Supplementary Information: *Rtpca: an R package for differential thermal coaggregation analysis*

***Nils Kurzawa*[1], *André Mateus*[1], *and Mikhail M. Savitski*[1]**

[1]European Molecular Biology Laboratory (EMBL), Genome Biology Unit

08 April, 2020

Package

Rtpca 0.0.99

Contents

1 Introduction

Thermal proteome profiling (TPP) (Mateus et al., 2020; Savitski et al., 2014) is a mass spectrometry-based, proteome-wide implemention of the cellular thermal shift assay (Molina et al., 2013). It was originally developed to study drug-(off-)target engagement. However, it was realized that profiles of interacting protein pairs appeared more similar than by chance (Tan et al., 2018, Becher et al. (2018)) which was coined as 'thermal proximity co-aggregation' (TPCA) (Tan et al., 2018). The R package `Rtpca` enables analysis of TPP datasets using the TPCA concept for studying protein-protein interactions and protein complexes and also allows to test for differential protein-protein interactions (PPIs) across different conditions.
Here, we exemplify the analysis based on a dataset by Becher et al. (2018) which provides temperature range TPP (TPP-TR) experiments for synchronized HeLa cells in G1/S cell cycle stage versus M phase.
Note: The paper by Becher et al. (2018) also includes 2D-TPP (Becher et al., 2016) data which is in general more sensitive to changes in protein abundance or stability. This data can also be informative on dynamics of protein-protein interactions based on correlations analysis of 2D-TPP profiles of annotated interactors. However, the advantage of TPP-TR data is that one can test for coaggregation which, if significant, is directly indicative of protein-protein interaction or complex assembly.

2 Step-by-step walk through the data analysis

First, we need to load the required libraries (these need to be installed as specified in the comments):

```
library(dplyr) # install.packages("dplyr")
library(readxl) # install.packages("readxl")
library(Rtpca) # require(devtools); devtools::install_github("nkurzaw/Rtpca")
library(ggplot2) # install.packages("ggplot2")
```

Then, we download the supplementary data from Tan et al. which contains the TPP data which we'll be using:

```
if(!file.exists("1-s2.0-S0092867418303854-mmc4.xlsx")){
    download.file(
        url = "https://ars.els-cdn.com/content/image/1-s2.0-S0092867418303854-mmc4.xlsx",
        destfile = "1-s2.0-S0092867418303854-mmc4.xlsx")
}
```

Next, we read in the annotation information of the supplementary table as a data frame

```
supp_tab_becher_s4 <- read_xlsx("1-s2.0-S0092867418303854-mmc4.xlsx",
                                sheet = "TableS4_TPP-TR")

temperature_anno <-
    as.numeric(
        gsub("T", "", gsub("_.+", "", colnames(
            supp_tab_becher_s4 %>%
                dplyr::select(matches("mean\\.fc"))))))
```

We then extract the data for G1/S:

```
g1s_df <- supp_tab_becher_s4 %>%
    filter(cell.cycle == "G1_S") %>%
    dplyr::select(
        gene_name,
        replicates = found.in.reps,
        max_qupm = max.qupm,
        min_qupm = min.qupm,
        matches("mean\\.fc")) %>%
    filter(min_qupm > 3,
           replicates == 3)
```

```
g1s_mat <- as.matrix(
    g1s_df %>% dplyr::select(dplyr::matches("mean\\.fc"))
    )
rownames(g1s_mat) <- g1s_df$gene_name
attributes(g1s_mat)$temperature <- temperature_anno
```

And for M phase:

```
m_df <- supp_tab_becher_s4 %>%
    filter(cell.cycle == "M") %>%
    dplyr::select(
        gene_name,
        replicates = found.in.reps,
        max_qupm = max.qupm,
        min_qupm = min.qupm,
        matches("mean\\.fc")) %>%
    filter(min_qupm > 3,
           replicates == 3)
```

```
m_mat <- as.matrix(
    m_df %>% dplyr::select(dplyr::matches("mean\\.fc"))
    )
rownames(m_mat) <- m_df$gene_name
attributes(m_mat)$temperature <- temperature_anno
```

We then load an annotation of mammalian complexes by Ori et al. (2016), which comes with the `Rtpca` package:

```
data("ori_et_al_complexes_df")
```

2.1 Complex-centric analysis

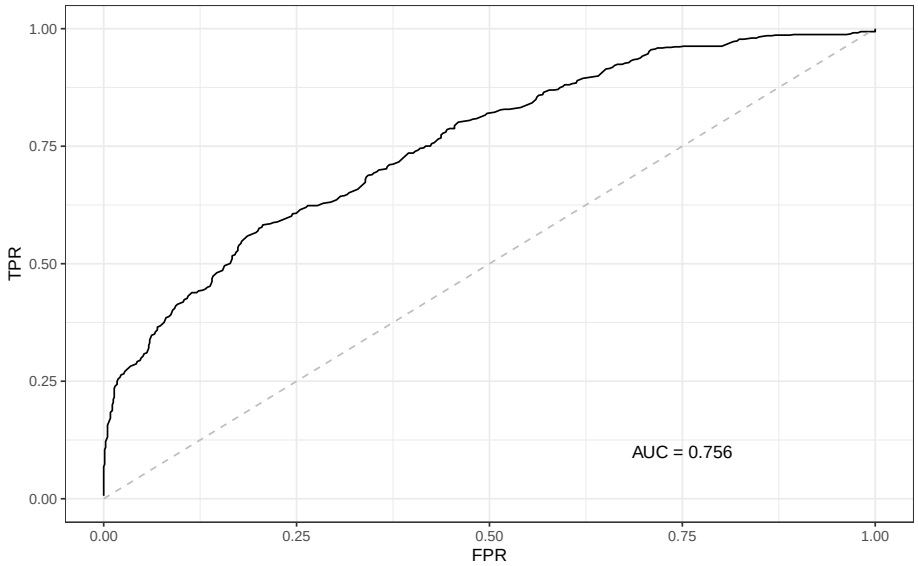
Now, we perform a TPCA anaylsis based on complexes only in the G1/S condition:

```
G1S_TPCA <- runTPCA(
    objList = list(g1s_mat),
    complexAnno = ori_et_al_complexes_df)
## Checking input arguments.
```

```
##
## Creating distance matrices.
##
## Testing for complex co-aggregation.
##
## Performning Complex ROC analysis.
```

