ComDim_PCA in R

Francesc Puig-Castellví

19/03/2021

Contents

1 Quick start

```
# Load the functions
source('BuildMultiBlock.R')
source('splitRB.R')
source('comdim_PCA.R')

# Load the data
# (For this quick example, it is simulated)
## Block 1: Matrix of 10 rows and 50 columns, all from the same batch.
b1 = matrix(rnorm(500),10,50)
batch_b1 = rep(1,10)
## Block 2: Matrix of 30 rows and 80 columns, from three different batches.
b2 = matrix(rnorm(800),30,80)
batch_b2 = c(rep(1,10),rep(2,10),rep(3,10))

# Generate the multi-block (mb)
## mb is created from b1.
mb <- BuildMultiBlock(b1, batches = batch_b1)
## b2 is added to mb.
mb <- BuildMultiBlock(b2, growingMB = mb, batches = batch_b2, equalSampleNumber = FALSE)

# Split the each block from mb into smaller blocks (one per batch).
# These smaller blocks are the replicate blocks (rb).
rb <- splitRB(mb)
```

```
## [1] "The sample names are:"
##       b1_R1 b2_R1 b2_R2 b2_R3
##  [1,]     1     1    11    21
##  [2,]     2     2    12    22
##  [3,]     3     3    13    23
##  [4,]     4     4    14    24
##  [5,]     5     5    15    25
##  [6,]     6     6    16    26
##  [7,]     7     7    17    27
##  [8,]     8     8    18    28
##  [9,]     9     9    19    29
## [10,]    10    10    20    30
```

```
# Do ComDim
results<-comdim_PCA(rb, 2) # In this analysis, we used 2 components.
```

```
## Warning: package 'pracma' was built under R version 3.6.3
```

```
##   |                                                                              |
## [1] "Checking sample correspondence across blocks."
## [1] "The data can be used for ComDim."
##   |                                                                              |=====
```

2 Introduction

This is a guide on how to apply ComDim under the R environment.

ComDim is a chemometric method for the analysis of multi-block data sets.

To successfully extract all the potential of the ComDim method, 3 functions programmed in R are proposed in this tutorial:

- **BuildMultiBlock.R**: To merge several single data-blocks into a multi-block data set.
- **splitRB.R**: To split one or more blocks into several smaller blocks, corresponding each new block to one batch.
- **comdim_PCA.R**: This function applies the ComDim algorithm on the multi-block object resulting from **BuildMultiBlock.R** or from **splitRB.R**.

This guide aims to reflect all the capabilities of the three functions. Please read the *Quick Guide* from the previous section for a rapid start. To know more about the versitality of the functions and how to represent the results, continue reading this guide.

3 Building the Multi-Blocks (MBs)

The multi-block object can be built by adding one block at a time with **BuildMultiBlock.R**.

In this guide's example, we will build the multi-block object from the following data:

```
# Block 1: Matrix of 10 rows and 50 columns, filled with random values.
 b1 = matrix(rnorm(500),10,50)
# Block 2: Matrix of 10 rows and 80 columns, filled with random values.
 b2 = matrix(rnorm(800),10,80)
# Block 3: Matrix of 10 rows and 70 columns, filled with random values.
 b3 = matrix(rnorm(700),10,70)
```

These three blocks are stored into the object named **mb**:

```
source('BuildMultiBlock.R')                # To load the function.
mb <- BuildMultiBlock(b1)                  # mb is created from b1.
mb <- BuildMultiBlock(b2, growingMB = mb) # b2 is added to the existing mb object.
mb <- BuildMultiBlock(b3, growingMB = mb) # b3 is added to the existing mb object.
str(mb) # To show the content of mb
```

```
## List of 3
##  $ b1:List of 3
##   ..$ Data     : num [1:10, 1:50] -1.934 1.293 -0.02 0.262 -1.174 ...
##   ..$ Variables: num [1:50] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Samples  : num [1:10] 1 2 3 4 5 6 7 8 9 10
##  $ b2:List of 3
##   ..$ Data     : num [1:10, 1:80] -0.49749 1.84495 0.25926 0.00537 0.13239 ...
##   ..$ Variables: num [1:80] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Samples  : num [1:10] 1 2 3 4 5 6 7 8 9 10
##  $ b3:List of 3
##   ..$ Data     : num [1:10, 1:70] -0.3263 -0.6867 0.3502 0.6277 0.0826 ...
##   ..$ Variables: num [1:70] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Samples  : num [1:10] 1 2 3 4 5 6 7 8 9 10
```

The **mb** object is a "list of lists" (or "nested list") that contains 3 elements: the blocks **b1**, **b2**, and **b3**. Then, at the same time, these blocks contain 3 elements: **Data** (the matrix values), **Variables** (the variable names), and **Samples** (the sample names).

