nature portfolio

https://doi.org/10.1038/s41596-022-00699-2

Supplementary information

SubCellBarCode: integrated workflow for robust spatial proteomics by mass spectrometry

In the format provided by the authors and unedited

SubCellBarCode: Integrated workflow for robust classification and visualization of spatial proteome

***Taner Arslan*[1], *Lukas Orre*[1], *Mattias Vesterlund*[1], *Yanbo Pan*[1] *and Janne Lehtiö*[1]**

[1]Karolinska Institute, Stockholm, Sweden

2021-06-10

Abstract

In the SubCellBarCode project, mass spectrometry (MS) based proteomics was used for proteome-wide mapping of protein subcellular localization (Orre et al. 2019, Molecular Cell). Subcellular fractionation and MS-based quantification can be reproduced following the methods described in Orre et al. Here, the SubCellBarCode R package offers tools to reproduce the underlying bioinformatics pipeline for robust classification and visualization of proteins into corresponding subcellular localizations. Moreover, the R-package can be used for differential localization analysis to investigate e.g. treatment induced protein relocalization, condition dependent localization or cell type specific localization. Last but not least, it provides peptide/ exon/transcript centric classification as well as PTM (Post translation modifications) modified peptide classification.

Package

SubCellBarCode 1.0.0

Contents

1 Installation of the package

`SubCellBarCode` can be installed through `BiocManager` package as follows:

```
if (!requireNamespace("BiocManager"))
    install.packages("BiocManager")
BiocManager::install("SubCellBarCode")
```

2 Load the package

```
library(SubCellBarCode)
```

3 Data preparation and classification

3.1 Example Data

As example data we here provide the publicly available HCC827 (human lung adenocarcinoma cell line) TMT10plex labelled proteomics dataset (*Orre et al.* 2019, Molecular Cell). The `data.frame` consists of 10480 proteins as rows (rownames are gene - centric protein ids) and 5 fractions with duplicates as columns (replicates must be named ".A." and ".B.", repectively).

```
head(hcc827Ctrl)
```

```
##        FS1.A.HCC827 FS1.B.HCC827 FS2.A.HCC827 FS2.B.HCC827 FP1.A.HCC827
## A2M          6.6567       4.8238       0.8265       0.8279       0.4475
## A2ML1        1.2876       1.0878       0.6390       0.7828       0.8760
## A4GALT       0.4711       0.4106       0.2742       0.2689       0.8389
## AAAS         0.5108       0.4514       0.4470       0.4752       1.6576
## AACS         4.5593       4.4522       1.5694       1.6417       0.5294
## AAED1        0.8170       0.7031       0.5415       0.5902       0.9429
##        FP1.B.HCC827 FP2.A.HCC827 FP2.B.HCC827 FP3.A.HCC827 FP3.B.HCC827
## A2M          0.5803       0.6414       0.6014       0.6497       0.6216
## A2ML1        0.9138       2.6957       1.3606       0.7715       0.7859
## A4GALT       0.8043       1.9637       1.8340       1.6739       1.6848
## AAAS         1.7316       1.3773       1.4688       1.0071       1.0457
## AACS         0.5197       0.5109       0.5232       0.4197       0.4243
## AAED1        0.9505       1.9035       1.8475       1.1060       1.1636
```

3.2 Marker Proteins

The classification of protein localisation using the SubCellBarCode method is dependent on 3365 marker proteins as defined in Orre et al. The markerProteins `data.frame` contain protein names (gene symbol), associated subcellular localization (compartment),

color code for the compartment and the median normalized fractionation profile (log2) based on five different human cell lines (NCI-H322, HCC827, MCF7, A431and U251) here called the “5CL marker profile”.

```
head(markerProteins)
```

```
##          Proteins Compartments       Cyto        Nsol       NucI        Horg
## AAAS         AAAS           S4 -1.0033518 -1.15489468  0.9303367  0.48554266
## AACS         AACS           C4  2.1716569  0.13246046 -1.0394634 -1.13265585
## AAK1         AAK1           C3  1.8556445  0.10015281 -0.6605511 -0.36985132
## AARS         AARS           C4  2.1012831  0.05855811 -1.0439250 -1.08451971
## AASDHPPT AASDHPPT           C5  1.7065897  0.51507061 -0.7196594 -0.65267201
## AATF         AATF           N3 -0.8922053 -1.10101864  1.2202070  0.05316807
##                Lorg       Colour
## AAAS      0.1618329      tomato2
## AACS     -1.3217619 deepskyblue2
## AAK1     -0.6741959         cyan
## AARS     -1.2180979 deepskyblue2
## AASDHPPT -1.0059841   turquoise3
## AATF      0.1898676       grey50
```

3.3 Load and normalize data

Input `data.frame` is checked with *"NA"* values and for the correct format. If there is any *"NA"* value, corresponding row is deleted. Then, data frame is `log2` transformmed.

```
df <- loadData(protein.data = hcc827Ctrl)
```

```
cat(dim(df))
```

```
## 10480 10
```

```
head(df)
```

```
##        FS1.A.HCC827 FS1.B.HCC827 FS2.A.HCC827 FS2.B.HCC827 FP1.A.HCC827
## A2M       2.7348072    2.2701701   -0.2749133   -0.2724716  -1.16004041
## A2ML1     0.3646845    0.1214133   -0.6461122   -0.3532843  -0.19099723
## A4GALT   -1.0858948   -1.2841945   -1.8666995   -1.8948583  -0.25342925
## AAAS     -0.9691696   -1.1475217   -1.1616533   -1.0733933   0.72909591
## AACS      2.1888123    2.1545184    0.6502131    0.7151905  -0.91756990
## AAED1    -0.2915920   -0.5081982   -0.8849668   -0.7607242  -0.08482332
##        FP1.B.HCC827 FP2.A.HCC827 FP2.B.HCC827 FP3.A.HCC827 FP3.B.HCC827
## A2M     -0.78512917   -0.6407037   -0.7336032  -0.62215439  -0.68594159
## A2ML1   -0.13004965    1.4306600    0.4442430  -0.37426194  -0.34758234
## A4GALT  -0.31419437    0.9735745    0.8749936   0.74321334   0.75257734
## AAAS     0.79210571    0.4618428    0.5546380   0.01020694   0.06446902
## AACS    -0.94424904   -0.9688872   -0.9345656  -1.25256963  -1.23684342
## AAED1   -0.07324147    0.9286546    0.8855744   0.14535139   0.21859520
```

Additional step: We use gene symbols for the protein identification. Therefore, we require gene symbols for the identifiaction. However, if the input data has other identifier e.g. UNIPROT, IPI, Entrez ID, you can convert it to gene symbol by our defined function.
Please be aware of possible (most likely few) id loss during the conversion to one another.

```
##Run if you have another identifier than gene symbols.
##Function will convert UNIPROT identifier to gene symbols.
##Deafult id is "UNIPROT", make sure you change it if you use another.

#df <- convert2symbol(df = df, id = "UNIPROT")
```

For the downstream analysis, we used the randomly selected subset data.

```
set.seed(2)
```

```
df <- df[sample(nrow(df), 6000),]
```

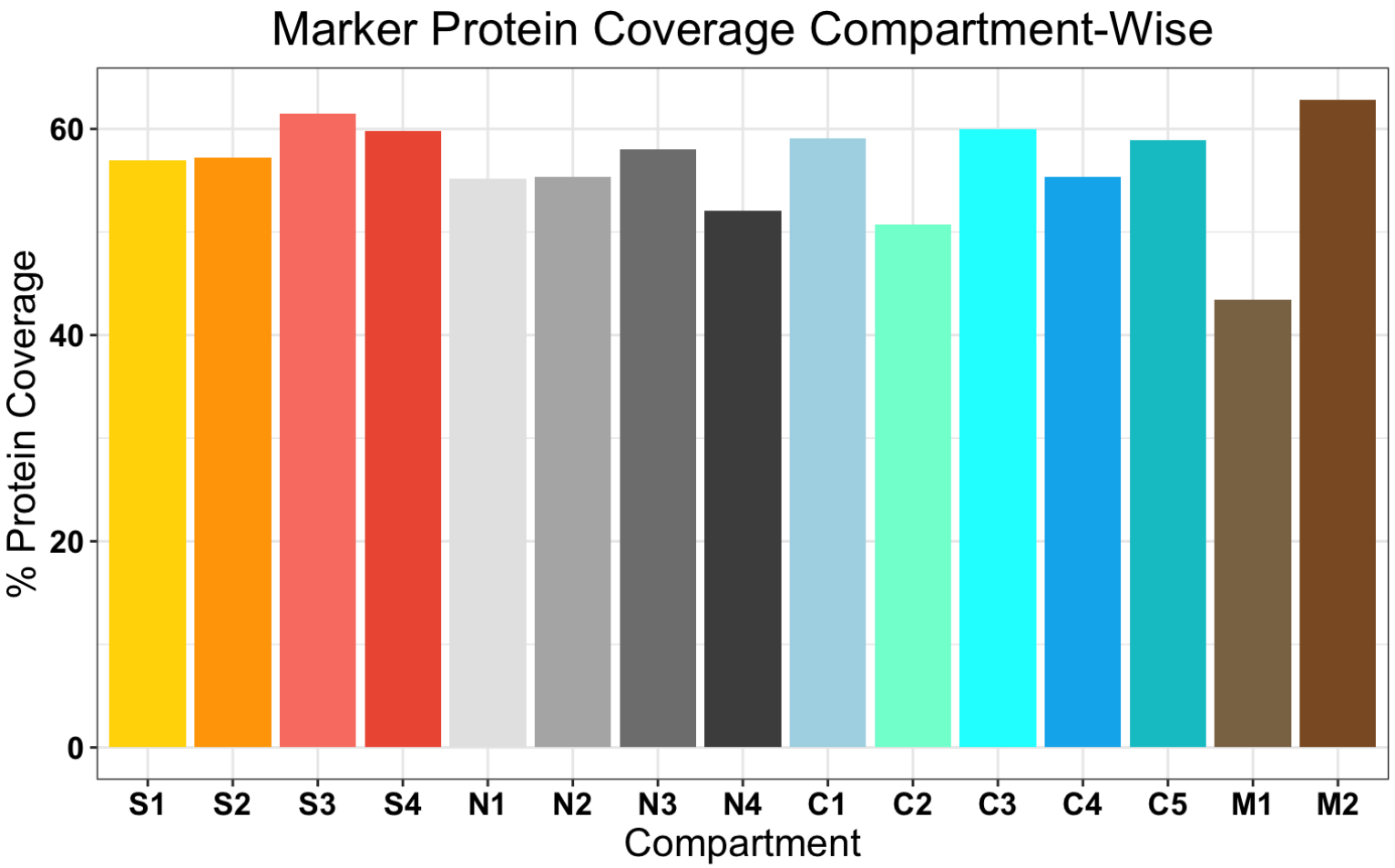
3.4 Calculate covered marker proteins

Marker proteins (3365) were defined in the original publication *(Orre et. al, 2019).* They were embedded into the package and further used for the downstream analysis. In step 94, we calculate the marker protein coverage per compartments.