We can plot the ROC curve for how predictive our data is on recovering protein complexes by evoking:

```
plotComplexRoc(G1S_TPCA, computeAUC = TRUE)
```



And we can inspect significantly co-melting protein complexes, like this:

```
G1S_TPCA@tpcaResultTable %>% filter(p_adj < 0.1)
## # A tibble: 45 x 5
##    complex_name                     count mean_dist  p_value    p_adj
##    <chr>                            <int>     <dbl>    <dbl>    <dbl>
##  1 26S Proteasome                      33    0.441  1.67e- 2 7.08e- 2
##  2 Nuclear pore complex (NPC)          24    0.341  2.00e- 4 2.30e- 3
##  3 BAF complex                          7    0.214  6.00e- 4 6.44e- 3
##  4 Spliceosome-U2                       9    0.333  1.54e- 2 7.05e- 2
##  5 Anaphase promoting complex (APC)     4    0.0948 2.22e-16 3.25e-15
##  6 multi-tRNAsynthase complex          10    0.123  2.22e-16 3.25e-15
##  7 RNA polymerase III core complex      3    0.189  2.41e- 2 8.82e- 2
##  8 RNA polymerase II core complex       4    0.190  8.70e- 3 5.19e- 2
##  9 COP9 signalosome                     8    0.236  1.00e- 3 9.47e- 3
## 10 MCM complex                          7    0.190  2.00e- 4 2.30e- 3
## # ... with 35 more rows
```

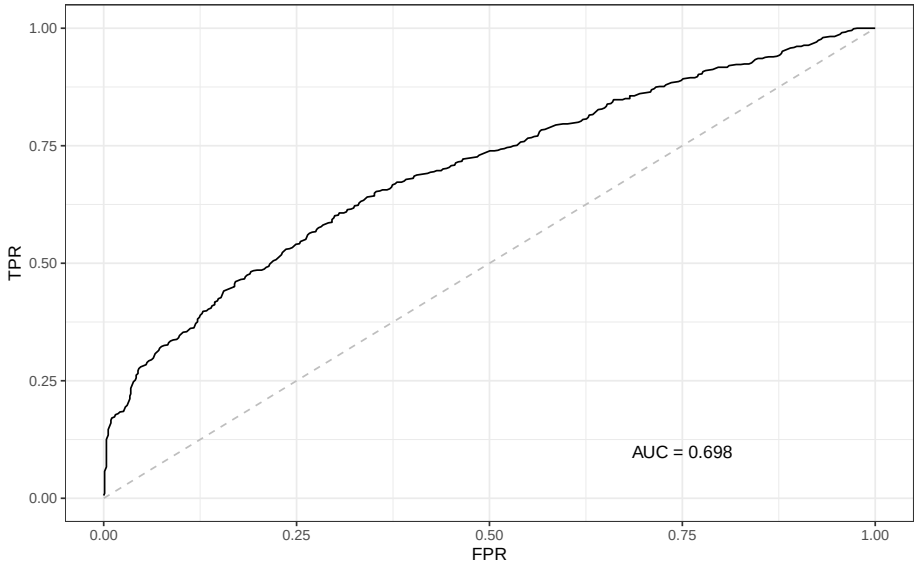
```
g1s_significant_complex_comelting <-
    filter(G1S_TPCA@tpcaResultTable,p_adj < 0.1)$complex_name
```

Next, we perform the same analysis for only the M phase condition:

```
M_TPCA <- runTPCA(
    objList = list(m_mat),
    complexAnno = ori_et_al_complexes_df)
## Checking input arguments.
##
## Creating distance matrices.
##
## Testing for complex co-aggregation.
##
## Performning Complex ROC analysis.
```

