Tutorial 4: More on mixtures and transformations

DFM & PL

2022-03-12

Contents

In this tutorial, we explore the effect of using normalized specific amount (NSA), relative specific amount (RSA), and relative amount (Acup) data, and their log transformations on CPA. We also discuss the underlying reasons why the different transformations can yield different results, which may be useful in choosing the appropriate transformation for a particular protein distribution. We illustrate using two mixtures: Cyto with Lyso and Cyto with Nuc. We begin by creating the NSA profiles for the reference compartments, which we will use for further transformations and to create mixtures.

Applying CPA to relative specific amount (RSA), normalized specific amount (NSA), or relative amount (Acup) data derived from simulated mixtures of Cyto and Lyso

For clarity of presentation, we simplify embedded data sets by dropping experiment-level notation:

```
library(protlocassign)
data(protNSA_test)
data(totProtAT5)
protNSA <- protNSA_test
totProt <- totProtAT5
```

```
data(markerListJadot)
refLocationProfilesNSA <- locationProfileSetup(profile=protNSA,
                            markerList=markerListJadot, numDataCols=9)
refLocationProfilesAcup <- AcupFromNSA(refLocationProfilesNSA,
                            NstartMaterialFractions=6, totProt=totProt)
```

Now create markers using refLocationProfilesAcup compartments Cyto (row 1) and Lyso (row 4) in the standard way, and transform the mixtures to relative specific amounts:

```
i=1
j=4
mixProt1Prot4Acup <- proteinMix(refLocationProfilesAcup, Loc1=i, Loc2=j)
mixProt1Prot4RSA <- RSAfromAcup(Acup=mixProt1Prot4Acup,
                            NstartMaterialFractions=6, totProt=totProt)
# increment.prop is not specified so we use the default value of 0.1
```

As in Tutorial 3, we display the Acup and RSA profiles for these Cyto-Lyso mixtures:

```
round(mixProt1Prot4Acup, digits=3)
```

```
#>                         N     M    L1    L2     P     S  Nyc1  Nyc2  Nyc3
#> 0_Cyto:1_Lyso     0.234 0.297 0.017 0.014 0.120 0.317 0.001 0.004 0.010
#> 0.1_Cyto:0.9_Lyso 0.222 0.277 0.016 0.013 0.112 0.360 0.001 0.003 0.009
#> 0.2_Cyto:0.8_Lyso 0.210 0.258 0.015 0.012 0.104 0.402 0.001 0.003 0.008
#> 0.3_Cyto:0.7_Lyso 0.198 0.238 0.013 0.011 0.096 0.445 0.000 0.003 0.008
#> 0.4_Cyto:0.6_Lyso 0.185 0.218 0.012 0.010 0.088 0.487 0.000 0.002 0.007
#> 0.5_Cyto:0.5_Lyso 0.173 0.198 0.010 0.009 0.079 0.530 0.000 0.002 0.006
#> 0.6_Cyto:0.4_Lyso 0.161 0.179 0.009 0.008 0.071 0.572 0.000 0.002 0.005
#> 0.7_Cyto:0.3_Lyso 0.149 0.159 0.007 0.007 0.063 0.615 0.000 0.001 0.004
#> 0.8_Cyto:0.2_Lyso 0.137 0.139 0.006 0.006 0.055 0.657 0.000 0.001 0.004
#> 0.9_Cyto:0.1_Lyso 0.124 0.119 0.005 0.005 0.047 0.700 0.000 0.001 0.003
#> 1_Cyto:0_Lyso     0.112 0.100 0.003 0.004 0.039 0.742 0.000 0.000 0.002
```

```
round(mixProt1Prot4RSA, digits=3)
```

```
#>                         N     M    L1    L2     P     S  Nyc1  Nyc2  Nyc3
#> 0_Cyto:1_Lyso     0.916 1.094 2.266 1.619 0.899 0.983 3.131 9.571 1.399
#> 0.1_Cyto:0.9_Lyso 0.868 1.021 2.081 1.498 0.838 1.114 2.890 8.673 1.289
#> 0.2_Cyto:0.8_Lyso 0.821 0.948 1.896 1.378 0.778 1.246 2.648 7.776 1.179
#> 0.3_Cyto:0.7_Lyso 0.773 0.876 1.712 1.257 0.717 1.378 2.407 6.878 1.069
#> 0.4_Cyto:0.6_Lyso 0.725 0.803 1.527 1.137 0.656 1.509 2.166 5.981 0.959
#> 0.5_Cyto:0.5_Lyso 0.677 0.730 1.342 1.016 0.596 1.641 1.925 5.083 0.849
#> 0.6_Cyto:0.4_Lyso 0.630 0.657 1.158 0.896 0.535 1.773 1.683 4.186 0.739
#> 0.7_Cyto:0.3_Lyso 0.582 0.585 0.973 0.775 0.474 1.904 1.442 3.288 0.630
#> 0.8_Cyto:0.2_Lyso 0.534 0.512 0.788 0.655 0.414 2.036 1.201 2.391 0.520
#> 0.9_Cyto:0.1_Lyso 0.487 0.439 0.604 0.534 0.353 2.168 0.960 1.493 0.410
#> 1_Cyto:0_Lyso     0.439 0.366 0.419 0.413 0.293 2.299 0.718 0.596 0.300
```

Next, obtain CPA estimates using the RSA-transformed mixtures and RSA-transformed references profiles. We see that the CPA estimates match the proportions we used to create the mixtures:

```
refLocationProfilesRSA <- RSAfromNSA(NSA=refLocationProfilesNSA,
                           NstartMaterialFractions=6,
                           totProt=totProt)
mixProt1Prot4CPAfromRSA <- fitCPA(profile=mixProt1Prot4RSA,
                       refLocationProfiles=refLocationProfilesRSA,
                       numDataCols=9)
round(mixProt1Prot4CPAfromRSA, digits=3)
```

```
#>                   Cyto ER Golgi Lyso Mito Nuc Perox PM
#> 0_Cyto:1_Lyso      0.0  0     0  1.0    0   0     0  0
#> 0.1_Cyto:0.9_Lyso  0.1  0     0  0.9    0   0     0  0
#> 0.2_Cyto:0.8_Lyso  0.2  0     0  0.8    0   0     0  0
#> 0.3_Cyto:0.7_Lyso  0.3  0     0  0.7    0   0     0  0
#> 0.4_Cyto:0.6_Lyso  0.4  0     0  0.6    0   0     0  0
#> 0.5_Cyto:0.5_Lyso  0.5  0     0  0.5    0   0     0  0
#> 0.6_Cyto:0.4_Lyso  0.6  0     0  0.4    0   0     0  0
#> 0.7_Cyto:0.3_Lyso  0.7  0     0  0.3    0   0     0  0
#> 0.8_Cyto:0.2_Lyso  0.8  0     0  0.2    0   0     0  0
#> 0.9_Cyto:0.1_Lyso  0.9  0     0  0.1    0   0     0  0
#> 1_Cyto:0_Lyso      1.0  0     0  0.0    0   0     0  0
```

We may obtain the normalized specific amounts transformation of `mixProtiProtjRSA` by using the `NSAfromRSA` function:

```
mixProt1Prot4NSA <- NSAfromRSA(mixProt1Prot4RSA)
round(mixProt1Prot4NSA, digits=3)
```

```
#>                       N     M    L1    L2     P     S  Nyc1  Nyc2  Nyc3
#> 0_Cyto:1_Lyso     0.042 0.050 0.104 0.074 0.041 0.045 0.143 0.437 0.064
#> 0.1_Cyto:0.9_Lyso 0.043 0.050 0.103 0.074 0.041 0.055 0.143 0.428 0.064
#> 0.2_Cyto:0.8_Lyso 0.044 0.051 0.102 0.074 0.042 0.067 0.142 0.416 0.063
#> 0.3_Cyto:0.7_Lyso 0.045 0.051 0.100 0.074 0.042 0.081 0.141 0.403 0.063
#> 0.4_Cyto:0.6_Lyso 0.047 0.052 0.099 0.074 0.042 0.098 0.140 0.387 0.062
#> 0.5_Cyto:0.5_Lyso 0.049 0.053 0.097 0.073 0.043 0.118 0.139 0.367 0.061
#> 0.6_Cyto:0.4_Lyso 0.051 0.054 0.094 0.073 0.044 0.145 0.137 0.342 0.060
#> 0.7_Cyto:0.3_Lyso 0.055 0.055 0.091 0.073 0.045 0.179 0.135 0.309 0.059
#> 0.8_Cyto:0.2_Lyso 0.059 0.057 0.087 0.072 0.046 0.225 0.133 0.264 0.057
#> 0.9_Cyto:0.1_Lyso 0.065 0.059 0.081 0.072 0.047 0.291 0.129 0.201 0.055
#> 1_Cyto:0_Lyso     0.075 0.063 0.072 0.071 0.050 0.393 0.123 0.102 0.051
```

We next apply the CPA routine to these normalized specific amount (NSA) profiles:

```
mixProt1Prot4CPAfromNSA <- fitCPA(profile=mixProt1Prot4NSA,
                            refLocationProfiles=refLocationProfilesNSA,
                            numDataCols=9)
round(mixProt1Prot4CPAfromNSA, digits=3)
```

```
#>                    Cyto ER Golgi  Lyso Mito Nuc Perox PM
#> 0_Cyto:1_Lyso     0.000  0     0 1.000    0   0     0  0
#> 0.1_Cyto:0.9_Lyso 0.029  0     0 0.971    0   0     0  0
#> 0.2_Cyto:0.8_Lyso 0.063  0     0 0.937    0   0     0  0
#> 0.3_Cyto:0.7_Lyso 0.103  0     0 0.897    0   0     0  0
#> 0.4_Cyto:0.6_Lyso 0.151  0     0 0.849    0   0     0  0
#> 0.5_Cyto:0.5_Lyso 0.211  0     0 0.789    0   0     0  0
#> 0.6_Cyto:0.4_Lyso 0.286  0     0 0.714    0   0     0  0
#> 0.7_Cyto:0.3_Lyso 0.384  0     0 0.616    0   0     0  0
#> 0.8_Cyto:0.2_Lyso 0.517  0     0 0.483    0   0     0  0
#> 0.9_Cyto:0.1_Lyso 0.706  0     0 0.294    0   0     0  0
#> 1_Cyto:0_Lyso     1.000  0     0 0.000    0   0     0  0
```

Note that the results assign the simulated multi-compartment proteins to the correct Cyto and Lyso compartments, but the estimates of the proportions deviate somewhat from those used to for the simulations.

