**<https://cran.r-project.org/web/packages/susieR/vignettes/finemapping.html>**

**Fine-mapping examples**

**Gao Wang**

**2021-06-07**

This vignette demonstrates susieR in the context of genetic fine-mapping. We use simulated data of expression level of a gene (yy) in N≈600N≈600 individuals. We want to identify with the genotype matrix XN×PXN×P (P=1001P=1001) the genetic variables that causes changes in expression level.

The simulated data set is simulated to have exactly 3 non-zero effects.

**library**(susieR)

**set.seed**(1)

**The data-set**

**data**(N3finemapping)

**attach**(N3finemapping)

*# The following objects are masked from N3finemapping (pos = 3):*

*#*

*# allele\_freq, chrom, pos, residual\_variance, true\_coef, V, X, Y*

The loaded dataset contains regression data XX and yy, along with some other relevant properties in the context of genetic studies. It also contains the “true” regression coefficent the data is simulated from.

Notice that we’ve simulated 2 sets of YY as 2 simulation replicates. Here we’ll focus on the first data-set.

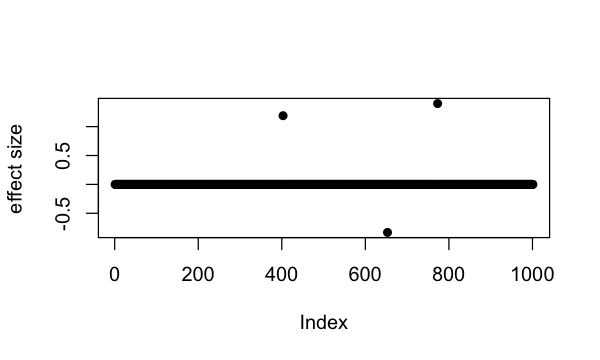
**dim**(Y)

*# [1] 574 2*

Here are the 3 “true” signals in the first data-set:

b <- true\_coef[,1]

**plot**(b, pch=16, ylab='effect size')



**which**(b != 0)

*# [1] 403 653 773*

So the underlying causal variables are 403, 653 and 773.

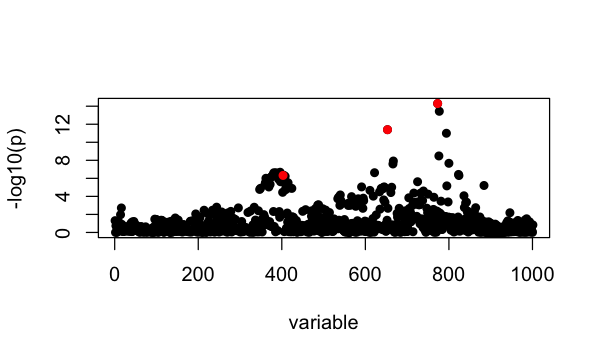
**Simple regression summary statistics**

univariate\_regression function can be used to compute summary statistics by fitting univariate simple regression variable by variable. The results are β^β^ and SE(β^)SE(β^) from which z-scores can be derived. Again we focus only on results from the first data-set:

sumstats <- **univariate\_regression**(X, Y[,1])

z\_scores <- sumstats$betahat / sumstats$sebetahat

**susie\_plot**(z\_scores, y = "z", b=b)



**Fine-mapping with susieR**

For starters, we assume there are at most 10 causal variables, i.e., set L = 10, although SuSiE is robust to the choice of L.

