Package 'Pirat'

December 17, 2024

Type Package

Title Precursor or Peptide Imputation under Random Truncation

Description Pirat enables the imputation of missing values (either MNARs or MCARs) in bottom-up LC-MS/MS proteomics data using a penalized maximum likelihood strategy. It does not require any parameter tuning, it models the instrument censorship from the data available. It accounts for sibling peptides correlations and it can leverage complementary transcriptomics measurements.

Version 1.1.0

License GPL-2

Encoding UTF-8

biocViews Proteomics, MassSpectrometry, Preprocessing, Software

RoxygenNote 7.3.2

BiocViews Proteomics, MassSpectrometry

Depends R (>= 4.4.0)

Imports basilisk, reticulate, progress, ggplot2, MASS, invgamma, grDevices, stats, graphics, SummarizedExperiment, S4Vectors

Suggests knitr, BiocStyle

VignetteBuilder knitr

URL http://www.prostar-proteomics.org/

BugReports https://github.com/prostarproteomics/Pirat/issues

StagedInstall no

git_url https://git.bioconductor.org/packages/Pirat

git_branch devel

git_last_commit d00d343

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2024-12-16

Author Lucas Etourneau [aut],
Laura Fancello [aut],
Samuel Wieczorek [cre, aut] (ORCID:
<https://orcid.org/0000-0002-5016-1203>),
Nelle Varoquaux [aut],
Thomas Burger [aut]

Maintainer Samuel Wieczorek <samuel.wieczorek@cea.fr>

Contents

`envPirat` *Creates a BasiliskEnvironment class*

Description

Please refer to the package 'basilisk'.

Usage

```
envPirat
```

Format

An object of class `BasiliskEnvironment` of length 1.

Value

An instance of the class 'BasiliskEnvironment'

`estimate_gamma` *Estimate missingness parameters Gamma*

Description

Estimate missingness parameters Gamma

Usage

```
estimate_gamma(pep.ab.table, mcar = FALSE)
```

Arguments

`pep.ab.table`	The peptide or precrursor abundance matrix, with molecules in columns and samples in row.
`mcar`	If TRUE, forces gamma_1 = 0.

Value

A list of the containing missingness parameters gamma_0 and gamma_1.

Examples

```
data(subbouyssie)
estimate_gamma(subbouyssie$peptides_ab)
```

`estimate_psi_df` *Estimate psi and degrees of freedom*

Description

Estimate the inverse-gamma parameters from the distribution of observed peptide variances in an abundance table.

Usage

```
estimate_psi_df(pep.ab.table)
```

Arguments

`pep.ab.table`	The peptide or precursor abundance matrix, with molecules in columns and samples in row (can contain missing values).

Value

List containing estimated fitted hyperparameters df (degrees of freedom) and psi (inverse scale).

Examples

```
data(subbouyssie)
obj <- subbouyssie
# Keep only fully observed peptides
obs2NApep <- obj$peptides_ab[ ,colSums(is.na(obj$peptides_ab)) <= 0]
estimate_psi_df(obs2NApep)
```

---

get_indexes_embedded_prots

Indexes of PGs embedded in each others

Description

Returns indexes of PGs that are embedded in others

Usage

```
get_indexes_embedded_prots(adj)
```

Arguments

adj	An adjacency matrix between precursors/peptides and PGs

Value

A vector of indices

Examples

```
data(subbouyssie)
get_indexes_embedded_prots(subbouyssie$adj)
```

`impute_block_llk_reset` *Impute each PG.*

Description

Imputes each PG separately and return the results for each PG.

Usage

```
impute_block_llk_reset(
  data.pep.rna.crop,
  psi,
  pep_ab_or = NULL,
  df = 1,
  nu_factor = 2,
  max_pg_size = NULL,
  min.pg.size2imp = 1,
  verbose = FALSE,
  ...
)
```

Arguments

`data.pep.rna.crop`	A list representing dataset
`psi`	Inverse scale parameter for IW prior of peptides abundances
`pep_ab_or`	In case we impute a dataset with pseudo-MVS, we can provide the ground truth abundance table, such that imputation will by done only for pseudo-MVs. This will accelerate imputation algorithm.
`df`	Estimate degree of freedom of the IG distribution fitted on observed variance.
`nu_factor`	Multiplication factor on degree of freedom. 2 by default.
`max_pg_size`	Maximum PGs size authorized for imputation. PG size is plitted if its size is above this threshold.
`min.pg.size2imp`	Minimum PG size to impute after splitting. PGs for which size is greater are not imputed. Should be lower than max_pg_size to have effect.
`verbose`	A boolean (FALSE as default) which indicates whether to display more details on the process
`...`	Additional arguments

Value

A list containing imputation results for each PG, the execution time, and adjacency matrix between peptides and PGs corresponding to the imputed PGs.