Now, instead of transforming the Acup mixtures to RSA's or using NSA's, what happens if we just use the Acup mixtures themselves, i.e., the relative amounts? There are significant departures of the CPA estimates, including assignments to compartments not used in the simulations from the proportions used to create the mixtures:

```
mixProt1Prot4CPAfromAcup <-
                  fitCPA(profile=mixProt1Prot4Acup,
                         refLocationProfiles=refLocationProfilesAcup,
                         numDataCols=9)
round(mixProt1Prot4CPAfromAcup, digits=3)
```

```
#>                     Cyto    ER Golgi  Lyso  Mito Nuc Perox    PM
#> 0_Cyto:1_Lyso      0.020 0.012     0 0.908 0.017   0 0.030 0.012
#> 0.1_Cyto:0.9_Lyso  0.156 0.034     0 0.645 0.048   0 0.084 0.033
#> 0.2_Cyto:0.8_Lyso  0.241 0.025     0 0.611 0.036   0 0.062 0.024
#> 0.3_Cyto:0.7_Lyso  0.318 0.011     0 0.619 0.015   0 0.027 0.010
#> 0.4_Cyto:0.6_Lyso  0.482 0.050     0 0.225 0.071   0 0.124 0.048
#> 0.5_Cyto:0.5_Lyso  0.574 0.045     0 0.164 0.064   0 0.111 0.043
#> 0.6_Cyto:0.4_Lyso  0.631 0.019     0 0.258 0.027   0 0.047 0.018
#> 0.7_Cyto:0.3_Lyso  0.723 0.014     0 0.197 0.019   0 0.034 0.013
#> 0.8_Cyto:0.2_Lyso  0.816 0.010     0 0.127 0.014   0 0.024 0.009
#> 0.9_Cyto:0.1_Lyso  0.912 0.008     0 0.045 0.011   0 0.018 0.006
#> 1_Cyto:0_Lyso      1.000 0.000     0 0.000 0.000   0 0.000 0.000
```

Displaying the Acup values using `refLocationProfilesAcup` helps explain these discrepancies. Note that the three Nyc columns, which are important for classifying lysosomal proteins, are very small, which effectively down-weights their importance in the CPA procedure:

```
round(refLocationProfilesAcup, digits=4)
```

```
#>            N      M     L1     L2      P      S  Nyc1   Nyc2   Nyc3
#> Cyto  0.1121 0.0996 0.0032 0.0036 0.0390 0.7424 1e-04 0.0002 0.0021
#> ER    0.2691 0.2075 0.0084 0.0209 0.3681 0.1259 2e-04 0.0004 0.0061
#> Golgi 0.2445 0.1758 0.0050 0.0089 0.4623 0.1035 4e-04 0.0007 0.0026
#> Lyso  0.2341 0.2972 0.0174 0.0141 0.1199 0.3173 6e-04 0.0036 0.0099
#> Mito  0.2474 0.5842 0.0089 0.0048 0.0430 0.1117 1e-04 0.0002 0.0063
#> Nuc   0.8761 0.0385 0.0005 0.0014 0.0229 0.0606 0e+00 0.0001 0.0009
#> Perox 0.2329 0.3273 0.0376 0.0209 0.0888 0.2925 2e-04 0.0005 0.0267
#> PM    0.3925 0.2296 0.0083 0.0136 0.1982 0.1578 6e-04 0.0009 0.0049
```

Plots of the CPA estimated vs actual mixture proportions can be generated as described in the previous tutorial. 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 show in parentheses.

```
par(mfrow=c(1,3))
# In the following, the argument increment.prop is not specified so
#  we use the default value of 0.1
mixturePlot(mixProtiProtjCPA=mixProt1Prot4CPAfromRSA,
            NstartMaterialFractions=6, Loc1=i, Loc2=j,
            errorReturn = TRUE, subTitle="RSA")
```

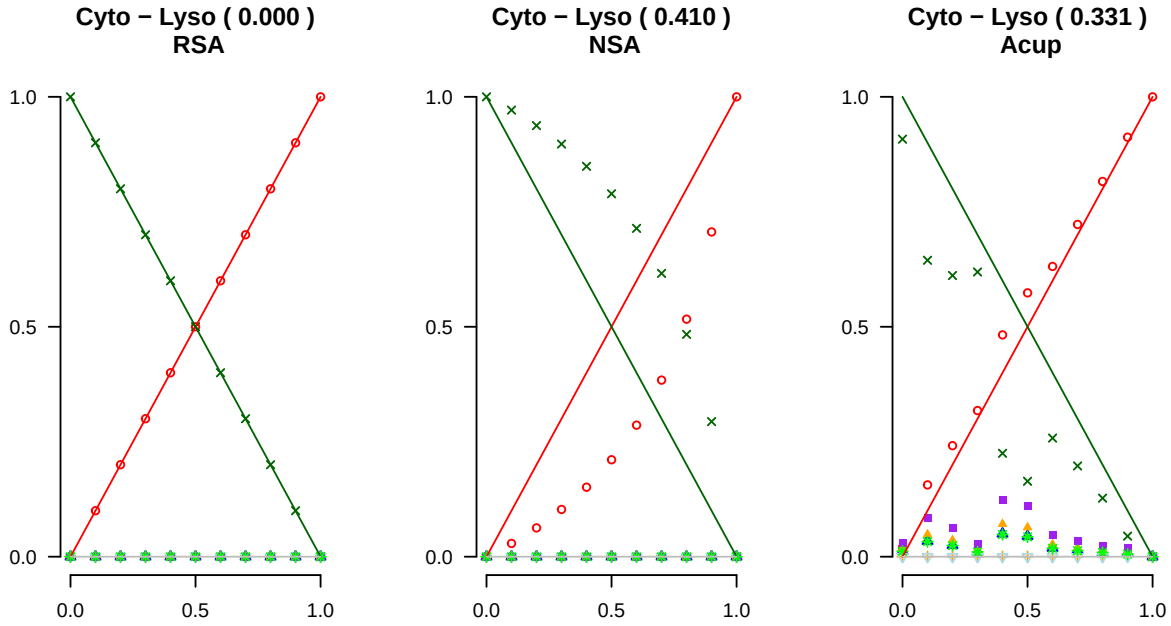
```
#>   Loc1 Loc2    ErrorArea
#> 1 Cyto Lyso 9.154619e-07
```

```
mixturePlot(mixProtiProtjCPA=mixProt1Prot4CPAfromNSA,
            NstartMaterialFractions=6, Loc1=i, Loc2=j,
            errorReturn = TRUE, subTitle="NSA")
```

```
#>   Loc1 Loc2 ErrorArea
#> 1 Cyto Lyso 0.4102294
```

```
mixturePlot(mixProtiProtjCPA=mixProt1Prot4CPAfromAcup,
            NstartMaterialFractions=6, Loc1=i, Loc2=j,
            errorReturn = TRUE, subTitle="Acup")
```

```
#>   Loc1 Loc2 ErrorArea
#> 1 Cyto Lyso 0.3309176
```



We may obtain only the areas indicating the prediction error by using the `mixtureAreaError` function.

```
# As discussed earlier, the argument increment.prop=0.1 does not
#  need to be specified because it is the default
#  and is consistent with the previously generated mixtures.
mixtureAreaError(mixProtiProtjCPA=mixProt1Prot4CPAfromRSA,
            NstartMaterialFractions=6, Loc1=i, Loc2=j)
```

```
#> [1] 9.154619e-07
```

```
mixtureAreaError(mixProtiProtjCPA=mixProt1Prot4CPAfromNSA,
            NstartMaterialFractions=6, Loc1=i, Loc2=j)
```

```
#> [1] 0.4102294
```

```
mixtureAreaError(mixProtiProtjCPA=mixProt1Prot4CPAfromAcup,
            NstartMaterialFractions=6, Loc1=i, Loc2=j)
```

```
#> [1] 0.3309176
```

Applying CPA to log transformations of Cyto and Lyso mixtures

As an alternative, let us apply log2 transformations to the different types of profile data. The log transformation results in a marked improvement for the Acup data and a modest improvement for the NSA data, and poorer results for the RSA data. In all cases, the log transformed data do not provide CPA estimates that are as accurate as were obtained using the RSA transformation alone:

```
eps <- 0.001


mixProt1Prot4CPAfromRSAlog2 <- fitCPA(profile=log2(mixProt1Prot4RSA + eps),
                      refLocationProfiles=log2(refLocationProfilesRSA + eps),
                      numDataCols=9)
round(mixProt1Prot4CPAfromRSAlog2, digits=3)
```

```
#>                     Cyto ER Golgi  Lyso Mito Nuc Perox PM
#> 0_Cyto:1_Lyso      0.000  0     0 1.000    0   0     0  0
#> 0.1_Cyto:0.9_Lyso 0.052  0     0 0.948    0   0     0  0
#> 0.2_Cyto:0.8_Lyso 0.107  0     0 0.893    0   0     0  0
#> 0.3_Cyto:0.7_Lyso 0.167  0     0 0.833    0   0     0  0
#> 0.4_Cyto:0.6_Lyso 0.232  0     0 0.768    0   0     0  0
#> 0.5_Cyto:0.5_Lyso 0.306  0     0 0.694    0   0     0  0
#> 0.6_Cyto:0.4_Lyso 0.389  0     0 0.611    0   0     0  0
#> 0.7_Cyto:0.3_Lyso 0.487  0     0 0.513    0   0     0  0
#> 0.8_Cyto:0.2_Lyso 0.606  0     0 0.394    0   0     0  0
#> 0.9_Cyto:0.1_Lyso 0.760  0     0 0.240    0   0     0  0
#> 1_Cyto:0_Lyso      1.000  0     0 0.000    0   0     0  0
```

```
mixProt1Prot4CPAfromNSAlog2 <-
            fitCPA(profile=log2(mixProt1Prot4NSA + eps),
                refLocationProfiles=log2(refLocationProfilesNSA + eps),
                numDataCols=9)
round(mixProt1Prot4CPAfromNSAlog2, digits=3)
```

```
#>                     Cyto ER Golgi  Lyso Mito Nuc Perox PM
#> 0_Cyto:1_Lyso      0.000  0     0 1.000    0   0     0  0
#> 0.1_Cyto:0.9_Lyso 0.065  0     0 0.935    0   0     0  0
#> 0.2_Cyto:0.8_Lyso 0.130  0     0 0.870    0   0     0  0
#> 0.3_Cyto:0.7_Lyso 0.195  0     0 0.805    0   0     0  0
#> 0.4_Cyto:0.6_Lyso 0.262  0     0 0.738    0   0     0  0
```

```
#> 0.5_Cyto:0.5_Lyso 0.334  0     0 0.666    0   0     0  0
#> 0.6_Cyto:0.4_Lyso 0.413  0     0 0.587    0   0     0  0
#> 0.7_Cyto:0.3_Lyso 0.503  0     0 0.497    0   0     0  0
#> 0.8_Cyto:0.2_Lyso 0.612  0     0 0.388    0   0     0  0
#> 0.9_Cyto:0.1_Lyso 0.757  0     0 0.243    0   0     0  0
#> 1_Cyto:0_Lyso     1.000  0     0 0.000    0   0     0  0
```

```
mixProt1Prot4CPAfromAcupLog2 <-
            fitCPA(profile=log2(mixProt1Prot4Acup + eps),
              refLocationProfiles=log2(refLocationProfilesAcup + eps),
              numDataCols=9)
round(mixProt1Prot4CPAfromAcupLog2, digits=3)
```

```
#>                    Cyto ER Golgi  Lyso Mito Nuc Perox PM
#> 0_Cyto:1_Lyso     0.000  0     0 1.000    0   0 0.000  0
#> 0.1_Cyto:0.9_Lyso 0.066  0     0 0.932    0   0 0.002  0
#> 0.2_Cyto:0.8_Lyso 0.135  0     0 0.861    0   0 0.004  0
#> 0.3_Cyto:0.7_Lyso 0.209  0     0 0.785    0   0 0.006  0
#> 0.4_Cyto:0.6_Lyso 0.288  0     0 0.705    0   0 0.008  0
#> 0.5_Cyto:0.5_Lyso 0.373  0     0 0.618    0   0 0.009  0
#> 0.6_Cyto:0.4_Lyso 0.467  0     0 0.523    0   0 0.010  0
#> 0.7_Cyto:0.3_Lyso 0.572  0     0 0.418    0   0 0.010  0
#> 0.8_Cyto:0.2_Lyso 0.692  0     0 0.299    0   0 0.009  0
#> 0.9_Cyto:0.1_Lyso 0.831  0     0 0.163    0   0 0.006  0
#> 1_Cyto:0_Lyso     1.000  0     0 0.000    0   0 0.000  0
```

