The CATE Package for High Dimensional Factor Analysis and Confounder Adjusted Multiple Testing

Jingshu Wang and Qingyuan Zhao

Department of Statistics, Stanford University
{jingshuw,qyzhao}@stanford.edu

October 4, 2015

In this document, we demonstrate how to use the R package cate to adjust for confounding effects in multiple hypothesis testing. Users who are only interested in high dimensional factor analysis may skip the motivating example and go to Section 4.

1 Introduction

The dataset we will be using is a genome-wide association study of gender (Vawter et al., 2004). In this study, samples were taken postmortem from the brains of 10 individuals, 5 men and 5 women. Three samples were taken from different regions of the brain of each individual, and one aliquot of each sample was sent to each of 3 laboratories for analysis. There were two different microarray platforms used by these labs. The dataset was downloaded from .

First, let’s load the data.

```
library(cate)

## Creating a generic function for ’nchar’ from package ’base’ in package ’S4Vectors’

data(gender.sm)
names(gender.sm)

##  [1] "Y"             "X"             "Z"             "spikectl"      "geneinfo"
##  [6] "xchrom"        "ychrom"        "xychrom"       "pctl"          "autosomal"
## [11] "genecoloring"  "samplecoloring"

cbind(X = dim(gender.sm$X), Y = dim(gender.sm$Y), Z = dim(gender.sm$Z)) # matrix dimensions

##       X   Y  Z
## [1,] 84  84 84
## [2,]  1 500  4
```

There are in total 84 samples and 500 genes in this dataset. There are 12600 genes measured in the original dataset (Vawter et al., 2004; Gagnon-Bartsch and Speed, 2012) and here we look at a 500 subsample. There should have been $10 \times 3 \times 3 = 90$ samples in total, but 6 of them are missing. In the data object gender.sm, X is the gender of each person, Y is the gene expression matrix, and Z includes the batch labels (lab and microarray platform, not individual or brain region).

Since there are several batch variables and unmeasured covariates in this dataset, it should come as no surprise that these confounders can seriously bias the association tests. In other words, the marginal effects of X on Y may be different from the actual effects of X on Y. Wang et al. (2015) describe a general

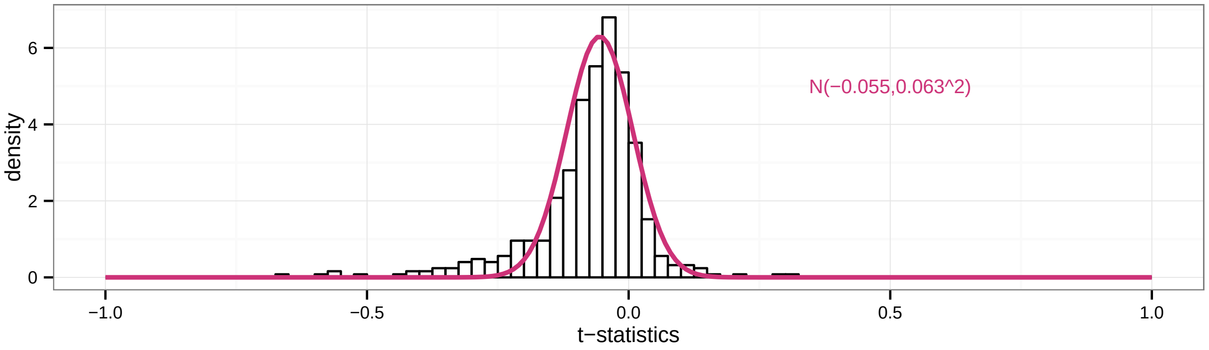

Figure 1: Histogram of t-statistics before confounder adjustment

framework and a two-step solution for this problem. In the first step, we apply factor analysis to the part of gene expression matrix Y that is unrelated to the variable(s) of interest X. In the second step, we correct the marginal effects of X on Y by using the factors obtained in the first step.

2 Confounder adjusted multiple testing

To illustrate how the confounders in the `gender.sm` dataset can bias the association tests, we first run the two-sample t-test for each gene

```
t.stats <- apply(gender.sm$Y, 2, function(y, x) t.test(y~x)$statistic, gender.sm$X)
```

Under the null hypothesis that the gene is unrelated to the gender, the corresponding t-statistic is expected to follow a t-distribution with $n-1$ degrees of freedom. Since $n = 84$ is fairly large in this study, this t-distribution is very close to the standard normal distribution $\mathrm{N}(0,1)$. However, the empirical distribution of `t.stats` is shown in the next histogram (code not echoed):

It clearly departs from the theoretical null distribution. The median absolute deviation (MAD) of this histogram is only 0.066, much less than the theoretical value 1, i.e. the t-statistics are extremely underdispersed. The histogram is also a little skewed compared to normal or t distribution.

