isobar package for iTRAQ and TMT protein quantification

Florian P. Breitwieser, Jacques Colinge

October 30, 2024

Contents

1 Introduction

The `isobar` package is an extensible and interactive environment for the data analysis and exploration of iTRAQ and TMT data. `isobar` implements the theory presented in Breitwieser et al., Journal of Proteome Research 2011. Further extensions for the analysis of PTM data are presented in Breitwieser and Colinge, Journal of Proteomics 2013.

1.1 Installation

`isobar` is part of Bioconductor. To install the latest stable version available for your version of R, start R and enter:

```
> if (!requireNamespace("BiocManager", quietly=TRUE))
+     install.packages("BiocManager")
> BiocManager::install("isobar")
```

Bioconductor has semi-annual releases, and the releases are tied to specific to R versions. The latest development version of `isobar` are always available at GitHub (`https://github.com/fbreitwieser/isobar`). Use the `devtools` package to install it from the source:

```
> library(devtools)
> install_github("fbreitwieser/isobar")
```

1.2 Loading package

The first thing you need to do is load the package.

```
>   library(isobar) ## load the isobar package
```

Check the installed version of isobar:

```
> packageVersion("isobar")
```

or equivalently for older versions of R:

```
> packageDescription("isobar")$Version
```

2 Loading data

`isobar` can read identifications and quantifications from tab-separated and MGF files. Perl scripts are supplied to generate a tab-separated version from the vendor formats of Mascot and Phenyx, see appendix C. The "ID.CSV" format is described in appendix A. Rockerbox and MSGF+ CSV file formats can be read, too, and are recognized by their file extension (`peptides.csv` and `msgfp.csv`). Experimental support for the mzIdentML format is also available - please contact the maintainer in case of problems.

ID.CSV tab-separated file containing peptide-spectra matches and spectrum meta-information such as retention time, m/z and charge. Generated by parser scripts.

MGF contains peak lists from which quantitative information on reporter tags are extracted. Must be centroided.

IBSPECTRA.CSV tab-separated file containing the same columns as ID.CSV plus *quantitative information* extracted from MGF file - that means the reporter tag masses and intensities as additional columns.

`readIBSpectra` is the primary function to generate a `IBSpectra` object. The first argument is one of iTRAQ4plexSpectra, iTRAQ8plexSpectra, TMT2plexSpectra, TMT6plexSpectra and TMT10plexSpectra denotes the tag type and therefore class.

```
>   ## generating IBSpectra object from ID.CSV and MGF
>   ib <- readIBSpectra("iTRAQ4plexSpectra",list.files(pattern=".id.csv"),
+           list.files(pattern=".mgf"))
>   ## write in tabular IBSPECTRA.CSV format to file
>   write.table(as.data.frame(ib),sep="\t",row.names=F,
+           file="myexperiment.ibspectra.csv")
>   ## generate from saved IBSPECTRA.CSV - MGF does not have to be supplied
>   ib.2 <- readIBSpectra("iTRAQ4plexSpectra","myexperiment.ibspectra.csv")
```

In case the MGF file is very big, it can be advanteguous to generate a smaller version containing only meta- and quantitative information before import in `R`. On Linux, the tool `grep` is readily available.

```
egrep '^[A-Z]|^1[12][0-9]\.' BIG.mgf > SMALL.mgf
```

2.1 `ibspiked` test samples

The examples presented are based on the dataset `ibspiked_set1` which has been designed to test `isobar`'s functionality and searched against the Swissprot human database with Mascot and Phenyx. `ibspiked_set1` is an iTRAQ 4-plex data set comprised of a complex background (albumin- and IgG-depleted human plasma) and spiked proteins. MS analysis was performed in ThermoFisher Scientific LTQ Orbitrap HCD instrument with 2D shotgun peptide separation (see original paper for more details). The samples used for each iTRAQ channel are as follows:

- Depleted human plasma background (>150 protein detected);
- Spiked-in proteins with the following ratios
  - `CERU_HUMAN (P00450)` at concentrations $1:1:1:1$;
  - `CERU_RAT (P13635)` at concentrations $1:2:5:10$;
  - `CERU_MOUSE (Q61147)` at concentrations $10:5:2:1$.

A second data set with ratios 1:10:50:100 is available as `ibspiked_set2` from `http://bininformatics.cemm.oeaw.ac.at/isobar`.

The Ceruplasmins have been selected as the share peptides. Hereafter, we load the data pacakage and the ceru protein IDs are identified via the `protein.g` function, which provides a mean to retrieve data from `ProteinGroup` objects. `ProteinGroup` is a slot of `IBSpectra` objects and contains informations on proteins and their grouping. See 2.2.

```
>   data(ibspiked_set1)
>   ceru.human <- protein.g(proteinGroup(ibspiked_set1),"CERU_HUMAN")
>   ceru.rat <- protein.g(proteinGroup(ibspiked_set1),"CERU_RAT")
>   ceru.mouse <- protein.g(proteinGroup(ibspiked_set1),"CERU_MOUSE")
>   ceru.proteins <- c(ceru.human,ceru.rat,ceru.mouse)
```

2.2 Protein information and grouping in `ProteinGroup`

When an `ibspectra.csv` is read, protein are grouped to identify proteins which have unique peptides. By default, only peptides with unique peptides are grouped.

The algorithm to infer protein groups works as follows:

1. Group proteins together which have been seen with exactly the same peptides (`indistinguishableProteins`) - these are the `protein.g` identifiers.
2. Create protein groups (`proteinGroupTable`):
   (a) Define proteins with specific peptides as reporters (`reporterProteins`)
   (b) Get proteins which are contained [1] by reporterProteins and group them below.
3. Create protein groups for proteins without specific peptides as above.

2.3 Loading data from CID/HCD (or CID/MS3, etc) experiments

A combined CID/HCD approach, in which for each precursor two fragmentation spectra are acquired, has proven useful to increase the number of identified and quantified peptide-spectrum matches. Usually, the reporter intensity information is taken from the HCD spectrum, and the peptide is identified based on the fragment ions in the CID spectrum.