We can see that the predictive performance of this dataset for protein complexes is not quite as good as for the G1/S one:

```
plotComplexRoc(M_TPCA, computeAUC = TRUE)
```




```
M_TPCA@tpcaResultTable %>% filter(p_adj < 0.1)
## # A tibble: 56 x 5
##    complex_name                        count mean_dist  p_value    p_adj
##    <chr>                               <int>     <dbl>    <dbl>    <dbl>
##  1 Nuclear pore complex (NPC)             25    0.332  1.10e- 3 1.25e- 2
##  2 BAF complex                             9    0.140  2.22e-16 4.22e-15
##  3 Integrator                              7    0.254  1.10e- 2 4.87e- 2
##  4 NuRD complex                            9    0.254  4.10e- 3 2.34e- 2
##  5 Anaphase promoting complex (APC)        4    0.166  1.13e- 2 4.87e- 2
##  6 Cohesin complex                         7    0.245  9.00e- 3 4.53e- 2
##  7 Transcription-export (TREX) complex     8    0.312  2.51e- 2 7.80e- 2
##  8 multi-tRNAsynthase complex             10    0.0811 2.22e-16 4.22e-15
##  9 RANBP9-containing complex               3    0.0898 2.90e- 3 2.05e- 2
## 10 RNA polymerase II core complex          4    0.174  1.48e- 2 5.50e- 2
## # ... with 46 more rows
```

Based on the protein complexes which we find significantly assembled in either condition, we will select the protein-protein interactions to test for in a differential TPCA:

```
m_significant_complex_comelting <-
    filter(M_TPCA@tpcaResultTable,p_adj < 0.1)$complex_name

all_significant_complex_comelting <-
    unique(c(g1s_significant_complex_comelting,
             m_significant_complex_comelting))
```

We load the annotation of protein-protein interactions within complexes that is composed of PPIs from StringDb (Szklarczyk et al., 2019) and the complex annotation by Ori et al. (2016) and filter it for protein complexes that we have seen to coaggregate in the analysis above.

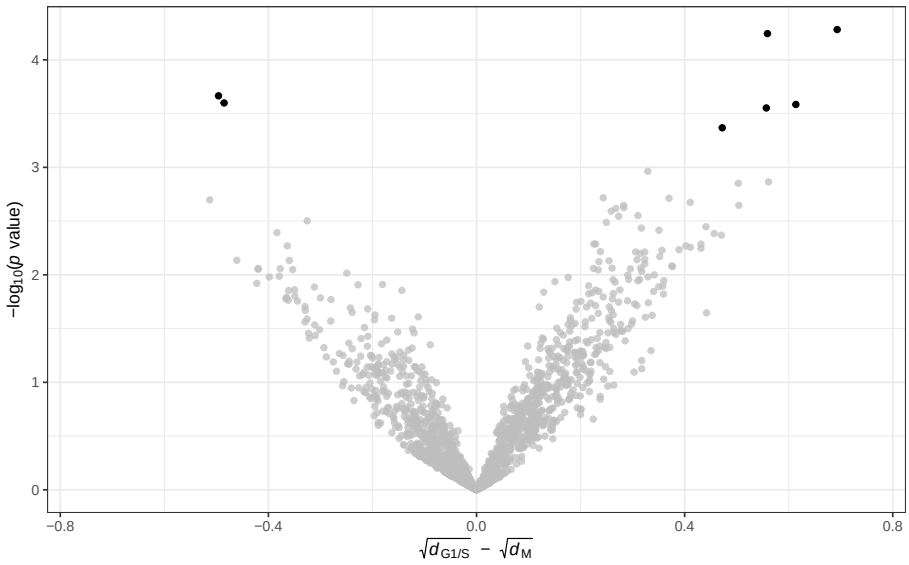
```
data("ori_et_al_complex_ppis")

filtered_complex_ppis <- ori_et_al_complex_ppis %>%
    filter(complex_name %in% all_significant_complex_comelting)
```

We now run the differential TPCA by evoking:

```
set.seed(123)
M_vs_G1S_diff_TPCA <- runDiffTPCA(
    objList = list(g1s_mat),
    contrastList = list(m_mat),
    ctrlCondName = "G1/S",
    contrastCondName = "M",
    ppiAnno = filtered_complex_ppis,
    n = 10^6
)
## Checking input arguments.
## Creating distance matrices.
## Comparing annotated protein-pairs across conditions.
## Comparing random protein-pairs across conditions.
## Generating result table.
```

We can now plot the result in form of a volcano plot:

```
plotDiffTpcaVolcano(M_vs_G1S_diff_TPCA,
                    setXLim = TRUE,
                    xlimit = c(-0.75, 0.75))
```



As a result, we obtain an `tpcaResult` object which looks like this:

```
M_vs_G1S_diff_TPCA
## class: tpcaResult
## Slot "ObjList": of class list and length 1
## Slot "ContrastList": of class list and length 1
## Slot "DistMat" with dimension: 2658 2658
## Slot "ContrastDistMat" with dimension: 3086 3086
## Slot "ComplexAnnotation" of class: data.frame with dim: 0 0
## Slot "ComplexBackgroundDistributionList" of class: list with length: 0
## Slot "PPiAnnotation" of class: tbl_df tbl data.frame with dim: 4466 5
## Slot "PPiRocTable" of class: data.frame with dim: 0 0
## Slot "PPiRocTableAnno" of class: data.frame with dim: 0 0
## Slot "ComplexRocTable" of class: data.frame with dim: 0 0
## Slot "summaryMethod": median
## Slot "distMethod": euclidean
## Slot "tpcaResultTable" of class: data.frame with dim: 0 0
## Slot "diffTpcaResultTable" of class: tbl_df tbl data.frame with dim: 1549 10
```