The overlap between *marker proteins (3365)* and *input data.frame* is calculated and visualized for each compartment by a bar plot.

Note that we recommend at least 20% coverage of marker proteins for each compartment. If certain compartments are underrrepresented we recommend you to perform the cell fractionation again. If all compartments are low in coverage we recommend increasing the analytical depth of the MS-analysis.

```
c.prots <- calculateCoveredProtein(proteinIDs = rownames(df),
                        markerproteins = markerProteins[,1])
```




```
## Overall Coverage of marker proteins :  0.58
```

3.5 Quality control of the marker proteins

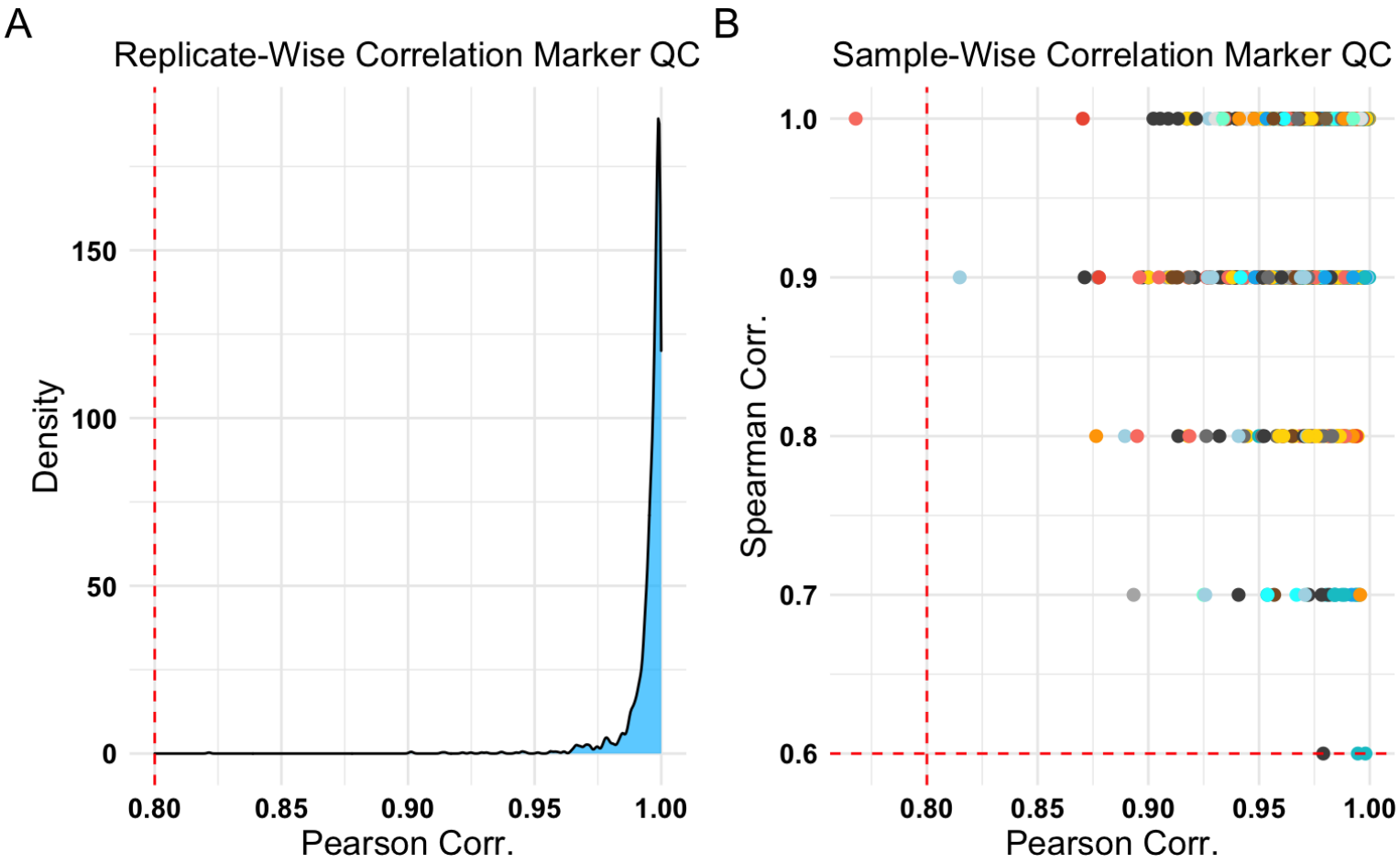
To avoid reduced classification accuracy, marker proteins with noisy quantification and marker proteins that are not representative of their associated compartment (e.g.due to cell type specific localization) are filtered out by a two-step quality control.

1. Marker proteins with pearson correlations less than 0.8 between *A* and *B* duplicates for each cell line were filtered out (Figure A).
2. Pairwise correlations between 5CL marker profile and input data for each protein (A and B replicate experiments separately) were calculated using both Pearson and Spearman correlation. The lowest value for each method were then used for filtering with cut-offs set to 0.8 and 0.6 respectively, to exclude non-representative marker proteins (Figure

B).

```
r.markers <- markerQualityControl(coveredProteins = c.prots,protein.data = df)
```

```
## Number of removed replicate-wise proteins: 0
## Number of removed sample-wise proteins: 1
## Number of total removed marker proteins: 1
```




```
r.markers[1:5]
```

```
## [1] "N4BP2L2" "FURIN"   "ATG9A"   "ZNF205"  "MAVS"
```

Optional step: After removing non-marker proteins, you can re-calculate and visualize the final coverage of the marker proteins.

```
## uncomment the function when running
# f.prots <- calculateCoveredProtein(r.markers, markerProteins[,1])
```

3.6 Visualization of marker proteins in t-SNE map

The spatial distribution of the marker proteins is visualized in t-SNE map. This plot will be informative for the quality control of the generated data as it offers evaluation of the spatial distribution and separation of marker proteins.

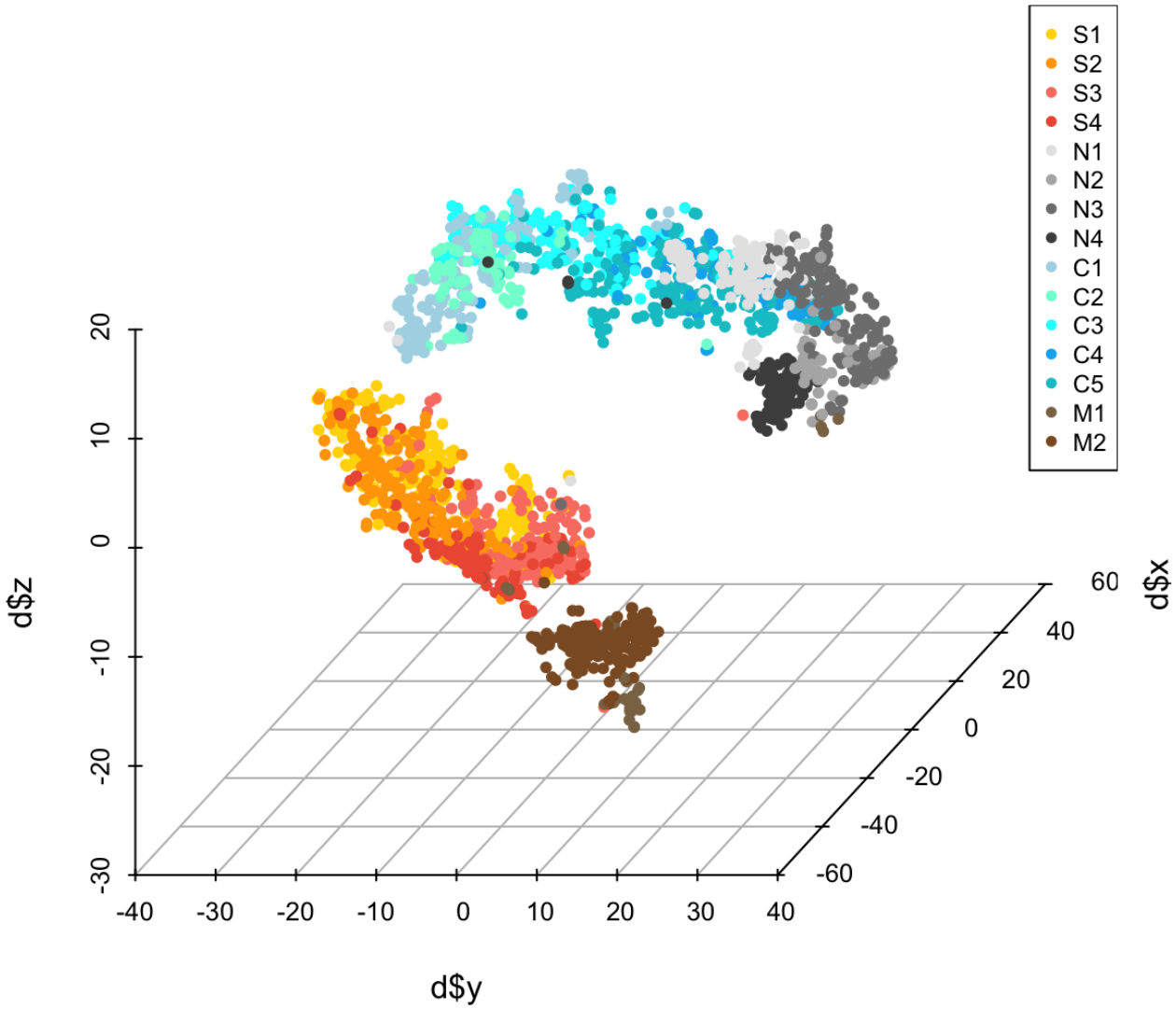
```
#Default parameters
#Run the broad-range parameters to optimize well !!!
#Output dimensionality
#dims = 3
#Speed/accuracy trade-off (increase for less accuracy)
#theta = c(0.1, 0.2, 0.3, 0.4, 0.5)
#Perplexity parameter
#perplexity = c(5, 10, 20, 30, 40, 50, 60)
```