As mentioned earlier, this phenomenon is most likely due to the existence of **confounders**, unmeasured variables that are correlated with both the response Y and the variable of interest X. To correct for such confounders, we first need to estimate its number. This is implemented in the function `est.confounder.num`

```
n <- nrow(gender.sm$Y) # number of samples
gender.data <- data.frame(gender = gender.sm$X, gender.sm$Z)
factor.num <- est.confounder.num(~ gender | . - gender + 0,
                                 gender.data, gender.sm$Y,
                                 method = "bcv", bcv.plot = FALSE,
                                 rmax = 30, nRepeat = 20)
factor.num$r

## [1] 16
```

By default, `est.confounder.num` uses `method = "bcv"`, which calls the `EsaBcv` function in the `esaBcv` package to estimate the number of factors by bi-cross-validation (BCV). To look at the curve of BCV error using different number of factors, the user can turn on the `bcv.plot` argument. It is recommended to use at least 20 for `nRepeat` to obtain a stable BCV error for `n` around a few hundreds or less.

Another method to estimate the number of confounders of is the eigenvalue difference method (Onatski, 2010)

```r
est.confounder.num(~ gender | . - gender + 0,
                   gender.data, gender.sm$Y, method = "ed")

## [1] 2
```

The formula " gender | . - gender + 0" here means gender is the primary variable and all other variables in `gender.data` are nuisance variables. The intercept is not included because it is already included in `gender.data`. The `"bcv"` method is better at estimating weak factors but takes longer time than `"ed"`. We recommend to use `"bcv"` method for most datasets; see Owen and Wang (2015) for more detail.

After finding the number of factors, the user can call the main `cate` function to adjust for the confounders

```r
cate.results <- cate(~ gender | . - gender + 0,
                     gender.data, gender.sm$Y, r = factor.num$r)
names(cate.results)

##  [1] "Gamma"        "Sigma"        "Z"            "niter"        "converged"
##  [6] "alpha"        "beta"         "beta.cov.row" "beta.cov.col" "beta.t"
## [11] "beta.p.value" "Y.tilde"      "alpha.p.value"
```

For most users, the interesting returned values are

`beta`: the estimated effects after adjustment;

`beta.t`: the t-statistics after adjustment;

`beta.p.value`: the p-values of the estimated effects;

`alpha.p.value`: the p-value of a χ^2-test for confounding.

The first thing to look at is perhaps the confounding test, whose null hypothesis is that the estimated factors are not correlated with X so the individual association tests are not biased. For the gender dataset, the p-value of this test is

```r
cate.results$alpha.p.value

##       gender
## 0.0001005444
```

This is much smaller than 0.05, which indicates, together with the previous histogram, that there are some confounders in the experiment. When this is the case, the user may want to look at the factor analysis results to search for possible sources of confounding (in particular the loadings `Gamma` and the estimated confounders `Z` returned by `cate`).

To discover candidate genes, the user can apply the `p.adjust` function in the `stats` package to control certain multiple testing error. For controlling the family-wise error rate (FWER), the user may use the `bonferroni` or `holm` option in `p.adjust`. To increase the number of findings and still control the false discovery rate (FDR), the user may use the `BH` option. Here are a couple of examples:

```r
which(p.adjust(cate.results$beta.p.value, "bonferroni") < 0.05) # control FWER at 0.05
which(p.adjust(cate.results$beta.p.value, "BH") < 0.2) # control FDR at 0.2
```

Finally, let's review all the available options in `cate`.

```r
args(cate)

## function (formula, X.data = NULL, Y, r, fa.method = c("ml", "pc",
##     "esa"), adj.method = c("rr", "nc", "lqs", "naive"), psi = psi.huber,
##     nc = NULL, nc.var.correction = TRUE, calibrate = TRUE)
## NULL
```

Here are some detailed descriptions of all the arguments

`formula`: R formula indicating primary and nuisance predictors, which are separated by "|". Example: `formula =   treatment | batch.label`.

`Y`: response matrix (e.g. gene expression).

`X.data`: a data frame including both primary treatment variables whose effects are of interest and nuisance covariates to adjust for (for example the known batch variables and other demographics variables such as age or gender). `X` can also be a formula starting with .

`primary.var.names` a vector of strings indicating the names of the primary treatment variables, or an integer vector indicating the columns of the primary treatment variables in `X` when `X` is a matrix or data frame. If `primary.var.names` is a vector of string, then the names should match the column names of `X` or `X.data`.

`r`: number of confounders, usually estimated by `est.confounder.num`.

`fa.method`: method used to estimate the confounders. See Section 4 for more detail.

