A Vignette for DeMixT

Zeya Wang and Fan Gao

2019-02-16

Contents

1. Introduction

Transcriptomic deconvolution in cancer and other heterogeneous tissues remains challenging. Available methods lack the ability to estimate both component-specific proportions and expression profiles for individual samples. We develop a three-component deconvolution model, DeMixT, for expression data from a mixture of cancerous tissues, infiltrating immune cells and tumor microenvironment. DeMixT is a software package that performs deconvolution on transcriptome data from a mixture of two or three components.

DeMixT is a frequentist-based method and fast in yielding accurate estimates of cell proportions and compartment-specific expression profiles for two-component and three-component deconvolution problem. Our method promises to provide deeper insight into cancer biomarkers and assist in the development of novel prognostic markers and therapeutic strategies.

The function DeMixT is designed to finish the whole pipeline of deconvolution for two or three components. DeMixT.S1 function is designed to estimate the proportions of all mixed samples for each mixing component. DeMixT.S2 function is designed to estimate the component-specific deconvolved expressions of individual mixed samples for a given set of genes.

2. Installation

2.1 Source file

DeMixT source files are compatible with Windows, Linux and macOS.

DeMixT_0.2 is the latest version, which is for a computer that has OpenMP. DeMixT_0.2.1 is for computers that do not have OpenMP. To install DeMixT_0.2.1, start R and enter:

```
#devtools::install_github("wwylab/DeMixTallmaterials/DeMixT_0.2.1")
```

2.2 More information

For a computer that does not have OpenMP or the older versions of DeMixT, please visit https://github.com/wwylab/DeMixTallmaterials. You can also download the installation files directly from the website:

DeMixT_0.2: http://bioinformatics.mdanderson.org/Software/DeMixT/DeMixT_0.2.tar.gz

DeMixT_0.2.1: http://bioinformatics.mdanderson.org/Software/DeMixT/DeMixT_0.2.1.tar.gz

For more information, please visit: http://bioinformatics.mdanderson.org/main/DeMixT

2.3 Functions

The following table shows the functions included in DeMixT.

Table Header	Second Header
DeMixT	Deconvolution of heterogeneous tumor samples with two or three components using expression data from RNAseq or microarray platforms
DeMixT.S1	Estimates the proportions of mixed samples for each mixing component
DeMixT.S2	Deconvolves expressions of each individual sample for unknown component
Optimum.KernelC	Kernel function to call the C function used for parameter estimation in DeMixT

3. Session Info

```
sessionInfo(package = "DeMixT")
```

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Mojave 10.14.3
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## character(0)
##
## other attached packages:
## [1] DeMixT_0.3.1
##
## loaded via a namespace (and not attached):
```