To import these experiments, a comma-separated *mapping file* is needed, which contains the association from the `identification` to the `quantification` spectrum title.

Example mapping file (`mapping.csv`):

```
"hcd","cid"
"spectrum 1","spectrum 2"
"spectrum 3","spectrum 4"
```

By calling `readIBSpectra` with `mapping.file="mapping.csv"`, the spectra titles in the `id.file` are matched to those in the `peaklist.file`. If the column names for the quantification and identification spectrum are not `hcd` and `cid`, resp., they can be set with the argument `mapping=c(identification.spectrum="column name 1",quantification.spectrum="column name 2")`.

Example mapping file `mapping2.csv`:

```
"quant-spectrum-ms3","id-spectrum-ms2"
"spectrum 1","spectrum 2"
"spectrum 3","spectrum 4"
```

```
> readIBSpectra(...,
+               mapping.file="mapping2.csv",
+               mapping=c(identification.spectrum="id-spectrum-ms2",
+                         quantification.spectrum="quant-spectrum-ms3")
+               )
```

The argument `mapping.file` can take multiple files as argument (which are read and concatenated), or a `data.frame`.

[1] That means these proteins have a subset of the peptides of the reporter

2.4 Loading data from CID/HCD experiments with full HCD spectrum

If a full HCD spectrum was acquired, and both the HCD and CID spectrum are searched against a protein database, the argument `id.file.domap` to `readIBSpectra` can be used to merge both CID and HCD identifications:

```
> readIBSpectra("TMT6plexSpectra",
+               id.file="cid.identifications.csv",
+               peaklist.file="hcd.peaklist.mgf",
+               id.file.domap="hcd.identifications.csv",
+               mapping.file="mapping.csv",
+               ...
+               )
```

Here, the CID (`cid.identifications.csv`) and HCD identifications (`hcd.identifications.csv`) are combined and mapped according to `mapping.csv`. Diverging identifications are discarded.

2.5 Integrating and thresholding precursor purity measures

Savitzki et al., 2011 describe a measure for the precursor purity of fragment spectra. An implementation of the algorithm for the `multiplierz` platform is provided at `http://sourceforge.net/apps/trac/multiplierz/wiki/Precursors`. The script creates `_precursors.txt` files for each `RAW` file. To integrate and threshold based on the `s2i` measure, use the following code:

```
> precursors <- read.delim("file1_precursors.txt",header=TRUE,sep="\t",stringsAsFactors=F
> idfile <- read.delim("file1.id.csv",header=TRUE,sep="\t",stringsAsFactors=FALSE)
> idfile <- merge(idfile,precursors,
+                 by.x=c("fixed_spectrum","scans.from"),
+                 by.y=c("fixed_spectrum","SCAN"),all.x=TRUE) # FIXME see /work/analysis/
```

2.6 `MSnbase` integration

NOTE: It has been reported that the conversion functions are not functional for current versions of the package. We will try to resolve this.

`MSnbase` by Laurent Gatto provides data manipulation and processing methods for MS-based proteomics data. It provides import, representation and analysis of raw MS data stored in `mzXML`, `mzML` and `mzData` using the `mzR` package and centroided and un-centroided `MGF` peak lists. It allows to use and preprocess raw data whereas `isobar` requires centroided peak lists. In the future, the `isobar` class `IBSpectra` might be based on or replaced by `MSnbase`'s class `MSnSet`. For now, methods for coercion are implemented:

```
>   as(ibspectra,"MSnSet")
>   as(msnset,"IBSpectra")
```

3 Data Analysis

3.1 Reporter mass precision

The distribution of observed masses from the reporter tags can be used to visualize the precision of the MS setup on the fragment level and used to set the correct window for isolation.

The expected masses of the reporter tags are in the slot `reporterTagMasses` of the implementations of the `IBSpectra` class. The experimental masses are in the matrix `mass` of `AssayData`; they can also be accessed by the method `reporterMasses(x)`.

3.2 MSnbase integration

NOTE: It has been reported that the conversion functions are not functional for current versions of the package. We will try to resolve this.

MSnbase by Laurent Gatto provides data manipulation and processing methods for MS-based proteomics data. It provides import, representation and analysis of raw MS data stored in mzXML, mzML and mzData using the mzR package and centroided and un-centroided MGF peak lists. It allows to use and preprocess raw data whereas isobar requires centroided peak lists. In the future, the isobar class IBSpectra might be based on or replaced by MSnbase's class MSnSet. For now, methods for coercion are implemented:

```
>   as(ibspectra,"MSnSet")
>   as(msnset,"IBSpectra")
```

4 Data Analysis

4.1 Reporter mass precision

The distribution of observed masses from the reporter tags can be used to visualize the precision of the MS setup on the fragment level and used to set the correct window for isolation.

The expected masses of the reporter tags are in the slot reporterTagMasses of the implementations of the IBSpectra class. The experimental masses are in the matrix mass of AssayData; they can also be accessed by the method reporterMasses(x).

```
> sprintf("%.4f",reporterTagMasses(ibspiked_set1))  ## expected masses

[1] "114.1112" "115.1083" "116.1116" "117.1150"

> mass <- assayData(ibspiked_set1)[["mass"]] ## observerd masses
> apply(mass,2,function(x) sprintf("%.4f",quantile(x,na.rm=TRUE,probs=c(0.025,0.975))))

     114        115        116        117
[1,] "114.1110" "115.1081" "116.1115" "117.1148"
[2,] "114.1116" "115.1087" "116.1120" "117.1153"
```

reporterMassPrecision provides a plot of the distribution.

```
>   print(reporterMassPrecision(ibspiked_set1))
```

4.2 Normalization and isotope impurity correction

Isotope impurity correction factors are supplied by labelling reagent manufacturers. Default values that can be modified by the user are available in isobar and corrections are obtained by simple linear algebra.

Due to differences between samples it is advisable to normalize data before further processing. By default, normalize corrects by a factor such that the median intensities in all reporter channels are equal.