`adj.method`: method used to adjust for the confounders. There are two main approaches: robust regression (`rr`) and negative control (`nc`). If a fair amount (rule of thumb: ≥ 30) of negative control genes (e.g. spike-in controls) are available, it is recommended to use the `nc` option. Housekeeping genes can also be used as negative controls, but they are not as reliable as spike-in controls. The robust regression (`rr`) option assumes the true effects are sparse and can be used when negative controls are not available.

`psi`: estimating equation function used when `adj.method = "rr"`. See the `rlm` function in package `MASS` for more detail.

`nc`: positions of the negative controls. Can be a vector of numbers between 1 and p, or a logical vector of length p.

`nc.var.correction`: if `TRUE` (default and recommended), use the variance correction formula in Wang et al. (2015) when `adj.method = "nc"`; if `FALSE`, use the oracle variance (same as the `RUV4` function in package `ruv`). See Wang et al. (2015) for more detail.

`calibrate`: if `TRUE` (default), scale the t-statistics `beta.t` to have median equal to 0 and median absolute deviation (with respect to normal distribution) equal to 1 .

For example, to use the `nc` adjustment method, the user must specify the positions of negative control genes. In the dataset `gender.sm`, these are given in `spikectl` (33 spike-in controls) and `hkctl` (799 housekeeping genes). Here is a sample usage:

```
cate.results.nc <- cate(~ gender | . - gender + 0,
                         gender.data, gender.sm$Y, r = factor.num$r,
                         adj.method = "nc", nc = gender.sm$spikectl)
```

The housekeeping genes are usually less reliable than the spike-in controls.

Alternatively, `cate.fit` provides a non-formula interface for the same purpose. To use `cate.fit`, the user needs to specify

`X.primary`: primary treatment variable(s) whose effects are of interest. These are the variables that come before "|" in the `formula` in `cate`.

`X.nuis`: nuisance covariate(s) to include in the regression whose effects are not of interest. These are the variables that come after "|" in the `formula` in `cate`.

`Y`: response matrix (e.g. gene expression).

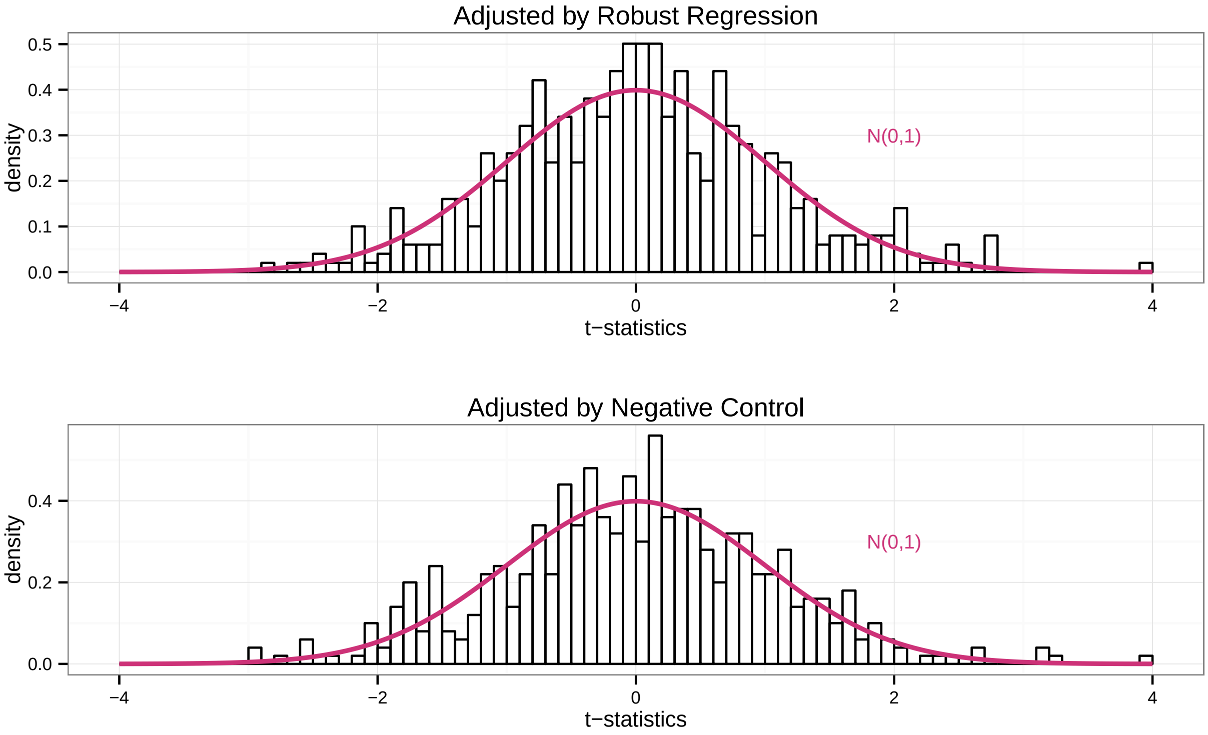

Figure 2: Histograms of test statistics after adjustment

All the other arguments are the same as `cate`.

