Tutorial for `swfdr` package

Jeff Leek, Simina Boca

30 October 2017

Contents

0.1 Package overview

This package allows users to estimate the science-wise false discovery rate from Jager and Leek (2013), using an EM approach due to the presence of rounding and censoring. It also allows users to estimate the proportion of true null hypotheses in the presence of covariates, using a regression framework, as per Boca and Leek (2017).

The package is loaded using:

```
library(swfdr)
```

0.2 Estimating the science-wise false discovery rate

The science-wise false discovery rate (swfdr) is defined in Jager and Leek (2013) as the rate that published research results are false positives. It is based on using reported p-values reported in biomedical journals and assuming that, for the p-values below 0.05, those corresponding to false positives are distributed as U$(0, 0.05)$, while those corresponding to true positives are distributed as tBeta$(\alpha, \beta; 0.05)$, where α and β are unknown and tBeta is a Beta distribution truncated at 0.05. The estimation of the swfdr is complicated by truncation (e.g. reporting `p < 0.01`) and rounding (e.g. p-values are often rounded to two significant digits). An EM algorithm is used to estimate α, β, as well as the swfdr.

0.2.1 Example: Estimate the swfdr based on p-values from biomedical journals

We include a dataset containing 15,653 p-values from articles in 5 biomedical journals (American Journal of Epidemiology, BMJ, Jama, Lancet, New England Journal of Medicine), over 11 years (2000-2010). This is obtained from web-scraping, using the code at https://github.com/jtleek/swfdr/blob/master/getPvalues.R and is already loaded in the package.

```
colnames(journals_pVals)
## [1] "pvalue"          "pvalueTruncated" "pubmedID"        "year"
## [5] "journal"
```

A description of the variables in `journals_pVals`can be found on its help page. In particular, `pvalue` gives the p-value, `pvalueTruncated` is a flag for whether it is truncated, `year` is the year of publication, and `journal` the journal.

```
table(journals_pVals$year)
##
## 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010
## 1481 1215 1487 1504 1564 1455 1207 1339 1485 1414 1502
table(journals_pVals$journal)
##
## American Journal of Epidemiology                              BMJ
##                             1199                             1740
##                             JAMA                           Lancet
##                             4960                             3532
##  New England Journal of Medicine
##                             4222
```

0.2.2 The `calculateSwfdr` function

This function estimates the swfdr. It inputs the following parameters:

- `pValues` Numerical vector of p-values
- `truncated` Vector of 0s and 1s with indices corresponding to those in pValues; 1 indicates that the p-values is truncated, 0 that it is not truncated
- `rounded` Vector of 0s and 1s with indices corresponding to those in pValues; 1 indicates that the p-values is rounded, 0 that it is not rounded
- `pi0` Initial prior probability that a hypothesis is null (default is 0.5)
- `alpha` Initial value of parameter alpha from Beta(alpha, beta) true positive distribution (default is 1)
- `beta` Initial value of parameter beta from Beta(alpha, beta) true positive distribution (default is 50)
- `numEmIterations` The number of EM iterations (default is 100)

Given that it runs an EM algorithm, it is somewhat computationally intensive. We show an example of applying it to all the p-values from the abstracts for articles published in the American Journal of Epidemiology in 2015. First, we subset the `journals_pVals` and only consider the p-values below 0.05, as in Jager and Leek (2013):

```
journals_pVals1 <- dplyr::filter(journals_pVals,
                                 year == 2005,
                                 journal == "American Journal of Epidemiology",
                                 pvalue < 0.05)

dim(journals_pVals1)
## [1] 75  5
```

Next, we define vectors corresponding to the truncation status and the rouding status (defined as rounding to 2 significant digits) and use these vectors, along with the vector of p-values, and the number of EM iterations, as inputs to the `calculateSwfdr` function:

```
tt <- data.frame(journals_pVals1)[,2]
rr <- rep(0,length(tt))
rr[tt == 0] <- (data.frame(journals_pVals1)[tt==0,1] ==
                  round(data.frame(journals_pVals1)[tt==0,1],2))
pVals <- data.frame(journals_pVals1)[,1]
resSwfdr <- calculateSwfdr(pValues = pVals,
                           truncated = tt,
                           rounded = rr, numEmIterations=100)
names(resSwfdr)
## [1] "pi0"   "alpha" "beta"  "z"     "n0"    "n"
```

The following values are returned:

- `pi0` Final value of prior probability - estimated from EM - that a hypothesis is null, i.e. estimated swfdr
- `alpha` Final value of parameter alpha - estimated from EM - from Beta(alpha, beta) true positive distribution
- `beta` Final value of parameter beta - estimated from EM - from Beta(alpha, beta) true positive distribution

