DeMixT Log

Cell type-specific deconvolution of heterogeneous tumor samples with two or three components using expression data from RNAseq or microarray platforms.

Update from version 1.2.4 to 1.2.5

Error appears if the number of genes is not mutiplied by 10.

```
ngene.Profile.selected<-min(1500, 0.3*nrow(data.Y))
```

```
ngene.Profile.selected<-min(1500, round(0.3*nrow(data.Y)))
```

Update from version 1.2.3 to 1.2.4

The step for computing MuN mis-specify the groupid, since MuN corresponds to goupdid == 1.

```
mun.obs<-rowMeans(log2(inputdata2[, groupid == 3]+0.001))
```

```
mun.obs<-rowMeans(log2(inputdata2[, groupid == 1]+0.001))
```

Update from version 1.2.1 to 1.2.3

Major Changes

1. Added pi01 and pi02 as input values for users to initialize the proportion estimation.
2. Added nspikein as an input value in the DeMixT, DeMixT_S1 and DeMixT_GS functions to specify how many spike-in normal reference samples need to be generated; Setting nspikein at null as a default value, the number of spike-in normal reference samples equal the $min(200, 0.3 \times My)$, where My is the number of mixed samples; By setting nspikein equals 0, no spike-in normal reference will be generated; If the input value of data.N2 is not null, nspikein will be forced to be 0.
3. Added DeMixT_GS function, new proposed gene selection method which applies profile likelihood, for proportion estimation.
4. Added simulate_2comp function for users to simulate test data for 2-component de-convolution.
5. Added simulate_3comp function for users to simulate test data for 3-component de-convolution.
6. Added row names and column names for all output values.
7. Added gene.selection.method as an input value for DeMixT function. The default is 'GS'.
8. Added ngene.Profile.selected as an input value for DeMixT function. The default is NA.

Bug fix

The filter step failed to set the value into inputdata in the previous version, and it has been resolved in the 1.3.0 version.

```
## filter out genes with constant value across all samples
  inputdata < ifelse(is.null(data.comp2),
              inputdata[apply(data.comp1, 1, function(x) length(unique(x)) > 1), ],
```

```
            inputdata[apply(data.comp1, 1, function(x) length(unique(x)) > 1) &
                      apply(data.comp2, 1, function(x) length(unique(x)) > 1), ])
```

```
## filter out genes with constant value across all samples
if (is.null(data.comp2)) {
        if (dim(inputdata)[1] == 1) {
            inputdata <- t(as.matrix(inputdata[apply(data.comp1,
                1, function(x) length(unique(x)) > 1), ]))
        }
        else {
            inputdata <- inputdata[apply(data.comp1, 1, function(x) length(unique(x)) >
                1), ]
        }
    }
    else {
        if (dim(inputdata)[1] == 1) {
            inputdata <- t(as.matrix(inputdata[apply(data.comp1,
                1, function(x) length(unique(x)) > 1) & apply(data.comp2,
                1, function(x) length(unique(x)) > 1), ]))
        }
        else {
            inputdata <- inputdata[apply(data.comp1, 1, function(x) length(unique(x)) >
                1) & apply(data.comp2, 1, function(x) length(unique(x)) >
                1), ]
        }
    }
```