An Empirical Bayes Approach to Identification

Charles Zheng

05/21/2015

Setup

A scientist and statistician are collaborating on understanding the visual system. The scientist presents a series of K stimuli (pictures) to the subject and records the subject's response to each stimuli as a p-dimensional vector. Let y_i denote the subject's responses to the i th picture.

The statistican seeks to model the subject's response to the stimuli in terms of a set of q stimulus features. For example, the statistician might define a library of $q = 10000$ Gabor filters to featurize each picture. Let x_i denote the q-dimensional feature vector for each picture.

Let us assume the following multivariate regression model for the data. We have

$$y_i = x_i^T B + \epsilon_i$$

so the subject's response is a linear function of the image features plus noise. Here B is a $q \times p$ matrix. Let us write $\vec{B}$ as the $qp \times 1$ vectorized representation of B, and further suppose that $\vec{B}$ is a random variate

$$\vec{B} \sim N(0, \Sigma_B)$$

and that the ϵ_i are indenpendent of the B, with ϵ_i i.i.d.

$$\epsilon_i \sim N(0, \Sigma_\epsilon)$$

Let Y denote the matrix of responses $Y = [(y_1^1)^T, ..., (y_K^2)^T]$ and X denote the corresponding design matrix $X = [x_1^T, ..., x_K^T]$, so that the model can be written

$$Y = XB + E$$

Further, let $\vec{Y}$ denote the vectorized matrix of responses and $Z = I_p \otimes X$ the corresponding expanded design matrix, so that the model can be written

$$\vec{Y} = Z\vec{B} + \vec{E}$$

The assumed dsitribution of the noise can be restated as

$$\vec{E} \sim N(0, \Sigma_E)$$

where $\Sigma_E = \Sigma_\epsilon \otimes I_{2K}$.

```
set.seed(0)
library(pracma)
library(magrittr)
```

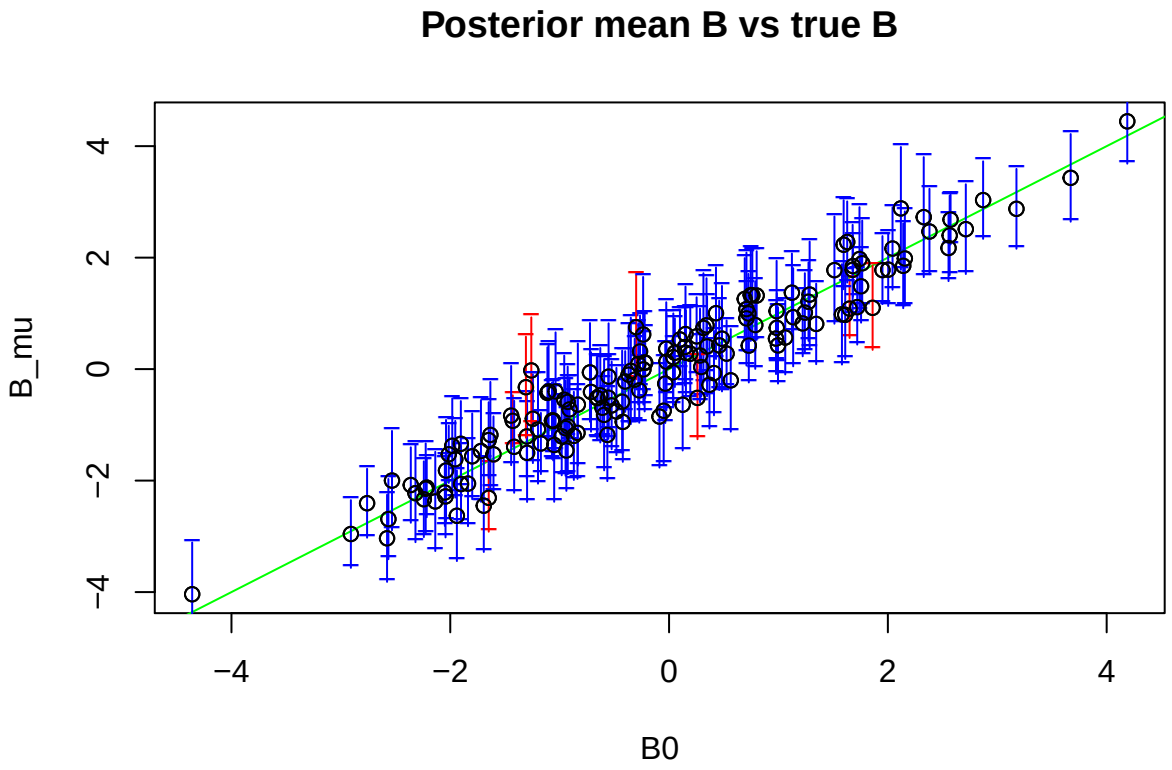
```
##
## Attaching package: 'magrittr'
##
## The following objects are masked from 'package:pracma':
##
##     and, mod, or
```

```r
library(MASS)
K <- 30 # number of stimuli
p <- 20 # dimension of response
q <- 8 # dimension of features
# true noise covariance
Sigma_E <- 2/p * randn(2 * p, p) %>% { t(.) %*% . }
# true signal covariance = sB * I
sB0 <- 2
B0 <- sqrt(sB0) * randn(q, p)
# random features
X <- randn(K, q)
# noise
E <- mvrnorm(K, rep(0, p), Sigma_E)
# observations
Y <- X %*% B0 + E
```