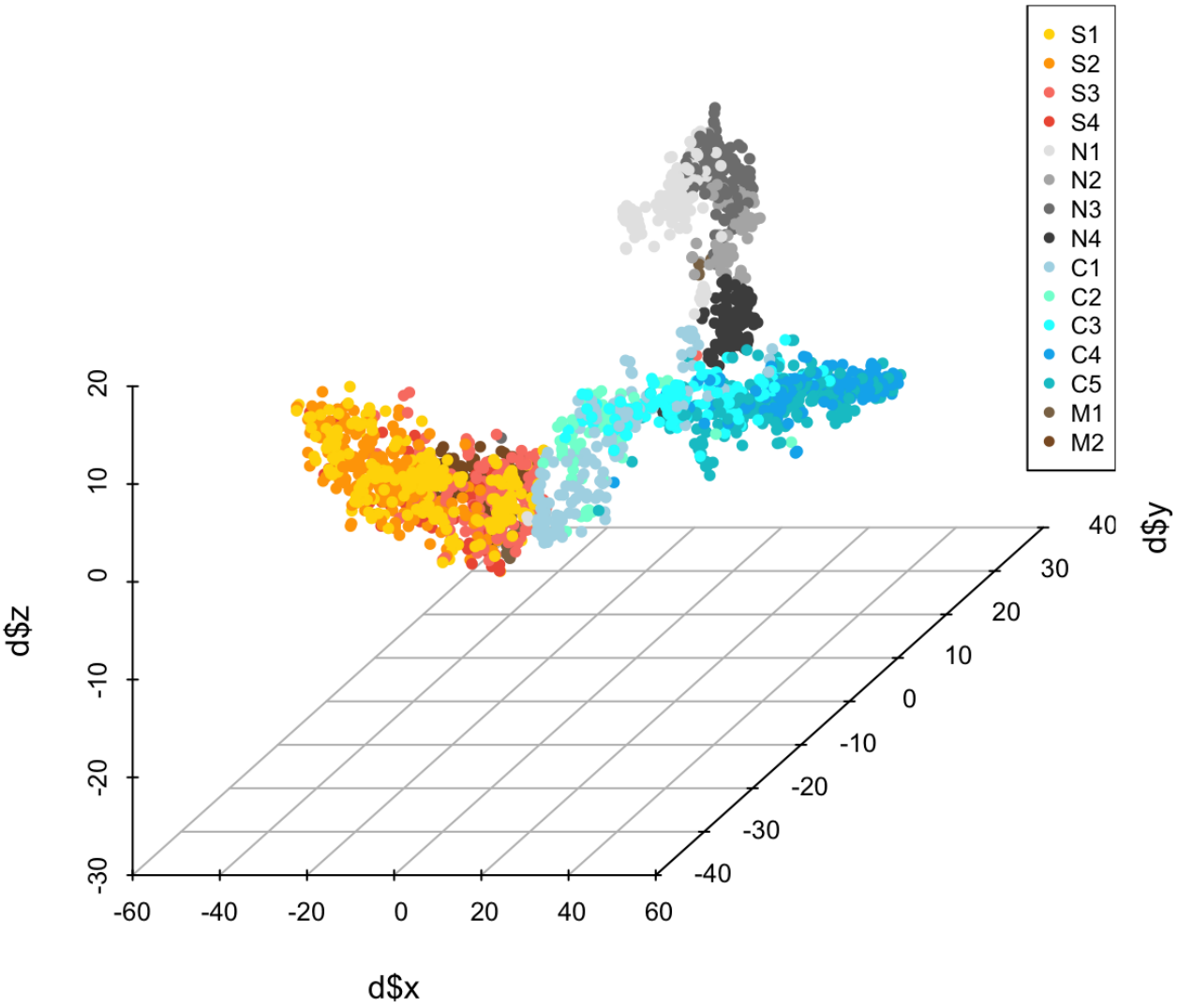
Information about the different t-SNE parameters that can be modified by the user is available by typing `?Rtsne` in the console.

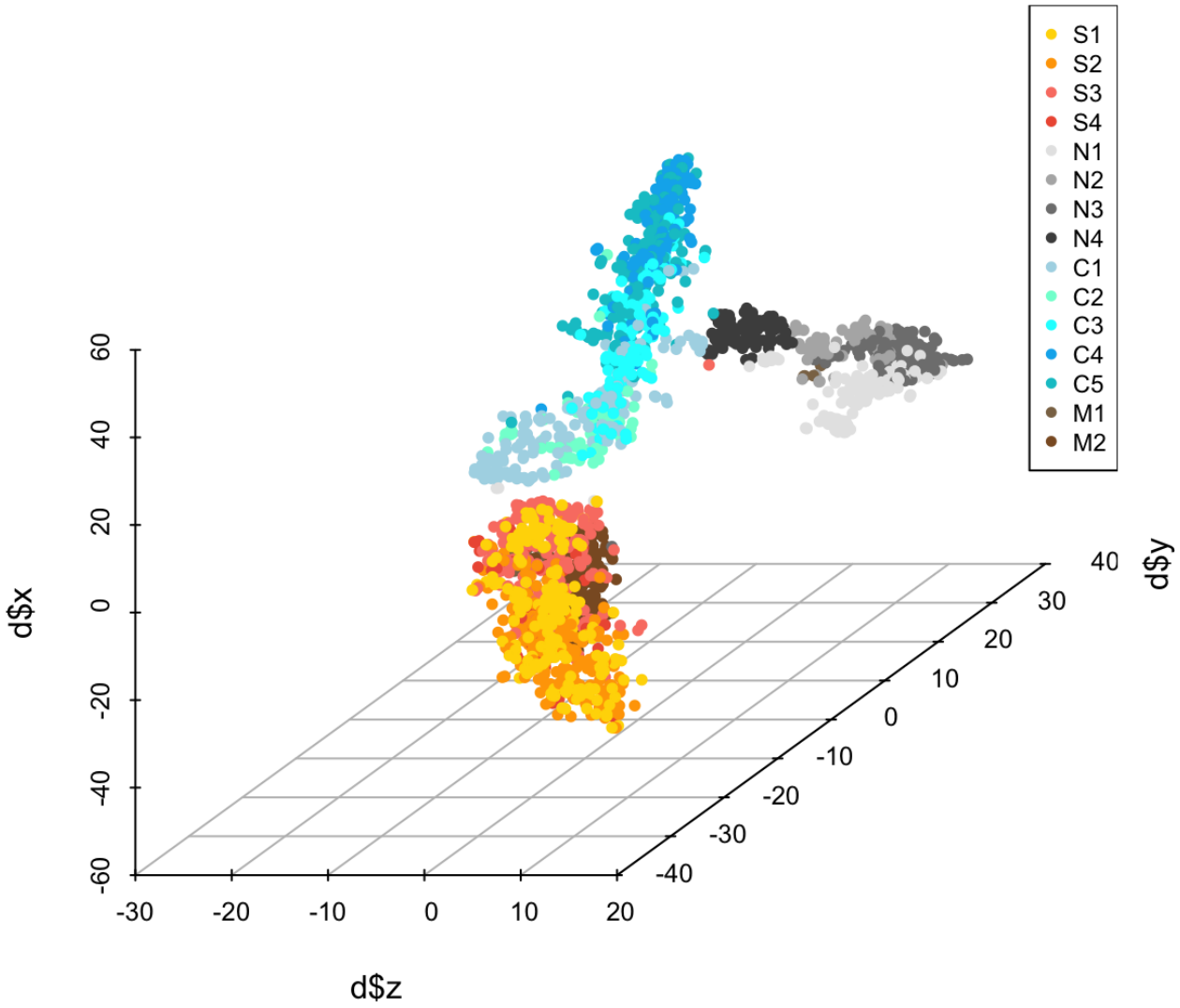
Although the applications of t-SNE is widespread in the field of machine learning, it can be misleading if it is not well optimized. Therefore, we optimize t-SNE map by grid search, a process that can take some time

```
set.seed(6)
tsne.map <- tsneVisualization(protein.data = df,
                    markerProteins = r.markers,
                    dims = 3,
                    theta = c(0.1),
                    perplexity = c(60))
```

```
## Optimization process started. This may take some time!
## Optimization was performed.
## Theta value:  0.1
## Perplexity value:  60
```



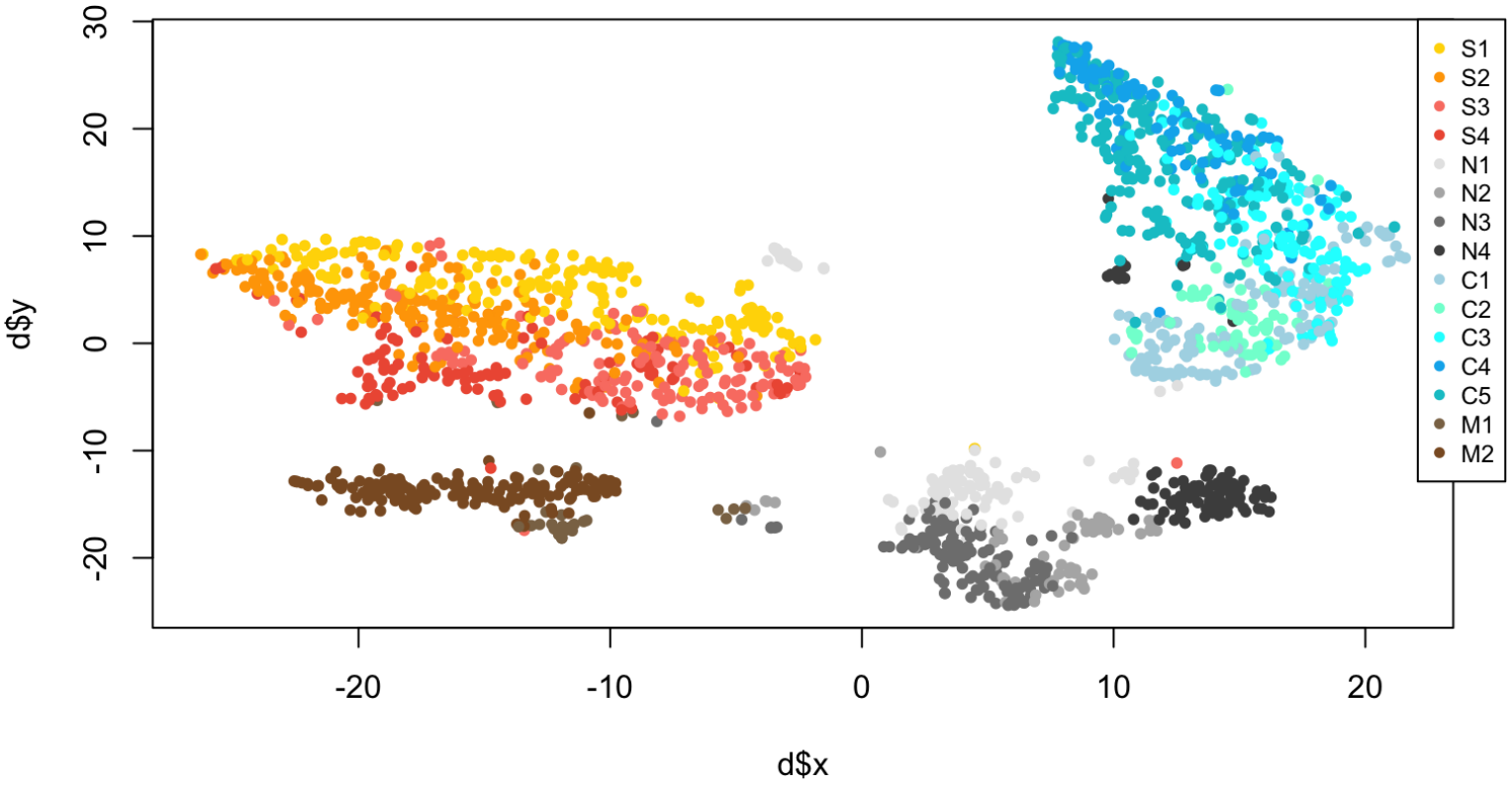

S1
S2
S3
S4
N1
N2
N3
N4
C1
C2
C3
C4
C5
M1
M2
d$z
d$x
d$y
20
10
0
-10
-20
-30
-60
-40
-20
0
20
40
60
40
30
20
10
0
-10
-20
-30
-40



We recommend 3D vizualisation by setting `dims = 3`, for optimal evaluation of marker protein cluster separation and data modularity. You can also visualize the marker proteins in 2 dimensional space by setting `dims = 2`, although reducing the dimensionality results in loss of information and underestimation of data resolution.

```
set.seed(9)
tsne.map2 <- tsneVisualization(protein.data = df,
                    markerProteins = r.markers,
                    dims = 2,
                    theta = c(0.5),
                    perplexity = c(60))
```

```
## Optimization process started. This may take some time!
## Optimization was performed.
## Theta value:  0.5
## Perplexity value:  60
```



3.7 Build model and classify proteins

We used the *e1071* package to build the classifier. It is important to note that, support-vector-machine classifier does not produce classprobability. To benefit from the class probability, we trained additional sigmoid function to map the SVM outputs into probabilities, after training the original classifier (Platt 2000). The method was embedded into the *e1071* R package which we used for the SVM classifier.

