Tutorial 5: Using five classical fractionation channels on CPA

DFM & PL

2022-03-12

Contents

Introduction and creation of five-channel data

In previous tutorials we investigated the use of all nine TMT-MS2 fractions when performing CPA. In this tutorial, we investigate the performance of CPA using only the five classical differential centrifugation fractions N, M, L, P, and S. We derive the five classical fractions by removing the Nyc fractions and combining L1 and L2 to make the L fraction. We load the `protlocassign` fraction and summarize the full nine-fraction data as follows:

```
library(protlocassign)
data(protNSA_test)
dim(protNSA_test)
```

```
#> [1] 40 11
```

Next, we convert the data from the original NSA data to the Acup-transformed data, which is the appropriate transformation for combining fractions.

```
data(totProtAT5)
protAcup_test_data <- AcupFromNSA(protNSA_test[,1:6],
                                  NstartMaterialFractions=6,
                                  totProt=totProtAT5[1:6])
```

We complete the conversion to five-channel data as follows:

```
# take the first six fractions:
protAcupSixTemp <- protAcup_test_data[,1:6]
# add L1 and L2 together, and put into L1; then re-name L1 as L
# make L1 the sum of L1 and L2:
protAcupFiveTemp <- within(protAcupSixTemp, L1 <- L1+L2)
# L is now the name of third columm:
names(protAcupFiveTemp)[3] <- "L"
# drop L2 column (column 4)
protAcup <- protAcupFiveTemp[, -4]
head(round(protAcup,3))
```

```
#>              N     M     L     P     S
#> ACP2     0.246 0.282 0.031 0.125 0.316
#> AIF1        NA    NA    NA    NA    NA
#> ATP1A1   0.403 0.235 0.026 0.215 0.121
#> B4GALT1  0.262 0.167 0.026 0.545 0.000
#> CANX     0.293 0.187 0.029 0.420 0.072
#> CAT      0.211 0.327 0.055 0.074 0.333
```

We also must create a vector `totProt` with five values, corresponding to N, M, L (=L1+L2), P, S. Here are the original values of total protein in the nine channels:

```
round(totProtAT5, 3)
```

```
#>      N      M     L1     L2      P      S  Nyc.1  Nyc.2  Nyc.3
#> 46.045 48.956  1.384  1.566 24.046 58.182  0.037  0.068  1.273
```

Now we complete the calculation:

```
totProtTemp <- totProtAT5
totProtTemp[3] <- totProtAT5[3] + totProtAT5[4]
names(totProtTemp)[3] <- "L"
# remove L2 (component 4) and Nyc values (columns 7, 8, 9)
totProt <- totProtTemp[c(-4, -7, -8, -9)]
round(totProt, 3)
```

```
#>      N      M      L      P      S
#> 46.045 48.956  2.950 24.046 58.182
```

Transformations of the five-channel data and reference profiles

Next we can convert the five-channel Acup data to NSA and also to RSA:

```
protNSA<- NSAfromAcup(Acup=protAcup, NstartMaterialFractions=5,
                      totProt=totProt)
head(round(protNSA,3))
```

```
#>              N     M     L     P     S
#> ACP2     0.165 0.178 0.328 0.161 0.168
#> AIF1        NA    NA    NA    NA    NA
```

```
#> ATP1A1  0.263 0.144 0.263 0.268 0.063
#> B4GALT1 0.140 0.084 0.215 0.560 0.000
#> CANX    0.164 0.099 0.255 0.451 0.032
#> CAT     0.119 0.173 0.480 0.080 0.148
```

```
protRSA <- RSAfromNSA(NSA=protNSA,NstartMaterialFractions = 5,
                      totProt = totProt)
```

We may obtain the three transformations of the reference profiles as follows:

```
data(markerListJadot)
refLocationProfilesNSA <- locationProfileSetup(profile=protNSA,
                            markerList=markerListJadot, numDataCols=5)
refLocationProfilesAcup <- AcupFromNSA(refLocationProfilesNSA,
                            NstartMaterialFractions=5, totProt=totProt)
refLocationProfilesRSA <- RSAfromNSA(refLocationProfilesNSA,
                            NstartMaterialFractions=5, totProt=totProt)

round(refLocationProfilesNSA, 3)
```

```
#>           N     M     L     P     S
#> Cyto  0.115 0.096 0.109 0.077 0.603
#> ER    0.156 0.113 0.266 0.407 0.058
#> Golgi 0.152 0.102 0.144 0.554 0.048
#> Lyso  0.157 0.189 0.332 0.154 0.168
#> Mito  0.209 0.463 0.183 0.071 0.075
#> Nuc   0.844 0.036 0.029 0.043 0.047
#> Perox 0.126 0.165 0.493 0.092 0.124
#> PM    0.268 0.149 0.235 0.261 0.088
```

```
round(refLocationProfilesRSA, 3)
```

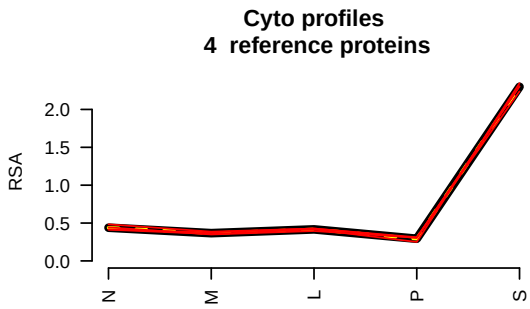
```
#>           N     M     L     P     S
#> Cyto  0.439 0.367 0.416 0.293 2.299
#> ER    1.052 0.765 1.793 2.749 0.393
#> Golgi 0.962 0.642 0.907 3.496 0.305
#> Lyso  0.916 1.098 1.930 0.898 0.979
#> Mito  0.968 2.145 0.848 0.328 0.347
#> Nuc   3.418 0.145 0.119 0.174 0.192
#> Perox 0.915 1.203 3.582 0.672 0.901
#> PM    1.522 0.846 1.337 1.483 0.499
```

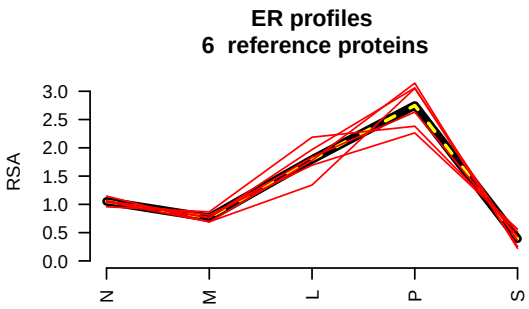
```
round(refLocationProfilesAcup, 3)
```

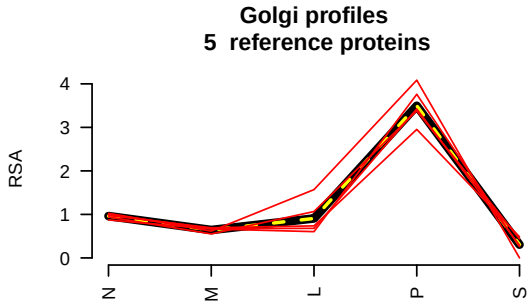
```
#>           N     M     L     P     S
#> Cyto  0.112 0.100 0.007 0.039 0.742
#> ER    0.269 0.208 0.029 0.367 0.127
#> Golgi 0.246 0.174 0.015 0.467 0.098
#> Lyso  0.234 0.298 0.032 0.120 0.316
#> Mito  0.247 0.583 0.014 0.044 0.112
#> Nuc   0.874 0.039 0.002 0.023 0.062
#> Perox 0.234 0.327 0.059 0.090 0.291
#> PM    0.389 0.230 0.022 0.198 0.161
```

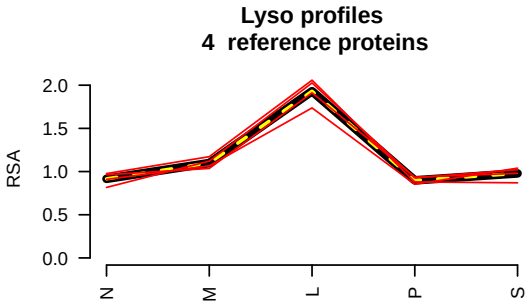
We then generate profile plots:

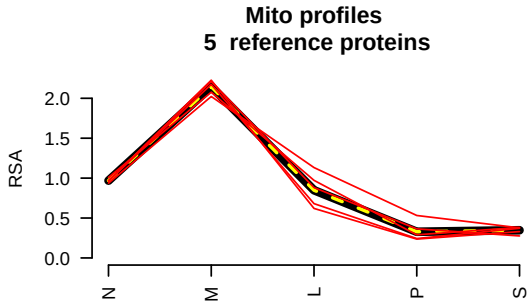
```
loc.list <- rownames(refLocationProfilesRSA)
n.loc <- length(loc.list)
par(mfrow=c(4,2))
for (i in 1:n.loc) {
  markerProfilePlot(refLoc=loc.list[i], profile=protRSA,
                    markerList=markerListJadot,
                    refLocationProfiles=refLocationProfilesRSA,
                    ylab="RSA")
}
```