See figure 2.

```
> ib.old <- ibspiked_set1
> ibspiked_set1 <- correctIsotopeImpurities(ibspiked_set1)
> ibspiked_set1 <- normalize(ibspiked_set1)
```

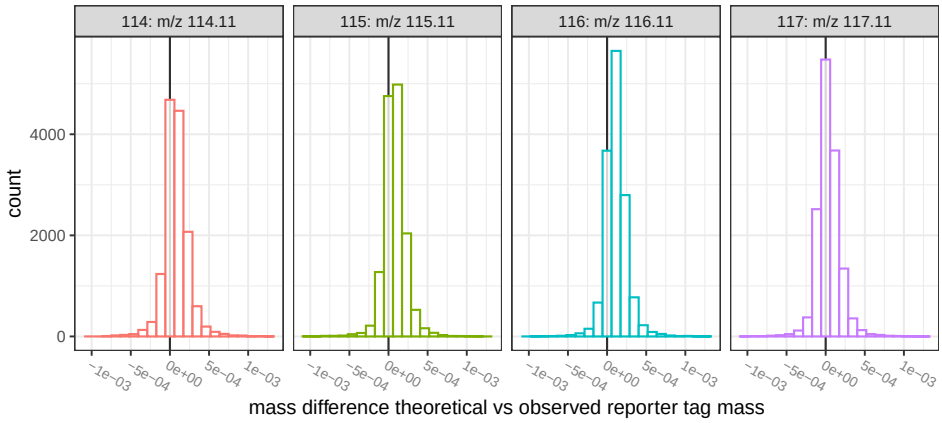

Figure 1: Reporter mass precision plot.

```
> par(mfrow=c(1,2))
> maplot(ib.old,channel1="114",channel2="117",ylim=c(0.5,2),
+        main="before normalization")
> abline(h=1,col="red",lwd=2)
> maplot(ibspiked_set1,channel1="114",channel2="117",ylim=c(0.5,2),
+        main="after normalization")
> abline(h=1,col="red",lwd=2)
```

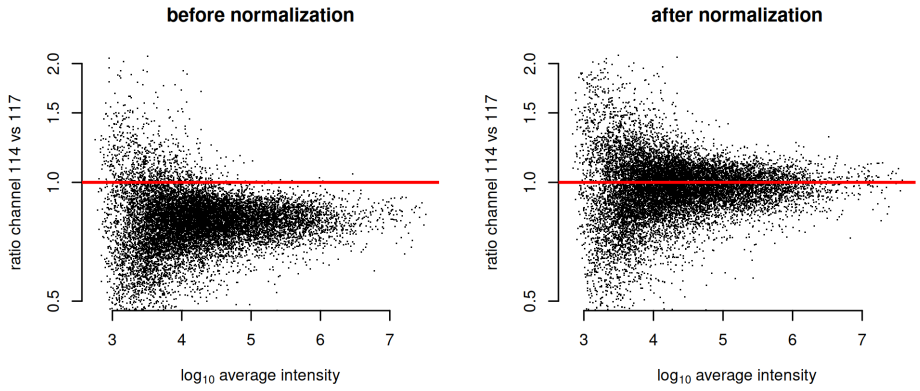

Figure 2: Ratio versus intensity plots ('MA plots') before and after applying normalization.

4.3 Fitting a noise model

A noise model is a approximation of the expected technical variation based on signal intensity. It is stable for a certain experimental setup and thus can be learned once. Noise is observed directly when comparing identical samples in multiple channels (1:1 iTRAQ/TMT sample) and we can use `ibspiked_set1` background proteins as a 1:1 sample. Therefore we exclude the ceruplasmins before fitting a noise model using `NoiseModel`. See figure 3.

```
> ib.background <- subsetIBSpectra(ibspiked_set1,protein=ceru.proteins,direction="exclude
> noise.model <- NoiseModel(ib.background)

[1] 0.03333677 9.57308831 1.35782023
```

Though only recommended when sufficient data are available, a method exist for the estimation of a noise model without a 1:1 dataset. It takes longer time as it first computes all the protein ratios to shift spectrum ratios to 1:1. To examplify this procedure, we only take rat and mouse CERU proteins from `ibspiked_set1`, see figure 3. The resultant noise model is a rough approximation only because of the very limitted data, see Breitwieser et al. Supporting Information, submitted, for a real example.

```
> ib.ceru <- subsetIBSpectra(ibspiked_set1,protein=ceru.proteins,
+                            direction="include",
+                            specificity="reporter-specific")
> nm.ceru <- NoiseModel(ib.ceru,one.to.one=FALSE,pool=TRUE)

3 proteins with more than 10 spectra, taking top 50.
[1] 0.0000000001 0.4177838500 0.1950037333

> maplot(ib.background,noise.model=c(noise.model,nm.ceru),
+        channel1="114",channel2="115",ylim=c(0.2,5),
+        main="95% CI noise model")
```

4.4 Protein and peptide ratio calculation

`estimateRatio` calculates the relative abundance of a peptide or protein in one tag compared to another. It calculates a weighted average (after outlier removal) of the spectrum ratios. The weights are the inverse of the spectrum ratio variances. It requires a `IBSpectra` and `NoiseModel` object and definitions of channel1, channel2, and the protein or peptide. The result is channel2/channel1.

```
> ## Calculate ratio based on all spectra of peptides specific
> ##  to CERU_HUMAN, CERU_RAT or CERU_MOUSE. Returns a named
> ##  numeric vector.
> 10^estimateRatio(ibspiked_set1,noise.model,
+                  channel1="114",channel2="115",
+                  protein=ceru.proteins)['lratio']

   lratio
0.9344642

> ## If argument 'combine=FALSE', estimateRatio returns a data.frame
> ##  with one row per protein
> 10^estimateRatio(ibspiked_set1,noise.model,
+                  channel1="114",channel2="115",
+                  protein=ceru.proteins,combine=FALSE)[,'lratio']

[1] 1.0492478 1.8665298 0.4956441

> ## spiked material channel 115 vs 114:
> ##                  CERU_HUMAN (P00450): 1:1
> ##                  CERU_RAT   (P13635): 2:1  = 2
> ##                  CERU_MOUSE (Q61147): 5:10 = 0.5
```