Here are some ways to consult the content of **mb**.

```
str(mb[[1]])                              # To overview of b1
```

```
## List of 3
##  $ Data     : num [1:10, 1:50] -1.934 1.293 -0.02 0.262 -1.174 ...
##  $ Variables: num [1:50] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Samples  : num [1:10] 1 2 3 4 5 6 7 8 9 10
```

```
mb[[2]]$Data                              # To Check values in b2 matrix
```

```
##              [,1]        [,2]       [,3]        [,4]        [,5]       [,6]
##  [1,] -0.49749116 -0.54978480 -0.4376225 -1.39080189 -0.75902985  0.8094914
##  [2,]  1.84494731  1.17322753  0.9228069  1.13519408 -0.51291123  0.6945164
##  [3,]  0.25925712  0.04310666 -2.0693758 -1.95543239 -0.89678946 -0.6452140
##  [4,]  0.00536525  0.32763736  1.4545602 -0.57964443 -0.53546491 -2.1767145
##  [5,]  0.13239147 -0.07397687 -0.5669883 -1.88447792 -0.70089291  0.3202081
##  [6,] -0.34507434 -0.85897853  1.6915741  1.05431219  0.62483885 -0.5051128
##  [7,]  1.26011736  0.75239830  0.7303685 -0.31120038  0.62013448 -0.8308220
##  [8,] -1.88572441  1.25808525  0.4870537 -0.82796594  2.38689534 -0.2471541
##  [9,]  1.51073812 -0.48162840 -0.3694111  0.05668706  0.09058247 -0.5057756
## [10,] -0.35802730 -0.05312422 -0.1481046  0.25169513 -0.29927030  0.4413662
##              [,7]       [,8]       [,9]       [,10]       [,11]       [,12]
##  [1,]  0.4396802 -0.7857719 -2.0762357  0.79068210  0.68399878  0.37482993
##  [2,]  0.5204997 -0.7272427  0.8467880  1.39599490  0.06799165  0.07821213
##  [3,] -1.4783943  1.4116214  0.2232156 -0.02598626 -1.90238847  0.95634325
##  [4,] -0.2518573 -1.5787041 -0.9220531  0.09703309  0.33312624  0.53058981
##  [5,]  0.4247272 -0.5965190 -0.9184428  0.45300765 -0.80647776  1.20841152
##  [6,] -1.2372376 -0.9795973  0.9892646  1.12889645 -1.00372729 -0.20520761
##  [7,]  1.1707185 -0.1830136  1.4275453 -0.58850143 -0.83437051 -0.07820784
##  [8,]  1.0059648  0.2667924  1.6297125  0.20499844 -0.88979412  1.53304888
##  [9,] -0.3181668 -0.1012472  1.1838421 -1.64742563  1.41913837  0.85817936
## [10,]  0.6076056 -0.4651706 -0.6314819 -1.66180242 -0.49096794  0.77924056
##             [,13]      [,14]      [,15]       [,16]      [,17]       [,18]
##  [1,] -0.6576937  0.2131094 -0.7141735  1.59165307 -1.0133512  0.67853924
##  [2,] -0.1716660 -2.4267080  0.2104557 -0.09626215  0.3867863  0.55323065
##  [3,]  1.1310901  0.0387633 -1.1969507 -1.40085233 -0.4129297 -0.03534562
##  [4,]  0.2869895  0.1959075  0.7768877 -0.62940907  0.2802799  1.27306406
##  [5,]  0.7607929  0.2811046 -0.2179369  1.15096321 -1.1481778 -0.65559178
##  [6,] -0.9940227  0.2462468 -0.2152095 -0.36686163  0.7440792  1.58640488
##  [7,] -0.2101040 -1.3540363  1.5794987  0.76586691 -0.2628760  0.77261894
##  [8,] -0.2520386  0.1204695 -0.5473661 -0.66488539  0.2295431  1.73716764
##  [9,]  0.5633500  1.0642745  0.1435902 -0.79247958  0.5515127  0.94045756
## [10,] -0.1642242 -0.1101115 -0.7134744 -0.35129167 -1.5797022 -0.74977734
##             [,19]       [,20]        [,21]       [,22]       [,23]       [,24]
##  [1,] -0.3435444 -0.28403832 -0.209421612 -0.94901815  0.24085438 -3.56546758
##  [2,]  1.3831321  0.12430314 -0.405135385 -0.40304875  3.09049181 -0.02031757
##  [3,] -0.4974586 -0.45267703  0.652979547  0.36285730  0.38107028  1.07583044
##  [4,] -0.7280114  0.97278733 -0.007276479 -0.81376270 -0.36246542 -0.17797460
```

```
##  [5,] -0.3588814 -0.27001734  0.470659467 -0.12456586 -0.44806604  0.90726917
##  [6,] -0.1584106  0.70243957 -0.818406507  0.06731867  0.30973089  1.92380314
##  [7,]  2.0494921  0.01607848  0.492807709  0.73900197  1.28335208  0.73678662
##  [8,]  1.8085854 -1.49908986 -0.354567368 -1.43977979 -1.13388702  0.19704994
##  [9,]  3.0783796  1.03002361  0.878442336  0.93644438 -0.02241596  0.50479815
## [10,] -1.0435984 -2.00430512 -0.093251350  0.52296127  2.28701556  1.83499969
##           [,25]      [,26]      [,27]      [,28]       [,29]      [,30]
##  [1,]  1.0803098 -0.2709516 -0.4505596 -1.2458351  1.01478701  0.5955444
##  [2,] -1.0984444 -0.2353374  0.3783422  0.5266843  0.28205120  0.3545815
##  [3,] -1.2397207 -0.6656729  2.0387299  0.6865393  0.77623523  0.2574353
##  [4,] -0.3999895  0.7165500 -0.5778062 -1.9429861  1.54655651 -1.7121784
##  [5,] -1.1183589  0.7367005  1.2405424  1.3773467 -0.70623391 -0.6549219
##  [6,]  2.3141582  1.6379133 -0.7240106  1.5880407 -0.77391362  0.3178245
##  [7,]  1.6702286 -1.4486289 -1.0775163 -0.5053823 -1.20257306  0.5493226
##  [8,] -0.7992809 -1.8238338  1.3768943 -0.2361238  0.04980158  0.1039445
##  [9,]  0.6704676 -1.1184750 -0.1683777  0.1478018 -1.13831151 -0.7704428
## [10,] -1.3092267  1.7907381 -0.4381742  0.3522512  0.36288765 -1.2849464
##            [,31]      [,32]      [,33]       [,34]      [,35]       [,36]
##  [1,] -1.58104019 -0.2826050 -0.6186233 -0.37047479 -0.7353166 -0.9359574
##  [2,] -0.48460965  0.5456288 -0.3915776 -1.42540916  0.8649132  1.8994852
##  [3,] -0.15161393 -0.6898492 -3.4272023 -2.18931828  0.6346782 -1.1349297
##  [4,]  0.34316172  2.0597532 -0.5524279  0.83953048  0.0741285 -0.7479593
##  [5,]  0.03170306 -0.1117867 -0.2446954 -0.27402246 -0.5598597  0.8952191
##  [6,] -0.74301002  0.4869046  1.4139290 -0.07633408  1.3358815  1.3994651
##  [7,] -0.34632262 -0.1580252 -0.7518682  1.28209010 -1.0744291  0.6427468
##  [8,] -0.05858485  1.9015647  0.1098279  0.04225518  0.3392788  2.1147013
##  [9,] -1.00698610  0.5499615  0.3521661  1.43168728  0.7171107 -1.3219533
## [10,]  0.76728856  0.1326966  0.8498958 -0.50163165  0.5705802  1.8563398
##           [,37]       [,38]      [,39]      [,40]       [,41]      [,42]
##  [1,] -0.1557535  0.62972233  1.9677102  0.2813951  0.31445116  0.9710274
##  [2,]  0.4328749 -0.35642306  1.4705457  0.4925677 -0.51883097  0.1023186
##  [3,] -1.4448496  0.21877863 -0.7711513  0.8605169  0.33232931 -0.1286925
##  [4,]  1.1904131 -1.73369693 -0.7819742  0.5962612  0.03343445 -1.1720947
##  [5,] -0.2192413  0.45713785 -0.2532747  0.7680439  0.47813691 -0.7688787
##  [6,]  2.8623257 -1.38589937 -0.6066036  1.2296826 -1.77501561 -1.6588894
##  [7,] -2.0216607  1.49663196  0.1756807  0.4023216 -0.30585306 -0.3458247
##  [8,] -0.7395301 -0.40672360  0.6304631 -1.8764050  1.38828186 -1.2588960
##  [9,]  2.8176584  0.18988064 -0.6416095 -1.4420422  0.41317676  0.2680968
## [10,] -0.5113731  0.07942885 -1.3152025 -0.8990692 -1.93739899 -0.5650264
##            [,43]      [,44]      [,45]      [,46]      [,47]       [,48]
##  [1,]  0.09503009 -0.5188458 -0.3861624 -2.3966839  0.4647319  1.71573936
##  [2,]  0.98242287  0.7206267  0.4664917 -0.9504760  1.7258127 -0.64904163
##  [3,] -0.50974365 -1.4089770 -1.1349020 -0.5157700  0.5840906 -0.91099169
##  [4,] -0.05824706 -0.2976816  1.0455860  0.1406822 -0.5256240  2.35644751
##  [5,]  0.94897485 -1.9873362 -0.6518191 -0.9194095  0.3209142 -0.26041320
##  [6,] -0.82433815 -1.2421166  0.8489192  0.2430664 -0.1671760 -0.01050722
##  [7,]  0.91485760  0.3610199  0.6798236  0.6518837 -0.6688038 -0.15201348
##  [8,] -1.27380674  0.2890418 -1.4367236 -0.8223132  1.1637779 -1.30675750
##  [9,] -0.91787826 -0.4171209 -1.3782627 -0.7182594 -0.7916382  1.27993621
## [10,] -1.04420241 -0.6713538  0.1662823 -0.7727626 -1.0793926  0.89973569
##           [,49]      [,50]       [,51]        [,52]       [,53]        [,54]
##  [1,]  1.0734610 -0.2490429  0.81253365  0.143547548 -1.22411626 -0.012267694
##  [2,] -0.2156093 -1.6882106  1.30419692  0.195061382  0.96356211 -0.645616718
##  [3,]  0.6563387  0.7907132  0.52648489 -0.070314580 -0.05585682 -0.030212300
```

```
##  [4,] -0.6446664 -1.2521455 -0.71068116 -0.752397181 -1.54212114  0.820772475
##  [5,]  0.6082391 -0.9937086  0.01190573 -0.677826213 -0.03459919  0.558745992
##  [6,]  1.0068146 -0.6820108 -0.07307675 -0.004782536  0.93964410 -0.005520795
##  [7,] -0.9818140 -2.5575636 -1.34547441  0.856966603 -0.36996422  0.767600392
##  [8,]  1.4463058 -0.4589163 -0.30546920 -0.193071203 -1.04014647  0.969216589
##  [9,]  1.3213091  0.3901681 -0.36714561  0.889009141  1.57189062  0.369306622
## [10,] -1.0517286  1.3863058  1.80944134  1.717764511 -0.29295648 -0.084514595
##             [,55]        [,56]       [,57]      [,58]      [,59]        [,60]
##  [1,] -0.30047698 -0.131693318 -0.14783523  1.7591706 -1.1513845  0.864376683
##  [2,]  1.54788707 -0.025387707 -0.23168345  1.4626088  0.3011123 -1.116141135
##  [3,] -1.21606320  0.002639726  2.29543455  0.2018709  0.7837639 -0.008246575
##  [4,] -0.06819395  0.245789338  0.28317954 -0.9526384  1.4662012  1.121977085
##  [5,] -1.17991116 -0.303483080 -0.89963609  0.1732294 -0.2415200  1.132401823
##  [6,]  0.09113827 -0.266780629 -0.28431039  0.6551305  0.4404200 -0.067820703
##  [7,] -0.58923470 -0.149566399  0.04124026  0.9989553 -0.5753614 -0.470710158
##  [8,] -1.19022189 -0.145374861  0.15903017  1.2334175 -1.1216240  1.262584333
##  [9,]  0.97516703 -3.065398456 -0.72143495  0.9504912  0.7024958 -0.037587589
## [10,] -0.06857073 -0.194568342 -1.19462164 -0.3251091 -1.4574718 -0.844262826
##             [,61]       [,62]      [,63]       [,64]      [,65]      [,66]
##  [1,]  0.7447219 -1.17776935  1.1791562 -0.15228648 -0.6101461 -1.5628570
##  [2,]  0.4918702  1.27392674  0.6541587  0.14191647 -1.0705352 -0.6364338
##  [3,] -0.9508562 -0.52692960  1.2542916  1.02790127 -1.8052043 -0.7996837
##  [4,] -0.5895866  0.26670912 -0.1658820  0.61320492  0.8077760 -0.4570090
##  [5,] -0.8202345 -0.26521315  1.8556688  0.06244798  1.2268736 -1.1281415
##  [6,]  1.0793464 -0.54848525  2.4441915 -1.01204971  1.2137380  1.4496536
##  [7,] -1.8620452 -1.06887042  2.5266335 -0.30530743  2.0150536  0.5655202
##  [8,] -0.9049131 -0.02649498 -1.6667106 -0.18477613  0.5593329  0.7148810
##  [9,]  1.0787841  0.86346794 -2.1986304 -1.71935862  0.3376478  1.3035556
## [10,]  0.2603570  1.88206539  1.1172500  0.64655994 -1.5389755  0.9007369
##             [,67]       [,68]       [,69]       [,70]      [,71]      [,72]
##  [1,]  1.92405774  1.52841107 -0.04349297  0.60974430 -0.7935816 -0.3776018
##  [2,]  0.03923253  1.52342466  0.14262667  2.11620751  0.6334356  0.3703459
##  [3,] -0.58402065 -0.34807192  0.98733658  0.53787191  0.6216641 -0.1383519
##  [4,]  0.78101327  1.29062471 -0.41018717 -0.16485378  0.7721799 -0.8208353
##  [5,]  0.32294967  0.69689125 -0.03942016 -1.16954061  0.6666712  0.5535722
##  [6,] -1.20226480  0.09044004 -0.52978884 -1.00480691  0.4660562 -1.1728949
##  [7,]  0.18356597  0.12836601  0.30320307  0.04913177  0.8654932 -0.1042334
##  [8,]  0.76698265  1.47699124  0.44717024  0.95173166 -0.6009023 -1.8881465
##  [9,] -0.03278841  0.78884074  0.58514947 -0.04291863 -2.0358668 -0.1078838
## [10,]  0.10390741  0.02408946  0.04083453 -0.28793519 -0.1461606  0.3484819
##             [,73]      [,74]       [,75]      [,76]      [,77]      [,78]
##  [1,]  1.5587189  0.2055185 -0.578876690  1.2581097  0.3374880  0.4864911
##  [2,]  0.4442001 -0.5711558 -0.422142373  0.1504374  1.6892256  0.0189059
##  [3,] -1.8618736 -0.4808484 -0.446893551  0.7644845 -0.1041426 -0.3354641
##  [4,]  1.5124627 -0.1687853 -0.266563744  0.1808127 -1.3531825 -0.9307962
##  [5,]  0.7333389  0.9702380 -0.982070925 -0.1863850 -0.4371151 -0.3755854
##  [6,]  0.1400260  0.3421322  0.870699626 -1.6235460  0.5054491 -0.1387680
##  [7,] -1.7684440  1.3105608 -1.427560009 -0.6154741 -0.6182477 -0.5613128
##  [8,]  0.8593289 -0.4740121  0.853264535  0.0467195  1.1737257 -0.3280387
##  [9,] -0.5049132  1.1517703 -0.005190369  0.8428874  1.3190428  0.1178909
## [10,]  0.5285347  0.9277631 -0.809457609 -0.1646248  0.6141446 -2.1404548
##             [,79]      [,80]
##  [1,] -0.69379338 -0.4697479
##  [2,]  0.53257829  1.2071504
```

```
##  [3,]  0.32494637 -1.3916363
##  [4,] -0.46971579  0.3833679
##  [5,] -1.81898652  1.6213089
##  [6,]  0.07754022 -3.1302245
##  [7,] -0.12657967  0.6059834
##  [8,] -1.53504161  0.4199102
##  [9,]  0.40286342 -0.8452174
## [10,] -1.21387461 -1.2910836
```

```
mb[['b2']]$Data                          # I can subset by the block's name.
```

```
##             [,1]        [,2]       [,3]        [,4]        [,5]       [,6]
##  [1,] -0.49749116 -0.54978480 -0.4376225 -1.39080189 -0.75902985  0.8094914
##  [2,]  1.84494731  1.17322753  0.9228069  1.13519408 -0.51291123  0.6945164
##  [3,]  0.25925712  0.04310666 -2.0693758 -1.95543239 -0.89678946 -0.6452140
##  [4,]  0.00536525  0.32763736  1.4545602 -0.57964443 -0.53546491 -2.1767145
##  [5,]  0.13239147 -0.07397687 -0.5669883 -1.88447792 -0.70089291  0.3202081
##  [6,] -0.34507434 -0.85897853  1.6915741  1.05431219  0.62483885 -0.5051128
##  [7,]  1.26011736  0.75239830  0.7303685 -0.31120038  0.62013448 -0.8308220
##  [8,] -1.88572441  1.25808525  0.4870537 -0.82796594  2.38689534 -0.2471541
##  [9,]  1.51073812 -0.48162840 -0.3694111  0.05668706  0.09058247 -0.5057756
## [10,] -0.35802730 -0.05312422 -0.1481046  0.25169513 -0.29927030  0.4413662
##             [,7]       [,8]       [,9]       [,10]       [,11]       [,12]
##  [1,]  0.4396802 -0.7857719 -2.0762357  0.79068210  0.68399878  0.37482993
##  [2,]  0.5204997 -0.7272427  0.8467880  1.39599490  0.06799165  0.07821213
##  [3,] -1.4783943  1.4116214  0.2232156 -0.02598626 -1.90238847  0.95634325
##  [4,] -0.2518573 -1.5787041 -0.9220531  0.09703309  0.33312624  0.53058981
##  [5,]  0.4247272 -0.5965190 -0.9184428  0.45300765 -0.80647776  1.20841152
##  [6,] -1.2372376 -0.9795973  0.9892646  1.12889645 -1.00372729 -0.20520761
##  [7,]  1.1707185 -0.1830136  1.4275453 -0.58850143 -0.83437051 -0.07820784
##  [8,]  1.0059648  0.2667924  1.6297125  0.20499844 -0.88979412  1.53304888
##  [9,] -0.3181668 -0.1012472  1.1838421 -1.64742563  1.41913837  0.85817936
## [10,]  0.6076056 -0.4651706 -0.6314819 -1.66180242 -0.49096794  0.77924056
##             [,13]      [,14]      [,15]       [,16]      [,17]       [,18]
##  [1,] -0.6576937  0.2131094 -0.7141735  1.59165307 -1.0133512  0.67853924
##  [2,] -0.1716660 -2.4267080  0.2104557 -0.09626215  0.3867863  0.55323065
##  [3,]  1.1310901  0.0387633 -1.1969507 -1.40085233 -0.4129297 -0.03534562
##  [4,]  0.2869895  0.1959075  0.7768877 -0.62940907  0.2802799  1.27306406
##  [5,]  0.7607929  0.2811046 -0.2179369  1.15096321 -1.1481778 -0.65559178
##  [6,] -0.9940227  0.2462468 -0.2152095 -0.36686163  0.7440792  1.58640488
##  [7,] -0.2101040 -1.3540363  1.5794987  0.76586691 -0.2628760  0.77261894
##  [8,] -0.2520386  0.1204695 -0.5473661 -0.66488539  0.2295431  1.73716764
##  [9,]  0.5633500  1.0642745  0.1435902 -0.79247958  0.5515127  0.94045756
## [10,] -0.1642242 -0.1101115 -0.7134744 -0.35129167 -1.5797022 -0.74977734
##             [,19]       [,20]        [,21]       [,22]       [,23]       [,24]
##  [1,] -0.3435444 -0.28403832 -0.209421612 -0.94901815  0.24085438 -3.56546758
##  [2,]  1.3831321  0.12430314 -0.405135385 -0.40304875  3.09049181 -0.02031757
##  [3,] -0.4974586 -0.45267703  0.652979547  0.36285730  0.38107028  1.07583044
##  [4,] -0.7280114  0.97278733 -0.007276479 -0.81376270 -0.36246542 -0.17797460
##  [5,] -0.3588814 -0.27001734  0.470659467 -0.12456586 -0.44806604  0.90726917
##  [6,] -0.1584106  0.70243957 -0.818406507  0.06731867  0.30973089  1.92380314
##  [7,]  2.0494921  0.01607848  0.492807709  0.73900197  1.28335208  0.73678662
##  [8,]  1.8085854 -1.49908986 -0.354567368 -1.43977979 -1.13388702  0.19704994
##  [9,]  3.0783796  1.03002361  0.878442336  0.93644438 -0.02241596  0.50479815
```

```
## [10,] -1.0435984 -2.00430512 -0.093251350  0.52296127  2.28701556  1.83499969
##            [,25]      [,26]      [,27]      [,28]       [,29]      [,30]
##  [1,]  1.0803098 -0.2709516 -0.4505596 -1.2458351  1.01478701  0.5955444
##  [2,] -1.0984444 -0.2353374  0.3783422  0.5266843  0.28205120  0.3545815
##  [3,] -1.2397207 -0.6656729  2.0387299  0.6865393  0.77623523  0.2574353
##  [4,] -0.3999895  0.7165500 -0.5778062 -1.9429861  1.54655651 -1.7121784
##  [5,] -1.1183589  0.7367005  1.2405424  1.3773467 -0.70623391 -0.6549219
##  [6,]  2.3141582  1.6379133 -0.7240106  1.5880407 -0.77391362  0.3178245
##  [7,]  1.6702286 -1.4486289 -1.0775163 -0.5053823 -1.20257306  0.5493226
##  [8,] -0.7992809 -1.8238338  1.3768943 -0.2361238  0.04980158  0.1039445
##  [9,]  0.6704676 -1.1184750 -0.1683777  0.1478018 -1.13831151 -0.7704428
## [10,] -1.3092267  1.7907381 -0.4381742  0.3522512  0.36288765 -1.2849464
##             [,31]      [,32]      [,33]       [,34]      [,35]      [,36]
##  [1,] -1.58104019 -0.2826050 -0.6186233 -0.37047479 -0.7353166 -0.9359574
##  [2,] -0.48460965  0.5456288 -0.3915776 -1.42540916  0.8649132  1.8994852
##  [3,] -0.15161393 -0.6898492 -3.4272023 -2.18931828  0.6346782 -1.1349297
##  [4,]  0.34316172  2.0597532 -0.5524279  0.83953048  0.0741285 -0.7479593
##  [5,]  0.03170306 -0.1117867 -0.2446954 -0.27402246 -0.5598597  0.8952191
##  [6,] -0.74301002  0.4869046  1.4139290 -0.07633408  1.3358815  1.3994651
##  [7,] -0.34632262 -0.1580252 -0.7518682  1.28209010 -1.0744291  0.6427468
##  [8,] -0.05858485  1.9015647  0.1098279  0.04225518  0.3392788  2.1147013
##  [9,] -1.00698610  0.5499615  0.3521661  1.43168728  0.7171107 -1.3219533
## [10,]  0.76728856  0.1326966  0.8498958 -0.50163165  0.5705802  1.8563398
##             [,37]       [,38]      [,39]      [,40]       [,41]      [,42]
##  [1,] -0.1557535  0.62972233  1.9677102  0.2813951  0.31445116  0.9710274
##  [2,]  0.4328749 -0.35642306  1.4705457  0.4925677 -0.51883097  0.1023186
##  [3,] -1.4448496  0.21877863 -0.7711513  0.8605169  0.33232931 -0.1286925
##  [4,]  1.1904131 -1.73369693 -0.7819742  0.5962612  0.03343445 -1.1720947
##  [5,] -0.2192413  0.45713785 -0.2532747  0.7680439  0.47813691 -0.7688787
##  [6,]  2.8623257 -1.38589937 -0.6066036  1.2296826 -1.77501561 -1.6588894
##  [7,] -2.0216607  1.49663196  0.1756807  0.4023216 -0.30585306 -0.3458247
##  [8,] -0.7395301 -0.40672360  0.6304631 -1.8764050  1.38828186 -1.2588960
##  [9,]  2.8176584  0.18988064 -0.6416095 -1.4420422  0.41317676  0.2680968
## [10,] -0.5113731  0.07942885 -1.3152025 -0.8990692 -1.93739899 -0.5650264
##             [,43]      [,44]      [,45]      [,46]      [,47]       [,48]
##  [1,]  0.09503009 -0.5188458 -0.3861624 -2.3966839  0.4647319  1.71573936
##  [2,]  0.98242287  0.7206267  0.4664917 -0.9504760  1.7258127 -0.64904163
##  [3,] -0.50974365 -1.4089770 -1.1349020 -0.5157700  0.5840906 -0.91099169
##  [4,] -0.05824706 -0.2976816  1.0455860  0.1406822 -0.5256240  2.35644751
##  [5,]  0.94897485 -1.9873362 -0.6518191 -0.9194095  0.3209142 -0.26041320
##  [6,] -0.82433815 -1.2421166  0.8489192  0.2430664 -0.1671760 -0.01050722
##  [7,]  0.91485760  0.3610199  0.6798236  0.6518837 -0.6688038 -0.15201348
##  [8,] -1.27380674  0.2890418 -1.4367236 -0.8223132  1.1637779 -1.30675750
##  [9,] -0.91787826 -0.4171209 -1.3782627 -0.7182594 -0.7916382  1.27993621
## [10,] -1.04420241 -0.6713538  0.1662823 -0.7727626 -1.0793926  0.89973569
##             [,49]      [,50]       [,51]        [,52]       [,53]        [,54]
##  [1,]  1.0734610 -0.2490429  0.81253365  0.143547548 -1.22411626 -0.012267694
##  [2,] -0.2156093 -1.6882106  1.30419692  0.195061382  0.96356211 -0.645616718
##  [3,]  0.6563387  0.7907132  0.52648489 -0.070314580 -0.05585682 -0.030212300
##  [4,] -0.6446664 -1.2521455 -0.71068116 -0.752397181 -1.54212114  0.820772475
##  [5,]  0.6082391 -0.9937086  0.01190573 -0.677826213 -0.03459919  0.558745992
##  [6,]  1.0068146 -0.6820108 -0.07307675 -0.004782536  0.93964410 -0.005520795
##  [7,] -0.9818140 -2.5575636 -1.34547441  0.856966603 -0.36996422  0.767600392
##  [8,]  1.4463058 -0.4589163 -0.30546920 -0.193071203 -1.04014647  0.969216589
```

```
##  [9,]  1.3213091  0.3901681 -0.36714561  0.889009141  1.57189062  0.369306622
## [10,] -1.0517286  1.3863058  1.80944134  1.717764511 -0.29295648 -0.084514595
##             [,55]        [,56]       [,57]      [,58]      [,59]        [,60]
##  [1,] -0.30047698 -0.131693318 -0.14783523  1.7591706 -1.1513845  0.864376683
##  [2,]  1.54788707 -0.025387707 -0.23168345  1.4626088  0.3011123 -1.116141135
##  [3,] -1.21606320  0.002639726  2.29543455  0.2018709  0.7837639 -0.008246575
##  [4,] -0.06819395  0.245789338  0.28317954 -0.9526384  1.4662012  1.121977085
##  [5,] -1.17991116 -0.303483080 -0.89963609  0.1732294 -0.2415200  1.132401823
##  [6,]  0.09113827 -0.266780629 -0.28431039  0.6551305  0.4404200 -0.067820703
##  [7,] -0.58923470 -0.149566399  0.04124026  0.9989553 -0.5753614 -0.470710158
##  [8,] -1.19022189 -0.145374861  0.15903017  1.2334175 -1.1216240  1.262584333
##  [9,]  0.97516703 -3.065398456 -0.72143495  0.9504912  0.7024958 -0.037587589
## [10,] -0.06857073 -0.194568342 -1.19462164 -0.3251091 -1.4574718 -0.844262826
##           [,61]       [,62]      [,63]       [,64]      [,65]      [,66]
##  [1,]  0.7447219 -1.17776935  1.1791562 -0.15228648 -0.6101461 -1.5628570
##  [2,]  0.4918702  1.27392674  0.6541587  0.14191647 -1.0705352 -0.6364338
##  [3,] -0.9508562 -0.52692960  1.2542916  1.02790127 -1.8052043 -0.7996837
##  [4,] -0.5895866  0.26670912 -0.1658820  0.61320492  0.8077760 -0.4570090
##  [5,] -0.8202345 -0.26521315  1.8556688  0.06244798  1.2268736 -1.1281415
##  [6,]  1.0793464 -0.54848525  2.4441915 -1.01204971  1.2137380  1.4496536
##  [7,] -1.8620452 -1.06887042  2.5266335 -0.30530743  2.0150536  0.5655202
##  [8,] -0.9049131 -0.02649498 -1.6667106 -0.18477613  0.5593329  0.7148810
##  [9,]  1.0787841  0.86346794 -2.1986304 -1.71935862  0.3376478  1.3035556
## [10,]  0.2603570  1.88206539  1.1172500  0.64655994 -1.5389755  0.9007369
##             [,67]       [,68]       [,69]       [,70]      [,71]      [,72]
##  [1,]  1.92405774  1.52841107 -0.04349297  0.60974430 -0.7935816 -0.3776018
##  [2,]  0.03923253  1.52342466  0.14262667  2.11620751  0.6334356  0.3703459
##  [3,] -0.58402065 -0.34807192  0.98733658  0.53787191  0.6216641 -0.1383519
##  [4,]  0.78101327  1.29062471 -0.41018717 -0.16485378  0.7721799 -0.8208353
##  [5,]  0.32294967  0.69689125 -0.03942016 -1.16954061  0.6666712  0.5535722
##  [6,] -1.20226480  0.09044004 -0.52978884 -1.00480691  0.4660562 -1.1728949
##  [7,]  0.18356597  0.12836601  0.30320307  0.04913177  0.8654932 -0.1042334
##  [8,]  0.76698265  1.47699124  0.44717024  0.95173166 -0.6009023 -1.8881465
##  [9,] -0.03278841  0.78884074  0.58514947 -0.04291863 -2.0358668 -0.1078838
## [10,]  0.10390741  0.02408946  0.04083453 -0.28793519 -0.1461606  0.3484819
##           [,73]      [,74]        [,75]      [,76]      [,77]      [,78]
##  [1,]  1.5587189  0.2055185 -0.578876690  1.2581097  0.3374880  0.4864911
##  [2,]  0.4442001 -0.5711558 -0.422142373  0.1504374  1.6892256  0.0189059
##  [3,] -1.8618736 -0.4808484 -0.446893551  0.7644845 -0.1041426 -0.3354641
##  [4,]  1.5124627 -0.1687853 -0.266563744  0.1808127 -1.3531825 -0.9307962
##  [5,]  0.7333389  0.9702380 -0.982070925 -0.1863850 -0.4371151 -0.3755854
##  [6,]  0.1400260  0.3421322  0.870699626 -1.6235460  0.5054491 -0.1387680
##  [7,] -1.7684440  1.3105608 -1.427560009 -0.6154741 -0.6182477 -0.5613128
##  [8,]  0.8593289 -0.4740121  0.853264535  0.0467195  1.1737257 -0.3280387
##  [9,] -0.5049132  1.1517703 -0.005190369  0.8428874  1.3190428  0.1178909
## [10,]  0.5285347  0.9277631 -0.809457609 -0.1646248  0.6141446 -2.1404548
##             [,79]      [,80]
##  [1,] -0.69379338 -0.4697479
##  [2,]  0.53257829  1.2071504
##  [3,]  0.32494637 -1.3916363
##  [4,] -0.46971579  0.3833679
##  [5,] -1.81898652  1.6213089
##  [6,]  0.07754022 -3.1302245
##  [7,] -0.12657967  0.6059834
```

```
##  [8,] -1.53504161  0.4199102
##  [9,]  0.40286342 -0.8452174
## [10,] -1.21387461 -1.2910836
```

```
mb[[3]]$Variables                          # To check variable names in b3
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## [26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
## [51] 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
```

```
mb[['b3']]$Samples                         # To Check sample names in b3
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

As shown above, since we did not provide nor the *variable names* nor the *sample names* in this example, this information was filled with their indexes. However, if available, it can be added by using the parameters **newVars** and **newSamples**.

Let's start again.