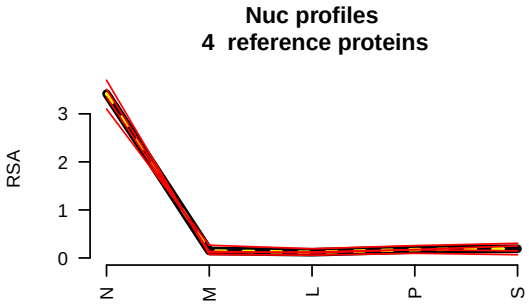

Cyto profiles
4 reference proteins
RSA
2.0
1.5
1.0
0.5
0.0
N
M
L
P
S

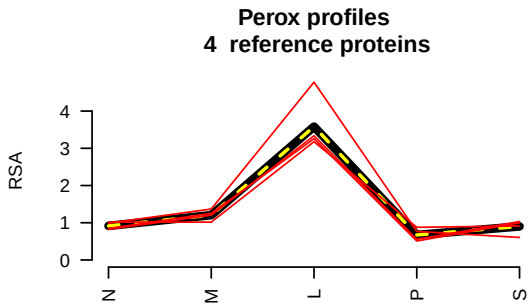

ER profiles
6 reference proteins
RSA
3.0
2.5
2.0
1.5
1.0
0.5
0.0
N
M
L
P
S

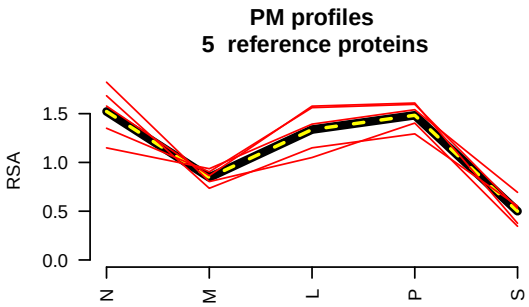

Golgi profiles
5 reference proteins
RSA
4
3
2
1
0
N
M
L
P
S


Lyso profiles
4 reference proteins
RSA
2.0
1.5
1.0
0.5
0.0
N
M
L
P
S


Mito profiles
5 reference proteins
RSA
2.0
1.5
1.0
0.5
0.0
N
M
L
P
S


Nuc profiles
4 reference proteins
RSA
3
2
1
0
N
M
L
P
S


Perox profiles
4 reference proteins
RSA
4
3
2
1
0
N
M
L
P
S


PM profiles
5 reference proteins
RSA
1.5
1.0
0.5
0.0
N
M
L
P
S

Mixture simulations using five-channel data

One can generate any desired mixture and transformation by adjusting values of i and j.

```
i=1  #Cyto
j=2  #ER
mixProtiProtj12Acup <- proteinMix(refLocationProfilesAcup, Loc1=i, Loc2=j)
mixProtiProtj12RSA <- RSAfromAcup(Acup=mixProtiProtj12Acup,
                            NstartMaterialFractions=5, totProt=totProt)
mixProtiProtjCPAfromRSA12 <- fitCPA(profile=mixProtiProtj12RSA,
                            refLocationProfiles=refLocationProfilesRSA,
                            numDataCols=5)

i=1  #Cyto
j=3  #Golgi
mixProtiProtj13Acup <- proteinMix(refLocationProfilesAcup, Loc1=i, Loc2=j)
mixProtiProtj13RSA <- RSAfromAcup(Acup=mixProtiProtj13Acup,
                                NstartMaterialFractions=5, totProt=totProt)
mixProtiProtjCPAfromRSA13 <- fitCPA(profile=mixProtiProtj13RSA,
                                refLocationProfiles=refLocationProfilesRSA,
                                numDataCols=5)
i=1  #Cyto
j=4  #Lyso
mixProtiProtj14Acup <- proteinMix(refLocationProfilesAcup, Loc1=i, Loc2=j)
mixProtiProtj14RSA <- RSAfromAcup(Acup=mixProtiProtj14Acup,
                            NstartMaterialFractions=5, totProt=totProt)
mixProtiProtjCPAfromRSA14 <- fitCPA(profile=mixProtiProtj14RSA,
                                refLocationProfiles=refLocationProfilesRSA,
                                numDataCols=5)

i=1  #Cyto
j=5  #Mito
mixProtiProtj15Acup <- proteinMix(refLocationProfilesAcup, Loc1=i, Loc2=j)
mixProtiProtj15RSA <- RSAfromAcup(Acup=mixProtiProtj15Acup,
                            NstartMaterialFractions=5, totProt=totProt)
mixProtiProtjCPAfromRSA15 <- fitCPA(profile=mixProtiProtj15RSA,
                                refLocationProfiles=refLocationProfilesRSA,
                                numDataCols=5)

i=1  #Cyto
j=6  #Nuc
mixProtiProtj16Acup <- proteinMix(refLocationProfilesAcup, Loc1=i, Loc2=j)
mixProtiProtj16RSA <- RSAfromAcup(Acup=mixProtiProtj16Acup,
                            NstartMaterialFractions=5, totProt=totProt)
mixProtiProtjCPAfromRSA16 <- fitCPA(profile=mixProtiProtj16RSA,
                                refLocationProfiles=refLocationProfilesRSA,
                                numDataCols=5)
i=1  #Cyto
j=7  #Perox
mixProtiProtj17Acup <- proteinMix(refLocationProfilesAcup, Loc1=i, Loc2=j)
mixProtiProtj17RSA <- RSAfromAcup(Acup=mixProtiProtj17Acup,
                            NstartMaterialFractions=5, totProt=totProt)
mixProtiProtjCPAfromRSA17 <- fitCPA(profile=mixProtiProtj17RSA,
                                refLocationProfiles=refLocationProfilesRSA,
```

```
                                numDataCols=5)

i=1  #Cyto
j=8  #PM
mixProtiProtj18Acup <- proteinMix(refLocationProfilesAcup, Loc1=i, Loc2=j)
mixProtiProtj18RSA <- RSAfromAcup(Acup=mixProtiProtj18Acup,
                              NstartMaterialFractions=5, totProt=totProt)
mixProtiProtjCPAfromRSA18 <- fitCPA(profile=mixProtiProtj18RSA,
                               refLocationProfiles=refLocationProfilesRSA,
                              numDataCols=5)
```

Now we can plot all simulations for mixing Cyto with each of the other seven compartments. Note that as before, the x-coordinate represents the theoretical distribution based on simulation parameters and the y-coordinate represents the predicted values based on CPA. The assignment errors are shown in parentheses.

We see that CPA performs well except for Cyto-Lyso and Cyto-PM:

```
#library(pracma)   # for trapz function
par(mfrow=c(2,4))

mixturePlot(mixProtiProtjCPA=mixProtiProtjCPAfromRSA12,
            NstartMaterialFractions=5, Loc1=1, Loc2=2,
            errorReturn = TRUE, subTitle="RSA")
```

```
#>   Loc1 Loc2   ErrorArea
#> 1 Cyto   ER 0.001634335
```

```
mixturePlot(mixProtiProtjCPA=mixProtiProtjCPAfromRSA13,
            NstartMaterialFractions=5, Loc1=1, Loc2=3,
            errorReturn = TRUE, subTitle="RSA")
```

```
#>   Loc1  Loc2    ErrorArea
#> 1 Cyto Golgi 2.029203e-06
```

```
mixturePlot(mixProtiProtjCPA=mixProtiProtjCPAfromRSA14,
            NstartMaterialFractions=5, Loc1=1, Loc2=4,
            errorReturn = TRUE, subTitle="RSA")
```

```
#>   Loc1 Loc2 ErrorArea
#> 1 Cyto Lyso 0.8890269
```

```
mixturePlot(mixProtiProtjCPA=mixProtiProtjCPAfromRSA15,
            NstartMaterialFractions=5, Loc1=1, Loc2=5,
            errorReturn = TRUE, subTitle="RSA")
```

```
#>   Loc1 Loc2    ErrorArea
#> 1 Cyto Mito 5.263772e-07
```

```
mixturePlot(mixProtiProtjCPA=mixProtiProtjCPAfromRSA16,
            NstartMaterialFractions=5, Loc1=1, Loc2=6,
            errorReturn = TRUE, subTitle="RSA")
```

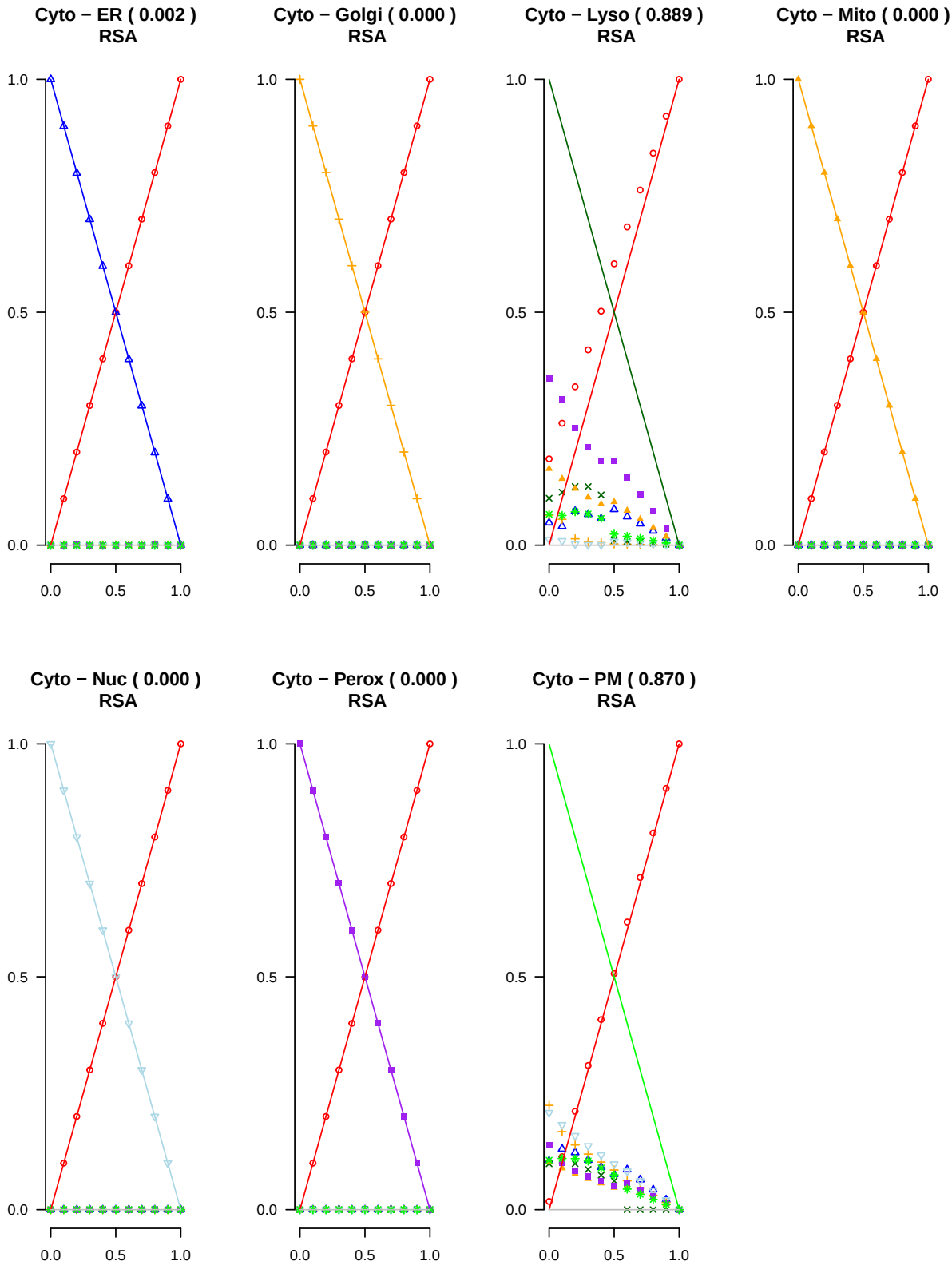
```
#>   Loc1 Loc2    ErrorArea
#> 1 Cyto  Nuc 4.257653e-07
```

```
mixturePlot(mixProtiProtjCPA=mixProtiProtjCPAfromRSA17,
            NstartMaterialFractions=5, Loc1=1, Loc2=7,
            errorReturn = TRUE, subTitle="RSA")
```

```
#>   Loc1  Loc2   ErrorArea
#> 1 Cyto Perox 2.90932e-06
```

```
mixturePlot(mixProtiProtjCPA=mixProtiProtjCPAfromRSA18,
            NstartMaterialFractions=5, Loc1=1, Loc2=8,
            errorReturn = TRUE, subTitle="RSA")
```