```
##  [1] Rcpp_0.12.18      rstudioapi_0.7    knitr_1.20
##  [4] magrittr_1.5      usethis_1.4.0     devtools_2.0.1
##  [7] grDevices_3.5.2   pkgload_1.0.1     R6_2.2.2
## [10] rlang_0.3.0.1     stringr_1.3.1     tools_3.5.2
## [13] utils_3.5.2       pkgbuild_1.0.2    sessioninfo_1.1.0
## [16] cli_1.0.0         withr_2.1.2       htmltools_0.3.6
## [19] stats_3.5.2       remotes_2.0.1     datasets_3.5.2
## [22] yaml_2.2.0        assertthat_0.2.0  digest_0.6.16
## [25] rprojroot_1.3-2   base_3.5.2        crayon_1.3.4
## [28] processx_3.2.0    callr_3.0.0       base64enc_0.1-3
## [31] graphics_3.5.2    fs_1.2.6          ps_1.1.0
## [34] evaluate_0.11     memoise_1.1.0     glue_1.3.0
## [37] rmarkdown_1.10    stringi_1.2.4     compiler_3.5.2
## [40] methods_3.5.2     desc_1.2.0        backports_1.1.2
## [43] prettyunits_1.0.2
```

4. Methods

4.1 Model

Let Y_{ig} be the observed expression levels of the raw measured data from clinically derived malignant tumor samples for gene $g, g = 1, \cdots, G$ and sample $i, i = 1, \cdots, S$. G denotes the total number of probes/genes and S denotes the number of samples. The observed expression levels for solid tumors can be modeled as a linear combination of raw expression levels from three components:

$$Y_{ig} = \pi_{1,i} N_{1,ig} + \pi_{2,i} N_{2,ig} + (1 - \pi_{1,i} - \pi_{2,i}) T_{ig}$$

Here $N_{1,ig}$, $N_{2,ig}$ and T_{ig} are the unobserved raw expression levels from each of the three components. We call the two components for which we require reference samples the N_1-component and the N_2-component. We call the unknown component the T-component. We let $\pi_{1,i}$ denote the proportion of the N_1-component, $\pi_{2,i}$ denote the proportion of the N_2-component, and $1 - \pi_{1,i} - \pi_{2,i}$ denote the proportion of the T-component. We assume that the mixing proportions of one specific sample remain the same across all genes.

Our model allows for one component to be unknown, and therefore does not require reference profiles from all components. A set of samples for $N_{1,ig}$ and $N_{2,ig}$, respectively, needs to be provided as input data. This three-component deconvolution model is applicable to the linear combination of any three components in any type of material. It can also be simplified to a two-component model, assuming there is just one N-component. For application in this paper, we consider tumor (T), stromal (N_1) and immune components (N_2) in an admixed sample (Y).

Following the convention that $\log_2$-transformed microarray gene expression data follow a normal distribution, we assume that the raw measures $N_{1,ig} \sim LN(\mu_{N_1 g}, \sigma^2_{N_1 g})$, $N_{2,ig} \sim LN(\mu_{N_2 g}, \sigma^2_{N_2 g})$ and $T_{ig} \sim LN(\mu_{Tg}, \sigma^2_{Tg})$, where LN denotes a $\log_2$-normal distribution and $\sigma^2_{N_1 g}$,$\sigma^2_{N_2 g}$,σ^2_{Tg} reflect the variations under $\log_2$-transformed data. Consequently, our model can be expressed as the convolution of the density function for three $\log_2$-normal distributions. Because there is no closed form of this convolution, we use numerical integration to evaluate the complete likelihood function (see the full likelihood in the Supplementary Materials).

4.2 The DeMixT algorithm for deconvolution

DeMixT estimates all distribution parameters and cellular proportions and reconstitutes the expression profiles for all three components for each gene and each sample. The estimation procedure (summarized in Figure 1b) has two main steps as follows.

1. Obtain a set of parameters $\{\pi_{1,i}, \pi_{2,i}\}_{i=1}^{S}$, $\{\mu_T, \sigma_T\}_{g=1}^{G}$ to maximize the complete likelihood function, for which $\{\mu_{N_{1,g}}, \sigma_{N_{1,g}}, \mu_{N_{2,g}}, \sigma_{N_{2,g}}\}_{g=1}^{G}$ were already estimated from the available unmatched samples of the N_1 and N_2 component tissues. (See further details in our paper.)
2. Reconstitute the expression profiles by searching each set of $\{n_{1,ig}, n_{2,ig}\}$ that maximizes the joint density of $N_{1,ig}$, $N_{2,ig}$ and T_{ig}. The value of t_{ig} is solved as $y_{ig} - \hat{\pi}_{1,i} n_{1,ig} - \hat{\pi}_{2,i} n_{2,ig}$.

These two steps can be separately implemented using the function DeMixT.S1 and DeMixT.S2, which are combined in the function DeMixT.

5. Examples

5.1 Simulated two-component data