- `z` Vector of expected values of the indicator of whether the p-value is null or not - estimated from EM - for the non-rounded p-values (values of NA represent the rounded p-values)
- `n0` Expected number of rounded null p-values - estimated from EM - between certain cutpoints (0.005, 0.015, 0.025, 0.035, 0.045, 0.05)
- `n` Number of rounded p-values between certain cutpoints (0.005, 0.015, 0.025, 0.035, 0.045, 0.05)

0.2.3 Results from example dataset

For the example dataset we considered, the results are as follows:

```
resSwfdr
## $pi0
## [1] 0.0761073
##
## $alpha
##         a
## 0.1807012
##
## $beta
##        b
## 11.18069
##
## $z
##  [1] 3.118229e-03 4.732968e-04 2.049222e-02 3.118229e-03 3.118229e-03
##  [6] 7.176823e-05 3.118229e-03 3.118229e-03 1.162923e-02 5.497744e-03
## [11] 3.118229e-03 3.118229e-03 3.118229e-03 3.118229e-03 3.118229e-03
## [16] 3.118229e-03 4.732968e-04 4.732968e-04 3.118229e-03 2.049222e-02
## [21] 3.118229e-03 4.732968e-04 3.118229e-03 3.118229e-03 3.118229e-03
## [26] 3.118229e-03 3.118229e-03 3.118229e-03 3.118229e-03 3.118229e-03
## [31] 3.118229e-03 3.118229e-03 3.118229e-03 8.349926e-04 3.118229e-03
## [36]           NA 1.302711e-01 5.309398e-02           NA 1.029131e-01
## [41]           NA           NA           NA 3.019103e-02 5.309398e-02
## [46] 8.380096e-02 5.309398e-02 1.574988e-02           NA 1.478523e-01
## [51] 1.029131e-01 1.818187e-01 2.461694e-01 7.187669e-02 3.214756e-01
## [56]           NA 3.019103e-02           NA           NA 8.125652e-03
## [61] 4.200316e-02 4.611767e-03 1.430734e-02           NA           NA
## [66]           NA 1.212977e-01           NA 3.986255e-01 1.212977e-01
## [71] 5.309398e-02 7.389248e-02 1.302711e-01 7.389248e-02 1.716278e-02
##
## $n0
##
##     (0,0.005] (0.005,0.015] (0.015,0.025] (0.025,0.035] (0.035,0.045]
##     0.0000000     0.3140311     0.9757573     0.5499050     1.0464010
##  (0.045,0.05]
##     0.0000000
##
## $n
##
##     (0,0.005] (0.005,0.015] (0.015,0.025] (0.025,0.035] (0.035,0.045]
```

```
##            0            3            5            2            3
##  (0.045,0.05]
##            0
```

Thus, the estimated swfdr for papers published in American Journal of Epidemiology in 2005 is 0.076 i.e. 7.6% of the discoveries - defined as associations with p $<$ 0.05 - are expected to be false discoveries.

0.3 Estimating the proportion of true null hypothesis in the presence of covariates

As in Boca and Leek (2017), we denote by $\pi_0(x)$ the proportion of true null hypotheses as a function of a covariate x. This is estimated based on a list of p-values $p_1, \ldots, p_m$ corresponding to a set of null hypotheses, $H_{01}, \ldots, H_{0m}$, and a design matrix X. The design matrix considers relevant meta-data, which could be valuable for improving estimatong of the prior probability that a hypothesis is true or false.

0.3.1 Example: Adjust for sample size and allele frequency in BMI GWAS meta-analysis

We consider an example from the meta-analysis of data from a genome-wide association study (GWAS) for body mass index (BMI) from Locke et al. (2015). A subset of this data, corresponding to 50,000 single nucleotide polymorphisms (SNPs) is already loaded with the package.

```
head(BMI_GIANT_GWAS_sample)
## # A tibble: 6 x 9
##          SNP    A1    A2 Freq_MAF_Hapmap       b     se      p      N
##        <chr> <chr> <chr>           <dbl>   <dbl>  <dbl>  <dbl>  <dbl>
## 1 rs10510371     T     C          0.0250  0.0147 0.0152 0.3335 212965
## 2   rs918232     A     G          0.3417 -0.0034 0.0037 0.3581 236084
## 3  rs4816764     A     C          0.0083  0.0163 0.0131 0.2134 221771
## 4 rs17630047     A     G          0.1667  0.0004 0.0048 0.9336 236177
## 5  rs4609437     C     G          0.2500  0.0011 0.0042 0.7934 236028
## 6 rs11130746     G     A          0.2333 -0.0006 0.0042 0.8864 235634
## # ... with 1 more variables: Freq_MAF_Int_Hapmap <fctr>
dim(BMI_GIANT_GWAS_sample)
## [1] 50000     9
```

