`mash` No Baseline

Sarah Urbut

November 4, 2016

Contents

1 Purpose

The purpose of this document is to propose a method for extending `mash` to estimate 'true' effects across conditions in a setting in which no obvious baseline exists. We assume that we observe noisy, uncentered averages $\hat{\boldsymbol{C}}_{jr}$ in each of R conditions, and seek to estimate the underlying true 'deviations' from average measurement across conditions and can be seen as the effects in `mash`.

Here, the use of bold-face notation indicates a vector, while matrix quantities are typeset in capital but unboldface letters.

2 Defining the Model

Now, we observe for each gene j a vector of uncentered noisy average feature expression $\hat{\boldsymbol{C}}$ across R conditions:

$$\hat{\boldsymbol{C}}|\boldsymbol{C} \sim N_R(\boldsymbol{C}, \hat{V}) \tag{1}$$

where the 'true' uncentered averages $\boldsymbol{C}$ can be written as follows:

$$\boldsymbol{C}|\mu, \boldsymbol{v} = \mu\mathbf{1} + \boldsymbol{v} \tag{2}$$

Where μ is a scalar that is the mean of the 'true' uncentered averages $\boldsymbol{C}$.

$\boldsymbol{v}$ is a zero-centered mixture of multivariate normals:

$$\boldsymbol{v}|\boldsymbol{\pi}, \mathbf{U} \sim \sum_{\mathbf{k,l}} \pi_{\mathbf{k,l}}\, N_{\mathbf{R}}(\mathbf{0}, \omega_{\mathbf{l}}\mathbf{U_k}) \tag{3}$$

Critically, our quantity of interest now, $\boldsymbol{v}$ represents the true 'deviations' from average gene expression across each condition and can be seen as the effects in `mash`.

3 Applications

We will again apply a two-step process to our selection of covariance matrices, where we select a set of denoised 'pattern' matrices U_k by using the EM algorithm on the max effects across conditions, and then expanding this list by a fixed grid of scalar weights ω_l such that we conclude with a list of $P = KxL$ covariance matrices Σ. We can then:

- estimate the P prior weights $\boldsymbol{\pi}$ on this fixed P-list of covariance matrices from a training matrix of randomly selected feature expression measurements across conditions
- compute the posterior distribution $\boldsymbol{v}|L\hat{\boldsymbol{C}}, \boldsymbol{s_j}$

  Let

$$L\boldsymbol{C} = L\mu\mathbf{1} + L\boldsymbol{v} \tag{4}$$

  L is the RxR centering matrix $L_r = I_r - \frac{1}{r}\mathbf{1}\mathbf{1}^\top$ which removes the mean of each R column vector.

  Then :

$$\begin{aligned} L\boldsymbol{C} &= L\mu\mathbf{1} + L\boldsymbol{v} \\ L\boldsymbol{C} &= 0 + L\boldsymbol{v} \\ L\hat{\boldsymbol{C}} &= L\boldsymbol{v} + E \end{aligned} \tag{5}$$

  Where $E \sim \mathcal{N}(0, L\hat{V}L')$. Note that this is identical to Bovy equation (1).

4 Likelihood with `mashnobaseline`

With `mashnobaseline`, we will replace the RxR matrix L with the R-1xR matrix L, effectively removing a data point from the observed uncentered statistics, such that the rank of the marginal variance of $\boldsymbol{w} is guaranteed to be equal to the dimension of \boldsymbol{w}$.

Now for each gene J at each component k, integrating over $\boldsymbol{v}$,

$$\begin{aligned} \mathrm{L}\boldsymbol{C} &\sim \mathcal{N}(0,\ \mathrm{L}U_k\ \mathrm{L}') \\ \mathrm{L}\hat{\boldsymbol{C}} &\sim \mathcal{N}(0,\ \mathrm{L}U_k\ \mathrm{L}' + \mathrm{L}\hat{V}\ \mathrm{L}') \end{aligned} \tag{6}$$

Note that because L removes one rank from every vector, adding a rank 1 matrix ($\mathbf{1}\mathbf{1}'$) to U_k will not change the likelihood.

