SEC-SWATH-MS data processing and complex-centric analysis with CCprofiler

Isabell Bludau, Moritz Heusel & Ruedi Aebersold

2019-09-30

Setting-up CCprofiler and the working environment

```
library(devtools)
library(data.table)

install_github("CCprofiler/CCprofiler")
```

```
## Skipping install of 'CCprofiler' from a github remote, the SHA1 (29f56669) has not changed since las
##   Use `force = TRUE` to force installation
```

```
library('CCprofiler')
# Some specifications only relevant for creating the PDF protocol:
knitr::opts_knit$set(warning=FALSE, message=FALSE)
evaluateCode <- FALSE
plotPDF <- FALSE

Sys.time()
```

```
## [1] "2019-09-30 18:01:25 CEST"
```

Saving your workspace

You can save your progress any time during the analysis by using following command:

```
save.image(file='CCprofiler_analysis.RData')
```

This stores all current objects in your workspace. We recommend to save the workspace occasionaly during your analysis in order to ensure no analysis steps need to be repeated in case of an unexpected session interruption. If you whish to proceed your analysis at a later stage, you can use following command:

```
load(file='CCprofiler_analysis.RData')
```

Preparing your data for CCprofiler import

Quantitative peptide-level data

The main input for CCprofiler are quantitative peptide-level matrices derived from native complex fractionation coupled to mass spectrometry, here generated by SEC coupled to SWATH-MS.

A.Quantitative peptide-level data generated by OpenSWATH:

```
quantData_OpenSWATH <- fread("quantData_OpenSWATH.tsv")
```

```
## Warning in require_bit64(): Some columns are type 'integer64' but
## package bit64 is not installed. Those columns will print as strange
## looking floating point data. There is no need to reload the data. Simply
## install.packages('bit64') to obtain the integer64 print method and print
## the data again.
```

```
# # CCprofiler also has a reduced version of the full
# # input dataset stored internaly as a reference
# quantData_OpenSWATH <- exampleOpenSWATHinput
# head(quantData_OpenSWATH)
```

B. Quantitative peptide matrix generated by any software tool:

In long format:

```
quantData_long <- fread("examplePCPdataLong.tsv")
# CCprofiler also has a reduced version of the full
# input dataset stored internaly as a reference
# quantData_long <- examplePCPdataLong
head(quantData_long)
```

```
##    protein_id        peptide_id            filename intensity
## 1:     P11021       MKETAEAYLGK heuselm_J130730_011     12188
## 2:     P12956 EVAALC(UniMod:4)R heuselm_J130730_011      4310
## 3:     Q00839           FIEIAAR heuselm_J130730_011     50195
## 4:     O14980           FLVTVIK heuselm_J130730_011      3206
## 5:     Q14839           GNFLEIK heuselm_J130730_011     20002
## 6:     O00161    AHQITDESLESTRR heuselm_J130730_011      3894
```

In wide format:

```
quantData_wide <- fread("examplePCPdataWide.tsv")
# # CCprofiler also has a reduced version of the full
# # input dataset stored internaly as a reference
# quantData_wide <- examplePCPdataWide
head(quantData_wide[,1:5])
```

```
##      protein_id                   peptide_id heuselm_J130729_001
## 1: DECOY_O00161                   KNIQDLGEEI                   0
## 2: DECOY_O00161                    KQEDLMTIT                   0
## 3: DECOY_O00267             KTPTIPELHESSISVV                 200
## 4: DECOY_O00267 KVGPLLQQQTIDDSLEDSGGYVTEGVSS                   0
## 5: DECOY_O00267             RIVGTQGVVQTDLYTD                   0
## 6: DECOY_O00267             RPSMPAFGGVTFNTVD                   0
##    heuselm_J130729_003 heuselm_J130729_005
## 1:                   0                   0
## 2:                   0                   0
## 3:                   0                   0
## 4:                   0                   0
## 5:                1325                1586
## 6:                   0                   0
```