Now supposing the statistican knows the exact value of Σ_β and Σ_ϵ, it is a standard result in Bayesian inference that the posterior distribution for $\vec{B}$ can be obtained as

$$\vec{B} \sim N((Z^T\Sigma_E^{-1}Z + \Sigma_B^{-1})^{-1}Z^T\Sigma_E^{-1}\vec{Y}, (Z^T\Sigma_E^{-1}Z + \Sigma_B^{-1})^{-1})$$

```r
# posterior mean
B_mu <- solve(t(X) %*% X + 1/sB0 * diag(rep(1, q))) %*% t(X) %*% Y
# posterior variance per column
post_cov_B_col <- solve(t(X) %*% X + 1/sB0 * diag(rep(1, q)) )
# marginal variances of B
post_var_B <- t(t(diag(post_cov_B_col))) %*% diag(Sigma_E)
# upper and lower intervals
B_up <- B_mu + 2 * sqrt(post_var_B)
B_low <- B_mu - 2 * sqrt(post_var_B)
# plot
plot(B0, B_mu, main = "Posterior mean B vs true B"); abline(0, 1, col = "green")
for (i in 1:q) { for (j in 1:p) {
  if (B_low[i,j] < B0[i, j] && B_up[i, j] > B0[i, j]) {
    cc <- "blue"
  } else {
    cc <- "red"
  }
  lines(rep(B0[i, j], 2), c(B_low[i, j], B_up[i, j]), col = cc)
  points(rep(B0[i, j], 2), c(B_low[i, j], B_up[i, j]), pch = "-", col = cc)
}}
points(B0, B_mu)
```



The posterior distribution of μ_i are given as follows:

$$\mathrm{E}(\mu_i) = (x_i)^T \mathrm{E}(B)$$

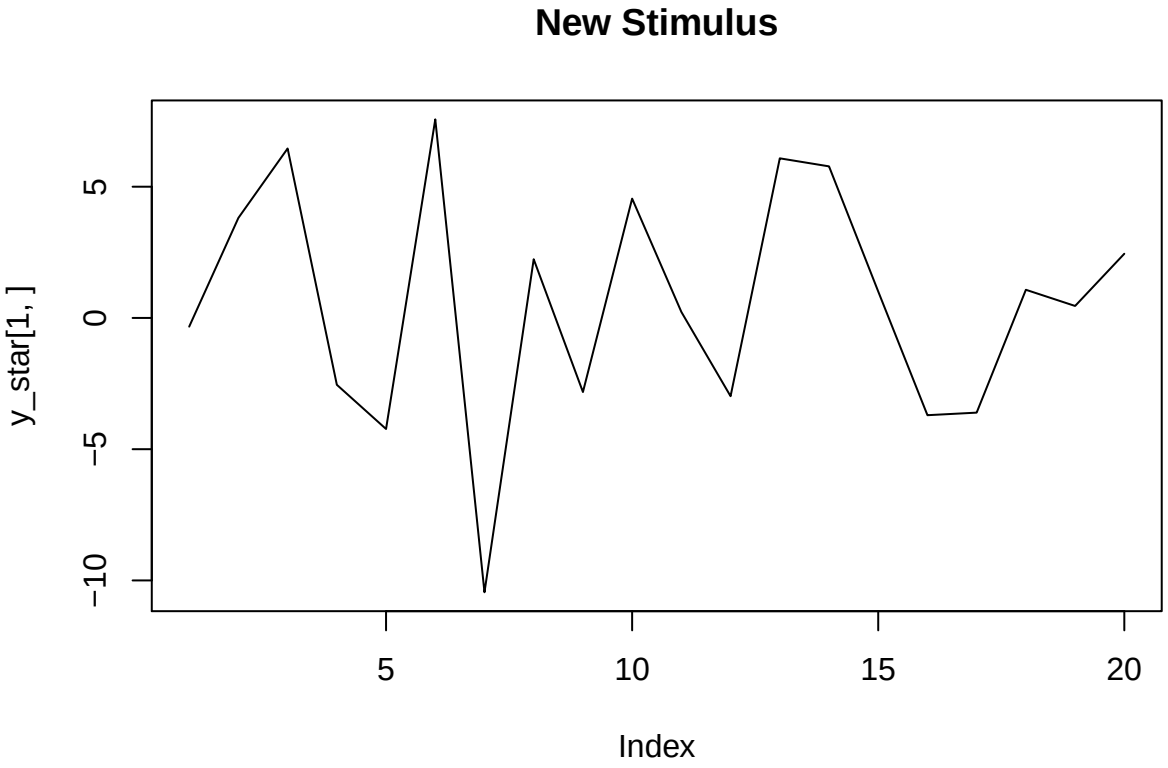
$$\mathrm{Cov}(\mu_i, \mu_j) = (I_p \otimes (x_i)^T)\mathrm{Cov}(B)(I_p \otimes x_j)$$

```
post_cov_B_vec <- solve(solve(Sigma_E) %x% (t(X) %*% X) + 1/sB0 * diag(rep(1, p * q)))
f_post_cov_mu <- function(xi, xj) {
  (diag(rep(1, p)) %x% t(xi)) %*% post_cov_B_vec %*% (diag(rep(1, p)) %x% t(t(xj)))
}
```

Identification

Now the scientist wishes to test the statistician's model. The scientist obtains *new* stimuli $x_1^{te}, ..., x_L^{te}$. They randomly present one of the stimuli $i^* \sim Unif\{1, ..., L\}$ and record the response of the subject, y^*. The scientist challenges the statistician to identify which stimuli x_i^{te} from among $\{x_1^{te}, ..., x_L^{te}\}$ was presented to the subject, based on the response y^*.

```
# new stimuli
L <- 3
X_te <- randn(L, q)
i_chosen <- sample(L, 1)
y_star <- X_te[i_chosen, , drop = FALSE] %*% B0 +  mvrnorm(1, rep(0, p), Sigma_E)
plot(y_star[1, ], type = "l", main = "New Stimulus")
```