And thus we can use the Bovy et al algorithm invoked in both the Extreme Deconvolution package and in 'Sarah's MixEm' where according to Bovy's language we observe a noisy estimate of uncentered averages $\hat{\boldsymbol{C}}$ and can center them to and $w_j = \mathrm{L}\hat{\boldsymbol{C}}_j$.

Then we can apply Bovy:

$$
\begin{aligned}
\mathbf{w} &= \mathrm{L}\hat{\boldsymbol{C}} \\
\mathbf{w}|\boldsymbol{v}, \hat{V} &\sim \mathcal{N}(\ \mathrm{L}\boldsymbol{v},\ \mathrm{L}\hat{V}\ \mathrm{L}') \\
\boldsymbol{v}|\boldsymbol{\pi}, \mathbf{U} &\sim \sum_{\mathbf{k},\mathbf{l}} \pi_{\mathbf{k},\mathbf{l}}\ N_{\mathbf{R}}(\mathbf{0}, \omega_{\mathbf{l}}\mathbf{U}_{\mathbf{k}})
\end{aligned}
\tag{7}
$$

As before, we are interested in returning the prior covariance U_k matrices of the 'true' deviations $\boldsymbol{v}$ that maximizes (8):

$$
\begin{aligned}
L(\theta) &:= p(\mathbf{w}|\pi, V, U_k) \\
&= \prod_{j=1}^{J} p(\mathbf{w}_j|\pi, V, U_k) \\
&= \prod_{j=1}^{J} \sum_{k}^{K} \pi_k N_R(\mathbf{w}_j; \mathbf{0},\ \mathrm{L}U_k\ \mathrm{L}' +\ \mathrm{L}V_j\ \mathrm{L}').
\end{aligned}
\tag{8}
$$

Identical to the framework in Bovy.

We will then rescale each of the U_k by choosing a set of ω that are appropriate to $L\hat{\boldsymbol{C}}$ to comprise a set of $P = KxL$ prior covariance matrices Σ that maximizes :

$$
\begin{aligned}
L(\pi) &:= p(\mathbf{w}|, V, \Sigma \\
&= \prod_{j=1}^{J} p(\mathbf{w}_j|\pi, V, \Sigma) \\
&= \prod_{j=1}^{J} \sum_{p}^{P} \pi_p N_R(\mathbf{w}_j; \mathbf{0},\ \mathrm{L}\Sigma_p\ \mathrm{L}' +\ \mathrm{L}V_j\ \mathrm{L}').
\end{aligned}
$$

(9)

We assemble a matrix of likelihoods that will compute the following likelihood at each of the P components:

$$
\begin{aligned}
\mathrm{L}\hat{\boldsymbol{C}}_j &\sim \mathcal{N}(0,\ \mathrm{L}\Sigma_p\ \mathrm{L}' + \mathrm{L}\hat{V}_j\ \mathrm{L}') \\
T_{jp} &= \mathrm{L}\Sigma_p\ \mathrm{L}' + \mathrm{L}\hat{V}_j\ \mathrm{L}' \\
\mathrm{L}\hat{\boldsymbol{C}}_j &\sim \mathcal{N}(0, T_{jp})
\end{aligned}
\tag{10}
$$

5 Posteriors

Now, as before we can compute a posterior distribution such that:

$$\boldsymbol{v}|\ \mathrm{L}\hat{\boldsymbol{C}}, \pi, \Sigma, \boldsymbol{s} \sim N(\mu^1, U^1) \tag{11}$$

Where at each of the P components for each gene J

$$\begin{aligned} \mu^1_{jp} &= \Sigma_p\ \mathrm{L}' T^{-1}_{jp}\ \mathrm{L}\hat{\boldsymbol{C}}_j \\ U^1_{jp} &= \Sigma_p - \Sigma_p\ \mathrm{L}' T^{-1}_{jp}\ \mathrm{L}\Sigma_p \end{aligned} \tag{12}$$

Analogous to Bovy et al equation 13 and 14, because our $\boldsymbol{v}_{jk}$ is centered at 0 (and so their $\mathbf{m_j}$ is effectively 0 for all components.