Examples

```
Py_impute_block_llk_reset <- function(data.pep.rna.mis, psi) {
proc <- basilisk::basiliskStart(envPirat)

func <- basilisk::basiliskRun(proc,
    fun = function(arg1, arg2) {

        imputed_pgs <- Pirat::impute_block_llk_reset(arg1, arg2)
        imputed_pgs
    }, arg1 = data.pep.rna.mis, arg2 = psi)

basilisk::basiliskStop(proc)
func
}

data(subbouyssie)
obs2NApep <- subbouyssie$peptides_ab[ ,colSums(is.na(subbouyssie$peptides_ab)) <= 0]
res_hyperparam <- estimate_psi_df(obs2NApep)
psi <- res_hyperparam$psi
Py_impute_block_llk_reset(subbouyssie, psi)
```

`impute_block_llk_reset_PG` *Impute each PG.*

Description

Imputes each PG separately accounting for transcriptomic dataset and returns the results for each PG.

Usage

```
impute_block_llk_reset_PG(
  data.pep.rna.crop,
  psi,
  psi_rna,
  rna.cond.mask,
  pep.cond.mask,
  pep_ab_or = NULL,
  df = 2,
  nu_factor = 1,
  max_pg_size = NULL,
  max.pg.size2imp = 1,
  verbose = FALSE,
  ...
)
```

Arguments

data.pep.rna.crop	A list representing dataset, with mRNA normalized counts and mRNA/PGs adjacecy table.
psi	Inverse scale parameter for IW prior of peptides abundances
psi_rna	Inverse scale parameter for IW prior of mRNA abundances
rna.cond.mask	Vector of size equal to the number of samples in mRNA abundance table, containing indices of conditions of each sample.
pep.cond.mask	Vector of size equal to the number of samples in peptide abundance table, containing indices of conditions of each sample.
pep_ab_or	In case we impute a dataset with pseudo-MVS, we can provide the ground truth abundance table, such that imputation will by done only for pseudo-MVs. This will accelerate imputation algorithm.
df	Estimate degree of freedom of the IG distribution fitted on observed variance.
nu_factor	Multiplication factor on degree of freedom. 2 by default.
max_pg_size	Maximum PGs size authorized for imputation. PG size is plitted if its size is above this threshold.
max.pg.size2imp	Maximum PG size to impute after splitting. PGs for which size is greater are not imputed. Should be lower than max_pg_size to have effect.
verbose	A boolean (FALSE as default) which indicates whether to display more details ont the process
...	Additional parameters

Value

A list containing imputation results for each PG, the execution time, and adjacency matrix between peptides and PGs corresponding to the imputed PGs.

Examples

```
Py_impute_block_llk_reset_PG <- function(data.pep.rna.crop, ...) {
proc <- basilisk::basiliskStart(envPirat)

func <- basilisk::basiliskRun(proc,
    fun = function(arg1, ...) {
        Pirat::impute_block_llk_reset_PG(arg1, ...)
    }, arg1 = data.pep.rna.crop, ...)
basilisk::basiliskStop(proc)
func
}

data(subropers)
obj <- subropers
# Keep only fully observed peptides
obs2NApep <- obj$peptides_ab[ ,colSums(is.na(obj$peptides_ab)) <= 0]
res_hyperparam_pep = estimate_psi_df(obs2NApep)
```

```
psi_pep <- res_hyperparam_pep$psi
obs2NArna <- obj$rnas_ab[ ,colSums(obj$rnas_ab == 0) <= 0]
res_hyperparam_rna = estimate_psi_df(obs2NArna)
psi_rna <- res_hyperparam_rna$psi
# paired proteomic transcriptomic setting
cond_mask <- seq(nrow(obj$peptides_ab))
imputed_pgs <- Py_impute_block_llk_reset_PG(
    data.pep.rna.crop = obj,
    psi = psi_pep,
    psi_rna = psi_rna,
    rna.cond.mask = cond_mask,
    pep.cond.mask = cond_mask)
```

`impute_from_blocks` *Impute abundance table from PGs results*

Description

From imputation results in each PG and the associate adjacency peptide/PG matrix,imputes the original abundance table. .