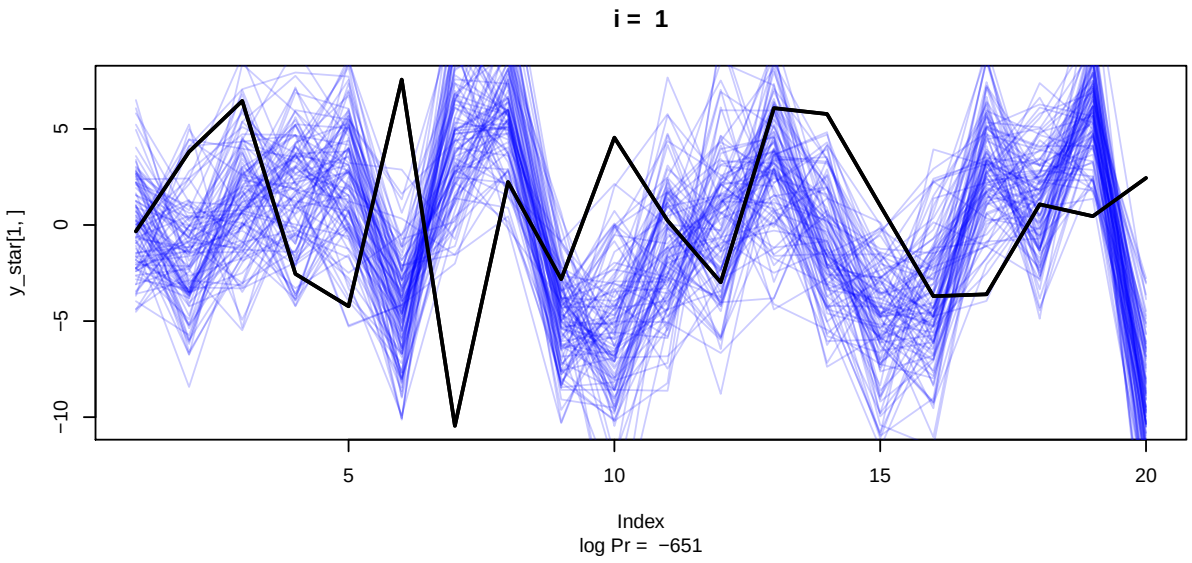


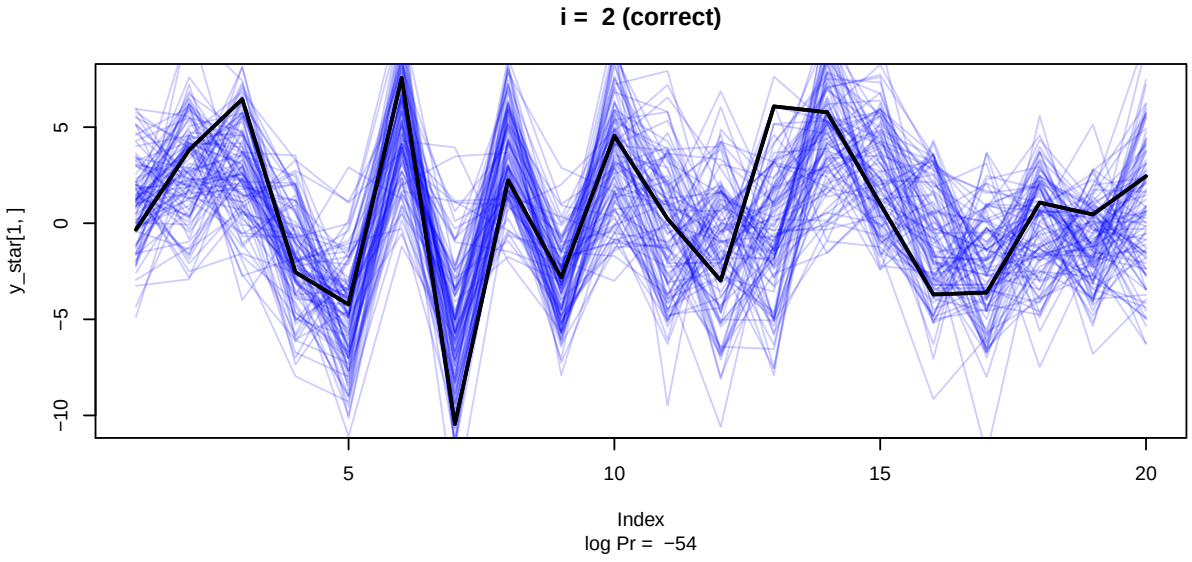
The statistician can compute the posterior distribution of μ_i^{te} for each of the candidate images, and compute the posterior probability that the new response came from each stimulus. The log posterior probability is proportional to