```
library(DeMixT)
```

```
## Loading required package: parallel
## Loading required package: SummarizedExperiment
## Loading required package: GenomicRanges
## Loading required package: stats4
## Loading required package: BiocGenerics
##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind,
##     colMeans, colnames, colSums, dirname, do.call, duplicated,
##     eval, evalq, Filter, Find, get, grep, grepl, intersect,
##     is.unsorted, lapply, lengths, Map, mapply, match, mget, order,
##     paste, pmax, pmax.int, pmin, pmin.int, Position, rank, rbind,
##     Reduce, rowMeans, rownames, rowSums, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which, which.max,
##     which.min
## Loading required package: S4Vectors
##
## Attaching package: 'S4Vectors'
## The following object is masked from 'package:base':
##
##     expand.grid
```

```
## Loading required package: IRanges
## Loading required package: GenomeInfoDb
## Loading required package: Biobase
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
## Loading required package: DelayedArray
## Loading required package: matrixStats
##
## Attaching package: 'matrixStats'
## The following objects are masked from 'package:Biobase':
##
##     anyMissing, rowMedians
## Loading required package: BiocParallel
##
## Attaching package: 'DelayedArray'
## The following objects are masked from 'package:matrixStats':
##
##     colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges
## The following objects are masked from 'package:base':
##
##     aperm, apply
data(test.data.y)
## Warning in data(test.data.y): data set 'test.data.y' not found
data(test.data.comp1)
## Warning in data(test.data.comp1): data set 'test.data.comp1' not found
res <- DeMixT(data.Y = test.data1.y, data.comp1 = test.data1.comp1, if.filter = FALSE, output.more.info
res$pi
##             1         2         3         4         5         6         7
## pi1 0.1074078 0.2196809 0.3047277 0.3523041 0.4248352 0.4888389 0.6179312
##            8         9        10
## pi1 0.608743 0.6632821 0.7670111
head(res$ExprT, 3)
##          1         2         3        4        5        6        7
## 1 96.83726  83.56023  80.82836 80.72725 84.47915 83.32202 87.44895
## 2 77.09617 112.44931 136.70835 85.97539 81.43849 68.38339 87.89490
## 3 38.68176  35.21445  38.30982 37.65722 32.67902 73.87686 83.91727
##           8        9       10
## 1  87.23697 85.60188 85.84117
## 2 101.27924 98.38278 72.67417
## 3  47.83651 47.25389 56.85126
```

```r
head(res$ExprN1, 3)
```

```
##           1         2         3         4         5         6         7
## 1 129.09181 115.73951 105.97676 103.13745 116.22402 108.05379 172.66879
## 2  90.39363  96.06401 102.69785  92.02293  89.36471  77.15229  96.40235
## 3  53.11342  52.57825  52.81754  52.58788  50.68354  57.91523  62.84381
##           8         9        10
## 1 162.01438 130.74263 153.04748
## 2 116.43629 120.28931  65.71293
## 3  56.23678  56.85083  68.33056
```

```r
head(res$Mu, 3)
```

```
##       MuN1      MuT
## 1 6.940477 6.420368
## 2 6.582707 6.493136
## 3 5.767947 5.531644
```

```r
head(res$Sigma, 3)
```

```
##     SigmaN1    SigmaT
## 1 0.2675802 0.1339120
## 2 0.3201497 0.3714339
## 3 0.2231435 0.5109864
```