```
# Samples are named s1-s10, and variables are named v1-v50.
mb <- BuildMultiBlock(b1, newSamples = paste0('s',rep(1:5,2)),newVars = paste0('v',1:50))
# I don't need to fill these parameters for all the blocks.
mb <- BuildMultiBlock(b2, growingMB = mb)
# Here I have only provided the sample names.
mb <- BuildMultiBlock(b3, growingMB = mb, newSamples = mb$b1$Samples)
# To show the content of mb
str(mb)
```

```
## List of 3
##  $ b1:List of 3
##   ..$ Data     : num [1:10, 1:50] -1.934 1.293 -0.02 0.262 -1.174 ...
##   ..$ Variables: chr [1:50] "v1" "v2" "v3" "v4" ...
##   ..$ Samples  : chr [1:10] "s1" "s2" "s3" "s4" ...
##  $ b2:List of 3
##   ..$ Data     : num [1:10, 1:80] -0.49749 1.84495 0.25926 0.00537 0.13239 ...
##   ..$ Variables: num [1:80] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Samples  : num [1:10] 1 2 3 4 5 6 7 8 9 10
##  $ b3:List of 3
##   ..$ Data     : num [1:10, 1:70] -0.3263 -0.6867 0.3502 0.6277 0.0826 ...
##   ..$ Variables: num [1:70] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Samples  : chr [1:10] "s1" "s2" "s3" "s4" ...
```

The **BuildMultiBlock.R** function is very flexible since it accepts any other information data (ex. metadata) as long as it is of the same size as the number of samples.

```
# The first 5 samples are from batch 1, and the remaining 5 from batch 2. Vector of length 10.
batch = c(1,1,1,1,1,2,2,2,2,2)
# 6 samples from batch 1, 5 from batch 2. Vector of length 11.
batch11 = c(1,1,1,1,1,1,2,2,2,2,2)
# Another vector of length 10.
month = c('Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct')
```

```
# This line of code will not work because batch11 is of length 11 and there are only 10 samples.
mb <- BuildMultiBlock(b1, batches = batch11)
```

```
## [1] "Please use a batches object with 10 items."
```

```
## Error in BuildMultiBlock(b1, batches = batch11): The length of batches is incorrect.
```

```
# This line of code will works
mb <- BuildMultiBlock(b1, batches = batch)
# From b2 I know both the batch information and the month information.
mb <- BuildMultiBlock(b2, growingMB = mb, batches = batch, month = month)
# From b3 I only know the month information
mb <- BuildMultiBlock(b3, growingMB = mb, newSamples = mb$b1$Samples)

# To show the content of mb
str(mb)
```

```
## List of 3
##  $ b1:List of 4
##   ..$ Data     : num [1:10, 1:50] -1.934 1.293 -0.02 0.262 -1.174 ...
##   ..$ Variables: num [1:50] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Samples  : num [1:10] 1 2 3 4 5 6 7 8 9 10
##   ..$ Batch    : num [1:10] 1 1 1 1 1 2 2 2 2 2
##  $ b2:List of 5
##   ..$ Data     : num [1:10, 1:80] -0.49749 1.84495 0.25926 0.00537 0.13239 ...
##   ..$ Variables: num [1:80] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Samples  : num [1:10] 1 2 3 4 5 6 7 8 9 10
##   ..$ Batch    : num [1:10] 1 1 1 1 1 2 2 2 2 2
##   ..$ month    : chr [1:10] "Jan" "Feb" "Mar" "Apr" ...
##  $ b3:List of 3
##   ..$ Data     : num [1:10, 1:70] -0.3263 -0.6867 0.3502 0.6277 0.0826 ...
##   ..$ Variables: num [1:70] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Samples  : num [1:10] 1 2 3 4 5 6 7 8 9 10
```

So, the multi-blocks can be always built except when there is no consistency in the data dimensions. This also includes the case when the different blocks to add have different sample numbers.

Since **ComDim** uses matrix concatenation in its algorithm, it is imperative that all the blocks have the same sample number.