A description of the variables in `BMI_GIANT_GWAS_sample` can be found on its help page. In particular, `p` gives the p-values for the association between the SNPs and BMI; `N` gives the total sample size considered in the study of a particular SNP; and `Freq_MAF_Hapmap` gives the frequency of the minor (less frequent allele) for a particular SNP in Hapmap. The column `Freq_MAF_Int_Hapmap` provides 3 approximately equal intervals for the Hapmap MAFs:

```
table(BMI_GIANT_GWAS_sample$Freq_MAF_Int_Hapmap)
##
## [0.000,0.127) [0.127,0.302) [0.302,0.500]
##         16813         16887         16300
```

0.3.2 The `lm_pi0` function

This function estimates $\pi_0(x)$. It inputs the following parameters:

- `pValues` Numerical vector of p-values
- `lambda` Numerical vector of thresholds in $[0, 1)$ at which $\pi_0(x)$ is estimated. Default thresholds are $(0.05, 0.10, \ldots, 0.95)$.
- `X` Design matrix (one test per row, one variable per column). Do not include the intercept.
- `type` Type of regression, "logistic" or "linear." Default is logistic.
- `smooth.df` Number of degrees of freedom when estimating $\pi_0(x)$ by smoothing. Default is 3.
- `threshold` If `TRUE` (default), all estimates are thresholded at 0 and 1, if `FALSE`, none of them are.

To apply it to the BMI GWAS dataset, we first create the design matrix, using natural cubic splines with 5 degrees of freedom to model `N` and 3 discrete categories for the MAFs:

```
X <- model.matrix(~ splines::ns(N,5) + Freq_MAF_Int_Hapmap, data = BMI_GIANT_GWAS_sample)[,-1]
head(X)
##   splines::ns(N, 5)1 splines::ns(N, 5)2 splines::ns(N, 5)3
## 1       7.242962e-01          0.0000000       -0.096623916
## 2       6.214473e-05          0.9873057        0.010529926
## 3       8.482281e-01          0.0000000       -0.054593655
## 4       0.000000e+00          0.9847066        0.012746260
## 5       4.971578e-04          0.9884279        0.009232155
## 6       3.080761e-02          0.9658228        0.002791946
##   splines::ns(N, 5)4 splines::ns(N, 5)5 Freq_MAF_Int_Hapmap[0.127,0.302)
## 1        0.193411872      -0.0967879566                                 0
## 2        0.004207151      -0.0021049077                                 0
## 3        0.109279996      -0.0546863405                                 0
## 4        0.005093468      -0.0025463522                                 1
## 5        0.003688505      -0.0018457495                                 1
## 6        0.001127918      -0.0005644374                                 1
##   Freq_MAF_Int_Hapmap[0.302,0.500]
## 1                                0
## 2                                1
## 3                                0
## 4                                0
## 5                                0
## 6                                0
```

We then run the `lm_pi0` function:

```
pi0x <- lm_pi0(pValues=BMI_GIANT_GWAS_sample$p,
               X=X, smooth.df=3)
## At test #: 10000
## At test #: 20000
## At test #: 30000
## At test #: 40000
## At test #: 50000
names(pi0x)
## [1] "pi0"        "pi0.lambda" "lambda"     "pi0.smooth"
```

The following values are returned:

- `pi0` Numerical vector of smoothed estimate of $\pi_0(x)$. The length is the number of rows in X.
- `pi0.lambda` Numerical matrix of estimated $\pi_0(x)$ for each value of lambda. The number of columns is the number of tests, the number of rows is the length of lambda.
- `lambda` Vector of the values of `lambda` used in calculating `pi0.lambda`.
- `pi0.smooth` Matrix of fitted values from the smoother fit to the $\pi_0(x)$ estimates at each value of lambda (same number of rows and columns as `pi0.lambda`).

0.3.3 Results from BMI GWAS meta-analysis example

We first add the estimates of $\pi_0(x)$ for $\lambda = 0.8$, $\lambda = 0.9$, and the final smoothed value to the `BMI_GIANT_GWAS_sample` object:

```
BMI_GIANT_GWAS_sample$fitted0.8 <- pi0x$pi0.lambda[,round(pi0x$lambda,2)==0.8]
BMI_GIANT_GWAS_sample$fitted0.9 <- pi0x$pi0.lambda[,round(pi0x$lambda,2)==0.9]
BMI_GIANT_GWAS_sample$fitted.final.smooth <- pi0x$pi0
```

We next create a long data frame so that we can use the plotting tools in `ggplot2`:

