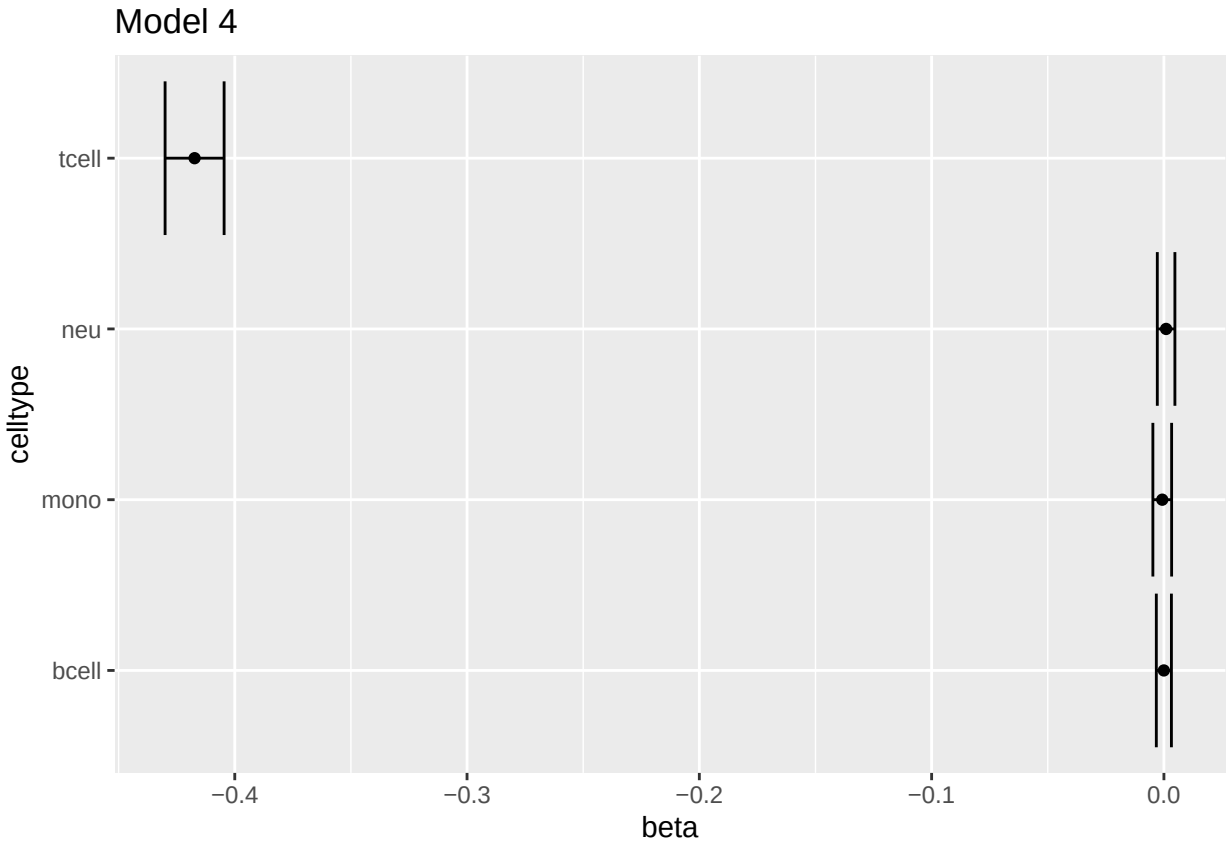
```
##    code                          beta      se      Q   Qdf   Qpval    fdr
##    <chr>                        <dbl>   <dbl>  <dbl> <dbl>   <dbl>  <dbl>
##  1 1:100001201_G_T cg17157234 -0.0315  0.00364  4.75      3 0.191   0.339
##  2 1:100015837_A_G cg05108386  0.00770 0.00167  4.22      3 0.239   0.397
##  3 1:100015837_A_G cg23599391  0.00582 0.00111 12.1       3 0.00695 0.0244
##  4 1:10004882_C_T cg15685931  -0.00538 0.00361  6.95      3 0.0737  0.165
##  5 1:100065568_A_G cg07721612 -0.0150  0.00544  5.54      3 0.136   0.264
##  6 1:100079353_A_C cg06661172  0.00334 0.00213  0.488     3 0.921   0.963
##  7 1:100081347_C_G cg00673963 -0.00138 0.00115  7.17      3 0.0668  0.153
##  8 1:100084101_A_G cg17310086 -0.00106 0.00164  1.63      3 0.653   0.781
##  9 1:10009114_C_T cg15226751   0.00742 0.00153  5.46      3 0.141   0.271
## 10 1:100125121_C_T cg08874824 -0.0159  0.00270  0.256     1 0.613   0.751
## # i 23,963 more rows
```

