Introduction to adaptMT package

Lihua Lei

Contents

1 Introduction

This is a demo of our adaptMT package. adaptMT has two main components: a generic interface that allows users to specify the working model and algorithms to fit them, as well as a pool of easy-to-use end-to-end wrappers. The former is captured by function adapt. The latter includes adapt_glm, adapt_gam and adapt_glmnet for using GLM, GAM and L1-regularized GLM.

```
# install the "adaptMT" package from github.
# will be submitted to CRAN very soon.
devtools::install_github("lihualei71/adaptMT")
```

```
## Downloading GitHub repo lihualei71/adaptMT@master
## from URL https://api.github.com/repos/lihualei71/adaptMT/zipball/master
```

```
## Installing adaptMT
```

```
library("adaptMT")
```

We illustrate one of the main function adapt_glm, for AdaPT with logistic-Gamma GLM as the working model, on estrogen dataset, a gene/drug response dataset from NCBI Gene Expression Omnibus (GEO). estrogen dataset consists of gene expression measurements for $n = 22283$ genes, in response to estrogen treatments in breast cancer cells for five groups of patients, with different dosage levels and 5 trials in each. The task is to identify the genes responding to a low dosage. The p-values pi for gene i is obtained by a one-sided permutation test which evaluates evidence for a change in gene expression level between the control group (placebo) and the low-dose group. $\{p_i : i \in [n]\}$ are then ordered according to permutation t-statistics comparing the control and low-dose data, pooled, against data from a higher dosage (with genes that appear to have a strong response at higher dosages placed earlier in the list). The code to compute p-values and the ordering can be found in Rina Barber's website.

In this demo, we subsample the top 5000 genes for illustration.

```
# load the data.
data("estrogen")
# Take the first 5000 genes
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

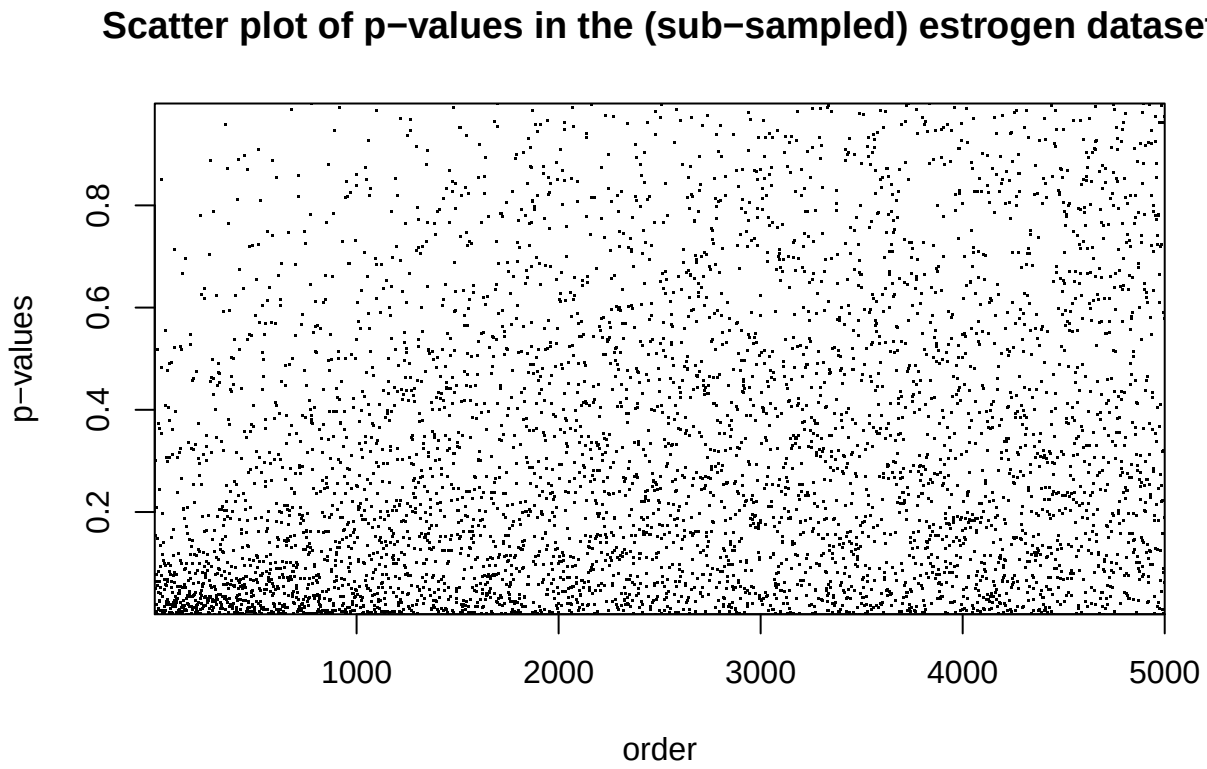
```
estrogen <- select(estrogen, pvals, ord_high) %>%
  filter(ord_high <= 5000)
rownames(estrogen) <- NULL
```