For replicate *A* and *B* separately, marker proteins are used for the training Support Vector Machine (SVM) classifier with a Gaussian radial basis function kernel algorithm. After tuning the parameters, the SVM model predicts (classifies) the subcellular localization for all proteins in the input data with corresponding probabilities for *A* and *B* replicate classification.

```
set.seed(4)
cls <- svmClassification(markerProteins = r.markers,
                                  protein.data = df,
                                  markerprot.df = markerProteins)
```

```
# testing data predictions for replicate A and B
test.A <- cls[[1]]$svm.test.prob.out
test.B <- cls[[2]]$svm.test.prob.out
head(test.A)
```

```
##         Observation svm.pred           S1          S2          S3          S4
## N4BP2L2          N2       N2 0.0034204982 0.002713603 0.003135572 0.002424749
## ZNF205           N1       N1 0.0539341925 0.005611837 0.009373929 0.003251983
## MAVS             M2       M2 0.0005904669 0.001188552 0.001075943 0.001988799
## NGLY1            C2       C2 0.0020717253 0.001835965 0.001706640 0.001777443
## RAB15            S2       S2 0.1005330801 0.850062711 0.017526771 0.022986723
## CHD4             N2       N2 0.0093310932 0.009050335 0.009640417 0.007007189
##                   N1           N2          N3           N4           C1
## N4BP2L2 0.0370000247 0.8956709656 0.026904787 0.0127955674 0.0026575959
## ZNF205  0.9047763896 0.0040535626 0.005468579 0.0024351136 0.0022360891
## MAVS    0.0009300487 0.0010955494 0.001556564 0.0010128144 0.0008742056
## NGLY1   0.0021911577 0.0017599204 0.001974421 0.0090865761 0.0608829293
## RAB15   0.0006195267 0.0006601397 0.000699539 0.0009793257 0.0009498403
```

```
## CHD4    0.0147735324 0.5904820545 0.016469668 0.3054409175 0.0060342981
##                   C2           C3           C4           C5           M1
## N4BP2L2 0.0020172675 0.0019633826 0.0022083395 0.0027406564 0.0026281681
## ZNF205  0.0014451171 0.0011720556 0.0013531775 0.0016422573 0.0016849212
## MAVS    0.0008075442 0.0006829592 0.0008801080 0.0010014552 0.0341655989
## NGLY1   0.8509162130 0.0393558124 0.0031946915 0.0198804447 0.0016605889
## RAB15   0.0008681537 0.0007352767 0.0007446856 0.0008892102 0.0009869426
## CHD4    0.0044846791 0.0035101058 0.0037620803 0.0055981694 0.0090452900
##                   M2
## N4BP2L2 0.0017188220
## ZNF205  0.0015607954
## MAVS    0.9521493909
## NGLY1   0.0017054720
## RAB15   0.0007580757
## CHD4    0.0053701723
```

We can investigate the models (classifiers for replicate A and B).
For the simplicity, the model for replicate A is shown.

Model summary:

```
model.A <- cls[[1]]$modelSummary
model.A
```

```
## svm(formula = factor(replicate.train.label) ~ ., data = rep.train.df,
##     probability = TRUE, kernel = "radial", cost = svm.tune$best.parameters[1],
##     gamma = svm.tune$best.parameters[2], cross = 10, scale = FALSE)
```

Optimized cost paramter:

```
cost <- cls[[1]]$modelCost
cost
```

```
## [1] 1
```

Optimized gamma paramter:

```
gamma <- cls[[1]]$modelGamma
gamma
```

```
## [1] 0.5
```

Training score (accuracy):

```
training <- cls[[1]]$modelAccuracy
training
```

```
## [1] 74.26471
```

Model evaluation on test data

```
confusionMatrixA <- caret::confusionMatrix(test.A$Observation,
                                           test.A$svm.pred)
```

Confussion Matrix

```
confusionMatrixA$table
```

```
##           Reference
## Prediction C1 C2 C3 C4 C5 M1 M2 N1 N2 N3 N4 S1 S2 S3 S4
##         C1 30  3  9  0  0  0  0  0  0  0  0  0  0  0  0
##         C2  4 14  0  0  2  0  0  0  0  0  0  0  0  0  0
##         C3  6  0 23  2  8  0  0  0  0  0  0  0  0  0  0
##         C4  1  0  1 14 19  0  0  0  0  0  0  0  0  0  0
##         C5  1  0  8  4 47  0  0  0  0  0  0  0  0  0  0
```

```
##         M1  0  0  0  0  0  6  1  0  0  0  0  0  0  0  2
##         M2  0  0  0  0  0  0 49  0  0  0  0  0  0  1  0
##         N1  0  0  0  0  0  0  0 29  0  2  0  0  0  0  0
##         N2  0  0  0  0  0  0  0  0 18  2  0  0  0  0  0
##         N3  0  0  0  0  0  0  0  2  1 36  0  0  0  1  0
##         N4  0  0  0  0  0  0  0  0  1  0 29  0  0  0  0
##         S1  0  0  0  0  0  0  0  0  0  0  0 48 11  2  0
##         S2  0  0  0  0  0  0  0  0  0  0  0  5 46  7  2
##         S3  0  0  0  0  0  0  0  0  0  0  0  5  4 35  2
##         S4  0  0  0  0  0  0  0  0  0  0  0  0  6  4 21
```

Precision, Recal anf F1 score for each compartment

```
confusionMatrixA$byClass
```

```
##           Sensitivity Specificity Pos Pred Value Neg Pred Value Precision
## Class: C1   0.7142857   0.9774436      0.7142857      0.9774436 0.7142857
## Class: C2   0.8235294   0.9892280      0.7000000      0.9945848 0.7000000
## Class: C3   0.5609756   0.9699812      0.5897436      0.9663551 0.5897436
## Class: C4   0.7000000   0.9620939      0.4000000      0.9888683 0.4000000
## Class: C5   0.6184211   0.9738956      0.7833333      0.9435798 0.7833333
## Class: M1   1.0000000   0.9947183      0.6666667      1.0000000 0.6666667
## Class: M2   0.9800000   0.9980916      0.9800000      0.9980916 0.9800000
## Class: N1   0.9354839   0.9963168      0.9354839      0.9963168 0.9354839
## Class: N2   0.9000000   0.9963899      0.9000000      0.9963899 0.9000000
## Class: N3   0.9000000   0.9925094      0.9000000      0.9925094 0.9000000
## Class: N4   1.0000000   0.9981651      0.9666667      1.0000000 0.9666667
## Class: S1   0.8275862   0.9748062      0.7868852      0.9805068 0.7868852
## Class: S2   0.6865672   0.9723866      0.7666667      0.9591440 0.7666667
## Class: S3   0.7000000   0.9790076      0.7608696      0.9715909 0.7608696
## Class: S4   0.7777778   0.9817185      0.6774194      0.9889503 0.6774194
##              Recall        F1 Prevalence Detection Rate Detection Prevalence
## Class: C1 0.7142857 0.7142857 0.07317073     0.05226481           0.07317073
## Class: C2 0.8235294 0.7567568 0.02961672     0.02439024           0.03484321
## Class: C3 0.5609756 0.5750000 0.07142857     0.04006969           0.06794425
## Class: C4 0.7000000 0.5090909 0.03484321     0.02439024           0.06097561
## Class: C5 0.6184211 0.6911765 0.13240418     0.08188153           0.10452962
## Class: M1 1.0000000 0.8000000 0.01045296     0.01045296           0.01567944
## Class: M2 0.9800000 0.9800000 0.08710801     0.08536585           0.08710801
## Class: N1 0.9354839 0.9354839 0.05400697     0.05052265           0.05400697
## Class: N2 0.9000000 0.9000000 0.03484321     0.03135889           0.03484321
## Class: N3 0.9000000 0.9000000 0.06968641     0.06271777           0.06968641
## Class: N4 1.0000000 0.9830508 0.05052265     0.05052265           0.05226481
## Class: S1 0.8275862 0.8067227 0.10104530     0.08362369           0.10627178
## Class: S2 0.6865672 0.7244094 0.11672474     0.08013937           0.10452962
## Class: S3 0.7000000 0.7291667 0.08710801     0.06097561           0.08013937
## Class: S4 0.7777778 0.7241379 0.04703833     0.03658537           0.05400697
##           Balanced Accuracy
## Class: C1         0.8458647
## Class: C2         0.9063787
## Class: C3         0.7654784
## Class: C4         0.8310469
## Class: C5         0.7961583
## Class: M1         0.9973592
## Class: M2         0.9890458
## Class: N1         0.9659003
## Class: N2         0.9481949
## Class: N3         0.9462547
```

```
## Class: N4         0.9990826
## Class: S1         0.9011962
## Class: S2         0.8294769
## Class: S3         0.8395038
## Class: S4         0.8797481
```

```
# all predictions for replicate A and B
all.A <- cls[[1]]$all.prot.pred
all.B <- cls[[2]]$all.prot.pred
```

3.7.1 Estimate classification thresholds for compartment level

Classification probabilities close to 1 indicate high confidence predictions, whereas probabilities close to 0 indicate low confidence predictions. To increase the overal prediction accuracy and to filter out poor predictions, one criterion and two cut-offs are defined.