```
# Block 4: Matrix of 11 rows and 100 columns, filled with random values.
b4 = matrix(rnorm(1100),11,100)

# This will not work because b4 is of length 11 and blocks in mb are of length 10.
mb <- BuildMultiBlock(b4, growingMB = mb)
```

```
## Error in BuildMultiBlock(b4, growingMB = mb): The number of samples in newBlock is incorrect.
```

Having said this, it is still possible to build an inconsistent multi-block if the **equalSampleNumber** is set to **FALSE**. This may be convenient when dealing with **incomplete datasets**.

```r
# Block 4: Matrix of 11 rows and 100 columns, filled with random values.
b4 = matrix(rnorm(1100),11,100)

# With equalSampleNumber = FALSE, we can bypass the error.
# However, the resulting block will not be YET compatible with ComDim.
mb <- BuildMultiBlock(b4, growingMB = mb, equalSampleNumber = FALSE)
```

3.1 Error messages in BuildMultiBlock

To avoid overwriting the blocks, each one should have a different names. If the name is re-used in a new block, the function will show an *error message*.

```r
# This does not work because b4 was already added to mb
# (in the previous section)
mb <- BuildMultiBlock(b4, growingMB = mb)
```

```
## Error in BuildMultiBlock(b4, growingMB = mb): There already exists a block named b4 in the multi-bloc
```

```r
# We need to remove b4 first.
mb<-mb[c(1:3)]
# After that, b4 can be added.
mb <- BuildMultiBlock(b4, growingMB = mb)
```

```
## Error in BuildMultiBlock(b4, growingMB = mb): The number of samples in newBlock is incorrect.
```

Additional message errors were added for **incompatible situations**:

- *Inconsistence in the dimensions* of the input data:

1) Between the vector of sample names and the data matrix.

```r
# Block 5: Matrix of 10 rows and 15 columns, filled with random values.
b5 = matrix(rnorm(150),10,15)
# Generating a sample names vector of the wrong length.
bad_samples = seq(1,11,1)

# This will not work because the length of bad_samples is not 10
mb <- BuildMultiBlock(b5, growingMB = mb, newSamples = bad_samples)
```

```
## [1] "Please use a newSamples object with 10 items."
```

```
## Error in BuildMultiBlock(b5, growingMB = mb, newSamples = bad_samples): The length of newSamples is
```

2) Between the vector of variable names and the data matrix.

```r
# Block 5: Matrix of 10 rows and 15 columns, filled with random values.
b5 = matrix(rnorm(150),10,15)
# Variable names vector of the wrong length.
bad_variables = seq(12,50,1)

# This will not work because the length of bad_variables is not 15
mb <- BuildMultiBlock(b5, growingMB = mb, newVars = bad_variables)
```

```
## [1] "Please use a newVars object with 15 items."
```

```
## Error in BuildMultiBlock(b5, growingMB = mb, newVars = bad_variables): The length of newVars is inco
```

- *Wrong input format*:

1) The data-block to incorporate to growingMB must be a matrix.

```
# This will not work because the block to be added must be a matrix, and mb is a list.
mb <- BuildMultiBlock(mb, growingMB = mb)
```

```
## Error in BuildMultiBlock(mb, growingMB = mb): newBlock is not a matrix.
```

2) And the growingMB must be a list.

```
# This does not work because the block to be added must be a matrix object.
# The code fails to convert 'mb$b1$Data' into a suitable block name.
mb <- BuildMultiBlock(mb$b1$Data, growingMB = mb)
```

```
## [1] "The provided newBlock is contained in a list, while it must be a stand-alone matrix object."
```

```
## Error in BuildMultiBlock(mb$b1$Data, growingMB = mb): Newblock must be a matrix object.
```

4 Building the Replicate Blocks (RBs)

When the data in the blocks contain samples from different batches, it is possible to split each of these blocks into smaller blocks. We call these smaller blocks as **Replicate Blocks** or **RBs**.

To generate the RBs, we can simply use the **splitRB.R** function with a multi-block object that contains the *batches* information, which must be a *numeric vector*.

```
source('splitRB.R')  # Load the function.
batch = c(1,1,1,1,1,2,2,2,2,2)             # The batches information.
mb <- BuildMultiBlock(b1, batches = batch) # We generate first the MB
rb <- splitRB(mb)                          # We generate the RB
```

```
## [1] "The sample names are:"
##      b1_R1 b1_R2
## [1,]     1     6
## [2,]     2     7
## [3,]     3     8
## [4,]     4     9
## [5,]     5    10
```

```
str(rb)                                    # To check the content of RB
```

```
## List of 2
##  $ b1_R1:List of 4
##   ..$ Data     : num [1:5, 1:50] -1.934 1.293 -0.02 0.262 -1.174 ...
##   ..$ Variables: num [1:50] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Samples  : num [1:5] 1 2 3 4 5
##   ..$ Batch    : num [1:5] 1 1 1 1 1
##  $ b1_R2:List of 4
##   ..$ Data     : num [1:5, 1:50] -0.788 -0.605 -0.805 1.414 -1.375 ...
##   ..$ Variables: num [1:50] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Samples  : num [1:5] 6 7 8 9 10
##   ..$ Batch    : num [1:5] 2 2 2 2 2
```

In the output object, **rb**, the 5 first samples (those from batch 1 according to the **batch** vector) were split into the first RB and the 5 remaining samples (those from batch 2) were split into the second RB. Moreover, the two RBs generated from b1 are named **b1_R1** (for 'replicate 1') and **b1_R2** (for 'replicate 2'). This is summarized in the data frame generated by the same function.

In the previous example, **mb** was composed of 1 block only. Of course, splitRB can be used with a real multi-block contaning more than 1 block.

```
mb <- BuildMultiBlock(b1, batches = batch)                  # We generate first the MB
mb <- BuildMultiBlock(b2, growingMB = mb, batches = batch)  # We add b2 to mb.
rb <- splitRB(mb)                                           # We generate the RB
```

```
## [1] "The sample names are:"
##      b1_R1 b1_R2 b2_R1 b2_R2
## [1,]     1     6     1     6
## [2,]     2     7     2     7
## [3,]     3     8     3     8
## [4,]     4     9     4     9
## [5,]     5    10     5    10
```

```
str(rb)                                                     # To check the content of RB
```

```
## List of 4
##  $ b1_R1:List of 4
##   ..$ Data     : num [1:5, 1:50] -1.934 1.293 -0.02 0.262 -1.174 ...
##   ..$ Variables: num [1:50] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Samples  : num [1:5] 1 2 3 4 5
##   ..$ Batch    : num [1:5] 1 1 1 1 1
##  $ b1_R2:List of 4
##   ..$ Data     : num [1:5, 1:50] -0.788 -0.605 -0.805 1.414 -1.375 ...
##   ..$ Variables: num [1:50] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Samples  : num [1:5] 6 7 8 9 10
##   ..$ Batch    : num [1:5] 2 2 2 2 2
##  $ b2_R1:List of 4
##   ..$ Data     : num [1:5, 1:80] -0.49749 1.84495 0.25926 0.00537 0.13239 ...
##   ..$ Variables: num [1:80] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Samples  : num [1:5] 1 2 3 4 5
##   ..$ Batch    : num [1:5] 1 1 1 1 1
##  $ b2_R2:List of 4
##   ..$ Data     : num [1:5, 1:80] -0.345 1.26 -1.886 1.511 -0.358 ...
##   ..$ Variables: num [1:80] 1 2 3 4 5 6 7 8 9 10 ...
```

```
##   ..$ Samples  : num [1:5] 6 7 8 9 10
##   ..$ Batch    : num [1:5] 2 2 2 2 2
```

If some of the blocks contain metadata, these data will also be kept in the RBs.

```
temperature = paste0(seq(25,34,1),'°C')                          # Metadata
mb <- BuildMultiBlock(b1, batches = batch, temp = temperature) # We generate first the MB
mb <- BuildMultiBlock(b2, growingMB = mb, batches = batch)     # We add b2 to mb.
rb <- splitRB(mb)                                                # We generate the RB
```

```
## [1] "The sample names are:"
##      b1_R1 b1_R2 b2_R1 b2_R2
## [1,]     1     6     1     6
## [2,]     2     7     2     7
## [3,]     3     8     3     8
## [4,]     4     9     4     9
## [5,]     5    10     5    10
```

```
str(rb)                                                          # To check the content of RB
```

```
## List of 4
##  $ b1_R1:List of 5
##   ..$ Data     : num [1:5, 1:50] -1.934 1.293 -0.02 0.262 -1.174 ...
##   ..$ Variables: num [1:50] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Samples  : num [1:5] 1 2 3 4 5
##   ..$ Batch    : num [1:5] 1 1 1 1 1
##   ..$ temp     : chr [1:5] "25°C" "26°C" "27°C" "28°C" ...
##  $ b1_R2:List of 5
##   ..$ Data     : num [1:5, 1:50] -0.788 -0.605 -0.805 1.414 -1.375 ...
##   ..$ Variables: num [1:50] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Samples  : num [1:5] 6 7 8 9 10
##   ..$ Batch    : num [1:5] 2 2 2 2 2
##   ..$ temp     : chr [1:5] "30°C" "31°C" "32°C" "33°C" ...
##  $ b2_R1:List of 4
##   ..$ Data     : num [1:5, 1:80] -0.49749 1.84495 0.25926 0.00537 0.13239 ...
##   ..$ Variables: num [1:80] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Samples  : num [1:5] 1 2 3 4 5
##   ..$ Batch    : num [1:5] 1 1 1 1 1
##  $ b2_R2:List of 4
##   ..$ Data     : num [1:5, 1:80] -0.345 1.26 -1.886 1.511 -0.358 ...
##   ..$ Variables: num [1:80] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ Samples  : num [1:5] 6 7 8 9 10
##   ..$ Batch    : num [1:5] 2 2 2 2 2
```

4.1 Error messages in splitRB

If the multi-block is NOT consistent with the output from **BuildMultiBlock.R**, the function will report an *error*:

1) If the input *is not a list*.

```
# It fails because the function input must be a multi-block structure, not a matrix.
rb <- splitRB(b1)
```

```
## [1] "The provided multi-block is not a list."
```

```
## Error in splitRB(b1): The multi-block must be a list.
```

2) If the input is not a nested list (*list of lists*).

```
b1_list <- list(b1)
# It fails because the list b1_list is not a nested list.
rb <- splitRB(b1_list)
```

```
## [1] "The block 1 is not a list."
```

```
## Error in splitRB(b1_list): The block 1 must be a list.
```

3) Inconsistence in the length of the **batches** vector.

```
# It fails because the length of the batch vector is different
# from the sample number in the corresponding data.
mb2 <- BuildMultiBlock(b1, batches = batch[1:9]) # Creating a mock multiblock
```

```
## [1] "Please use a batches object with 10 items."
```

```
## Error in BuildMultiBlock(b1, batches = batch[1:9]): The length of batches is incorrect.
```

4) Inconsistence in the length of the **Samples** data.

```
mb2 <- BuildMultiBlock(b1, batches = batch[1:9]) # Creating a mock multiblock
```

```
## [1] "Please use a batches object with 10 items."
```

```
## Error in BuildMultiBlock(b1, batches = batch[1:9]): The length of batches is incorrect.
```

```
mb2$b1$Samples <- paste0('s',1:9)      # Altering the Sample names vector.
```

```
## Error in mb2$b1$Samples <- paste0("s", 1:9): objet 'mb2' introuvable
```

```
# It fails because the length of the Sample names vector is different
# from the sample number in the corresponding data.
rb <- splitRB(mb2)
```

```
## Error in splitRB(mb2): objet 'mb2' introuvable
```

5) Inconsistence in the length of the **Variables** data.

```
mb2 <- BuildMultiBlock(b1, batches = batch) # Creating a mock multiblock
mb2$b1$Variables <- paste0('v',1:9)    # Altering the Variable names vector.

# It fails because the length of the Variable names vector is different
# from the Variable number in the corresponding data.
rb <- splitRB(mb2)
```

```
## Error in splitRB(mb2): The length of the Variables vector in block 1 is incorrect.
```

6) In consistence in the *number of samples across blocks.*

```
mb2 <- BuildMultiBlock(b1, batches = batch) # Creating a mock multiblock
mb2$b1$Data <-  mb2$b1$Data[1:9,]           # Altering the number of samples in data.
mb2$b1$Batch <-  mb2$b1$Batch[1:9]          # Altering the number of samples in data.
mb2$b1$Samples <- paste0('s',1:9)     # Altering the Sample names vector.

# It fails because the sample number in batch 1 is not the same as in batch 2.
rb <- splitRB(mb2)
```

```
## Warning in splitRB(mb2): The replicate blocks in b1 have a different number of
## samples.