```
head(estrogen, 5)
```

```
##        pvals ord_high
## 1 0.05011062      366
## 2 0.71404053     4772
## 3 0.06675970     3562
## 4 0.40392007      790
## 5 0.40614415     2291
```

```
summary(estrogen)
```

```
##      pvals              ord_high
##  Min.   :0.000011   Min.   :   1
##  1st Qu.:0.076082   1st Qu.:1251
##  Median :0.238279   Median :2500
##  Mean   :0.315094   Mean   :2500
##  3rd Qu.:0.501009   3rd Qu.:3750
##  Max.   :0.999289   Max.   :5000
```



Now we execute `adapt_glm` on this dataset. `adapt_glm` assumes a conditional logistic-Gamma GLM as the working model. Specifically, it models the p-values as

$$H_i \mid x_i \sim \pi(x_i), \quad \text{logit}(\pi(x_i)) = \phi(x_i)^T \beta$$

$$-\log p_i \mid H_i, x_i \sim \begin{cases} \text{Exp}(1) & H_i = 0 \\ \text{Exp}(\mu(\text{x})) & H_i = 1 \end{cases}, \quad \frac{1}{\mu(x_i)} = \phi(x_i)^T \gamma$$

where $\phi(x)$ is a featurization of x. In this example, we use the spline bases, given by `ns` function from `splines` package. For illustration, we choose our candidate models as the above GLMs with $\phi(x)$ being the

spline bases with equal-spaced knots and the number of knots ranging from 6-10. We use BIC to select the best model at the initial stage and use the selected model for the following model fitting.

```
# prepare the inputs of AdaPT
# need "splines" package to construct the formula for glm
library("splines")
pvals <- as.numeric(estrogen$pvals)
x <- data.frame(x = as.numeric(estrogen$ord))
formulas <- paste0("ns(x, df = ", 6:10, ")")
formulas
```

```
## [1] "ns(x, df = 6)"  "ns(x, df = 7)"  "ns(x, df = 8)"  "ns(x, df = 9)"
## [5] "ns(x, df = 10)"
```

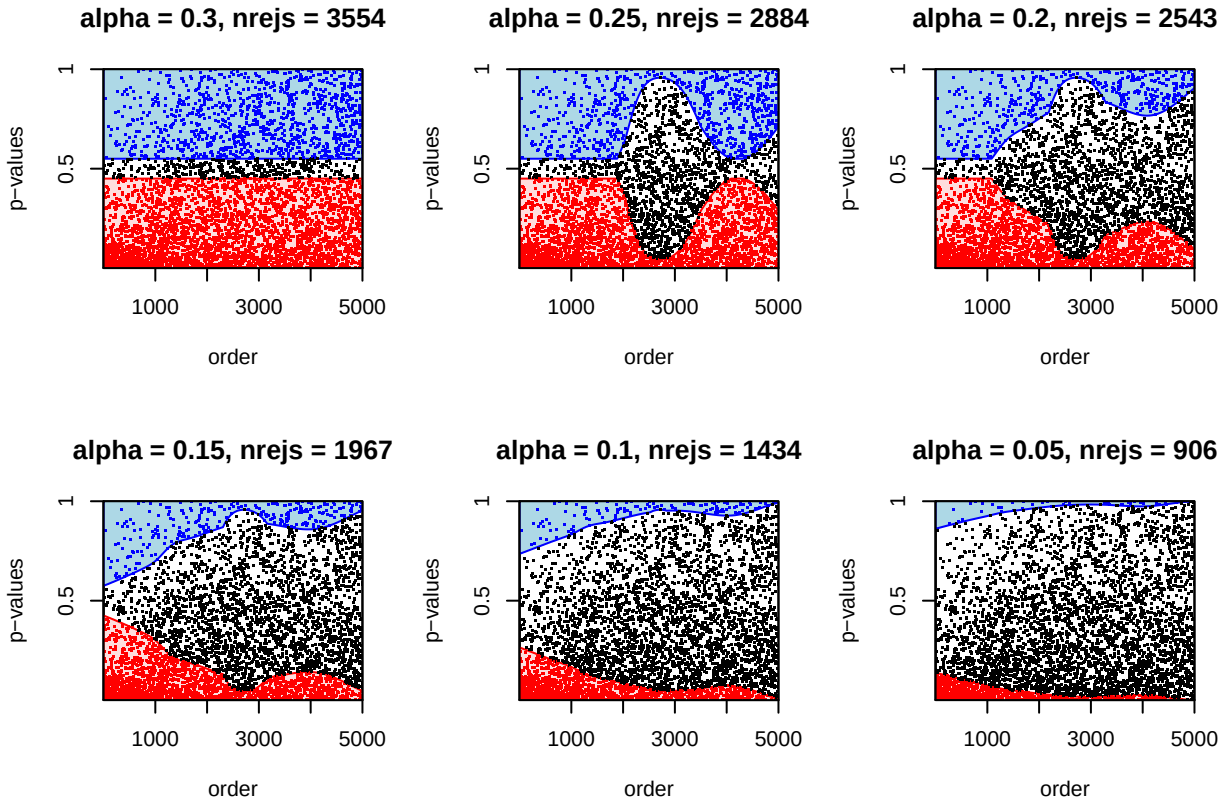
adapt_glm function provides several user-friendly tools to monitor the progress. For model selection, a progress bar will, by default, be shown in the console that indicates how much proportion of models have been fitted. This can be turned off by setting verbose$ms = FALSE. Similarly for model fitting, a progress bar can be shown in the console, though not by default, by setting verbose$fit = TRUE. Also, by default, the progress of the main process will be shown in the console that indicates (1) which target FDR level has been achieved; (2) FDPhat for each target FDR level; (3) number of rejections for each target FDR level.

