Assignment of Proteins to Subcellular Locations Using protlocassign

Dirk Moore

2019-05-02

Determining the locations of proteins in the cell is an important but complex problem. Differential centrifugation of cellular components into multiple fractions is a useful tool for doing this. This package takes data from centrifugation fractions for large numbers of proteins to determine their relative abundance among the fractions, and by extension to their relative abundance among subcelluar locations. The input data consisist of a file with one column, `geneName`, of gene/protein identifers, and multiple columns of the relative abundances of the proteins among the centrifugal fracions, with the abundance levels constrained to sum to 1. The fractions typically comprise the N, M, L1, L2, P, and S fractions, and may be supplemented by one to three additional "Nyc" fractons. Another key input to the program is a curated list of reference genes and their subcellular locations.This list is used to classify addtional genes to subcelluar locations.

Two assignment methods are implemented. One is known as "constrained proportional assignmet", or CPA (Jatot et al., 2016). This method first determines the profiles of abundance levels of genes in the reference set. Then, for each gene, finds either a profile matching a single compartment or a linear combination of compartment profiles that match the profile of that gene. The relative weights of the linear combination in principle reflect the relative abundance of proteins for that gene among different compartments. The CPA method, unlike metamass, can thus account for proteins that have multiple residences, and estimate the relative proportion among these residences.

Another assignment method, which has been deemed "metamass", uses unsupervised clustering to find a large number of small clusters, members of which are then assigned to the subcellular location of the location of the majority of reference genes in that cluster; genes in clusters with no reference genes remain unassigned. This method is designed to assign genes which are mainly resident in a single compartment.

Tutorial on gene assignment

Let us illustrate with an example. Jadot et al. (2016) presented abundance levels of proteins among seven fractions: N, M, L1, L2, P, and S, and an additional fraction, "Nyc2", from a Nycodenz fraction. Eight subcellular compartments were considered: nucleus, mitochondria, lysosomes, peroxisomes, golgi apparatus, plasma membrane, and cytosol. The CPA method assigns each of a large number of genes to one or more of these compartments, based on profiles from a set of reference genes. To run the program, the `prolocate` library must be installed. Also, the `BB` library is required; it may be downloaded from CRAN by typing `install.packages("BB")`.

```
library("protlocassign")
dim(geneProfileSummaryJadotExptA)
#> [1] 8071   10
head(geneProfileSummaryJadotExptA)
#>        geneName         N          M         L1         L2          P
#> 1 0610009D07RIK 0.5875969 0.02312890 0.04528676 0.02813943 0.10588824
#> 2 0610012H03RIK 0.2186240 0.46027291 0.21519842 0.05234903 0.01877835
#> 3 0610040J01RIK 0.2260143 0.02368458 0.11324577 0.19182315 0.17064529
#> 4 1110038F14RIK 0.2776881 0.03792107 0.06369487 0.06476193 0.22298680
#> 5 1600012H06RIK 0.1420997 0.05279001 0.10023629 0.24021942 0.34647028
#> 6 1600014C10RIK 0.1170937 0.05020569 0.16565689 0.14915697 0.11178078
#>            S       Nyc2 Nspectra Nseq
#> 1 0.14862414 0.06133561        4    2
#> 2 0.01557216 0.01920511       12    2
#> 3 0.03756328 0.23702364        4    2
#> 4 0.25094495 0.08200228        2    2
#> 5 0.03494385 0.08324044        2    2
#> 6 0.25962393 0.14648204        4    1
```

The data set gives relative fraction levels for 8071 genes. The last two columns give the number of spectra and the number of peptides (sequences) for each gene. (The spectra were averaged using a nested random effects model described in Jadot, 2016, to produce the given average for each gene.)

The reference genes may be accessed as follows:

```
dim(refLocProteinsJadot)
#> [1] 39  2
refLocProteinsJadot[c(1,2,6,7,12,13,17,18,21,22,27,28,31,32,35,36),]
#>    geneName referenceCompartment
#> 1      ADH1              Cytosol
#> 2      DPP3              Cytosol
#> 6      CANX                   ER
#> 7     GANAB                   ER
#> 12  B4GALT1                Golgi
#> 13   MAN1A2                Golgi
#> 17     ACP2             Lysosome
#> 18     CTSD             Lysosome
#> 21   Mt-CO2         Mitochondria
#> 22     CPS1         Mitochondria
#> 27 HIST1H1D              Nucleus
#> 28     LMNA              Nucleus
#> 31      CAT          Peroxisomes
#> 32     HAO1          Peroxisomes
#> 35   ATP1A1      Plasma membrane
#> 36     CD38      Plasma membrane
```

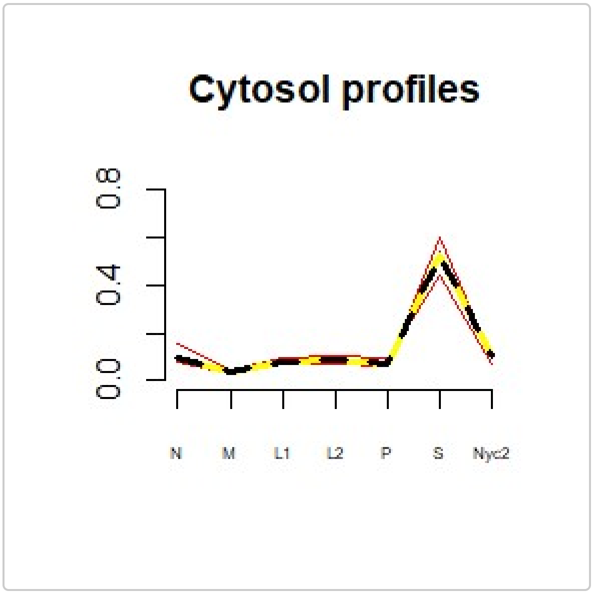
There are 39 genes in this reference set; two for each compartment are shown here.