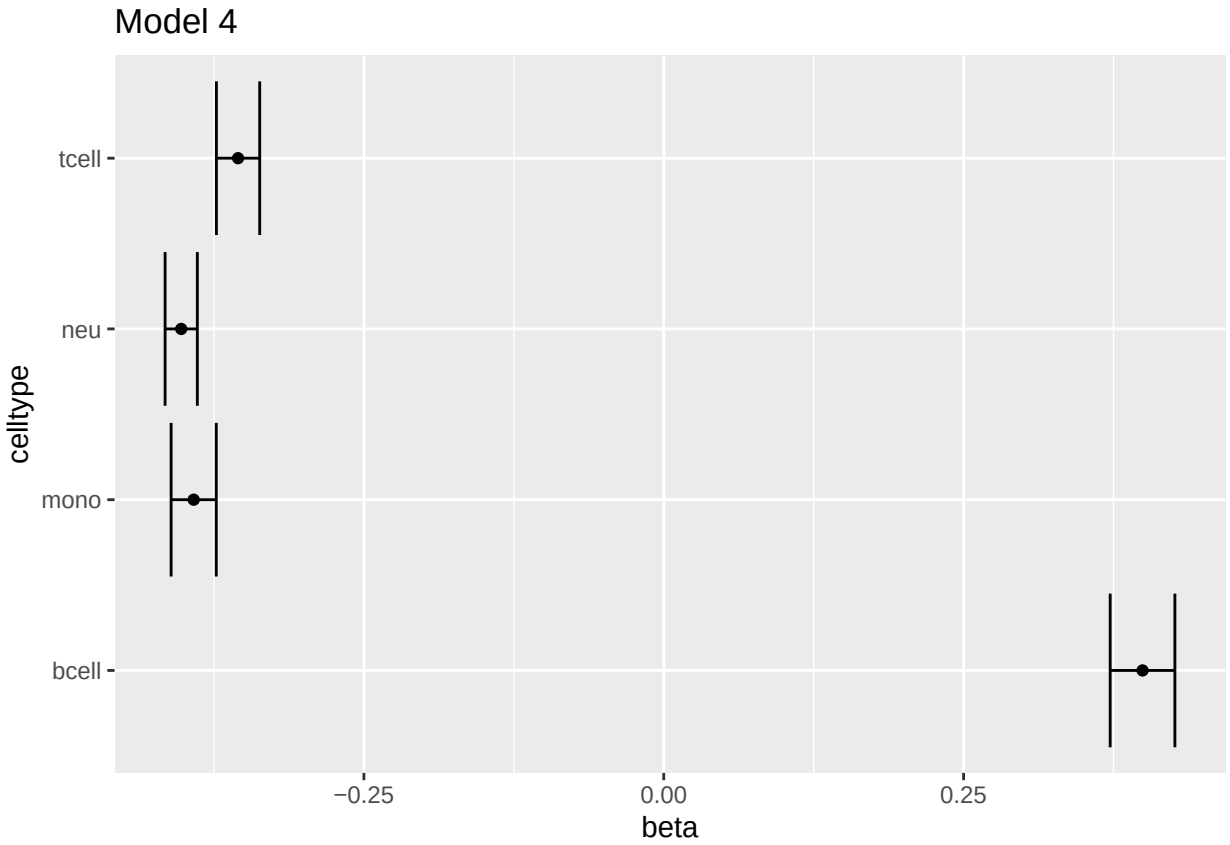
```
## # A tibble: 93,914 x 3
## # Groups:   code [23,973]
##    code                       celltype qjpval
##    <chr>                      <chr>     <dbl>
##  1 1:100001201_G_T cg17157234 neu      0.973
##  2 1:100001201_G_T cg17157234 mono     0.652
##  3 1:100001201_G_T cg17157234 tcell    0.0992
##  4 1:100001201_G_T cg17157234 bcell    0.177
##  5 1:100015837_A_G cg05108386 neu      0.152
##  6 1:100015837_A_G cg05108386 mono     0.394
##  7 1:100015837_A_G cg05108386 tcell    0.265
##  8 1:100015837_A_G cg05108386 bcell    0.658
##  9 1:100015837_A_G cg23599391 neu      0.832
## 10 1:100015837_A_G cg23599391 mono     0.147
## # i 93,904 more rows
```

