Package: MOTL (via r-universe)

June 21, 2026

Title Multi-omics matrix factorization with transfer learning

Version 0.99.1

Description A transfer learning algorithm for multi-omics matrix factorization called 'MOTL' (Multi-Omics Transfer Learning). 'MOTL' is a Bayesian transfer learning method, based on 'MOFA'. 'MOTL' infers latent factor values for a multi-omics target dataset, consisting of a small number of samples, by incorporating latent factor values already inferred with a 'MOFA' factorization of a large, heterogeneous, learning dataset.

License GPL-3 + file LICENCE

URL https://github.com/MOohTus/MOTL, https://github.com/david-hirst/MOTL

BugReports https://github.com/MOohTus/MOTL/issues, https://github.com/david-hirst/MOTL/issues

Depends R (>= 4.2.0)

Imports base, data.table, DESeq2, dplyr, matrixStats, rhdf5, stats, stats4, SummarizedExperiment, utils, methods, MOFA2

Suggests knitr, rmarkdown, testthat (>= 3.0.0), withr, BiocStyle

VignetteBuilder knitr

biocViews DimensionReduction, FeatureExtraction, Bayesian, Normalization

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

LazyData false

Config/pak/sysreqs libicu-dev libpng-dev libssl-dev python3 zlib1g-dev

Repository https://biocstaging.r-universe.dev

Date/Publication 2026-06-19 07:04:32 UTC

RemoteUrl https://github.com/BiocStaging/MOTL

RemoteRef HEAD

RemoteSha 2b258d0a0ade3280d1d1d27dd0252b568ea04a3c

Contents

countsNormalization *Normalize counts data*

Description

Normalize counts data using DESeq2 normalization. Two ways of normalization:

1. Use the pre-calculated Geometric means of the learning dataset
2. Use calculated Geometric means of the `expdat` dataset given in input

Usage

```
countsNormalization(expdat, GeoMeans)
```

Arguments

`expdat`	SE object of experimental data (could be miRNA or mRNA)
`GeoMeans`	if it's a character, Geometric means will be calculated for the `expdat` variable given in input (learning or target dataset). If it's a numerical vector, given Geometric means will be used for the normalization (the ones pre-calculated from the learning dataset)

Details

If `is.numeric(GeoMeans) == TRUE`, input data are normalized with pre-calculated Geometric means (from learning dataset). If non values are provided, Geometric means is calculated based on the input dataset using `GeoMeanFun` function. Then, the input dataset is normalized using these Geometric means.

Value

list of data.frame of the counts normalized and Geometric means calculated

Examples

```
## Create a matrix with "counts" data
## Then, create a summarized experiment object
expdat <- matrix(rexp(200, rate = .1), ncol = 20)
expdat <- apply(expdat, MARGIN = 2, round)
expdat <- SummarizedExperiment::SummarizedExperiment(expdat)

## With "newGeoMeans", geometric means will be calculated based on the
## input matrix
GeoMeans <- "newGeoMeans"

expdat_counts_norm <- countsNormalization(expdat, GeoMeans)
```

countsTransformation *Log2 transform and select top data based on variance*

Description

Log2 transform and select top data based on variance

Usage

```
countsTransformation(expdat_count, TopD)
```

Arguments

expdat_count	data.frame of the counts
TopD	number of features to keep

Value

data.frame of the log2 transformed and filtered data

Examples

```
##
expdat_count <- matrix(rexp(200, rate = .1), ncol = 20)
expdat_count <- apply(expdat_count, MARGIN = 2, round)

## input matrix
TopD <- 20

expdat_counts_fltr <- countsTransformation(expdat_count, TopD)
```

E_Z_SqE_W_Sq_update *Calculate* E_Z_SqE_W_Sq

Description

E_Z_SqE_W_Sq is the multiplication of the squared expected values of Z matrix with the squared expected values of W matrix $((E[Z])^2) * ((E[W])^2)$.

Usage

```
E_Z_SqE_W_Sq_update(view, ZMu_0, ZMu, Fctrzn_Lrn_W0, Fctrzn_Lrn_W)
```

Arguments

view	current view name
ZMu_0	vector of coefficients for weight intercepts
ZMu	matrix of Z values
Fctrzn_Lrn_W0	list of factorized learning set weight intercept matrices
Fctrzn_Lrn_W	list of factorized learning set weight matrices

Value

E_Z_SqE_W_Sq for current view

Examples

```
data("TL_param", package = "MOTL")
view <- "mRNA"
ZMu <- TL_param$ZMu
ZMu_0 <- TL_param$ZMu_0
Fctrzn_Lrn_W0 <- TL_param$Fctrzn_Lrn_W0
Fctrzn_Lrn_W <- TL_param$Fctrzn_Lrn_W

E_Z_SqE_W_Sq <-
    E_Z_SqE_W_Sq_update(view, ZMu_0, ZMu, Fctrzn_Lrn_W0, Fctrzn_Lrn_W)
```

E_ZE_W_update *Calculate* E_ZE_W

Description

E_ZE_W is the multiplication of the expected values of Z matrix with the expected values of W matrix $E[Z]E[W]$.

