LFQbench

Jörg Kuharev & Pedro Navarro

2016-08-30

LFQbench is an open source R package for the automated evaluation of label-free quantification performance. The evaluation bases on the interpretation of the quantitative analysis results of hybrid proteome samples prepared in known ratios.

LFQbench calculates and represents graphically a set of qualitative and quantitative performance metrics like identification rates, precision and accuracy of quantification, providing developers and end-users with a standardized set of reports to enable an in-depth performance evaluation of their software and analysis platforms.

Step by step example

In this section we show and comment a stepwise analysis of a single report produced by ISOQuant (software for the quantitative post-processing analysis of Waters HDMS^E data). The sample set is composed of two hybrid proteome samples with symmetric 10:1 and 1:10 expression ratios:

- 65% HUMAN proteins in both samples
- 30% YEAST proteins in sample A and 3% in sample B
- 3% ECOLI proteins in sample A and 30% in sample B

Define sample set composition

The sample set composition is defined as a *data.frame* that contains a row per species and a column per sample. In addition to the number of samples, species names are stored in the first column. Following columns define corresponding relative protein amounts for each sample.

```
sampleComposition = data.frame(
    species = c("HUMAN","YEAST", "ECOLI"),
    A       = c(  67,     30,       3   ),
    B       = c(  67,      3,      30   )
)
```

Define the data sets you want to analyse.

Each dataset is defined as a named vector of strings. The strings define the raw file names of the dataset as they are defined at the report(s) of the software tool(s) you want to benchmark. FSWE can analyse two samples (sample A and sample B), which defer on their species composition ratio. It is required to input the same number of replicates for both samples A and B. The first half of values of the named vector defining raw names is interpreted as sample A, and the second half is interpreted as sample B.

```
dataSets = data.frame(
    "HYE110_SynaptG2S" = c(
        paste(rep("HYE110_A"), 1:3, sep = "."),
        paste(rep("HYE110_B"), 1:3, sep = ".")
    ),
```

```
    row.names = c( "A1", "A2", "A3", "B1", "B2", "B3" )
)
```

Define the species tags

Protein names should provide a clue about the species. A convenient format is Uniprot's entry name (example: 1433B_HUMAN is the entry name of the 14-3-3 protein beta/alpha of human). These species tags (clues) must be defined in FSWE as a named list.

```
speciesTags = list(
    HUMAN = "_HUMAN",
    YEAST = "_YEAS",
    ECOLI = "_ECOLI"
)
```

Initialize LFQbench

Before using any LFQbench functionality, LFQbench library must be loaded and its modules must be initialized using *LFQbench.initConfiguration()* and *FSWE.initConfiguration()* functions. LFQbench module must be initialized before FSWE.

```
library(LFQbench)

LFQbench.initConfiguration(
    SampleComposition = sampleComposition
)

FSWE.initConfiguration(
    injectionNames = dataSets,
    speciesTags = speciesTags
)
```

Note: The initialization process will create new objects in the global R environment

- **LFQbench.Config**: containing a list of configuration parameters for the analysis. The configuration values can be directly as paremeters of the command *LFQbench.initConfiguration()* or modified by using the command *LFQbench.changeConfiguration()* at a later time point.
- **FSWE.modificationsToUniMod**: a list of default fixed and variable modifications as they appear within the sequence, and their corresponding translation to UniMod. If you miss any modification, you may add it by using the function FSWE.addModification.
- **FSWE.configFunctionForSoftware**: this object contains configuration parameters to read each specific software tool report. There are many default software tool configurations (see FSWE.softwareNames), and you can add your own configurations by using FSWE.addSoftwareConfiguration.

Define data root folder

Set the directory that stores all software tool reports as data root folder. You may also create a subfolder structure, which will contain LFQbench analyses.

```
srcDir = "../ext/data/vignette_examples/hye110"

LFQbench.setDataRootFolder(
    rootFolder = srcDir,
```

```
    createSubfolders = T
)
```

Add configuration for ISOQuant peptide report

FSWE already contains some predefined software report formats.

```
print( paste( FSWE.softwareNames, collapse="," ) )
```

[1] "DIAumpire,DIAumpBuiltinProteins,OpenSWATH,PeakView,PViewNoFilter,PViewBuiltinProteins,Skyline,Spectronaut"