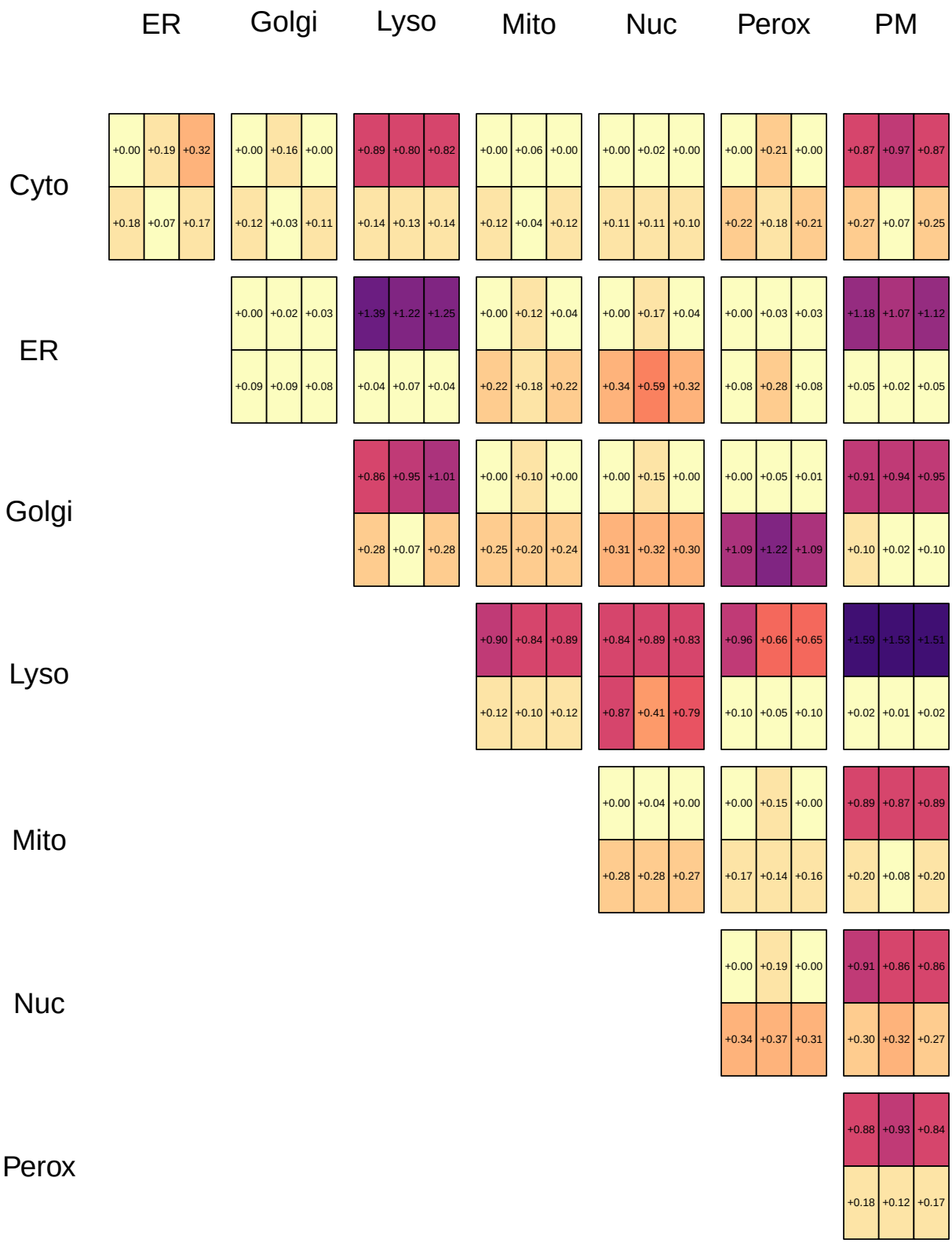
```
#>   Loc1 Loc2 ErrorArea
#> 1 Cyto   PM 0.8697084
```



As described in the previous tutorial, we can look at a heatmap of the errors for the different pairwise mixtures and transformations. As a reminder, these values are presented as a 2 by 3 array, with the three

transformations as columns (RSA, left; NSA, center; Acup, right). The top row uses the original values (identity transformation), and the bottom row uses a log2 transformation of these values. The prediction errors are listed in each box with larger errors indicated by darker colors. CPA on mixtures involving Cyto and Lyso or PM have the highest error rates. The errors are reduced for these mixtures when conducting CPA on log-transformed NSA profiles but this has adverse effects on the fit for other compartments (see below, Appendix).

```
#install.packages(c("plot.matrix", "viridis", "grid", "gridExtra"))

par(mfrow=c(1,1))
errorMatAll <- mixtureHeatMap(Acup=refLocationProfilesAcup,
                        totProt=totProt, NstartMaterialFractions=5)
```



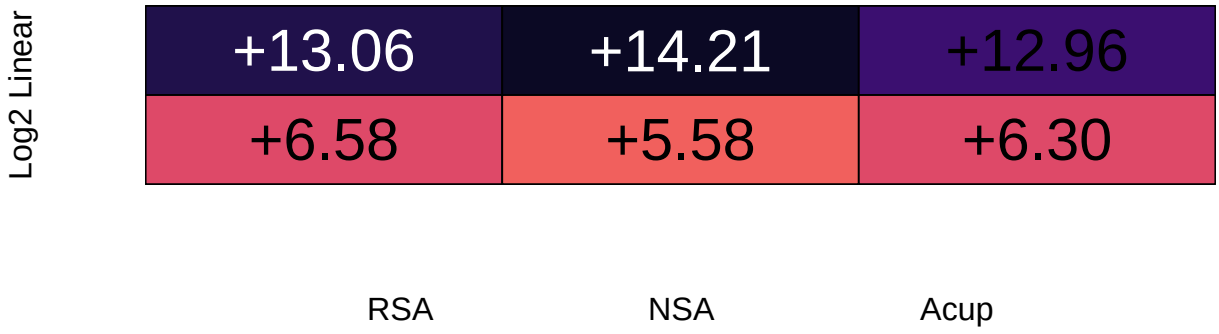
We can also either print out overall errors (i.e., sum of all errors for each combination of data transformation

method), or, alternatively, generate a heatmap of the overall errors.

```
errorMatAll
```

```
#>           [,1]      [,2]      [,3]
#> [1,] 13.064310 14.205022 12.964319
#> [2,]  6.577352  5.583711  6.295502
```

```
#library(plot.matrix)
#library("viridis")
op <- par(mar=c(4,4,1,1))  # save default parameters in variable op

col <- rev(viridis::magma(16))
library(plot.matrix)
plot(errorMatAll, col=col, breaks=seq(0, 15, 1), key=NULL, main="",
     axis.col=NULL, axis.row=NULL,
     xlab="RSA                          NSA                          Acup",
     ylab="Log2 Linear", digits=2, cex=2, cex.lab=1.1)
par(op) # restore default parameters
```



Restricted constrained proportional assignment

From the examples above, our modeling shows that `fitCPA` does not report the correct assignment for some of the simulated binary mixtures from a five-fraction differential centrifugation experiment. To investigate this further, we have conducted modeling where we restrict the CPA algorithm to only assign weights to specified subsets of compartments. In the following code, we examine the effect of restricting CPA to fit a simulated protein with a 50:50 distribution between Cyto and Lyso. We have already created this earlier in this tutorial as the sixth entry in the data frame `mixProtiProtj14RSA` and create a new one-row data frame that only contains this simulated protein.

```
data.frame(rownames(mixProtiProtj14RSA))
```

```
#>    rownames.mixProtiProtj14RSA.
#> 1                0_Cyto:1_Lyso
#> 2           0.1_Cyto:0.9_Lyso
#> 3           0.2_Cyto:0.8_Lyso
#> 4           0.3_Cyto:0.7_Lyso
#> 5           0.4_Cyto:0.6_Lyso
#> 6           0.5_Cyto:0.5_Lyso
```

```
#> 7             0.6_Cyto:0.4_Lyso
#> 8             0.7_Cyto:0.3_Lyso
#> 9             0.8_Cyto:0.2_Lyso
#> 10            0.9_Cyto:0.1_Lyso
#> 11                1_Cyto:0_Lyso
```

```
mixCyto50Lyso50RSA <- mixProtiProtj14RSA[6,]
```

In our first example, we force the algorithm to only assign positive weights to Cyto and Lyso, holding the remaining weights fixed at zero. We do this using the `fitCPA` function where the optional parameter `ind.vary` specifies the row numbers for the compartments whose proportions are allowed to vary (Cyto=1, Lyso=4) with the remaining compartment proportions fixed at zero. The option `minVal=TRUE` causes the value of the sum of squares goodness of fit (`value`) to be printed. From the output we see that the CPA routine with these constraints correctly reports the distribution of the simulated protein.

```
mixCyto50Lyso50CPAfromRSAforced <- fitCPA(profile= mixCyto50Lyso50RSA,
                                refLocationProfiles=refLocationProfilesRSA,
                                numDataCols=5, ind.vary=c(1,4), minVal=TRUE)
round(mixCyto50Lyso50CPAfromRSAforced, digits=4)
```

```
#>                   Cyto ER Golgi Lyso Mito Nuc Perox PM value
#> 0.5_Cyto:0.5_Lyso  0.5  0     0  0.5    0   0     0  0     0
```