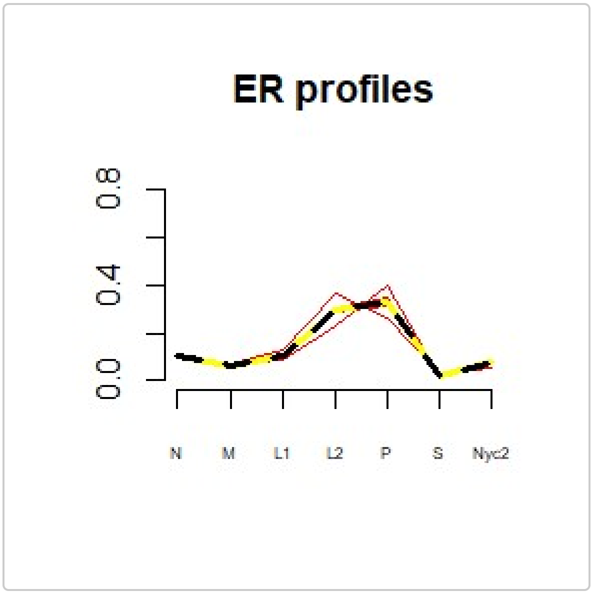
To obtain profiles for the reference genes, use the function `cpaSetup`:

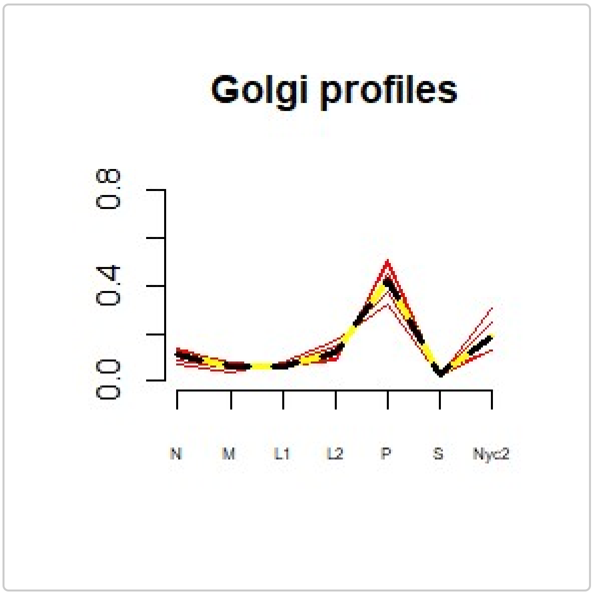
```
matLocR <- cpaSetup(geneProfileSummary=geneProfileSummaryJadotExptA,
refLocProteins=refLocProteinsJadot, n.channels=7)
matLocR
#>                          N          M         L1         L2          P
#> Cytosol         0.09819630 0.03998721 0.07980746 0.08795881 0.07250421
#> ER              0.10137398 0.06382059 0.10804811 0.29722085 0.33162059
#> Golgi           0.11044999 0.06453344 0.06593486 0.11789582 0.42642843
#> Lysosome        0.02372596 0.04463563 0.13307320 0.09611970 0.03247701
#> Mitochondria    0.17199159 0.42946893 0.20654120 0.08728049 0.03772394
#> Nucleus         0.86302244 0.03135913 0.02140297 0.02069292 0.03092168
#> Peroxisomes     0.05679574 0.08153497 0.43140095 0.23619886 0.04476925
#> Plasma membrane 0.16867566 0.07843462 0.11052456 0.18412540 0.17098242
#>                          S        Nyc2
#> Cytosol         0.51896756 0.102578450
#> ER              0.02195555 0.075960332
#> Golgi           0.02754639 0.187211067
#> Lysosome        0.02210800 0.647860499
#> Mitochondria    0.02885494 0.038138908
#> Nucleus         0.02837987 0.004220997
#> Peroxisomes     0.05685481 0.092445422
#> Plasma membrane 0.02407736 0.263179980

##referenceProfilePlot(refLocProteins = refLocProteinsJadot,
##  geneProfileSummary = geneProfileSummaryJadotExptA, matLocR = matLocR, n.channels=7)
```

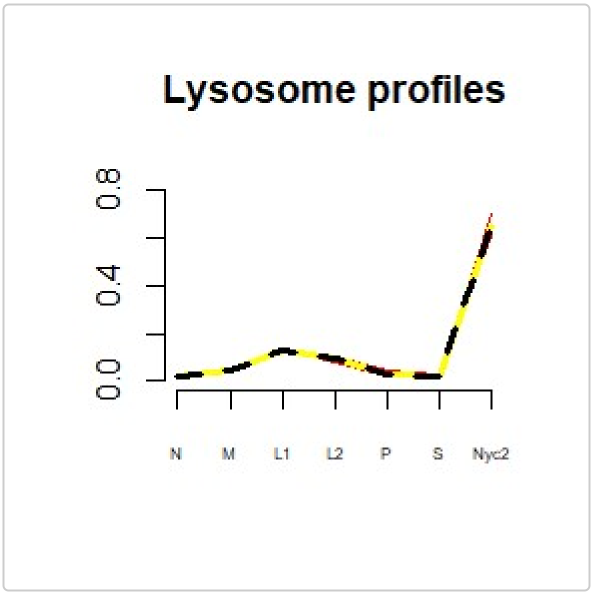
To view them, use `referenceProfilePlot`:

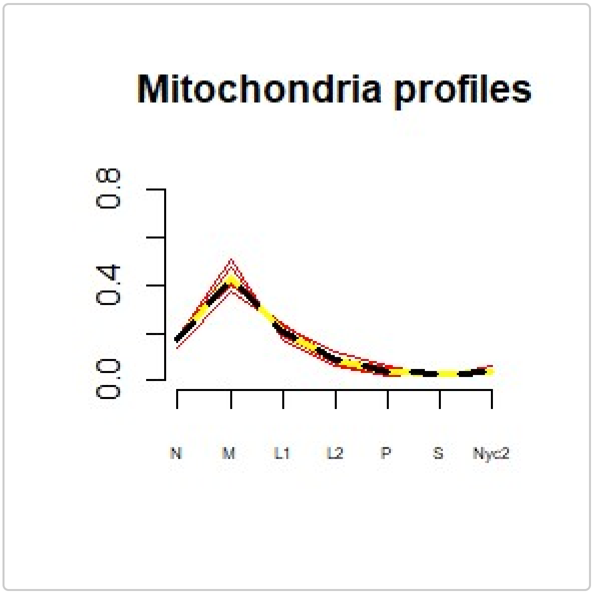
```
#par(mfrow=c(4,3))
referenceProfilePlot(refLocProteins=refLocProteinsJadot,
geneProfileSummary=geneProfileSummaryJadotExptA,
                     matLocR=matLocR, dataUse=dataUse, n.channels=7)
```