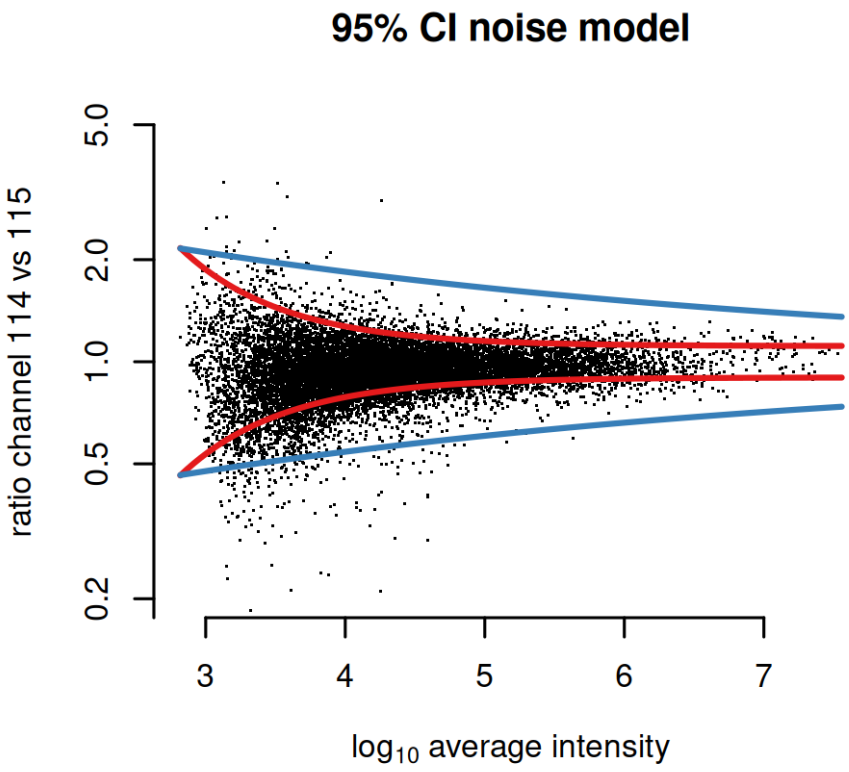

Figure 3: Red lines denote the 95 % confidence interval as estimated by the noise model on background proteins. The blue line is estimated as non 1:1 noise model based on only spectra of CERU proteins.

```
>
> ## Peptides shared between rat and mouse
> pep.shared <- peptides(proteinGroup(ibspiked_set1),
+                        c(ceru.rat,ceru.mouse),set="intersect",
+                        columns=c('peptide','n.shared.groups'))
> ## remove those which are shared with other proteins
> pep.shared <- pep.shared$peptide[pep.shared$n.shared.groups==2]
> ## calculate ratio: it is between the rat and mouse ratios
> 10^estimateRatio(ibspiked_set1,noise.model,
+                  channel1="114",channel2="115",
+                  peptide=pep.shared)['lratio']

   lratio
0.6314602
```

When examining the global differences and differences in between classes, `proteinRatios` can be used. It is also suitable to inspect sample variability. The argument `cl` can be used to define class labels. If `combn.method='interclass'` or `intraclass` and `summarize=TRUE`, `proteinRatios` return a single summarized ratio across and within classes, resp..

```
> protein.ratios <- proteinRatios(ibspiked_set1,noise.model,cl=c("1","0","0","0"))
> ## defined class 114 and 115 as class 'T', 116 and 117 as class 'C'
> classLabels(ibspiked_set1) <- c("T","T","C","C")
> proteinRatios(ibspiked_set1,noise.model,protein=ceru.proteins,
+               cl=classLabels(ibspiked_set1),combn.method="interclass",
+               summarize=T)[,c("ac","lratio","variance")]

      ac       lratio     variance
1 P00450  0.006986124 0.0006221153
2 P13635  0.583020063 0.0504077253
3 Q61147 -0.564037705 0.0471635852
```

4.5 Protein ratio distribution and selection

Protein ratio distributions can be calculated ideally on biological replicated. To examine differentially expressed proteins, both sample variability information (random protein ratios) as a *fold-change* constraint, and ratio *precision* can be used. For a experimental setup with biolgical replicates in the same experiment (but different channels), the distribution of biological variability can be learned by computing the ratios between the replicates. With no replicates available, one has the choice to (a) model the actual protein ratios and just select the most extreme ratios; (b) learn the distribution from a previous experiment; or (c) assume a standard Cauchy distribution with location 0 and scale 0.1, 0.05, and 0.025, which correspond with $\alpha = 0.05$ roughly to fold changes of 4, 2, and 1.5.

A Cauchy distribution fits accurately this type of random protein ratio distribution: Cauchy is displayed in red, Gaussian in blue. In the case of `ibspiked_set1`, the many 1:1 proteins provide us with adequate data to learn the random protein ratio distribution, however only of the *technical* variation.

```
> #protein.ratios <- proteinRatios(ibspiked_set1,noise.model)
> protein.ratiodistr.wn <- fitWeightedNorm(protein.ratios[,'lratio'],
+                                          weights=1/protein.ratios[,'variance'])
> protein.ratiodistr.cauchy <- fitCauchy(protein.ratios[,"lratio"])
```

```
> library(distr) # required library
> limits=seq(from=-0.5,to=0.5,by=0.001)
> curve.wn <- data.frame(x=limits,y=d(protein.ratiodistr.wn)(limits))
> curve.cauchy<-data.frame(x=limits,y=d(protein.ratiodistr.cauchy)(limits))
> g <- ggplot(data.frame(protein.ratios),aes(x=lratio)) +
+   geom_histogram(colour = "darkgreen", fill = "white",aes(y=..density..),
+                  binwidth=0.02) + geom_rug() +
+   geom_line(data=curve.wn,aes(x=x,y=y),colour="blue") +
+   geom_line(data=curve.cauchy,aes(x=x,y=y),colour="red")
> print(g)
```