1. The criterion is the consensus of preliminary predictions between biological duplicates. Proteins are kept in the analysis, if there is an agreement between biological duplicates. Subsequently, prediction probabilities from the two duplicates are averaged for each protein.
2. Cut-off is (precision - based) set when precision reach 0.9 in the test data.
3. Cut-off is (recall - based) set as the probability of the lowest true positive in the test data.

```
t.c.df <- computeThresholdCompartment(test.repA = test.A, test.repB = test.B)
```

```
head(t.c.df)
```

```
##   Compartment PrecisionBasedThreshold RecallBasedThreshold OptedThreshold
## 1          S1                    0.47                0.495          0.495
## 2          S2                   0.785                 0.51          0.785
## 3          S3                   0.535                0.375          0.535
## 4          S4                    0.68                0.445          0.680
## 5          N1                       0                 0.72          0.720
## 6          N2                       0                 0.53          0.530
```

3.7.2 Apply threshold to compartment level classifications

The determined thresholds for the compartment levels are applied to all classifications.

```
c.cls.df <- applyThresholdCompartment(all.repA = all.A, all.repB = all.B,
                                      threshold.df = t.c.df)
```

```
head(c.cls.df)
```

```
##         Proteins svm.pred        S1          S2         S3           S4
## FURIN      FURIN       S1 0.8044625 0.118285221 0.06148853 0.0039862411
## ATG9A      ATG9A       S1 0.8482675 0.013538605 0.01131488 0.0042270023
## DISC1      DISC1       S1 0.6003724 0.034584719 0.13175231 0.0231587190
## HVCN1      HVCN1       S1 0.9382960 0.005664071 0.01047046 0.0019191809
## VTI1A      VTI1A       S1 0.9802552 0.002545447 0.01070461 0.0004074506
## SLC35F2  SLC35F2       S1 0.7683913 0.019822117 0.16210238 0.0081724862
##                    N1           N2           N3           N4           C1
## FURIN   0.001063479 0.0011090607 0.0008779197 0.0010555088 0.0009208802
## ATG9A   0.085590105 0.0032762068 0.0045218381 0.0053467703 0.0066286150
## DISC1   0.045068474 0.0075255478 0.0069405674 0.0109770978 0.0939001191
## HVCN1   0.032281961 0.0013498275 0.0013276950 0.0014380712 0.0015681802
## VTI1A   0.001194382 0.0005728802 0.0005238379 0.0005018539 0.0005017726
## SLC35F2 0.017392373 0.0046300609 0.0025411416 0.0030319662 0.0041498757
##                    C2           C3           C4           C5           M1
## FURIN   0.0008966154 0.0007727800 0.0007533706 0.0009464182 0.0019590025
## ATG9A   0.0034716395 0.0024226870 0.0020141834 0.0035120378 0.0027487307
```

```
## DISC1   0.0175978565 0.0071947713 0.0035340674 0.0065186738 0.0044177332
## HVCN1   0.0010475977 0.0008637610 0.0008279222 0.0010973763 0.0008732059
## VTI1A   0.0005178166 0.0004366861 0.0004628942 0.0005308447 0.0004443072
## SLC35F2 0.0019535196 0.0013716268 0.0012046753 0.0017430045 0.0016115742
##                   M2
## FURIN   0.0014224819
## ATG9A   0.0031192271
## DISC1   0.0064569012
## HVCN1   0.0009747184
## VTI1A   0.0004000427
## SLC35F2 0.0018819501
```

3.7.3 Estimate classification thresholds for neighborhood level

Compartment level classification probabilities are summed to neighborhood probabilities and thresholds for neighborhood analysis are estimated as described above for compartment level analysis except precision based cut-off is set to 0.95.

```
t.n.df <- computeThresholdNeighborhood(test.repA = test.A, test.repB = test.B)
```

```
head(t.n.df)
```

```
##   Neighborhood PrecisionBasedThreshold RecallBasedThreshold OptedThreshold
## 1    Secretory                       0                 0.65          0.650
## 2      Nuclear                       0                 0.47          0.470
## 3      Cytosol                       0                 0.58          0.580
## 4 Mitochondria                       0                0.655          0.655
```

3.7.4 Apply threshold to neighborhood level classifications

The determined thresholds for the neighborhood levels are applied to all classifications.

```
n.cls.df <- applyThresholdNeighborhood(all.repA = all.A, all.repB = all.B,
                                       threshold.df = t.n.df)
```

```
head(n.cls.df)
```

```
##       Proteins svm.pred.all Secretory     Nuclear     Cytosol Mitochondria
## FURIN    FURIN    Secretory 0.9882225 0.004105968 0.004290065  0.003381484
## ATG9A    ATG9A    Secretory 0.8773480 0.098734920 0.018049163  0.005867958
## DISC1    DISC1    Secretory 0.7898682 0.070511687 0.128745488  0.010874634
## HVCN1    HVCN1    Secretory 0.9563497 0.036397555 0.005404837  0.001847924
## VTI1A    VTI1A    Secretory 0.9939127 0.002792954 0.002450014  0.000844350
## CEP68    CEP68    Secretory 0.9815079 0.006802636 0.007935830  0.003753659
```

3.7.5 Merge compartment and neighborhood classification

Individual classifications (compartment and neighborhood) are merged into one data frame.

```
cls.df <- mergeCls(compartmentCls = c.cls.df, neighborhoodCls = n.cls.df)
```

```
head(cls.df)
```

```
##        Protein NeighborhoodCls CompartmentCls   Secretory     Nuclear
## A2M        A2M         Cytosol   Unclassified 0.108974693 0.148019019
## A2ML1    A2ML1    Unclassified   Unclassified 0.289312766 0.106674311
## A4GALT  A4GALT       Secretory             S1 0.984063133 0.006326616
## AACS      AACS         Cytosol   Unclassified 0.007062643 0.009852283
## AAGAB    AAGAB         Cytosol   Unclassified 0.006046029 0.006551317
## AAK1      AAK1         Cytosol   Unclassified 0.005356143 0.006359080
##             Cytosol Mitochondria          S1          S2          S3          S4
## A2M    0.707476254  0.035530034 0.039866660 0.035645043 0.019382476 0.014080514
## A2ML1  0.313939713  0.290073210 0.068059420 0.052162470 0.104065175 0.065025702
## A4GALT 0.007018236  0.002592015 0.967094493 0.013246951 0.002326302 0.001395387
```

```
## AACS   0.979996757  0.003088316 0.002163661 0.001971189 0.001603501 0.001324293
## AAGAB  0.984621943  0.002780711 0.001702959 0.001659116 0.001446875 0.001237080
## AAK1   0.985783207  0.002501570 0.001509276 0.001434766 0.001288690 0.001123411
##                 N1          N2          N3          N4          C1          C2
## A2M    0.043316074 0.018332245 0.024952969 0.061417730 0.173374778 0.043123461
## A2ML1  0.028041362 0.013005800 0.017333591 0.048293558 0.107401033 0.127785177
## A4GALT 0.001279695 0.001378964 0.001444322 0.002223634 0.001873890 0.001430963
## AACS   0.002292800 0.001597228 0.001781031 0.004181224 0.004083417 0.002940375
## AAGAB  0.001966653 0.001460906 0.001589922 0.001533835 0.490438977 0.018607661
## AAK1   0.001677854 0.001288555 0.001389773 0.002002898 0.367893604 0.018527206
##                 C3          C4          C5          M1          M2
## A2M    0.120829788 0.209961620 0.160186606 0.013156336 0.022373699
## A2ML1  0.045365976 0.011057367 0.022330159 0.025059542 0.265013668
## A4GALT 0.001156778 0.001033019 0.001523587 0.001447960 0.001144055
## AACS   0.016174572 0.580101553 0.376696841 0.001440217 0.001648100
## AAGAB  0.444608775 0.009851353 0.021115177 0.001371630 0.001409080
## AAK1   0.557600944 0.011618468 0.030142985 0.001246429 0.001255142
```

4 Visualization of the protein subcellular localization

4.1 SubCellBarCode plot

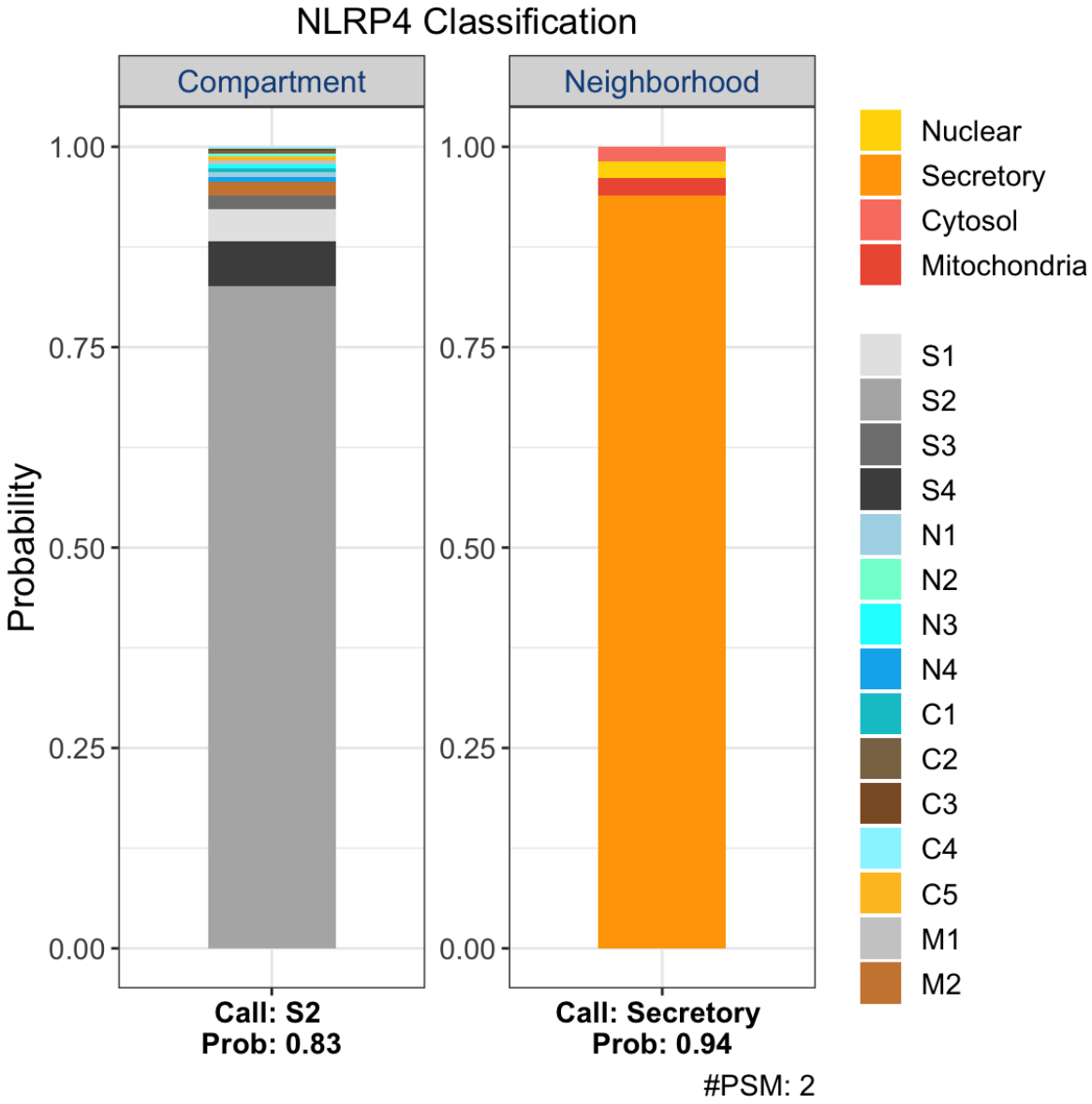
You can query one protein at a time to plot barcode of the protein of the interest.