Following are plots of the CPA estimated vs actual mixture proportions for simulated proteins where the different profiles (RSA, NSA, and Acup) are log2-transformed.

```
par(mfrow=c(1,3))
mixturePlot(mixProtiProtjCPA=mixProt1Prot4CPAfromRSAlog2,
            NstartMaterialFractions=6,
            Loc1=i, Loc2=j, errorReturn = TRUE,
            subTitle="Log2 RSA")
```

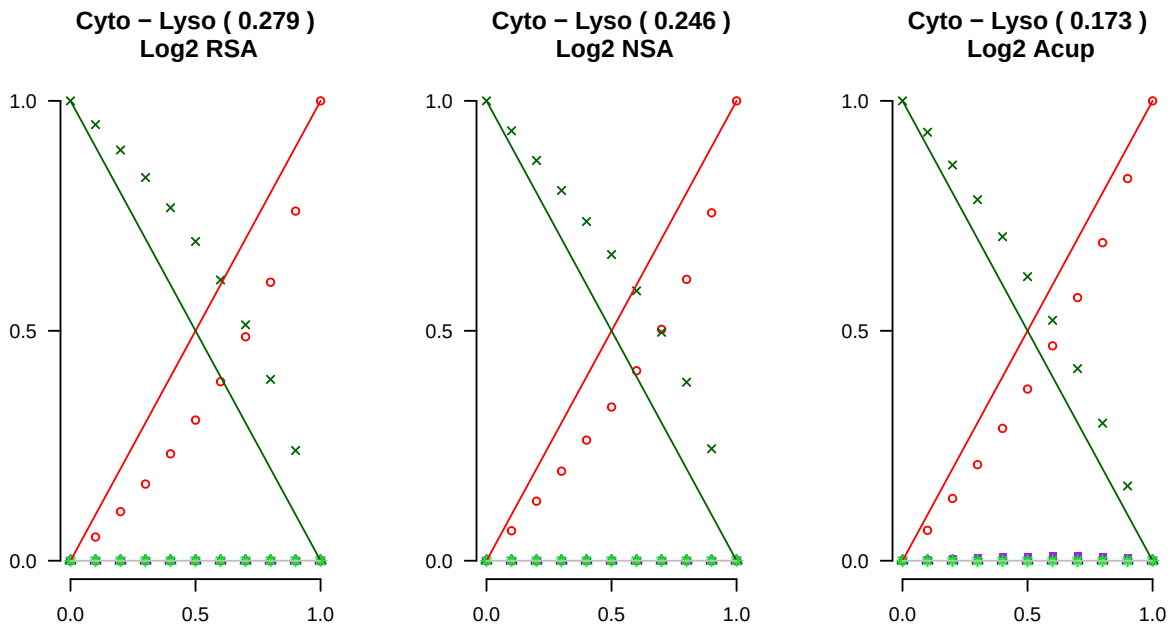
```
#>   Loc1 Loc2 ErrorArea
#> 1 Cyto Lyso 0.2788644
```

```
mixturePlot(mixProtiProtjCPA=mixProt1Prot4CPAfromNSAlog2,
            NstartMaterialFractions=6,
            Loc1=i, Loc2=j, errorReturn = TRUE,
            subTitle="Log2 NSA")
```

```
#>   Loc1 Loc2 ErrorArea
#> 1 Cyto Lyso 0.2458659
```

```
mixturePlot(mixProtiProtjCPA=mixProt1Prot4CPAfromAcupLog2,
            NstartMaterialFractions=6, Loc1=i, Loc2=j,
            errorReturn = TRUE, subTitle="Log2 Acup")
```

```
#>   Loc1 Loc2 ErrorArea
#> 1 Cyto Lyso 0.1732408
```



Applying CPA to RSA, NSA, or Acup data derived from simulated mixtures of Cyto and Nuc

In the previous sections, we showed that one can apply CPA to Cyto-Lyso mixtures using a range of transformations, and we saw that the best results were obtained using RSA values, and the worst results from using relative amounts (Acup transformations). Then we saw that log-transformations of the relative amounts improved the quality of the estimates considerably but decreased accuracy of RSA estimates.

In this section we consider Cyto and Nuc mixtures, which we generate as described above. We shall see that here, Acup-transformed values produce good CPA estimates, just as RSA-transformed values. The reason is that, unlike with Lyso, the Nuc (and Cyto) profiles do not depend on the Nyc portions of the values, and thus the estimated subcellular residence proportions do not suffer from the extremely small Nyc profile values.

Here are the CPA estimates from RSA-transformed profiles:

```
i=1  # Cyto
j=6  # Nuc
mixProt1Prot6Acup <- proteinMix(refLocationProfilesAcup, Loc1=i, Loc2=j)
mixProt1Prot6RSA <- RSAfromAcup(Acup=mixProt1Prot6Acup,
                          NstartMaterialFractions=6, totProt=totProtAT5)


mixProt1Prot6CPAfromRSA <- fitCPA(profile=mixProt1Prot6RSA,
            refLocationProfiles=refLocationProfilesRSA, numDataCols=9)
round(mixProt1Prot6CPAfromRSA, digits=3)
```

```
#>                  Cyto ER Golgi Lyso Mito Nuc Perox PM
#> 0_Cyto:1_Nuc      0.0  0     0    0    0 1.0     0  0
#> 0.1_Cyto:0.9_Nuc  0.1  0     0    0    0 0.9     0  0
```

```
#> 0.2_Cyto:0.8_Nuc  0.2  0     0    0    0 0.8     0  0
#> 0.3_Cyto:0.7_Nuc  0.3  0     0    0    0 0.7     0  0
#> 0.4_Cyto:0.6_Nuc  0.4  0     0    0    0 0.6     0  0
#> 0.5_Cyto:0.5_Nuc  0.5  0     0    0    0 0.5     0  0
#> 0.6_Cyto:0.4_Nuc  0.6  0     0    0    0 0.4     0  0
#> 0.7_Cyto:0.3_Nuc  0.7  0     0    0    0 0.3     0  0
#> 0.8_Cyto:0.2_Nuc  0.8  0     0    0    0 0.2     0  0
#> 0.9_Cyto:0.1_Nuc  0.9  0     0    0    0 0.1     0  0
#> 1_Cyto:0_Nuc      1.0  0     0    0    0 0.0     0  0
```

The simulated proportions are estimated very accurately.

Following Tutorial 3, we may see what happens if we apply the CPA routine to profiles containing NSA data:

```
mixProt1Prot6NSA <- NSAfromRSA(mixProt1Prot6RSA)
mixProt1Prot6CPAfromNSA <-
                    fitCPA(profile=mixProt1Prot6NSA,
                    refLocationProfiles=refLocationProfilesNSA,
                    numDataCols=9)
round(mixProt1Prot6CPAfromNSA, digits=3)
```

```
#>                    Cyto ER Golgi Lyso Mito   Nuc Perox PM
#> 0_Cyto:1_Nuc      0.000  0     0    0    0 1.000     0  0
#> 0.1_Cyto:0.9_Nuc 0.123  0     0    0    0 0.877     0  0
#> 0.2_Cyto:0.8_Nuc 0.240  0     0    0    0 0.760     0  0
#> 0.3_Cyto:0.7_Nuc 0.351  0     0    0    0 0.649     0  0
#> 0.4_Cyto:0.6_Nuc 0.457  0     0    0    0 0.543     0  0
#> 0.5_Cyto:0.5_Nuc 0.558  0     0    0    0 0.442     0  0
#> 0.6_Cyto:0.4_Nuc 0.654  0     0    0    0 0.346     0  0
#> 0.7_Cyto:0.3_Nuc 0.747  0     0    0    0 0.253     0  0
#> 0.8_Cyto:0.2_Nuc 0.835  0     0    0    0 0.165     0  0
#> 0.9_Cyto:0.1_Nuc 0.919  0     0    0    0 0.081     0  0
#> 1_Cyto:0_Nuc      1.000  0     0    0    0 0.000     0  0
```

The estimates of the proportions deviate somewhat from those used to generate the mixtures.