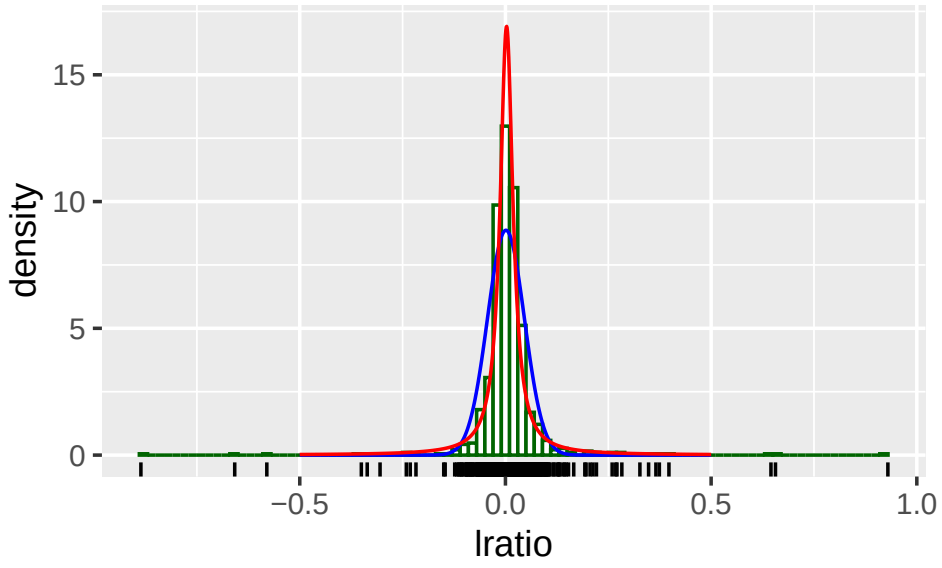

Figure 4: Histogram of all protein ratios in `ibspiked_set1`. A fit with a Gaussian and Cauchy probability density function is shown in blue and red, respectively.

Now, when supplying a `ratiodistr` parameter to `estimateRatio` and `proteinRatios`, sample and signal p-values are calculated, what we illustrate in the code below

```
> rat.list <-
+   estimateRatio(ibspiked_set1,noise.model=noise.model,channel1="114",channel2="115",
+                 protein=reporterProteins(proteinGroup(ibspiked_set1)),combine=F,
+                 ratiodistr=protein.ratiodistr.cauchy)
> rat.list[rat.list[,"is.significant"]==1,]

           lratio    variance   var_hat_vk n.spectra n.na1 n.na2  p.value.rat
P13635  0.2710349 0.006274397 8.461878e-05       242     0     0 3.535261e-04
Q61147 -0.3048301 0.001038084 8.877274e-07       139     0     0 6.689859e-22
       p.value.sample    p.value is.significant
P13635     0.02229517 0.02539988              1
Q61147     0.01945302 0.01967245              1
```

4.6 Detection of proteins with no specific peptides

It is well known that MS analysis only reveals the presence of so-called protein groups, defined as sets of proteins identified by the same set of peptides. The protein that contains all the peptides is the group reporter (there are possibly several group reporters) and if it has one specific peptide at least then its presence in the sample is certain. The status of the other proteins in the group is in general impossible to determine. When quantitative information is available, there is a potential to elucidate the structure of part of the protein groups.

In the example below, a subset `IBSpectra` object is created, containing only peptides shared between CERU_RAT and CERU_MOUSE, and those specific to CERU_RAT.

```
> ## peptides shared between CERU_RAT and CERU_MOUSE have been computed before
> pep.shared

 [1] "AGLQAFFQVR"      "DNEEFLESNK"      "DTANLFPHK"       "EMGPTYADPVCLSK"
 [5] "ETFTYEWTVPK"     "GSLLADGR"        "KGSLLADGR"       "LYHSHVDAPK"
 [9] "NMATRPYSLHAHGVK" "RDTANLFPHK"      "VFFEQGATR"

> ## peptides specific to CERU_RAT
> pep.rat <- peptides(proteinGroup(ibspiked_set1),protein=ceru.rat,
+                     specificity="reporter-specific")
> ## create an IBSpectra object with only CERU_RAT and shared peptides
> ib.subset <- subsetIBSpectra(ibspiked_set1,
+                              peptide=c(pep.rat,pep.shared),direction="include")
> ## calculate shared ratios
> sr <- shared.ratios(ib.subset,noise.model,
+                     channel1="114",channel2="117",
+                     ratiodistr=protein.ratiodistr.cauchy)
> sr

       reporter.protein protein2    ratio1 ratio1.var n.spectra.1     ratio2
lratio           P13635   Q61147 0.9296973  0.0132681         241 0.00909977
        ratio2.var n.spectra.2
lratio 0.001535065         273

>
```

```
>    ## plot significantly different protein groups where 90% CI does not overlap
>    ## CERU_MOUSE and CERU_RAT is detected, as expected.
>    shared.ratios.sign(sr,z.shared=1.282)

    reporter.protein protein2 n.spectra.1 n.spectra.2           proteins
1.1           P13635   Q61147         241         273 P13635 \nvs Q61147
1.2           P13635   Q61147         241         273 P13635 \nvs Q61147
           g      ratio         var n.spectra id
1.1 reporter 0.92969725 0.013268101      > 10  1
1.2   member 0.00909977 0.001535065      > 10  1
```

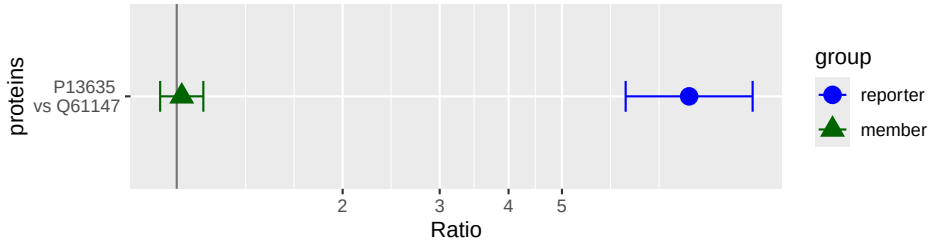

Figure 5: Peptides of spiked ceruplasmins have significantly different ratios between groups. Group *reporter* consists of peptides specific to `CERU_RAT (P13635)`, group *member* are peptides shared between `CERU_RAT` and `CERU_MOUSE (Q61147)`.