Fraction annotation table

CCprofiler requires a fraction annotation table that maps each MS run to a given chromatographic fraction number:

```
fractionAnnotation <- fread("exampleFractionAnnotation.tsv")
# # CCprofiler also has an example fraction annotation
# # stored internaly as a reference
# fractionAnnotation <- exampleFractionAnnotation
head(fractionAnnotation)
```

```
##               filename fraction_number
## 1: heuselm_J130729_001               1
## 2: heuselm_J130729_003               2
## 3: heuselm_J130729_005               3
## 4: heuselm_J130729_007               4
## 5: heuselm_J130729_009               5
## 6: heuselm_J130729_011               6
```

Molecular weight calibration table

For native complex separation via SEC, a molecular weight (MW) calibration table can be generated by measuring the apex fractions of an external standard set of reference proteins fractionated on the same SEC setup. By providing such a MW calibration table, CCprofiler can establish a transformation function based on the log-linear relationship between elution fractions and apparent MWs inherent to SEC, thus enabling the annotation of all sampled fractions with an apparent MW:

```
calibrationTable <- fread("exampleCalibrationTable.tsv")
# # CCprofiler also has an example calibration table
# # stored internaly as a reference
# calibrationTable <- exampleCalibrationTable
calibrationTable
```

```
##    std_weights_kDa std_elu_fractions
## 1:            1398              19.0
## 2:             699              29.0
## 3:             300              37.0
## 4:             150              46.0
## 5:              44              54.5
## 6:              17              61.0
```

Trace annotation table

A trace annotation table can be provided to CCprofiler, e.g. containing information from UniProt: https://www.uniprot.org/. It can be used to annotate proteins with according gene names and monomeric MWs. A trace annotation table including information about the monomeric MWs of the analyzed proteins is especially recommended for the analysis of SEC datasets and is required for the assessment of global proteome assembly states. CAUTION: The protein_id column in the quantitative matrix needs to match one of the column entries in the annotation table. In our case, the common entry are the UniProt identifiers.

```
uniprotAnnotation <- fread("exampleTraceAnnotation.tsv")
# # CCprofiler also has an example UniProt annotation
# # stored internaly as a reference
```

```
# uniprotAnnotation <- exampleTraceAnnotation
head(uniprotAnnotation, n=2)
```

```
##     Entry Entry_name   Status
## 1: P30450 1A26_HUMAN reviewed
## 2: P10314 1A32_HUMAN reviewed
##                                                                                Protein_names
## 1: HLA class I histocompatibility antigen, A-26 alpha chain (MHC class I antigen A*26)
## 2: HLA class I histocompatibility antigen, A-32 alpha chain (MHC class I antigen A*32)
##    Gene_names             Organism Length   Mass
## 1: HLA-A HLAA Homo sapiens (Human)    365 41,062
## 2: HLA-A HLAA Homo sapiens (Human)    365 41,048
```

Prior protein connectivity information

Complex-centric analysis requires prior protein connectivity information. There are two options:

A. Defined complex hypotheses, e.g. provided by CORUM

(Ruepp et al, 2009)

```
corumComplexes <- fread("corumComplexHypothesesRedundant.csv")
# # CCprofiler also has all CORUM complexes
# # stored internaly as a reference
# corumComplexes <- corumComplexHypothesesRedundant
head(corumComplexes)
```

```
##    complex_id                              complex_name protein_id
## 1:          1                      BCL6-HDAC4 complex     P41182
## 2:          1                      BCL6-HDAC4 complex     P56524
## 3:       1000                   TorsinA-TorsinB complex     O14656
## 4:       1000                   TorsinA-TorsinB complex     O14657
## 5:       1003 RC complex (Replication competent complex)     P09884
## 6:       1003 RC complex (Replication competent complex)     P20248
```

B. A binary protein-protein interaction network, e.g. provided by BioPlex (v1.0 (Huttlin et al, 2015),

http://bioplex.hms.harvard.edu)

```
BioPlexPPIs <- fread("BioPlexPPIs.tsv")
head(BioPlexPPIs)
```

```
##         a      b
## 1: P00813 A5A3E0
## 2: P00813 P60709
## 3: Q8N7W2 P14373
## 4: Q8N7W2 Q07021
## 5: Q8N7W2 O75096
## 6: Q6ZMN8 P08107
```

```
save.image(file='CCprofiler_analysis.RData')
Sys.time()
```

```
## [1] "2019-09-30 18:03:04 CEST"
```

Import traces

The traces object is the main data class used in the CCprofiler package. It stores the quantitative profiles ('traces') of peptide or protein intensities across the analyzed chromatographic fractions. Additionally, a traces object can store specific information about each of the peptides, proteins and chromatographic fractions.