Our question is, if you were to add the matrix of 1s (i.e., $\mathbf{1}\mathbf{1}'$) to each Σ_p, μ^1_{jp} doesn't change.

$$\begin{aligned} \mu^1_{jp}(\Sigma_p) &= \Sigma_p\ \mathrm{L}' T^{-1}_{jp}\ \mathrm{L}\hat{\boldsymbol{C}}_j \\ \mu^1_{jp}(\Sigma_p + \mathbf{1}\mathbf{1}') &= \Sigma_p\ \mathrm{L}' T^{-1}_{jp}\ \mathrm{L}\hat{\boldsymbol{C}}_j + \mathbf{1}\mathbf{1}'\ \mathrm{L}' T^{-1}_{jp}\ \mathrm{L}\hat{\boldsymbol{C}}_j \\ \mu^1_{jp}(\Sigma_p + \mathbf{1}\mathbf{1}') &= \Sigma_p\ \mathrm{L}' T^{-1}_{jp}\ \mathrm{L}\hat{\boldsymbol{C}}_j + \mathbf{0}' T^{-1}_{jp}\ \mathrm{L}\hat{\boldsymbol{C}}_j \\ \mu^1_{jp}(\Sigma_p + \mathbf{1}\mathbf{1}') &= \Sigma_p\ \mathrm{L}' T^{-1}_{jp}\ \mathrm{L}\hat{\boldsymbol{C}}_j \end{aligned} \tag{13}$$

And thus we see that $\mu^1_{jp}(\Sigma_p) = \mu^1_{jp}(\Sigma_p + \mathbf{1}\mathbf{1}')$ if the projection matrix is equal to the centering matrix L. But this result seems strange!

This is correct. We can see that in Bovy (11) and (12), to derive the posterior condition distribution of $\boldsymbol{v}|\boldsymbol{w}$, he uses the following formulat:

If x is partitioned as follows:

$\mathrm{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}$

and accordingly $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are partitioned as follows

$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}$

$\begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}$

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}$$

then the distribution of $\mathbf{x}_1$ conditional on $\mathbf{x}_2 = \mathbf{a_1}$ is multivariate normal

$$\mathbf{x}_1 | \mathbf{x}_2 = a \sim N(\bar{\boldsymbol{\mu}}, \overline{\boldsymbol{\Sigma}}) \tag{14}$$

$$\bar{\boldsymbol{\mu}} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\left(\mathbf{a} - \boldsymbol{\mu}_2\right) \tag{15}$$

and covariance matrix

$$\overline{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}. \tag{16}$$

Plugging in our matrices at any given component, we have that:

$$\begin{gathered} \mathbf{x}_1 = \boldsymbol{v} \\ \\ \mathbf{x}_2 = \mathbf{a} = \boldsymbol{w} \\ \boldsymbol{\mu}_1 = 0 \\ \boldsymbol{\mu}_2 = 0 \\ \boldsymbol{\Sigma}_{11} = \Sigma \\ \boldsymbol{\Sigma}_{22} = T \\ \boldsymbol{\Sigma}_{12} = \mathbb{C}(\boldsymbol{v}\boldsymbol{w}) = \Sigma\ \mathrm{L}' \end{gathered} \tag{17}$$

The equation below:

$$\mathbb{C}(\boldsymbol{v}\boldsymbol{w}) = \Sigma\ \mathrm{L}' \tag{18}$$

Is the case because $\boldsymbol{v} \perp\!\!\!\perp \mathbf{E}$:

$$\begin{aligned}\mathbb{C}(\boldsymbol{v}\boldsymbol{w}) &= \mathbb{C}(\boldsymbol{v},\ \mathrm{L}\boldsymbol{v} + \mathbf{E}) \\ &= \mathbb{C}(\boldsymbol{v},\ \mathrm{L}\boldsymbol{v}) + \mathbb{C}(\boldsymbol{v}, \mathbf{E}) \\ &= \mathbb{C}(\boldsymbol{v},\ \mathrm{L}\boldsymbol{v}) + 0 \\ &= \Sigma\ \mathrm{L}'\end{aligned} \tag{19}$$