Usage

```
impute_from_blocks(logs.blocks, data.pep.rna, idx_blocks = NULL)
```

Arguments

`logs.blocks`	List of PGs imputation results, that also contains related peptide/PGs adjacency matrix.
`data.pep.rna`	List representing the dataset not yet imputed
`idx_blocks`	Indices of PGs for which imputation results should be integrated

Value

The original peptide abundance table with imputed values.

Examples

```
Py_impute_block_llk_reset <- function(data.pep.rna.mis, psi) {
proc <- basilisk::basiliskStart(envPirat)

func <- basilisk::basiliskRun(proc,
    fun = function(arg1, arg2) {

        imputed_pgs <- Pirat::impute_block_llk_reset(arg1, arg2)
        imputed_pgs
```

```
    }, arg1 = data.pep.rna.mis, arg2 = psi)

basilisk::basiliskStop(proc)
func
}


data(subbouyssie)
obj <- subbouyssie
# Keep only fully observed peptides
obs2NApep <- obj$peptides_ab[ ,colSums(is.na(obj$peptides_ab)) <= 0]
res_hyperparam <- estimate_psi_df(obs2NApep)
psi <- res_hyperparam$psi
imputed_pgs <- Py_impute_block_llk_reset(obj, psi)
impute_from_blocks(imputed_pgs, obj)
```

`pipeline_llkimpute` *Pirat imputation function*

Description

Imputation pipeline of Pirat. First, it creates PGs. Then, it estimates parameters of the penalty term (that amounts to an inverse-Wishart prior). Second, it estimates the missingness mechanism parameters. Finally, it imputes the peptide/precursor-level dataset with desired extension.

Usage

```
my_pipeline_llkimpute(data.pep.rna.mis, ...)

pipeline_llkimpute(
  data.pep.rna.mis,
  pep.ab.comp = NULL,
  alpha.factor = 2,
  rna.cond.mask = NULL,
  pep.cond.mask = NULL,
  extension = c("base", "2", "T", "S"),
  mcar = FALSE,
  degenerated = FALSE,
  max.pg.size.pirat.t = 1,
  verbose = FALSE
)
```

Arguments

`data.pep.rna.mis`
: Parameter 'data.pep.rna.mis' of the function 'pipeline_llkimpute()'

`...`	Additional parameters for the function 'pipeline_llkimpute()'
`pep.ab.comp`	The pseudo-complete peptide or precursor abundance matrix, with samples in row and peptides or precursors in column. Useful only in mask-and-impute experiments, if one wants to impute solely peptides containing pseudo-MVs.
`alpha.factor`	Factor that multiplies the parameter alpha of the penalty described in the original paper.
`rna.cond.mask`	Vector of indexes representing conditions of samples of mRNA table, only mandatory if extension == "T". For paired proteomic and transcriptomic tables, should be c(1:n_samples).
`pep.cond.mask`	Vector of indexes representing conditions of samples of mRNA table, only mandatory if extension == "T". For paired proteomic and transcriptomic tables, should be c(1:n_samples).
`extension`	If NULL (default), classical Pirat is applied. If "2", only imputes PGs containing at least 2 peptides or precursors, and remaining peptides are left unchanged. If "S", Pirat-S is applied, considering sample-wise correlations only for singleton PGs. It "T", Pirat-T is applied, thus requiring **rnas_ab** and **adj_rna_pg** in list **data.pep.rna.mis**, as well as non-NULL **rna.cond.mask** and **pep.cond.mask**. Also, the maximum size of PGs for which transcriptomic data can be used is controlled with **max.pg.size.pirat.t**.
`mcar`	If TRUE, forces gamma_1 = 0, thus no MNAR mechanism is considered.
`degenerated`	If TRUE, applies Pirat-Degenerated (i.e. its univariate alternative) as described in original paper. Should not be TRUE unless for experimental purposes.
`max.pg.size.pirat.t`	When extension == "T", the maximum PG size for which transcriptomic information is used for imputation.
`verbose`	A boolean (FALSE as default) which indicates whether to display more details on the process

Value

The imputed **data.pep.rna.mis$peptides_ab** table.

The imputed **data.pep.rna.mis$peptides_ab** table.