`PSM` (Peptide-spectra-matching) count table is required for the plotting SubCellBarCode. It is in `data.frame` format;

```
head(hcc827CtrlPSMCount)
```

```
##        Protein PSMs.for.quant
## A2M        A2M             11
## A2ML1    A2ML1              2
## A4GALT  A4GALT              1
## AAAS      AAAS             33
## AACS      AACS             60
## AAED1    AAED1              9
```

```
plotBarcode(sampleClassification = cls.df, protein = "NLRP4",
        s1PSM = hcc827CtrlPSMCount)
```



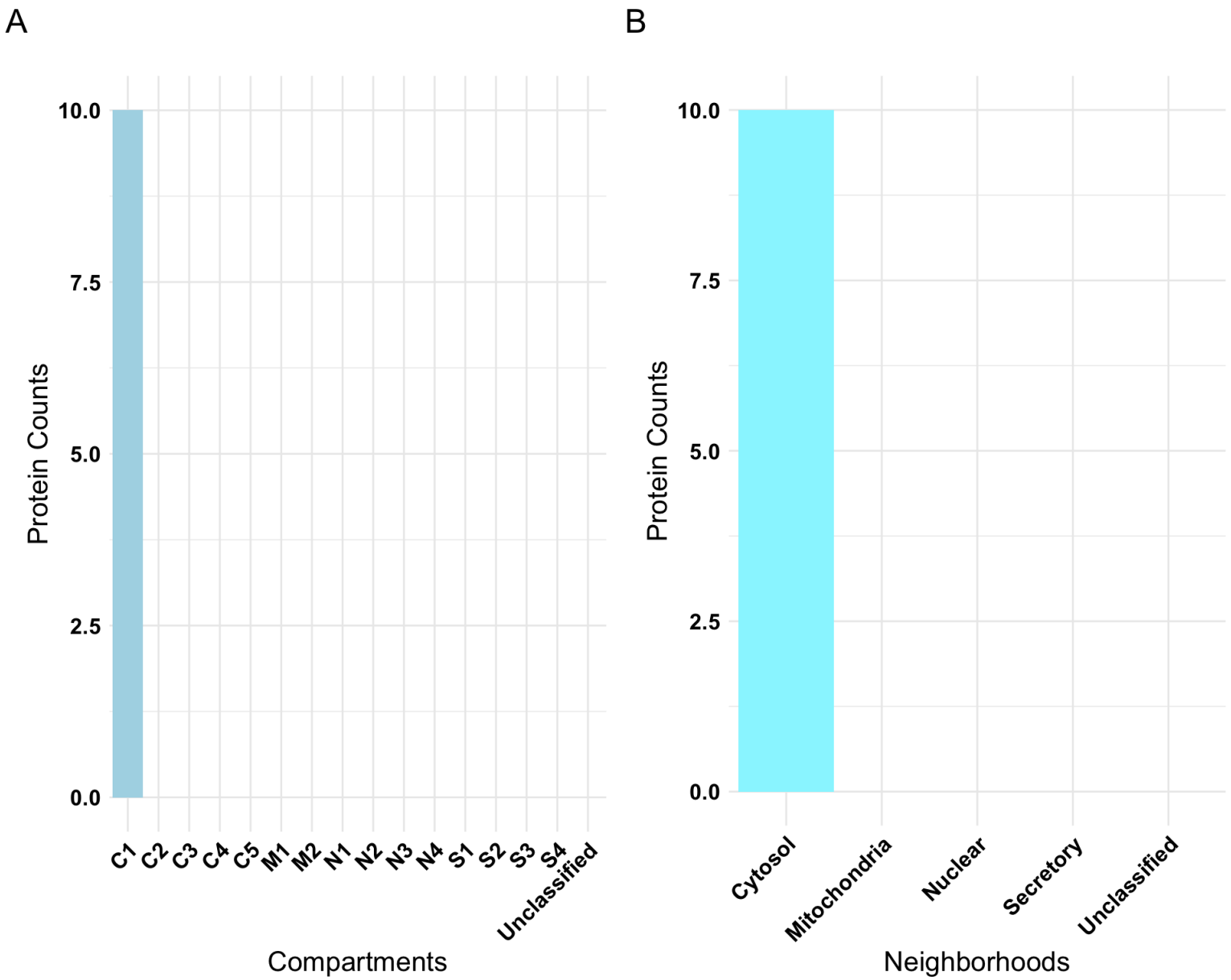
4.2 Co-localization plot

To evaluate localization and of multiple proteins at the same time, a vector of proteins (identified by gene symbols) can be prepared and used to create a barplot showing the distribution of classifications across compartments and neighborhoods. This analysis could be helpful when evaluating co-localization of proteins, protein complex formation and compartmentalized protein level regulation.

```
# 26S proteasome complex (26s proteasome regulatory complex)
proteasome26s <- c("PSMA7", "PSMC3","PSMA4", "PSMB4",
                   "PSMB6", "PSMB5", "PSMC2","PSMC4",
                   "PSMB3", "PSMA6","PSMC5","PSMC6")

plotMultipleProtein(sampleClassification = cls.df, proteinList = proteasome26s)
```

```
## PSMB4,PSMC6 not exist in data
```



5 Differential localization analysis

Regulation of protein localization is a the key process in cellular signalling. The `SubCellBarCode` method can be used for differential localization analysis given two conditions such as control vs treatment, cancer cell *vs* normal cell, cell state A *vs* cell state B, *etc*. As example, we compared untreated and gefitinib (EGFR inhibitor) treated HCC827 cells (for details, see Orre et al.).

5.1 Plot differentially localizing proteins

Neighborhood classifications for condition 1 (untreated) and condition 2 (gefitinib) is first done separately, and classifications for overlapping proteins are then vizualized by a sankey plot.
The HCC827 gefitinib cell lines classification was embedded into the package for example analysis.

```
head(hcc827GEFClass)
```

```
##         Protein NeighborhoodCls CompartmentCls
## A2M         A2M         Cytosol   Unclassified
## A2ML1     A2ML1    Unclassified   Unclassified
## A4GALT  A4GALT       Secretory             S1
## AAAS       AAAS       Secretory             S4
```

```
## AACS      AACS         Cytosol   Unclassified
## AAED1    AAED1       Secretory   Unclassified
```

```
sankeyPlot(sampleCls1 = cls.df, sampleCls2 = hcc827GEFClass)
```

```
##           Cond1        Cond2 value
## 1     Secretory    Secretory  1323
## 2     Secretory      Nuclear    19
## 3     Secretory      Cytosol    14
## 4     Secretory Mitochondria     9
## 5       Nuclear    Secretory    15
## 6       Nuclear      Nuclear  1263
## 7       Nuclear      Cytosol    23
## 8       Nuclear Mitochondria     0
## 9       Cytosol    Secretory    39
## 10      Cytosol      Nuclear    36
## 11      Cytosol      Cytosol  2019
## 12      Cytosol Mitochondria     1
## 13 Mitochondria    Secretory     5
## 14 Mitochondria      Nuclear     1
## 15 Mitochondria      Cytosol     1
## 16 Mitochondria Mitochondria   334
```

5.2 Filter Candidates

As the differential localization analysis is an outlier analysis, it will include analytical noise. To filter out such noise, `PSM` (Peptide-spectra-matching) counts and fractionation profile correlation analysis (Pearson) was done to identify strong candidates. The PSM count format for the input have to be the same between the compared conditions;

```
head(hcc827CtrlPSMCount)
```

```
##        Protein PSMs.for.quant
## A2M        A2M             11
## A2ML1    A2ML1              2
## A4GALT  A4GALT              1
## AAAS      AAAS             33
## AACS      AACS             60
## AAED1    AAED1              9
```

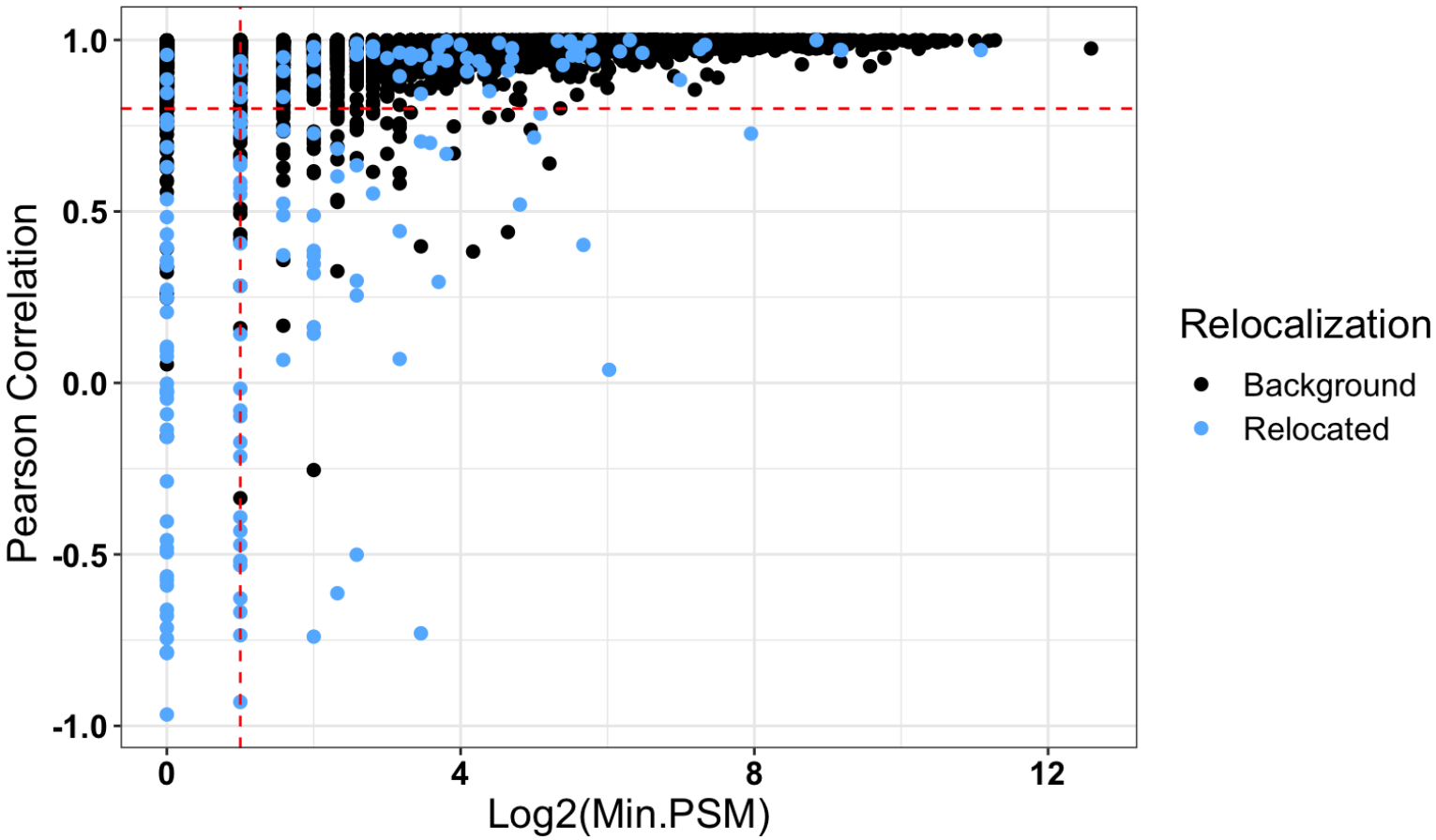
For each protein, the minimum PSM count between the two conditions is plotted against the fractionation profile (median) correlation between the two conditions. For proteins with different localizations between conditions, the fractionation profile differs and therefore we are expecting a low fractionation profile correlation. As a standard setting for filtering of analytical noise in the