Notice that by default SuSiE estimates prior effect size from data. For fine-mapping applications, however, we often have some knowledge of SuSiE prior effect size since it is parameterized as percentage of variance explained. Here for this eQTL fine-mapping application we set SuSiE prior variance to 0.1, and update residual variance in the variational algorithm that fits SuSiE model. The susieR function call is:

fitted <- **susie**(X, Y[,1],

L = 10,

estimate\_residual\_variance = TRUE,

estimate\_prior\_variance = FALSE,

scaled\_prior\_variance = 0.1,

verbose = TRUE)

*# [1] "objective:-1392.42460921237"*

*# [1] "objective:-1388.98662572569"*

*# [1] "objective:-1387.87893344487"*

*# [1] "objective:-1387.87158194815"*

*# [1] "objective:-1387.73100740437"*

*# [1] "objective:-1387.73099852765"*

*# [1] "objective:-1387.67616609041"*

*# [1] "objective:-1387.67615783404"*

*# [1] "objective:-1386.3771843175"*

*# [1] "objective:-1386.36402706426"*

*# [1] "objective:-1381.99570801789"*

*# [1] "objective:-1381.86807518103"*

*# [1] "objective:-1381.18590761064"*

*# [1] "objective:-1381.18342754332"*

*# [1] "objective:-1381.13344267981"*

*# [1] "objective:-1381.13343270768"*

*# [1] "objective:-1381.13113705137"*

*# [1] "objective:-1381.1311369139"*

*# [1] "objective:-1381.13106293455"*

**Credible sets**

By default, we output 95% credible set:

**print**(fitted$sets)

*# $cs*

*# $cs$L3*

*# [1] 653*

*#*

*# $cs$L1*

*# [1] 773 777*

*#*

*# $cs$L2*

*# [1] 362 372 373 374 379 381 383 384 386 387 388 389 391 392 396 397 398 399 400*

*# [20] 401 403 404 405 407 408 415*

*#*

*#*

*# $purity*

*# min.abs.corr mean.abs.corr median.abs.corr*

*# L3 1.0000000 1.0000000 1.0000000*

*# L1 0.9815726 0.9907863 0.9907863*

*# L2 0.8686309 0.9673674 0.9756251*

*#*

*# $cs\_index*

*# [1] 3 1 2*

*#*

*# $coverage*

*# [1] 0.9998995 0.9992271 0.9506579*

*#*

*# $requested\_coverage*

*# [1] 0.95*

The 3 causal signals have been captured by the 3 CS reported here. The 3rd CS contains many variables, including the true causal variable 403. The minimum absolute correlation is 0.86.

If we use the default 90% coverage for credible sets, we still capture the 3 signals, but “purity” of the 3rd CS is now 0.91 and size of the CS is also a bit smaller.

sets <- **susie\_get\_cs**(fitted,

X = X,

coverage = 0.9,

min\_abs\_corr = 0.1)

**print**(sets)

*# $cs*

*# $cs$L3*

*# [1] 653*

*#*

*# $cs$L1*

*# [1] 773 777*

*#*

*# $cs$L2*

*# [1] 373 374 379 381 383 384 386 387 388 389 391 392 396 399 400 401 403 404 405*

*# [20] 407 408*

*#*

*#*

*# $purity*

*# min.abs.corr mean.abs.corr median.abs.corr*

*# L3 1.0000000 1.0000000 1.0000000*

*# L1 0.9815726 0.9907863 0.9907863*

*# L2 0.9119572 0.9744588 0.9779763*

*#*

*# $cs\_index*

*# [1] 3 1 2*

*#*

*# $coverage*

*# [1] 0.9998995 0.9992271 0.9046596*

*#*

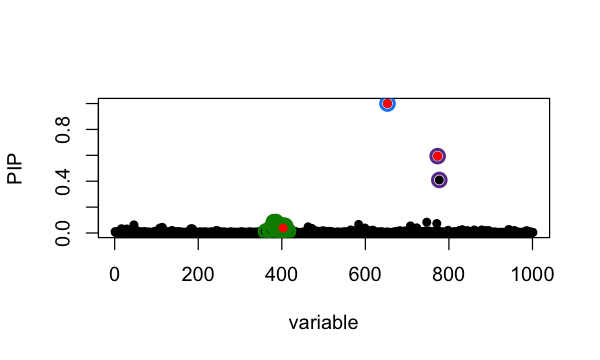
*# $requested\_coverage*

*# [1] 0.9*

**Posterior inclusion probabilities**

Previously we’ve determined that summing over 3 single effect regression models is approperate for our application. Here we summarize the variable selection results by posterior inclusion probability (PIP):