As a second example, we allow positive weights to only two of the eight compartments at a time as described above, cycling through all possible pairs of compartments. Here we introduce a new function, `fCPAsubsets`, that automates this procedure, with an argument `nCPAcomparts` that specifies the number of compartments allowed to vary in each cycle. Note that when we specify `nCPAcomparts=2` for eight compartments, there are 8 choose 2 = 28 possible pairs. The output is in a data frame that contains each of the assignments and goodness of fit values, sorted by the parameter `value`. Here, the top CPA assignment has a goodness of fit value markedly better than the others and gives the correct assignment between Cyto and Lyso.

```
mixCyto50Lyso50CPAfromRSApairs <- fCPAsubsets(profile= mixCyto50Lyso50RSA,
                                  refLocationProfiles=refLocationProfilesRSA,
                                  numDataCols=5,nCPAcomparts=2)
round(mixCyto50Lyso50CPAfromRSApairs, digits=4)
```

```
#>              Cyto     ER  Golgi   Lyso   Mito    Nuc  Perox     PM  value
#> CytoLyso   0.5000 0.0000 0.0000 0.5000 0.0000 0.0000 0.0000 0.0000 0.0000
#> CytoPerox  0.7046 0.0000 0.0000 0.0000 0.0000 0.0000 0.2954 0.0000 0.1529
#> CytoPM     0.6121 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.3879 0.2520
#> CytoER     0.7242 0.2758 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.3711
#> CytoMito   0.6770 0.0000 0.0000 0.0000 0.3230 0.0000 0.0000 0.0000 0.5153
#> CytoGolgi  0.8055 0.0000 0.1945 0.0000 0.0000 0.0000 0.0000 0.0000 0.7309
#> CytoNuc    0.8693 0.0000 0.0000 0.0000 0.0000 0.1307 0.0000 0.0000 1.0615
#> LysoNuc    0.0000 0.0000 0.0000 0.9291 0.0000 0.0709 0.0000 0.0000 1.2333
#> LysoMito   0.0000 0.0000 0.0000 0.9401 0.0599 0.0000 0.0000 0.0000 1.2809
#> GolgiLyso  0.0000 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 0.0000 1.2916
#> LysoPerox  0.0000 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 0.0000 1.2916
#> LysoPM     0.0000 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 0.0000 1.2916
#> ERLyso     0.0000 0.0000 0.0000 1.0000 0.0000 0.0000 0.0000 0.0000 1.2916
#> MitoPM     0.0000 0.0000 0.0000 0.0000 0.3488 0.0000 0.0000 0.6512 2.4036
#> PeroxPM    0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.2018 0.7982 2.5817
```

```
#> ERPM       0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000 2.8405
#> GolgiPM    0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000 2.8405
#> NucPM      0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000 2.8405
#> MitoPerox  0.0000 0.0000 0.0000 0.0000 0.6539 0.0000 0.3461 0.0000 2.8735
#> ERMito     0.0000 0.3390 0.0000 0.0000 0.6610 0.0000 0.0000 0.0000 2.9306
#> GolgiMito  0.0000 0.0000 0.2388 0.0000 0.7612 0.0000 0.0000 0.0000 3.2253
#> NucPerox   0.0000 0.0000 0.0000 0.0000 0.0000 0.3855 0.6145 0.0000 3.6405
#> MitoNuc    0.0000 0.0000 0.0000 0.0000 0.8457 0.1543 0.0000 0.0000 3.6746
#> GolgiPerox 0.0000 0.0000 0.3823 0.0000 0.0000 0.0000 0.6177 0.0000 4.3224
#> ERPerox    0.0000 0.4946 0.0000 0.0000 0.0000 0.0000 0.5054 0.0000 4.6788
#> ERNuc      0.0000 0.6464 0.0000 0.0000 0.0000 0.3536 0.0000 0.0000 4.7847
#> GolgiNuc   0.0000 0.0000 0.5248 0.0000 0.0000 0.4752 0.0000 0.0000 6.2997
#> ERGolgi    0.0000 1.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 6.7179
```

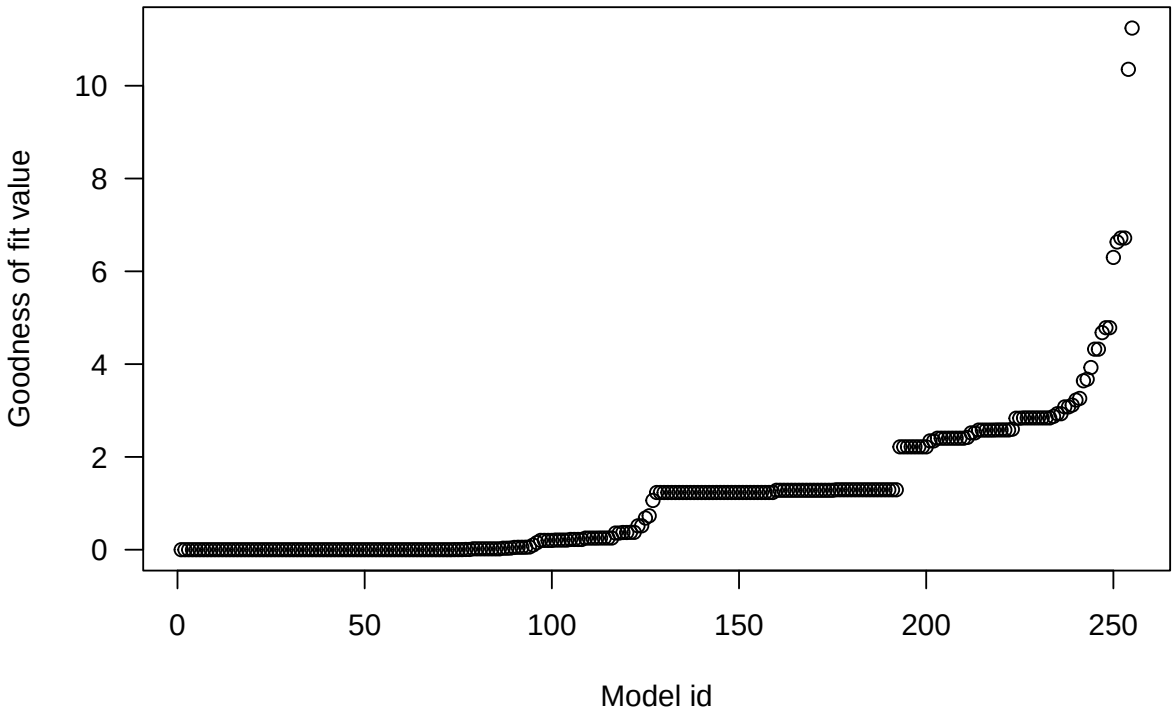
It is also possible to consider other fixed combinations. For example, `nCPAcomparts=3` would cycle through 8 choose 3 = 56 combinations where three compartment proportions are allowed to vary with the remaining five fixed at zero. As a final example we consider all 2^8 - 1 = 255 possible subsets of compartments and conduct CPA using each of these subsets. In the code below, we successively set `nCPAcomparts` equal to 1, 2, 3..., 8 and create a data frame that contains the estimated proportions and goodness of fit parameter values. We then sort by the parameter `value` and observe that many CPA assignments are nearly as good as the top assignment.

```
mixCyto50Lyso50CPAfromRSAallSubsets<- NULL
for (kk in 1:8) {
   temp  <- fCPAsubsets(profile=mixCyto50Lyso50RSA,
                              refLocationProfiles=refLocationProfilesRSA,
                              numDataCols=5, nCPAcomparts=kk)
   mixCyto50Lyso50CPAfromRSAallSubsets <-
                        rbind(mixCyto50Lyso50CPAfromRSAallSubsets, temp)
}
# sort dataframe by the variable "value" by creating an intermediate
#   list of indices "ord_fit".
ord_fit <- order(mixCyto50Lyso50CPAfromRSAallSubsets$value)
mixCyto50Lyso50CPAfromRSAallSubsetsOrd <-
         mixCyto50Lyso50CPAfromRSAallSubsets[ord_fit,]
round(head(mixCyto50Lyso50CPAfromRSAallSubsetsOrd), 3)
```

```
#>                             Cyto    ER Golgi  Lyso  Mito   Nuc Perox    PM value
#> CytoLyso                   0.500 0.000 0.000 0.500 0.000 0.000 0.000 0.000     0
#> CytoGolgiLyso              0.500 0.000 0.000 0.500 0.000 0.000 0.000 0.000     0
#> CytoERGolgiMitoNucPerox    0.607 0.083 0.008 0.000 0.098 0.014 0.190 0.000     0
#> CytoERLysoMitoNucPeroxPM   0.604 0.079 0.000 0.011 0.093 0.007 0.180 0.025     0
#> CytoLysoMitoNuc            0.500 0.000 0.000 0.500 0.000 0.000 0.000 0.000     0
#> CytoERGolgiMitoNucPeroxPM 0.606 0.078 0.003 0.000 0.095 0.008 0.185 0.025     0
```

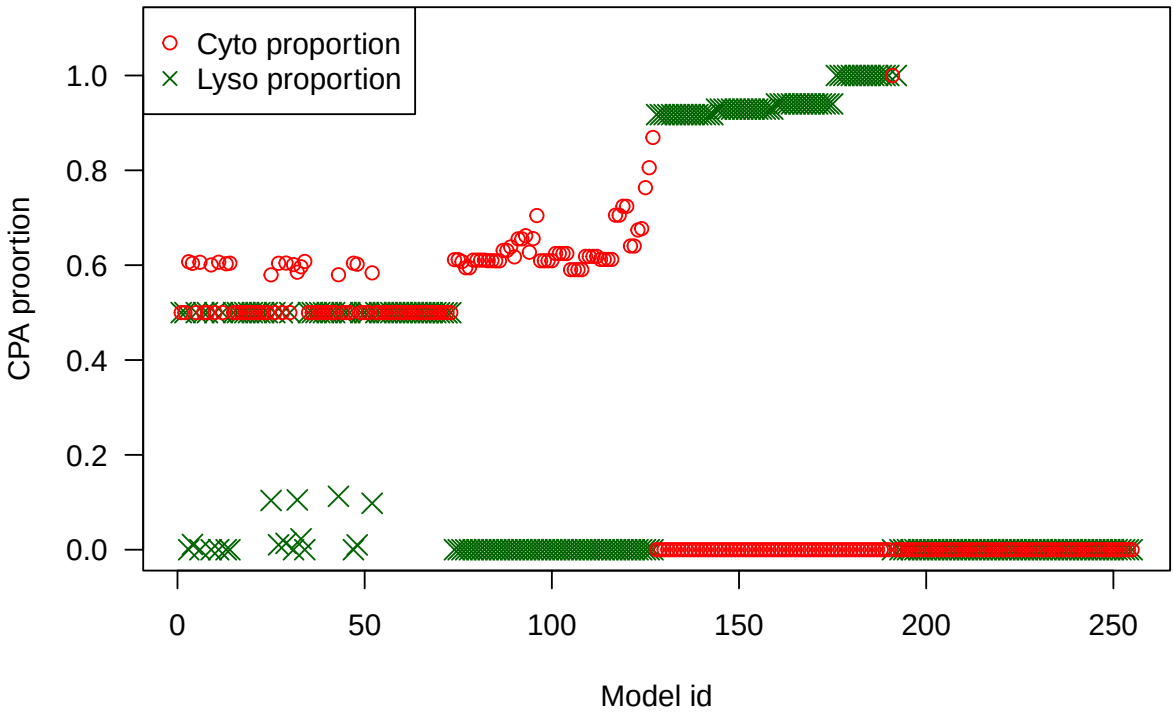
We may plot the goodness of fit statistic `value` and the CPA assignment proportions for Cyto and Lyso versus model number as follows, with the best fitting models first. This plot shows that the goodness of fit statistic is extremely low for a large number of models:

```
modelVals <- 1:255
par(mfrow=c(1,1))  # reset the plot space
plot(mixCyto50Lyso50CPAfromRSAallSubsetsOrd$value ~ modelVals,
     xlab="Model id", ylab="Goodness of fit value", las=1)
```



In the following plot we show the CPA estimates for Cyto and Lyso for all models. Even for the best fitting models (with id numbers 1 to about 90), the CPA estimates vary quite a lot.