Import from OpenSWATH

```
pepTraces <- importFromOpenSWATH(data = quantData_OpenSWATH,
                                 annotation_table = fractionAnnotation,
                                 verbose = FALSE)
```

Import from any other quantification tool

Long format

```
pepTraces_exampleSubset_long <- importPCPdata(input_data = quantData_long,
                                              fraction_annotation = fractionAnnotation,
                                              rm_decoys = FALSE)
```

Wide format

```
pepTraces_exampleSubset_wide <- importPCPdata(input_data = quantData_wide,
                                              fraction_annotation = fractionAnnotation,
                                              rm_decoys = FALSE)

save.image(file='CCprofiler_analysis.RData')
Sys.time()
```

Molecular weight calibration

Perform molecular weight calibration based on a provided calibration_table:

```
calibration = calibrateMW(calibration_table = calibrationTable,
                          PDF = plotPDF)
```

Annotate traces with the apparent molecular weight associated with each SEC fraction as extrapolated from the standard protein molecular weights and associated elution fraction numbers:

```
pepTraces <- annotateMolecularWeight(traces = pepTraces,
                                     calibration = calibration)
```

Annotate traces with trace annotation table UniProt

```
pepTraces <- annotateTraces(traces = pepTraces,
                            trace_annotation = uniprotAnnotation,
```

```
                                traces_id_column = "protein_id",
                                trace_annotation_id_column = "Entry")
```

Detect and impute missing values

In most proteomics pipelines, zero intensity values indicate either that the signal is missing at random (no detection due to technical reasons such as interferences from other peptides) or missing not at random (no detection due to cellular concentrations below the detection limit). We suggest that a zero value is considered as missing at random in case a quantitative (non-zero) signal has been detected in both the previous and following fraction. The detected missing at random values are subsequently imputed by a spline fit across the fractionation dimension.

Convert zeros in missing value locations to NA:

```
pepTracesMV <- findMissingValues(traces = pepTraces,
                                 bound_left = 2,
                                 bound_right = 2,
                                 consider_borders = FALSE)
```

Impute NA values by fitting a spline:

```
pepTracesImp <- imputeMissingVals(traces = pepTracesMV,
                                  method = "spline")
```

Plot imputation summary:

```
plotImputationSummary(traces = pepTracesMV,
                      tracesImp = pepTracesImp,
                      max_n_traces = 5,
                      PDF = plotPDF)
```

Filter peptides by consecutive peptide detection

Peptides that have never been detected in more than N consecutive fractions, here N=2, are removed from the traces object. This effectively removes false positive peptide detections from the dataset.

```
pepTracesConsIds <- filterConsecutiveIdStretches(traces = pepTracesImp,
                                                 min_stretch_length = 3,
                                                 remove_empty = TRUE)
```

Select high-quality proteins based on their average sibling peptide correlation

Calculate the average sibling peptide correlation (SPC) for each peptide:

For each peptide, the average pairwise correlation with the quantitative traces of its sibling peptides, i.e. peptides derived from the same protein, is calculated.

```
pepTracesSibPepCorr <- calculateSibPepCorr(traces = pepTracesConsIds,
                                           PDF = plotPDF)
```

Filter by SPC

Peptides below a minimum average SPC cutoff are removed. The rational is that outlier peptides as well as proteins with very heterogeneous quantitative peptide traces are excluded from further analysis. The filtering cutoff can either be automatically determined by a target-decoy based FDR estimation approach (A), or a fixed cutoff can be applied (B):