And so this turns out our formula for μ^1 and U^1:

$$\begin{aligned}\mu^1(\Sigma) &= \Sigma\ \mathrm{L}' T^{-1} \boldsymbol{w} \\ \mathrm{U}^1(\Sigma_p) &= \Sigma - \Sigma\ \mathrm{L}' T^{-1}\ \mathrm{L}\Sigma_p\end{aligned} \tag{20}$$

5.1 Selecting The Covariance Matrices

We initiate our set of covariance matrices for the denoising step as before in `mash`, where now we compute the empirical covariance matrices and a variety of dimensional reductions on the feature-centered JxR *matrix* of maximum average, $L\hat{C}'$ instead of $\hat{C}$ alone. In practice, we actually use the matrix of maximum uncentered T statistics. Three critical things to note:

1. Here, L will be RxR because we need U_k to be RxR
2. When denoising with Bovy, our previous approach used the matrix of maximum T statistics T_{Mxr} to both initialize and train the BovyEM. Now, we will initalize with the MXR matrix of t($L_{R,R}T'$) (or alternatively, TL) and train the EM on the MxR-1 matrix of maximum $t(\ \mathrm{L}T')$
3. When choosing ω, we will use the diagonal of LVL', where V is $D(s.j^2)$, and select from the MxR matrix of centered T statistics and their centered standard errors.
4. We will choose the maxes as those that have a maximum centered t statistic of at least some threshold in at least one (or averaged across tissues) rather than those that satisfy an ash criteria in at least one tissue because we know that choosing uncorrelated LVL' as the standard errors with which to input to ash is incorrect.

6 Previous Model

If we were to apply the old `mash` to data simulated as in (5) in which we assumed the mean was known, we would write

$$\begin{aligned} \boldsymbol{C}|\mu, \boldsymbol{v} &= \mu\mathbf{1} + \boldsymbol{v} \\ \hat{\boldsymbol{C}} &= \mu + \boldsymbol{v} + E \\ \hat{\boldsymbol{C}} - \mu &= \boldsymbol{v} + E \end{aligned} \tag{21}$$

where $E \sim \mathcal{N}(0, V)$

So using our previous notation,

$$L\hat{\boldsymbol{C}} \sim \mathcal{N}(\boldsymbol{v}, V) \tag{22}$$

Critically, note that here L is RxR, not $R-1$ x R because this model does not require that the likelihood of w be non-degenerate.

Here, the likelihood of $\boldsymbol{w}$ =L$\hat{\boldsymbol{C}}$ integrating over $\boldsymbol{v}$ is:

$$\boldsymbol{w}_R \sim N_R(0, V + U) \tag{23}$$

7 Comparing Models

In the new model, we incorporate the centering matrix L into our selection of covariance matrices by modeling $L\hat{\boldsymbol{C}} = L\boldsymbol{v} + E$ where $E \sim N(0, LVL')$.

Using old mash, we assume that the mean centered estimates directly approximate $\boldsymbol{v}$, that is $\hat{\boldsymbol{C}} - \mu = \boldsymbol{v} + E$ where $E \sim N(0, V)$. We want to estimate effects with no baseline, such that null effects aren't forced to be negative (by a competitive default, given that non-null effects are seen as positive, or vice versa). We hope that this method will simply shrink null effects to zero and recognize the direction of the non-null effects, instead of making both null and non-null positive (negative) relative to some intermediate average.

8 Intuition

Consider the situation in which we observe uncentered expression $\hat{\boldsymbol{C}} = \begin{bmatrix} 1 & 4 & 1 \end{bmatrix}$

We want to produce estimates which recognize that $\boldsymbol{v}$ is $\begin{bmatrix}0 & 3 & 0\end{bmatrix}$, not $\begin{bmatrix}-1 & 2 & -1\end{bmatrix}$

We hope to improve upon the measurement $L\hat{\boldsymbol{C}}\begin{bmatrix}-1 & 2 & -1\end{bmatrix}$ as an approximation of $\boldsymbol{v}$ so that rather than

$$\begin{aligned} L\hat{\boldsymbol{C}} &\sim N(v, E) \\ \mathrm{L}\hat{\boldsymbol{C}} &\sim N(L\boldsymbol{v}, E) \end{aligned} \tag{24}$$

Note that the error in the first line will by $N \sim (0, V)$, while in the second line, it will be $N \sim (0,\ \mathrm{L}V\ \mathrm{L}')$, and as we previously pointed out, L is R-1 while L is R.