NA

See Also

[pipeline_llkimpute()]

Examples

```
# Pirat classical mode
data(subbouyssie)
myResult <- my_pipeline_llkimpute(subbouyssie)

# Pirat with transcriptomic integration for singleton PGs
data(subropers)
```

```
nsamples = nrow(subropers$peptides_ab)
myResult <- my_pipeline_llkimpute(subropers,
extension = "T",
rna.cond.mask = seq(nsamples),
pep.cond.mask = seq(nsamples),
max.pg.size.pirat.t = 1)

## Not run:
myResult <- pipeline_llkimpute(subbouyssie)

## End(Not run)
```

`pirat2SE` *COnvert Pirat dataset to SummarizedExperiment*

Description

This function converts the original dataset structure into a SummarizedExperiment .

Usage

```
pirat2SE(peptides_ab, adj, mask_prot_diff = NULL, mask_pep_diff = NULL)
```

Arguments

`peptides_ab`	the peptide or precursor abundance matrix to impute, with samples in row and peptides or precursors in column;
`adj`	a n_peptide x n_protein adjacency matrix between peptides and proteins containing 0 and 1, or TRUE and FALSE. Can contain: **rnas_ab**, the mRNA normalized count matrix, with samples in row and mRNAs in column; **adj_rna_pg**, a n_mrna x n_protein adjacency matrix n_mrna and proteins containing 0 and 1, or TRUE and FALSE;
`mask_prot_diff`	(Optional) boolean vector of size equal to the number of proteins, indicating whether proteins are ground truth differentially abundant (typically in spike-in benchmark datasets).
`mask_pep_diff`	(Optional) boolean vector of size equal to the number of peptides, indicating whether peptides are ground truth differentially abundant (typically in spike-in benchmark datasets).

Value

An instance of the class 'SummarizedExperiment'

Examples

```
data(subbouyssie)
peptides_ab <- subbouyssie$peptides_ab
adj <- subbouyssie$adj
mask_prot_diff <- subbouyssie$mask_prot_diff
mask_pep_diff <- subbouyssie$mask_pep_diff
obj <- pirat2SE(peptides_ab, adj, mask_prot_diff, mask_pep_diff )
obj
```

`plot2hists` *Plot 2 histograms*

Description

Plot 2 histograms on the same graph.

Usage

```
plot2hists(
  d1,
  d2,
  name1 = "name1",
  name2 = "name2",
  titlename = "myTitle",
  xlab = "",
  freq = TRUE
)
```

Arguments

`d1`	vector of values for the first histogram
`d2`	vector of values for the first histogram
`name1`	Label for first histogram
`name2`	Label for 2nd histogram
`titlename`	Title of figure
`xlab`	X-axis label
`freq`	If True, bins heights correspond to raw counts, otherwise bins are normalized.

Value

A plot

Examples

```
v1 <- 1:10
v2 <- 5:25
plot2hists(v1, v2)
```

`plot_pep_correlations` *Empirical density of peptide correlations*

Description

Plot empirical densities of correlations between peptides within PG and at random, estimated by gaussian kernel. Note that only correlations between fully observed peptides are considered here.

Usage

```
plot_pep_correlations(pep.data, titlename = NULL, xlabel = "Correlations")
```

Arguments

`pep.data`	List representing dataset
`titlename`	Title of the graph displayed
`xlabel`	Label of x-axis

Value

The ggplot2 graph

Examples

```
data(subbouyssie)
plot_pep_correlations(subbouyssie, 'test')
```

`rm_pg_from_idx_merge_pg`
Remove PGs by index and merge

Description

Remove PG by index and merge transcripts (if transcriptomic information is available) of PG included in one another (under condition that they have peptide). Then it removes transcripts without PG. Do not remove peptides that are left without PG.

Usage

```
rm_pg_from_idx_merge_pg(l_pep_rna, pg_idx)
```

Arguments

l_pep_rna	A list representing dataset, formatted as in pipeline_llkimpute function
pg_idx	Vector of indices

Value

A list representing dataset.

Examples

```
data(ropers)
idxs_emb_prot = get_indexes_embedded_prots(ropers$adj)
ropers_wo_emb_prot = rm_pg_from_idx_merge_pg(ropers, idxs_emb_prot)
```

---

ropers *Ropers dataset*

Description

This dataset corresponds to 'Ropers2021' dataset, described in Pirat article.