**susie\_plot**(fitted, y="PIP", b=b)



The true causal variables are colored red. The 95% CS identified are circled in different colors. Of interest is the cluster around position 400. The true signal is 403 but apparently it does not have the highest PIP. To compare ranking of PIP and original z-score in that CS:

i <- fitted$sets$cs[[3]]

z3 <- **cbind**(i,z\_scores[i],fitted$pip[i])

**colnames**(z3) <- **c**('position', 'z-score', 'PIP')

z3[**order**(z3[,2], decreasing = TRUE),]

*# position z-score PIP*

*# [1,] 396 5.189811 0.06026797*

*# [2,] 381 5.164794 0.08444210*

*# [3,] 386 5.164794 0.08444210*

*# [4,] 379 5.077563 0.05804100*

*# [5,] 391 5.068388 0.06299894*

*# [6,] 383 5.057053 0.06048718*

*# [7,] 384 5.057053 0.06048718*

*# [8,] 389 5.052519 0.04925977*

*# [9,] 405 5.039617 0.05629383*

*# [10,] 403 5.035949 0.03808209*

*# [11,] 387 5.013526 0.04831886*

*# [12,] 388 4.997955 0.04696986*

*# [13,] 408 4.994865 0.05477586*

*# [14,] 404 4.954407 0.04024608*

*# [15,] 374 4.948060 0.03589437*

*# [16,] 373 4.934410 0.02832642*

*# [17,] 362 4.894243 0.01244577*

*# [18,] 399 4.860780 0.03073966*

*# [19,] 392 4.856384 0.02490617*

*# [20,] 407 4.849285 0.02139881*

*# [21,] 400 4.827361 0.02610013*

*# [22,] 398 4.751205 0.01761845*

*# [23,] 401 4.723184 0.02014360*

*# [24,] 397 4.716886 0.01240547*

*# [25,] 415 4.663208 0.01437857*

*# [26,] 372 4.581560 0.01032276*

**Choice of priors**

We found that SuSiE is generally robust to choice of priors. Here we set scaled prior variance (percentage of variance explained) to 0.2, and compare with previous results:

fitted = **susie**(X, Y[,1],

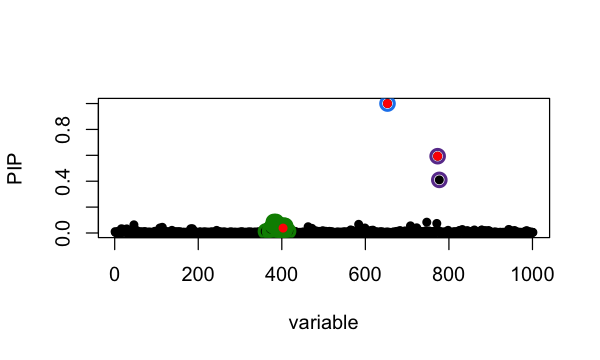
L = 10,

estimate\_residual\_variance = TRUE,

estimate\_prior\_variance = FALSE,

scaled\_prior\_variance = 0.2)

**susie\_plot**(fitted, y='PIP', b=b)



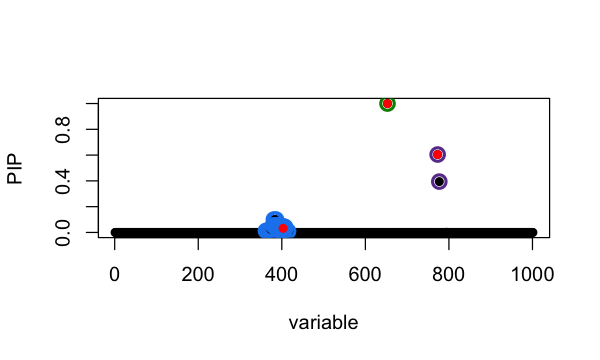
which largely remains unchanged. We also illustrate below analysis using default prior estimated from data,

fitted = **susie**(X, Y[,1],

L = 10,

estimate\_residual\_variance = TRUE)