9 Summary of Discoveries

- When using `mash` and `mashnobaseline` to generate inference matrices and compute likelihoods and posteriors, respectively, `mashnobaseline` seems no better (see ../Simulations/fixedomegawithinference/Untitled.html)
- When we use the true prior for both frameworks, `mashnobaseline` is much better (see ../Simulations/Withbothtrueandestimatedprior.html) because we can see that using the same set of covariance matrices, `mashnobaseline` is better able to emphasize the true configurations because the likelihood and posterior computations incorporate L.
- Recall, for mash:

  $$\begin{aligned} w_{j,R}\ N(0, V + U) \\ w_{j,R-1}\ N(0, LVL' + LUL') \end{aligned}$$

  where L is the R-1 x R centering matrix.

  In both:

  $$\begin{aligned} c &= \mu + \beta \\ chat &= c + E \\ v &\sim \sum_p N(0, Uk) \\ E &\sim N(0, V) \end{aligned}$$

  So even if the Uks are different, the likelihood and corresponding πs will be different.

10 Differences required over `mash` implementation

* We will now work with a matrix of observed column-centered gene averages, $L\hat{C}'$ in order to:
    1. initialize our choice of U_k;
    2. choose the maxes by which to denoise,
    3. choose our set of scales, ω_l
    4. compute our hierarchical weights, $\boldsymbol{\pi}_p$ as well as our posteriors.
* It is critical to note that here L **will need to be RxR because** U_k **must be RxR**
* The new distribution we seek to estimate for each j is then $v|$ L$\hat{\boldsymbol{C}}, \boldsymbol{s}_j$
* To choose the maxes, I think we ought to use a w_j cutoff since computing the univariate lfsr on w_j and the diagonal of LVL' assumes that LVL' is diagonal when we know it cannot be.

11 `mash` implementation

* We will now work with a matrix of observed column-centered gene averages, $L\hat{C}'$ in order to:
    1. initialize our choice of U_k;
    2. choose the maxes by which to denoise,
    3. choose our set of scales, ω_l
    4. compute our hierarchical weights, $\boldsymbol{\pi}_p$ as well as our posteriors.
* It is critical to note that here **L will need to be RxR because** U_k **must be RxR**
* Now, we ignore L and continue with L because we see that 22 only requires the matrix of centered average gene expression
* The new distribution we seek to estimate for each j is then $v|L\hat{\boldsymbol{C}}, \boldsymbol{s}_j$
* We proceed as before and as in `mashnobaseline`, feeding in the matrix of centered summary statistics and this time, the matrix of standard errors (rather than the list of L V L' arrays)

12 Simulation

In this simulation framework, there are 1000 real associations in 10000 null across 44 tissues.

Each 'real association' is simulated in the following manner:

```
{function(n=1000,d=44,betasd=1,esd=0.1,K=10){
  library("MASS")
  library("mvtnorm")
  J=0.10*n

  configs = matrix((rnorm(d*K)),byrow=T,ncol=d) # A matrix of K classes (patterns)
  F=as.matrix(configs);
  covmat=lapply(seq(1:K),function(k){
    A=F[k,]%*%$t(F[k,]);
    A/max(diag(A))})
  ## each entry of F is the the factor of decomposition of covariance of effect si
  z = sample(K,J,replace=TRUE) # randomly sample factor to be loaded on for each r


  mus=rnorm(n)  ###generate a list of n mus
  mumat=matrix(rep(mus,d),ncol=d)##generate a matrix of mus for each gene
  omega=abs(rnorm(J,mean=0,sd=betasd))##effect size variance can be big or small
  beta=t(sapply(seq(1:J),function(j){
    k=z[j]
    mvrnorm(1,mu=rep(0,d),Sigma=omega[j]*covmat[[k]])
    #rmvnorm(1,mean = rep(0,d),sigma=omega*covmat[[k]])
  }))

  beta=rbind(beta,matrix(rep(0,(n-J)*d),ncol=d))
  c=beta+mumat
  sj=abs(matrix(rnorm(n*d,esd,0.001),ncol=d))##use uniform to simulate 'shrunken'
  e=t(apply(sj,1,function(x){rmvnorm(1,mean=rep(0,d),sigma=diag(x)^2)}))
  chat=c+e
  t=chat/sj
  return(list(beta=beta,chat=chat,covmat=covmat,components=z,t=t,mumat=mumat,
  shat=sj,error=e,ceff=c,F=F,omega=omega))}
```