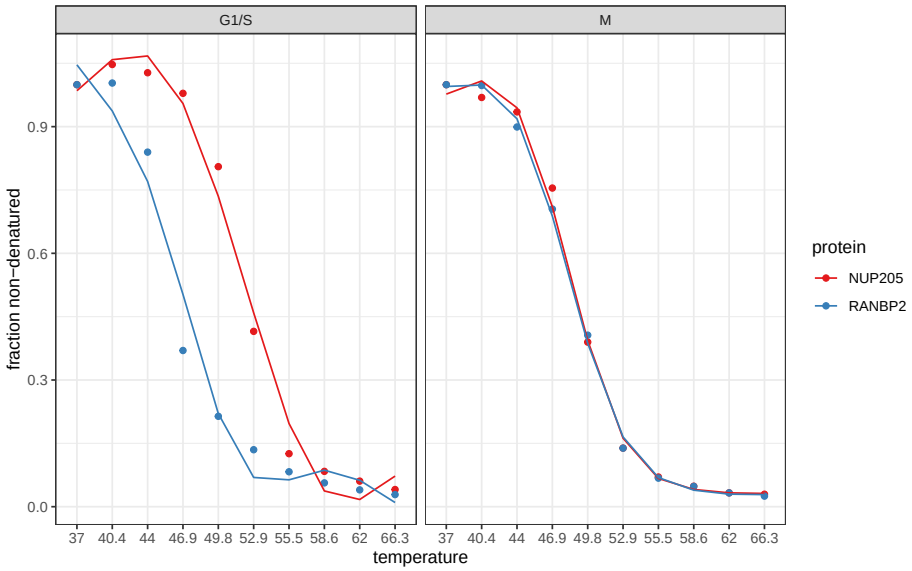
We can now inspect the significant results:

```
M_vs_G1S_diff_TPCA@diffTpcaResultTable %>%
    dplyr::select(pair, rssC1_rssC2, p_value, p_adj) %>%
    arrange(p_value)
## # A tibble: 1,549 x 4
##    pair          rssC1_rssC2   p_value  p_adj
##    <chr>               <dbl>     <dbl>  <dbl>
##  1 NUP205:RANBP2       0.834  0.0000523 0.0442
##  2 NUP88:RANBP2        0.357  0.0000570 0.0442
##  3 RPS6:RPSA          -0.279  0.000216  0.0724
##  4 RPS23:RPSA         -0.263  0.000252  0.0724
##  5 NUP93:RANBP2        0.679  0.000260  0.0724
##  6 NUP188:RANBP2       0.470  0.000280  0.0724
##  7 AAAS:TPR            0.260  0.000429  0.0949
```

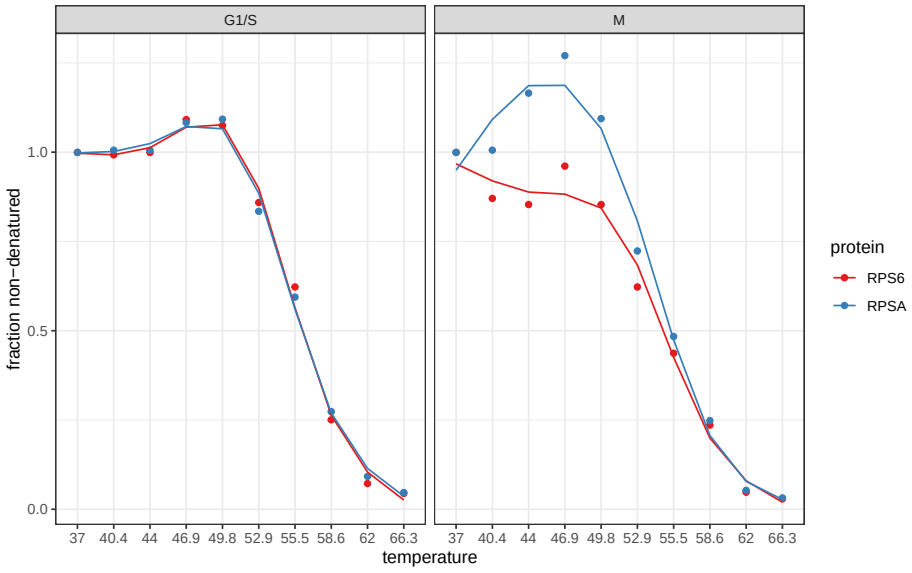
```
##  8 PSMB1:PSMB4        0.0786 0.00109   0.207
##  9 NUP54:RANBP2       0.710  0.00137   0.207
## 10 NUP107:RANBP2      0.461  0.00141   0.207
## # ... with 1,539 more rows
```

To validate significant PPIs we can inspect their melting curves:

```
plotPPiProfiles(M_vs_G1S_diff_TPCA, pair = c("NUP205", "RANBP2"))
```




```
plotPPiProfiles(M_vs_G1S_diff_TPCA, pair = c("RPS6", "RPSA"))
```



2.2 PPI-centric analysis

First, we load annotated PPIs by StringDb (Szklarczyk et al., 2019):

```
data("string_ppi_df")

string_ppi_975_df <- string_ppi_df %>%
    filter(combined_score >= 975)
```

The we start our analysis based on PPIs:

