Tutorial for humanFC package

Weizhuang Zhou

February 7, 2018

Introduction

This tutorial introduces the functions in the R package *humanFC* and provides a walkthrough using humanFC on a sample dataset. We will use the processed data from GSE71370 (RMA and bias correction); both expression set file and the meta data can be found at https://simtk.org/projects/humanFC

Before proceeding, please ensure that Biobase has been installed. If not, you may do so by starting R and entering:

```
source("http://bioconductor.org/biocLite.R")
biocLite("Biobase")
biocLite("affy")
```

The humanFC package can be installed using:

```
install.packages("humanFC_xxx.tar.gz")
```

where xxx is the version number of the package you downloaded. If installing within RStudio, you may need to use the following command instead:

```
install.packages("humanFC_xxx.tar.gz", type = "source", repos=NULL)
```

Additionally, if you would like to follow along the tutorial, check that the meta data file (GSE71370_meta.txt) and the expression R Data file (eset_rmaBC.RData) are in the same directory as the terminal.

Objects in the humanFC package

The main object is the projection matrix (S), which can be accessed by a user once the package is installed.

```
library(humanFC)
data(S)
dim(S)  # Columns are the FCs
```

```
## [1] 20089   139
```

```
apply(S,2,mean)  # Mean of each FC is 0
```

```
##   [1]  1.891165e-17  2.656781e-17 -1.672828e-17  4.892836e-17  3.808314e-17
##   [6] -6.529389e-17 -2.256804e-17 -2.540340e-17  6.401386e-18  3.060028e-17
##  [11]  2.360575e-17  1.201205e-17  4.651816e-17 -1.394745e-17 -1.286272e-17
##  [16]  3.343960e-17 -1.441568e-18 -2.006958e-17  3.448517e-17 -1.406244e-17
##  [21] -6.560811e-19  1.609065e-17  1.420584e-17  3.570358e-17  3.987854e-17
##  [26]  2.508337e-17  2.742578e-17  2.070102e-17 -6.311982e-17 -1.860884e-19
##  [31] -3.289284e-17 -1.681553e-17 -4.258142e-18 -8.763261e-18  2.851049e-17
##  [36] -2.118127e-17 -4.789162e-17 -1.124131e-17 -3.876346e-17  3.782225e-17
##  [41] -4.864844e-17  8.225993e-18  1.397977e-17  4.686462e-17  1.229051e-17
##  [46] -1.284725e-17 -5.354545e-17  2.314495e-17 -1.891468e-17 -3.508392e-17
##  [51]  5.568705e-17 -2.074227e-18  1.114843e-17  6.714279e-18 -5.507358e-18
##  [56] -1.798171e-17 -1.217882e-17  1.203002e-17 -7.483754e-17  3.995018e-17
##  [61]  2.151399e-17  1.124258e-17  3.922699e-17 -3.762921e-17 -4.804577e-18
```

```
##  [66] -1.457946e-17 -3.243305e-18 -1.349315e-17 -3.412171e-17 -6.310075e-17
##  [71] -1.924092e-17 -3.143453e-17  1.525890e-17 -1.384611e-17 -4.984034e-18
##  [76]  4.677832e-17  2.817147e-17  2.221137e-17  1.483111e-17 -2.420440e-17
##  [81] -1.142797e-16 -3.035287e-17 -1.244638e-17  1.935055e-17  4.678693e-17
##  [86]  9.943556e-18  1.454132e-17 -1.032350e-17  1.434158e-17 -6.456791e-18
##  [91]  8.990286e-19  1.212113e-17  3.809332e-17  3.907460e-17  1.809396e-17
##  [96] -5.152748e-18 -1.198712e-17  5.592944e-17  5.892375e-17 -2.128782e-17
## [101] -9.104577e-18 -2.868740e-17  8.301430e-18 -3.023275e-17 -9.677313e-18
## [106]  4.474309e-17 -2.556249e-17  2.220890e-17 -4.469121e-17 -3.613756e-17
## [111]  1.670583e-18  6.922574e-17  1.442454e-17 -5.186452e-18  4.568137e-17
## [116] -2.428370e-17 -6.949473e-18 -2.417665e-17  2.006772e-17  2.631030e-17
## [121] -1.447010e-17 -4.351366e-17 -1.498681e-18  2.830813e-17 -7.804495e-18
## [126] -9.036080e-18  2.968694e-17  1.085866e-17 -4.225217e-17  4.356427e-17
## [131] -1.453122e-17  2.163927e-17  1.694033e-17 -1.484984e-17 -2.280267e-17
## [136] -3.600049e-17  2.471850e-17  2.144528e-18  1.751812e-17
```

```
apply(S,2,sd)    # SD of each FC is 1
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [71] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [106] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Other objects such as the tissue fingerprint libraries and representative compendium are stored internally within the package and cannot be directly accessed by an end user.

Load expression data from GSE71370

We will now use GSE71370 data to demonstrate the use of the *humanFC* R package. The data has already been RMA normalized and bias corrected.