## [1] "Information is missing regarding the splitting."
## [1] "Using checkSampleCorrespondence as TRUE is recommended."

## Error in splitRB(mb2): The data cannot be split into replicate blocks.
```

4.2 Sample correspondence in the splitting of the blocks

So far, in all these tests, we have assumed that samples are sorted in the same order within each batch, implying a *perfectly aligned* sample correspondence across RBs (ex. sample 1 from b1_R1 corresponds to sample 6 from b1_R2, sample 2 to sample 7, and so on).

However, if the sample correspondence may not be *TRUE*, then we can build the replicate blocks by checking their sample names first. This can be made by setting the *checkSampleCorrespondence* to *TRUE*. In addition, with this additional parameter, only the samples with common names across all the blocks will be kept.

```
# There is sample correspondency for samples s1-s4, but not for s5 nor s6.
sample_names = paste0('s',c(1,2,3,4,5,1,2,3,4,6))
print(sample_names)
```

```
##  [1] "s1" "s2" "s3" "s4" "s5" "s1" "s2" "s3" "s4" "s6"
```

```
mb <- BuildMultiBlock(b1, batches = batch, newSamples = sample_names) # We generate the MB
rb <- splitRB(mb, checkSampleCorrespondence = TRUE)                   # We generate the RB
```

```
## [1] "The sample names are:"
##      b1_R1 b1_R2
## [1,] "s1"  "s1"
## [2,] "s2"  "s2"
## [3,] "s3"  "s3"
## [4,] "s4"  "s4"
```

In case the data we are analyzing do not have replicates, this function can still be used to ensure that samples are correctly sorted across blocks.

```
# FOR UNSORTED BLOCKS:
batch1 = rep(1,10) # Only one batch per block.
sample_names1 = paste0('s',c(1:10)) # Sample names for b1
sample_names2 = paste0('s',c(10:1)) # Sample names for b2
# There is sample correspondence, but samples do not have the same order

mb <- BuildMultiBlock(b1, batches = batch1, newSamples = sample_names1) # We generate the MB
mb <- BuildMultiBlock(b2, growingMB = mb, batches = batch1, newSamples = sample_names2) # We add b2
rb <- splitRB(mb, checkSampleCorrespondence = TRUE)                    # Check for the common samples
```

```
## [1] "The sample names are:"
##       b1_R1 b2_R1
##  [1,] "s1"  "s1"
##  [2,] "s10" "s10"
##  [3,] "s2"  "s2"
##  [4,] "s3"  "s3"
##  [5,] "s4"  "s4"
##  [6,] "s5"  "s5"
##  [7,] "s6"  "s6"
##  [8,] "s7"  "s7"
##  [9,] "s8"  "s8"
## [10,] "s9"  "s9"
```

```
# FOR BLOCKS WITH PARTIAL SAMPLE CORRESPONDENCE
sample_names1 = paste0('s',c(1:10)) # Sample names for b1
sample_names2 = paste0('s',c(6:15)) # Sample names for b2
# There is sample correspondence for samples s6-s10.

mb <- BuildMultiBlock(b1, batches = batch1, newSamples = sample_names1) # We generate the MB
mb <- BuildMultiBlock(b2, growingMB = mb, batches = batch1, newSamples = sample_names2) # We add b2
rb <- splitRB(mb, checkSampleCorrespondence = TRUE)                    # Check for the common samples
```

```
## [1] "The sample names are:"
##      b1_R1 b2_R1
## [1,] "s10" "s10"
## [2,] "s6"  "s6"
## [3,] "s7"  "s7"
## [4,] "s8"  "s8"
## [5,] "s9"  "s9"
```

Finally, it could occur that we have several replicates in one or more blocks, but not for all of them. This is the case for the *incomplete datasets*. In order to build a multi-block structure (and avoid getting an error due to the divergent sample size), the parameter *equalSampleNumber* is set to *FALSE* in *BuildMultiBlock.R*.

```
b6 = matrix(rnorm(2400),30,80)
  # Block 6: Matrix of 30 rows and 80 columns, filled with random values.
batch6 = c(rep(1,10),rep(2,10),rep(3,10)) # b6 contains samples from 3 batches.
sample_names6 = c(sample_names1, sample_names1, sample_names1)
```

```
mb <- BuildMultiBlock(b1, batches = batch1, newSamples = sample_names1) # We generate the MB
mb <- BuildMultiBlock(b6, growingMB = mb, batches = batch6,
                      newSamples = sample_names6, equalSampleNumber = FALSE) # We add b6
rb <- splitRB(mb)
```

```
## [1] "The sample names are:"
##       b1_R1 b6_R1 b6_R2 b6_R3
##  [1,] "s1"  "s1"  "s1"  "s1"
##  [2,] "s10" "s10" "s10" "s10"
##  [3,] "s2"  "s2"  "s2"  "s2"
##  [4,] "s3"  "s3"  "s3"  "s3"
##  [5,] "s4"  "s4"  "s4"  "s4"
##  [6,] "s5"  "s5"  "s5"  "s5"
##  [7,] "s6"  "s6"  "s6"  "s6"
##  [8,] "s7"  "s7"  "s7"  "s7"
##  [9,] "s8"  "s8"  "s8"  "s8"
## [10,] "s9"  "s9"  "s9"  "s9"
```

5 ComDim

ComDim is executed using the *comdim_PCA.R* function. This function accepts as inputs the multi-block data sets generated with *splitRB.R* and *BuildMultiBlock.R* (as long as their number of samples are conserved for all the blocks).

In this example we will use *Dataset3*. The data from this dataset was obtained from *Gomez-Cabrero et al. (2019)*. It consists of 4 data-blocks (LC-MS, GC-MS, RNAseq, and miRNAseq), and each block contains data from 36 samples (12 samples per batch). These 12 samples correspond to 2 treatments measured at 6 different time-points.

This data set was initially stored into a Matlab .mat file. To read it in *R*, we use the package *R.matlab*.

```
path = 'dataset3b.mat'

library('R.matlab')
```

```
## Warning: package 'R.matlab' was built under R version 3.6.3
```

```
## R.matlab v3.6.2 (2018-09-26) successfully loaded. See ?R.matlab for help.
```

```
##
## Attaching package: 'R.matlab'
```

```
## The following objects are masked from 'package:base':
##
##     getOption, isOpen
```

```
data3 <- readMat(path)
lcms <- data3$dataset1b[[1]]
gcms <- data3$dataset1b[[4]]
rnaseq <- data3$dataset1b[[7]]
mirnaseq <- data3$dataset1b[[10]]
```

```
batch_d3 <- c(rep(1,12),rep(2,12),rep(3,12))
time <- c(rep(c(0,0,2,2,4,4,6,6,10,10,24,24),3)) # Time in h.
exposition <- rep(c(0,1),18) # Odd samples are controls 'O', even samples are exposed '1'.
```

At this stage, we can build the multi-block:

```
mb_d3 <- BuildMultiBlock(lcms, batches = batch_d3, time = time, exposition = exposition)
mb_d3 <- BuildMultiBlock(gcms, growingMB = mb_d3, batches = batch_d3,
                         time = time, exposition = exposition)
mb_d3 <- BuildMultiBlock(rnaseq, growingMB = mb_d3, batches = batch_d3,
                         time = time, exposition = exposition)
mb_d3 <- BuildMultiBlock(mirnaseq, growingMB = mb_d3, batches = batch_d3,
                         time = time, exposition = exposition)
rb_d3 <- splitRB(mb_d3)
```

```
## [1] "The sample names are:"
##       lcms_R1 lcms_R2 lcms_R3 gcms_R1 gcms_R2 gcms_R3 rnaseq_R1 rnaseq_R2
##  [1,]       1      13      25       1      13      25         1        13
##  [2,]       2      14      26       2      14      26         2        14
##  [3,]       3      15      27       3      15      27         3        15
##  [4,]       4      16      28       4      16      28         4        16
##  [5,]       5      17      29       5      17      29         5        17
##  [6,]       6      18      30       6      18      30         6        18
##  [7,]       7      19      31       7      19      31         7        19
##  [8,]       8      20      32       8      20      32         8        20
##  [9,]       9      21      33       9      21      33         9        21
## [10,]      10      22      34      10      22      34        10        22
## [11,]      11      23      35      11      23      35        11        23
## [12,]      12      24      36      12      24      36        12        24
##       rnaseq_R3 mirnaseq_R1 mirnaseq_R2 mirnaseq_R3
##  [1,]        25           1          13          25
##  [2,]        26           2          14          26
##  [3,]        27           3          15          27
##  [4,]        28           4          16          28
##  [5,]        29           5          17          29
##  [6,]        30           6          18          30
##  [7,]        31           7          19          31
##  [8,]        32           8          20          32
##  [9,]        33           9          21          33
## [10,]        34          10          22          34
## [11,]        35          11          23          35
## [12,]        36          12          24          36
```

Finally, *ComDim* can be run:

```
results <- comdim_PCA(data = rb_d3, 4) # We will calculate 4 components.
```

```
##   |                                                                      |
## [1] "Checking sample correspondence across blocks."
## [1] "The data can be used for ComDim."
##   |                                                                      |==
##   |                                                                      |==================
```

Several sets of information are obtained from the *ComDim* analysis:

```
names(results)
```

```
##  [1] "MeanMB"            "NormMB"            "SingVal"
##  [4] "Q"                 "explained"         "Sum_saliences_Dim"
##  [7] "Sum_saliences_Tab" "saliences"         "b"
## [10] "T_Loc"             "P"                 "P_Loc"
## [13] "Lx"                "Lx_Loc"            "runtime"
```

We will focus now on the most important results: the *saliences*, the global scores (Q), the local scores (T_Loc), the loadings (P), and percentage of the explained variance (*explained*).

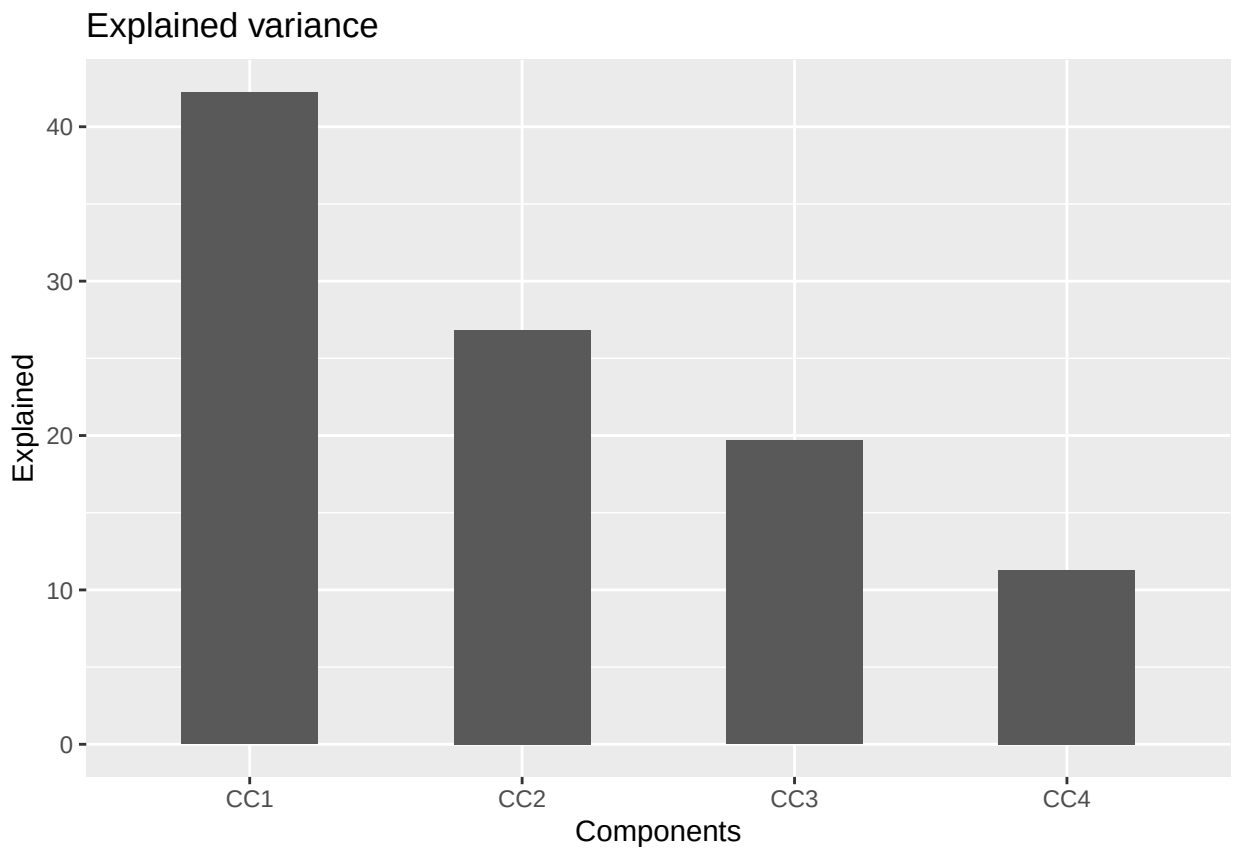
6 Explained variance

This result gives the relative variance of the data set explained for every component.

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
df <- data.frame(Components= names(results$explained$Data),
                 Explained = results$explained$Data)
ggplot(data=df, aes(x=Components, y=Explained)) +
  geom_bar(stat="identity", width=0.5)  +
    ggtitle("Explained variance")
```



The *1st* component (or *CC1*) explains *40% of the data set variance*, while the *4th* component (or *CC4*) only explains approximately the *10% of the data set variance*.