A. SPC based FDR cutoff:

```
# First estimate the FFT:
# The combined human assay library contains 10316 proteins.
cummulativeProteins <- length(unique(pepTracesSibPepCorr$trace_annotation$protein_id[grep("DECOY",pepTra
estimatedFFT <- (10316-cummulativeProteins)/10316
# Filter by FDR cutoff using the estimated FFT:
pepTraces_filtered_FDR <- filterBySibPepCorr(traces = pepTracesSibPepCorr,
                                             fdr_cutoff = 0.01,
                                             FFT = estimatedFFT,
                                             rm_decoys = TRUE,
                                             PDF = plotPDF)
```

B. Absolut sibling peptide correlation cutoff:

```
pepTraces_filtered_absoluteCutoff <- filterBySibPepCorr(traces = pepTracesSibPepCorr,
                                                        fdr_cutoff = NULL,
                                                        absolute_spcCutoff = 0.25,
                                                        rm_decoys = TRUE,
                                                        PDF = plotPDF)
```

Data inspection and quality control

Summary statistics

```
summary(pepTraces_filtered_FDR)
```

```
## $metrics
##  No. of Traces No. of Targets  No. of Decoys       % Decoys
##          61958          61958              0              0
##
## $type
## [1] "peptide"
##
## $annotations
##  [1] "protein_id"    "id"            "Entry_name"    "Status"
##  [5] "Protein_names" "Gene_names"    "Organism"      "Length"
##  [9] "Mass"          "protein_mw"    "SibPepCorr"    "DECOY"
##
## $fraction_count
## [1] 81
##
```

```
## $SibPepCorr_summary
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.03194  0.78267  0.88941  0.83045  0.94364  0.99928
```

Plotting of example traces

Exemplary visualization of the Proteasome subunit alpha type-1 (UniProt ID = P25786)

```
test_protein <- c("P25786")

test_peptide_traces <- subset(traces = pepTraces_filtered_FDR,
                              trace_subset_ids = test_protein,
                              trace_subset_type = "protein_id")

plot(test_peptide_traces,
     PDF = plotPDF,
     name = paste0("pepTraces_",test_protein))

save.image(file='CCprofiler_analysis.RData')
Sys.time()
```

Protein quantification

Protein quantification is performed by selecting the top N peptides based on their global intensity across all fractions. Here, we select the two highest peptides with the highest intensity for quantification.

```
protTraces <- proteinQuantification(traces = pepTraces_filtered_FDR,
                                    topN = 2,
                                    keep_less = FALSE)
```

Summary statistics

```
summary(protTraces)
```

Plotting of example traces

Exemplary visualization of the Proteasome subunit alpha type-1 (UniProt ID = P25786)

```
test_protein_traces <- subset(traces = protTraces,
                              trace_subset_ids = test_protein,
                              trace_subset_type = "protein_id")

plot(test_protein_traces,
     colour_by = "Entry_name",
     PDF = plotPDF,
     name = paste0("protTraces_",test_protein))
```

Overall workflow QC to evaluate the global proteome assembly state

The protein-level profiles can then be used to estimate the overall complex assembly state observed in the sample as a quality control to ensure the successful extraction and profiling of largely intact complexes. Here, we evaluate the total MS signal in assembled vs. monomeric range.

```
summarizeMassDistribution(protTraces,
                          PDF = plotPDF)

save.image(file='CCprofiler_analysis.RData')
Sys.time()
```

Optimize data analysis based on protein-level parameter grid search

A grid search can be performed to determine an optimal set of parameters for the protein- and/or complex-centric proteome profiling workflow. This optimal parameter set depends mostly on the co-fractionation characteristics and MS setup.