Such that for every true associations a factor is chosen and 'standardized' such that the maximum value across the diagonal is one. The true effects are then simulated according to the assigned component, scaled by some factor ω, and then and this scaling is added to a chosen mean for the gene, centered at o with σ^2 of 1.

The true $ceff$ is then computed as

$$ceff = \mu + \beta$$

and

$$chat = ceff + E$$

where $E \sim N(0, V)$ and V is diagonal.

This function reports the true μ, the true β for the 1000 real genes and their associated componenent, as well as the standard error.

13 Simpler Simulation

In this simulation framework, there are 1000 real associations in 10000 null across 8 and 20 tissues. Here, the goal was to simulate a case in which the effect was 'off' in a subset of tissues and on in some:

Each 'real association' is simulated in the following manner:

```
{function(n=1000,d=8,betasd=1,esd=0.1,K=10){
  library("MASS")
  library("mvtnorm")

  J=0.10*n
  temp=rep(list(c(0,1)),d)
  configs = expand.grid(temp) # all possible 2^d combinations
  S=sample(seq(1:nrow(configs)),size = K,replace = FALSE)##which factors will be used
  F=as.matrix(configs[S,])
  covmat=lapply(seq(1:K),function(k){
    A=F[k,]%*%t(F[k,]);
    A/max(diag(A))})
  ## each entry of F is the the factor of decomposition of covariance of effect sizes
  z = sample(K,J,replace=TRUE) # randomly sample factor to be loaded on for each real
mus=rnorm(n)  ###generate a list of n mus
  mumat=matrix(rep(mus,d),ncol=d)##generate a matrix of mus for each gene
  omega=abs(rnorm(J,mean=0,sd=betasd))##effect size variance can be big or small
```

```
beta=t(sapply(seq(1:J),function(j){
  k=z[j]
  mvrnorm(1,mu=rep(0,d),Sigma=omega[j]*covmat[[k]])
  #rmvnorm(1,mean = rep(0,d),sigma=omega*covmat[[k]])
}))

beta=rbind(beta,matrix(rep(0,(n-J)*d),ncol=d))
c=beta+mumat
sj=abs(matrix(rnorm(n*d,esd,0.001),ncol=d))##use uniform to simulate 'shrunken'
e=t(apply(sj,1,function(x){rmvnorm(1,mean=rep(0,d),sigma=diag(x)^2)}))
chat=c+e
t=chat/sj
return(list(beta=beta,chat=chat,covmat=covmat,components=z,factors=F,t=t,mumat=mumat
```

Such that for every true associations a factor is chosen and 'standardized' such that the maximum value across the diagonal is one. The true effects are then simulated according to the assigned component, scaled by some factor ω, and then and this scaling is added to a chosen mean for the gene, centered at o with σ^2 of 1.

The true $ceff$ is then computed as

$$ceff = \mu + \beta$$

and

$$chat = ceff + E$$

where $E \sim N(0, V)$ and V is diagonal.

This function reports the true μ, the true β for the 1000 real genes and their associated componenent, as well as the standard error.

As mentioned, in `mashnobaseline` we will use the input matrix of $w = t(\ \mathrm{L}t(\hat{\boldsymbol{C}}))$ and list of $\mathrm{L}V\ \mathrm{L}'$ covariance of the errors, while in `mash`, we simply input w $= t(Lt(\hat{\boldsymbol{C}}))$ and the JxR matrix of standard errors.