In case you need to add an extra software report format, it only requires to know some necessary column names. In this example, we use a non default software tool report from ISOQuant. We can configure it with the following command:

```
FSWE.addSoftwareConfiguration(
    # Software configuration name
    softwareName = "ISOQuant_pep",

    # input_format can be wide or long.
    # Wide contains all quantitative values (all samples and replicates)
    # for a peptide in a single row,
    # whereas long contains a single quantitative value (just one replicate) in a row.
    input_format = "wide",

    # it is important to know that LFQbench honours the extension:
    # csv are COMMA separated values,
    # tsv are TAB separated values
    input.extension = "*.csv$",

    # how NA (not available) values are reported
    nastrings = " ",

    # in long formats, how the quantitative value column is named
    quantitative.var = make.names("intensity in"),

    # in wide formats, how quantitative values are tagged
    #(they also should include the injection names reported at the datasets object)
    quantitative.var.tag = make.names("intensity in"),

    # name of the protein name variable.
    # Remember: protein names should include species information (speciesTags)
    protein.var = "entry",

    # variable name of sequence
    # (including modifications as defined in FSWE.modificationsToUniMod)
    sequence.mod.var = "sequence",

    # variable name of the precursar charge state.
    charge.var = make.names("signal_charge")
)
```

Add an amino acid modification

You may add new modifications (or modifications reported in different ways for each software tool) by using the command *FSWE.addModification()*. You must use a regular expression compatible format to add modifications.

```
FSWE.addModification(
    modificationRegExps = "\\[Oxi\\]",
    UniModStrings = "\\(UniMod:35\\)"
)

# list modifications available
print( FSWE.modificationsToUniMod )
```

Generate reports in LFQbench format

FSWE parses software tool reports, makes some common data processing (like adding up quantitative values of the different charge states for a same peptide), and outputs each analysed report file in two different files (peptide and protein summaries) compatible with LFQbench at the newly created *input* subfolder of the data root folder.

```
inputFiles = list.files(
    path = LFQbench.Config$DataRootFolder,
    pattern = "\\..+"
)

nix = sapply(
    inputFiles,
    FSWE.generateReports,
        softwareSource = "guess",
        keep_original_names = T,
        singleHits = F,
        plotHistogram = T,
        plotHistNAs = T,
        reportSequences = F
)
```

Note: By default FSWE assigns the reading report configuration in function of the report name (softwareSource = "guess"). This is done by parsing the start of the file name and comparing it with the available *FSWE.softwareNames*. It is then recommendable to rename your files by including at the start the software name as it appears at the *FSWE.configFunctionForSoftware* (you can see the current available names at *FSWE.softwareNames*). In this execution, the current available names are: . By renaming the file names this way, you are able to run all LFQbench analyses of different software tools in the same LFQbench analysis batch.

Perform LFQbench analysis

LFQbench analyses can be run in a batch.

```
# some configuration changes for beautifying plots
LFQbench.changeConfiguration(
    LogIntensityPlotRange = c(9,21),
    LogRatioPlotRange = c(-7,7)
)
```

```
# run batch analysis and keep result set
res = LFQbench.batchProcessRootFolder()
```

All output files of a batch analysis will be stored at *log* and *plot* subfolders of the data root folder, and also they can be further used to represent data online.

Display metrics

LFQbench produces a set of predefinet metrics for every benchmarked file. You can access this metrics using the function *LFQbench.getMetrics()*.

```
# getting the result set of the first benchmarked file
rs = res[[1]]

m = LFQbench.getMetrics(
    resultSet = rs
)

# get local accuracy and precision (by intensity tertiles)
acc = m$`Local accuracy`$`A:B`
prec = m$`Local precision`$`A:B`
```