```r
res$pi.iter
```

```
## , , 1
##
##             [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
##  [1,] 0.09043013 0.2139893 0.2937994 0.3356124 0.4489045 0.5187335
##  [2,] 0.09198396 0.2256721 0.3003950 0.3433463 0.4443040 0.5215801
##  [3,] 0.09218502 0.2259228 0.3000308 0.3437418 0.4438475 0.5214382
##  [4,] 0.10617830 0.2252988 0.3015530 0.3608652 0.4454556 0.5067351
##  [5,] 0.10977845 0.2275840 0.3089533 0.3611274 0.4456627 0.5040962
##  [6,] 0.11123518 0.2424661 0.3081845 0.3580924 0.4228940 0.5039239
##  [7,] 0.11172841 0.2424911 0.3064393 0.3525787 0.4233200 0.4874916
##  [8,] 0.11071479 0.2216530 0.3054770 0.3513610 0.4243426 0.4873702
##  [9,] 0.10976194 0.2201011 0.3011756 0.3494824 0.4265594 0.4883885
## [10,] 0.10864671 0.2200628 0.3032327 0.3513427 0.4246896 0.4886372
##             [,7]      [,8]      [,9]     [,10]
##  [1,] 0.6544423 0.7281499 0.8230019 0.8744931
##  [2,] 0.6532641 0.7273786 0.8232727 0.8754047
##  [3,] 0.6528566 0.7269844 0.8231059 0.8336147
##  [4,] 0.6526147 0.7277983 0.7440691 0.8335681
##  [5,] 0.5813924 0.6121490 0.7436529 0.7410122
##  [6,] 0.5811898 0.6119788 0.7448583 0.7415068
##  [7,] 0.5804985 0.6122466 0.7454209 0.7417283
##  [8,] 0.5806471 0.6130025 0.7456191 0.7651868
##  [9,] 0.6154044 0.6110913 0.7453916 0.7661025
## [10,] 0.6176156 0.6089009 0.7454029 0.7668798
```

```r
res$gene.name
```

```
##   [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "11"
##  [12] "12"  "13"  "14"  "15"  "16"  "17"  "18"  "19"  "20"  "21"  "22"
##  [23] "23"  "24"  "25"  "26"  "27"  "28"  "29"  "30"  "31"  "32"  "33"
```

```
##  [34] "34"  "35"  "36"  "37"  "38"  "39"  "40"  "41"  "42"  "43"  "44"
##  [45] "45"  "46"  "47"  "48"  "49"  "50"  "51"  "52"  "53"  "54"  "55"
##  [56] "56"  "57"  "58"  "59"  "60"  "61"  "62"  "63"  "64"  "65"  "66"
##  [67] "67"  "68"  "69"  "70"  "71"  "72"  "73"  "74"  "75"  "76"  "77"
##  [78] "78"  "79"  "80"  "81"  "82"  "83"  "84"  "85"  "86"  "87"  "88"
##  [89] "89"  "90"  "91"  "92"  "93"  "94"  "95"  "96"  "97"  "98"  "99"
## [100] "100" "101" "102" "103" "104" "105" "106" "107" "108" "109" "110"
## [111] "111" "112" "113" "114" "115" "116" "117" "118" "119" "120"
```

5.2 Simulated three-component data

```
# data(test.data2.y)
# data(test.data2.comp1)
# data(test.data2.comp2)
# res <- DeMixT(data.Y = test.data2.y, data.comp1 = test.data2.comp1, data.comp2 = test.data2.comp2, if
```

5.3 Laser-capture microdissection (LCM) prostate cancer FFPE microarray dataset

This dataset was generated at the Dana Farber Cancer Institute (GSE97284). Radical prostatectomy specimens were annotated in detail by pathologists, and regions of interest were identified that corresponded to benign epithelium, prostatic intraepithelial neoplasia (abnormal tissue that is possibly precancerous), and tumor, each with its surrounding stroma. FFPE samples are known to generate overall lower quality expression data than those from fresh frozen samples. We observed a small proportion of probesets that presented large differences in mean expression levels between the dissected tissues: tumor (T) and stroma (N) in this dataset. Only 53 probesets presented a mean difference ($|\bar{T} - \bar{N}|) > 1$, as compared to 10,397 probesets in GSE19830. We therefore chose the top 80 genes with the largest mean differences and ran DeMixT under two settings: tumor unknown and stroma unknown. DeMixT is able to obtain concordant estimates of the tumor proportions when the proportion of the stromal component was unknown and when the proportion of tumor tissue was unknown and also tended to provide accurate component-specific mean expression levels.