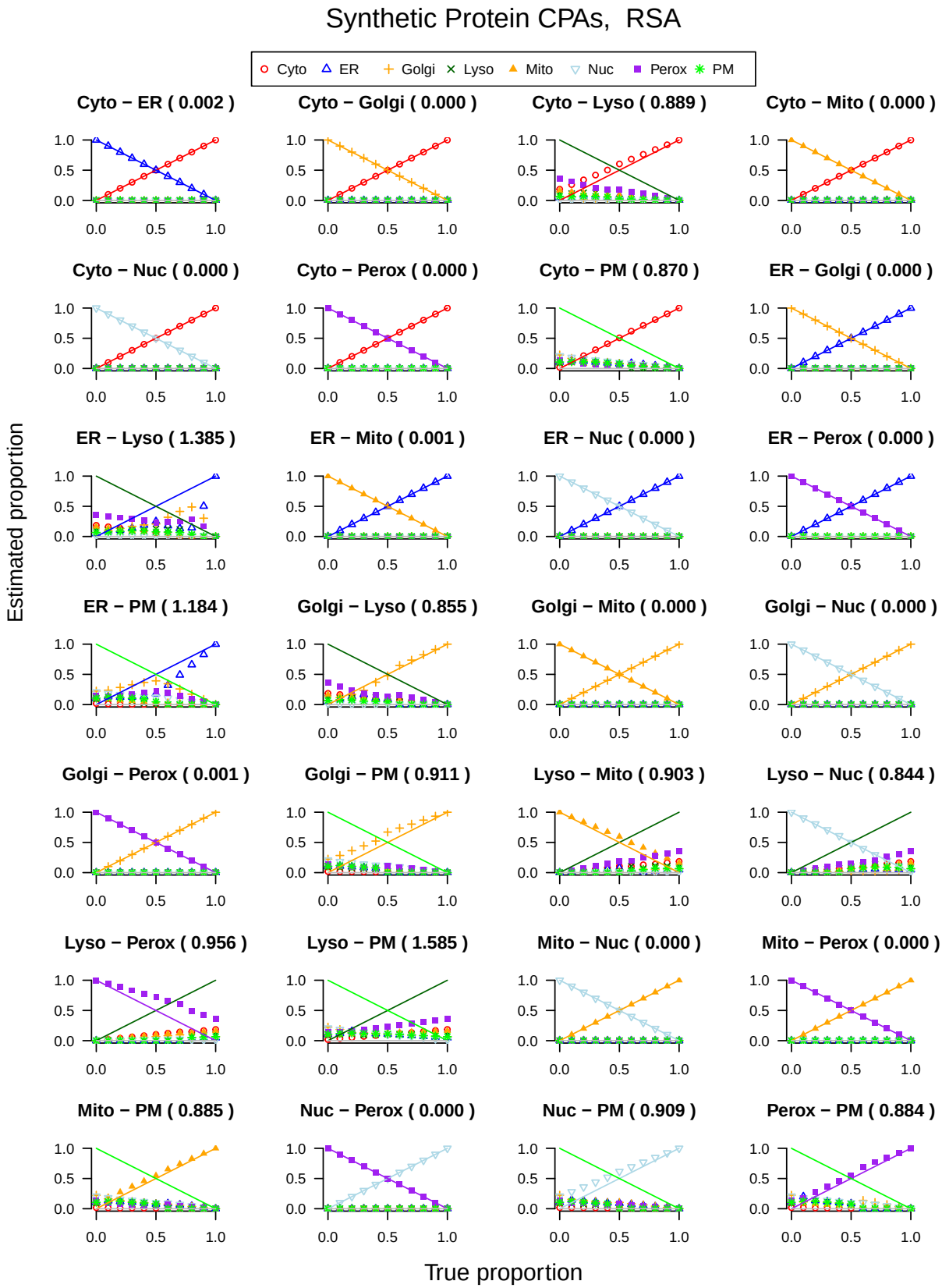
```
par(mfrow=c(1,1))  # reset the plot space
plot(mixCyto50Lyso50CPAfromRSAallSubsetsOrd$Lyso ~ modelVals, xlab="Model id",
     ylab="CPA proortion", type="n", ylim=c(0, 1.1), las=1)

points(mixCyto50Lyso50CPAfromRSAallSubsetsOrd$Lyso ~ modelVals, pch=4,
       col="darkgreen", cex=1.5)
points(mixCyto50Lyso50CPAfromRSAallSubsetsOrd$Cyto ~ modelVals, pch=1, col="red")
legend(x="topleft", legend=c("Cyto proportion", "Lyso proportion"),
       pch=c( 1, 4), col=c("red", "darkgreen"))
```



From this it is clear that the model is over-parameterized with multiple models that fit nearly as well yet give inaccurate CPA estimates. Further study of the properties of estimation methods when multiple models fit equally well is warranted.

Appendix: Plots of CPA assignments of all two-compartent simulated proteins using linear and log RSA, NSA, and Acup data

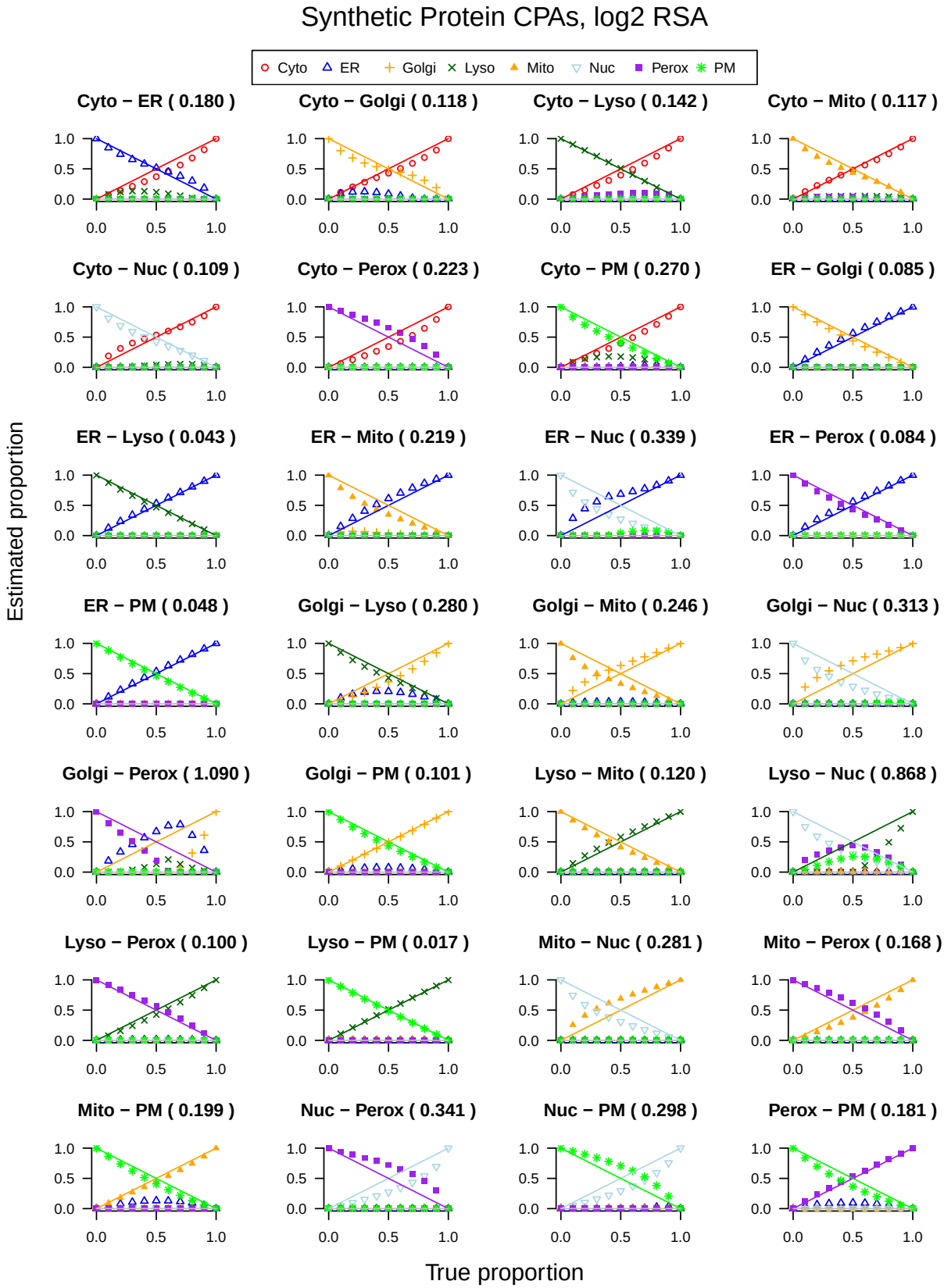
```
errorAllRSAlinear <- mixturePlotPanel(refLocationProfilesAcup=
                                refLocationProfilesAcup,
                                totProt=totProt, NstartMaterialFractions=5,
                                errorReturn = TRUE,
                                fitType="RSA", log2Transf=FALSE)
```