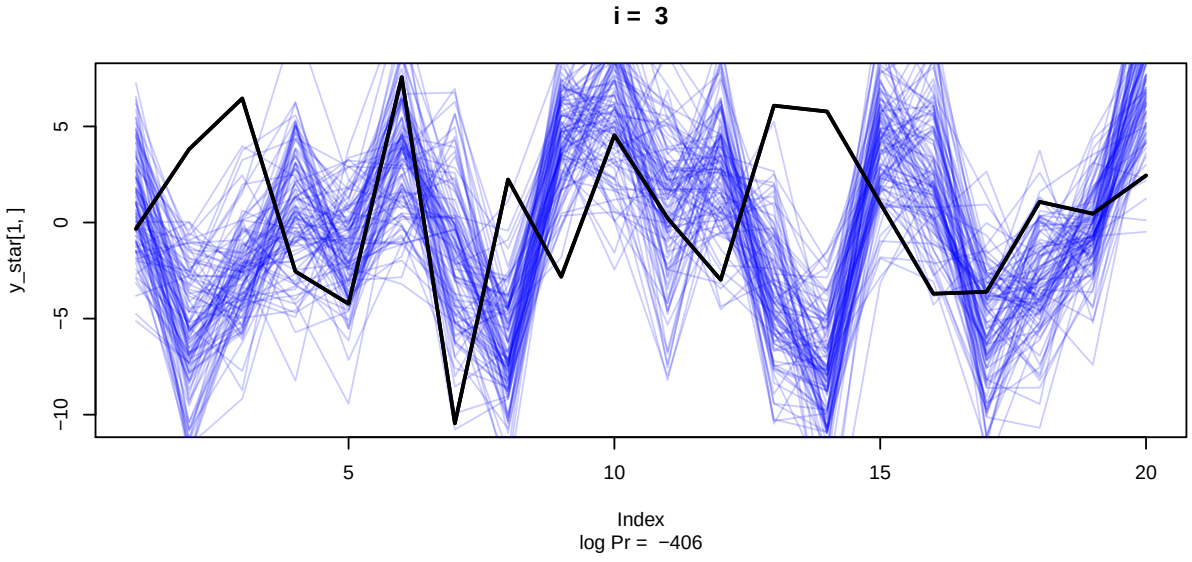
$$\Pr(i^* = i) = -\log\det(\mathrm{Cov}(\mu_i^{te})) - (y^* - \mathrm{E}[\mu_i^{te}])^T(\mathrm{Cov}(\mu_i^{te}) + \Sigma_\epsilon)^{-1}(y^* - \mathrm{E}[\mu_i^{te}])$$

The plot shows y^* superimposed on posterior draws from each distribution.

```
layout(matrix(1:3, 3, 1))
mus <- list(L)
covs <- list(L)
cc <- rgb(0, 0, 1, alpha = 0.2)
for (i in 1:L) {
  mus[[i]] <- X_te[i, , drop = FALSE] %*% B_mu
  covs[[i]] <- f_post_cov_mu(X_te[i, ], X_te[i, ]) + Sigma_E
  pprob <- -log(det(covs[[i]])) - (y_star - mus[[i]]) %*% solve(covs[[i]]) %*% t(y_star - mus[[i]])
  plot(y_star[1, ], type = "l", lwd = 2)
  for (j in 1:100) lines(mvrnorm(1, mu = mus[[i]], Sigma = covs[[i]] + Sigma_E), col = cc)
  if (i == i_chosen) {
    title(paste("i = ", i, "(correct)"), sub = paste("log Pr = ", floor(pprob)))
  } else {
    title(paste("i = ", i), sub = paste("log Pr = ", floor(pprob)))
  }
  lines(y_star[1, ], type = "l", lwd = 2)
}
```


i = 1
y_star[1,]
5
0
−5
−10
5
10
15
20
Index
log Pr = −651


i = 2 (correct)
y_star[1,]
5
0
−5
−10
5
10
15
20
Index
log Pr = −54


i = 3
y_star[1,]
5
0
−5
−10
5
10
15
20
Index
log Pr = −406

```
layout(1)
```

Empirical Bayes

The previous approach relies on knowing Σ_ϵ, Σ_B. However, in practice, we seldom have a prior estimate of these quantities. We can estimate $\hat{\Sigma}_\epsilon$ from the covariance of the residuals. But what about Σ_B? Taking advantage of the connection between ridge regression and the normal prior $\beta \sim N(0, \lambda^{-1} I)$, we propose to estimate Σ_B using the optimal ridge regression shrinkage values.