```
library(affy)
load("eset_rmaBC.RData")
meta <- read.table("GSE71370_meta.txt", header=T, as.is=T, sep="\t")
```

The meta data has four columns: the GSM id, the tissue type (peripheral blood vs synovial fluid), the condition (patient vs healthy control) and the Patient ID. It is convenient to abbreviate the 3 groups as in the paper: Healthy Control Peripheral Blood (HCPBM), Rheumatoid Arthritis Peripheral Blood (RAPBM) and Rheumatoid Arthritis Synovial Fluid (RASFM).

Projection of expression set into FC space

The *mapToFC* function performs the mapping of a HG-U133 Plus 2.0 array to the FC space. Typing *?mapToFC* or *help(mapToFC)* will launch the documentation for the function.

```
#help(mapToFC)
A <- mapToFC(exprs(eset.rma.bc), IsUnitBasis = FALSE)
#dim(A)  # 139 FCs by 26 samples
```

We will reorder the samples to match the ordering in the meta data.

```
gsms <- unname(sapply(colnames(A), function(x) strsplit(x, split="_")[[1]][1]))
A <- A[, match(gsms, meta[,1])]
colnames(A) <- meta[,1]
rownames(A) <- 1:139  # FCs
```

Finding DE FCs

We will consider the three possible group pairs: RAPBM-RASFM, RAPBM-HCPBM and RASFM-HCPBM

```
# Get indicies of groups
RAPBM <- which(meta$Tissue == "CD14+ monocytes from peripheral blood" & meta$Cond == "patient")
RASFM <- which(meta$Tissue == "CD14+ monocytes from synovial fluid" & meta$Cond == "patient")
HCPBM <- which(meta$Cond == "healthy control")

# Perform t.test, apply BH correction.
# Welsh df modification is implemented by default in t.test
runTtest <- function(A, grp1.ind, grp2.ind, is.pair) {
    tests <- list()
    pvals <- rep(0, nrow(A))
    for (fc in 1:nrow(A)) {
        t <- t.test(A[fc,grp1.ind], A[fc,grp2.ind], paired=is.pair)
        tests[[fc]] <- t
        pvals[fc] <- t$p.value
    }
    pvals <- p.adjust(pvals, method="BH")
    return(list(tests=tests, pvals=pvals))
}


# Perform paired t-test between RA SFM and RA PBM
RAPBMvsRASFM <- runTtest(A, RAPBM, RASFM, TRUE)

# Perform unpaired t-test between RA PBM and HC PBM
RAPBMvsHCPBM <- runTtest(A, RAPBM, HCPBM, FALSE)

# Perform unpaired t-test between RA SFM and HC PBM
RASFMvsHCPBM <- runTtest(A, RASFM, HCPBM, FALSE)
```

We are interested in the FCs that are differentially expressed at BH p-value < 0.05.

