Week 3 Drosophila Work

Eric Weine

3/17/2022

Overview

This week, the goal was to examine additional ways to demonstrate that Pallares et al. are missing interesting biological features of their dataset by not having an explicit interaction term in their model. We look at three ways to do this: (1) running mash on the estimated parameters from the model in pallares et al. (2) looking at the correlation between the interaction and linear terms in the interaction model and (3) setting up a Bayesian model of a discrete amplification effect.

Applying mash to the drosophila dataset

To apply mash to the drosophila dataset, I followed the same procedure as Carrie in her GxSex project. Specifically, I first built a set of covariance matrices at varying levels of amplification and correlation between the HS group and the C group, and then ran mash with this set of covariance matrices.

First, we build a function to easily generate covariance matrices given a set of specifications.

```
#' Create 2x2 covariance matrices with optional levels of amplification
#'
#' The function assumes that we have two groups, with the standard deviation of
#' the effect size in one group being 1, and the standard deviation of the effect
#' size in the other group being \code{1 * amp_coef} if \code{amp} is set to
#' \code{TRUE}
#'
#' @param desired_corr Desired level of correlation. Must be in [-1, 1].
#' @param amp_coef Coefficient of amplification, as described above.
#' @param amp Boolean indicating if any amplificaiton should take place
#' @param amp_hs Boolean indicating if amplification should take place in the hs
#' group or in the c group. Only used if \code{amp} is set to \code{TRUE}.
#'
#' @return 2x2 covariance matrix
#' @export
#'
#' @examples
make_amp_cov_mat <- function(
  desired_corr, amp_coef = 1, amp = TRUE, amp_hs = TRUE
) {

  if (amp_hs && amp) {

    ctrl_sd <- 1
    hs_sd <- ctrl_sd * amp_coef
```

```
  } else if(!amp_hs && amp) {

    hs_sd <- 1
    ctrl_sd <- hs_sd * amp_coef

  } else {

    hs_sd <- 1
    ctrl_sd <- 1

  }

  # derive covariance from correlation and sds
  cov_hs_ctrl <- desired_corr * hs_sd * ctrl_sd

  cov_mat <- matrix(
    data = c(ctrl_sd ^ 2, cov_hs_ctrl, cov_hs_ctrl, hs_sd ^ 2),
    nrow = 2,
    byrow = TRUE,
    dimnames = list(
      rows = c("ctrl", "hs"), cols = c("ctrl", "hs")
    )
  )

  return(cov_mat)

}
```