Now do this using the relative amounts for markers ("Acup markers") and simulated protein mixtures (Acup) instead of RSA-transformed values.

```
mixProt1Prot6CPAfromAcup <- fitCPA(profile=mixProt1Prot6Acup,
                                refLocationProfiles=refLocationProfilesAcup,
                                numDataCols=9)
```

Note that, unlike with the Cyto-Lyso mixture, the estimates using Acup profiles of the Cyto-Nuc mixture are very accurate, and superior to those using NSA profiles:

```
round(mixProt1Prot6CPAfromAcup, digits=3)
```

```
#>                   Cyto ER Golgi Lyso Mito Nuc Perox PM
#> 0_Cyto:1_Nuc       0.0  0     0    0    0 1.0     0  0
#> 0.1_Cyto:0.9_Nuc  0.1  0     0    0    0 0.9     0  0
#> 0.2_Cyto:0.8_Nuc  0.2  0     0    0    0 0.8     0  0
#> 0.3_Cyto:0.7_Nuc  0.3  0     0    0    0 0.7     0  0
```

```
#> 0.4_Cyto:0.6_Nuc  0.4  0     0    0    0 0.6     0  0
#> 0.5_Cyto:0.5_Nuc  0.5  0     0    0    0 0.5     0  0
#> 0.6_Cyto:0.4_Nuc  0.6  0     0    0    0 0.4     0  0
#> 0.7_Cyto:0.3_Nuc  0.7  0     0    0    0 0.3     0  0
#> 0.8_Cyto:0.2_Nuc  0.8  0     0    0    0 0.2     0  0
#> 0.9_Cyto:0.1_Nuc  0.9  0     0    0    0 0.1     0  0
#> 1_Cyto:0_Nuc      1.0  0     0    0    0 0.0     0  0
```

The CPA for these different transformations can be plotted versus the true proportions as before:

```
par(mfrow=c(1,3))
mixturePlot(mixProtiProtjCPA=mixProt1Prot6CPAfromRSA,
            NstartMaterialFractions=6, Loc1=i, Loc2=j,
            errorReturn = TRUE, subTitle="RSA")
```

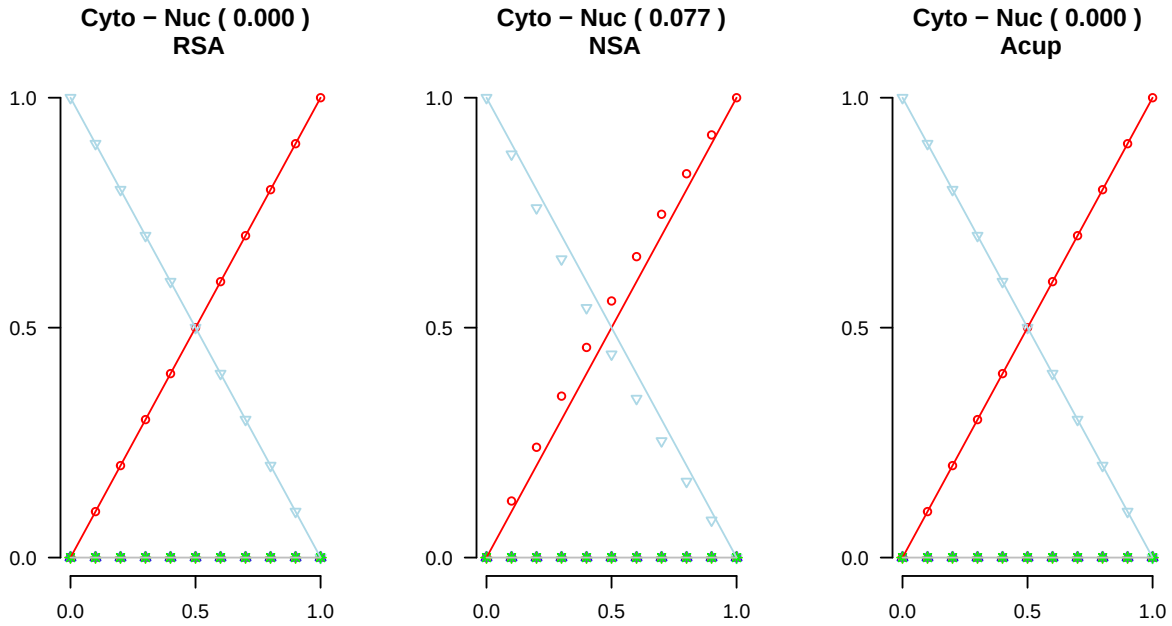
```
#>   Loc1 Loc2   ErrorArea
#> 1 Cyto  Nuc 2.19689e-07
```

```
mixturePlot(mixProtiProtjCPA=mixProt1Prot6CPAfromNSA,
            NstartMaterialFractions=6, Loc1=i, Loc2=j,
            errorReturn = TRUE, subTitle="NSA")
```

```
#>   Loc1 Loc2  ErrorArea
#> 1 Cyto  Nuc 0.07674126
```

```
mixturePlot(mixProtiProtjCPA=mixProt1Prot6CPAfromAcup,
            NstartMaterialFractions=6, Loc1=i, Loc2=j,
            errorReturn = TRUE, subTitle="Acup")
```

```
#>   Loc1 Loc2    ErrorArea
#> 1 Cyto  Nuc 3.955232e-06
```



Applying CPA to log transformations of Cyto and Nuc mixtures

Now try a log2 transformation on different types of simulated protein and marker profiles (RSA's, NSA's, and Acup). Note that in all cases, the CPA values using log-transformed data are less accurate than when using untransformed data.

```
eps <- 0.001

mixProt1Prot6CPAfromRSAlog2 <-
           fitCPA(profile=log2(mixProt1Prot6RSA + eps),
                refLocationProfiles=log2(refLocationProfilesRSA + eps),
                numDataCols=9)
round(mixProt1Prot6CPAfromRSAlog2, digits=3)
```

```
#>                   Cyto ER Golgi  Lyso Mito   Nuc Perox    PM
#> 0_Cyto:1_Nuc     0.000  0     0 0.000    0 1.000 0.000 0.000
#> 0.1_Cyto:0.9_Nuc 0.193  0     0 0.000    0 0.807 0.000 0.000
#> 0.2_Cyto:0.8_Nuc 0.323  0     0 0.002    0 0.674 0.001 0.000
#> 0.3_Cyto:0.7_Nuc 0.408  0     0 0.015    0 0.574 0.004 0.000
#> 0.4_Cyto:0.6_Nuc 0.479  0     0 0.026    0 0.489 0.007 0.000
#> 0.5_Cyto:0.5_Nuc 0.545  0     0 0.035    0 0.412 0.009 0.000
#> 0.6_Cyto:0.4_Nuc 0.610  0     0 0.040    0 0.339 0.011 0.000
#> 0.7_Cyto:0.3_Nuc 0.680  0     0 0.041    0 0.267 0.011 0.002
#> 0.8_Cyto:0.2_Nuc 0.759  0     0 0.034    0 0.191 0.010 0.005
#> 0.9_Cyto:0.1_Nuc 0.858  0     0 0.022    0 0.106 0.007 0.006
#> 1_Cyto:0_Nuc     1.000  0     0 0.000    0 0.000 0.000 0.000
```

```
mixProt1Prot6CPAfromNSAlog2 <-
           fitCPA(profile=log2(mixProt1Prot6NSA + eps),
                refLocationProfiles=log2(refLocationProfilesNSA + eps),
                numDataCols=9)
round(mixProt1Prot6CPAfromNSAlog2, digits=3)
```

```
#>                   Cyto ER Golgi Lyso Mito   Nuc Perox PM
#> 0_Cyto:1_Nuc     0.000  0     0    0    0 1.000     0  0
#> 0.1_Cyto:0.9_Nuc 0.192  0     0    0    0 0.808     0  0
#> 0.2_Cyto:0.8_Nuc 0.322  0     0    0    0 0.678     0  0
#> 0.3_Cyto:0.7_Nuc 0.426  0     0    0    0 0.574     0  0
#> 0.4_Cyto:0.6_Nuc 0.514  0     0    0    0 0.486     0  0
#> 0.5_Cyto:0.5_Nuc 0.593  0     0    0    0 0.407     0  0
#> 0.6_Cyto:0.4_Nuc 0.667  0     0    0    0 0.333     0  0
#> 0.7_Cyto:0.3_Nuc 0.740  0     0    0    0 0.260     0  0
#> 0.8_Cyto:0.2_Nuc 0.814  0     0    0    0 0.186     0  0
#> 0.9_Cyto:0.1_Nuc 0.897  0     0    0    0 0.103     0  0
#> 1_Cyto:0_Nuc     1.000  0     0    0    0 0.000     0  0
```

```
mixProt1Prot6CPAfromAcupLog2 <-
           fitCPA(profile=log2(mixProt1Prot6Acup + eps),
               refLocationProfiles=log2(refLocationProfilesAcup + eps),
               numDataCols=9)
round(mixProt1Prot6CPAfromAcupLog2, digits=3)
```

```
#>                   Cyto ER Golgi  Lyso Mito   Nuc Perox    PM
#> 0_Cyto:1_Nuc     0.000  0     0 0.000    0 1.000 0.000 0.000
#> 0.1_Cyto:0.9_Nuc 0.186  0     0 0.000    0 0.814 0.000 0.000
#> 0.2_Cyto:0.8_Nuc 0.309  0     0 0.000    0 0.691 0.000 0.000
#> 0.3_Cyto:0.7_Nuc 0.407  0     0 0.000    0 0.593 0.000 0.000
#> 0.4_Cyto:0.6_Nuc 0.488  0     0 0.004    0 0.508 0.000 0.000
#> 0.5_Cyto:0.5_Nuc 0.555  0     0 0.015    0 0.431 0.000 0.000
#> 0.6_Cyto:0.4_Nuc 0.620  0     0 0.019    0 0.357 0.004 0.000
#> 0.7_Cyto:0.3_Nuc 0.689  0     0 0.022    0 0.283 0.007 0.000
#> 0.8_Cyto:0.2_Nuc 0.767  0     0 0.021    0 0.204 0.008 0.000
#> 0.9_Cyto:0.1_Nuc 0.863  0     0 0.015    0 0.115 0.007 0.001
#> 1_Cyto:0_Nuc     1.000  0     0 0.000    0 0.000 0.000 0.000
```

Following are plots of the CPA estimated vs actual proportions using the log2-transformed profiles.

```
par(mfrow=c(1,3))
mixturePlot(mixProtiProtjCPA=mixProt1Prot6CPAfromRSAlog2,
            NstartMaterialFractions=6, Loc1=i, Loc2=j,
            errorReturn = TRUE, subTitle="Log2RSA")
```