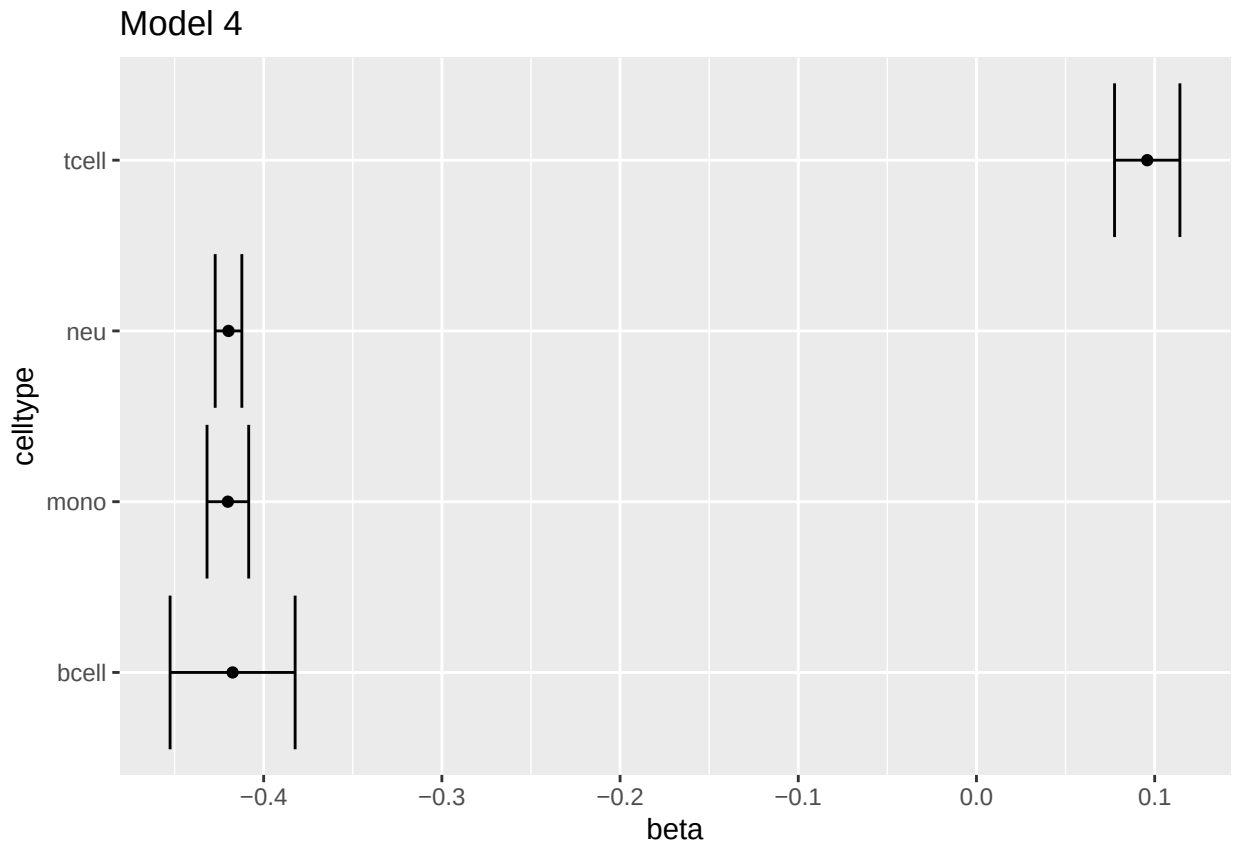
```
## # A tibble: 23,973 x 4
## # Groups:   code [23,973]
##    code                       celltype  qjpval cg_id
##    <chr>                      <chr>      <dbl> <chr>
##  1 1:100001201_G_T cg17157234 tcell    0.0992  cg17157234
##  2 1:100015837_A_G cg05108386 neu      0.152   cg05108386
##  3 1:100015837_A_G cg23599391 tcell    0.00194 cg23599391
##  4 1:10004882_C_T cg15685931  neu      0.0538  cg15685931
##  5 1:100065568_A_G cg07721612 tcell    0.0238  cg07721612
##  6 1:100079353_A_C cg06661172 mono     0.703   cg06661172
##  7 1:100081347_C_G cg00673963 bcell    0.0419  cg00673963
##  8 1:100084101_A_G cg17310086 tcell    0.366   cg17310086
##  9 1:10009114_C_T cg15226751  bcell    0.0574  cg15226751
## 10 1:100125121_C_T cg08874824 bcell    0.668   cg08874824
## # i 23,963 more rows
```

```
##
## FALSE  TRUE
## 16002  7971
```