**susie\_plot**(fitted, y='PIP', b=b)



**A note on covariate adjustment**

To include covariate Z in SuSiE, one approach is to regress it out from both y and X, and then run SuSiE on the residuals. The code below illustrates the procedure:

remove.covariate.effects <- **function** (X, Z, y) {

*# include the intercept term*

**if** (**any**(Z[,1]!=1)) Z = **cbind**(1, Z)

A <- **forceSymmetric**(**crossprod**(Z))

SZy <- **as.vector**(**solve**(A,**c**(y %\*% Z)))

SZX <- **as.matrix**(**solve**(A,**t**(Z) %\*% X))

y <- y - **c**(Z %\*% SZy)

X <- X - Z %\*% SZX

**return**(**list**(X = X,y = y,SZy = SZy,SZX = SZX))

}

out = **remove.covariate.effects**(X, Z, Y[,1])

fitted\_adjusted = **susie**(out$X, out$y,

L = 10,

estimate\_residual\_variance = TRUE)

Note that the covariates Z should have a column of ones as the first column. If not, the above function remove.covariate.effects will add such a column to Z before regressing it out. Data will be centered as a result. Also the scale of data is changed after regressing out Z. This introduces some subtleties in terms of interpreting the results. For this reason, we provide covariate adjustment procedure as a tip in the documentation and not part of susieR::susie() function. Cautions should be taken when applying this procedure and interpreting the result from it.

**Session information**

Here are some details about the computing environment, including the versions of R, and the R packages, used to generate these results.

**sessionInfo**()

*# R version 3.6.2 (2019-12-12)*

*# Platform: x86\_64-apple-darwin15.6.0 (64-bit)*

*# Running under: macOS Catalina 10.15.7*

*#*

*# Matrix products: default*

*# BLAS: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib*

*# LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib*

*#*

*# locale:*

*# [1] en\_US.UTF-8/en\_US.UTF-8/en\_US.UTF-8/C/en\_US.UTF-8/en\_US.UTF-8*

*#*

*# attached base packages:*

*# [1] stats graphics grDevices utils datasets methods base*

*#*

*# other attached packages:*

*# [1] susieR\_0.11.42*

*#*

*# loaded via a namespace (and not attached):*

*# [1] Rcpp\_1.0.5 knitr\_1.26 magrittr\_1.5 tidyselect\_0.2.5*

*# [5] munsell\_0.5.0 colorspace\_1.4-1 lattice\_0.20-38 R6\_2.4.1*

*# [9] rlang\_0.4.5 highr\_0.8 plyr\_1.8.5 dplyr\_0.8.3*

*# [13] stringr\_1.4.0 tools\_3.6.2 grid\_3.6.2 gtable\_0.3.0*

*# [17] xfun\_0.11 irlba\_2.3.3 htmltools\_0.4.0 assertthat\_0.2.1*

*# [21] yaml\_2.2.0 digest\_0.6.23 tibble\_2.1.3 lifecycle\_0.1.0*

*# [25] crayon\_1.3.4 mixsqp\_0.3-46 Matrix\_1.2-18 purrr\_0.3.3*

*# [29] ggplot2\_3.3.0 glue\_1.3.1 evaluate\_0.14 rmarkdown\_2.3*

*# [33] stringi\_1.4.3 compiler\_3.6.2 pillar\_1.4.3 scales\_1.1.0*

*# [37] reshape\_0.8.8 pkgconfig\_2.0.3*