NPBin

Anthony Aylward

2018-03-20

We will analyze a subset of one of the sample dataset for illustration purposes.

```
library(npbin)
library(data.table)
#> data.table 1.10.4.3
#>   The fastest way to learn (by data.table authors): https://www.datacamp.com/courses/data
#>   Documentation: ?data.table, example(data.table) and browseVignettes("data.table")
#>   Release notes, videos and slides: http://r-datatable.com

minimum_coverage <- 5 # minimum total coverage allowed
n_cores <- 47 # the number of cores to be used, can ONLY be 1 if run on Windows.

dt <- atac
colnames(dt)
#>  [1] "chr"              "location"         "m"
#>  [4] "xm"               "winning.chip"     "motif"
#>  [7] "pval.mat.atSNP"   "pval.pat.atSNP"   "pval.rank.atSNP"
#> [10] "winnig.motif"     "potential_TP"     "potential_FP"

dt.ct <- data.table(dt)[m >= minimum_coverage, ]
```

for illustration purpose, keep only the 2000 points, remove this line will end up with analyzing the whole data set. It could be slow if only one core is used.

```
dt.ct <- dt.ct[1:2000, ]
dt.ct[, p_hat:=xm / m]
n <- nrow(dt.ct)
```

NPBin

```
n_breaks <- 11 # number of breaks
spline_order <- 4 # order of splines
```

```r
breaks <- seq(0, 1, length.out = n_breaks)
pi_init <- hist(
  dt.ct[, p_hat],
  breaks = seq(0, 1, length.out = n_breaks + spline_order - 3),
  plot = FALSE
)[["density"]] # initialized the weights using the histogram of p_hat
```

estimate the overall model

```r
overall_model_estimate <- emBinBspl(
  dt.ct[, xm],
  dt.ct[, m],
  breaks = breaks,
  k = spline_order,
  pi.init = pi_init,
  ncores = n_cores,
  err.max = 1e-3,
  iter.max = 200
)
```

estimate the null model

```r
null_model_estimate <- estNull(
  dt.ct[, xm],
  dt.ct[, m],
  overall_model_estimate,
  init = NULL,
  iter.max = 200,
  ncores = n_cores,
  ub = rep(log(1e4), 2),
  err.max = 1e-4
)
dt.ct[,
  fnp := null_model_estimate[["f"]]
][,
  f0np := null_model_estimate[["f0"]]
][,
  locfdrnp := null_model_estimate[["locfdr"]]
][,
  fdrnp := locfdr2FDR(locfdrnp)
][,
  ranknp := rank(locfdrnp, ties.method = "max")
]
names(null_model_estimate)
#>  [1] "pi"              "post"              "bspl"
#>  [4] "dmtx"             "f"                "ll.all"
#>  [7] "err.all"          "convergence"      "controls"
#> [10] "coef.null"        "pi0"              "f0"
#> [13] "locfdr"           "convergence.null"
```

```
null_model_estimate$coef.null
#> $shape1
#> [1] 13.41442
#>
#> $shape2
#> [1] 12.97
#>
#> $pi0
#> [1] 0.8604282
```

Empirical Bayes test using p__hat

```
pct0 <- 0.45
empirical_bayes_beta_hat <- ebBeta(
  dt.ct[, xm],
  dt.ct[, m],
  dt.ct[, p_hat],
  breaks = breaks,
  k = spline_order,
  pi.init = pi_init,
  pct0 = pct0,
  init = NULL,
  iter.max = 200,
  err.max = 1e-4,
  ncores = n_cores
)
dt.ct[,
  fhat := empirical_bayes_beta_hat[["f"]]
][,
  f0hat := empirical_bayes_beta_hat[["f0"]]
][,
  locfdrhat := empirical_bayes_beta_hat[["locfdr"]]
][,
  fdrhat := locfdr2FDR(locfdrhat)
][,
  rankhat := rank(locfdrhat, ties.method = "max")
]
names(empirical_bayes_beta_hat)
#>  [1] "pi"          "post"        "bspl"        "dmtx"        "binmtx"
#>  [6] "f"           "ll.all"      "err.all"     "convergence" "controls"
#> [11] "coef.null"   "f0"           "pi0"          "locfdr"       "locfdrnorm"
#> [16] "null"
```

null parameters of EBE

```
empirical_bayes_beta_hat[["coef.null"]]
#> $shape1
#> [1] 3.427515
#>
#> $shape2
#> [1] 3.341993
#>
#> $pi0
#> [1] 0.9883402
```

Binomial test

```
p_binomial <- sapply(
  1:n,
  function(y) binom.test(dt.ct[y, xm], dt.ct[y, m])[["p.value"]]
)
dt.ct[,
  pvbin := p_binomial
][,
  fdrbin := p.adjust(pvbin, method = "BH")
][,
  rankbin := rank(pvbin, ties.method = "max")
]
```

Evaluate the results using motifs

number of potential TP defined based on motif

```
dt.ct[, sum(potential_TP)]
#> [1] 42
```

number of potential FP defined based on motif

```
dt.ct[, sum(potential_FP)]
#> [1] 2
```

find the number of TP and FP in top ranked SNPs

```
dt.ct[,
  tpnp := rank2nhit(ranknp ,potential_TP)
][,
  fpnp := rank2nhit(ranknp, potential_FP)
]
#> Error in rank2nhit(ranknp, potential_TP): could not find function "rank2nhit"
dt.ct[,
  tp_hat := rank2nhit(rankhat, potential_TP)
][,
  fp_hat := rank2nhit(rankhat, potential_FP)
]
#> Error in rank2nhit(rankhat, potential_TP): could not find function "rank2nhit"
dt.ct[,
  tpbin := rank2nhit(rankbin, potential_TP)
][,
  fpbin := rank2nhit(rankbin, potential_FP)
]
#> Error in rank2nhit(rankbin, potential_TP): could not find function "rank2nhit"
```

plot the accuracy measure as in the main paper. We presented a zoom-in version in the main paper to the top 20%, because usually there are not many ALI SNPs. Note that the default of the demo only select a subset of the data for illustration purposes. Thus the figure may not an exact replica of the one in the paper. To reproduce the results in the paper, please use the whole dataset

```
cbfpalette <- c(
  "#D55E00",
  "#0072B2",
  "#CC79A7",
  "#009E73",
  "#E69F00",
  "#56B4E9",
  "#F0E442"
)
plotidac <- c(' NPB','EBE','Binom')
```