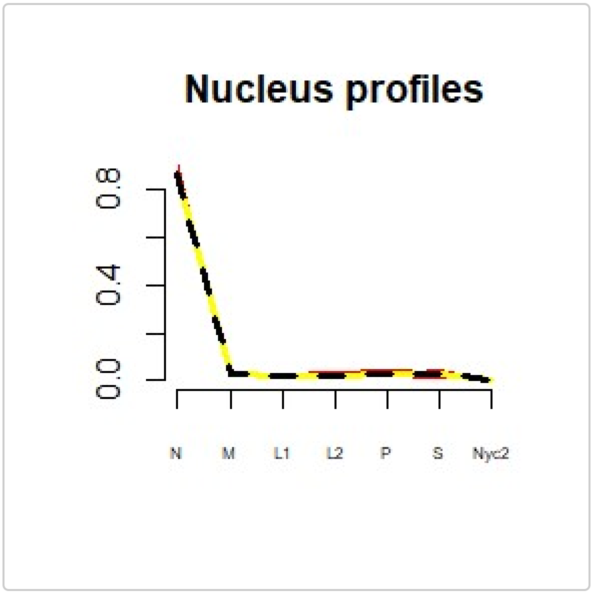

Cytosol profiles
0.0
0.4
0.8
N
M
L1
L2
P
S
Nyc2

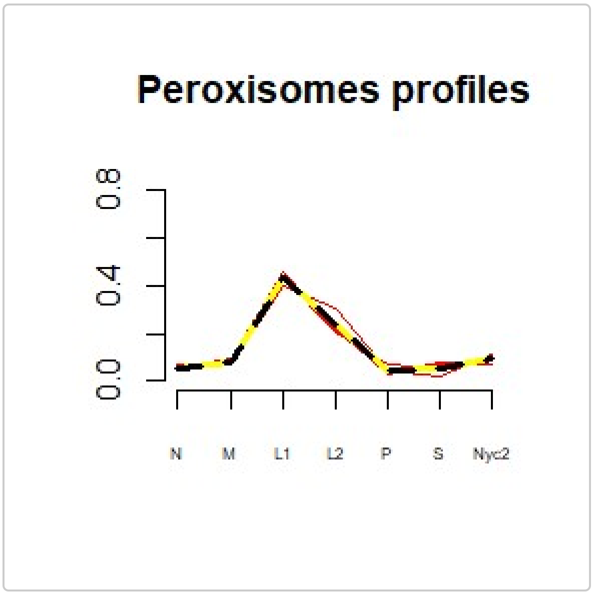

ER profiles
0.0
0.4
0.8
N
M
L1
L2
P
S
Nyc2

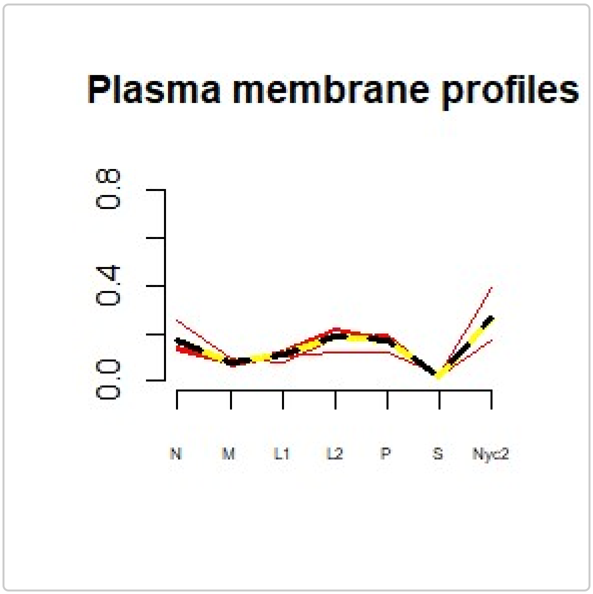

Golgi profiles
0.0
0.4
0.8
N
M
L1
L2
P
S
Nyc2


Lysosome profiles
0.0
0.4
0.8
N
M
L1
L2
P
S
Nyc2









Now run the CPA routine; this may take several minutes to complete:

```
assignProbsOut <- proLocAll(geneProfileSummary=geneProfileSummaryJadotExptA,
matLocR=matLocR,
                           n.channels=7)
#> [1] "500 genes"
#> [1] "1000 genes"
#> [1] "1500 genes"
#> [1] "2000 genes"
#> [1] "2500 genes"
#> [1] "3000 genes"
#> [1] "3500 genes"
#> [1] "4000 genes"
#> [1] "4500 genes"
#> [1] "5000 genes"
#> [1] "5500 genes"
#> [1] "6000 genes"
#> [1] "6500 genes"
#> [1] "7000 genes"
#> [1] "7500 genes"
#> [1] "8000 genes"
```

Now make a list of assignments:

```
Locations <- row.names(matLocR)
n.compartments <- 8
propMat <- assignProbsOut[,2:(n.compartments+1)]

catAssign <- apply(propMat, 1, assignCPAloc, Locations=Locations)
table(catAssign)
#> catAssign
#>         Cytosol              ER           Golgi        Lysosome
#>             908             353             112             276
#>    Mitochondria         Nucleus     Peroxisomes Plasma membrane
#>             334             108              81               5
#>    unclassified
#>            5894
assignProbsOutAssign <- data.frame(assignProbsOut, catAssign)
head(assignProbsOutAssign)
#>        geneName    Cytosol        ER      Golgi   Lysosome Mitochondria
#> 1 0610009D07RIK 0.22998361 0.0000000 0.13677155 0.00166318    0.0000000
#> 2 0610012H03RIK 0.00000000 0.0000000 0.00000000 0.00000000    0.9682433
#> 3 0610040J01RIK 0.01841482 0.4598755 0.00000000 0.29692631    0.0000000
#> 4 1110038F14RIK 0.42030626 0.0000000 0.35992362 0.00000000    0.0000000
#> 5 1600012H06RIK 0.01189244 0.7267370 0.21376948 0.00000000    0.0000000
#> 6 1600014C10RIK 0.46034186 0.1343331 0.04380622 0.09157282    0.0000000
#>      Nucleus Peroxisomes Plasma.membrane Nspectra Npeptides    catAssign
#> 1 0.63158166  0.00000000               0        4         2 unclassified
#> 2 0.03175671  0.00000000               0       12         2 Mitochondria
#> 3 0.19055148  0.03423187               0        4         2 unclassified
#> 4 0.21977012  0.00000000               0        2         2 unclassified
#> 5 0.04760106  0.00000000               0        2         2 unclassified
#> 6 0.04399695  0.22594907               0        4         1 unclassified
```