Model 4
tcell
neu
mono
bcell
celltype
−0.3
−0.2
−0.1
0.0
beta


Model 4
tcell
neu
mono
bcell
celltype
−0.4
−0.3
−0.2
−0.1
0.0
beta


Model 4
tcell
neu
mono
bcell
celltype
−0.25
0.00
0.25
beta



Question: Are the ones with the biggest heterogeneity the one with smallest or highest standard deviation ?

```
## # A tibble: 93,914 x 5
##    code                       cg_id      celltype qjpval      sd
##    <chr>                      <chr>      <chr>     <dbl>   <dbl>
##  1 1:100001201_G_T cg17157234 cg17157234 neu      0.973  0.0226
##  2 1:100001201_G_T cg17157234 cg17157234 mono     0.652  0.0231
##  3 1:100001201_G_T cg17157234 cg17157234 tcell    0.0992 0.0309
##  4 1:100001201_G_T cg17157234 cg17157234 bcell    0.177  0.0259
##  5 1:100015837_A_G cg05108386 cg05108386 neu      0.152  0.0140
##  6 1:100015837_A_G cg05108386 cg05108386 mono     0.394  0.0128
##  7 1:100015837_A_G cg05108386 cg05108386 tcell    0.265  0.0219
##  8 1:100015837_A_G cg05108386 cg05108386 bcell    0.658  0.0144
##  9 1:100015837_A_G cg23599391 cg23599391 neu      0.832  0.00746
## 10 1:100015837_A_G cg23599391 cg23599391 mono     0.147  0.00751
## # i 93,904 more rows
```

```
## # A tibble: 23,973 x 5
## # Groups:   code [23,973]
##    code                       celltype  qjpval cg_id           sd
##    <chr>                      <chr>      <dbl> <chr>        <dbl>
##  1 1:100001201_G_T cg17157234 tcell    0.0992  cg17157234 0.0309
##  2 1:100015837_A_G cg05108386 neu      0.152   cg05108386 0.0140
##  3 1:100015837_A_G cg23599391 tcell    0.00194 cg23599391 0.0261
##  4 1:10004882_C_T cg15685931  neu      0.0538  cg15685931 0.0157
##  5 1:100065568_A_G cg07721612 tcell    0.0238  cg07721612 0.0702
```

```
##  6 1:100079353_A_C cg06661172 mono     0.703   cg06661172 0.00967
##  7 1:100081347_C_G cg00673963 bcell    0.0419  cg00673963 0.00828
##  8 1:100084101_A_G cg17310086 tcell    0.366   cg17310086 0.0165
##  9 1:10009114_C_T cg15226751  bcell    0.0574  cg15226751 0.0110
## 10 1:100125121_C_T cg08874824 bcell    0.668   cg08874824 0.00927
## # i 23,963 more rows
```

3.3 Correlations between models

3.3.1 Model 4 hits with most heterogeneity vs model 1

3.3.2 Model1 and model2

```
## Cell Types: neu vs neu
## Correlation between models: 0.03276922
## Cell Types: neu vs mono
## Correlation between models: -0.003914543
## Cell Types: neu vs tcell
## Correlation between models: -0.008831725
## Cell Types: neu vs bcell
## Correlation between models: -0.006717542
## Cell Types: mono vs neu
## Correlation between models: 0.0159923
## Cell Types: mono vs mono
## Correlation between models: 0.01369849
## Cell Types: mono vs tcell
## Correlation between models: -0.009344959
## Cell Types: mono vs bcell
## Correlation between models: 0.001400644
## Cell Types: tcell vs neu
## Correlation between models: 0.02149393
## Cell Types: tcell vs mono
## Correlation between models: -0.01667452
## Cell Types: tcell vs tcell
## Correlation between models: -0.01198171
## Cell Types: tcell vs bcell
## Correlation between models: 0.01729556
## Cell Types: bcell vs neu
## Correlation between models: 0.004986724
## Cell Types: bcell vs mono
## Correlation between models: -0.01918466
## Cell Types: bcell vs tcell
## Correlation between models: 0.007024559
## Cell Types: bcell vs bcell
## Correlation between models: 0.0003556767
```