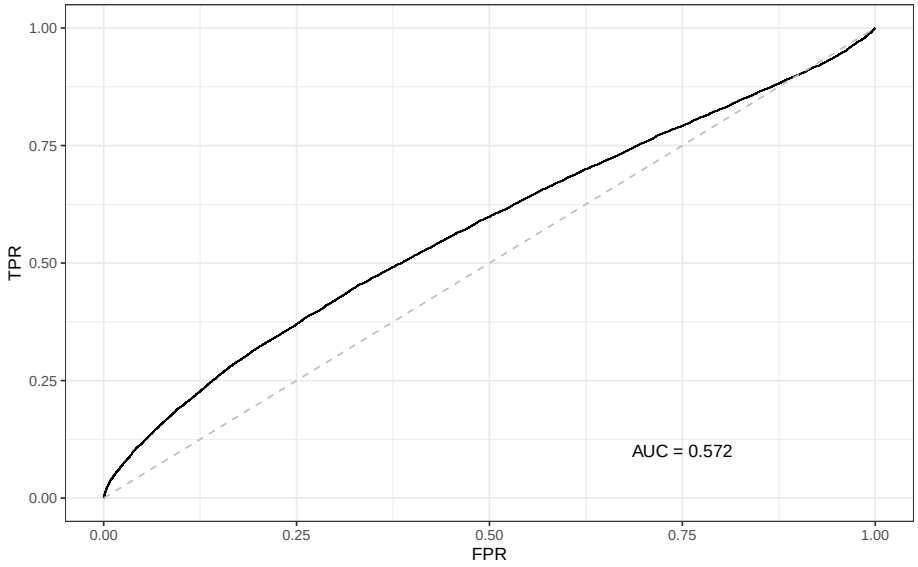
```
G1S_PPI_TPCA <- runTPCA(
    objList = list(g1s_mat),
    ppiAnno = string_ppi_975_df,
    nSamp = 10^6)
## Checking input arguments.
##
## Creating distance matrices.
##
## Testing for complex co-aggregation.
##
## Performing PPi ROC analysis.
```

As for the complex-centric analysis we get back a `tpcaResult` object:

```
G1S_PPI_TPCA
## class: tpcaResult
## Slot "ObjList": of class list and length 1
## Slot "ContrastList": of class list and length 0
## Slot "DistMat" with dimension: 2658 2658
## Slot "ContrastDistMat" with dimension: 0 0
## Slot "ComplexAnnotation" of class: tbl_df tbl data.frame with dim: 16460 3
## Slot "ComplexBackgroundDistributionList" of class: list with length: 1
## Slot "PPiAnnotation" of class: tbl_df tbl data.frame with dim: 39176 4
## Slot "PPiRocTable" of class: tbl_df tbl data.frame with dim: 3531153 3
## Slot "PPiRocTableAnno" of class: tbl_df tbl data.frame with dim: 3531153 2
## Slot "ComplexRocTable" of class: data.frame with dim: 0 0
## Slot "summaryMethod": median
## Slot "distMethod": euclidean
## Slot "tpcaResultTable" of class: tbl_df tbl data.frame with dim: 8230 5
## Slot "diffTpcaResultTable" of class: data.frame with dim: 0 0
```

And we can also inpect a ROC curve for this analysis:

```
plotPPiRoc(G1S_PPI_TPCA, computeAUC = TRUE)
```



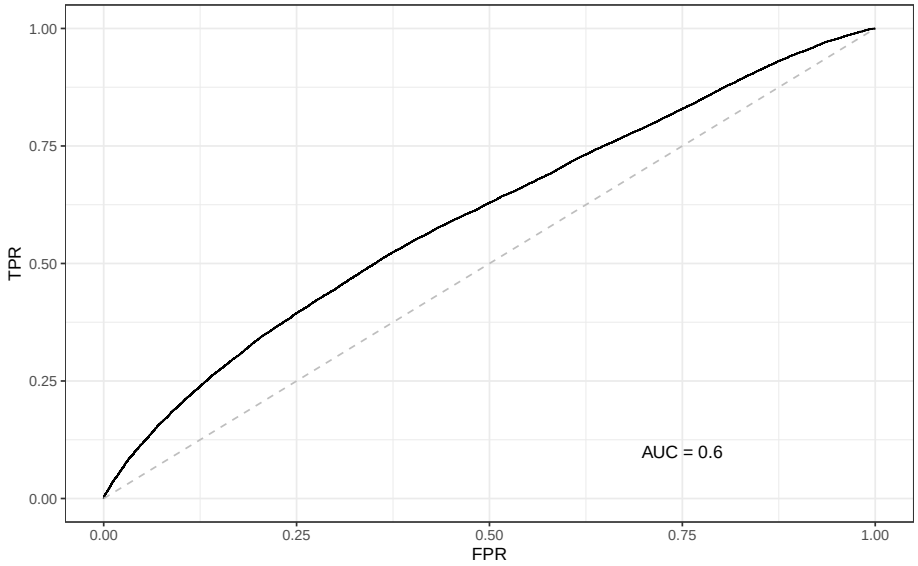
To inspect which PPIs coaggregated significantly, we can evoke:

```
G1S_PPI_TPCA@tpcaResultTable %>% filter(p_adj < 0.1) %>% arrange(p_value)
## # A tibble: 25 x 5
##    complex_name  count mean_dist  p_value    p_adj
##    <chr>         <int>     <dbl>    <dbl>    <dbl>
##  1 EIF3D:EIF3E       2    0.0165 2.22e-16 3.05e-13
##  2 HSPA1A:HSPA1B     2    0      2.22e-16 3.05e-13
##  3 IARS:LARS         2    0.0266 2.22e-16 3.05e-13
##  4 IARS:RARS         2    0.0227 2.22e-16 3.05e-13
##  5 LAMA5:LAMC1       2    0.0262 2.22e-16 3.05e-13
##  6 MCM2:MCM4         2    0.0220 2.22e-16 3.05e-13
##  7 CANX:GANAB        2    0.0298 1.00e- 4 5.88e- 2
##  8 CCT2:CCT3         2    0.0331 1.00e- 4 5.88e- 2
##  9 ERLIN1:ERLIN2     2    0.0301 1.00e- 4 5.88e- 2
## 10 EXOC2:EXOC8       2    0.0328 1.00e- 4 5.88e- 2
## # ... with 15 more rows
```