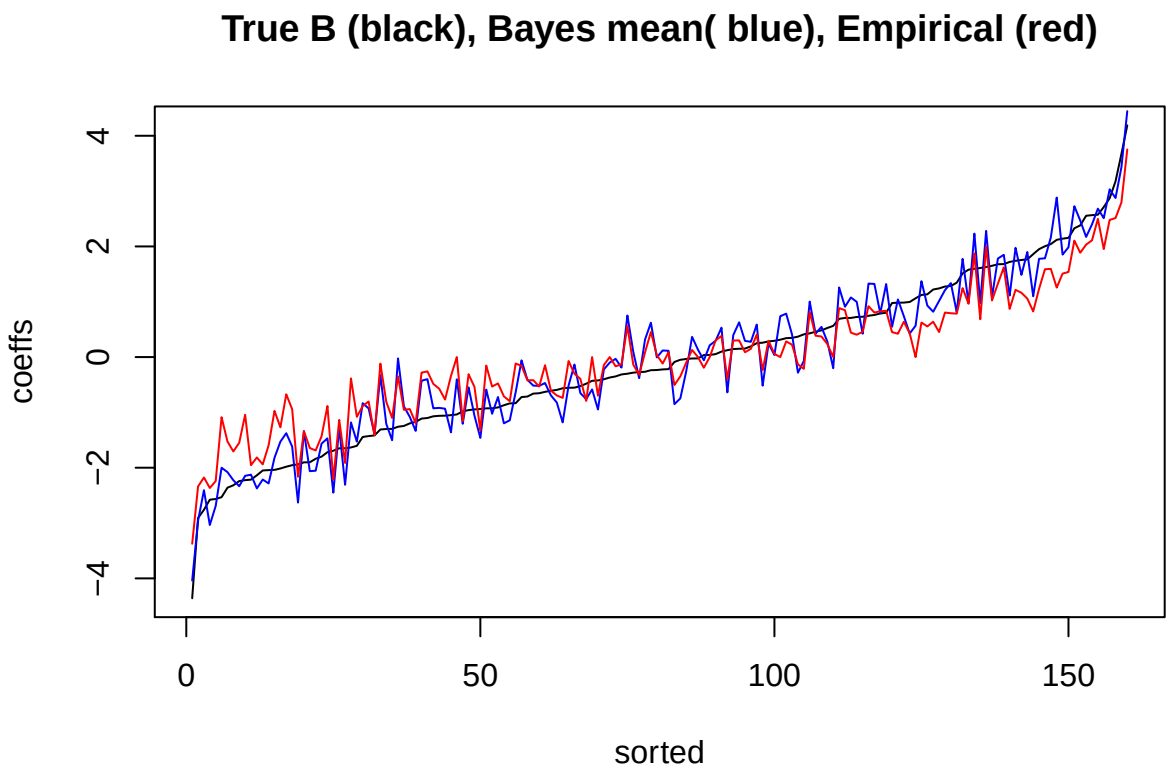
```
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:pracma':
##
##     expm, lu, tril, triu
##
## Loading required package: foreach
## Loaded glmnet 2.0-2
```

```
B_mu_EB <- matrix(0, q, p) # placeholder for estimate
resids <- matrix(0, K, p)
lambdas <- numeric(p) # lambda for each column of B
for (i in 1:p) {
  res <- cv.glmnet(X, Y[, i], alpha = 0, intercept = FALSE, standardize = FALSE)
  lambdas[i] <- res$lambda.1se
  pre <- predict(res, s = lambdas[i], newx = X)
  resids[, i] <- pre - Y[, i]
  B_mu_EB[, i] <- as.numeric(coefficients(res))[-1]
}
```

Compare $\hat{B}_{EB}$ from empirical bayes, $\hat{B}_{bayes}$ and true B.

```
o <- order(as.numeric(B0))
plot(as.numeric(B0)[o], type = "l", xlab = "sorted", ylab = "coeffs")
lines(as.numeric(B_mu)[o], col = "blue")
lines(as.numeric(B_mu_EB)[o], col = "red")
title("True B (black), Bayes mean( blue), Empirical (red)")
```