3.4 mashr

```
# Step 1: Select strong signals.
## Running a condition-by-condition (1by1) analysis on all the data
data = mash_set_data(reshaped_df$Beta, reshaped_df$SE)
```

```
m.1by1 = mash_1by1(data)
strong = get_significant_results(m.1by1,0.05)
## This sets up a vector strong containing the indices corresponding to
## the significant rows of the tests results in data.

# Step 2: Obtain initial data-driven covariance matrices
## Here we use the function cov_pca to produce covariance matrices based on the
## top 4 PCs of the strong signals. The result is a list of 6 covariance matrices:
## one based on all 4 PCs, and the others each based on one PC
U.pca = cov_pca(data,4,subset=strong)
print(names(U.pca))
```

```
## [1] "PCA_1" "PCA_2" "PCA_3" "PCA_4" "tPCA"
```

```
## Note that the number of conditions in the data is greater than or equal to
## the number of principal components we are trying to compute.

# Step 3: Apply Extreme Deconvolution
## Use these data-driven covariance matrices as initializations for the extreme
## deconvolution (ED) algorithm (using cov_ed), to get some refined data-driven
## covariance matrix estimates
U.ed = cov_ed(data, U.pca, subset=strong)

# Step 4: Calculate canonical covariances
U.c = cov_canonical(data)

# Step 5: Run mash with both data-driven covariances and canonical covariances
m.c    = mash(data, U.c)
```

```
##  - Computing 22735 x 307 likelihood matrix.
##  - Likelihood calculations took 5.83 seconds.
##  - Fitting model with 307 mixture components.
##  - Model fitting took 33.11 seconds.
##  - Computing posterior matrices.
##  - Computation allocated took 0.75 seconds.
```

```
m.ed   = mash(data, U.ed)
```

```
##  - Computing 22735 x 171 likelihood matrix.
##  - Likelihood calculations took 3.36 seconds.
##  - Fitting model with 171 mixture components.
##  - Model fitting took 8.41 seconds.
##  - Computing posterior matrices.
##  - Computation allocated took 0.63 seconds.
```

```
m   = mash(data, c(U.c,U.ed))
```

```
##  - Computing 22735 x 477 likelihood matrix.
##  - Likelihood calculations took 9.06 seconds.
##  - Fitting model with 477 mixture components.
##  - Model fitting took 81.74 seconds.
##  - Computing posterior matrices.
##  - Computation allocated took 0.91 seconds.
```

```
print(get_loglik(m.c),digits = 10)
```

```
## [1] 218411.4733
```

```
print(get_loglik(m.ed),digits = 10)
```

```
## [1] 213583.8907
```

```
print(get_loglik(m),digits = 10)
```