Usage

```
E_ZE_W_update(view, ZMu_0, ZMu, Fctrzn_Lrn_W0, Fctrzn_Lrn_W)
```

Arguments

view	current view name
ZMu_0	vector of coefficients for weight intercepts
ZMu	matrix of Z values
Fctrzn_Lrn_W0	list of factorized learning set weight intercept matrices
Fctrzn_Lrn_W	list of factorized learning set weight matrices

Value

E_ZE_W for current view

Examples

```
data("TL_param", package = "MOTL")
view <- "mRNA"
ZMu <- TL_param$ZMu
ZMu_0 <- TL_param$ZMu_0
Fctrzn_Lrn_W0 <- TL_param$Fctrzn_Lrn_W0
Fctrzn_Lrn_W <- TL_param$Fctrzn_Lrn_W

E_ZE_W <- E_ZE_W_update(view, ZMu_0, ZMu, Fctrzn_Lrn_W0, Fctrzn_Lrn_W)
```

E_ZSqE_WSq_update *Calculate* E_ZSqE_WSq

Description

E_ZSqE_WSq is the multiplication of the expected values of the squared Z matrix with the expected values of the squared W matrix $E[Z^2] * E[W^2]$

Usage

```
E_ZSqE_WSq_update(view, ZMu_0, ZMuSq, Fctrzn_Lrn_W0, Fctrzn_Lrn_WSq)
```

Arguments

view	current view name
ZMu_0	vector of coefficients for weight intercepts
ZMuSq	matrix of squared Z values
Fctrzn_Lrn_W0	list of factorized learning set weight intercept matrices
Fctrzn_Lrn_WSq	list of factorized learning set weight squared matrices

Value

E_ZSqE_WSq for current view

Examples

```
data("TL_param", package = "MOTL")
view <- "mRNA"
ZMuSq <- TL_param$ZMuSq
ZMu_0 <- TL_param$ZMu_0
Fctrzn_Lrn_W0 <- TL_param$Fctrzn_Lrn_W0
Fctrzn_Lrn_WSq <- TL_param$Fctrzn_Lrn_WSq

E_ZSqE_WSq <-
    E_ZSqE_WSq_update(view, ZMu_0, ZMuSq, Fctrzn_Lrn_W0, Fctrzn_Lrn_WSq)
```

E_ZWSq_update *Calculate* E_ZWSq

Description

E_ZWSq is the expected values of the multiplication of the Z matrix with weight squared W matrix $E[(ZW)^2]$.

Usage

```
E_ZWSq_update(view, E_ZE_W, ZMuSq, E_Z_SqE_W_Sq, E_ZSqE_WSq)
```

Arguments

view	current view name
E_ZE_W	matrix of $E[Z]E[W]$ values for current view
ZMuSq	matrix of squared ZMu values for current view
E_Z_SqE_W_Sq	matrix of $((E[Z])^2)((E[W])^2)$ values for current view
E_ZSqE_WSq	matrix of $E[Z^2]E[W^2]$ values for current view

Value

E_ZWSq values for current view

Examples

```
data("TL_param", package = "MOTL")
view <- "mRNA"
ZMuSq <- TL_param$ZMuSq
ZMu <- TL_param$ZMu
ZMu_0 <- TL_param$ZMu_0
Fctrzn_Lrn_W0 <- TL_param$Fctrzn_Lrn_W0
Fctrzn_Lrn_W <- TL_param$Fctrzn_Lrn_W
Fctrzn_Lrn_WSq <- TL_param$Fctrzn_Lrn_WSq
```

```
E_ZE_W <- list()
E_Z_SqE_W_Sq <- list()
E_ZSqE_WSq <- list()
E_ZWSq <- list()


E_ZE_W$mRNA <- E_ZE_W_update(view, ZMu_0, ZMu, Fctrzn_Lrn_W0, Fctrzn_Lrn_W)
E_Z_SqE_W_Sq$mRNA <-
    E_Z_SqE_W_Sq_update(view, ZMu_0, ZMu, Fctrzn_Lrn_W0, Fctrzn_Lrn_W)
E_ZSqE_WSq$mRNA <-
    E_ZSqE_WSq_update(view, ZMu_0, ZMuSq, Fctrzn_Lrn_W0, Fctrzn_Lrn_WSq)

E_ZWSq$mRNA <- E_ZWSq_update(view, E_ZE_W, ZMuSq, E_Z_SqE_W_Sq, E_ZSqE_WSq)
```