```
# library(DeMixT)
# data <- as.matrix(read.table("Data/input.lcm.txt", header = FALSE))
# normal <- data[, 1:25]
# adm <- data[, 26:48]
# tumor <- data[, 49:73]
#
# testr.TA <- DeMixT(data.Y = 2^adm, data.comp1 = 2^tumor,
#                    niter = 20, nbin = 60, if.filter = FALSE, tol = 10^-6)
# testr.SA <- DeMixT(data.Y = 2^adm, data.comp1 = 2^normal,
#                    niter = 20, nbin = 60, if.filter = FALSE, tol = 10^-6)
```

```
# # plot A
# dt_purT <- 1- as.numeric(testr.SA$pi)
# dt_purS <- 1- as.numeric(testr.TA$pi)
# plot(1 - dt_purS, dt_purT,
#      col = "blue", pch = 1, xlim = c(0, 1), ylim = c(0, 1),
#      xlab = expression(1 - hat(pi)[S]), ylab = expression(hat(pi)[T]))
# abline(0, 1, col = "red", lwd = 2)
#
# # Plot B - Mean expressions for Tumor
# OB_St <- log2(read.table("Data/lcm_normal.txt", header = F))
```

```
# OB_Tu <- log2(read.table("Data/lcm_tumor.txt", header = F))
# DT_Tu_mu <- as.numeric(testr.SA$Mu[, 1])
# DT_St_mu <- as.numeric(testr.TA$Mu[, 1])
# DT_Tu_sg <- as.numeric(testr.SA$Sigma[, 1])
# DT_St_sg <- as.numeric(testr.TA$Sigma[, 1])
# OB_St_m <- apply(OB_St, 1, mean)
# OB_Tu_m <- apply(OB_Tu, 1, mean)
# # filter out genes with large estimated standard deviations
# condSt <- (DT_St_sg < 0.99)
# condTu <- (DT_Tu_sg < 0.99)
# DT_Tu_m <- as.numeric(apply(log2(testr.SA$ExprT), 1, mean))
# DT_St_m <- as.numeric(apply(log2(testr.TA$ExprT), 1, mean))
# OB_St_m <- OB_St_m[condSt]
# OB_Tu_m <- OB_Tu_m[condTu]
# DT_St_m <- DT_St_m[condSt]
# DT_Tu_m <- DT_Tu_m[condTu]
#
# # TPlot B - Mean expressions for Tumor
# smoothScatter((DT_Tu_m + OB_Tu_m) / 2, DT_Tu_m - OB_Tu_m,
#               ylab = "Estimate - Truth", xlab = "(Estimate + Truth)/2",
#               xlim = c(2,16), ylim = c(-1.2,1.2),
#               main = "Mean expressions for Tumor",
#               pch = 1, nrpoints = 0, col = 'yellow',
#               colramp=colorRampPalette(c("white","yellow","yellow1","orange","orange1")))
# tmp01 <- lowess((DT_Tu_m - OB_Tu_m) ~ ((DT_Tu_m + OB_Tu_m) / 2))
# lines(tmp01$x, tmp01$y, col="blue", lwd = 5)
# abline(h = 0, col = 'red', lty = 2)
#
# # Plot C - Mean expressions for Stroma
# smoothScatter((DT_St_m + OB_St_m) / 2, DT_St_m - OB_St_m,
#               ylab = "Estimate - Truth", xlab = "(Estimate + Truth)/2",
#               xlim = c(2,16), ylim = c(-1.2,1.2),
#               main = "Mean expressions for Stroma", pch = 1, nrpoints = 0,
#               col = 'yellow',
#               colramp=colorRampPalette(c("white","yellow","yellow1","orange","orange1")))
# tmp01 <- lowess((DT_St_m - OB_St_m) ~ ((DT_St_m + OB_St_m) / 2))
# lines(tmp01$x, tmp01$y, col="blue", lwd = 5)
# abline(h = 0, col = 'red', lty = 2)
```