Randomly select a subset of proteins for the grid search

The selected subset of proteins should be representative of the proteome, thereby providing a trade-off between coverage and computational run-time. From our experience, selecting < 100 proteins suffers in regard to robustness, while >500 proteins will require a lot of processing time. We therefore propose a random selection of ~500 proteins.

```
all_proteins <- unique(pepTraces_filtered_absoluteCutoff$trace_annotation$protein_id)
testProtein_idx <- sample(1:length(all_proteins), 500)
testProteins = all_proteins[testProtein_idx]
peptideTracesSubset = subset(traces = pepTraces_filtered_FDR,
                             trace_subset_ids = testProteins,
                             trace_subset_type = "protein_id")
```

Perform parameter grid search

The grid search performs a peptide co-elution peak group finding for a selected combination of parameters with the goal to determine a good parameter set for the following analyses. Please note that the selection of suitable parameters is for the grid seach is critical.

```
gridFeatures <- performProteinGridSearch(traces = peptideTracesSubset,
                                         corrs = c(0.8,0.9,0.95),
                                         windows = c(8,10),
                                         smoothing = c(7,9),
                                         rt_heights = c(1,3),
                                         n_cores = 3)
```

Score protein features across all grid search parameters and select the best parameter set

```
gridFeatures_scored <- lapply(gridFeatures,
                              calculateCoelutionScore)
```

```
gridFeatures_qvalues <- lapply(gridFeatures_scored,
                               calculateQvalue,
                               plot = FALSE)

gridFeatures_stats <- qvaluePositivesPlotGrid(featuresGrid = gridFeatures_qvalues,
                                              colour_parameter = "corr",
                                              PDF = plotPDF)

bestParameters <- getBestQvalueParameters(stats = gridFeatures_stats,
                                          FDR_cutoff = 0.05)

bestParameters

write.table(bestParameters,
            "bestParameters.tsv",
            sep = "\t",
            quote = FALSE,
            row.names = FALSE)

save.image(file='CCprofiler_analysis.RData')
Sys.time()
```

Protein-centric analysis

Protein-centric analysis detects peptide co-elution peak groups along the chromatographic dimension. Each detected peak ('protein feature') represents the protein in a specific assembly state, i.e. monomeric or bound to different protein complexes.

Perform protein feature finding

```
proteinFeatures <- findProteinFeatures(traces = pepTraces_filtered_FDR,
                                       corr_cutoff = bestParameters$corr,
                                       window_size = bestParameters$window,
                                       rt_height = bestParameters$rt_height,
                                       smoothing_length = bestParameters$smoothing_length,
                                       collapse_method = "apex_only",
                                       perturb_cutoff = "5%",
                                       parallelized = TRUE,
                                       n_cores = 3,
                                       useRandomDecoyModel = TRUE)
```

Score detected protein features and estimate FDR

```
proteinFeatures_scored <- scoreFeatures(features = proteinFeatures,
                                        FDR = 0.05,
                                        PDF = plotPDF)

write.table(proteinFeatures_scored,
```

```
            "proteinFeatures_scored.tsv",
            sep = "\t",
            quote = FALSE,
            row.names = FALSE)
```

Inspect summary statistics on resulting protein features

The resulting figures provide information about the number of unique assembly states detected for all the proteins as well as about the number of proteins with at least one assembled protein signal (MW >= 2x monomeric MW in SEC).

```
summarizeFeatures(feature_table = proteinFeatures_scored,
                  PDF = plotPDF,
                  name = "proteinFeatures_summary")
```

Visualize and inspect protein features

```
plotFeatures(feature_table = proteinFeatures_scored,
             traces = pepTraces_filtered_FDR,
             calibration = calibration,
             feature_id = test_protein,
             annotation_label = "Entry_name",
             onlyBest = FALSE,
             peak_area = TRUE,
             monomer_MW = TRUE,
             PDF = plotPDF,
             name = paste0("proteinFeatures_",test_protein))
```

Plot all detected proteins

```
allDetectedProteins <- unique(proteinFeatures_scored$protein_id)
pdf("allDetectedProteins.pdf", height = 6, width = 8)
for (protein in allDetectedProteins) {
  plotFeatures(feature_table = proteinFeatures_scored,
               traces = pepTraces_filtered_FDR,
               calibration = calibration,
               feature_id = protein,
               annotation_label = "Entry_name",
               onlyBest = FALSE,
               peak_area = TRUE,
               monomer_MW = TRUE,
               PDF = FALSE)
}
dev.off()

save.image(file='CCprofiler_analysis.RData')
Sys.time()
```