5 Report generation

`isobar` provides a rich interface for creating Excel and PDF reports for further analysis and quality control. The main entry function is `create.reports`. Alternatively the Rscript `create_reports.R` can be used. It is located in the `report` folder of the `isobar` installation, and reads the properties from a file in the working directory.

The posible values are defined in the `report/properties.R` file in the isobar installation. To generate a report with standard properties the following code should do the trick:

```
> create.reports(type="iTRAQ4plexSpectra",
+                identifications="my.id.csv",peaklist="my.mgf")
```

The properties can also be defined in a `properties.R` file which is located in the working directory. The properties are set in the following order:

- 'global' properties in `ISOBAR-DIRECTORY/report/properties.R`[2]
- 'local' properies in `WORKING-DIRECTORY/properties.R`
- command line arguments to `create_reports.R` or `create.reports` function

Appendix B provides a syntax-highlighted version of the properties file supplied with isobar, which sets the default parameters and provides some help in the comments. The number of paramters which can be set may seem a lot at first, however most times only a few are needed.

For successful completion, LaTeX- for the PDF reports - and Perl - for the Excel reports - need to be installed.

[2] located in `system.file('report','properties.R',package='isobar')`

5.1 Files used for report generation

```
> ## execute to find the path and file location in your installation.
> system.file("report",package="isobar") ## path
> list.files(system.file("report",package="isobar")) ## files
```

create_reports.R R script which can be used to create QC and PDF reports It initializes the environment, reads properties and calls `Sweave` on QC and DA Sweave files. Additionally it generates a Excel data analysis report by calling `tab2xls.pl`.

isobar-qc.Rnw Sweave file with quality control plots.

isobar-analysis.Rnw Sweave file for generating a data analysis report with the list of all protein ratios and list of significantly different proteins.

properties.R Default configuration for `create_reports.R`. It is parsed as R code.

report-utils.R Helper R functions used in Sweave documents.

report-utils.tex Helper LaTeX functions used in Sweave documents.

A File formats

A.1 ID CSV file format

The Perl parsers create ID CSV files - identification information for all matched spectra without quantitative information. You can create your own parser, the resulting file should be tab-delimited and contain the following columns. Only bold columns are obligatory. The information is redundant - that means if a peptide may stem from two different proteins the information of the identification is repeated.

accession	Protein AC
peptide	Peptide sequence
modif	Peptide modification string
charge	Charge state
theo.mass	Theoretical peptide mass
exp.mass	Experimentally observed mass
parent.intens	Parent intensity
retention.time	Retention time
spectrum	Spectrum identifier
search.engine	Protein search engine and score

A.2 IBSpectra CSV file format

IBSpectra file format has the same columns as the ID CSV format and additionally columns containing the quantitation information, namely `X`*tagname*`_mass` and `X`*tagname*`_ions`, for mass and intensity of each tag *tagname*. Below an example of the further columns for an `iTRAQ 4plex` IBSpectra.

X114_mass	reporter ion mass
X115_mass	reporter ion mass
X116_mass	reporter ion mass
X117_mass	reporter ion mass
X114_ions	reporter ion intensity
X115_ions	reporter ion intensity
X116_ions	reporter ion intensity
X117_ions	reporter ion intensity