We end this section with Figure 2, two histograms of `beta.t` after the confounder adjustment using `adj.method = "rr"` and `adj.method = "nc"`. In both cases, the bulk of the statistics approximately follows the standard normal distribution.

3 Other available confounder adjustment methods on CRAN/Bioconductor

The preprocessed `gender.sm` dataset was first used by Gagnon-Bartsch and Speed (2012) to demonstrate their confounder correction method "Remove Unwanted Variation" (RUV) using negative controls. See also the R package `ruv`. The `RUV4` function therein is very similar to the `adj.method = "nc"` option in `cate`. Two other related packages are `sva` (Leek and Storey, 2008) and `leapp` (Sun et al., 2012), which motivate the robust regression method (`adj.method = "rr"`) in `cate`.

To compare the performance of different methods, the user can use the wrapper functions `sva.wrapper`, `ruv.wrapper`, `leapp.wrapper` in the `cate` package. They provide a uniform interface and call the corresponding functions in the original packages.

4 Factor analysis

For a data matrix $Y \in \mathbb{R}^{n\times p}$, n being the number of observations and p being the number of variables (e.g. genes), the factor model assumes

$$Y_{n\times p} = Z_{n\times r}\Gamma^T_{p\times r} + E_{n\times p},$$

where Z is a rotation matrix containing latent factors, Γ is a matrix of loadings and E is a noise matrix. The columns of E have covariance matrix Σ.

To perform factor analysis, one simply calls the function `factor.analysis` with the data matrix and number of factors:

```
mle <- factor.analysis(gender.sm$Y, r = 5)
names(mle)

## [1] "Gamma"     "Sigma"     "Z"         "niter"     "converged"
```

By default, `factor.analysis` estimates the latent factors by maximum likelihood (`method = "ml"`). Other available algorithms are principal component analysis (`method = "pc"`) and bi-convex optimization via early stopping alternation (`method = "esa"`). The default maximum likelihood estimator has good theoretical properties under heteroscedastic noise variance Σ (Bai and Li, 2012) or approximate factor model (Bai and Li, 2015). The `esa` method tries to minimize the prediction error for Y; for more details we refer the readers to the R package esaBcv. The `pc` option is usually more desirable when the noise variance is homoscedastic $\Sigma = \sigma^2 I_p$. Finally, using the `pc` method will always outputs the same results, but the iterative procedures `ml` and `esa` may converge to different values depending on the initial point.

The `factor.analysis` function can work for any n and p. In contrasts, both the `factanal` function in package `stats` and the `fa` function in package `psych` do not support the high dimensional problem where $p > n$.

References

Bai, J. and K. Li (2012). Statistical analysis of factor models of high dimension. *The Annals of Statistics 40*(1), 436–465.

Bai, J. and K. Li (2015). Maximum likelihood estimation and inference for approximate factor models of high dimension. *The Review of Economics and Statistics to appear*.

Gagnon-Bartsch, J. A. and T. P. Speed (2012). Using control genes to correct for unwanted variation in microarray data. *Biostatistics 13*(3), 539–552.

Leek, J. T. and J. D. Storey (2008). A general framework for multiple testing dependence. *Proceedings of the National Academy of Sciences 105*(48), 18718–18723.

Onatski, A. (2010). Determining the number of factors from empirical distribution of eigenvalues. *The Review of Economics and Statistics 92*(4), 1004–1016.

Owen, A. B. and J. Wang (2015). Bi-cross-validation for factor analysis. *arXiv:1503.03515*.

Sun, Y., N. R. Zhang, and A. B. Owen (2012). Multiple hypothesis testing adjusted for latent variables, with an application to the agemap gene expression data. *The Annals of Applied Statistics 6*(4), 1664–1688.

Vawter, M. P., S. Evans, P. Choudary, H. Tomita, J. Meador-Woodruff, M. Molnar, J. Li, J. F. Lopez, R. Myers, D. Cox, et al. (2004). Gender-specific gene expression in post-mortem human brain: localization to sex chromosomes. *Neuropsychopharmacology 29*(2), 373–384.

Wang, J., Q. Zhao, T. Hastie, and A. B. Owen (2015). Confounder adjustment in multiple testing. *arXiv:1508.04178*.