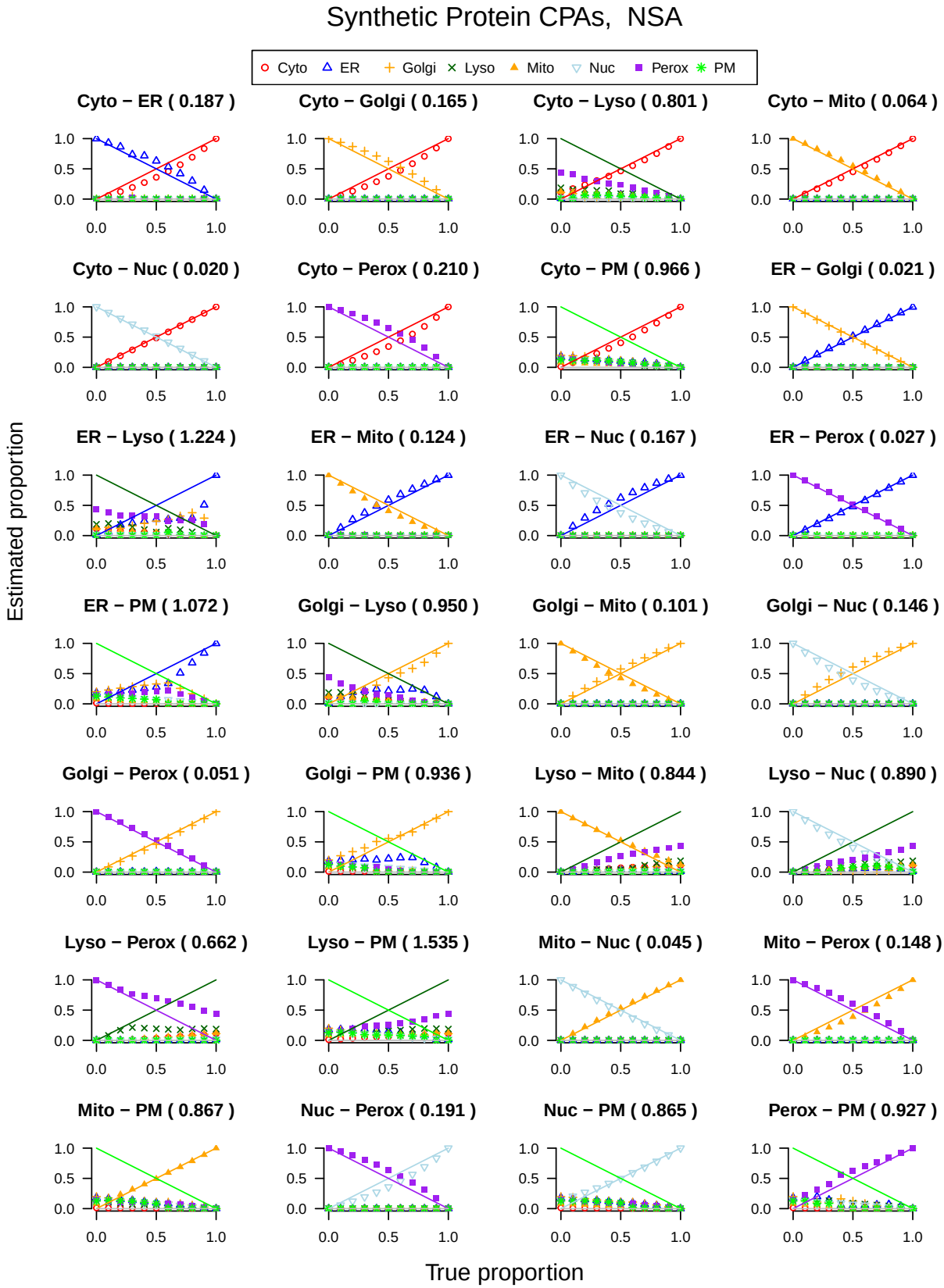


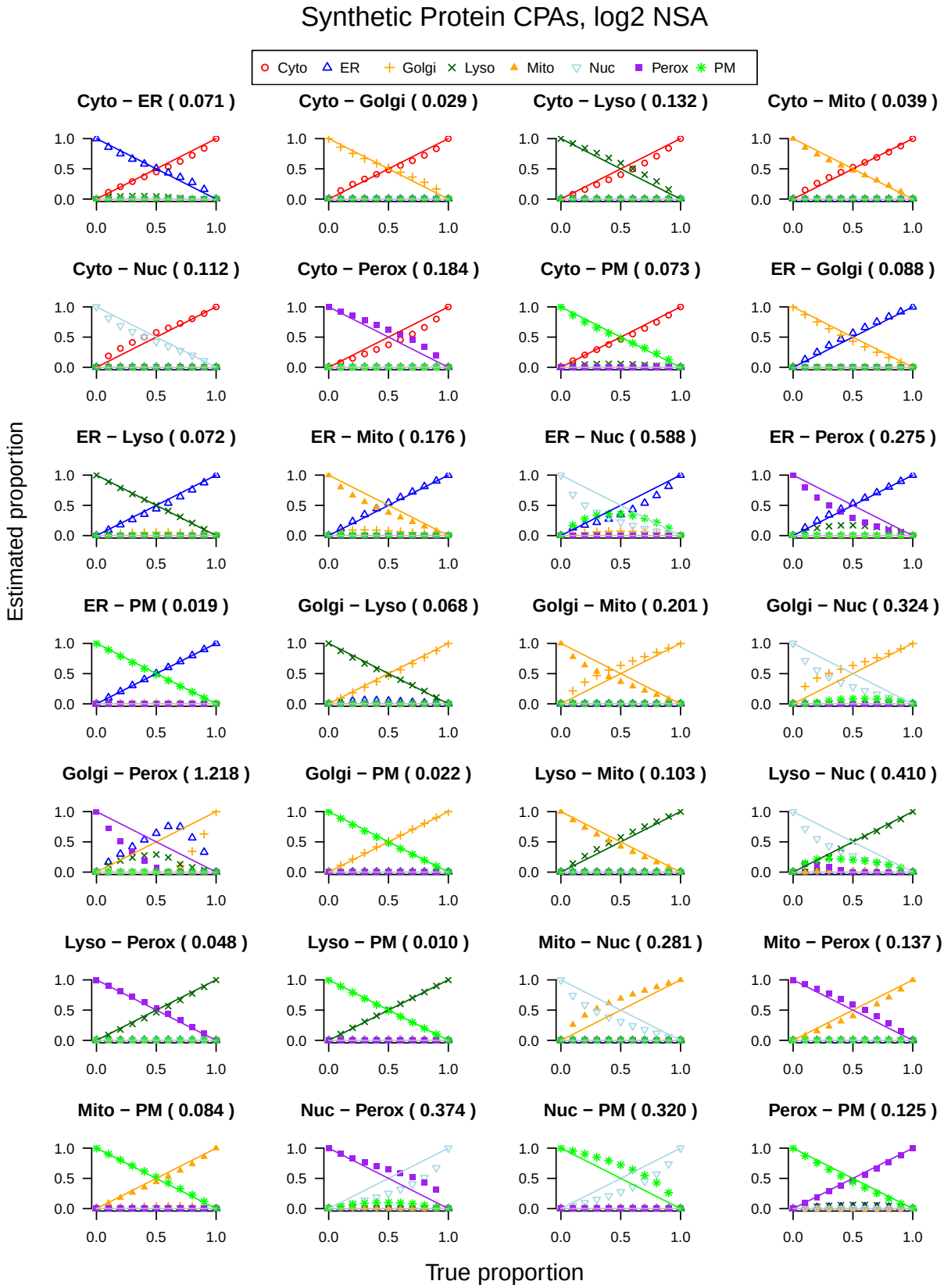
The mixing simulations indicate that for mixtures involving Lyso or PM, there is a poor fit.

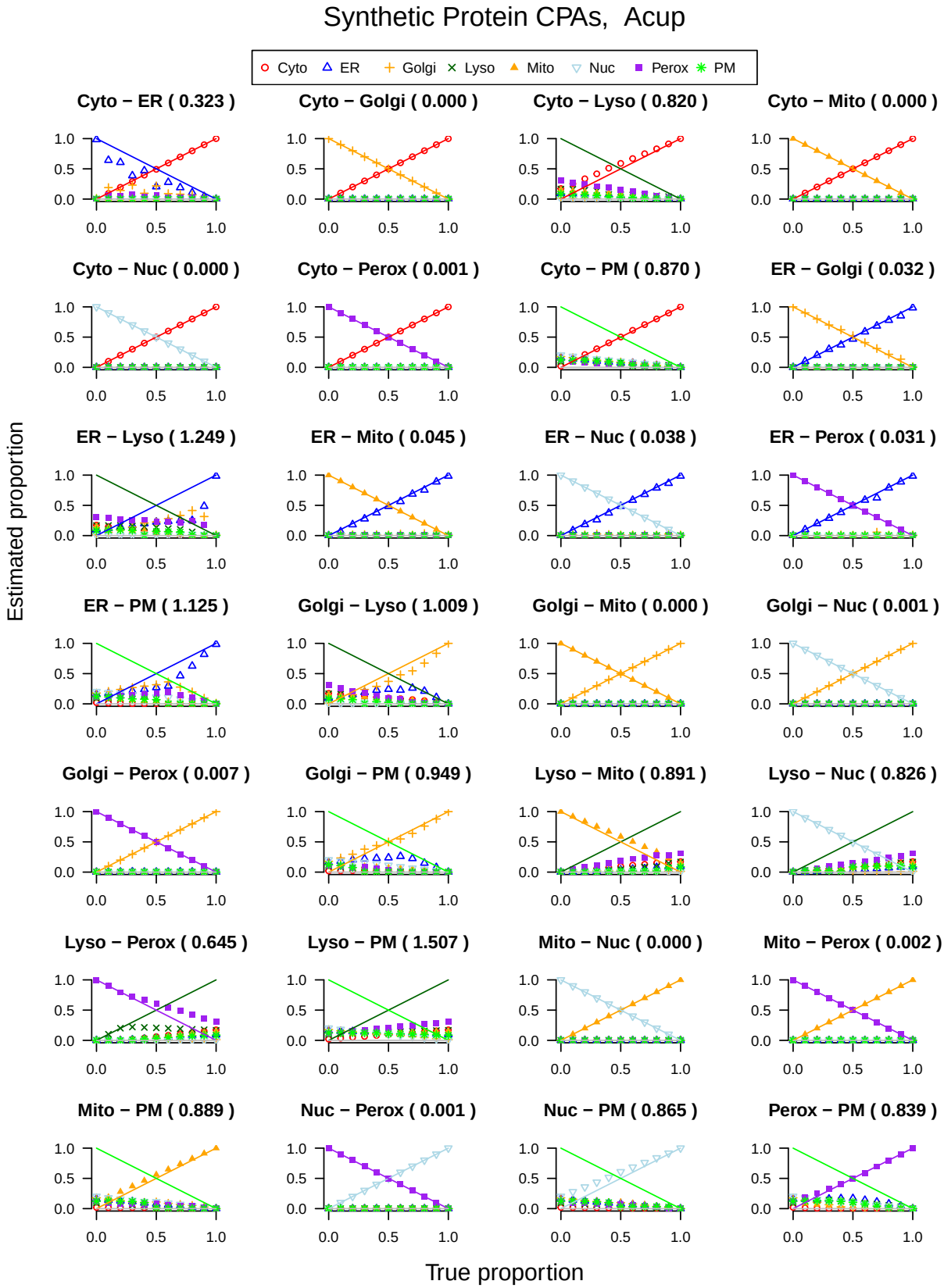
Analysis of pairwise simulations using log2-transformed RSA profiles as well as NSA and Acup profiles (with and without log2 transformations) are displayed below. Overall, the best results are obtained using log-transformed NSA data. Here, we see that pairwise mixtures involving Lyso and PM are improved compared to the fits with untransformed RSA profiles but many other combinations involving Nuc and ER are worsened:

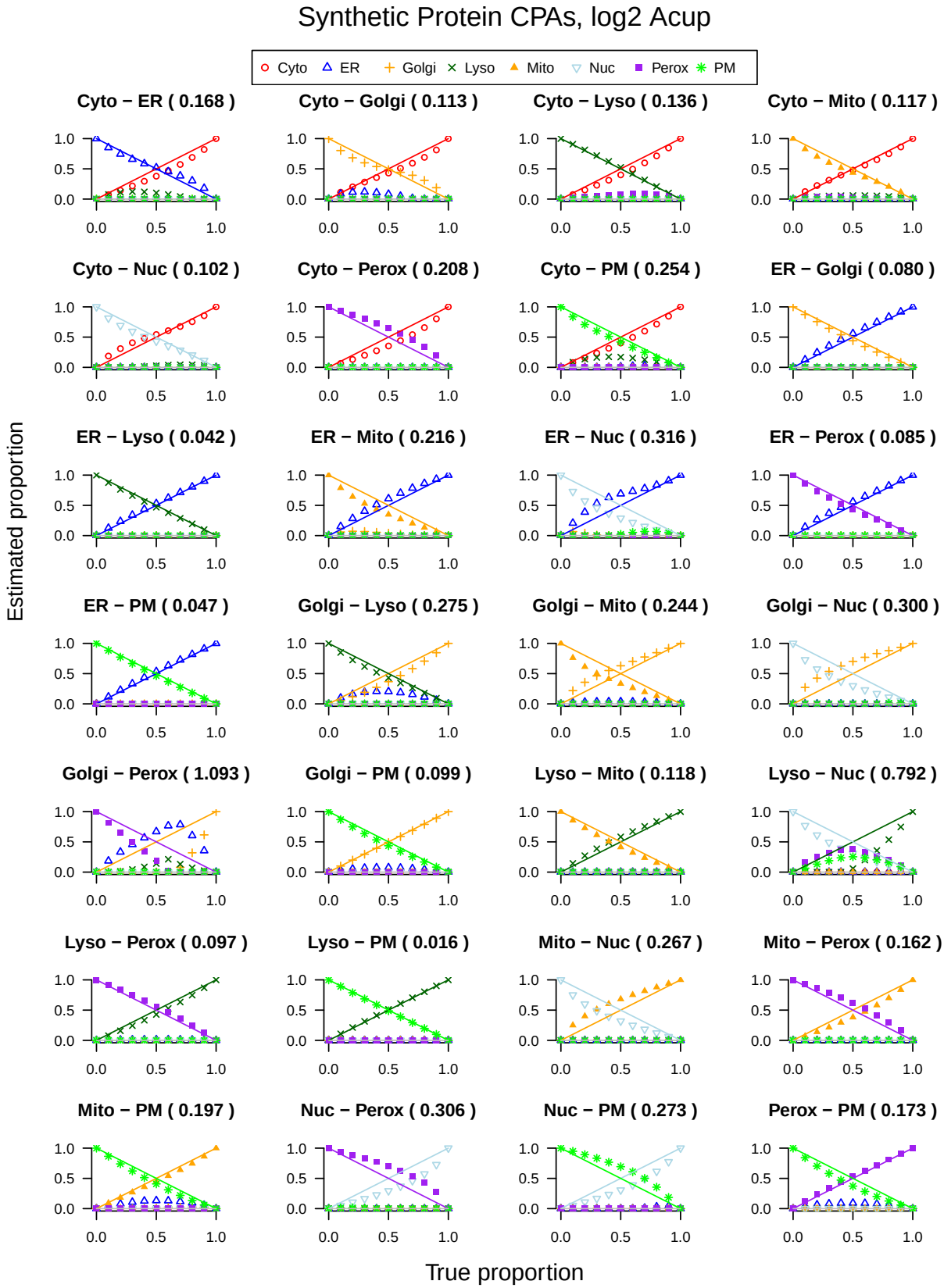
```
errorAllRSAlog2 <- mixturePlotPanel(refLocationProfilesAcup=
                                    refLocationProfilesAcup,
                                    totProt=totProt,
                                    NstartMaterialFractions=5,
                                    errorReturn = TRUE,
                                    fitType="RSA", log2Transf=TRUE)


errorAllNSAlinear <- mixturePlotPanel(refLocationProfilesAcup=
                                    refLocationProfilesAcup,
                                    totProt=totProt,
                                    NstartMaterialFractions=5,
                                    errorReturn = TRUE,
                                    fitType="NSA", log2Transf=FALSE)

errorAllNSAlog2 <- mixturePlotPanel(refLocationProfilesAcup=
                                    refLocationProfilesAcup,
                                    totProt=totProt,
                                    NstartMaterialFractions=5,
                                    errorReturn = TRUE,
                                    fitType="NSA", log2Transf=TRUE)

errorAllAcupLinear <- mixturePlotPanel(refLocationProfilesAcup=
                                    refLocationProfilesAcup,
                                    totProt=totProt,
                                    NstartMaterialFractions=5,
                                    errorReturn = TRUE,
                                    fitType="Acup", log2Transf=FALSE)

errorAllAcupLog2 <- mixturePlotPanel(refLocationProfilesAcup=
                                    refLocationProfilesAcup,
                                    totProt=totProt,
                                    NstartMaterialFractions=5,
                                    errorReturn = TRUE,
                                    fitType="Acup", log2Transf=TRUE)
```


Synthetic Protein CPAs, log2 RSA
Cyto
ER
Golgi
Lyso
Mito
Nuc
Perox
PM
Cyto − ER (0.180)
Cyto − Golgi (0.118)
Cyto − Lyso (0.142)
Cyto − Mito (0.117)
Cyto − Nuc (0.109)
Cyto − Perox (0.223)
Cyto − PM (0.270)
ER − Golgi (0.085)
ER − Lyso (0.043)
ER − Mito (0.219)
ER − Nuc (0.339)
ER − Perox (0.084)
ER − PM (0.048)
Golgi − Lyso (0.280)
Golgi − Mito (0.246)
Golgi − Nuc (0.313)
Golgi − Perox (1.090)
Golgi − PM (0.101)
Lyso − Mito (0.120)
Lyso − Nuc (0.868)
Lyso − Perox (0.100)
Lyso − PM (0.017)
Mito − Nuc (0.281)
Mito − Perox (0.168)
Mito − PM (0.199)
Nuc − Perox (0.341)
Nuc − PM (0.298)
Perox − PM (0.181)
Estimated proportion
True proportion
0.0
0.5
1.0


Synthetic Protein CPAs, NSA
Cyto
ER
Golgi
Lyso
Mito
Nuc
Perox
PM
Cyto − ER (0.187)
Cyto − Golgi (0.165)
Cyto − Lyso (0.801)
Cyto − Mito (0.064)
Cyto − Nuc (0.020)
Cyto − Perox (0.210)
Cyto − PM (0.966)
ER − Golgi (0.021)
ER − Lyso (1.224)
ER − Mito (0.124)
ER − Nuc (0.167)
ER − Perox (0.027)
ER − PM (1.072)
Golgi − Lyso (0.950)
Golgi − Mito (0.101)
Golgi − Nuc (0.146)
Golgi − Perox (0.051)
Golgi − PM (0.936)
Lyso − Mito (0.844)
Lyso − Nuc (0.890)
Lyso − Perox (0.662)
Lyso − PM (1.535)
Mito − Nuc (0.045)
Mito − Perox (0.148)
Mito − PM (0.867)
Nuc − Perox (0.191)
Nuc − PM (0.865)
Perox − PM (0.927)
Estimated proportion
True proportion
0.0
0.5
1.0


Synthetic Protein CPAs, log2 NSA
Cyto
ER
Golgi
Lyso
Mito
Nuc
Perox
PM
Cyto − ER (0.071)
Cyto − Golgi (0.029)
Cyto − Lyso (0.132)
Cyto − Mito (0.039)
Cyto − Nuc (0.112)
Cyto − Perox (0.184)
Cyto − PM (0.073)
ER − Golgi (0.088)
ER − Lyso (0.072)
ER − Mito (0.176)
ER − Nuc (0.588)
ER − Perox (0.275)
ER − PM (0.019)
Golgi − Lyso (0.068)
Golgi − Mito (0.201)
Golgi − Nuc (0.324)
Golgi − Perox (1.218)
Golgi − PM (0.022)
Lyso − Mito (0.103)
Lyso − Nuc (0.410)
Lyso − Perox (0.048)
Lyso − PM (0.010)
Mito − Nuc (0.281)
Mito − Perox (0.137)
Mito − PM (0.084)
Nuc − Perox (0.374)
Nuc − PM (0.320)
Perox − PM (0.125)
Estimated proportion
True proportion
0.0
0.5
1.0


Synthetic Protein CPAs, Acup
Cyto
ER
Golgi
Lyso
Mito
Nuc
Perox
PM
Cyto − ER (0.323)
Cyto − Golgi (0.000)
Cyto − Lyso (0.820)
Cyto − Mito (0.000)
Cyto − Nuc (0.000)
Cyto − Perox (0.001)
Cyto − PM (0.870)
ER − Golgi (0.032)
ER − Lyso (1.249)
ER − Mito (0.045)
ER − Nuc (0.038)
ER − Perox (0.031)
ER − PM (1.125)
Golgi − Lyso (1.009)
Golgi − Mito (0.000)
Golgi − Nuc (0.001)
Golgi − Perox (0.007)
Golgi − PM (0.949)
Lyso − Mito (0.891)
Lyso − Nuc (0.826)
Lyso − Perox (0.645)
Lyso − PM (1.507)
Mito − Nuc (0.000)
Mito − Perox (0.002)
Mito − PM (0.889)
Nuc − Perox (0.001)
Nuc − PM (0.865)
Perox − PM (0.839)
Estimated proportion
True proportion



If desired, these plots of pairwise mixtures may also be saved as pdf files as described in Tutorial 4.