Then, we run mash on the dataset. We randomly select 50% of the dataset to run mash on so that my local machine doesn't run out of RAM.

```
# read in 50% of data and select relevant columns
summary_table <- read.delim('data/SummaryTable_allsites_12Nov20.txt')
summary_table_samp <- summary_table %>%
  dplyr::sample_frac(.5) %>%
  dplyr::select(c(site, pval_CTRL, pval_HS, coef_CTRL, coef_HS, sig_cat))

# replace 0 p-values with small numbers
summary_table_samp <- summary_table_samp %>%
  dplyr::mutate(
    pval_CTRL = pmax(.0001, pval_CTRL),
    pval_HS = pmax(.0001, pval_HS)
  )

# construct std error estimates from coefficients and p-values
summary_table_samp <- summary_table_samp %>%
  dplyr::mutate(
    std_error_ctrl = abs(coef_CTRL) / qnorm((2 - pval_CTRL) / 2),
    std_error_hs = abs(coef_HS) / qnorm((2 - pval_HS) / 2)
  )

reg_fx_mat <- t(matrix(
  data = c(summary_table_samp$coef_CTRL, summary_table_samp$coef_HS),
  nrow = 2
```

```
))
colnames(reg_fx_mat) <- c("ctrl", "hs")

reg_se_mat <- t(matrix(
  data = c(summary_table_samp$std_error_ctrl, summary_table_samp$std_error_hs),
  nrow = 2
))
colnames(reg_se_mat) <- c("ctrl", "hs")

mash_data <- mashr::mash_set_data(reg_fx_mat, reg_se_mat)

# Now, want to construct covariance matrices to feed into mash
cov_mat_list <- list()

cov_mat_list[['no_effect']] <- matrix(
  data = rep(0, 4), nrow = 2, dimnames = list(
    rows = c("ctrl", "hs"), cols = c("ctrl", "hs")
  )
)

cov_mat_list[['hs_spec']] <- matrix(
  data = c(0, 0, 0, 1), nrow = 2, byrow = TRUE, dimnames = list(
    rows = c("ctrl", "hs"), cols = c("ctrl", "hs")
  )
)

cov_mat_list[['ctrl_spec']] <- matrix(
  data = c(1, 0, 0, 0), nrow = 2, byrow = TRUE, dimnames = list(
    rows = c("ctrl", "hs"), cols = c("ctrl", "hs")
  )
)

desired_corrs <- seq(from = -1, to = 1, by = .25)
desired_amp <- c(1.5, 1.2, 1.1)

for(corr in desired_corrs) {

  cov_mat_list[[glue::glue('equal_corr_{corr}')]] <- make_amp_cov_mat(
    desired_corr = corr, amp = FALSE
  )

  for(cond in c("hs", "ctrl")) {

    for(amp in desired_amp) {

      cov_mat_list[[glue::glue('{cond}_amp_{amp}_corr_{corr}')]] <- make_amp_cov_mat(
        desired_corr = corr, amp_hs = (cond == "hs"), amp_coef = amp
      )

    }

  }
```

```
}

mash_out <- mashr::mash(
  data = mash_data,
  Ulist = cov_mat_list,
  algorithm.version = "Rcpp",
  outputlevel = 1
)

cov_mat_ests <- mashr::get_estimated_pi(mash_out)
```

However, the weights that mash put on the covariance matrices were quite surprising, and looked very different from the weights that were estimated in the GxSex project. The non-zero estimated weights of covariance matrices are shown below:

```
print(cov_mat_ests[cov_mat_ests > .00001])
```

```
##                   null        equal_corr_0.25    hs_amp_1.5_corr_0.5
##            0.003203001            0.089906675            0.129786689
##  ctrl_amp_1.5_corr_0.5   hs_amp_1.5_corr_0.75 ctrl_amp_1.5_corr_0.75
##            0.153567461            0.307282066            0.306596240
##        hs_amp_2_corr_1
##            0.009657867
```

Of note, only .3% of the effects are null. I'm curious if perhaps due to LD between SNPs this occurs, where many SNPs have no causal effect but perhaps are correlated enough such that the estimated posterior effects are non-zero (but may be quite small). In Pallares et al, this effect is likely controlled for via the 10% FDR control. In addition, the weight on the amplification matrices seems roughly equivalent in the cases of high-sugar amplification vs. control. However, this is not what I would expect, given that in general the estimated regression coefficients for the high sugar group (amongst significant SNPs) has a larger mean in magnitude than the estimated regression coefficients for the control group. I attempted repeating the analysis with a different grid (i.e. making the range of amplification coefficients smaller), but the results were relatively similar. To make this analysis more useful, it may make sense to substantially downsample the SNPs using a method similar to what Carrie did.

Examining the correlation between interaction and linear effects

I'm still waiting on the data for this. To run the full analysis on my local machine, it would take about 10 days. I could just look at some subset of the data based on the original Pallares et al. analysis, but I imagine the analysis would be much better if I had access to the data. It should be easy to get this soon.

Bayesian model of amplification effect

Another potential route (much simpler but less flexible than MASH) to demonstrate the nature of the GxE effects seen in the Drosophila dataset is to explictly model some sort of amplification coefficient. One way to do this in a Bayesian context is as follows:

First, assume that we have SNPs labelled $1, ..., n$, where SNP i has true effect θ_i. Then, assume each value of θ is drawn independently as

$$\theta_1, ..., \theta_n \sim N(\mu_\theta, \sigma^2_\theta),$$

where we put flat, uninformative priors on μ_θ and σ^2_θ.