`ELBO_calculation` *Calculate the ELBO value for the current view/iterations*

Description

for poisson and bernoulli it is the bound which is used for gaussian it is expanded gaussian log likelihood

Usage

```
ELBO_calculation(
  view,
  likelihoods,
  Tau,
  TauLn,
  E_ZWSq,
  E_ZE_W,
  Zeta,
  YTrg,
  YGauss,
  PoisRateCstnt
)
```

Arguments

`view`	a character of current view name data
`likelihoods`	a named list of data types. The list can contain `gaussian`, `poisson` or `bernoulli` depending of the data type. Names are the view names.
`Tau`	list of Tau matrices
`TauLn`	list of log(Tau) matrices
`E_ZWSq`	expected values of the multiplication of the Z matrix with weight squared W matrix. See `E_ZWSq_update` function.

E_ZE_W	multiplication of the expected values of Z matrix with the expected values of W matrix. Seed E_ZE_W_update function.
Zeta	list of Zeta matrices
YTrg	list of data
YGauss	list of pseudo Y value matrices
PoisRateCstnt	small constant added for Poisson data to avoid errors

Value

the ELBO value for the current view/iteration

Examples

```
data("TL_param", package = "MOTL")

view <- "mRNA"
likelihoods <- c("mRNA" = "gaussian", "miRNA" = "gaussian",
                 "DNAme" = "gaussian", "SNV" = "bernoulli")
Tau <- TL_param$Tau
TauLn <- TL_param$TauLn
Zeta <- TL_param$Zeta
YTrg <- TL_param$YTrg
ZMu <- TL_param$ZMu
CenterTrg <- FALSE
PoisRateCstnt <- 0.0001
YGauss <- TL_param$YTrg

YGauss$mRNA <- YGauss_calculation(view = view,
                                  likelihoods = likelihoods,
                                  YTrg, Zeta, Tau, CenterTrg, PoisRateCstnt)

ZMuSq <- TL_param$ZMuSq
ZMu_0 <- TL_param$ZMu_0
Fctrzn_Lrn_W0 <- TL_param$Fctrzn_Lrn_W0
Fctrzn_Lrn_W <- TL_param$Fctrzn_Lrn_W
Fctrzn_Lrn_WSq <- TL_param$Fctrzn_Lrn_WSq

E_ZE_W <- list()
E_Z_SqE_W_Sq <- list()
E_ZSqE_WSq <- list()
E_ZWSq <- list()

E_ZE_W$mRNA <-
    E_ZE_W_update(view, ZMu_0, ZMu, Fctrzn_Lrn_W0, Fctrzn_Lrn_W)
E_Z_SqE_W_Sq$mRNA <-
    E_Z_SqE_W_Sq_update(view, ZMu_0, ZMu, Fctrzn_Lrn_W0, Fctrzn_Lrn_W)
E_ZSqE_WSq$mRNA <-
    E_ZSqE_WSq_update(view, ZMu_0, ZMuSq, Fctrzn_Lrn_W0, Fctrzn_Lrn_WSq)
E_ZWSq$mRNA <-
```

```
    E_ZWSq_update(view, E_ZE_W, ZMuSq, E_Z_SqE_W_Sq, E_ZSqE_WSq)

ELBO_L <-
    ELBO_calculation(view, likelihoods, Tau, TauLn, E_ZWSq, E_ZE_W,
                     Zeta, YTrg, YGauss, PoisRateCstnt)
```

GeoMeanFun *Calculate the Geometric mean of a vector*

Description

Calculate the Geometric mean of a vector

Usage

```
GeoMeanFun(x)
```

Arguments

x	vector of numeric values

Value

Geometric mean of vector x

Examples

```
x <- c(125,12,4545,7878,6777,454545,88979)
GeoMeans <- GeoMeanFun(x)
```

GeoMeans_Lrn_init *Retrieve the Geometric means calculated for the learning dataset during counts normalization*

Description

Retrieve the Geometric means calculated for the learning dataset during counts normalization

Usage

```
GeoMeans_Lrn_init(view, expdat_meta_Lrn, YTrgFtrs)
```

Arguments

view	current view data name
expdat_meta_Lrn	list of learning dataset factorization metadata
YTrgFtrs	feature names of the current view

Value

precalculated Geometric means of the learning dataset

Examples

```
data("Lrn", package = "MOTL")
data("Trg", package = "MOTL")

expdat_meta_Lrn <- Lrn$Fctrzn@data
YTrgFtrs <- Trg$Trg_meta$ftrs_mRNA

GeoMeans_Lrn <-
        GeoMeans_Lrn_init(view = "mRNA", expdat_meta_Lrn, YTrgFtrs)
```

---

initTransferLearningParamaters

Transfer learning parameters and data objects initialization

Description

The function performs the following steps:

1. Extract the factorized learning set weight intercepts `W0`
2. Extract the factorized learning set weights `W`
3. Extract the factorized learning set squared weights `Wsq`
4. Extract the learning set `Tau` and log(Tau) `TauLn` For each extracted parameter, common features between learning dataset and target dataset are kept. Then target data `YTrg`, `Tau` and `TauLn` are transposed.