Format

A list containing: - peptides_ab: numeric matrix of precrusors log2 abundances. - adj: adjacency matrix between peptides and PGs - rnas_ab: numeric matrix of gene expression log2 counts from mRNA analysis. - adj_rna_pg: adjacency matrix between genes and PGs

Value

A dataset

References

Ropers, D., Couté, Y., Faure, L., Ferré, S., Labourdette, D., Shabani, A., Trouilh, L., Vasseur, P., Corre, G., Ferro, M., Teste, M. A., Geiselmann, J., & de Jong, H. (2021). Multiomics Study of Bacterial Growth Arrest in a Synthetic Biology Application. ACS Synthetic Biology, 10(11), 2910–2926. https://doi.org/10.1021/ACSSYNBIO.1C00115/SUPPL_FILE/SB1C00115_SI_010.ZIP

split_large_pg *Split too large PGs*

Description

Randomly splits PGs with too many peptides/precursors, while keeping other PGs untouched. The new PGs created all have size equal to size max. Hence, some peptides can be duplicated in the new PGs created.

Usage

```
split_large_pg(adj, size_max)
```

Arguments

adj	Adjacency matrix between peptides and PGs.
size_max	Maximum PG size desired.

Value

New adjacency matrix between peptides and PGs.

Examples

```
data(subbouyssie)
split.obj <- split_large_pg(subbouyssie$adj, 5)
```

split_large_pg_PG *Splits too large PGs in proteogenomics context*

Description

Randomly splits PGs with too many peptides/precursors, while keeping other PGs untouched, and adapts adjacency matrix between mRNA and PGs accordingly. The new PGs created all have size equal to size_max (including peptides and mRNAs). Hence, some peptides and mRNA can be duplicated in the new PGs.

Usage

```
split_large_pg_PG(adj, size_max, adj_rna_pg)
```

Arguments

adj	Adjacency matrix between peptides and PGs.
size_max	Maximum PG size desired.
adj_rna_pg	Adjacency matrix between mRNA and PGs.

Value

List containing new adjacency matrix between peptides and PGs, and new adjacency matrix between mRNA and PGs.

Examples

```
data(subropers)
split.obj <- split_large_pg_PG(subropers$adj, 5, subropers$adj_rna_pg)
```

---

subbouyssie *Sub-Bouyssie dataset*

Description

This dataset is extracted from the original 'Bouyssie2020' dataset mentionned in Pirat article, where only 5 PGs were randomly selected.

Format

A list containing: - peptides_ab: numeric matrix of peptide (or precrusors) log2 abundances. - adj: adjacency matrix between peptides and PGs.

Value

A dataset

References

Bouyssié, D., Hesse, A. M., Mouton-Barbosa, E., Rompais, M., MacRon, C., Carapito, C., Gonzalez De Peredo, A., Couté, Y., Dupierris, V., Burel, A., Menetrey, J. P., Kalaitzakis, A., Poisat, J., Romdhani, A., Burlet-Schiltz, O., Cianférani, S., Garin, J., & Bruley, C. (2020). Proline: an efficient and user-friendly software suite for large-scale proteomics. Bioinformatics, 36(10), 3148–3155. https://doi.org/10.1093/BIOINFORMATICS/BTAA118

---

subropers *Sub-Ropers dataset*

Description

This dataset is extracted from the original 'Ropers2021' dataset described in Pirat article, where only 10 PGs were randomly selected.

Format

A list containing: - peptides_ab: numeric matrix of peptide (or precrusors) log2 abundances. - adj: adjacency matrix between peptides and PGs - rnas_ab: numeric matrix of gene expression log2 counts from mRNA analysis. - adj_rna_pg: adjacency matrix between genes and PGs

Value

A dataset

References

Ropers, D., Couté, Y., Faure, L., Ferré, S., Labourdette, D., Shabani, A., Trouilh, L., Vasseur, P., Corre, G., Ferro, M., Teste, M. A., Geiselmann, J., & de Jong, H. (2021). Multiomics Study of Bacterial Growth Arrest in a Synthetic Biology Application. ACS Synthetic Biology, 10(11), 2910–2926. https://doi.org/10.1021/ACSSYNBIO.1C00115/SUPPL_FILE/SB1C00115_SI_010.ZIP

---

wrapper_pipeline_llkimpute

Imputation method using SummarizedExperiment dataset

Description

This function imputes data from an instance of the SummarizedExperiment structure data. After a conversion step, it calls the function ‘my_pipeline_llkimpute‘.

Usage

```
wrapper_pipeline_llkimpute(se, ...)
```

Arguments

se	An instance of the class SummarizedExperiment
...	Additional arguments to pass to ‘my_pipeline_llkimpute()‘

Value

See my_pipeline_llkimpute() function

Examples

```
data(subbouyssie)
obj <- pirat2SE(subbouyssie$peptides_ab, subbouyssie$adj,
subbouyssie$mask_prot_diff, subbouyssie$mask_pep_diff )
res <- wrapper_pipeline_llkimpute(obj)
```

Index