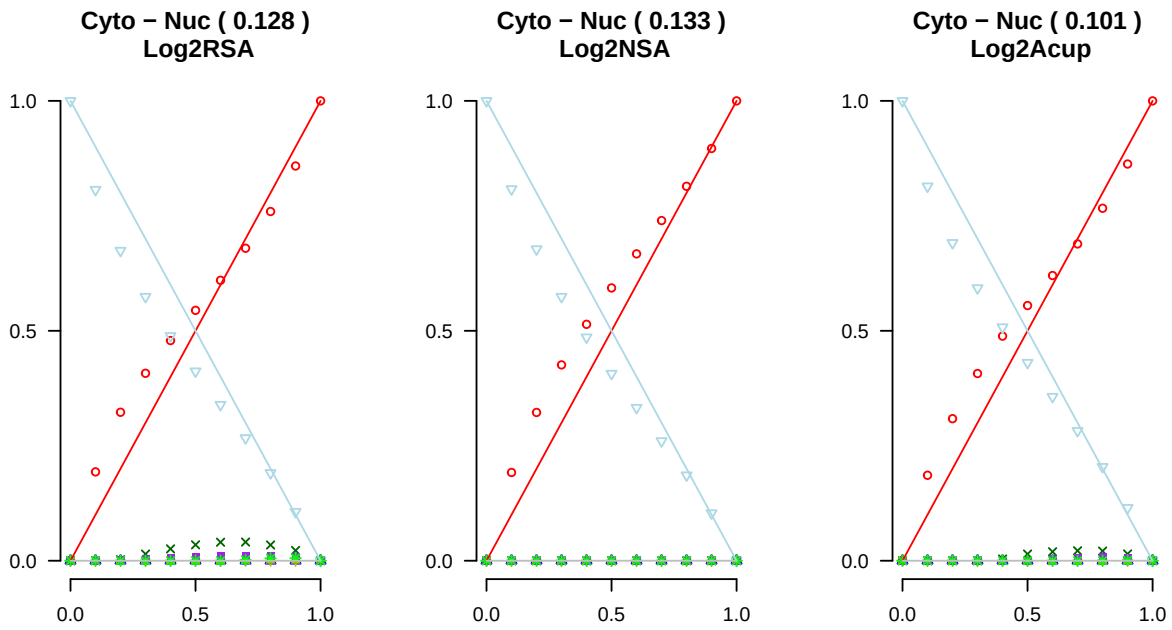
```
#>   Loc1 Loc2 ErrorArea
#> 1 Cyto  Nuc 0.1282022
```

```
mixturePlot(mixProtiProtjCPA=mixProt1Prot6CPAfromNSAlog2,
            NstartMaterialFractions=6, Loc1=i, Loc2=j,
            errorReturn = TRUE, subTitle="Log2NSA")
```

```
#>   Loc1 Loc2 ErrorArea
#> 1 Cyto  Nuc 0.1330879
```

```
mixturePlot(mixProtiProtjCPA=mixProt1Prot6CPAfromAcupLog2,
            NstartMaterialFractions=6, Loc1=i, Loc2=j,
            errorReturn = TRUE, subTitle="Log2Acup")
```

```
#>   Loc1 Loc2 ErrorArea
#> 1 Cyto  Nuc 0.1010286
```



Transformation fitting error heatmaps

We may visualize the area-based errors for all pairwise mixtures using the `mixtureHeatMap` function, which requires prior installation of the `plot.matrix` R library. For each pair of compartments, this function first creates the mixtures as described earlier using the `Acup` markers. Then it computes the three transformations we have discussed, as well as the log2 transformations of these. 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. These 2 by 3 heat maps are arranged in an upper triangular array, with each entry corresponding to a mixture of the indicated row and column compartments.

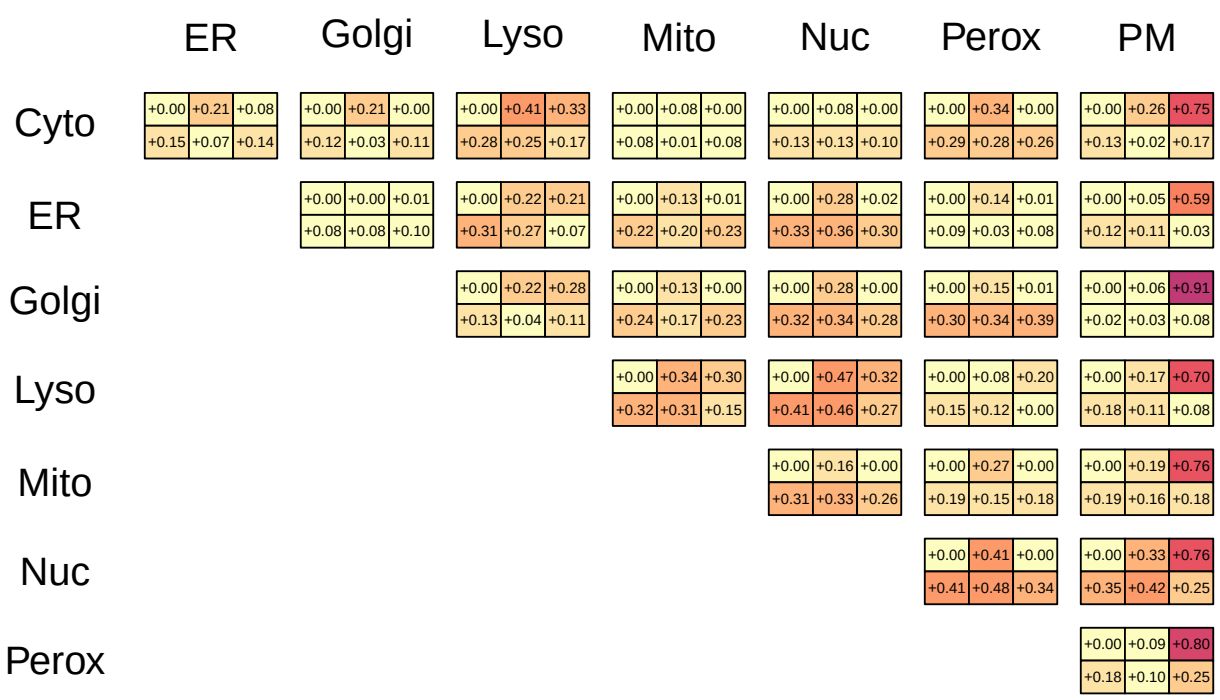
This function requires additional packages which should be installed prior to running the `mixtureHeatMap` function:

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

In addition to plotting a heat map, the `mixtureHeatMap` function returns a two by three matrix of total errors, which comprise a sum across all 28 tables, which we write to a variable named `errorMatAll`:

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

```
#> cpa does not converge for a protein
#>  returning missing values for cpa estimates for that protein
```



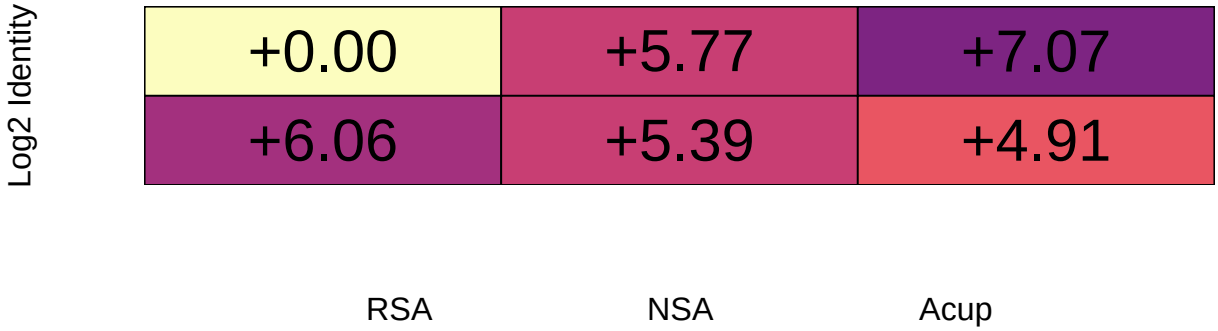
We can view these total errors as follows:

```
errorMatAll
```

```
#>              [,1]     [,2]     [,3]
#> [1,] 6.477323e-05 5.766977 7.071028
#> [2,] 6.064377e+00 5.391389 4.909009
```

Alternatively, we can visualize `errorMatAll` as a heatmap, with darker colors indicating greater errors:

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

col <- rev(viridis::magma(12))
plot(errorMatAll, col=col, breaks=seq(0, 12, 1), key=NULL, main="",
     axis.col=NULL, axis.row=NULL,
     xlab="RSA                          NSA                          Acup",
     ylab="Log2 Identity", digits=2, cex=2, cex.lab=1.1)
par(op) # restore default parameters
```



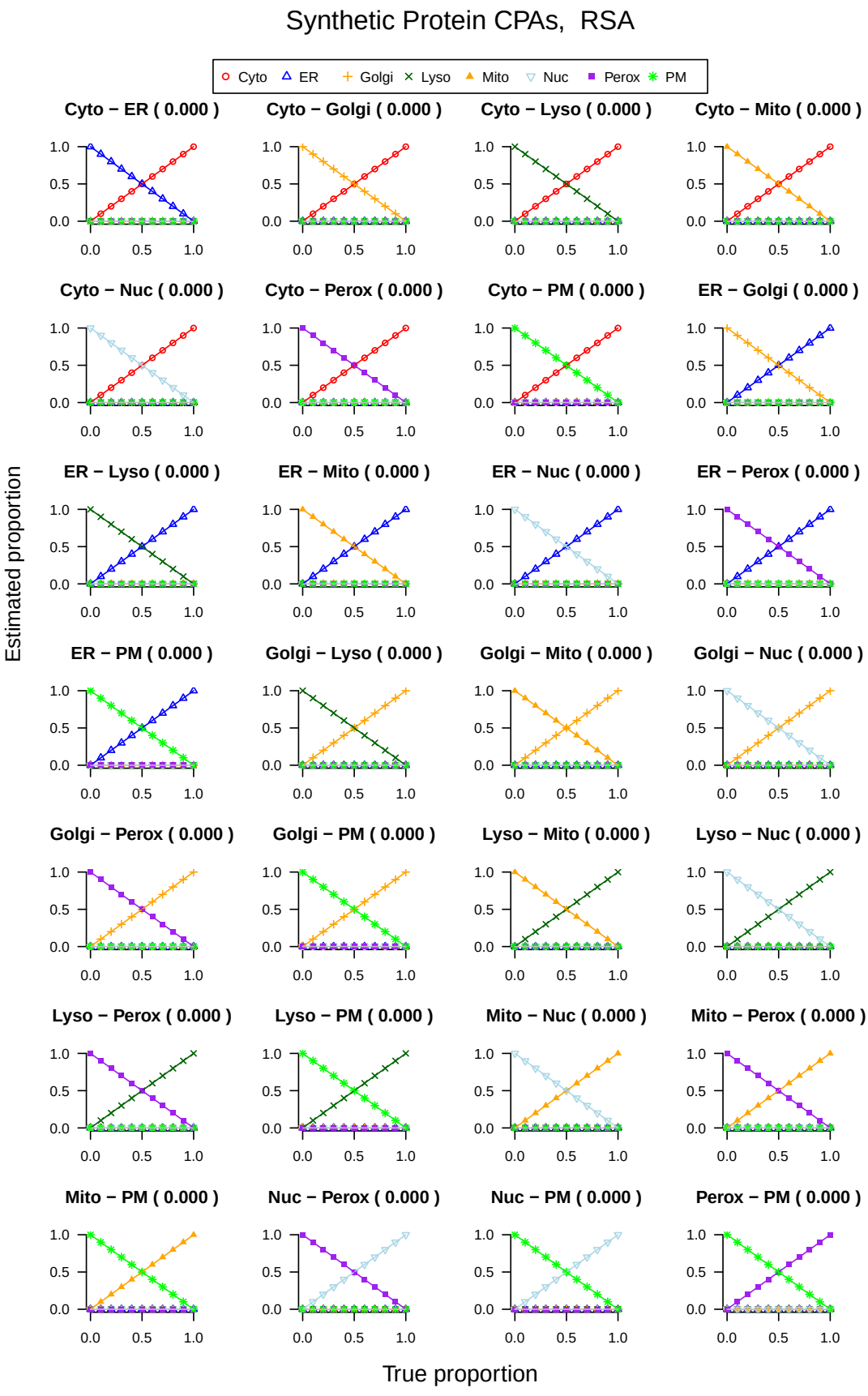
These heatmaps demonstrate that for all combinations of compartments using relative specific amounts without any further log transformation for CPA results in the most accurate estimates of the true mixtures.

Plotting all pairs of mixtures

We may use the function `mixturePlotPanel` to make these plots for all 8*7/2 = 28 possible pairs of mixtures. The function does this for CPA values using different types of mixture profiles, specifying with the argument fitType (e.g., fitType = "RSA", "NSA" or "Acup"). An option is also available to transform either of these using `y = log2(x + eps)`, where `eps` is a small number; by default, `eps = 0.001`, by specifying `log2Transf=TRUE`. The function also can output area-based errors for all 28 mixture pairs by specifying the option `errorReturn = TRUE`. causes the function to return a table of all area-based errors. If specifying this option, we typically write the output to a data frame, which is named `errorAll` in the example below.

We execute the function as follows: (one may first need to open a 7 by 11 window using the "windows" command):