differential localization analysis we suggest to demand a fractionation profile correlation below 0.8, and a minimum PSM count of at least 3.
However, for the exploratory analysis, you can adjust the `min.psm` and `pearson.cor` parameters.

```
##parameters
#sampleCls1 = sample 1 classification output
#s1PSM = sample 2 PSM count
#s1Quant = Sample 1 Quantification data
#sampleCls2 = sample 2 classification output
#s2PSM = sample 2 classification output
#sample2Quant = Sample 2 Quantification data
#min.psm = minumum psm count
#pearson.cor = perason correlation coefficient
```

```
candidate.df <- candidateRelocatedProteins(sampleCls1 = cls.df,
                                s1PSM = hcc827CtrlPSMCount,
                                s1Quant = hcc827Ctrl,
                                sampleCls2 = hcc827GEFClass,
                                s2PSM = hcc827GefPSMCount,
                                s2Quant = hcc827GEF,
                                min.psm = 2,
                                pearson.cor = 0.8)
```

```
## Warning: Use of `f.df$Min.PSMs` is discouraged. Use `Min.PSMs` instead.
```




```
cat(dim(candidate.df))
```

```
## 35 5
```

```
head(candidate.df)
```

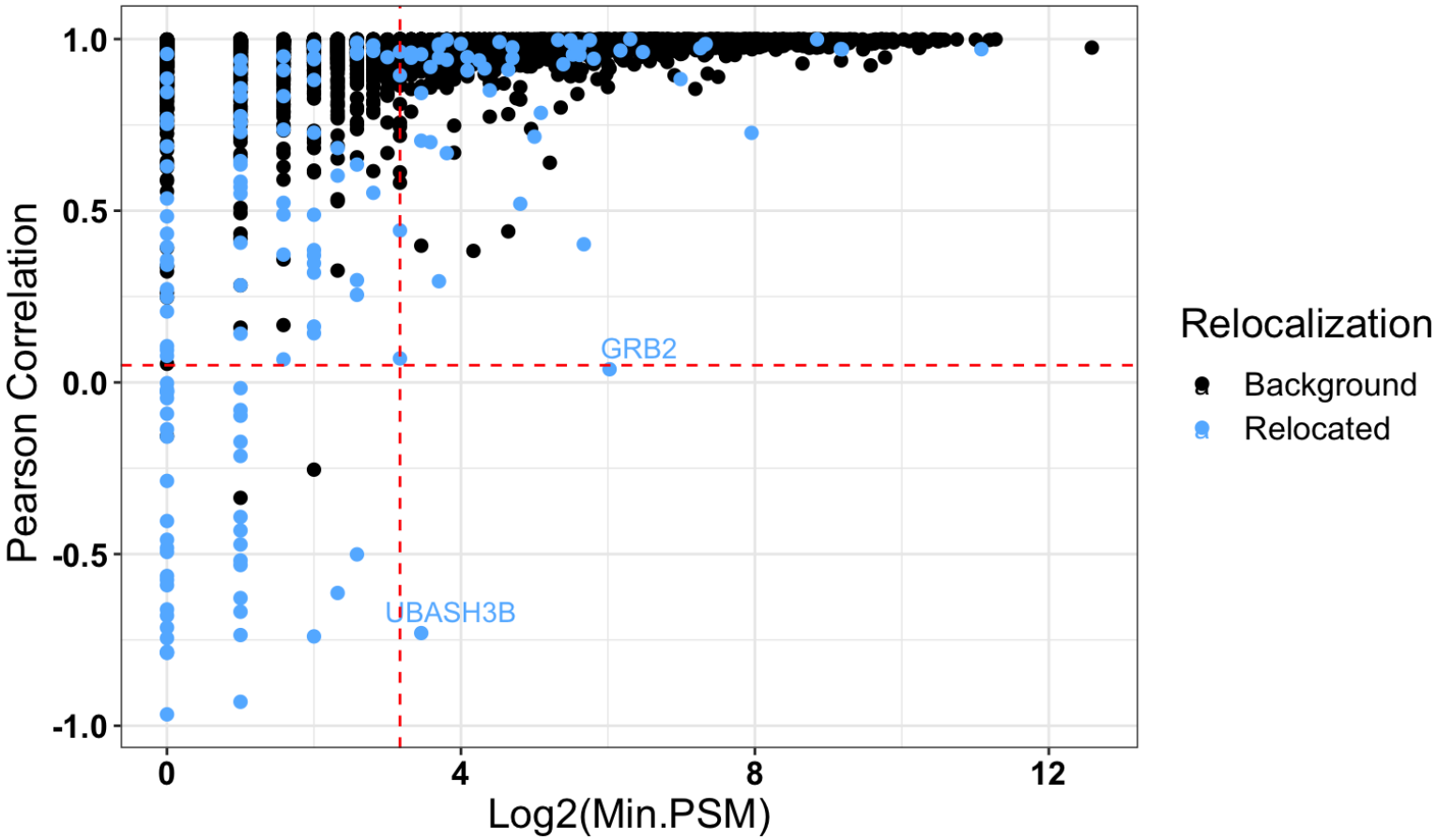
```
##         Protein       C.A       C.B Pearson.Corr Min.PSMs
## ACD         ACD   Cytosol   Nuclear    0.7274418        4
## AGAP2     AGAP2   Nuclear   Cytosol    0.4888424        3
```

```
## ANKEF1  ANKEF1   Cytosol Secretory    0.5230030        3
## BATF3    BATF3 Secretory   Cytosol   -0.6132878        5
## CGNL1    CGNL1   Cytosol   Nuclear    0.7850309       34
## CRY1      CRY1   Nuclear Secretory   -0.7397292        4
```

Candidate subset of differentially localizing proteins can be annotated with names by setting `annotation = TRUE`, `min.psm` and `pearson.cor`

```
candidate2.df <- candidateRelocatedProteins(sampleCls1 = cls.df,
                                s1PSM = hcc827CtrlPSMCount,
                                s1Quant = hcc827Ctrl,
                                sampleCls2 = hcc827GEFClass,
                                s2PSM = hcc827GefPSMCount,
                                s2Quant = hcc827GEF,
                                annotation = TRUE,
                                min.psm = 9,
                                pearson.cor = 0.05)
```

```
## Warning: Use of `annot.df$Min.PSMs` is discouraged. Use `Min.PSMs` instead.
## Warning: Use of `annot.df$Label` is discouraged. Use `Label` instead.
## Warning: Use of `annot.df$Min.PSMs` is discouraged. Use `Min.PSMs` instead.
## Warning: Use of `annot.df$Label` is discouraged. Use `Label` instead.
```



6 Peptide/Exon/Transcript centric or PTM regulated localization

Our main analysis is based on gene-centric. However, based on the analytical depth of the data, peptide/exon/transcript centric classifications can be performed. Moreover, this approach is also applicable for post translation modification (PTM) enriched data. Fundamentally, we will use the same classifiers, compartment and neighborhood levels thresholds and apply them to peptide/exon/transcript data.

6.1 Exon-centric classification

Here, we have provided subset of the HCC827 exon-centric data.

```
head(hcc827exon)
```

```
##                 Gene_Symbol FS1.A.HCC827 FS1.B.HCC827 FS2.A.HCC827 FS2.B.HCC827
## ENSE00000331191       CDC45    3.0054124    3.1763997    1.5782291    1.6024560
## ENSE00000331854       COPB1    2.0449415    2.0610331    1.1743850    1.1716880
## ENSE00000331855       COPB1    1.8797670    1.5457263    1.0152127    1.0173337
## ENSE00000331859       COPB1    1.7921392    2.0895850    1.0261406    1.0273679
## ENSE00000331990       LPIN1    2.4724838    2.7718197    1.4581675    1.7319290
## ENSE00000332881       PTPRA    0.4461752    0.6117839    0.5241694    0.3844395
##                 FP1.A.HCC827 FP1.B.HCC827 FP2.A.HCC827 FP2.B.HCC827
## ENSE00000331191    0.9467013    0.9600072    0.5214538    0.6500854
## ENSE00000331854    0.6919514    0.7403220    1.0573085    1.0332836
## ENSE00000331855    0.8060150    0.8152469    1.2035301    1.1714791
## ENSE00000331859    0.7443903    0.8078896    1.2703089    1.1974385
## ENSE00000331990    0.6075537    0.7533756    0.8890101    0.9232132
## ENSE00000332881    1.3494955    1.2581107    1.7331063    1.6535322
##                 FP3.A.HCC827 FP3.B.HCC827 PeptideCount
## ENSE00000331191    0.6017849    0.6252090            1
## ENSE00000331854    1.1354910    1.1858233            5
## ENSE00000331855    1.2002348    1.1099946            1
## ENSE00000331859    1.1113409    1.0545062            1
## ENSE00000331990    0.7477472    0.8117479            1
## ENSE00000332881    1.0713874    1.2001535            3
```

For the classification, we will use the gene-centric model that we built in section 3.7.

```
##recall the models
modelA <- cls[[1]]$model
modelB <- cls[[2]]$model
```