B `properties.R` for report generation

```
##
## Isobar properties.R file
##     for automatic report generation
##
## It is standard R code and parsed using sys.source

#######################################################################
## General properties

## Report type: Either 'protein' or 'peptide'
# report.level="peptide"
report.level="protein"
#attr(report.level,"allowed.values") <- c("protein","peptide")

## Isobaric tagging type. Use one of the following:
# type='iTRAQ4plexSpectra'
# type='iTRAQ8plexSpectra'
# type='TMT2plexSpectra'
# type='TMT6plexSpectra'
type=NULL
#attr(type,"allowed.values") <- IBSpectraTypes()

isotope.impurities=NULL
correct.isotope.impurities=TRUE

## Name of project, by default the name of working directory
## Will be title and author of the analysis reports.
name=basename(getwd())
author=paste0("isobar R package v",packageDescription("isobar")$Version)

## specifes the IBSpectra file or object
##   - can be a data.frame (e.g. ibspectra=as.data.frame(ibspiked_set1) )
##   - if it is a character string, it is assumed to be a file
##     - if it ends on .rda, then it is assumed to be a R data object
##     - if it does not exists, then it is may generated based on
##       the peaklist and identifications properties
ibspectra=paste(name,"ibspectra.csv",sep=".")

## When replicates or 'samples belonging together' are analyzed, a
## ProteinGroup object based on all data should be constructed
## beforehand. This then acts as a template and a subset is used.
protein.group.template=NULL

## Via database or internet connection, informations on proteins (such
## as gene names and length) can be gathered. protein.info.f defines
## the function which takes a ProteinGroup object as argument
protein.info.f=getProteinInfoFromTheInternet

## Where should cached files be saved? Will be created if it does not
## exist
# cachedir="."
cachedir="cache"
## Regenerate cache files? By default, chache files are used.
```

```
regen=FALSE

## An ibspectra object can be generated from peaklists and
## identifications.

## peaklist files for quantitation, by default all mgf file in
## directory
peaklist=list.files(pattern="*\\.mgf$")
## id files, by default all id.csv files in directory
identifications=list.files(pattern="*\\.id.csv$")
## mapping files, for data quantified and identified with different but
## correspoding spectra. For example corresponding HCD-CID files.

## masses and intensities which are outside of the 'true' tag mass
## +/- fragment.precision/2 are discarded
fragment.precision=0.01
## filter mass outliers
fragment.outlier.prob=0.001

## Additional arguments of readIBSpectra can be set here
##  decode.titles should be set to TRUE for Mascot search results
##  as Mascot encodes the spectrum title (e.g. space -> %20)
readIBSpectra.args = list(
    mapping.file=NULL,
    decode.titles=FALSE
)

######################################################################
## Quantification properties

normalize=TRUE
# if defined, normalize.factors will be used for normalization
normalize.factors=NULL
normalize.channels=NULL
normalize.use.protein=NULL
normalize.exclude.protein=NULL
normalize.function=median
normalize.na.rm=FALSE

peptide.specificity=REPORTERSPECIFIC

use.na=FALSE

## the parameter noise.model can be either a NoiseModel object or a file name
data(noise.model.hcd)
noise.model=noise.model.hcd
## If it is a file name, a noise model is estimated as non one-to-one
## and saved into the file. otherwise, the noise model is loaded from
## the file
# noise.model="noise.model.rda"

## Define channels for creation of a noise model, ideally a set of
##  channels which are technical replicates.
noise.model.channels=NULL
```

```
## If noise.model.is.technicalreplicates is FALSE, the intensities
##   are normalized for protein means, creating artifical technical
##   replicates. For this procedure, only proteins with more than
##   noise.model.minspectra are considered.
noise.model.is.technicalreplicates=FALSE
noise.model.minspectra=50

## class labels. Must by of type character and of same length as
## number of channels I. e. 4 for iTRAQ 4plex, 6 for TMT 6plex Example
## for iTRAQ 4plex:
## Class definitions of the isobaric tag channels.
## A character vector with the same length as channels
##   (e.g. 4 for iTRAQ 4plex, 6 for TMT 6plex)
## Example for iTRAQ 4plex:
# class.labels=as.character(c(1,0,0,0))
# class.labels=c("Treatment","Treatment","Control","Control")
## Also names are possible - these serves as description in the report
##  and less space is used in the rows
# class.labels=c("Treatment"="T","Treatment"="T","Control"="C","Control"="C")
class.labels=NULL

## The following parameters define which ratios are calculated.

## summarize ratios with equal class labels, set to TRUE when replicates are used
summarize=FALSE

## combn.method defines which ratios are calculated - versus a channel or a class,
##   all the ratios within or across classes, or all possible combinatioins.
## When summarize=TRUE is set, use "interclass", "versus.class", or "intraclass"
# combn.method="global"
# combn.method="versus.class"
# combn.method="intraclass"
# combn.method="interclass"
combn.method="versus.channel"
vs.class=NULL

cmbn=NULL

## Arguments given to 'proteinRatios' function. See ?proteinRatios
ratios.opts = list(
    sign.level.sample=0.05,
    sign.level.rat=0.05,
    groupspecific.if.same.ac=TRUE)

quant.w.grouppeptides=c()

min.detect=NULL

preselected=c()

### Biological Variability Ratio Distribution options
## ratiodistr can be set to a file or a 'Distribution object. ' If
##  NULL, or the specified file is not existent, the biological
##  variability of ratios is estimated on the sample at hand and
##  written to cachedir/ratiodistr.rda or the specified file.
```

```
ratiodistr=NULL

## Ideally, when the biological variability is estimated for the
## sample at hand, a biological replicate is present (/ie/ same class
## defined in class labels).  Classes can also be assigned just for
## estimation of the ratio distribution, /eg/ to choose biologically
## very similar samples as pseudo replicates.
ratiodistr.class.labels=NULL

## Function for fitting. Available: fitCauchy, fitTlsd
ratiodistr.fitting.f=fitCauchy

## Use symetrical ratios - i.e. for every ratio r add a ratio -r
##   prior to fitting of a distribution
ratiodistr.symmetry=TRUE

## If defined, use z-score instead of ratio distribution
# zscore.threshold=2.5
zscore.threshold=NULL

######################################################################
## PTM properties

## PhosphoSitePlus dataset which can be used to annotate known
## modification sites. Download site:
## http://www.phosphosite.org/staticDownloads.do
phosphosite.dataset <- NULL

## Modification to track. Use 'PHOS' for phosphorylation.
# ptm <- c('ACET','METH','UBI','SUMO', 'PHOS')
ptm <- NULL

## file name of rda or data.frame with known modification sites
## gathered with ptm.info.f.  defaults to 'cachedir/ptm.info.rda'
ptm.info <- NULL

## Function to get PTM modification sites from public datasets
# ptm.info.f <- getPtmInfoFromNextprot
# ptm.info.f <- function(...)
#    getPtmInfoFromPhosphoSitePlus(...,modification="PHOS")
# ptm.info.f <- function(...)
#    getPtmInfoFromPhosphoSitePlus(...,modification=ptm)
ptm.info.f <- getPtmInfoFromNextprot

## A protein quantification data.frame (generated with
##  'proteinRatios').  The ratio and variance are used to correct the
##  observed modified peptide ratios Needs to have the experimental
##  setup as the modified peptide experiment
correct.peptide.ratios.with <- NULL
## Protein groups to use with correct.peptide.ratios()
correct.peptide.ratios.with_protein.group <- NULL

## The correlation between peptide and protein ratios defines the
## covariance
```

```
##  Var(ratio m) = Var(ratio mp) + Var(ratio p)
##                                    + 2 * Cov(ratio mp, ratio p),
##  Cov(ratio mp, ratio p) = 2 * cor * Sd(ratio mp) * Sd(ratio p),
##  with m = modifcation, mp = modified peptide, p = protein
peptide.protein.correlation <- 0

## quantification table whose columns are attached to the XLS
## quantification table
compare.to.quant <- NULL

######################################################################
## Report properties

write.qc.report=TRUE
write.report=TRUE
write.xls.report=TRUE

## Use name for report, ie NAME.quant.xlsx instead of
## isobar-analysis.xlsx
use.name.for.report=TRUE

## PDF Analysis report sections: Significant proteins and protein
## details
show.significant.proteins=FALSE
show.protein.details=TRUE

### QC REPORT OPTIONS ###
#qc.maplot.pairs=FALSE # plot one MA plot per tag (versus all others)
qc.maplot.pairs=TRUE # plot MA plot of each tag versus each tag

### XLS REPORT OPTIONS ###
## Spreadsheet format: Either 'xlsx' or 'xls'
# spreadsheet.format="xlsx"
spreadsheet.format="xlsx"

## XLS report format 'wide' or 'long '.

## 'wide' format outputs ratios in separate columns of the same record
## (i.e. one line per protein)
## 'long' format outputs ratios in separate records (i.e. one line per
## ratio)
# xls.report.format="wide"
xls.report.format="long"

## XLS report columns in quantification tab
##  possible values: ratio, is.significant, CI95.lower, CI95.upper,
##                   ratio.minus.sd, ratio.plus.sd,
##                   p.value.ratio, p.value.sample, n.na1, n.na2,
##                   log10.ratio, log10.variance,
##                   log2.ratio, log2.variance
##  only for summarize=TRUE: n.pos, n.neg
xls.report.columns <- c("ratio","is.significant","ratio.minus.sd",
                        "ratio.plus.sd","p.value.ratio","p.value.sample",
                        "log10.ratio","log10.variance")
```

```
## Perl command to be used for Excel report generation
# perl.cmd = "C:/Strawberry/perl/bin/perl.exe"
# perl.cmd = "C:/Perl/bin/perl.exe"
# perl.cmd = "perl5"
perl.cmd = "perl"

######################################################################
## Etc

sum.intensities=FALSE

datbase="Uniprot"

scratch=list()

##
#  compile LaTeX reports into PDF files
compile=TRUE

# zip final report files into archive
zip=FALSE

# warning level (see 'warn' in ?options)
warning.level=1


############################################################
## Novel options

shrink.mean=TRUE
use.t.stat=TRUE
```