7 Saliences

By looking at the saliences, we can evaluate the importance of each block in the model.

```
library(ggplot2)
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 3.6.3
```

```
library(dplyr)
```

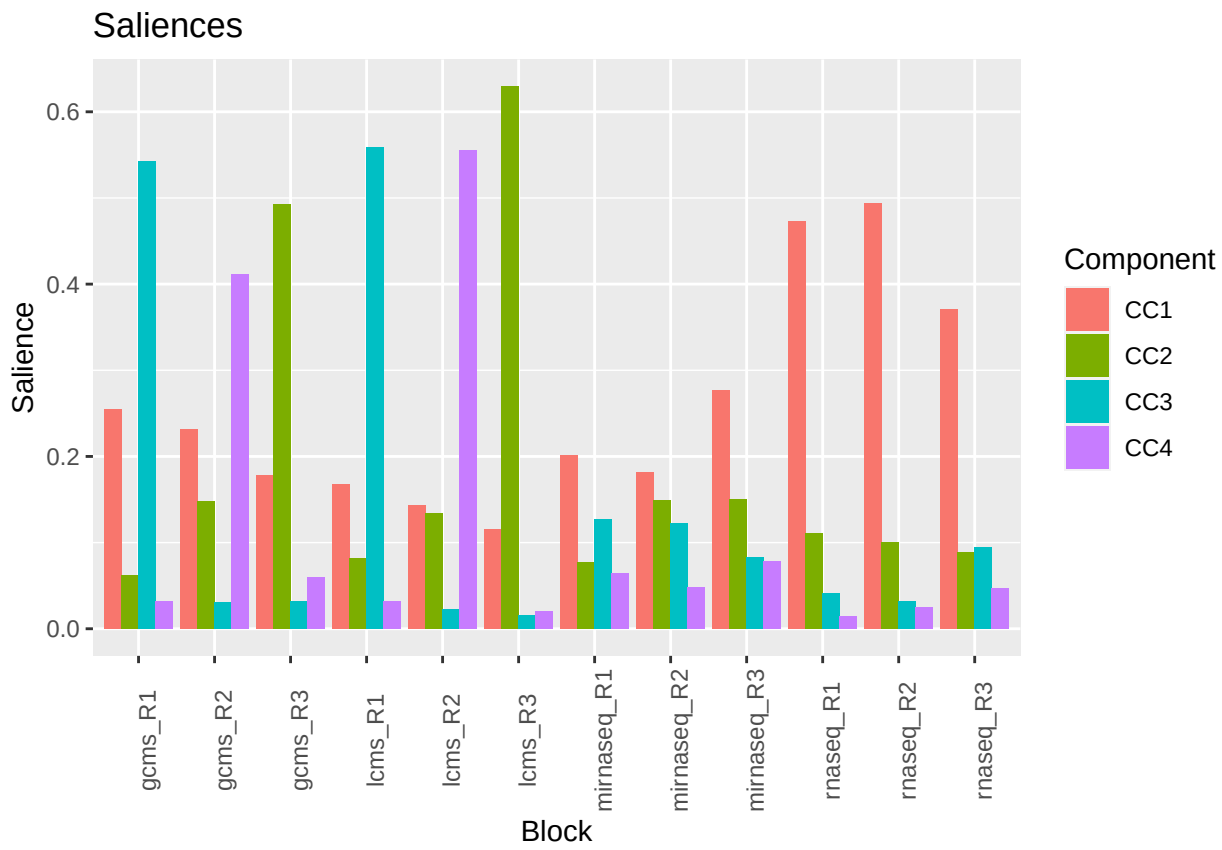
```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
df <- data.frame(rownames(results$saliences$Data),results$saliences$Data, stringsAsFactors = FALSE)
colnames(df) <- c('Block','CC1','CC2','CC3','CC4')

df2 <- df %>%
  pivot_longer(!Block, names_to = 'Component', values_to = 'Salience')

ggplot(data = df2, aes(x = Block, y = Salience, fill = Component)) +
    geom_bar(stat = "identity", position = position_dodge()) +
    theme(axis.text.x = element_text(angle = 90)) +
    ggtitle("Saliences")
```



We can see that the *1st* replicate blocks (*RB1*) of the GCMS and LCMS data are mainly explained by *CC3* in cyan, the *RB2* of the same data type for *CC4*, and the *RB3* for *CC2*. This denotes a batch effect *common* across the *LCMS* and *GCMS* data. On the other hand, *CC1* is mostly descriptive of the *RNAseq* and *miRNAseq* data.

8 Global scores

The *Global scores* show the trend of the samples *common across all the blocks*.

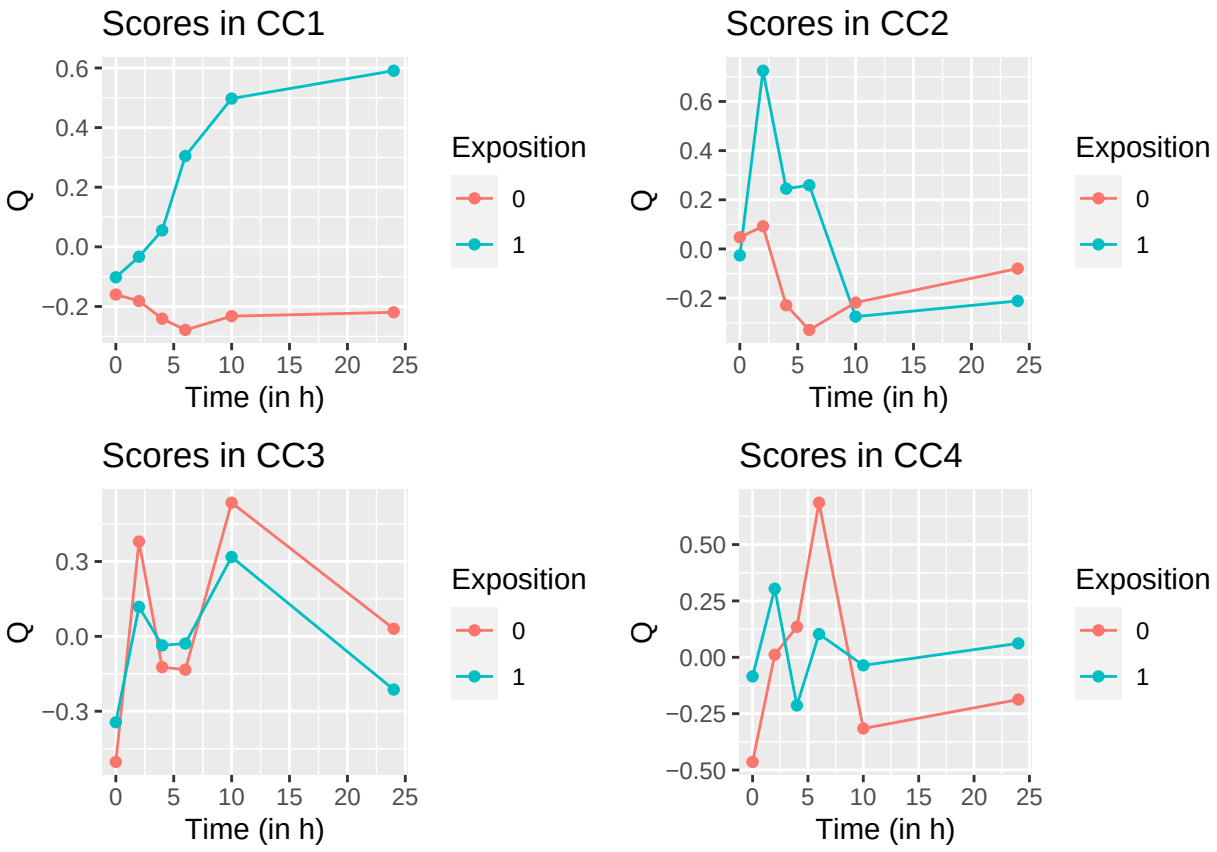
```r
library(ggplot2)
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 3.6.3
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
df_Q1 <- data.frame(Q = results$Q$Data[,1],
                Time = results$Q$time,
                Exposition = as.factor(results$Q$exposition))

df_Q2 <- data.frame(Q = results$Q$Data[,2],
                Time = results$Q$time,
                Exposition = as.factor(results$Q$exposition))

df_Q3 <- data.frame(Q = results$Q$Data[,3],
                Time = results$Q$time,
                Exposition = as.factor(results$Q$exposition))

df_Q4 <- data.frame(Q = results$Q$Data[,4],
                Time = results$Q$time,
                Exposition = as.factor(results$Q$exposition))

p1 <- ggplot(data=df_Q1, aes(x=Time, y=Q, group=Exposition)) +
  geom_line(aes(color=Exposition))+
  geom_point(aes(color=Exposition))+
  labs(x = "Time (in h)", title = "Scores in CC1")

p2 <- ggplot(data=df_Q2, aes(x=Time, y=Q, group=Exposition)) +
  geom_line(aes(color=Exposition))+
  geom_point(aes(color=Exposition))+
  labs(x = "Time (in h)", title = "Scores in CC2")

p3 <- ggplot(data=df_Q3, aes(x=Time, y=Q, group=Exposition)) +
  geom_line(aes(color=Exposition))+
  geom_point(aes(color=Exposition))+
  labs(x = "Time (in h)", title = "Scores in CC3")

p4 <- ggplot(data=df_Q4, aes(x=Time, y=Q, group=Exposition)) +
  geom_line(aes(color=Exposition))+
  geom_point(aes(color=Exposition))+
  labs(x = "Time (in h)", title = "Scores in CC4")

grid.arrange(p1, p2, p3, p4, nrow = 2)
```



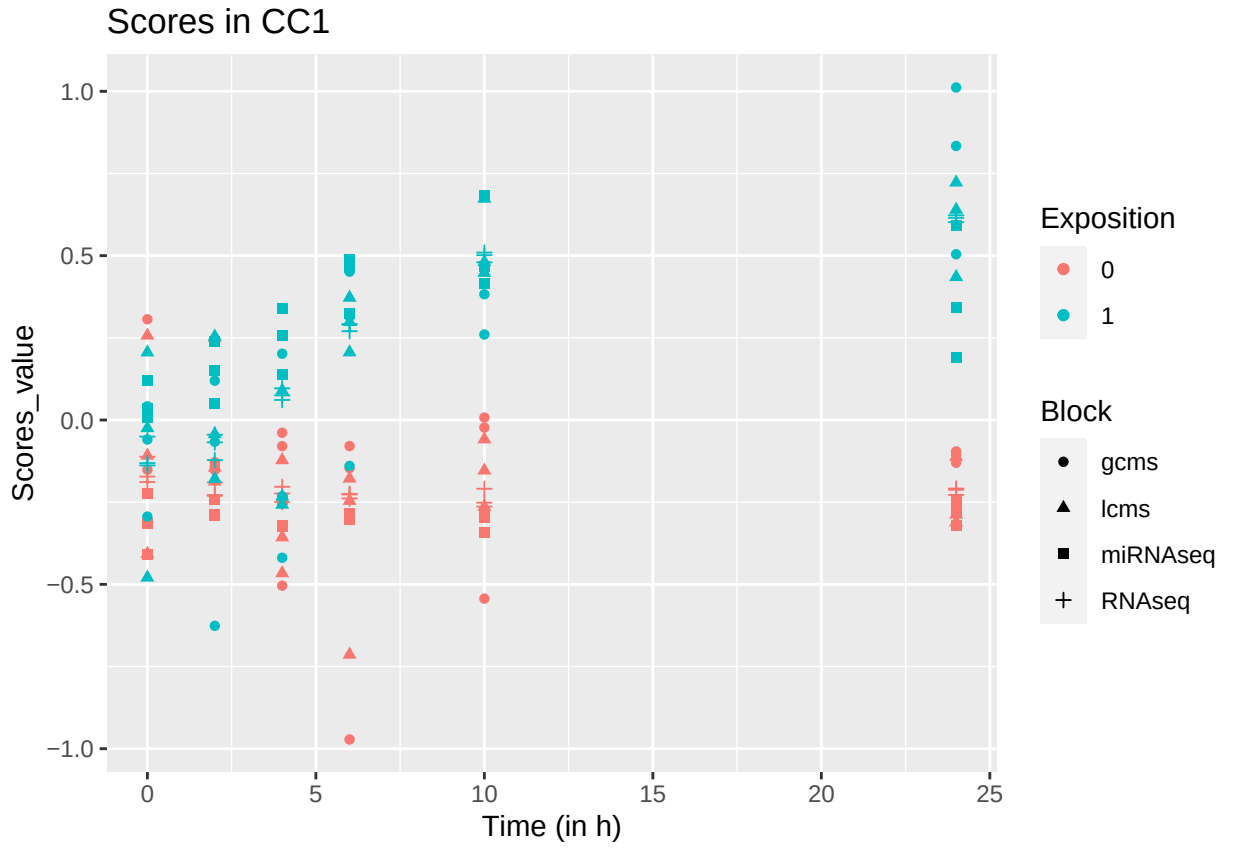
In *CC1*, an increase over time in is observed for the exposed samples.