```
# If the function produces a figure margins error message,
#    open a 7 by 11 window:
# windows(width=7, height=11)

errorAllRSAlinear <- mixturePlotPanel(refLocationProfilesAcup=
                                      refLocationProfilesAcup,
              totProt=totProtAT5, NstartMaterialFractions=6, errorReturn = TRUE,
              fitType="RSA", log2Transf=FALSE)
```


Synthetic Protein CPAs, RSA
Cyto
ER
Golgi
Lyso
Mito
Nuc
Perox
PM
Cyto − ER (0.000)
Cyto − Golgi (0.000)
Cyto − Lyso (0.000)
Cyto − Mito (0.000)
Cyto − Nuc (0.000)
Cyto − Perox (0.000)
Cyto − PM (0.000)
ER − Golgi (0.000)
ER − Lyso (0.000)
ER − Mito (0.000)
ER − Nuc (0.000)
ER − Perox (0.000)
ER − PM (0.000)
Golgi − Lyso (0.000)
Golgi − Mito (0.000)
Golgi − Nuc (0.000)
Golgi − Perox (0.000)
Golgi − PM (0.000)
Lyso − Mito (0.000)
Lyso − Nuc (0.000)
Lyso − Perox (0.000)
Lyso − PM (0.000)
Mito − Nuc (0.000)
Mito − Perox (0.000)
Mito − PM (0.000)
Nuc − Perox (0.000)
Nuc − PM (0.000)
Perox − PM (0.000)
Estimated proportion
True proportion
0.0
0.5
1.0

We then can display a table of the errors.

```
errorAllRSAlinear
```

```
#>     Loc1  Loc2    ErrorArea
#> 1   Cyto    ER 2.974605e-06
#> 2   Cyto Golgi 2.071270e-06
#> 3   Cyto  Lyso 9.154619e-07
#> 4   Cyto  Mito 3.297390e-07
#> 5   Cyto   Nuc 2.196890e-07
#> 6   Cyto Perox 7.449870e-07
#> 7   Cyto    PM 3.047688e-06
#> 8     ER Golgi 1.414151e-06
#> 9     ER  Lyso 5.370022e-06
#> 10    ER  Mito 2.835598e-06
#> 11    ER   Nuc 2.440336e-06
#> 12    ER Perox 2.445811e-06
#> 13    ER    PM 3.643631e-06
#> 14 Golgi  Lyso 2.777720e-06
#> 15 Golgi  Mito 4.287843e-06
#> 16 Golgi   Nuc 3.227431e-06
#> 17 Golgi Perox 4.785480e-06
#> 18 Golgi    PM 2.022235e-06
#> 19  Lyso  Mito 1.428861e-06
#> 20  Lyso   Nuc 7.817151e-07
#> 21  Lyso Perox 2.317633e-07
#> 22  Lyso    PM 4.734631e-06
#> 23  Mito   Nuc 1.991091e-06
#> 24  Mito Perox 3.194162e-07
#> 25  Mito    PM 3.513042e-06
#> 26   Nuc Perox 8.034618e-07
#> 27   Nuc    PM 3.294152e-06
#> 28 Perox    PM 2.703337e-06
```

By summing the third column of `errorAll` (the errors), we obtain a global measure of error.

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

```
#> [1] 6.535517e-05
```

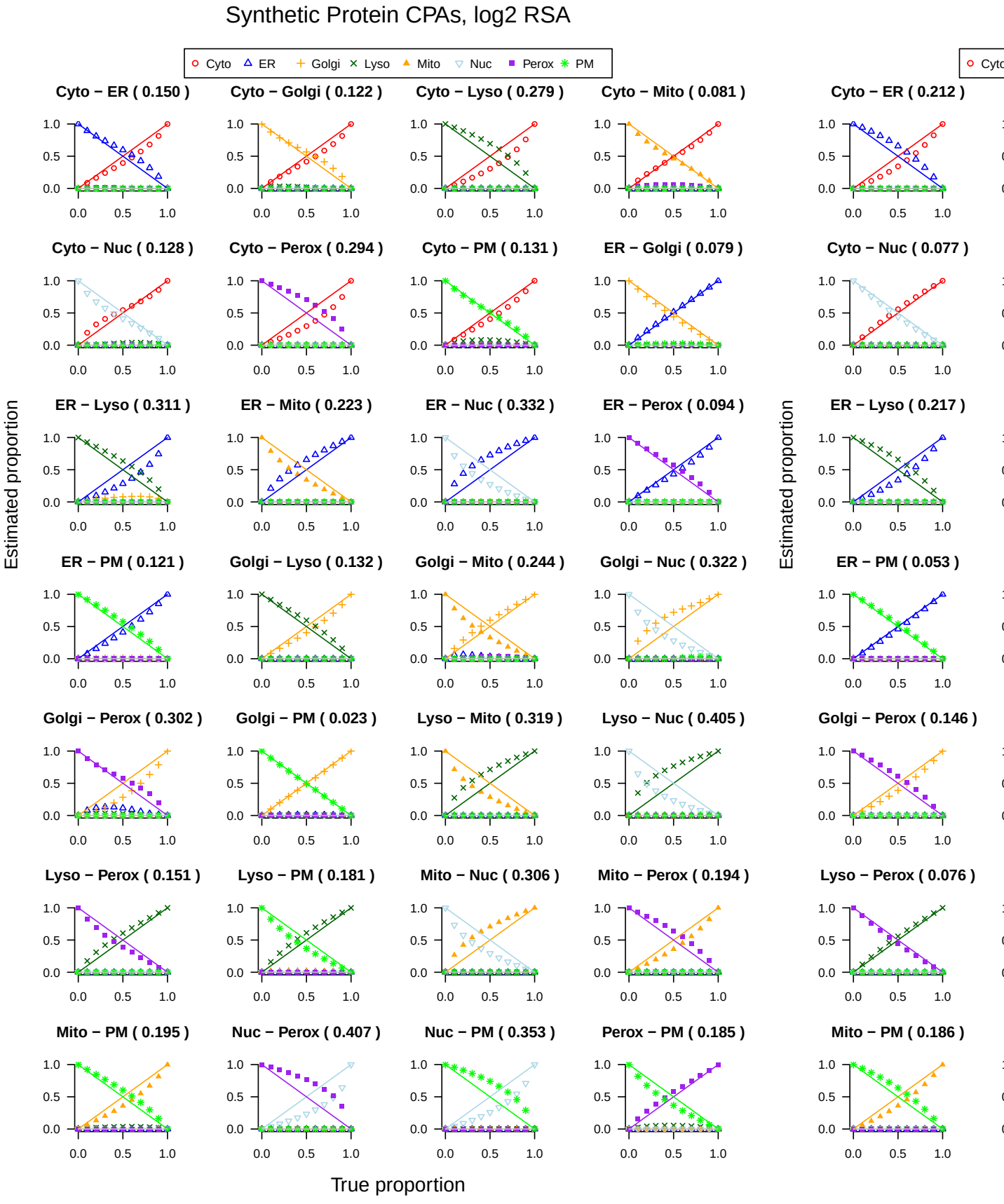
Appendix: CPA plots for all combinations of fit type and functional transformation

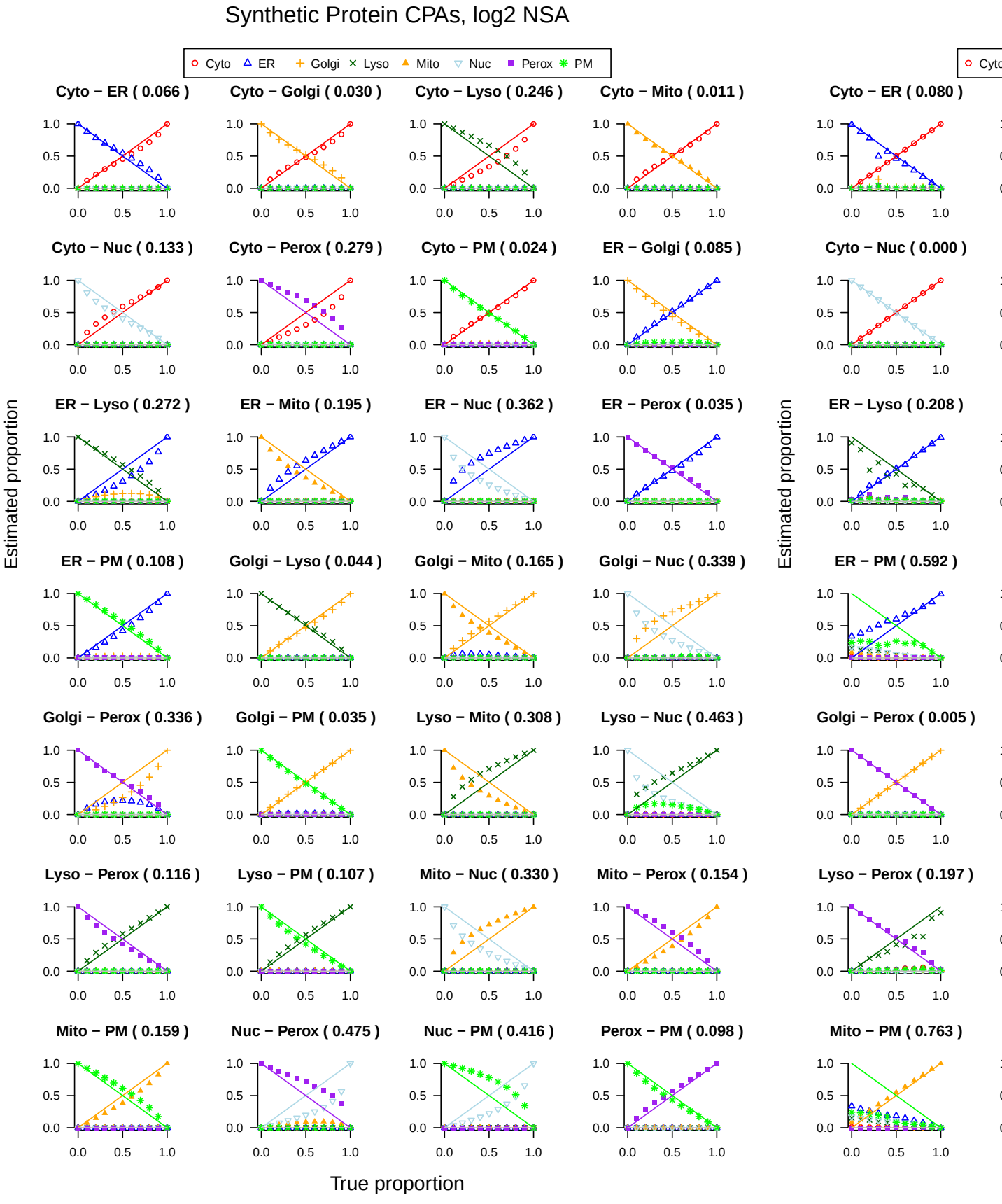
Here we present code that will produce the remaining combinations of fit type (RSA, NSA, and Acup) and log transformation (True or False). First is RSA with log-transformed values (linear RSA having been plotted previously in the tutorial).

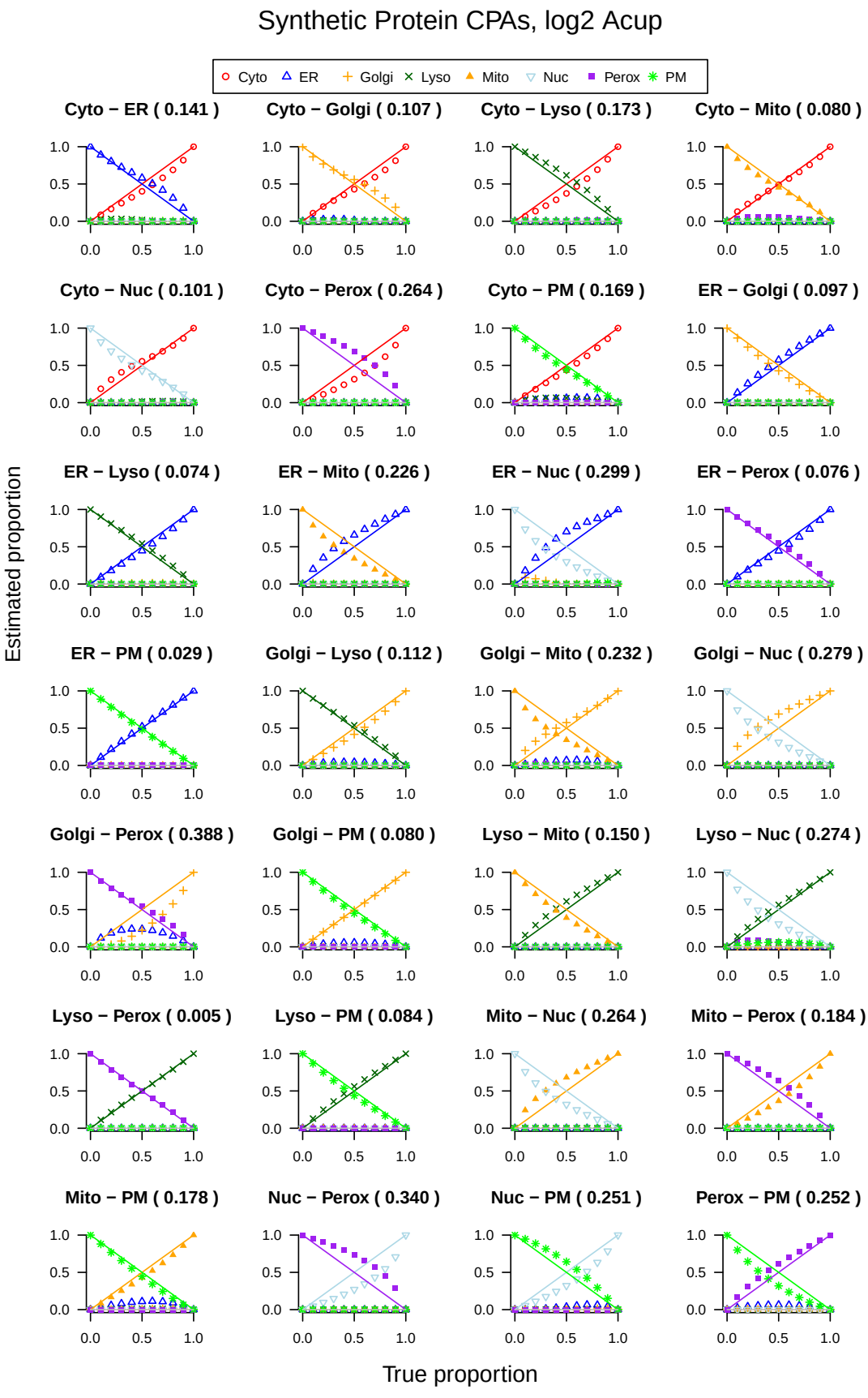
```
fitType <- "RSA"
log2Transf <- TRUE