Finally, we present the pairwise and overall errors for log2 RSA (errors for identity RSA are reported above), identity and log2 NSA, and identity and log2 Acup:

```
errorAllRSAlinear
```

```
#>     Loc1  Loc2    ErrorArea
#> 1   Cyto    ER 1.634335e-03
#> 2   Cyto Golgi 2.029203e-06
#> 3   Cyto  Lyso 8.890269e-01
#> 4   Cyto  Mito 5.263772e-07
#> 5   Cyto   Nuc 4.257653e-07
#> 6   Cyto Perox 2.909320e-06
#> 7   Cyto    PM 8.697084e-01
#> 8     ER Golgi 1.274977e-04
#> 9     ER  Lyso 1.385462e+00
#> 10    ER  Mito 5.235618e-04
#> 11    ER   Nuc 2.576128e-04
#> 12    ER Perox 1.270502e-04
#> 13    ER    PM 1.184002e+00
#> 14 Golgi  Lyso 8.551602e-01
#> 15 Golgi  Mito 2.731339e-06
#> 16 Golgi   Nuc 2.181434e-06
#> 17 Golgi Perox 1.431189e-03
#> 18 Golgi    PM 9.106582e-01
#> 19  Lyso  Mito 9.032125e-01
#> 20  Lyso   Nuc 8.441681e-01
#> 21  Lyso Perox 9.556024e-01
#> 22  Lyso    PM 1.585490e+00
#> 23  Mito   Nuc 5.582486e-07
#> 24  Mito Perox 2.400479e-06
#> 25  Mito    PM 8.852389e-01
#> 26   Nuc Perox 1.396260e-06
#> 27   Nuc    PM 9.086939e-01
#> 28 Perox    PM 8.836491e-01
```

```
sum(errorAllRSAlinear[,3])
```

```
#> [1] 13.06419
```

```
errorAllRSAlog2
```

```
#>     Loc1  Loc2  ErrorArea
#> 1   Cyto    ER 0.17973152
#> 2   Cyto Golgi 0.11790121
#> 3   Cyto  Lyso 0.14218672
#> 4   Cyto  Mito 0.11718309
#> 5   Cyto   Nuc 0.10876359
#> 6   Cyto Perox 0.22267706
#> 7   Cyto    PM 0.26960432
#> 8     ER Golgi 0.08505171
#> 9     ER  Lyso 0.04256264
#> 10    ER  Mito 0.21896424
```

```
#> 11    ER   Nuc 0.33945846
#> 12    ER Perox 0.08403438
#> 13    ER    PM 0.04765111
#> 14 Golgi  Lyso 0.27957118
#> 15 Golgi  Mito 0.24639893
#> 16 Golgi   Nuc 0.31285692
#> 17 Golgi Perox 1.08960247
#> 18 Golgi    PM 0.10091484
#> 19  Lyso  Mito 0.12002075
#> 20  Lyso   Nuc 0.86766281
#> 21  Lyso Perox 0.09964218
#> 22  Lyso    PM 0.01740127
#> 23  Mito   Nuc 0.28097910
#> 24  Mito Perox 0.16810506
#> 25  Mito    PM 0.19854884
#> 26   Nuc Perox 0.34124610
#> 27   Nuc    PM 0.29756883
#> 28 Perox    PM 0.18105381
```

```
sum(errorAllRSAlog2[,3])
```

```
#> [1] 6.577343
```

```
errorAllNSAlinear
```

```
#>     Loc1  Loc2  ErrorArea
#> 1   Cyto    ER 0.18740760
#> 2   Cyto Golgi 0.16479272
#> 3   Cyto  Lyso 0.80064738
#> 4   Cyto  Mito 0.06442394
#> 5   Cyto   Nuc 0.01969049
#> 6   Cyto Perox 0.21000207
#> 7   Cyto    PM 0.96613593
#> 8     ER Golgi 0.02094032
#> 9     ER  Lyso 1.22445837
#> 10    ER  Mito 0.12372506
#> 11    ER   Nuc 0.16731665
#> 12    ER Perox 0.02738182
#> 13    ER    PM 1.07188191
#> 14 Golgi  Lyso 0.94970730
#> 15 Golgi  Mito 0.10142550
#> 16 Golgi   Nuc 0.14556375
#> 17 Golgi Perox 0.05147364
#> 18 Golgi    PM 0.93555662
#> 19  Lyso  Mito 0.84438613
#> 20  Lyso   Nuc 0.88964465
#> 21  Lyso Perox 0.66219582
#> 22  Lyso    PM 1.53486935
#> 23  Mito   Nuc 0.04478843
#> 24  Mito Perox 0.14753229
#> 25  Mito    PM 0.86656833
#> 26   Nuc Perox 0.19108543
#> 27   Nuc    PM 0.86467560
#> 28 Perox    PM 0.92674492
```

```
sum(errorAllNSAlinear[,3])
```

```
#> [1] 14.20502
```

```
errorAllNSAlog2
```

```
#>     Loc1  Loc2   ErrorArea
#> 1   Cyto    ER 0.071349702
#> 2   Cyto Golgi 0.028732375
#> 3   Cyto  Lyso 0.131989945
#> 4   Cyto  Mito 0.038557816
#> 5   Cyto   Nuc 0.111732587
#> 6   Cyto Perox 0.184045704
#> 7   Cyto    PM 0.073321942
#> 8     ER Golgi 0.087957197
#> 9     ER  Lyso 0.071773861
#> 10    ER  Mito 0.176050302
#> 11    ER   Nuc 0.587884341
#> 12    ER Perox 0.275435395
#> 13    ER    PM 0.018602094
#> 14 Golgi  Lyso 0.067892585
#> 15 Golgi  Mito 0.201305957
#> 16 Golgi   Nuc 0.324342186
#> 17 Golgi Perox 1.218479260
#> 18 Golgi    PM 0.022025741
#> 19  Lyso  Mito 0.102952383
#> 20  Lyso   Nuc 0.409672662
#> 21  Lyso Perox 0.048194321
#> 22  Lyso    PM 0.009753017
#> 23  Mito   Nuc 0.281362159
#> 24  Mito Perox 0.137329762
#> 25  Mito    PM 0.084184646
#> 26   Nuc Perox 0.373537947
#> 27   Nuc    PM 0.320353730
#> 28 Perox    PM 0.124891374
```

```
sum(errorAllNSAlog2[,3])
```

```
#> [1] 5.583711
```

```
errorAllAcupLinear
```

```
#>     Loc1  Loc2    ErrorArea
#> 1   Cyto    ER 3.234184e-01
#> 2   Cyto Golgi 1.147505e-04
#> 3   Cyto  Lyso 8.201512e-01
#> 4   Cyto  Mito 4.617591e-05
#> 5   Cyto   Nuc 5.345681e-06
#> 6   Cyto Perox 1.493905e-03
#> 7   Cyto    PM 8.699487e-01
#> 8     ER Golgi 3.203018e-02
```

```
#> 9     ER  Lyso 1.249118e+00
#> 10    ER  Mito 4.451173e-02
#> 11    ER   Nuc 3.827435e-02
#> 12    ER Perox 3.060208e-02
#> 13    ER    PM 1.124628e+00
#> 14 Golgi  Lyso 1.008868e+00
#> 15 Golgi  Mito 2.030169e-04
#> 16 Golgi   Nuc 6.527809e-04
#> 17 Golgi Perox 7.235584e-03
#> 18 Golgi    PM 9.492534e-01
#> 19  Lyso  Mito 8.906562e-01
#> 20  Lyso   Nuc 8.255204e-01
#> 21  Lyso Perox 6.450895e-01
#> 22  Lyso    PM 1.506891e+00
#> 23  Mito   Nuc 7.082536e-06
#> 24  Mito Perox 1.674970e-03
#> 25  Mito    PM 8.888295e-01
#> 26   Nuc Perox 1.002255e-03
#> 27   Nuc    PM 8.646815e-01
#> 28 Perox    PM 8.394108e-01
```

```
sum(errorAllAcupLinear[,3])
```

```
#> [1] 12.96432
```

```
errorAllAcupLog2
```

```
#>     Loc1  Loc2  ErrorArea
#> 1   Cyto    ER 0.16849872
#> 2   Cyto Golgi 0.11345588
#> 3   Cyto  Lyso 0.13609218
#> 4   Cyto  Mito 0.11662875
#> 5   Cyto   Nuc 0.10178749
#> 6   Cyto Perox 0.20752857
#> 7   Cyto    PM 0.25365325
#> 8     ER Golgi 0.07963195
#> 9     ER  Lyso 0.04221200
#> 10    ER  Mito 0.21628815
#> 11    ER   Nuc 0.31602769
#> 12    ER Perox 0.08471409
#> 13    ER    PM 0.04726340
#> 14 Golgi  Lyso 0.27531830
#> 15 Golgi  Mito 0.24440641
#> 16 Golgi   Nuc 0.30020173
#> 17 Golgi Perox 1.09288844
#> 18 Golgi    PM 0.09934175
#> 19  Lyso  Mito 0.11756117
#> 20  Lyso   Nuc 0.79182564
#> 21  Lyso Perox 0.09668324
#> 22  Lyso    PM 0.01592133
#> 23  Mito   Nuc 0.26700030
#> 24  Mito Perox 0.16157381
#> 25  Mito    PM 0.19718609
```

```
#> 26   Nuc Perox 0.30636976
#> 27   Nuc    PM 0.27282690
#> 28 Perox    PM 0.17261501
```

```
sum(errorAllAcupLog2[,3])
```

```
#> [1] 6.295502
```

These can also be saved as tables as described in Tutorial 4.

Reproducibility

```
print(utils::sessionInfo(), width=80)
```

```
#> R version 4.1.3 (2022-03-10)
#> Platform: x86_64-w64-mingw32/x64 (64-bit)
#> Running under: Windows 10 x64 (build 19044)
#>
#> Matrix products: default
#>
#> locale:
#> [1] LC_COLLATE=English_United States.1252
#> [2] LC_CTYPE=English_United States.1252
#> [3] LC_MONETARY=English_United States.1252
#> [4] LC_NUMERIC=C
#> [5] LC_TIME=English_United States.1252
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#> [1] plot.matrix_1.6.1    pracma_2.3.8         protlocassign_0.99.1
#> [4] lme4_1.1-28          Matrix_1.4-0
#>
#> loaded via a namespace (and not attached):
#>  [1] Rcpp_1.0.8          lattice_0.20-45     prettyunits_1.1.1
#>  [4] ps_1.6.0            rprojroot_2.0.2     digest_0.6.29
#>  [7] utf8_1.2.2          R6_2.5.1            evaluate_0.15
#> [10] ggplot2_3.3.5       highr_0.9           pillar_1.7.0
#> [13] rlang_1.0.2         rstudioapi_0.13     minqa_1.2.4
#> [16] callr_3.7.0         nloptr_2.0.0        rmarkdown_2.13
#> [19] desc_1.4.1          devtools_2.4.3      splines_4.1.3
#> [22] BiocParallel_1.28.3 stringr_1.4.0       munsell_0.5.0
#> [25] tinytex_0.37        compiler_4.1.3      xfun_0.30
#> [28] pkgconfig_2.0.3     pkgbuild_1.3.1      htmltools_0.5.2
#> [31] tibble_3.1.6        gridExtra_2.3       BB_2019.10-1
#> [34] quadprog_1.5-8      fansi_1.0.2         viridisLite_0.4.0
#> [37] crayon_1.5.0        withr_2.5.0         MASS_7.3-55
#> [40] brio_1.1.3          grid_4.1.3          nlme_3.1-155
#> [43] gtable_0.3.0        lifecycle_1.0.1     magrittr_2.0.2
#> [46] scales_1.1.1        cli_3.2.0           stringi_1.7.6
```

```
#> [49] cachem_1.0.6        viridis_0.6.2       fs_1.5.2
#> [52] remotes_2.4.2       testthat_3.1.2      ellipsis_0.3.2
#> [55] vctrs_0.3.8         boot_1.3-28         tools_4.1.3
#> [58] outliers_0.14       glue_1.6.2          purrr_0.3.4
#> [61] processx_3.5.2      pkgload_1.2.4       parallel_4.1.3
#> [64] fastmap_1.1.0       yaml_2.3.5          colorspace_2.0-3
#> [67] sessioninfo_1.2.2   memoise_2.0.1       knitr_1.37
#> [70] usethis_2.1.5
```