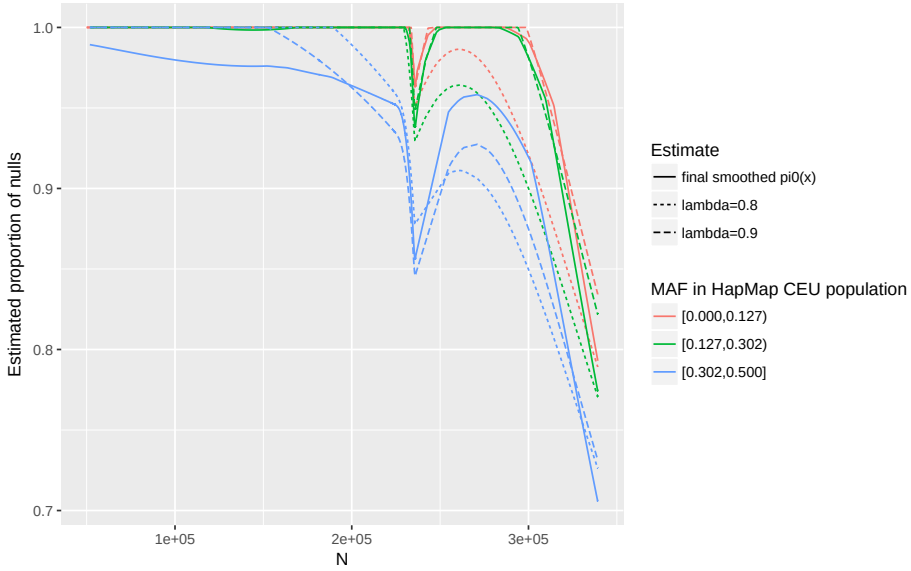
```
ldf <- reshape2::melt(BMI_GIANT_GWAS_sample,
                      id.vars=colnames(BMI_GIANT_GWAS_sample)[-grep("fitted",
                                                                    colnames(BMI_GIANT_GWAS_sample))],
                      value.name = "pi0",variable.name = "lambda")
ldf$lambda <- as.character(ldf$lambda)
ldf$lambda[ldf$lambda=="fitted0.8"] <- "lambda=0.8"
ldf$lambda[ldf$lambda=="fitted0.9"] <- "lambda=0.9"
ldf$lambda[ldf$lambda=="fitted.final.smooth"] <- "final smoothed pi0(x)"

head(ldf)
##          SNP A1 A2 Freq_MAF_Hapmap       b     se      p      N
## 1 rs10510371  T  C          0.0250  0.0147 0.0152 0.3335 212965
## 2   rs918232  A  G          0.3417 -0.0034 0.0037 0.3581 236084
## 3  rs4816764  A  C          0.0083  0.0163 0.0131 0.2134 221771
## 4 rs17630047  A  G          0.1667  0.0004 0.0048 0.9336 236177
## 5  rs4609437  C  G          0.2500  0.0011 0.0042 0.7934 236028
## 6 rs11130746  G  A          0.2333 -0.0006 0.0042 0.8864 235634
##   Freq_MAF_Int_Hapmap     lambda       pi0
## 1       [0.000,0.127) lambda=0.8 1.0000000
## 2       [0.302,0.500] lambda=0.8 0.8782535
## 3       [0.000,0.127) lambda=0.8 1.0000000
## 4       [0.127,0.302) lambda=0.8 0.9302119
## 5       [0.127,0.302) lambda=0.8 0.9298268
## 6       [0.127,0.302) lambda=0.8 0.9312780
```

The plot of the estimates of $\pi_0(x)$ against the sample size N, stratified by the MAF categories can thus be obtained:

```
library(ggplot2)
ggplot(ldf, aes(x=N, y=pi0))+
  geom_line(aes(col=Freq_MAF_Int_Hapmap, linetype=lambda)) +
```

```
        ylab("Estimated proportion of nulls") +
        guides(color=guide_legend(title="MAF in HapMap CEU population"),
               linetype=guide_legend(title="Estimate"))
```



References

Boca, Simina M, and Jeffrey T Leek. 2017. "A Regression Framework for the Proportion of True Null Hypotheses." *BioRxiv*. Cold Spring Harbor Labs Journals, 035675.

Jager, Leah R, and Jeffrey T Leek. 2013. "Empirical Estimates Suggest Most Published Medical Research Is True." *Biostatistics*.

Locke, Adam E, Bratati Kahali, Sonja I Berndt, Anne E Justice, Tune H Pers, Felix R Day, Corey Powell, et al. 2015. "Genetic Studies of Body Mass Index Yield New Insights for Obesity Biology." *Nature* 518 (7538). Nature Publishing Group: 197–206.