errorAllRSAlog2 <- mixturePlotPanel(refLocationProfilesAcup=
```

```
              refLocationProfilesAcup, totProt=totProtAT5,
              NstartMaterialFractions=6, errorReturn = TRUE,
              fitType=fitType, log2Transf=log2Transf, eps=0.001)



fitType <- "NSA"
log2Transf <- FALSE

errorAllNSAlinear <- mixturePlotPanel(refLocationProfilesAcup=
                              refLocationProfilesAcup,
              totProt=totProtAT5, NstartMaterialFractions=6,
              errorReturn = TRUE,
              fitType=fitType, log2Transf=log2Transf)

fitType <- "NSA"
log2Transf <- TRUE


errorAllNSAlog2 <- mixturePlotPanel(refLocationProfilesAcup=
                    refLocationProfilesAcup, totProt=totProtAT5,
                    NstartMaterialFractions=6, errorReturn = TRUE,
                    fitType=fitType, log2Transf=log2Transf, eps=0.001)

fitType <- "Acup"
log2Transf <- FALSE


errorAllAcupLinear <- mixturePlotPanel(refLocationProfilesAcup=
              refLocationProfilesAcup, totProt=totProtAT5,
              NstartMaterialFractions=6, errorReturn = TRUE,
              fitType=fitType, log2Transf=log2Transf)


#> cpa does not converge for a protein
#>  returning missing values for cpa estimates for that protein
```

```
fitType <- "Acup"
log2Transf <- TRUE


errorAllAcupLog2 <- mixturePlotPanel(refLocationProfilesAcup=
              refLocationProfilesAcup, totProt=totProtAT5,
              NstartMaterialFractions=6, errorReturn = TRUE,
              fitType=fitType, log2Transf=log2Transf, eps=0.001)
```


Synthetic Protein CPAs, log2 RSA
Cyto
ER
Golgi
Lyso
Mito
Nuc
Perox
PM
Cyto − ER (0.150)
Cyto − Golgi (0.122)
Cyto − Lyso (0.279)
Cyto − Mito (0.081)
Cyto − ER (0.212)
Cyto − Nuc (0.128)
Cyto − Perox (0.294)
Cyto − PM (0.131)
ER − Golgi (0.079)
Cyto − Nuc (0.077)
ER − Lyso (0.311)
ER − Mito (0.223)
ER − Nuc (0.332)
ER − Perox (0.094)
ER − Lyso (0.217)
ER − PM (0.121)
Golgi − Lyso (0.132)
Golgi − Mito (0.244)
Golgi − Nuc (0.322)
ER − PM (0.053)
Golgi − Perox (0.302)
Golgi − PM (0.023)
Lyso − Mito (0.319)
Lyso − Nuc (0.405)
Golgi − Perox (0.146)
Lyso − Perox (0.151)
Lyso − PM (0.181)
Mito − Nuc (0.306)
Mito − Perox (0.194)
Lyso − Perox (0.076)
Mito − PM (0.195)
Nuc − Perox (0.407)
Nuc − PM (0.353)
Perox − PM (0.185)
Mito − PM (0.186)
Estimated proportion
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 (0.066)
Cyto − Golgi (0.030)
Cyto − Lyso (0.246)
Cyto − Mito (0.011)
Cyto − ER (0.080)
Cyto − Nuc (0.133)
Cyto − Perox (0.279)
Cyto − PM (0.024)
ER − Golgi (0.085)
Cyto − Nuc (0.000)
ER − Lyso (0.272)
ER − Mito (0.195)
ER − Nuc (0.362)
ER − Perox (0.035)
ER − Lyso (0.208)
ER − PM (0.108)
Golgi − Lyso (0.044)
Golgi − Mito (0.165)
Golgi − Nuc (0.339)
ER − PM (0.592)
Golgi − Perox (0.336)
Golgi − PM (0.035)
Lyso − Mito (0.308)
Lyso − Nuc (0.463)
Golgi − Perox (0.005)
Lyso − Perox (0.116)
Lyso − PM (0.107)
Mito − Nuc (0.330)
Mito − Perox (0.154)
Lyso − Perox (0.197)
Mito − PM (0.159)
Nuc − Perox (0.475)
Nuc − PM (0.416)
Perox − PM (0.098)
Mito − PM (0.763)
Estimated proportion
Estimated proportion
True proportion


Synthetic Protein CPAs, log2 Acup
Cyto
ER
Golgi
Lyso
Mito
Nuc
Perox
PM
Cyto − ER (0.141)
Cyto − Golgi (0.107)
Cyto − Lyso (0.173)
Cyto − Mito (0.080)
Cyto − Nuc (0.101)
Cyto − Perox (0.264)
Cyto − PM (0.169)
ER − Golgi (0.097)
ER − Lyso (0.074)
ER − Mito (0.226)
ER − Nuc (0.299)
ER − Perox (0.076)
ER − PM (0.029)
Golgi − Lyso (0.112)
Golgi − Mito (0.232)
Golgi − Nuc (0.279)
Golgi − Perox (0.388)
Golgi − PM (0.080)
Lyso − Mito (0.150)
Lyso − Nuc (0.274)
Lyso − Perox (0.005)
Lyso − PM (0.084)
Mito − Nuc (0.264)
Mito − Perox (0.184)
Mito − PM (0.178)
Nuc − Perox (0.340)
Nuc − PM (0.251)
Perox − PM (0.252)
Estimated proportion
True proportion
0.0
0.5
1.0

If one needs to prepare the plots as pdf files which are written to an external directory, first set the working directory, e.g.,

```
setwd("c:\\temp\\myProteinOutput")
```

To output the plots to a pdf file, set things up as follows:

```
pdf(file="CPA assignProts.pdf", width=7, height=11)
```

Then issue the desired calls to `mixturePlotsPanel` as above. All plots requested will be written to the same file. Finally, close out the pdf file to complete writing it out:

```
dev.off()
```

We may show the pairwise and overall errors for all transformations as follows:

```
errorAllRSAlinear
```

```
#>     Loc1  Loc2    ErrorArea
#> 1   Cyto    ER 2.974605e-06
#> 2   Cyto Golgi 2.071270e-06
#> 3   Cyto  Lyso 9.154619e-07
#> 4   Cyto  Mito 3.297390e-07
#> 5   Cyto   Nuc 2.196890e-07
#> 6   Cyto Perox 7.449870e-07
#> 7   Cyto    PM 3.047688e-06
#> 8     ER Golgi 1.414151e-06
#> 9     ER  Lyso 5.370022e-06
#> 10    ER  Mito 2.835598e-06
#> 11    ER   Nuc 2.440336e-06
#> 12    ER Perox 2.445811e-06
#> 13    ER    PM 3.643631e-06
#> 14 Golgi  Lyso 2.777720e-06
#> 15 Golgi  Mito 4.287843e-06
#> 16 Golgi   Nuc 3.227431e-06
#> 17 Golgi Perox 4.785480e-06
#> 18 Golgi    PM 2.022235e-06
#> 19  Lyso  Mito 1.428861e-06
#> 20  Lyso   Nuc 7.817151e-07
#> 21  Lyso Perox 2.317633e-07
#> 22  Lyso    PM 4.734631e-06
#> 23  Mito   Nuc 1.991091e-06
#> 24  Mito Perox 3.194162e-07
#> 25  Mito    PM 3.513042e-06
#> 26   Nuc Perox 8.034618e-07
#> 27   Nuc    PM 3.294152e-06
#> 28 Perox    PM 2.703337e-06
```

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

```
#> [1] 6.535517e-05
```

```
errorAllRSAlog2
```

```
#>     Loc1  Loc2  ErrorArea
#> 1   Cyto    ER 0.15043728
#> 2   Cyto Golgi 0.12192127
#> 3   Cyto  Lyso 0.27886436
#> 4   Cyto  Mito 0.08101614
#> 5   Cyto   Nuc 0.12820222
#> 6   Cyto Perox 0.29446795
#> 7   Cyto    PM 0.13074451
#> 8     ER Golgi 0.07856732
#> 9     ER  Lyso 0.31059706
#> 10    ER  Mito 0.22301257
#> 11    ER   Nuc 0.33247877
#> 12    ER Perox 0.09445397
#> 13    ER    PM 0.12070646
#> 14 Golgi  Lyso 0.13166324
#> 15 Golgi  Mito 0.24392675
#> 16 Golgi   Nuc 0.32170211
#> 17 Golgi Perox 0.30249506
#> 18 Golgi    PM 0.02258567
#> 19  Lyso  Mito 0.31919351
#> 20  Lyso   Nuc 0.40543829
#> 21  Lyso Perox 0.15132232
#> 22  Lyso    PM 0.18098811
#> 23  Mito   Nuc 0.30604745
#> 24  Mito Perox 0.19371272
#> 25  Mito    PM 0.19478244
#> 26   Nuc Perox 0.40748777
#> 27   Nuc    PM 0.35265557
#> 28 Perox    PM 0.18490751
```

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

```
#> [1] 6.064378
```

```
errorAllNSAlinear
```

```
#>     Loc1  Loc2   ErrorArea
#> 1   Cyto    ER 0.212381310
#> 2   Cyto Golgi 0.209410577
#> 3   Cyto  Lyso 0.410229360
#> 4   Cyto  Mito 0.080619428
#> 5   Cyto   Nuc 0.076741255
#> 6   Cyto Perox 0.344852521
#> 7   Cyto    PM 0.262342369
#> 8     ER Golgi 0.003089918
#> 9     ER  Lyso 0.216688009
#> 10    ER  Mito 0.134009854
#> 11    ER   Nuc 0.284556106
#> 12    ER Perox 0.142848836
#> 13    ER    PM 0.052881052
```

```
#> 14 Golgi  Lyso 0.219606792
#> 15 Golgi  Mito 0.130956830
#> 16 Golgi   Nuc 0.281698756
#> 17 Golgi Perox 0.145877445
#> 18 Golgi    PM 0.055951135
#> 19  Lyso  Mito 0.340867748
#> 20  Lyso   Nuc 0.471975036
#> 21  Lyso Perox 0.076133861
#> 22  Lyso    PM 0.165659740
#> 23  Mito   Nuc 0.156415484
#> 24  Mito Perox 0.271713682
#> 25  Mito    PM 0.185579427
#> 26   Nuc Perox 0.410737336
#> 27   Nuc    PM 0.332464745
#> 28 Perox    PM 0.090688532
```

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

```
#> [1] 5.766977
```

```
errorAllNSAlog2
```

```
#>     Loc1  Loc2  ErrorArea
#> 1   Cyto    ER 0.06573459
#> 2   Cyto Golgi 0.03016796
#> 3   Cyto  Lyso 0.24586590
#> 4   Cyto  Mito 0.01141022
#> 5   Cyto   Nuc 0.13308793
#> 6   Cyto Perox 0.27928051
#> 7   Cyto    PM 0.02379466
#> 8     ER Golgi 0.08457692
#> 9     ER  Lyso 0.27244077
#> 10    ER  Mito 0.19532312
#> 11    ER   Nuc 0.36236786
#> 12    ER Perox 0.03462474
#> 13    ER    PM 0.10795633
#> 14 Golgi  Lyso 0.04417529
#> 15 Golgi  Mito 0.16541263
#> 16 Golgi   Nuc 0.33866624
#> 17 Golgi Perox 0.33608386
#> 18 Golgi    PM 0.03492783
#> 19  Lyso  Mito 0.30823696
#> 20  Lyso   Nuc 0.46254892
#> 21  Lyso Perox 0.11638748
#> 22  Lyso    PM 0.10724703
#> 23  Mito   Nuc 0.32983002
#> 24  Mito Perox 0.15406480
#> 25  Mito    PM 0.15853392
#> 26   Nuc Perox 0.47504211
#> 27   Nuc    PM 0.41604026
#> 28 Perox    PM 0.09756025
```

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

```
#> [1] 5.391389
```

```
errorAllAcupLinear
```

```
#>     Loc1  Loc2    ErrorArea
#> 1   Cyto    ER 8.037694e-02
#> 2   Cyto Golgi 1.334372e-04
#> 3   Cyto  Lyso 3.309176e-01
#> 4   Cyto  Mito 4.305178e-05
#> 5   Cyto   Nuc 3.955232e-06
#> 6   Cyto Perox 1.235282e-03
#> 7   Cyto    PM 7.548979e-01
#> 8     ER Golgi 9.335017e-03
#> 9     ER  Lyso 2.084682e-01
#> 10    ER  Mito 1.375684e-02
#> 11    ER   Nuc 1.971540e-02
#> 12    ER Perox 1.281084e-02
#> 13    ER    PM 5.919962e-01
#> 14 Golgi  Lyso 2.807322e-01
#> 15 Golgi  Mito 2.371936e-04
#> 16 Golgi   Nuc 1.352412e-04
#> 17 Golgi Perox 5.249567e-03
#> 18 Golgi    PM 9.138017e-01
#> 19  Lyso  Mito 2.996348e-01
#> 20  Lyso   Nuc 3.184944e-01
#> 21  Lyso Perox 1.967465e-01
#> 22  Lyso    PM 7.045064e-01
#> 23  Mito   Nuc 7.463855e-06
#> 24  Mito Perox 1.123795e-03
#> 25  Mito    PM 7.630772e-01
#> 26   Nuc Perox 1.627324e-03
#> 27   Nuc    PM 7.587303e-01
#> 28 Perox    PM 8.032334e-01
```

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

```
#> [1] 7.071028
```

```
errorAllAcupLog2
```

```
#>     Loc1  Loc2   ErrorArea
#> 1   Cyto    ER 0.140825170
#> 2   Cyto Golgi 0.106875551
#> 3   Cyto  Lyso 0.173240766
#> 4   Cyto  Mito 0.079966536
#> 5   Cyto   Nuc 0.101028618
#> 6   Cyto Perox 0.264301265
#> 7   Cyto    PM 0.169119585
#> 8     ER Golgi 0.097230598
```

```
#> 9     ER  Lyso 0.073983668
#> 10    ER  Mito 0.226081053
#> 11    ER   Nuc 0.298983688
#> 12    ER Perox 0.076137444
#> 13    ER    PM 0.029152628
#> 14 Golgi  Lyso 0.111834712
#> 15 Golgi  Mito 0.231500224
#> 16 Golgi   Nuc 0.279073932
#> 17 Golgi Perox 0.388146901
#> 18 Golgi    PM 0.080304384
#> 19  Lyso  Mito 0.149959891
#> 20  Lyso   Nuc 0.273781712
#> 21  Lyso Perox 0.004566261
#> 22  Lyso    PM 0.084078993
#> 23  Mito   Nuc 0.263623880
#> 24  Mito Perox 0.183946430
#> 25  Mito    PM 0.178065970
#> 26   Nuc Perox 0.340099358
#> 27   Nuc    PM 0.251011737
#> 28 Perox    PM 0.252088538
```

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

```
#> [1] 4.909009
```

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] pracma_2.3.8         protlocassign_0.99.1 lme4_1.1-28
#> [4] 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] plot.matrix_1.6.1   tinytex_0.37        compiler_4.1.3
#> [28] xfun_0.30           pkgconfig_2.0.3     pkgbuild_1.3.1
#> [31] htmltools_0.5.2     tibble_3.1.6        gridExtra_2.3
#> [34] BB_2019.10-1        quadprog_1.5-8      fansi_1.0.2
#> [37] viridisLite_0.4.0   crayon_1.5.0        withr_2.5.0
#> [40] MASS_7.3-55         brio_1.1.3          grid_4.1.3
#> [43] nlme_3.1-155        gtable_0.3.0        lifecycle_1.0.1
#> [46] magrittr_2.0.2      scales_1.1.1        cli_3.2.0
#> [49] stringi_1.7.6       cachem_1.0.6        viridis_0.6.2
#> [52] fs_1.5.2            remotes_2.4.2       testthat_3.1.2
#> [55] ellipsis_0.3.2      vctrs_0.3.8         boot_1.3-28
#> [58] tools_4.1.3         outliers_0.14       glue_1.6.2
#> [61] purrr_0.3.4         processx_3.5.2      pkgload_1.2.4
#> [64] parallel_4.1.3      fastmap_1.1.0       yaml_2.3.5
#> [67] colorspace_2.0-3    sessioninfo_1.2.2   memoise_2.0.1
#> [70] knitr_1.37          usethis_2.1.5
```