Table 1: Local accuracy

	HUMAN	YEAST	ECOLI
Q1	0.0407167	0.0961928	0.3142869
Q2	0.0099683	-0.0211730	-0.1345117
Q3	-0.0336584	-0.5847002	0.2553325

Table 2: Local precision

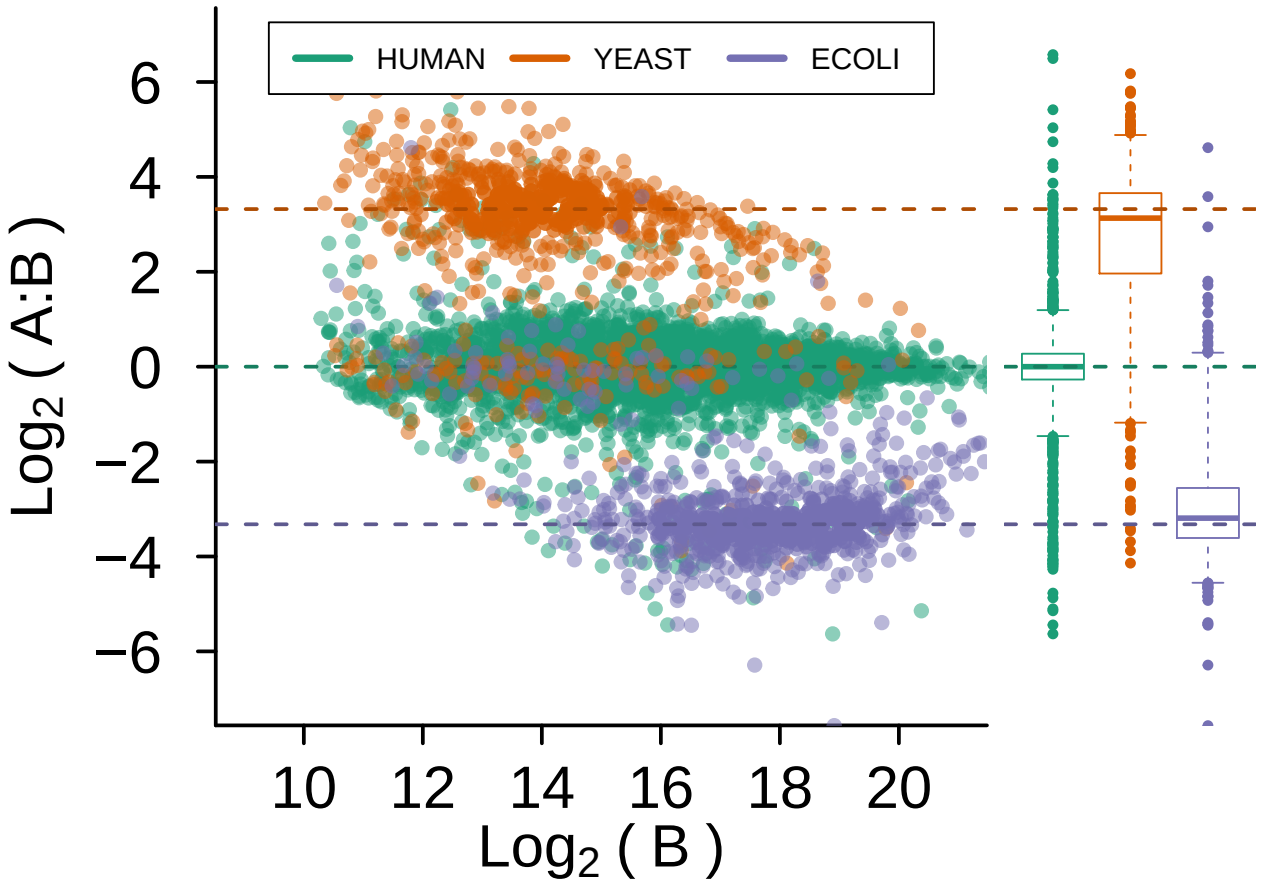
	HUMAN	YEAST	ECOLI
Q1	0.9431045	1.763697	1.7543298
Q2	0.7325138	1.505324	0.8127825
Q3	0.6213687	1.782097	0.9078830

Full list of predefined metrics: name,Identification statistics,Quantification statistics,Technical variance,Global accuracy,Global precision,Global species overlap,Local accuracy,Local precision,Local species overlap