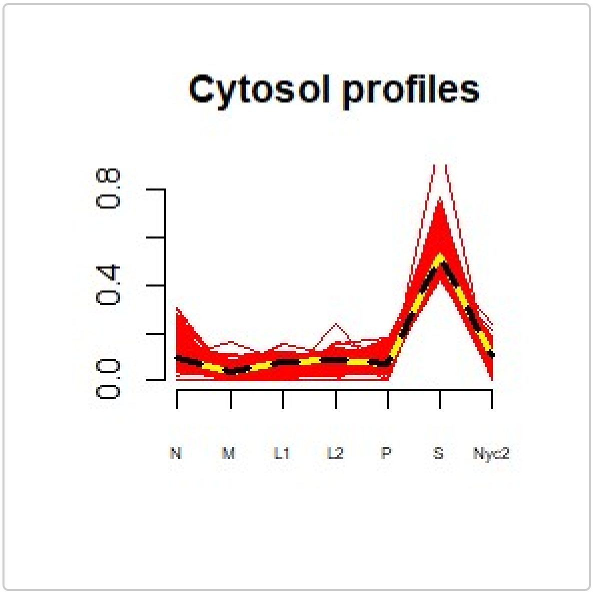
Now make a reference set, consisting of genes assigned with a proportion >= 0.8:

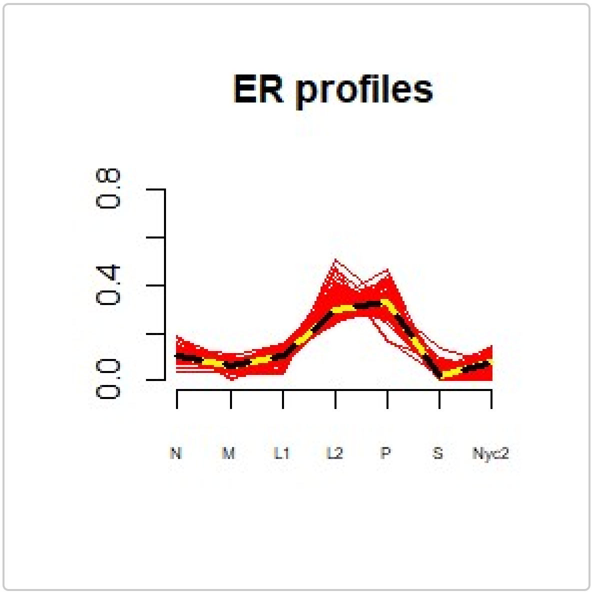
```
catAssign80.vec <- apply(propMat, 1, assignCPAloc, cutoff=0.8, Locations=Locations)


refLocProp80all <- data.frame(geneProfileSummaryJadotExptA$geneName, catAssign80.vec)
names(refLocProp80all) <- c("geneName", "referenceCompartment")
```

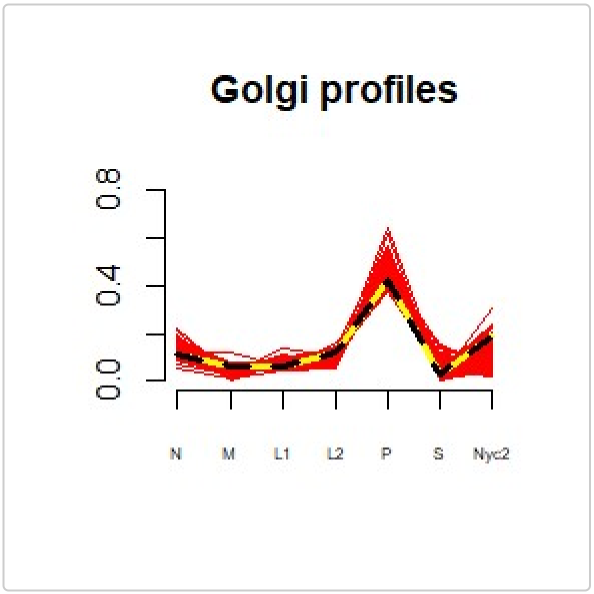
```
refLocProp80 <- refLocProp80all[refLocProp80all$referenceCompartment != "unclassified",]
dim(refLocProp80)
#> [1] 2177    2
head(refLocProp80)
#>          geneName referenceCompartment
#> 2  0610012H03RIK         Mitochondria
#> 8  1700021F05RIK         Mitochondria
#> 11 1810037I17RIK              Cytosol
#> 25 9130008F23RIK         Mitochondria
#> 34          AACS              Cytosol
#> 35         AADAC                   ER
```

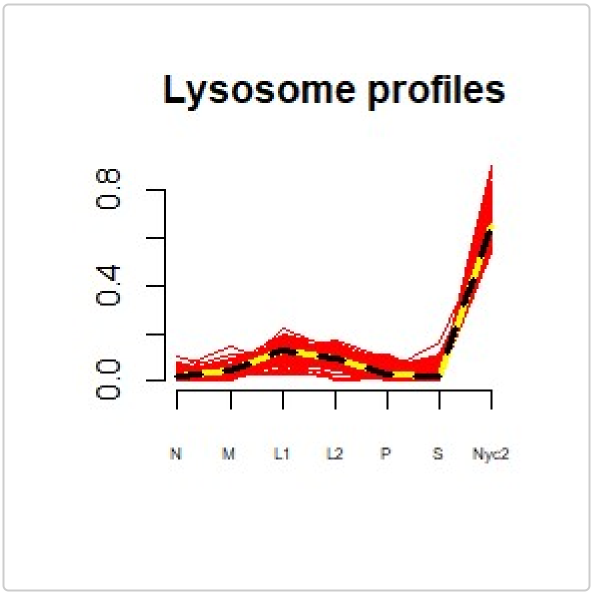
Here is a plot of the newly-created reference gene set:

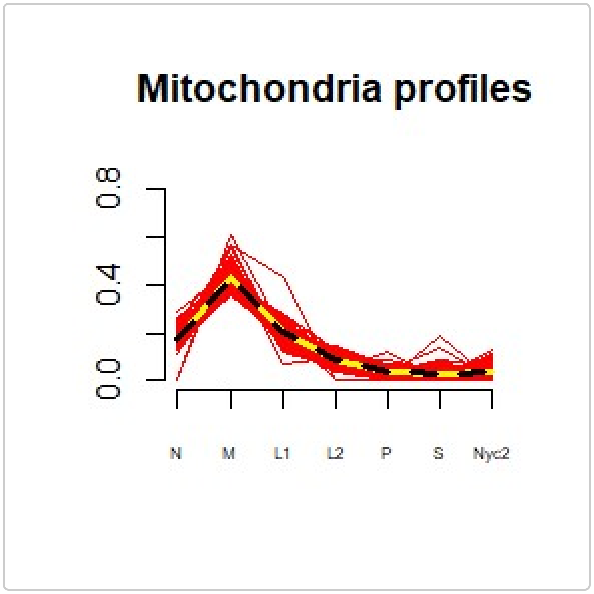
```
referenceProfilePlot(refLocProteins=refLocProp80,
geneProfileSummary=geneProfileSummaryJadotExptA,
                     matLocR=matLocR, dataUse=dataUse, markersUse=markersUse)
```

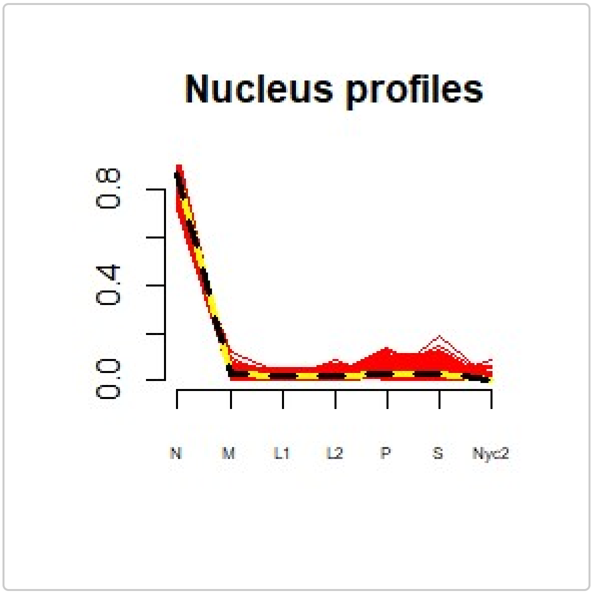

Cytosol profiles
0.0
0.4
0.8
N
M
L1
L2
P
S
Nyc2

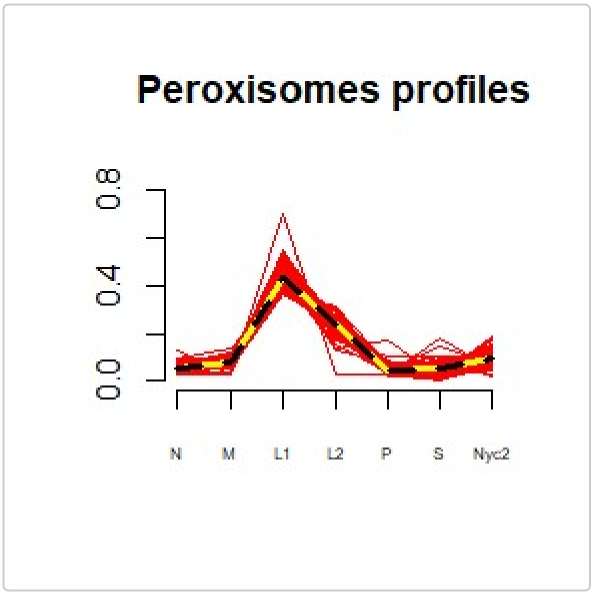

ER profiles
0.0
0.4
0.8
N
M
L1
L2
P
S
Nyc2

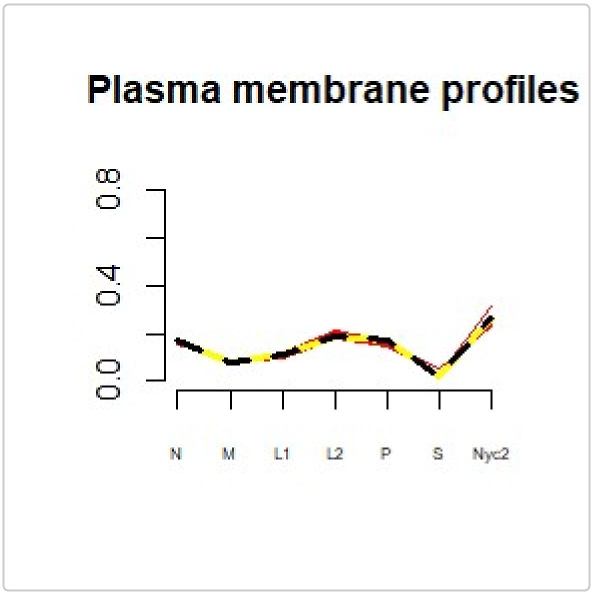

Golgi profiles
0.0
0.4
0.8
N
M
L1
L2
P
S
Nyc2


Lysosome profiles
0.0
0.4
0.8
N
M
L1
L2
P
S
Nyc2









Metamass classification

Lund-Johansen et al. (2016) proposed a metamass tool for subcellular proteomics data. A key component of metamass is an alternative method of classifying genes to subcellular locations. We have adapted this method as follows. First, we use unsupervised clustering with the “kmeans” procedure in the R system, to agglomerate all of the genes into a large number small clusters, the idea being that all genes in a cluster will belong to the same subcellular location. Following this step, a set of reference genes, with prsumably known subcellular residences, is merged with these clusters. Then for each cluster, the majority residence of the reference genes is identified, and then all genes in the cluster are assigned to that residence. Genes in clusters containing no reference genes remain unclassified. Also, if a cluster contains reference genes with multiple residences, and if there is a tie for the maximum, all genes in that cluster are marked as unclassified. When a cluster contains genes with multiple residences, the “purity” is defined as the ratio of the number of reference genes of the majority residence divided by the total number of reference genes in that cluster.