Usage

```
initTransferLearningParamaters(YTrg, views, Fctrzn, likelihoods)
```

Arguments

YTrg	a named list of target dataset matrices. Names correspond to the defined views.
views	a vector of target dataset view names (e.g. `c("mRNA", "miRNA")`)
Fctrzn	the learning dataset factorization model object (from `MOFA`)
likelihoods	a named list of data types. The list can contain `gaussian`, `poisson` or `bernoulli` depending of the data type. Names are the view names.

Details

Each parameter are extracted from the `Fctrzn` model created using `MOFA2`.

`YTrg` matrices should have the same columns order.

Value

a list of initialized parameters for transfer learning

1. `YTrg` - the transposed named list of target data
2. `Fctrzn_Lrn_W0` - the Factorized learning set weight intercepts with same features as YTrg
3. `Fctrzn_Lrn_W` - Factorized learning set weights with same features as YTrg
4. `Fctrzn_Lrn_WSq` - Factorized learning set squared weights with same features as YTrg
5. `Tau` - the transposed Tau matrix with same features as YTrg
6. `TauLn` - the transposed log2(Tau) matrix with same features as YTrg

Examples

```
data("Lrn", package = "MOTL")
data("Trg", package = "MOTL")

views <- c("mRNA", "miRNA", "DNAme", "SNV")
likelihoods <- c("mRNA" = "gaussian", "miRNA" = "gaussian",
                 "DNAme" = "gaussian", "SNV" = "bernoulli")
Fctrzn <- Lrn$Fctrzn_init
YTrg <- Trg$YTrg_prep

TLparameter <-
    initTransferLearningParamaters(YTrg = YTrg,
                                   views = views,
                                   Fctrzn = Fctrzn,
                                   likelihoods = likelihoods)
```

---

intercepts_calculation

Intercepts calculation

Description

Calculate feature weight intercepts for a MOFA factorization based on MLE. These can be calculated for the learning dataset factorization and then for the factorization of the target dataset with transfer learning.

Usage

```
intercepts_calculation(expdat_meta, Fctrzn, FctrznDir, ExpDataDir, Seed)
```

Arguments

expdat_meta	the named list of learning dataset factorization metadata
Fctrzn	learning dataset factorization model object (from MOFA)
FctrznDir	the learning dataset factorization directory name
ExpDataDir	the learning dataset directory name
Seed	random seed number

Details

For Gaussian observed data, weight intercepts are the weight mean for each feature. For Poisson and Bernoulli observed data, weight intercepts are calculated using the maximum likelihood and the mle function.

Value

a file, named EstimatedIntercepts.rds and saved into FctrznDir directory.

Examples

```
#


data("Lrn", package = "MOTL")

expdat_meta <- Lrn$Lrn_meta
Fctrzn <- Lrn$Fctrzn
FctrznDir <- "FctrznDir"
ExpDataDir <- "ExpDataDir"
Seed <- 1234567

intercepts_calculation(expdat_meta,
                       Fctrzn,
                       FctrznDir,
                       ExpDataDir,
                       Seed)
```

Lrn *Learning dataset*

Description

Contains factorization model results calculated using MOFA2 and initialized for transfer learning: Fctrzn and Fctrzn_init.

Usage

```
data("Lrn")
```

Format

List of two MOFA objects: Fctrzn and Fctrzn_init.

MOFA2 object is a S4 class and contains the following main slots (the ones relevant and used for the transfer learning):

data input data used for the factorization analysis (mRNA, miRNA, DNAme and SNV)

samples_metadata sample metadata (i.e. sample names and group)

features_metadata features metadata (feature identifiers and views)

expectations expected values of the factors and the loadings

training_stats model training statistics

data_options data processing options

model_options model options

training_options model training options

dimensions dimensions of the model (e.g. M number of views, N number of samples, D number of features)

For more information about the structure of MOFA object, see the MOFA2 vignette.

Details

Fctrzn output of the factorization analysis of the learning dataset

Fctrzn_init output of the factorization analysis of the learning dataset and the initialized values for the transfer learning

---

mRNA_addVersion *Format mRNA features to match with learning dataset*

Description

Get mRNA ensembl ID version from learning dataset (e.g. ENSG00000122133.17) and attach to the corresponding mRNA ensembl ID in the target dataset. Feature names need to be similar between target dataset and learning dataset.