```
# run AdaPT
res <- adapt_glm(x = x, pvals = pvals, pi_formulas = formulas, mu_formulas = formulas)
```

```
## Model selection starts!
## Shrink the set of candidate models if it is too time-consuming.
##
  |
  |                                                      |   0%
  |
  |==========                                            |  20%
  |
  |====================                                  |  40%
  |
  |==============================                        |  60%
  |
  |========================================              |  80%
  |
  |==================================================| 100%
## alpha = 0.29: FDPhat 0.2899, Number of Rej. 3474
## alpha = 0.28: FDPhat 0.28, Number of Rej. 3325
## alpha = 0.27: FDPhat 0.27, Number of Rej. 3082
## alpha = 0.26: FDPhat 0.2599, Number of Rej. 2928
## alpha = 0.25: FDPhat 0.25, Number of Rej. 2884
## alpha = 0.24: FDPhat 0.24, Number of Rej. 2842
## alpha = 0.23: FDPhat 0.2299, Number of Rej. 2762
## alpha = 0.22: FDPhat 0.22, Number of Rej. 2700
## alpha = 0.21: FDPhat 0.2098, Number of Rej. 2598
## alpha = 0.2: FDPhat 0.1998, Number of Rej. 2543
## alpha = 0.19: FDPhat 0.1898, Number of Rej. 2402
## alpha = 0.18: FDPhat 0.1798, Number of Rej. 2292
## alpha = 0.17: FDPhat 0.1697, Number of Rej. 2234
## alpha = 0.16: FDPhat 0.1598, Number of Rej. 2146
## alpha = 0.15: FDPhat 0.1497, Number of Rej. 2057
## alpha = 0.14: FDPhat 0.1398, Number of Rej. 1967
## alpha = 0.13: FDPhat 0.13, Number of Rej. 1893
```

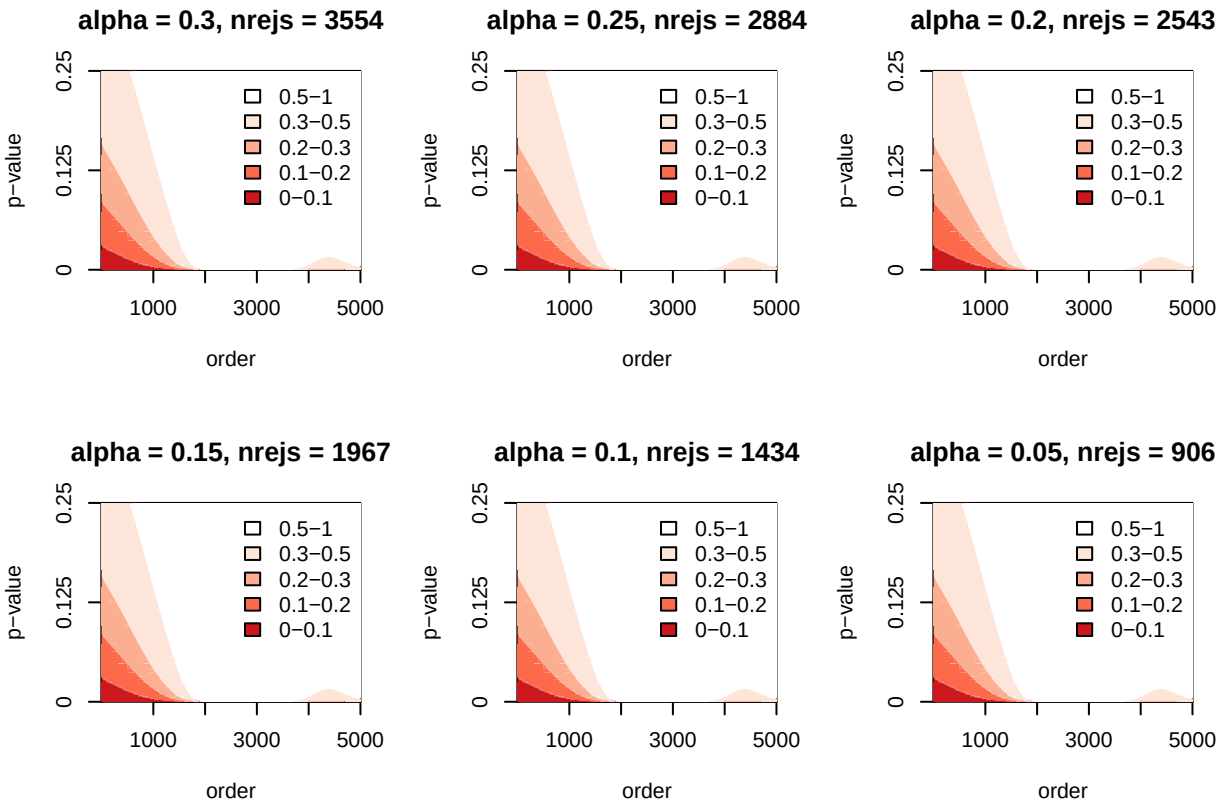
```