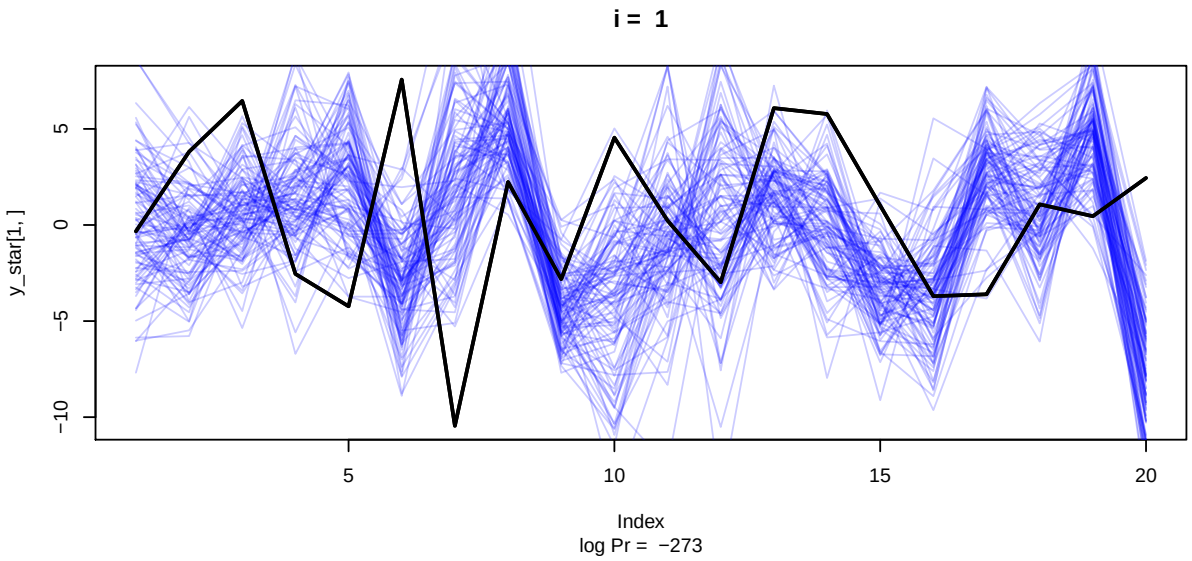


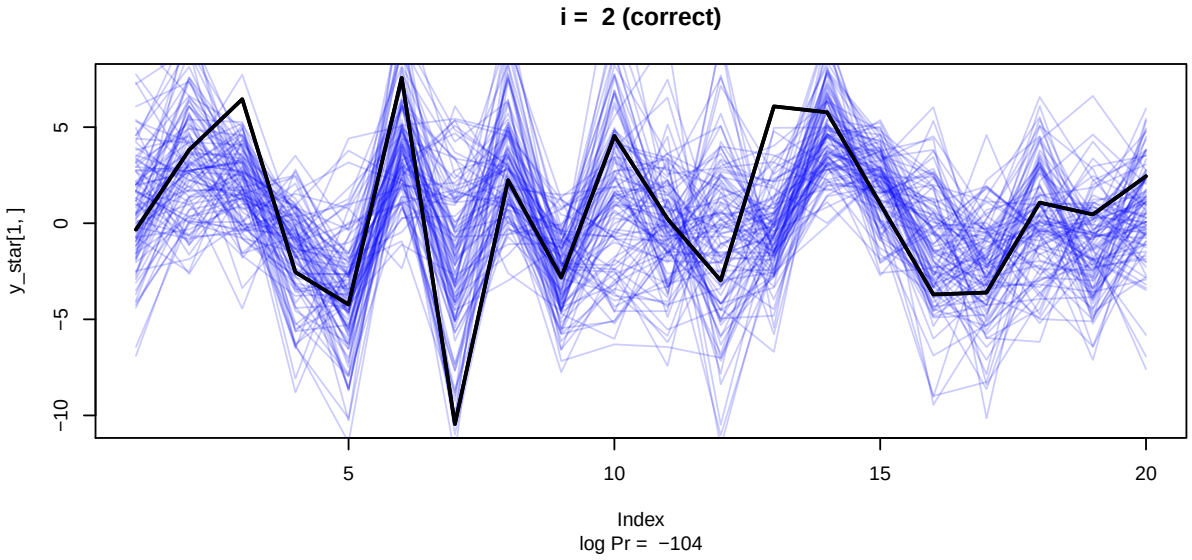
Form empirical bayes estimates of Σ_E and Σ_B