Usage

```
mRNA_addVersion(expdat, Lrndat)
```

Arguments

expdat	the mRNA matrix from the target dataset with genes in rows. Gene names should be in ensembl format and don't contain the version (e.g. ENSG00000122133). Rownames contain ensembl IDs and colnames sample names.
Lrndat	the mRNA W matrix from the learning dataset factorization with genes in rows. Gene names should be in ensembl format. Rownames contain ensembl IDs.

Value

the target mRNA matrix with versions attached

Examples

```
Lrn_names <-
        c("ENSG00000122133.17", "ENSG00000122194.9", "ENSG00000119411.1")
Lrn_views <- c("mRNA", "mRNA", "mRNA")
expdat_names <-
        c("ENSG00000122133", "ENSG00000122194", "ENSG00000119411")

Lrndat <- data.frame("view" = Lrn_views, row.names = Lrn_names)
expdat <- data.frame("sample1" = c(1, 52, 4), row.names = expdat_names)
expdat_prep <- mRNA_addVersion(expdat, Lrndat)
expdat
expdat_prep
```

---

preprocessCountsData *Preprocess counts data*

Description

Counts data (i.e. mRNA and miRNA) can be normalized and/or transformed.

Usage

```
preprocessCountsData(
  view,
  YTrg_list,
  normalization = FALSE,
  expdat_meta_Lrn,
  transformation = FALSE
)
```

Arguments

view	a data view name vector (i.e. mRNA or miRNA)
YTrg_list	a named list of target data. Names correspond to the defined views. The list contains matrices.
normalization	if FALSE, no normalization. If "LrnGeoMeans", normalization using the precalculated Geometric means. If "newGeoMeans", normalization using Geometric means from dataset. By default, it's set to FALSE.
expdat_meta_Lrn	the list of learning set factorization metadata
transformation	if FALSE, no transformation. If TRUE, log2 normalization.

Details

Normalization is performed using the countsNormalization function with pre-calculated or new calculated Geometric means.

Transformation is performed using the countsTransformation function with log2.

Value

Preprocessed counts data for the current view

Examples

```
data("Lrn", package = "MOTL")
data("Trg", package = "MOTL")

expdat_meta_Lrn <- Lrn$Fctrzn@data
YTrg_list <- Trg$YTrg_list

mRNA <- preprocessCountsData(view = "mRNA", YTrg_list = YTrg_list,
                             normalization = FALSE,
                             expdat_meta_Lrn = expdat_meta_Lrn,
                             transformation = FALSE)
```

TargetDataPrefiltering

Prepare the target data for a given view

Description

The function performs the following steps:

1. Remove the features with variance equal to zero
2. Harmonize features between the target data and the learning data. Only the shared features are kept.
3. Order columns according the order of samples (i.e. smpls)

Usage

```
TargetDataPrefiltering(view, YTrg_list, Fctrzn, smpls)
```

Arguments

view	current view data name (e.g. "mRNA", or "DNAme")
YTrg_list	a named list of target data. Names correspond to the views defined and the corresponding data are saved into matrix.
Fctrzn	the learning dataset factorization model object (from MOFA)
smpls	an ordered vector of sample names

Value

a matrix that contains the prepared data for the current view with the sample ordered.

Examples

```
data("Lrn", package = "MOTL")
data("Trg", package = "MOTL")

view <- "mRNA"
YTrg_list <- Trg$YTrg_prep
Fctrzn <- Lrn$Fctrzn
smpls <- colnames(YTrg_list$mRNA)

mRNA_prep <- TargetDataPrefiltering(view, YTrg_list, Fctrzn, smpls)
```

TargetDataPreparation *Target data preparation for transfer learning*

Description

The function follows these steps:

1. Prepare target data for each view
2. Normalize and/or transform counts data

Usage

```
TargetDataPreparation(
  views,
  YTrg_list,
  Fctrzn,
  smpls,
  expdat_meta_Lrn,
  normalization = FALSE,
  transformation = FALSE
)
```

Arguments

views	a list of target data views (e.g. c("mRNA", "miRNA"))
YTrg_list	a named list of target set data. Names correspond to the defined views. The list contains matrices.
Fctrzn	the learning factorization model object (from MOFA)
smpls	a vector of sample names (i.e. column names of the YTrg_list)

`expdat_meta_Lrn`
: the list of learning set factorization metadata

`normalization`
: if FALSE, no normalization. If "LrnGeoMeans", normalization using the learning Geometric means. If "newGeoMeans", Geometric means are calculated using the input data. By default it's set to FALSE.

`transformation`
: if FALSE, no transformation. If TRUE, log2 transformation of counts data. By default it's set to FALSE.

Details

Preparation of data consists on

- removing features with variance equal to zero
- features harmonization between target and learning data
- column ordering between views.