`svmExternalData` *function greps replicates A and B, repectively. So the input data can include other features like, here, peptide counts.*

```
exon.cls <- svmExternalData(df = hcc827exon, modelA = modelA, modelB= modelB)
```

```
exon.A <- exon.cls[[1]]
exon.B <- exon.cls[[2]]
```

```
head(exon.A)
```

```
##                 Gene_Symbol svm.pred.all          S1          S2          S3
## ENSE00000331191       CDC45           C5 0.005209818 0.004428667 0.003555885
## ENSE00000331854       COPB1           C1 0.002094528 0.001549502 0.001638616
## ENSE00000331855       COPB1           C1 0.004767946 0.002673107 0.004305875
## ENSE00000331859       COPB1           C2 0.005863040 0.003269242 0.005165852
## ENSE00000331990       LPIN1           C1 0.001952427 0.001807642 0.001534898
## ENSE00000332881       PTPRA           S4 0.005234257 0.041896085 0.382199347
##                           S4          N1          N2          N3          N4
## ENSE00000331191 0.003242003 0.005011824 0.003435750 0.004037131 0.014660840
## ENSE00000331854 0.001500378 0.002987032 0.001428675 0.001654051 0.003863658
## ENSE00000331855 0.003354011 0.004539758 0.001967439 0.002153011 0.006044831
## ENSE00000331859 0.003993647 0.005077045 0.002213735 0.002507557 0.008619431
## ENSE00000331990 0.001507256 0.002226228 0.001618174 0.001889669 0.008721204
## ENSE00000332881 0.431742588 0.002736390 0.007376895 0.003353097 0.002177273
##                           C1          C2          C3          C4          C5
## ENSE00000331191 0.029431200 0.02739400 0.104285450 0.079949261 0.708536210
```

```
## ENSE00000331854 0.545978939 0.38805113 0.041528864 0.002229591 0.002992308
## ENSE00000331855 0.512303625 0.38701929 0.059858677 0.002931555 0.004412370
## ENSE00000331859 0.394845949 0.48590743 0.069114609 0.003397616 0.005398430
## ENSE00000331990 0.495706311 0.20133219 0.223907357 0.010549736 0.044135377
## ENSE00000332881 0.001670115 0.00159227 0.001496738 0.001614183 0.001821697
##                           M1          M2
## ENSE00000331191 0.002993919 0.003828040
## ENSE00000331854 0.001215528 0.001287200
## ENSE00000331855 0.001570441 0.002098061
## ENSE00000331859 0.001887038 0.002739378
## ENSE00000331990 0.001460832 0.001650700
## ENSE00000332881 0.071655946 0.043433117
```

Next steps are exactly same as what we did in section 3.7. We will use same thresholdsthat we have estimated in 3.7.1 and 3.7.3. Finally, we will merge two classifications same function used in 3.7.5.

```
exon.comp.cls <- applyThresholdCompartment(all.repA = exon.A[,2:17],
                                   all.repB = exon.B[,2:17],
                                   threshold.df = t.c.df)

exon.neigh.cls <- applyThresholdNeighborhood(all.repA = exon.A[,2:17],
                                   all.repB = exon.B[,2:17],
                                   threshold.df = t.n.df)

exon.cls.df <- mergeCls(compartmentCls = exon.comp.cls,
                        neighborhoodCls = exon.neigh.cls)

#same order
exon.cls.df <- exon.cls.df[rownames(exon.A),]

# we will add gene symbols as well as peptide count
# (PSM count is also accepted) in case for comparing with
# gene-centric classifications

exon.cls.df$GeneSymbol <- exon.A$Gene_Symbol
exon.cls.df$PeptideCount <- hcc827exon$PeptideCount
```

```
head(exon.cls.df)
```

```
##                          Protein NeighborhoodCls CompartmentCls   Secretory
## ENSE00000331191 ENSE00000331191         Cytosol   Unclassified 0.014832469
## ENSE00000331854 ENSE00000331854         Cytosol   Unclassified 0.009278669
## ENSE00000331855 ENSE00000331855         Cytosol   Unclassified 0.020407456
## ENSE00000331859 ENSE00000331859         Cytosol   Unclassified 0.017173531
## ENSE00000331990 ENSE00000331990         Cytosol   Unclassified 0.012312932
## ENSE00000332881 ENSE00000332881       Secretory   Unclassified 0.916146239
##                      Nuclear     Cytosol Mitochondria          S1          S2
## ENSE00000331191 0.02279522 0.956085749  0.006286562 0.004519423 0.004117834
## ENSE00000331854 0.01318771 0.973521876  0.004011749 0.002772796 0.002635333
## ENSE00000331855 0.02129554 0.952790918  0.005506091 0.006079970 0.004366019
## ENSE00000331859 0.01947592 0.957615992  0.005734562 0.005211156 0.004031744
## ENSE00000331990 0.01937522 0.962862663  0.005449185 0.003536029 0.003621272
## ENSE00000332881 0.01224056 0.007886533  0.063726674 0.003485418 0.048403049
##                           S3          S4          N1          N2          N3
## ENSE00000331191 0.003371211 0.002824002 0.004648321 0.003236997 0.003644227
## ENSE00000331854 0.002135101 0.001735439 0.004252239 0.002268855 0.002583707
## ENSE00000331855 0.005846533 0.004114934 0.006833201 0.002990136 0.003239354
## ENSE00000331859 0.004496494 0.003434136 0.005684534 0.003019969 0.003403090
## ENSE00000331990 0.002855838 0.002299793 0.004253858 0.002914563 0.003272068
```

```
## ENSE00000332881 0.211302289 0.652955482 0.002453825 0.005093238 0.002598370
##                          N4          C1          C2          C3          C4
## ENSE00000331191 0.011265675 0.047842854 0.024531307 0.266857757 0.060419555
## ENSE00000331854 0.004082905 0.525834771 0.396835016 0.042711381 0.002802567
## ENSE00000331855 0.008232845 0.374392624 0.513289731 0.054718728 0.003373434
## ENSE00000331859 0.007368322 0.436126244 0.410295371 0.098762837 0.004447041
## ENSE00000331990 0.008934731 0.479872458 0.212323124 0.188181776 0.011141407
## ENSE00000332881 0.002095123 0.001659707 0.001598117 0.001446198 0.001439449
##                          C5          M1          M2 GeneSymbol PeptideCount
## ENSE00000331191 0.556434275 0.002860280 0.003426282      CDC45            1
## ENSE00000331854 0.005338141 0.002005498 0.002006251      COPB1            5
## ENSE00000331855 0.007016401 0.002588313 0.002917777      COPB1            1
## ENSE00000331859 0.007984499 0.002584865 0.003149697      COPB1            1
## ENSE00000331990 0.071343897 0.002559130 0.002890055      LPIN1            1
## ENSE00000332881 0.001743062 0.039156448 0.024570225      PTPRA            3
```

6.2 Comparison between gene and exon centric classification

The insteresting part of the exon classification is to detect deviated classification between gene centric classification. However, we enrich more noise than gene centric classification due to analtyical dept of the exon centric data. Therefore, we, additioanlly, internally calculate correlation between gene-centric data `hcc827Ctrl` and exon-centric data `hcc827exon`, and report number of peptides (PSM count works, too) that match to corresponding exon as an indication of the confidence of the results.

```
comp.df <- compareCls(geneCls = cls.df, exonCls = exon.cls.df)
```

```
head(comp.df)
```

```
##           Exon_id GeneSymbol Gene_Neighborhood Exon_Neighborhood
## 1 ENSE00000400044      ABCF2           Nuclear           Nuclear
## 2 ENSE00000400045      ABCF2           Nuclear           Cytosol
## 3 ENSE00000400052      ABCF2           Nuclear           Cytosol
## 4 ENSE00000400054      ABCF2           Nuclear           Nuclear
## 5 ENSE00000653853      ACIN1           Nuclear           Nuclear
## 6 ENSE00000655866       ACO2      Mitochondria      Mitochondria
##   Gene_Compartment Exon_Compartment PeptideCount Pearson.Corr
## 1               N4               N4            2    0.9632687
## 2               N4               C2            6    0.8533751
## 3               N4               C2            1    0.2251107
## 4               N4               N4            2    0.9894984
## 5               N2               N2           21    0.9903756
## 6               M1               M1            2    0.9323598
```

You can easily vizualize/filter the results using correlations and number of peptides.

7 References

- Orre., et al. "SubCellBarCode: Proteome-wide Mapping of Protein Localization and Relocalization." Molecular Cell (2019): 73(1):166-182.e7.

8 Session Information

```
sessionInfo()
```

```
## R version 4.0.0 (2020-04-24)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Catalina 10.15.7
```

```
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] BiocStyle_2.16.1     SubCellBarCode_1.0.0
##
## loaded via a namespace (and not attached):
##   [1] nlme_3.1-152         fs_1.5.0             usethis_2.0.1
##   [4] lubridate_1.7.10     devtools_2.4.1       bit64_4.0.5
##   [7] rprojroot_2.0.2      bslib_0.2.5.1        tools_4.0.0
##  [10] utf8_1.2.1           R6_2.5.0             rpart_4.1-15
##  [13] BiocGenerics_0.36.0  DBI_1.1.1            colorspace_2.0-1
##  [16] nnet_7.3-16          withr_2.4.2          tidyselect_1.1.1
##  [19] gridExtra_2.3        prettyunits_1.1.1    processx_3.5.2
##  [22] bit_4.0.4            compiler_4.0.0       Biobase_2.50.0
##  [25] cli_2.5.0            xml2_1.3.2           desc_1.3.0
##  [28] labeling_0.4.2       sass_0.4.0           bookdown_0.22
##  [31] scales_1.1.1         callr_3.7.0          proxy_0.4-25
##  [34] stringr_1.4.0        digest_0.6.27        rmarkdown_2.8
##  [37] pkgconfig_2.0.3      htmltools_0.5.1.1    sessioninfo_1.1.1
##  [40] highr_0.9            fastmap_1.1.0        htmlwidgets_1.5.3
##  [43] rlang_0.4.11         rstudioapi_0.13      RSQLite_2.2.7
##  [46] farver_2.1.0         jquerylib_0.1.4      generics_0.1.0
##  [49] jsonlite_1.7.2       dplyr_1.0.6          ModelMetrics_1.2.2.2
##  [52] magrittr_2.0.1       Matrix_1.3-3         S4Vectors_0.28.0
##  [55] Rcpp_1.0.6           munsell_0.5.0        fansi_0.5.0
##  [58] lifecycle_1.0.0      yaml_2.2.1           scatterplot3d_0.3-41
##  [61] stringi_1.6.2        pROC_1.17.0.1        MASS_7.3-54
##  [64] org.Hs.eg.db_3.11.4  pkgbuild_1.2.0       Rtsne_0.15
##  [67] plyr_1.8.6           recipes_0.1.16       grid_4.0.0
##  [70] blob_1.2.1           parallel_4.0.0       ggrepel_0.9.1
##  [73] crayon_1.4.1         lattice_0.20-44      splines_4.0.0
##  [76] magick_2.7.2         knitr_1.33           ps_1.6.0
##  [79] pillar_1.6.1         igraph_1.2.6         reshape2_1.4.4
##  [82] codetools_0.2-18     stats4_4.0.0         pkgload_1.2.1
##  [85] glue_1.4.2           evaluate_0.14        BiocManager_1.30.15
##  [88] data.table_1.14.0    remotes_2.3.0        vctrs_0.3.8
##  [91] foreach_1.5.1        testthat_3.0.2       networkD3_0.4
##  [94] gtable_0.3.0         purrr_0.3.4          assertthat_0.2.1
##  [97] cachem_1.0.5         ggplot2_3.3.3        xfun_0.23
## [100] gower_0.2.2          prodlim_2019.11.13   e1071_1.7-7
## [103] roxygen2_7.1.1       class_7.3-19         survival_3.2-11
## [106] timeDate_3043.102    tibble_3.1.2         iterators_1.0.13
## [109] IRanges_2.24.0       AnnotationDbi_1.52.0 memoise_2.0.0
## [112] lava_1.6.9           ellipsis_0.3.2       caret_6.0-88
## [115] ipred_0.9-11
```