Now carry out the metamass clustering procedure, using an average cluster size of 10 genes, and the reference set we just created.

```
geneProfileSummaryMclustOnce <-  protCluster
(geneProfileSummary=geneProfileSummaryJadotExptA,
                                             refLocProteins=refLocProp80, refCol=1,
nCol=7, clustSize=10)
head(geneProfileSummaryMclustOnce$clProtsOrd)
#>        cl geneName          N          M         L1         L2          P
#> PGPEP1  1   PGPEP1 0.02197244 0.10258042 0.07626914 0.02873832 0.02562998
#> AAMP    2     AAMP 0.10323287 0.04995598 0.07307953 0.07408913 0.07913469
#> ACSS2   2    ACSS2 0.10179294 0.03988908 0.08250803 0.08808567 0.08555039
#> AKR1A1  2   AKR1A1 0.10391786 0.05209112 0.08510958 0.08297541 0.07114628
#> AKR1C2  2   AKR1C2 0.11111885 0.03587673 0.08737973 0.08716807 0.08520215
#> BLVRB   2    BLVRB 0.12278730 0.04149948 0.07863697 0.07867181 0.06941121
#>                S       Nyc2 Nspectra Nseq classif
#> PGPEP1 0.5089358 0.23587393        2    1 Cytosol
#> AAMP   0.5276202 0.09288756       11    8 Cytosol
#> ACSS2  0.5121636 0.09001029       60   19 Cytosol
#> AKR1A1 0.5082134 0.09654631      196   20 Cytosol
#> AKR1C2 0.5216058 0.07164869        1    1 Cytosol
#> BLVRB  0.5178186 0.09117462      121   11 Cytosol
head(geneProfileSummaryMclustOnce$purityStats)
#>        purity.vec maxCat.vec num.classified.vec n.clust.vec
#> PGPEP1          1    Cytosol                  1           1
#> AAMP            1    Cytosol                 27          27
#> ACSS2           1    Cytosol                 27          27
#> AKR1A1          1    Cytosol                 27          27
#> AKR1C2          1    Cytosol                 27          27
#> BLVRB           1    Cytosol                 27          27
head(geneProfileSummaryMclustOnce$purity.df)
#>        Cytosol ER Golgi Lysosome Mitochondria Nucleus Peroxisomes
#> PGPEP1       1  0     0        0            0       0           0
#> AAMP         1  0     0        0            0       0           0
#> ACSS2        1  0     0        0            0       0           0
#> AKR1A1       1  0     0        0            0       0           0
#> AKR1C2       1  0     0        0            0       0           0
#> BLVRB        1  0     0        0            0       0           0
#>        Plasma membrane
#> PGPEP1               0
#> AAMP                 0
#> ACSS2                0
#> AKR1A1               0
#> AKR1C2               0
#> BLVRB                0
```

The first column, “cl”, is the cluster number. The column “classif” is the original reference classification, and “maxCat.vec” is the residency assigned by the metamass procedure.

Appendix

Outlier rejection and computation of mean profiles

The profiles of some compartments (e.g. nuclear and mitochondrial) show a high relative abundance level in the N or M fractions, respectively, and little in the others. Cytosolic proteins also are high in one fraction (S) and low in the others. Peroxisomes have high activity levels in the L1 and L2 fractions, and low levels in the others, including Nyc2. Lysosomes, like peroxisomes, have high level in the L1 and L2 fractions, but, unlike peroxisomes, high levels in Nyc2. The other compartments (ER, Golgi, and PM) have more complex profiles. Once we have obtained reference profiles for each of the eight compartments, we score each new protein as the optimal combination of the reference proteins, thereby obtaining estimates of the proportionate contributions of each reference profile to that of the protein being considered.

The first step is an outlier screen, for each protein, of the abundance levels for each fraction of all of the spectra. Abundance levels p are first log2 transformed via $y = log_2(p + \delta)$, where δ is an estimate of the background “noise” in the abundance estimates. Here we used $\delta = 10^{(-5)}$. Then boxplot outliers are identified as values more than three times the interquartile range beyond the first or third quartile. We discard any spectrum for which any observation is thereby classified as an outlier. This process is repeated for all proteins.

Constrained proportional assignment

Once outlier spectra have been removed, the next step is to determine a mean profile for each protein. If a protein has at least 4 spectra and at least 3 different sequences (peptides), we ordinarily have enough data to fit a random effects model. This model computes, for each EF in each fraction of this protein, a weighted average and standard error of the measures, accounting for the fact that spectra are nested within sequences. (This computation is carried out using the “lmer” function in the “lme4” R package.) . The result is essentially the mean and standard error of the observations, with an adjustment for the nested structure of the data. This procedure prevents a sequence with a very large number of spectra from dominating the estimates of the mean and standard error. Occasionally the “lmer” program will fail to converge for a particular abundance. If that happens or if the condition of having at least 4 spectra and 3 different sequences is not met, we compute the simple mean and standard error of the (log2 transformed) for that abundance level. If there are fewer than 3 spectra, we cannot compute a standard error, and thus only report the mean. When this process has been completed, every protein has a mean profile $\underset{\sim}{x} = (x_1, x_2, \ldots, x_q)$ of abundance levels in the q fractions N, M, L1, L2, P, S, and Nyc2. For proteins with at least 2 peptides and three spectra, each component of the profile has an estimate of its standard deviation.

Next, the profiles of the reference proteins are selected. For each of the eight compartments (Cytosol, ER, Golgi, Lysosome, Mitochondria, Nucleus, Peroxisome, and PM), the reference protein relative abundance levels are averaged to form eight compartment profiles $\underset{\sim}{s}_1, \underset{\sim}{s}_2, \ldots, \underset{\sim}{s}_8$, where each vector $\underset{\sim}{s}_j$ is a $q \times 1$ vector of mean levels of the $q = 7$ or $q = 9$ channels.