C Dependencies

C.1 LaTeX and PGF/TikZ

LaTeX is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation. It is available as free software[3]. PGF is a TeX macro package for generating graphics It comes with a user-friedly syntax layer called TikZ[4].

LaTeX is used for creating PDF analysis reports, with the PGF package creating the graphics. Go to `http://www.latex-project.org` to get information on how to download and install a LaTeX system and packages.

C.2 Perl

Perl is a high-level, general-purpose, interpreted, dynamic programming language. Perl is required for two tasks:

- Conversion of Pidres XML and Mascot DAT files to ID CSV format;
- Creation of Microsoft Excel format data analysis report.

Go to `http://www.perl.org` to download and get help on the installation of Perl on your Operating System.
For file format conversion, perl module `Statistics::Lite` is required. For Excel export `Spreadsheet::WriteExcel`.
All Perl scripts are in the subdirectory `pl` of the isobar package installation.

[3] `http://www.latex-project.org`
[4] `http://sourceforge.net/projects/pgf`

```
> ## execute to find the path and file location in your installation.
> system.file("pl",package="isobar") ## path
> list.files(system.file("pl",package="isobar")) ## files
```

`mascotParser2.pl` and `pidresParser2.pl` convert from respective protein search outputfiles to a XML file format, which can be converted into a CSV file readable by *isobar* by using `psx2tab2.pl`.

`mascotParser2.pl` coverts from Mascot format, and requires the file `modifconv.csv` as a definition of modification names. `pidresParser2.pl` converts from Phenyx output and requires the file `parsersConfig.xml`. `tab2xls.pl` converts csv file to different sheets of an Excel spreadsheet.

```
> ## execute on your system
> system(paste("perl",system.file("pl","mascotParser2.pl",package="isobar"),
+              "--help"))
> print(paste("perl",system.file("pl","pidresParser2.pl",package="isobar"),
+              "--help"))
```

D Session Information

The version number of R and packages loaded for generating the vignette were:

```
>   toLatex(sessionInfo())
```

- R version 4.4.1 (2024-06-14), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Time zone: `Etc/UTC`
- TZcode source: `system (glibc)`
- Running under: `Ubuntu 24.04.1 LTS`
- Matrix products: default
- BLAS: `/usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3`
- LAPACK: `/usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.26.so`; LAPACK version3.12.0
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: Biobase 2.67.0, BiocGenerics 0.53.0, distr 2.9.5, ggplot2 3.5.1, isobar 1.53.0, sfsmisc 1.1-19, startupmsg 0.9.7
- Loaded via a namespace (and not attached): AnnotationDbi 1.69.0, BiocFileCache 2.15.0, Biostrings 2.75.0, DBI 1.2.3, GenomeInfoDb 1.41.2, GenomeInfoDbData 1.2.13, IRanges 2.39.2, KEGGREST 1.45.1, MASS 7.3-61, R6 2.5.1, RSQLite 2.3.7, Rcpp 1.0.13, S4Vectors 0.43.2, UCSC.utils 1.1.0, XVector 0.45.0, biomaRt 2.63.0, bit 4.5.0, bit64 4.5.2, blob 1.2.4, buildtools 1.0.0, cachem 1.1.0, cli 3.6.3, colorspace 2.1-1, compiler 4.4.1, crayon 1.5.3, curl 5.2.3, dbplyr 2.5.0, digest 0.6.37, dplyr 1.1.4, fansi 1.0.6, farver 2.1.2, fastmap 1.2.0, filelock 1.0.3, generics 0.1.3, glue 1.8.0, grid 4.4.1, gtable 0.3.6, hms 1.1.3, httr 1.4.7, httr2 1.0.5, jsonlite 1.8.9, knitr 1.48, labeling 0.4.3, lifecycle 1.0.4, magrittr 2.0.3, maketools 1.3.1, memoise 2.0.1, munsell 0.5.1, pillar 1.9.0, pkgconfig 2.0.3, plyr 1.8.9, png 0.1-8, prettyunits 1.2.0, progress 1.2.3, rappdirs 0.3.3, rlang 1.1.4,

scales 1.3.0, stats4 4.4.1, stringi 1.8.4, stringr 1.5.1, sys 3.4.3, tibble 3.2.1, tidyselect 1.2.1, tools 4.4.1, utf8 1.2.4, vctrs 0.6.5, withr 3.0.2, xfun 0.48, xml2 1.3.6, zlibbioc 1.51.2