9 Local scores

The *Local scores* show the trend of the samples *for each block*. For this example, we only work with *CC1*.

```
library(ggplot2)
library(tidyr)
library(dplyr)

df <- data.frame(results$T_Loc$Data[[1]][,1],
                 results$T_Loc$Data[[2]][,1],
                 results$T_Loc$Data[[3]][,1],
                 results$T_Loc$Data[[4]][,1],
                 results$T_Loc$Data[[5]][,1],
                 results$T_Loc$Data[[6]][,1],
                 results$T_Loc$Data[[7]][,1],
                 results$T_Loc$Data[[8]][,1],
                 results$T_Loc$Data[[9]][,1],
                 results$T_Loc$Data[[10]][,1],
                 results$T_Loc$Data[[11]][,1],
                 results$T_Loc$Data[[12]][,1],
                Time = results$Q$time,
                Exposition = as.factor(results$Q$exposition))
colnames(df) <- c(paste0('gcms',1:3), paste0('lcms',1:3),
```

```
                  paste0('RNAseq',1:3), paste0('miRNAseq',1:3),'Time','Exposition')

df2 <- df %>%
  pivot_longer(cols = -c('Time','Exposition'), #These columns are extracted as they are.
               names_to = 'Block', # The new column to be created.
               names_pattern = "([a-zA-Z]+)", # Remove characters
               values_to = 'Scores_value')

ggplot(data=df2, aes(x=Time, y=Scores_value, shape = Block, color = Exposition)) +
  geom_point()+
  labs(x = "Time (in h)") +
  ggtitle("Scores in CC1")
```



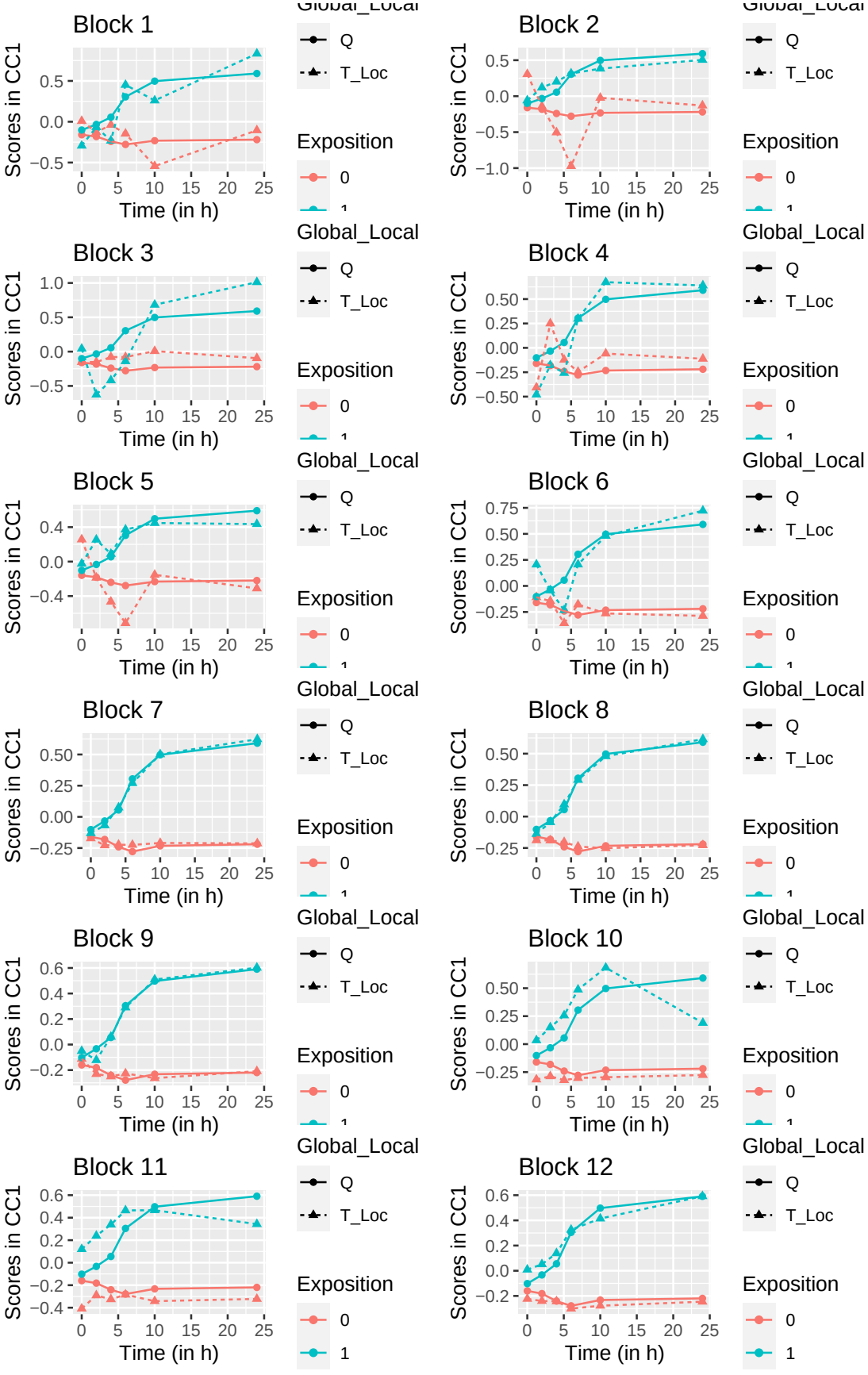
This analysis revealed not to be very useful since there is too much data plotted together. So, in the following section, we will plot the data from each block separately. Also, in the same plots we will overlay the *Global scores* to compare the global and the local trend.

10 Global and Local scores

The *Global scores* show the common trend of the samples *to all the blocks*. For this example, we only work with *CC1*.

```
library(ggplot2)
library(tidyr)
```

```r
library(dplyr)
library(gridExtra)

df <- list()
df2 <- list()
p2 <- list()

for(i in 1:12){
  df[[i]] <- data.frame(T_Loc = results$T_Loc$Data[[i]][,1],
                Q = results$Q$Data[,1],
                Time = results$Q$time,
                Exposition = as.factor(results$Q$exposition))

  df2[[i]] <- df[[i]] %>%
        pivot_longer(cols = -c('Time','Exposition'), #These columns are extracted as they are.
               names_to = 'Global_Local', # The new column to be created.
               values_to = 'Scores_value')

  p2[[i]] <- ggplot(data=df2[[i]], aes(x=Time, y=Scores_value,
        linetype = Global_Local,
        shape = Global_Local,
        color = Exposition)) +
        geom_point() +
        geom_line() +
        labs(x = "Time (in h)", y = "Scores in CC1", title = sprintf("Block %s",i))
}

grid.arrange(p2[[1]],p2[[2]],p2[[3]],p2[[4]],
             p2[[5]],p2[[6]],p2[[7]],p2[[8]],
             p2[[9]],p2[[10]],p2[[11]],p2[[12]], nrow = 6)
```

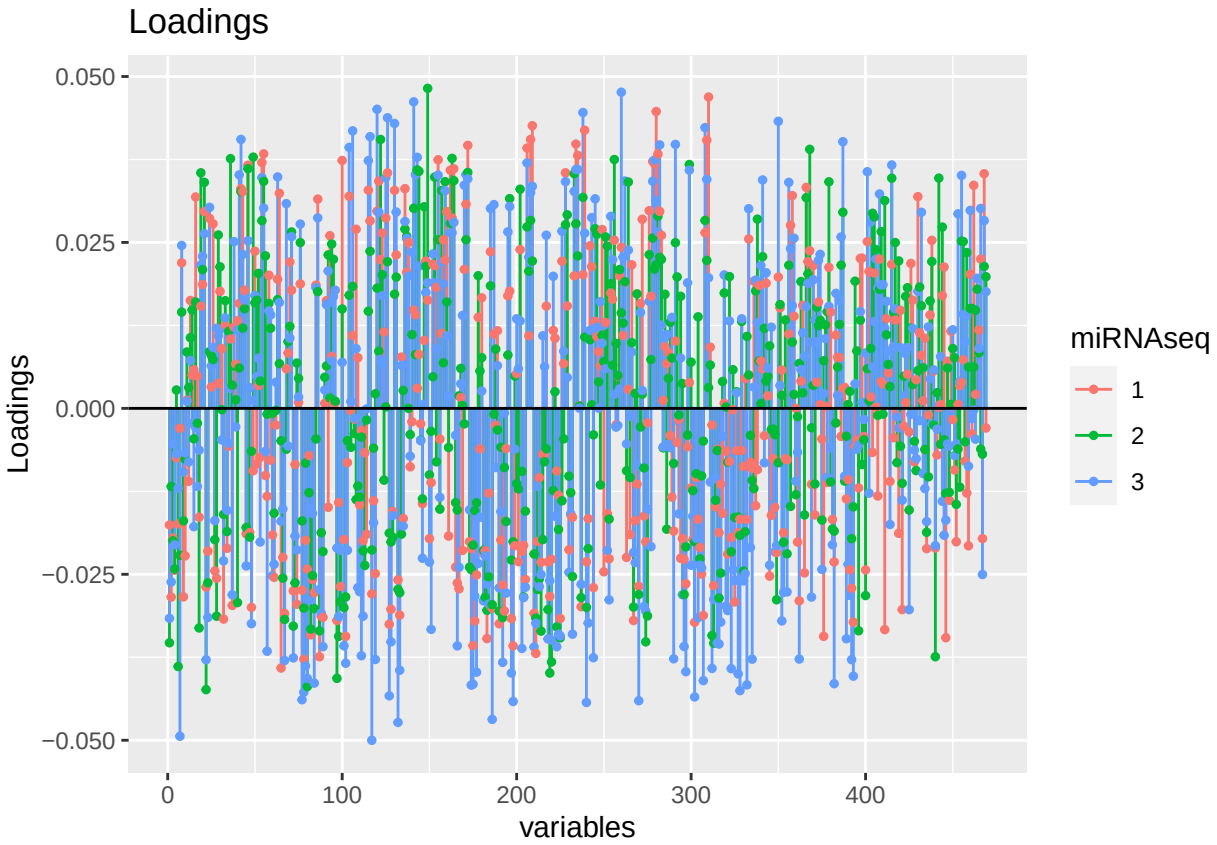

Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
Block 7
Block 8
Block 9
Block 10
Block 11
Block 12
Scores in CC1
Time (in h)
Global_Local
Q
T_Loc
Exposition
0
1

Now, we can see easily that the *Local Scores* that better matches the *Global scores* are the RNAseq (blocks 7-9) and the miRNAseq (blocks 10-12). Having said this, the evolution of the two types of metabolomics data is also very close to the *Global scores*.

11 Loadings

The *Loadings* show the contributions of each of the *variables* to the global model. In this example, we compare the loadings for the 3 miRNAseq blocks in CC1

```
library(ggplot2)
library(tidyr)
library(dplyr)

df <- data.frame(miRNAseq1 = results$P_Loc$Data[[10]][,1], # CC1 from block 10
                 miRNAseq2 = results$P_Loc$Data[[11]][,1], # CC1 from block 11
                 miRNAseq3 = results$P_Loc$Data[[12]][,1], # CC1 from block 12
                 variables = seq(1,length(results$P_Loc$Data[[10]][,1]),1))

df2 <- df %>%
  pivot_longer(cols = -c('variables'), names_to = 'miRNAseq',
               names_prefix = "miRNAseq", values_to = 'Loadings')

ggplot(data = df2, aes(x = variables, y = Loadings, col = miRNAseq)) +
    geom_segment(aes(variables,Loadings,xend=variables,yend=Loadings-Loadings)) +
    geom_point(aes(variables,Loadings),size=1) +
    geom_hline(aes(yintercept=0)) +
    ggtitle("Loadings")
```



We can appreciate a good correspondence of the loadings across the 3 miRNAseq blocks.

12 Additional settings for ComDim

12.1 Loquace

For long computations, it is possible to display the progress of the *ComDim* analysis by setting the *loquace* parameter to *TRUE*.

```
results <- comdim_PCA(data = rb_d3, 4, loquace = TRUE) # We will display the computational time.
```

```
##   |                                                                      |
## [1] "Checking sample correspondence across blocks."
## [1] "The data can be used for ComDim."
## [1] "Initialisation finished after : 0 millisecs"
##   |                                                                      |==
##   |                                                                      |====
##   |                                                                      |=======
##   |                                                                      |=========
##   |                                                                      |===========
##   |                                                                      |=============
##   |                                                                      |===============
##   |                                                                      |==================
##   |                                                                      |=================
```

```
##   |                                                                      |==================
##   |                                                                      |==================
##   |                                                                      |==================
##   |                                                                      |==================
##   |                                                                      |==================
##   |                                                                      |==================
##   |                                                                      |==================
##   |                                                                      |==================
## [1] "Scores finished after : 3952.20017433167 millisecs"
##   |                                                                      |==================
##   |                                                                      |==================
##   |                                                                      |
```