Finally, for each protein, we find estimated proportions $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_8$ so that

$$\underset{\sim}{y} = \hat{p}_1 \underset{\sim}{s}_1 + \ldots + \hat{p}_8 \underset{\sim}{s}_8$$

is as close as possible to the observed value $\underset{\sim}{x}$, subject to the constraints

$$0 \leq p_j \leq 1 \text{ for all } j, \text{ and}$$

$$\sum_{j=1}^{8} \hat{p}_j = 1$$

where "close" is defined by minimizing the sum of squares of the differences

$$Q = \sum_{i=1}^{q} (y_i - x_i)^2$$

Thus, we may view the proportions $\hat{p}_j$ as proportional allocations of the eight standard profiles to form $\underset{\sim}{y}$, which is as close as possible to the observed $\underset{\sim}{x}$ for this particular protein. This constrained optimization is carried out using the "spg" function in the R package "BB" to compute assignment probabilities for each profile for each organelle.

Adjustment of protein abundance levels

The input data, geneProfileSummary, consists of a list of gene names (first column) and the relative abundance of the corresponding proteins in the next columns. Typically, there will be six differential fraction columns (N, M, L1, L2, P, and S) and one to three additional Nycodenz fractions (which have been extracted from the L1 differential fraction). These quantities represent the relative amounts of a protein in each fraction. The amounts of protein extracted from each fraction, totProtein, differ dramatically from one fraction to another. Notably, the L1 and L2 fractions, which are heavily enriched in lysosomal and peroxixomal proteins, contain much smaller amounts of protein than the other differential fractions. Also, the samples used for the Nycodenz fractionization are taken from the L1 fraction, so they also represent very small proportions of the total sample. For technical reasons, when samples of material are prepared for analysis, equal amounts are selected for each channel, regardless of the amount of protein in the fractions. As a result, when the data are analyzed, adjustments need to be made in order to obtain meaningful comparisons.

The first transformation involves finding the amount of a given protein in a fraction divided by the amount of protein in the starting material. To see how this works, let us consider the Jadot reference protein profiles

```
matLocR <- cpaSetup(geneProfileSummary=geneProfileSummaryJadotExptA,
refLocProteins=refLocProteinsJadot, n.channels=7)
round(matLocR, digits=4)
```

```
#>                      N      M     L1     L2      P      S   Nyc2
#> Cytosol         0.0982 0.0400 0.0798 0.0880 0.0725 0.5190 0.1026
#> ER              0.1014 0.0638 0.1080 0.2972 0.3316 0.0220 0.0760
#> Golgi           0.1104 0.0645 0.0659 0.1179 0.4264 0.0275 0.1872
#> Lysosome        0.0237 0.0446 0.1331 0.0961 0.0325 0.0221 0.6479
#> Mitochondria    0.1720 0.4295 0.2065 0.0873 0.0377 0.0289 0.0381
#> Nucleus         0.8630 0.0314 0.0214 0.0207 0.0309 0.0284 0.0042
#> Peroxisomes     0.0568 0.0815 0.4314 0.2362 0.0448 0.0569 0.0924
#> Plasma membrane 0.1687 0.0784 0.1105 0.1841 0.1710 0.0241 0.2632

##referenceProfilePlot(refLocProteins = refLocProteinsJadot,
##  geneProfileSummary = geneProfileSummaryJadotExptA, matLocR = matLocR, n.channels=7)
```

Here, each row represents the profile for a protein resident solely in a particular compartment. The amount of starting material, totProt, in each fraction is as follows:

```
totProt=c(46.044776, 48.955954, 1.384083, 1.566324, 24.045584, 58.181818, 0.0684596)
totProtdf <- data.frame(t(matrix(totProt)))
names(totProtdf) <- colnames(matLocR)
totProtdf
#>          N        M       L1       L2        P        S      Nyc2
#> 1 46.04478 48.95595 1.384083 1.566324 24.04558 58.18182 0.0684596
```

The total amount of starting material is the sum of the amounts given in the first six (differential) fractions (N, M, L1, L2, P, and S).

```
sum(totProt[1:6])
#> [1] 180.1785
```

(The last amount, Nyc2, is the middle of three Nycodenz fractions, which were derived from L1.)

To compute the amount of protein in each fraction, we multiply each column by the amount of starting material in each fraction. The function abundanceTransform does this:

```
totProt=c(46.044776, 48.955954, 1.384083, 1.566324, 24.045584, 58.181818, 0.0684596)
protAbund <- abundanceTransform(matLocR,6,1, totProt=totProt)
round(protAbund$amtProtFrac, digits=4)
#>                       N       M     L1     L2       P       S   Nyc2
#> Cytosol          4.5214  1.9576 0.1105 0.1378  1.7434 30.1945 0.0070
#> ER               4.6677  3.1244 0.1495 0.4655  7.9740  1.2774 0.0052
#> Golgi            5.0856  3.1593 0.0913 0.1847 10.2537  1.6027 0.0128
#> Lysosome         1.0925  2.1852 0.1842 0.1506  0.7809  1.2863 0.0444
#> Mitochondria     7.9193 21.0251 0.2859 0.1367  0.9071  1.6788 0.0026
#> Nucleus         39.7377  1.5352 0.0296 0.0324  0.7435  1.6512 0.0003
#> Peroxisomes      2.6151  3.9916 0.5971 0.3700  1.0765  3.3079 0.0063
#> Plasma membrane  7.7666  3.8398 0.1530 0.2884  4.1114  1.4009 0.0180
round(protAbund$relAmtProtFrac, digits=4)
```

```
#>                       N      M     L1     L2      P      S   Nyc2
#> Cytosol         0.1169 0.0506 0.0029 0.0036 0.0451 0.7809 0.0002
#> ER              0.2643 0.1769 0.0085 0.0264 0.4516 0.0723 0.0003
#> Golgi           0.2496 0.1550 0.0045 0.0091 0.5032 0.0787 0.0006
#> Lysosome        0.1923 0.3847 0.0324 0.0265 0.1375 0.2265 0.0078
#> Mitochondria    0.2478 0.6580 0.0089 0.0043 0.0284 0.0525 0.0001
#> Nucleus         0.9087 0.0351 0.0007 0.0007 0.0170 0.0378 0.0000
#> Peroxisomes     0.2187 0.3338 0.0499 0.0309 0.0900 0.2766 0.0005
#> Plasma membrane 0.4423 0.2187 0.0087 0.0164 0.2341 0.0798 0.0010
```

The first component of the result, amtProtFrac, is the amount of protein in each fraction for each row. The second component, relAmtProtFrac, is the amount of given protein in fraction / amount of given protein in starting material. For example, for a cytosolic protein, the amount of protein in each fraction is given by

```
matLocR[,1]*totProt[1]
#>         Cytosol              ER           Golgi        Lysosome
#>        4.521426        4.667742        5.085645        1.092456
#>    Mitochondria         Nucleus     Peroxisomes Plasma membrane
#>        7.919314       39.737675        2.615147        7.766633
protAbund$amtProtFrac[1,]
#>                N        M        L1       L2        P        S       Nyc2
#> Cytosol 4.521426 1.957612 0.1104602 0.137772 1.743406 30.19448 0.00702248
```

This is the first column of amtProtFrac.