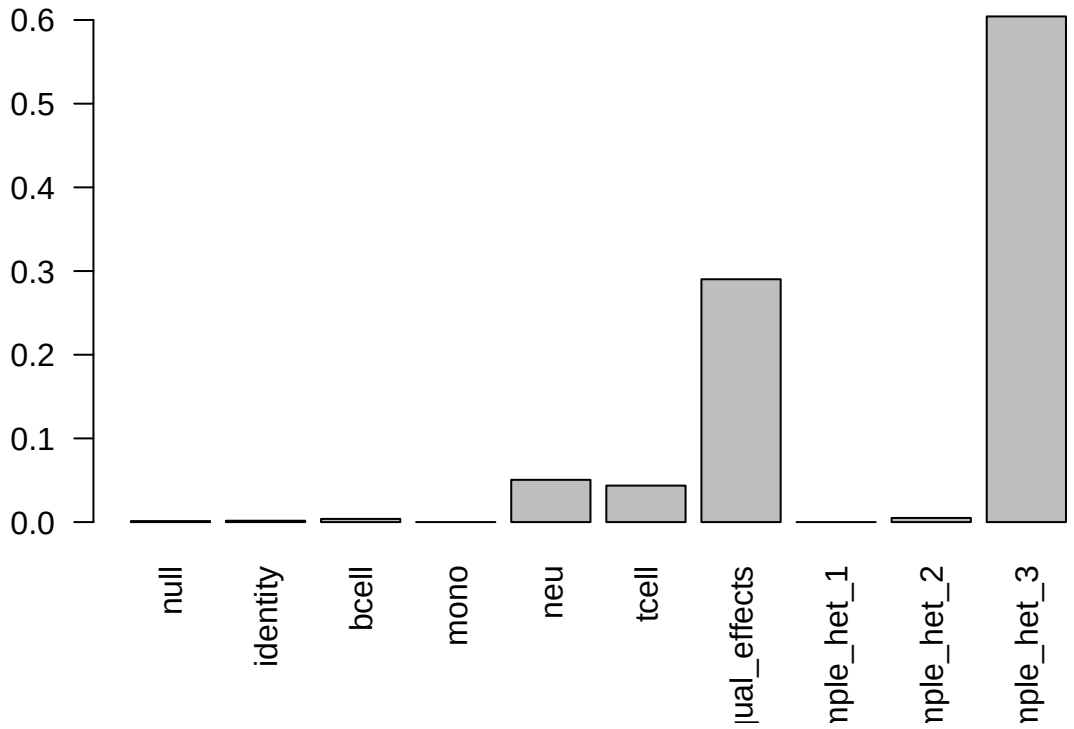
```
## [1] 219923.5069
```

When running mashr it is recommended to fit with both data-driven and canonical covariances. This can be easily checked observing the log likelihood values. The biggest number would indicate the best fit so using both covariances methods. We now can use **get_significant_results()** to find the indices of effects that are "significant", which here means they have **lfsr** less than t in at least one condition, where t is a threshold you specify (default 0.05). The output is ordered from most significant to least significant.

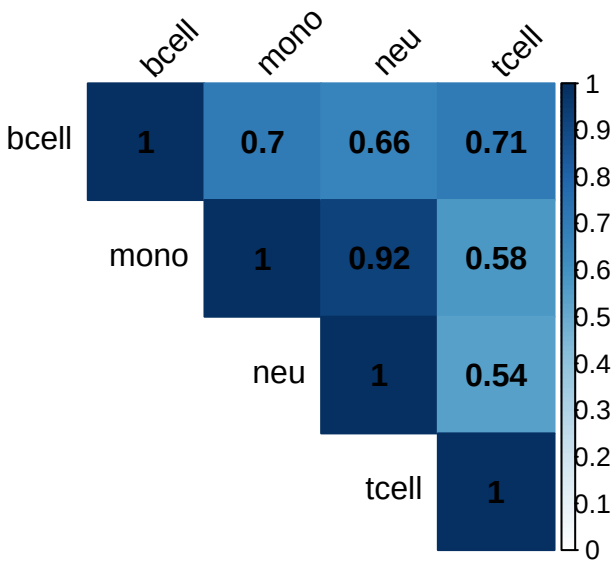
```
## 1:100194305_C_G_cg03360767 1:100302046_A_G_cg13642881
##                         17                         31
## 1:100308671_C_G_cg22017303 1:100339642_A_G_cg13642881
##                         32                         40
## 1:101012485_A_G_cg07412545 1:101031163_A_G_cg09408571
##                         74                         76
```

To extract the estimates of the mixture proportions for different types of covariance matrix we can use **get_estimated_pi()** function and plot the estimates.

```
##          null      identity         bcell          mono           neu
##   0.001173861   0.001628510   0.003777366   0.000000000   0.050473432
##         tcell equal_effects  simple_het_1  simple_het_2  simple_het_3
##   0.043579087   0.290173582   0.000000000   0.004951132   0.604243031
```



There is another function in mashr package, **get_pairwise_sharing()** that compute the proportion of significant signals shared by magnitude in the estimated effect sizes, for each pair of conditions. For each pair of conditions, first identify the effects that are significant in at least one of the two conditions. Then compute what fraction of these have an estimated (posterior mean) effect size within a factor factor of one another.


Pairwise Sharing by Magnitude
bcell
mono
neu
tcell
bcell
mono
neu
tcell
1
0.7
0.66
0.71
1
0.92
0.58
1
0.54
1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0