Complex-centric analysis

Complex feature finding represents the central step of complex-centric analysis using *CCprofiler*. Based on prior protein interaction data and quantitative fractionation profiles, CCprofiler detects groups or subgroups of locally co-eluting proteins, indicating the presence of protein-protein complexes in the biological sample. Target complex queries are supplemented with decoy complex queries to support error control of the reported results. The result is a table summarizing the presence and composition of protein-protein complexes in the biological sample analyzed.

Complex query generation

In general the complex feature detection is very similar to the protein feature detection with the difference that complex hypotheses have to be generated at the beginning. This includes both target and decoy complex hypotheses for error estimation.

Prepare target complex queries

There are two options for protein complex target generation in CCprofiler: (A) use defined protein complex models for direct use as queries (2 or more subunits, e.g. from CORUM) or (B) use a protein-protein interaction network from which target complex queries can be extracted.

A) Inspect the coverage of pre-defined protein complex queries

from the previously loaded CORUM database

```
plotSummarizedMScoverage(hypotheses = corumComplexes,
                         protTraces = protTraces,
                         PDF = plotPDF,
                         name_suffix = "CORUM")
```

B) Generate and inspect protein complex queries from binary PPI

networks based on the previously loaded BioPlex network

```
pathLengthBioPlexPPIs <- calculatePathlength(BioPlexPPIs)
networkTargetsBioPlexPPIs <- generateComplexTargets(dist_info = pathLengthBioPlexPPIs,
                                                    max_distance = 1,
                                                    redundancy_cutoff = 0)

head(networkTargetsBioPlexPPIs)

plotSummarizedMScoverage(hypotheses = networkTargetsBioPlexPPIs,
                         protTraces = protTraces,
                         PDF = plotPDF,
                         name_suffix = "BioPlex")

save.image(file='CCprofiler_analysis.RData')
Sys.time()
```

Prepare decoy complex queries

Decoy complex queries are generated based on the target complex query set and its underlying network structure. The minimum distance specifies the minimal number of edges between any two proteins within any generated decoy complex query. It is important that the interaction network based on the targets is large enough to generate a random decoy set that does not overlap with the target complex queries. We recommend complex query sets of at least 1000 targets for the decoy based approach.

```
binaryCorumComplexes <- generateBinaryNetwork(corumComplexes)

pathLengthCorumComplexes <- calculatePathlength(binaryCorumComplexes)

corumComplexesPlusDecoys <- generateComplexDecoys(target_hypotheses = corumComplexes,
                                                  dist_info = pathLengthCorumComplexes,
                                                  min_distance = 2,
                                                  append = TRUE)

save.image(file='CCprofiler_analysis.RData')
Sys.time()
```

Complex-centric detection of protein co-elution features

Protein complex features are determined similar to the protein features described above. First, a sliding window strategy is applied, where all proteins of a protein complex hypothesis are tested for local profile correlation. If a subset of the proteins within a protein complex hypothesis correlate better then the specified cutoff, a protein complex feature is initiated, followed by peak detection within the regions of high correlation.

Perform complex feature finding

```
complexFeatures <- findComplexFeatures(traces = protTraces,
                                       complex_hypothesis = corumComplexesPlusDecoys,
                                       corr_cutoff = bestParameters$corr,
                                       window_size = bestParameters$window,
                                       rt_height = bestParameters$rt_height,
                                       smoothing_length = bestParameters$smoothing_length,
                                       collapse_method = "apex_network",
                                       perturb_cutoff = "5%",
                                       parallelized = TRUE,
                                       n_cores = 3)
```

Filter complex features according to their apparent molecular weight

Detected protein complex features that elute at an apparent molecular weight lower than any of the monomeric molecular weights of its subunits are removed

```
complexFeaturesFilteredMW <- filterFeatures(feature_table = complexFeatures,
                                            min_monomer_distance_factor = 2)
```