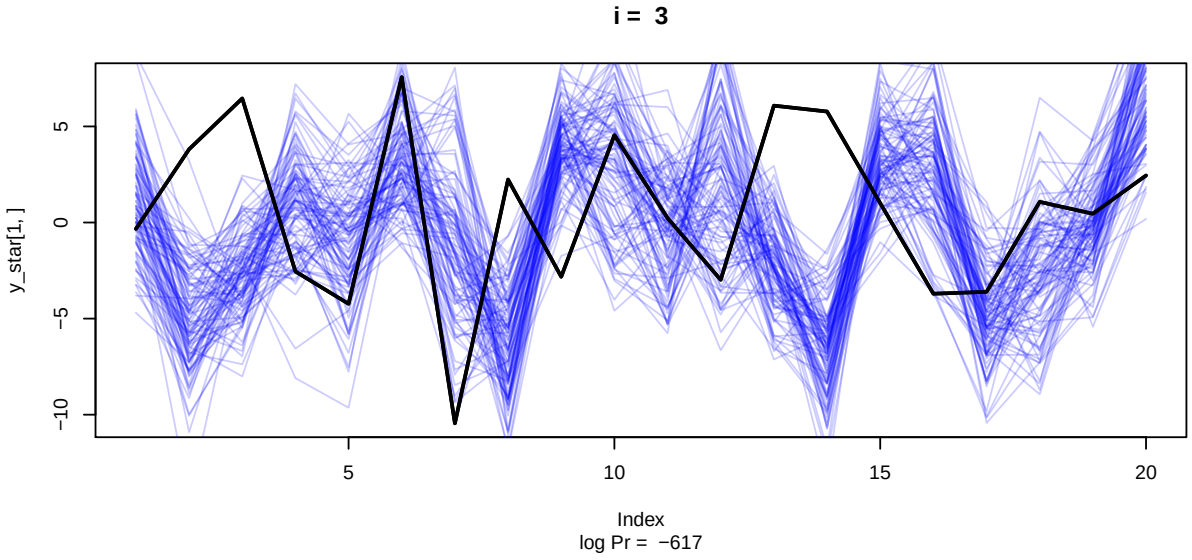
```
Sigma_E_hat <- cov(resids)
Inv_Sigma_B <- diag(rep(lambdas, each = q))
EB_cov_B_vec <- solve(solve(Sigma_E_hat) %x% (t(X) %*% X) + Inv_Sigma_B)
f_EB_cov_mu <- function(xi, xj) {
  (diag(rep(1, p)) %x% t(xi)) %*% EB_cov_B_vec %*% (diag(rep(1, p)) %x% t(t(xj)))
}
```

Identify y^* using Empirical Bayes.

```
layout(matrix(1:3, 3, 1))
mus <- list(L)
covs <- list(L)
cc <- rgb(0, 0, 1, alpha = 0.2)
for (i in 1:L) {
  mus[[i]] <- X_te[i, , drop = FALSE] %*% B_mu_EB
  covs[[i]] <- f_EB_cov_mu(X_te[i, ], X_te[i, ]) + Sigma_E_hat
  pprob <- -log(det(covs[[i]])) - (y_star - mus[[i]]) %*% solve(covs[[i]]) %*% t(y_star - mus[[i]])
  plot(y_star[1, ], type = "l", lwd = 2)
  for (j in 1:100) lines(mvrnorm(1, mu = mus[[i]], Sigma = covs[[i]] + Sigma_E_hat), col = cc)
  if (i == i_chosen) {
    title(paste("i = ", i, "(correct)"), sub = paste("log Pr = ", floor(pprob)))
  } else {
    title(paste("i = ", i), sub = paste("log Pr = ", floor(pprob)))
  }
  lines(y_star[1, ], type = "l", lwd = 2)
}
```


i = 1
y_star[1,]
5
0
−5
−10
5
10
15
20
Index
log Pr = −273


i = 2 (correct)
y_star[1,]
5
0
−5
−10
5
10
15
20
Index
log Pr = −104


i = 3
y_star[1,]
5
0
−5
−10
5
10
15
20
Index
log Pr = −617

```r
layout(1)
```