Preparation is perform using the `TargetDataPrefiltering` function. See the documentation for more details.

It could be possible to normalize and or transform counts data using the `preprocessCountsData` function. Normalization can be done using Geometric means (from learning or target dataset) and transformation is a log2 transformation of the counts.

Value

a list of prepared target data

Examples

```
data("Lrn", package = "MOTL")
data("Trg", package = "MOTL")

YTrg_list <- Trg$YTrg_prep
Fctrzn <- Lrn$Fctrzn
smpls <- colnames(YTrg_list$mRNA)
expdat_meta_Lrn <- Lrn$Lrn_meta


YTrg_prep <- TargetDataPreparation(views = c("mRNA", "miRNA", "DNAme"),
                                   YTrg_list = YTrg_list,
                                   Fctrzn = Fctrzn, smpls = smpls,
                                   expdat_meta_Lrn = expdat_meta_Lrn,
                                   normalization = FALSE,
                                   transformation = FALSE)
```

`Tau_calculation` *Update Tau values for the current view*

Description

Tau values are updated only for bernoulli data.

Usage

```
Tau_calculation(view, likelihoods, Zeta, Tau)
```

Arguments

`view`	a character of current view name data
`likelihoods`	a named list of data types. The list can contain `gaussian`, `poisson` or `bernoulli` depending of the data type. Names are the view names.
`Zeta`	list of Zeta matrix for the current view
`Tau`	list of tau matrices

Details

Tau values are updated using the following equation:

$Tau = (\frac{1}{2}) * (\frac{1}{Zeta[[view]]}) * tanh(\frac{Zeta[[view]]}{2})$

Value

(updated) Tau matrix for the current view data

Examples

```
data("TL_param", package = "MOTL")

view <- "mRNA"
likelihoods <- c("mRNA" = "gaussian", "miRNA" = "gaussian",
                 "DNAme" = "gaussian", "SNV" = "bernoulli")
Zeta <- TL_param$Zeta
Tau <- TL_param$Tau


Tau <- Tau_calculation(view = view,
                       likelihoods = likelihoods,
                       Zeta = Zeta, Tau = Tau)
```

Tau_init *Initialization of the Tau values for each view*

Description

Extract the Tau matrix from the MOFA object `Fctrzn` for each view. More explanation about Tau.

Usage

```
Tau_init(viewsLrn, Fctrzn, InputModel)
```

Arguments

`viewsLrn`	the list of learning data views. For TCGA learning data it will be `c("mRNA", "miRNA", "DNAme", "SNV")`).
`Fctrzn`	the learning dataset factorization from `MOFA2`.
`InputModel`	the factorization model object of learning set `MOFA2`

Value

a named list of Tau matrices. Names correspond to the view names.

Examples

```
library("MOFA2")

viewsLrn <- c("mRNA", "miRNA", "DNAme", "SNV")
InputModel <- "Model.hdf5"
Fctrzn <- load_model(file = InputModel)

Tau_list <- Tau_init(viewsLrn = viewsLrn,
                     Fctrzn = Fctrzn,
                     InputModel = InputModel)
```

TauLn_calculation *Initialization of the log(Tau) values*

Description

Two ways to initialize the log(Tau) values:

1. log transformation of the expected Tau (already init in the `Fctrzn`) variable
2. extract values from a .csv file that saved in `LrnFctrnDir` directory Tau is initialized only for gaussian data.

Usage

```
TauLn_calculation(view, likelihoodsLrn, Fctrzn, LrnFctrnDir, LrnSimple = TRUE)
```

Arguments

view	a character of current view name data (e.g. "mRNA")
likelihoodsLrn	a named list of data types. The list can contain gaussian, poisson or bernoulli depending of the data type. Names are the view names.
Fctrzn	learning dataset factorization model object (from MOFA)
LrnFctrnDir	directory where log(Tau) values are saved. Files should be named like "TauLn_mRNA.csv".
LrnSimple	if TRUE, initialization uses the Tau values. If FALSE, imports values from a .csv file. By default is set to "TRUE".

Details

For gaussian data, $TauLn < -log(Tau)$

Value

the log(Tau) matrix for the current view

Examples

```
library("MOFA2")

data("Lrn", package = "MOTL")

Fctrzn <- Lrn$Fctrzn_init
likelihoodsLrn <- get_default_model_options(Fctrzn)$likelihoods

TauLn_mRNA = TauLn_calculation(view = "mRNA",
                               likelihoodsLrn = likelihoodsLrn,
                               Fctrzn = Fctrzn,
                               LrnSimple = TRUE,
                               LrnFctrnDir = LrnFctrnDir)
```

TCGATargetDataPrefiltering

Filter out TCGA target subset data according variance

Description

This function performs a prefiltering analysis through the steps:

1. Extract data for the given sample names (brcds_SS)
2. Remove features with variance equal to zero
3. Match features between target data set and learning data set

Usage

```
TCGATargetDataPrefiltering(view, brcds_SS, SS, YTrgFull, Fctrzn)
```

Arguments

view	a character of current view name data
brcds_SS	a list of sample names for each view. The list is named according views (e.g. "brcds_mRNA_SS") and contains list of dataframes for each view. Each dataframe contains at least one column named "brcds" (the one used).
SS	current subset number
YTrgFull	a named list of target set data. Names correspond to the defined views. The list contains SummarizedExperiment for miRNA, mRNA and DNAme and matrix for SNV.
Fctrzn	learning factorization model object (from MOFA)

Details

The function return a pre-filtered target dataframe for the current view.

Value

the subset data for current view and SS number

Examples

```
# In the paper, several target datasets were created as subset of the
# reference dataset R. This function was used to generate them
# automatically.
# If you are not doing the paper analysis, you can create the brcds_SS
# using the following command line. Replace the "nameX" with the sample
# names of your data.
# You can as much as you want add dataframe on each view.

brcds_SS_ex <-
        list("brcds_mRNA_SS" =
            list(data.frame("brcds" = c("name01", "name02"))),
        "brcds_miRNA_SS" =
            list(data.frame("brcds" = c("name10", "name11"))))

# See the doc to create the input parameter

data("Trg", package = "MOTL")
data("Lrn", package = "MOTL")

brcds_SS <- Trg$brcds_SS
YTrg_list <- Trg$YTrg_list
Lrn_Fctrzn <- Lrn$Fctrzn


expdat_mRNA <- TCGATargetDataPrefiltering(view = "mRNA",
```

```
brcds_SS = brcds_SS, SS = 1, YTrgFull = YTrg_list, Fctrzn = Lrn_Fctrzn)
expdat_mRNA[c(1:5), c(1:5)]
```

`TCGATargetDataPreparation` *Prepare TCGA target dataset for transfer learning*

Description

This function follows these steps:

1. Filter out features according variance
2. Reshape data into matrices
3. Order samples to have the same columns order between different views
4. Normalize and/or transform counts data

Usage

```
TCGATargetDataPreparation(
  views,
  YTrgFull,
  brcds_SS,
  SS,
  Fctrzn,
  smpls,
  normalization = FALSE,
  expdat_meta_Lrn,
  transformation = TRUE
)
```

Arguments

`views`	a list of target data views (e.g. `c("mRNA", "miRNA")`)
`YTrgFull`	a named list of target set data. Names correspond to the defined views. The list contains `SummarizedExperiment` for miRNA, mRNA and DNAme and `matrix` for SNV.
`brcds_SS`	a list of sample names for each view. The list is named according views (e.g. "brcds_mRNA_SS") and contains list of dataframes for each view. Each dataframe contains at least one column named "brcds" (the one used).
`SS`	current subset number
`Fctrzn`	the learning factorization model object (from `MOFA`)
`smpls`	a vector of sample names (i.e. column names of the `YTrgFull`)

`normalization` if FALSE, no normalization. If "LrnGeoMeans", normalization using the learning Geometric means. If "newGeoMeans", normalization with target Geometric means. By default it's set to "FALSE".

`expdat_meta_Lrn`
the list of learning set factorization metadata

`transformation` if FALSE, no transformation. If TRUE, log2 transformation of counts data. By default it's set to "FALSE"

Details

First, samples included in the brcds_SS list are selected. Then features with variance equal to zero are removed. They are also remove if they are not retrieved in the learning dataset. These steps are performed using `TCGATargetDataPrefiltering` function.

The mRNA, miRNA et DNAme data are stored into `SummarizedExperiment` object. For the next step, data have to be stored into a `matrix`. SNV data are already a matrix. Then, samples are ordered in the same way between views.

Finally, counts data (e.g. mRNA and miRNA) can be normalized and/or transformed using `preprocessCountsData` function.

- if `normalization = FALSE`: counts data are not normalized
- if `normalization = "LrnGeoMeans"`: counts data are normalized using the learning dataset Geometric means calculated
- if `normalization = "newGeoMeans"`: counts data are normalized using the geometric means calculated on the target dataset.

Normalization is performed in the `countsNormalization` function using `estimateSizeFactors` from `DESeq2` package. And transformation is perform using `countsTransformation` with a log2 transformation.

Look `GeoMeans_Lrn_init` and `GeoMeanFun` to see how learning Geometric means are calculated.