Select best complex feature for scoring

Select only the best complex feature, i.e. the complex signal with most subunits and highest correlation. This step is necessary prior to the statistical scoring, because individual elution peaks are not independent.

```
complexFeaturesBest <- getBestFeatures(feature_table = complexFeaturesFilteredMW)

complexFeaturesBest_scored <- scoreFeatures(features = complexFeaturesBest,
                                            FDR = 0.05,
                                            PDF = plotPDF,
                                            name = "complex_qvalueStats")

summarizeFeatures(complexFeaturesBest_scored,
                  PDF = plotPDF,
                  name = "complexFeaturesBest_summary")
```

Append secondary features

Once the best complex feature per query was scored, secondary signals can be appended to the result table based on a user defined correlation cutoff.

```
complexFeaturesAll <- appendSecondaryComplexFeatures(
                        scoredPrimaryFeatures = complexFeaturesBest_scored,
                        allFeatures = complexFeaturesFilteredMW,
                        peakCorr_cutoff = 0.5)

write.table(complexFeaturesAll,
            "complexFeaturesAll.tsv",
            sep = "\t",
            quote = FALSE,
            row.names = FALSE)
```

Inspect summary statistics on resulting protein features

```
summarizeFeatures(complexFeaturesAll,
                  PDF = plotPDF,
                  name = "complexFeaturesAll_feature_summary")
```

Visualize and inspect detected complex features

```
plotSummarizedComplexes(complexFeatures = complexFeaturesAll,
                        hypotheses = corumComplexes,
                        protTraces = protTraces,
                        PDF = plotPDF)
```

Plot example complex

```
testComplex <- unique(complexFeaturesAll$complex_id)[1]
plotFeatures(feature_table = complexFeaturesAll,
```

```
              traces = protTraces,
              calibration = calibration,
              feature_id = testComplex,
              annotation_label = "Entry_name",
              onlyBest = FALSE,
              peak_area = TRUE,
              monomer_MW = TRUE,
              PDF = plotPDF,
              name = paste("complexFeatures_",testComplex))
```

Plot all detected complexes

```
allDetectedComplexes <- unique(complexFeaturesAll$complex_id)
pdf("allDetectedComplexes.pdf", height = 6, width = 8)
for (complex in allDetectedComplexes) {
  plotFeatures(feature_table = complexFeaturesAll,
               traces = protTraces,
               calibration = calibration,
               feature_id = complex,
               annotation_label = "Entry_name",
               onlyBest = FALSE,
               peak_area = TRUE,
               monomer_MW = TRUE,
               PDF = FALSE)
}
dev.off()

save.image(file='CCprofiler_analysis.RData')
Sys.time()
```

Protein complex feature collapsing

Collapse overlapping and redundant co-elution evidence to delineate complexes and complex families with defined co-elution of subunits

```
complexFeaturesUnique <- getUniqueFeatureGroups(
                            feature_table = complexFeaturesBest_scored,
                            rt_height = 0,
                            distance_cutoff = 1.25)

complexFeaturesCollapsed <- callapseByUniqueFeatureGroups(
                              feature_table = complexFeaturesUnique,
                              rm_decoys = TRUE)

write.table(complexFeaturesCollapsed,
            "complexFeaturesCollapsed.tsv",
            sep = "\t",
            quote = FALSE,
            row.names = FALSE)
```

Plot all collapsed complexes

```
allCollapsedComplexes <- unique(complexFeaturesCollapsed$complex_id)
pdf("allCollapsedComplexes.pdf", height = 6, width = 8)
for (complex in allCollapsedComplexes) {
  plotFeatures(feature_table = complexFeaturesCollapsed,
               traces = protTraces,
               calibration = calibration,
               feature_id = complex,
               annotation_label = "Entry_name",
               onlyBest = FALSE,
               peak_area = TRUE,
               monomer_MW = TRUE,
               PDF = FALSE)
}
dev.off()

save.image(file='CCprofiler_analysis.RData')
Sys.time()
```