## alpha = 0.12: FDPhat 0.1197, Number of Rej. 1821
## alpha = 0.11: FDPhat 0.1099, Number of Rej. 1711
## alpha = 0.1: FDPhat 0.0998, Number of Rej. 1583
## alpha = 0.09: FDPhat 0.09, Number of Rej. 1434
## alpha = 0.08: FDPhat 0.0798, Number of Rej. 1240
## alpha = 0.07: FDPhat 0.0697, Number of Rej. 1104
## alpha = 0.06: FDPhat 0.0593, Number of Rej. 961
## alpha = 0.05: FDPhat 0.0497, Number of Rej. 906
## alpha = 0.04: FDPhat 0.0389, Number of Rej. 796
## alpha = 0.03: FDPhat 0.0299, Number of Rej. 568
## alpha = 0.02: FDPhat 0.0179, Number of Rej. 224
```

`plot_thresh_1d` gives the plot for the rejection threshold as a function of x (must be univariate without repeated value) for given α. We display the plots for $\alpha \in \{0.3, 0.25, 0.2, 0.15, 0.1, 0.05\}$.



`plot_lfdr_1d` gives the plot for the estimated local FDR as a function of x (must be univariate without repeated value) for given α. We display the plots for $\alpha \in \{0.3, 0.25, 0.2, 0.15, 0.1, 0.05\}$. It is clear that the estimated local FDR almost remains the same, indicating that the information loss caused by partial masking is small.

```
par(mfrow = c(2, 3))
for (alpha in seq(0.3, 0.05, -0.05)){
  nrejs <- res$nrejs[floor(alpha * 100)]
  title <- paste0("alpha = ", alpha, ", nrejs = ", nrejs)
  plot_1d_lfdr(res, alpha, title, disp_ymax = 0.25, xlab = "order")
}
```


alpha = 0.3, nrejs = 3554
alpha = 0.25, nrejs = 2884
alpha = 0.2, nrejs = 2543
alpha = 0.15, nrejs = 1967
alpha = 0.1, nrejs = 1434
alpha = 0.05, nrejs = 906
p−value
0
0.125
0.25
1000
3000
5000
order
0.5−1
0.3−0.5
0.2−0.3
0.1−0.2
0−0.1