```
pThresh <- 0.05
which(RAPBMvsRASFM$pvals <= pThresh)  # 72 DE FCs between RAPBM and RASFM
```

```
##  [1]   4   5   6   7   9  12  13  19  23  25  26  29  31  32  34  38  39
## [18]  42  44  47  48  49  50  51  54  56  58  59  62  65  66  71  72  75
## [35]  77  78  79  80  81  84  85  86  87  88  92  95  96  98  99 100 102
## [52] 105 107 108 109 110 111 113 114 115 116 120 122 123 125 126 127 131
## [69] 132 137 138 139
```

```
which(RAPBMvsHCPBM$pvals <= pThresh)  # No DE FCs
```

```
## integer(0)
```

```
which(RASFMvsHCPBM$pvals <= pThresh)  # 89 DE FCs between RASFM and HCPBM
```

```
##  [1]   2   3   4   5   6   7   8   9  10  12  13  17  19  23  24  26  29
## [18]  30  31  32  33  34  35  38  39  40  41  42  44  47  49  50  52  53
## [35]  54  55  56  57  58  59  60  62  63  65  66  70  72  74  78  79  80
## [52]  84  85  86  88  89  90  92  94  95  96  98  99 101 102 103 104 107
## [69] 108 109 110 111 114 115 116 118 120 122 123 124 125 126 127 130 131
## [86] 132 137 138 139
```

GO annotations of FC

We find that FC 4 is a common DE FC to both RAPBM vs RASFM and RASFM vs HCPBM. The following command will pull up the GO codes related to FC 4.

```
getGO(4)  # GO codes for FC 4
```

```
##  [1] "GO:0007067" "GO:0000278" "GO:0000082" "GO:0007264" "GO:0034080"
##  [6] "GO:0006271" "GO:0006270" "GO:0000086" "GO:0032201" "GO:0000722"
## [11] "GO:0007059" "GO:0006260" "GO:0000083" "GO:0000079" "GO:0007076"
## [16] "GO:0007080" "GO:0045003" "GO:0008283" "GO:0000281" "GO:0000724"
## [21] "GO:0006284" "GO:0051726" "GO:0071897" "GO:0051988" "GO:0032508"
## [26] "GO:0032467" "GO:0000076" "GO:0010032" "GO:0000727" "GO:0006297"
## [31] "GO:0007018" "GO:0034501" "GO:0000070" "GO:0042769" "GO:0007088"
## [36] "GO:0007094" "GO:0042276" "GO:0051382" "GO:0007052" "GO:0051256"
## [41] "GO:0006298" "GO:0031577" "GO:0090307" "GO:0051321" "GO:0046602"
## [46] "GO:0085020" "GO:0001556" "GO:0007019" "GO:0070987" "GO:0006268"
## [51] "GO:0043137" "GO:0019886" "GO:0033683" "GO:0031110" "GO:0051488"
## [56] "GO:0060236" "GO:0048478" "GO:0031145"
```

Inferring tissue origin of a sample

We can also use the built-in tissue fingerprint library (based on GSE3526 and GSE7307) to annotate the sample with the most similar tissue type. For instance, the first sample (*A[,1]*) of our dataset is sample containing monocytes from peripheral blood. We can use it to find the top 5 closest tissue types as follows:

```
tissueType(A[,1], retain = 5)  # Using Pearson correlation (default)
```

```
##       TissueType medianCor  N
## 36   bone_marrow 0.8042006 10
## 55  thymus_gland 0.7748259  2
## 45        spleen 0.7608344  9
## 56        tonsil 0.7530917  6
## 102  lymph_nodes 0.7391832  8
```

```
tissueType(A[,1], method = "spearman", retain = 5)
```

```
##       TissueType medianCor  N
## 36   bone_marrow 0.7677376 10
## 55  thymus_gland 0.7257205  2
## 56        tonsil 0.7107444  6
## 45        spleen 0.7059163  9
## 102  lymph_nodes 0.6884691  8
```

Finding similar samples

We may also be interested in searching GEO for similar samples. The following code will pull up samples that are similar to the given sample. (Warning: memory intensive)

```
findSimilarGSM(A[,1], minCor = 0.95, retain = 10)
```