And we can run the same analysis for the M-phase dataset:

```
M_PPI_TPCA <- runTPCA(
    objList = list(m_mat),
    ppiAnno = string_ppi_975_df,
    nSamp = 10^6)
## Checking input arguments.
##
## Creating distance matrices.
##
## Testing for complex co-aggregation.
##
## Performing PPi ROC analysis.
```

```
plotPPiRoc(M_PPI_TPCA, computeAUC = TRUE)
```




```
M_PPI_TPCA@tpcaResultTable %>% filter(p_adj < 0.1) %>% arrange(p_value)
## # A tibble: 5 x 5
##   complex_name  count mean_dist  p_value    p_adj
##   <chr>         <int>     <dbl>    <dbl>    <dbl>
## 1 AP2A1:AP2M1       2    0.0198 2.22e-16 4.52e-13
## 2 CYFIP1:NCKAP1     2    0.0188 2.22e-16 4.52e-13
## 3 HADHA:HADHB       2    0.0198 2.22e-16 4.52e-13
## 4 HSPA1A:HSPA1B     2    0      2.22e-16 4.52e-13
## 5 NAE1:UBA3         2    0.0127 2.22e-16 4.52e-13
```

By now combining the significantly found coaggregating PPIs (we are a bit less stringent on the adjusted p-value filter here to not reduce the space of possible differential PPIs too strongly), we can define a list of PPIs which we can use to test for differential PPIs across the two cell cycle phases:

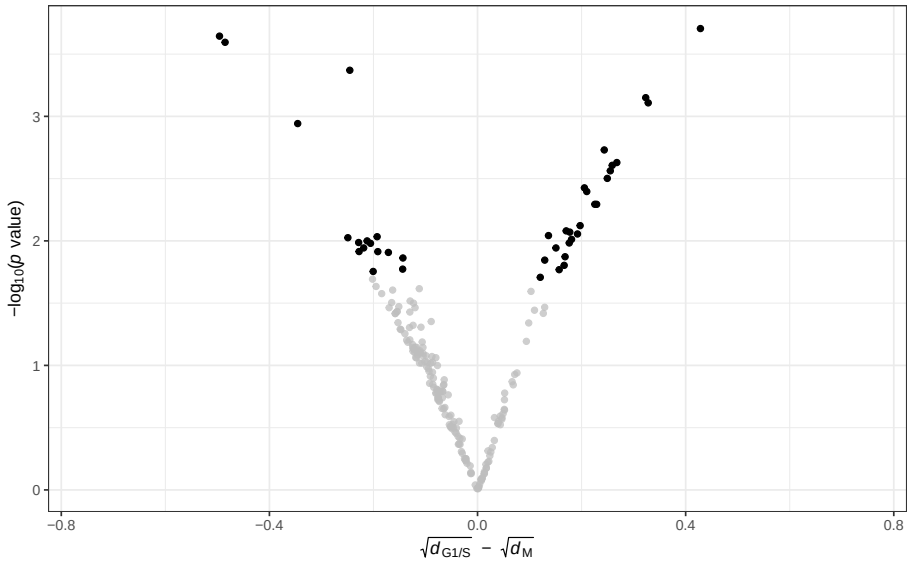
```
ppis_to_test_diff <- unique(
    c(filter(G1S_PPI_TPCA@tpcaResultTable, p_adj < 0.2)$complex_name,
      filter(M_PPI_TPCA@tpcaResultTable, p_adj < 0.2)$complex_name)
)

filtered_string_ppis <- string_ppi_975_df %>%
    filter(pair %in% ppis_to_test_diff)
```

Based on these PPIs we can now again run a differential TPCA:

```
M_vs_G1S_PPI_diff_TPCA <- runDiffTPCA(
    objList = list(g1s_mat),
    contrastList = list(m_mat),
    ctrlCondName = "G1/S",
    contrastCondName = "M",
    ppiAnno = filtered_string_ppis,
```

```
    n = 10^6
)
## Checking input arguments.
## Creating distance matrices.
## Comparing annotated protein-pairs across conditions.
## Comparing random protein-pairs across conditions.
## Generating result table.
```

```
plotDiffTpcaVolcano(M_vs_G1S_PPI_diff_TPCA,
                    setXLim = TRUE,
                    xlimit = c(-0.75, 0.75))
```