12.2 Normalise

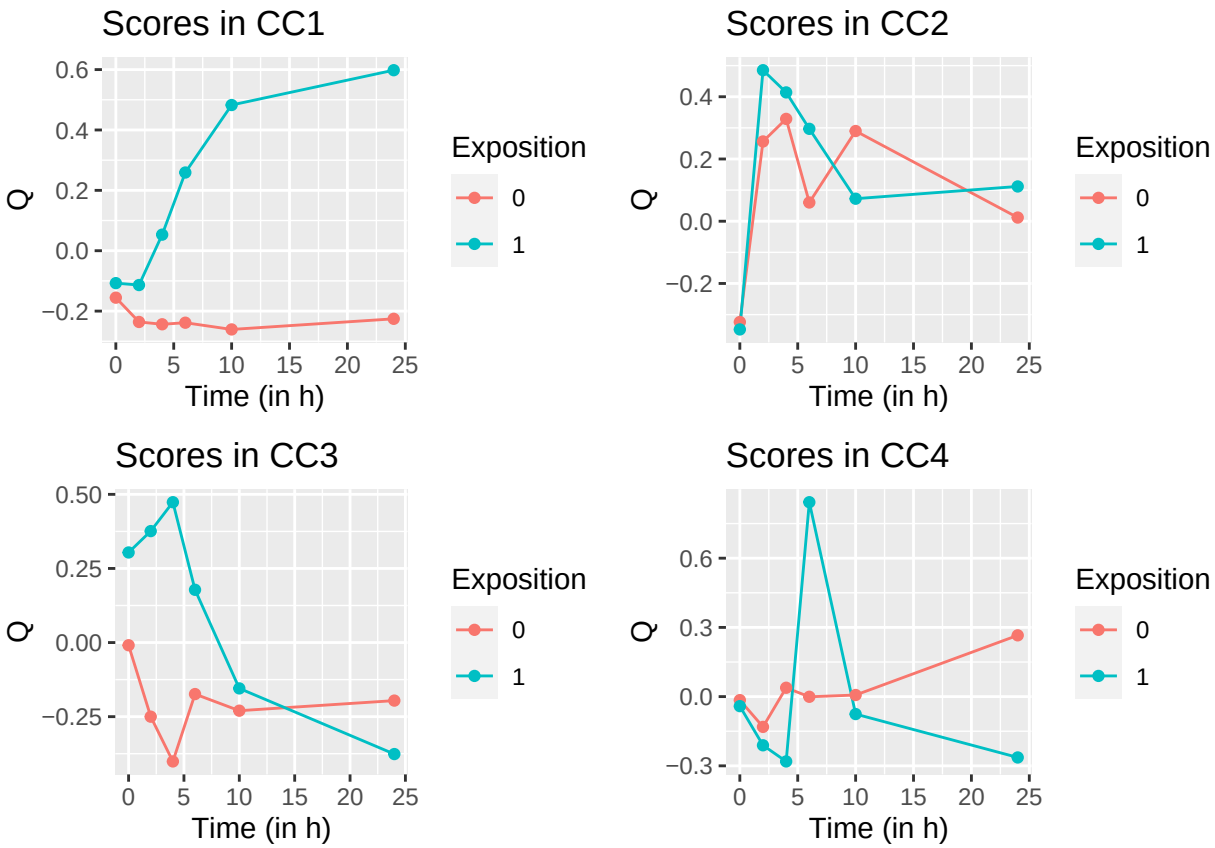
In the default settings, the data blocks are mean-centered and norm-scaled before the iteration process of determining the global scores and saliences. These two pre-processings can be omitted by setting *normalise* to *FALSE*. Having said this, for most of the times, the best results are obtained with *normalise = TRUE*.

```
results <- comdim_PCA(data = rb_d3, 4, normalise = FALSE)
```

```
##   |                                                                      |
## [1] "Checking sample correspondence across blocks."
## [1] "The data can be used for ComDim."
##   |                                                                      |==
##   |                                                                      |==================
```

```
  # The MB data is not mean-centered nor norm-scaled inside the ComDim routine.

# Plot of the Global scores
library(ggplot2)
library(gridExtra)

df_Q1 <- data.frame(Q = results$Q$Data[,1],
                Time = results$Q$time,
                Exposition = as.factor(results$Q$exposition))

df_Q2 <- data.frame(Q = results$Q$Data[,2],
                Time = results$Q$time,
                Exposition = as.factor(results$Q$exposition))

df_Q3 <- data.frame(Q = results$Q$Data[,3],
                Time = results$Q$time,
                Exposition = as.factor(results$Q$exposition))

df_Q4 <- data.frame(Q = results$Q$Data[,4],
                Time = results$Q$time,
                Exposition = as.factor(results$Q$exposition))

p1 <- ggplot(data=df_Q1, aes(x=Time, y=Q, group=Exposition)) +
  geom_line(aes(color=Exposition))+
```

```
  geom_point(aes(color=Exposition))+
  labs(x = "Time (in h)", title = "Scores in CC1")

p2 <- ggplot(data=df_Q2, aes(x=Time, y=Q, group=Exposition)) +
  geom_line(aes(color=Exposition))+
  geom_point(aes(color=Exposition))+
  labs(x = "Time (in h)", title = "Scores in CC2")

p3 <- ggplot(data=df_Q3, aes(x=Time, y=Q, group=Exposition)) +
  geom_line(aes(color=Exposition))+
  geom_point(aes(color=Exposition))+
  labs(x = "Time (in h)", title = "Scores in CC3")

p4 <- ggplot(data=df_Q4, aes(x=Time, y=Q, group=Exposition)) +
  geom_line(aes(color=Exposition))+
  geom_point(aes(color=Exposition))+
  labs(x = "Time (in h)")+
  labs(x = "Time (in h)", title = "Scores in CC4")

grid.arrange(p1, p2, p3, p4, nrow = 2)
```



In this case, we did not observe many differences since the 4 data-blocks were already pre-processed (SNV-normalised and mean-centered).

12.3 CompMethod and Partitions

For large matrices, it is possible to speed up the analysis with the parameters *CompMethod* and *Partitions*.

```
running_time <- as.vector(NULL)
results <- comdim_PCA(data = rb_d3, 4)
```

```
##   |                                                                      |
## [1] "Checking sample correspondence across blocks."
## [1] "The data can be used for ComDim."
##   |                                                                      |==
##   |                                                                      |==================
```

```
running_time[1] <- results$runtime
results <- comdim_PCA(data = rb_d3, 4, Partitions = 3)
```

```
##   |                                                                      |
## [1] "Checking sample correspondence across blocks."
## [1] "The data can be used for ComDim."
##   |                                                                      |==
##   |                                                                      |==================
```

```
running_time[2] <- results$runtime
results <- comdim_PCA(data = rb_d3, 4, CompMethod = 'Kernel')
```

```
##   |                                                                      |
## [1] "Checking sample correspondence across blocks."
## [1] "The data can be used for ComDim."
##   |                                                                      |==
##   |                                                                      |==================
```

```
running_time[3] <- results$runtime
results <- comdim_PCA(data = rb_d3, 4, CompMethod = 'PCT')
```

```
##   |                                                                      |
## [1] "Checking sample correspondence across blocks."
## [1] "The data can be used for ComDim."
##   |                                                                      |==
##   |                                                                      |==================
```

```
running_time[4] <- results$runtime
results <- comdim_PCA(data = rb_d3, 4, CompMethod = 'Kernel', Partitions = 3)
```

```
##   |                                                                      |
## [1] "Checking sample correspondence across blocks."
## [1] "The data can be used for ComDim."
##   |                                                                      |==
##   |                                                                      |==================
```

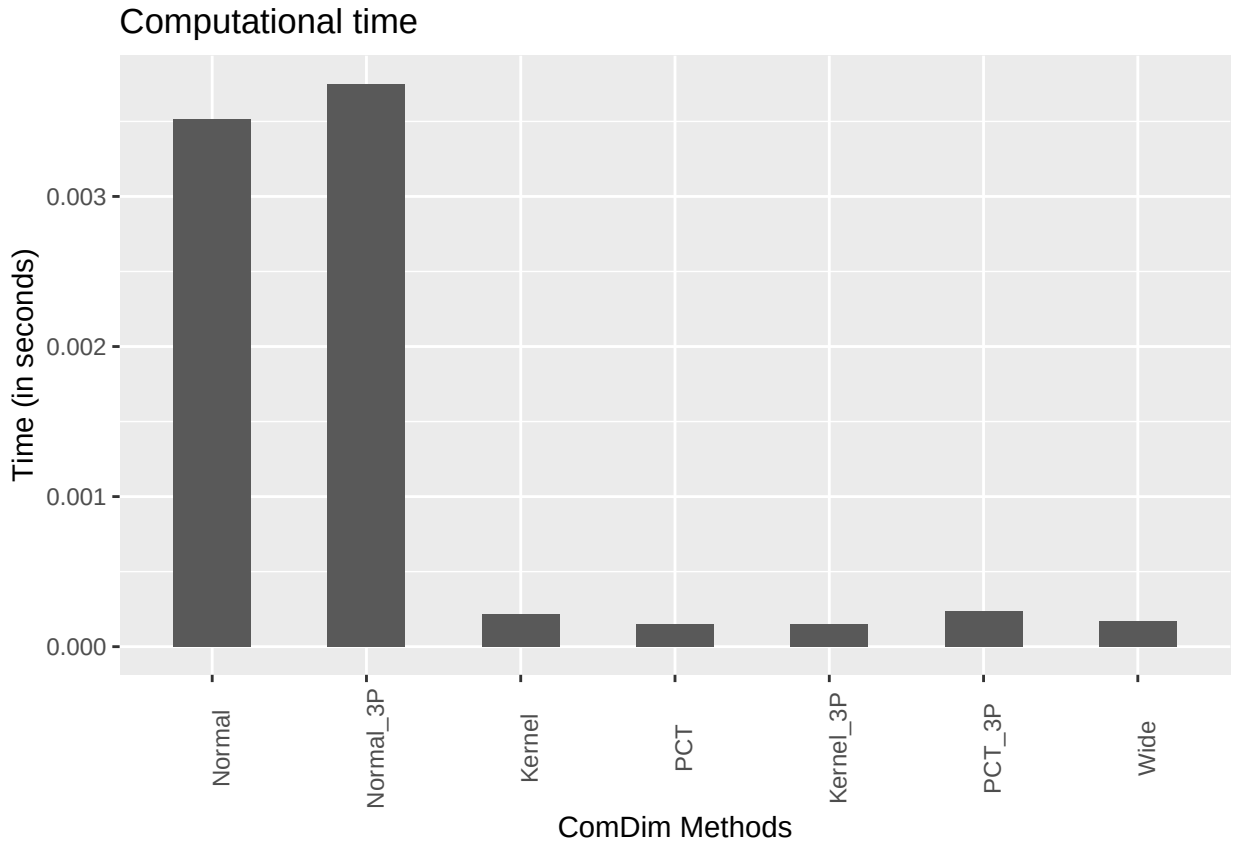
```
running_time[5] <- results$runtime
results <- comdim_PCA(data = rb_d3, 4, CompMethod = 'PCT', Partitions = 3)
```

```
##   |                                                                      |
## [1] "Checking sample correspondence across blocks."
## [1] "The data can be used for ComDim."
##   |                                                                      |==
##   |                                                                      |==================
```

```
running_time[6] <- results$runtime
results <- comdim_PCA(data = rb_d3, 4, CompMethod = 'Wide')
```

```
##   |                                                                      |
## [1] "Checking sample correspondence across blocks."
## [1] "The data can be used for ComDim."
##   |                                                                      |==
##   |                                                                      |==================
```

```
running_time[7] <- results$runtime
#results <- comdim_PCA(data = rb_d3, 4, loquace = TRUE, CompMethod = 'Tall')
    # Not recommended when the number of variables in a block is very large
    # As for the RNAseq blocks here.

#running_time[8] <- results$runtime

running_time <- running_time/1000 # To convert milliseconds to seconds

library(ggplot2)
df <- data.frame(Tests = c('Normal','Normal_3P','Kernel','PCT','Kernel_3P','PCT_3P','Wide'),
                 Time = running_time,
                 stringsAsFactors = FALSE)

ggplot(data=df, aes(x = Tests, y = running_time)) +
  geom_bar(stat="identity", width=0.5) +
  scale_x_discrete(limits=df$Tests) +
  labs(y = "Time (in seconds)", x='ComDim Methods') +
  ggtitle("Computational time") +
  theme(axis.text.x = element_text(angle = 90))
```



13 References

- Gomez-Cabrero, D.; Tarazona, S.; Ferreirós-Vidal, I.; Ramirez, R. N.; Company, C.; Schmidt, A.; Reijmers, T.; Paul, V. von Saint; Marabita, F.; Rodríguez-Ubreva, J.; Garcia-Gomez, A.; Carroll, T.; Cooper, L.; Liang, Z.; Dharmalingam, G.; van der Kloet, F.; Harms, A. C.; Balzano-Nogueira, L.; Lagani, V.; Tsamardinos, I.; Lappe, M.; Maier, D.; Westerhuis, J. A.; Hankemeier, T.; Imhof, A.; Ballestar, E.; Mortazavi, A.; Merkenschlager, M.; Tegner, J.; Conesa, A. STATegra, a Comprehensive Multi-Omics Dataset of B-Cell Differentiation in Mouse. Sci. data 2019, 6 (1), 256. https://doi.org/10.1038/s41597-019-0202-7.
- Puig-Castellví, F.; Jouan-Rimbaud Bouveresse, D.; Mazéas, L.; Chapleur, O.; Rutledge*, D. N. Rearrangement of incomplete multi-omics datasets combined with ComDim for evaluating replicate cross-platform variability and batch influence. Under revision.
- Qannari, E. M.; Courcoux, P.; Vigneau, E. Common Components and Specific Weights Analysis Performed on Preference Data. Food Qual. Prefer. 2001, 12 (5–7), 365–368. https://doi.org/10.1016/S0950-3293(01)00026-X.
- Mazerolles, G.; Hanafi, M.; Dufour, E.; Bertrand, D.; Qannari, E. M. Common Components and Specific Weights Analysis: A Chemometric Method for Dealing with Complexity of Food Products. Chemom. Intell. Lab. Syst. 2006, 81 (1), 41–49. https://doi.org/10.1016/J.CHEMOLAB.2005.09.004.
- Claeys-Bruno, M.; Béal, A.; Rutledge, D. N.; Sergent, M. Use of the Common Components and Specific Weights Analysis to Interpret Supersaturated Designs. Chemom. Intell. Lab. Syst. 2016, 152, 97–106. https://doi.org/10.1016/j.chemolab.2016.01.014.