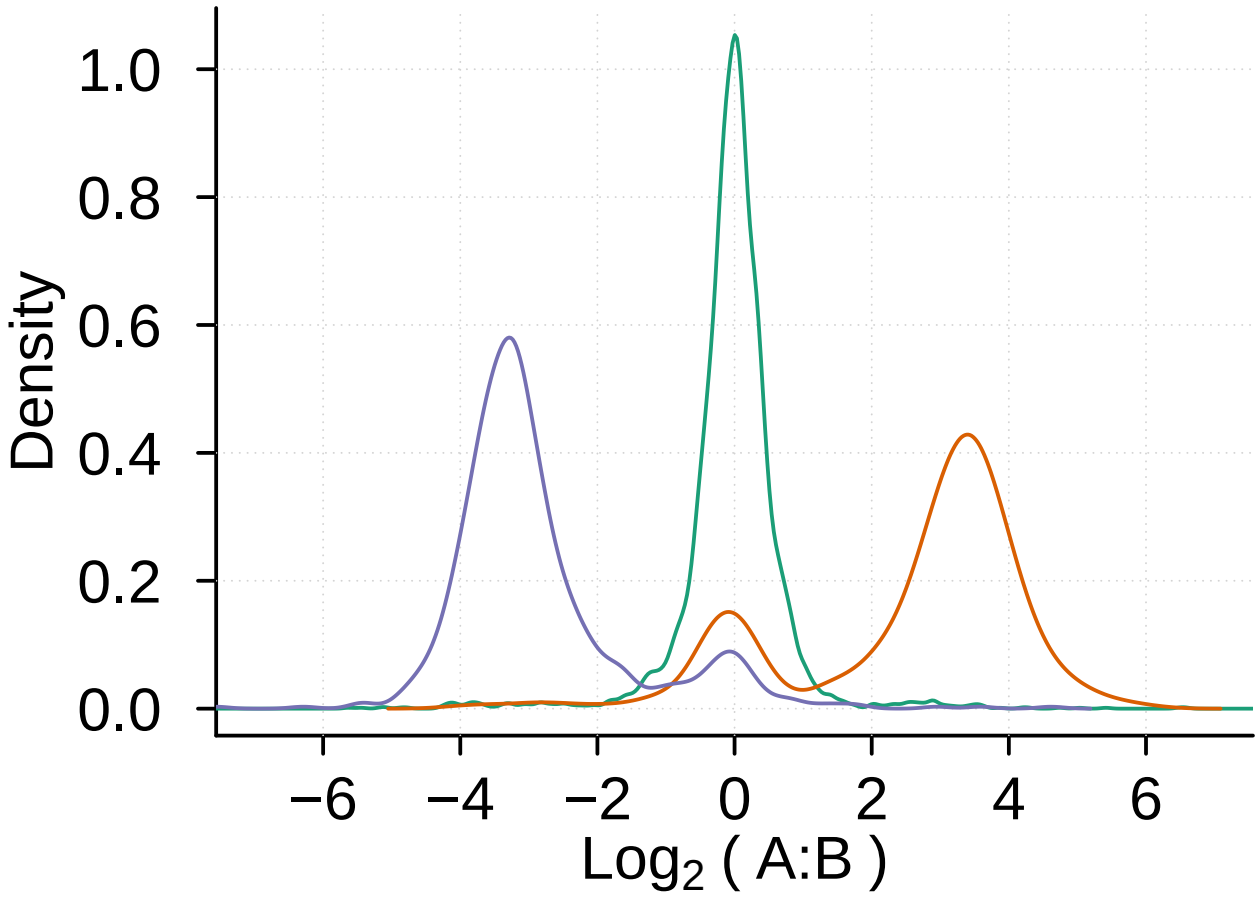
Display plots

```
# get the benchmark result for the first sample pair of the recently used result set
samplePairRes = rs$result[[1]]

# display the scatter plot
LFQbench.showScatterAndBoxPlot(
    samplePair = samplePairRes,
    showLegend = T
)
```




```
# display the distributions of log ratios
LFQbench.showDistributionDensityPlot(
    samplePair = samplePairRes,
    showLegend = F
)
```



At once example

In this section we show a complete analysis of two reports produced by Spectronaut and SWATH 2.0 (PeakView) software tools. The sample set is composed of two hybrid proteome samples with asymmetric 2:1 and 1:4 expression ratios:

- 65% HUMAN proteins in both samples
- 30% YEAST proteins in sample A and 15% in sample B
- 5% ECOLI proteins in sample A and 20% in sample B

```
sampleComposition = data.frame(
  species = c("HUMAN", "YEAST", "ECOLI"),
  A       = c(  65,       30,     05   ),
  B       = c(  65,       15,     20   )
)

dataSets = data.frame(
  "HYE124_TTOF6600_64var" = c(
    "lgillet_I150211_008", "lgillet_I150211_010", "lgillet_I150211_012", # A
    "lgillet_I150211_009", "lgillet_I150211_011", "lgillet_I150211_013"  # B
  ),
  row.names = c( "A1", "A2", "A3", "B1", "B2", "B3" )
)
```

```
speciesTags = list(
  HUMAN = "_HUMAN",
  YEAST = "_YEAS",
  ECOLI = "_ECOLI"
)

LFQbench.initConfiguration(
  SampleComposition = sampleComposition
)

FSWE.initConfiguration(
  injectionNames = dataSets,
  speciesTags = speciesTags
)

# we don't need to define new software report format in this example
# because Spectronaut and PeakView (SWATH 2.0) report formats are predefined in FSWE

srcDir = "../ext/data/vignette_examples/hye124"

LFQbench.setDataRootFolder(
  rootFolder = srcDir,
  createSubfolders = T
)

inputFiles = list.files(
  path = LFQbench.Config$DataRootFolder,
  pattern = "\\..+"
)

nix = sapply(
  inputFiles,
  FSWE.generateReports,
  softwareSource = "guess",
  keep_original_names = T,
  singleHits = F,
  plotHistogram = T,
  plotHistNAs = T,
  reportSequences = F
)

hye124.res = LFQbench.batchProcessRootFolder()
```