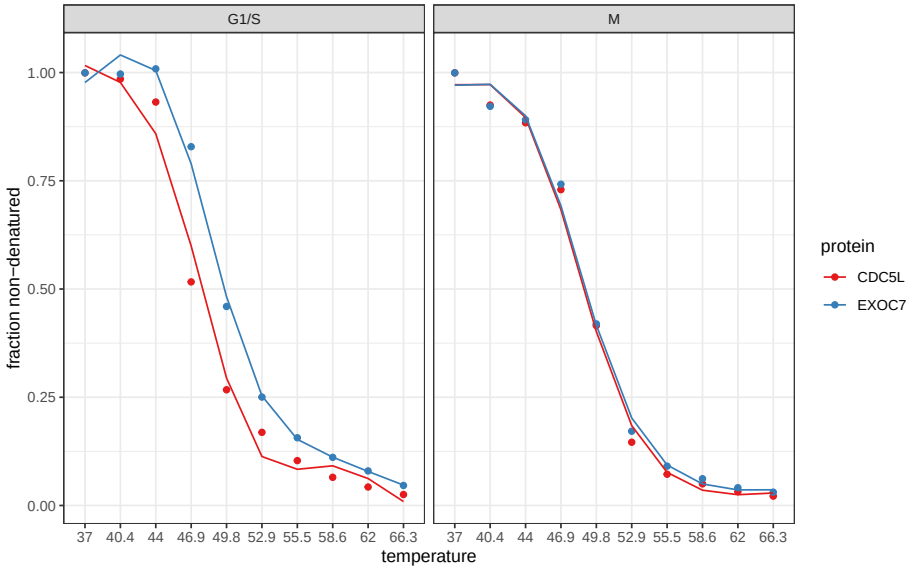


Again, we can now inspect the significant results:

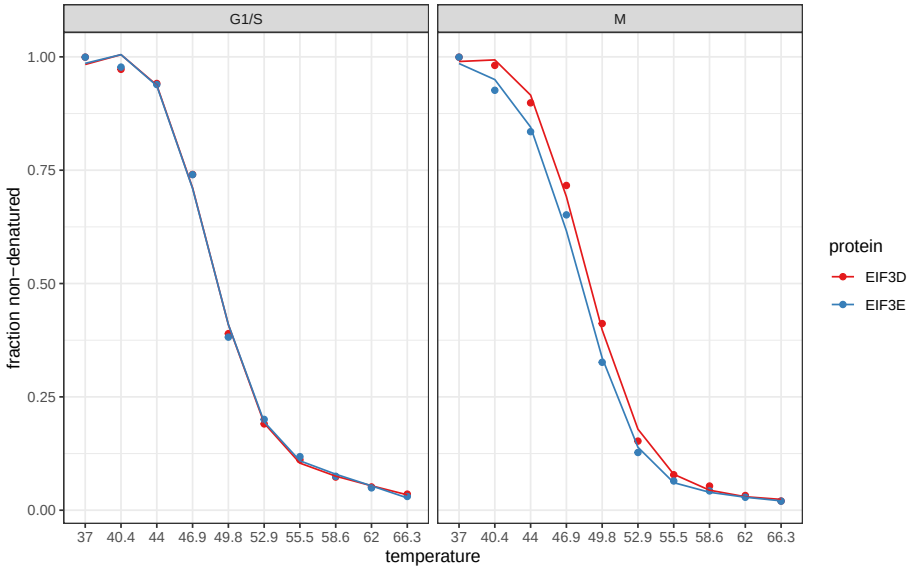
```
M_vs_G1S_PPI_diff_TPCA@diffTpcaResultTable %>%
    dplyr::select(pair, rssC1_rssC2, p_value, p_adj) %>%
    arrange(p_value)
## # A tibble: 207 x 4
##    pair         rssC1_rssC2  p_value  p_adj
##    <chr>              <dbl>    <dbl>  <dbl>
##  1 CDC5L:EXOC7       0.152  0.000197 0.0175
##  2 RPS6:RPSA        -0.279  0.000227 0.0175
##  3 RPS23:RPSA       -0.263  0.000254 0.0175
##  4 EIF3D:EIF3E      -0.0192 0.000426 0.0221
##  5 RPL10A:RPS16      0.0666 0.000708 0.0269
##  6 CDC73:RNF20       0.0727 0.000779 0.0269
##  7 PPID:PTGES3      -0.0991 0.00114  0.0338
##  8 AIMP1:RARS        0.0279 0.00186  0.0482
##  9 RPS23:RPS4X       0.0443 0.00235  0.0512
## 10 PSMA6:PSMA7       0.0395 0.00247  0.0512
## # ... with 197 more rows
```

And again we plot some of the significantly differentially coaggregating protein pairs:

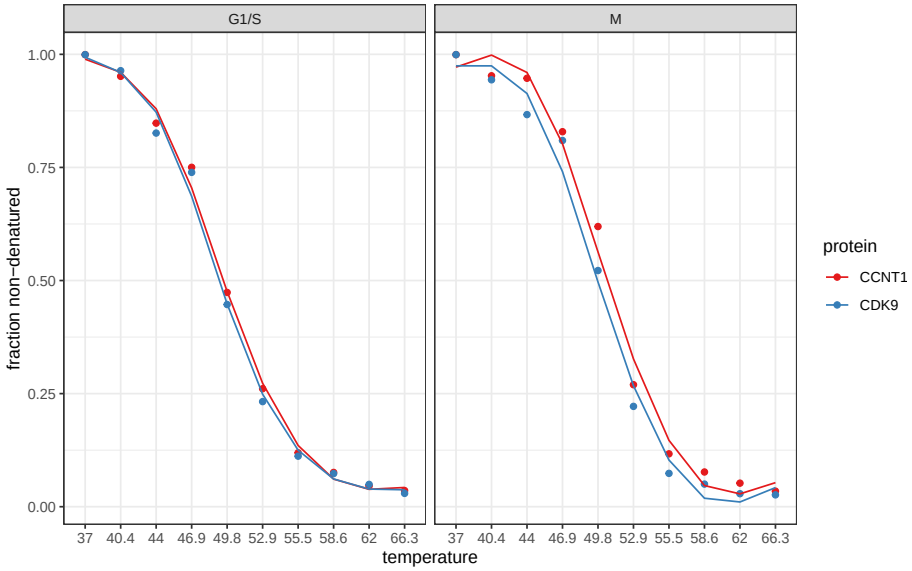
```
plotPPiProfiles(M_vs_G1S_PPI_diff_TPCA, c("CDC5L", "EXOC7"))
```




```
plotPPiProfiles(M_vs_G1S_PPI_diff_TPCA, c("EIF3D", "EIF3E"))
```




```
plotPPiProfiles(M_vs_G1S_PPI_diff_TPCA, c("CCNT1", "CDK9"))
```




```
sessionInfo()
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Mojave 10.14.6
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] ggplot2_3.2.1       Rtpca_0.0.99        tidyr_1.0.0
## [4] Biobase_2.44.0      BiocGenerics_0.30.0 readxl_1.3.1
## [7] dplyr_0.8.3         BiocStyle_2.12.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.2          RColorBrewer_1.1-2 plyr_1.8.4
##  [4] cellranger_1.1.0    pillar_1.4.2       compiler_3.6.1
##  [7] BiocManager_1.30.9 tools_3.6.1        zeallot_0.1.0
## [10] digest_0.6.22       evaluate_0.14      tibble_2.1.3
## [13] lifecycle_0.1.0     gtable_0.3.0       pkgconfig_2.0.3
## [16] rlang_0.4.1         cli_1.1.0          yaml_2.2.0
## [19] xfun_0.10           withr_2.1.2        stringr_1.4.0
## [22] knitr_1.25          pROC_1.15.3        vctrs_0.2.0
## [25] grid_3.6.1          tidyselect_0.2.5   glue_1.3.1
## [28] R6_2.4.0            fansi_0.4.0        fdrtool_1.2.15
## [31] rmarkdown_1.16      bookdown_0.14      farver_2.0.3
```

```
## [34] purrr_0.3.3        magrittr_1.5       ellipsis_0.3.0
## [37] splines_3.6.1      backports_1.1.5    scales_1.1.0
## [40] htmltools_0.4.0    assertthat_0.2.1   colorspace_1.4-1
## [43] labeling_0.3       utf8_1.1.4         stringi_1.4.3
## [46] lazyeval_0.2.2     munsell_0.5.0      crayon_1.3.4
```

References

Becher, I., Werner, T., Doce, C., Zaal, E.A., Tögel, I., Khan, C.A., Rueger, A., Muelbaier, M., Salzer, E., Berkers, C.R., et al. (2016). Thermal profiling reveals phenylalanine hydroxylase as an off-target of panobinostat. Nature Chemical Biology *12*, 908–910.

Becher, I., Andrés-Pons, A., Romanov, N., Stein, F., Schramm, M., Baudin, F., Helm, D., Kurzawa, N., Mateus, A., Mackmull, M.-T., et al. (2018). Pervasive Protein Thermal Stability Variation during the Cell Cycle. Cell 1–13.

Mateus, A., Kurzawa, N., Becher, I., Sridharan, S., Helm, D., Stein, F., Typas, A., and Savitski, M.M. (2020). Thermal proteome profiling for interrogating protein interactions. Molecular Systems Biology 1–11.

Molina, D.M., Jafari, R., Ignatushchenko, M., Seki, T., Larsson, E.A., Dan, C., Sreekumar, L., Cao, Y., and Nordlund, P. (2013). Monitoring Drug Target Engagement in Cells and Tissues Using the Cellular Thermal Shift Assay. Science *341*, 84–88.

Ori, A., Iskar, M., Buczak, K., Kastritis, P., Parca, L., Andrés-pons, A., Singer, S., Bork, P., and Beck, M. (2016). Spatiotemporal variation of mammalian protein complex stoichiometries. Genome Biology 1–15.

Savitski, M.M., Reinhard, F.B.M., Franken, H., Werner, T., Savitski, M.F., Eberhard, D., Martinez Molina, D., Jafari, R., Dovega, R.B., Klaeger, S., et al. (2014). Tracking cancer drugs in living cells by thermal profiling of the proteome. Science *346*, 1255784.

Szklarczyk, D., Gable, A.L., Lyon, D., Junge, A., Wyder, S., Huerta-Cepas, J., Simonovic, M., Doncheva, N.T., Morris, J.H., Bork, P., et al. (2019). STRING v11: Protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. Nucleic Acids Research *47*, D607–D613.

Tan, C.S.H., Go, K.D., Bisteau, X., Dai, L., Yong, C.H., Prabhu, N., Ozturk, M.B., Lim, Y.T., Sreekumar, L., Lengqvist, J., et al. (2018). Thermal proximity coaggregation for system-wide profiling of protein complex dynamics in cells. Science *0346*, 1–12.