The first row of relAmtProtFrac, to continue with the example, is the first row of amtProtFrac (amount of cystolic protein in each fraction) divided by the total amount of cystolic protein, which is the sum of the first row of amtProtFrac:

```
protAbund$amtProtFrac[1,] / sum(protAbund$amtProtFrac[1,])
#>                 N          M          L1          L2          P         S
#> Cytosol 0.1169168 0.05062068 0.002856321 0.003562561 0.04508167 0.7807804
#>               Nyc2
#> Cytosol 0.00018159
```

The relative amount of a cytosolic protein is given by the amounts of protein in the first row (a cytosolic protein) divided by the amounts in all six differential fractions (the first six elements of the first row):

```
protAbund$amtProtFrac[1,]/sum(protAbund$amtProtFrac[1,])
#>                 N          M          L1          L2          P         S
#> Cytosol 0.1169168 0.05062068 0.002856321 0.003562561 0.04508167 0.7807804
#>               Nyc2
#> Cytosol 0.00018159
protAbund$relAmtProtFrac[1,]
```

```
#>                N          M         L1          L2          P         S
#> Cytosol 0.116938 0.05062988 0.00285684 0.003563208 0.04508985 0.7809222
#>                Nyc2
#> Cytosol 0.000181623
```

This is the first row of relAmtProtFrac.

The values in relAmtProtFrac represent the amount of protein per fraction, normalized to same amount of protein in the starting material (before mixing). In theory, these values correspond approximately to the relative amounts of protein in compartments within a cell.

Finally, we compute the relative specific activity (RSA):

To get this, we first find Difp, the total protein in the six differential fractions (nDiffFractions = 6), and then the proportions propFrac of protein in the differential fractions:

```
rsaResult <- RSAtransform(protAbund$relAmtProtFrac, totProt=totProt)
round(rsaResult$rsa, digits=4)
#>                       N      M     L1     L2      P      S    Nyc2
#> Cytosol         0.4576 0.1863 0.3719 0.4099 0.3379 2.4184  0.4780
#> ER              1.0344 0.6512 1.1025 3.0327 3.3837 0.2240  0.7751
#> Golgi           0.9766 0.5706 0.5830 1.0424 3.7705 0.2436  1.6553
#> Lysosome        0.7527 1.4160 4.2216 3.0493 1.0303 0.7014 20.5526
#> Mitochondria    0.9698 2.4217 1.1647 0.4922 0.2127 0.1627  0.2151
#> Nucleus         3.5559 0.1292 0.0882 0.0853 0.1274 0.1169  0.0174
#> Peroxisomes     0.8558 1.2285 6.5000 3.5589 0.6746 0.8566  1.3929
#> Plasma membrane 1.7307 0.8048 1.1341 1.8893 1.7544 0.2471  2.7004
```

```
Difp <- sum(totProt[1:6])   # total protein in the differential fractions
Difp
#> [1] 180.1785
propFrac <- totProt/Difp  # proportion of protein in the differential fractions
propFrac
#> [1] 0.2555508345 0.2717080196 0.0076817306 0.0086931774 0.1334542068
#> [6] 0.3229120312 0.0003799542
```

For example, for the N fraction (first column), we have, which matches the first column of rsa:

```
protAbund$relAmtProtFrac[,1]/propFrac[1]
#> [1] 0.4575920 1.0343604 0.9766129 0.7526794 0.9698403 3.5558969 0.8557587
#> [8] 1.7307280
rsaResult$rsa[,1]
#> [1] 0.4575920 1.0343604 0.9766129 0.7526794 0.9698403 3.5558969 0.8557587
#> [8] 1.7307280
```

Finally, if we normalize the rsa matrix so that rows sum to one, we get original input data, matLocR

```
round(rsaResult$rsaFractions, digits=4)
#>                      N      M     L1     L2      P      S   Nyc2
#> Cytosol         0.0982 0.0400 0.0798 0.0880 0.0725 0.5190 0.1026
#> ER              0.1014 0.0638 0.1080 0.2972 0.3316 0.0220 0.0760
#> Golgi           0.1104 0.0645 0.0659 0.1179 0.4264 0.0275 0.1872
#> Lysosome        0.0237 0.0446 0.1331 0.0961 0.0325 0.0221 0.6479
#> Mitochondria    0.1720 0.4295 0.2065 0.0873 0.0377 0.0289 0.0381
#> Nucleus         0.8630 0.0314 0.0214 0.0207 0.0309 0.0284 0.0042
#> Peroxisomes     0.0568 0.0815 0.4314 0.2362 0.0448 0.0569 0.0924
#> Plasma membrane 0.1687 0.0784 0.1105 0.1841 0.1710 0.0241 0.2632
round(matLocR, digits=4)
#>                      N      M     L1     L2      P      S   Nyc2
#> Cytosol         0.0982 0.0400 0.0798 0.0880 0.0725 0.5190 0.1026
#> ER              0.1014 0.0638 0.1080 0.2972 0.3316 0.0220 0.0760
#> Golgi           0.1104 0.0645 0.0659 0.1179 0.4264 0.0275 0.1872
#> Lysosome        0.0237 0.0446 0.1331 0.0961 0.0325 0.0221 0.6479
#> Mitochondria    0.1720 0.4295 0.2065 0.0873 0.0377 0.0289 0.0381
#> Nucleus         0.8630 0.0314 0.0214 0.0207 0.0309 0.0284 0.0042
#> Peroxisomes     0.0568 0.0815 0.4314 0.2362 0.0448 0.0569 0.0924
#> Plasma membrane 0.1687 0.0784 0.1105 0.1841 0.1710 0.0241 0.2632
```