Then, assume we have observations $h_1, ..., h_n$ in the high sugar group with standard errors $s_{1h}, ..., s_{nh}$, and observations $c_1, ..., c_n$ with standard errors $s_{1c}, ..., s_{nc}$. We also assume there exists an amplification coefficient α for the effect in the high sugar group, which we give a flat, uninformative prior. Then, we have the following sampling statements:

$$
\begin{aligned}
c_i &\sim N(\theta_i, s_{ic}) \\
h_i &\sim \pi_0 N(\theta_i, s_{ih}) + (1 - \pi_0) N((1 + \alpha)\theta_i, s_{ih})
\end{aligned}
$$

Where π_0 is a mixture component (with an uninformative prior) indicating the probability of drawing an h_i without amplification.

This model is easily constructed in STAN

```
data {
 int < lower = 1 > N; // Sample size
 vector[N] h; // high sugar measured effects
 vector[N] c; // control measured effects
 vector<lower = 0>[N] s_h; // high sugar se
 vector<lower = 0>[N] s_c; // control se
}

parameters {

 real<lower = 0, upper = 1> pi_0; // Mixture model proportion
 vector[N] theta; // vector of mean parameters
 real mu_theta; // mean of theta parameters
 real<lower = 0> sigma_theta; // sd of the theta parameters
 real alpha; // amplification coefficient

}

model {

 pi_0 ~ beta(.5, .5); // uninformative prior on pi_0
 mu_theta ~ normal(0, 10);
 sigma_theta ~ normal(0, 5);
 theta ~ normal(mu_theta, sigma_theta);
 alpha ~ normal(0, 10);

 c ~ normal(theta, s_c);

 target += log_sum_exp(log(pi_0) + normal_lpdf(h | theta, s_h),
                       log(1 - pi_0) + normal_lpdf(h | (1 + alpha) * theta, s_h));

}
```

Now, we can generate simulated data from the distribution above and examine output in STAN.

```
generate_stan_model_data <- function(n, pi_0, mu_theta, sigma_theta, alpha) {

  thetas <- rnorm(n = n, mean = mu_theta, sd = sigma_theta)
  c_sds <- extraDistr::rhnorm(n = n, sigma = 1)
  c_vec <- rnorm(n = n, mean = thetas, sd = c_sds)
  h_sds <- extraDistr::rhnorm(n = n, sigma = 1)
```

```
  mixt_values <- rbinom(n = n, size = 1, prob = 1 - pi_0)
  num_alt <- sum(mixt_values)
  h_vec <- c(
    rnorm(
      n = n - num_alt,
      mean = thetas[1:(n - num_alt)],
      sd = h_sds[1:(n - num_alt)]
    ),
    rnorm(
      n = num_alt,
      mean = (1 + alpha) * thetas[(n - num_alt + 1):n],
      sd = h_sds[(n - num_alt + 1):n]
    )
  )

  sim_data <- list(
    N = n,
    h = h_vec,
    c = c_vec,
    s_h = h_sds,
    s_c = c_sds
  )

  return(sim_data)

}

model_data <- generate_stan_model_data(
  n = 2000, pi_0 = .5, mu_theta = 0, sigma_theta = .25, alpha = .5
)

fit_model <- sampling(amp_mixt_mod)
```

Unfortunately, this model seems to have a lot of convergence issues. This could perhaps be fixed by more informative priors or by changing modelling assumptions. One particular issue with this model is that the number of parameters grows linearly with the sample size n. But without a sufficiently large n, inferring the other parameters (especially the amplification coefficient), will be difficult.

Moreover, I'm not convinced that this model would really prove that there is some sort of amplification effect in the data. Because we're making relatively strong assumptions about the data, I fear that we're forcing the model into a very constrained space of hypotheses.