Value

list of prepared subset data for the current subset number

Examples

```
# see to create input data

data("Trg", package = "MOTL")
data("Lrn", package = "MOTL")

views <- c("mRNA", "miRNA", "DNAme", "SNV")
YTrgFull <- Trg$YTrg_list
brcds_SS <- Trg$brcds_SS
SS <- 1
Fctrzn <- Lrn$Fctrzn
smpls <- colnames(YTrgFull$mRNA)
expdat_meta_Lrn <- Lrn$Lrn_meta
```

```
YTrg_prep <- TCGATargetDataPreparation(views,
                                       YTrgFull,
                                       brcds_SS,
                                       SS,
                                       Fctrzn,
                                       smpls,
                                       normalization = FALSE,
                                       expdat_meta_Lrn,
                                       transformation = FALSE)
YTrg_prep$mRNA[c(1:5), c(1:5)]
YTrg_prep$DNAme[c(1:5), c(1:5)]

# In the paper, several target datasets were created as subset of the
# reference dataset R. This function was used to generate them
# automatically.
# If you are not doing the paper analysis, you can create the brcds_SS
# using the following command line. Replace the "nameX" with the sample
# names of your data.
# You can as much as you want add dataframe on each view.

brcds_SS_ex <-
        list("brcds_mRNA_SS" =
            list(data.frame("brcds" = c("name01", "name02"))),
        "brcds_miRNA_SS" =
            list(data.frame("brcds" = c("name10", "name11"))))

# The SS parameter corresponds to the index of the subset you want to
# prepare for a specific view. It's generated automatically if you used
# the workflow describe in the github paper
# \link{https://github.com/david-hirst/MOTL/blob/main/TCGAStudy/00_TCGAstudy_ReadMe.md}
```

TL_param *Transfer learning parameters*

Description

Contains a list of input variables used for the transfer learning.

Usage

```
data("TL_param")
```

Format

YTrg list of the prepared input target dataset (mRNA, miRNA, DNAme, SNV). Samples are in columns and features are in rows.

Fctrzn_Lrn_W0 list of 4 variables (mRNA, miRNA, DNAme and SNV). Each contains a W0 vector named with the corresponding feature names.

Fctrzn_Lrn_W list of 4 `data.frame` (mRNA, miRNA, DNAme and SNV). Each contains factors in columns and features in rows.

Fctrzn_Lrn_WSq list of 4 `data.frame` (mRNA, miRNA, DNAme and SNV). Each contains factors in columns and features in rows.

Tau list of 4 `data.frame` (mRNA, miRNA, DNAme and SNV). Each contains features in columns.

TauLn list of 4 `data.frame` (mRNA, miRNA, DNAme and SNV). Each contains features in columns.

ZVar `data.frame` with factors in columns.

ZMu `data.frame` with factors in columns and samples in rows.

ZMu_0 vector of numerics

ZMuSq `data.frame` with factors in columns.

`transferLearning_function`

Transfer Learning with Variational Inference

Description

This function performs multi-omics matrix factorization with transfer learning. The target dataset is factorized using the latent factor values inferred from the previous factorization of a learning dataset.

Usage

```
transferLearning_function(
  TL_param,
  MaxIterations,
  MinIterations,
  minFactors,
  views,
  likelihoods,
  Fctrzn,
  StartDropFactor,
  FreqDropFactor,
  StartELBO,
  FreqELBO,
  DropFactorTH,
  ConvergenceIts,
  ConvergenceTH,
  CenterTrg,
  PoisRateCstnt = 1e-04,
  ss_start_time = NULL,
  outputDir = "./"
)
```

Arguments

TL_param	a named list of initialized parameters and data objects for transfer learning. It contains target dataset, weigths and scores matrices from matrix factorization of the learning dataset calculated using MOFA. See the detail section for more informations.
MaxIterations	the maximum number of iterations for the matrix factorization convergence. After this number, the factorization is stopped.
MinIterations	the minimum number of iterations for the matrix factorization convergence. Before this number, even if the function converges, the factorization is not stopped.
minFactors	the minimum number of factors to retain
views	a named vector of the target dataset. It should contains the same names used for inside the learning dataset.
likelihoods	a named vector of the target dataset types. It can contain gaussian, poisson or bernoulli depending of the data type. Names are the view names.
Fctrzn	the learning factorization model object (from MOFA). Creating using the MOFA_functions.py python script.
StartDropFactor	number after which iteration to start dropping factors
FreqDropFactor	number that corresponds to how often to check whether to drop factors
StartELBO	number after which iteration to start checking ELBO on
FreqELBO	number that correspond to how often to assess the ELBO
DropFactorTH	threshold number to drop or not factors. If factor with lowest maximum variance explained is below this threshold, it's dropped.
ConvergenceIts	number of consecutive iterations that change in ELBO is below threshold before convergence
ConvergenceTH	threshold number for change in ELBO for checking convergence
CenterTrg	if TRUE, center the target dataset during processing, if FALSE, leave target dataset uncentered and use estimated learning dataset intercepts.
PoisRateCstnt	amount number added to the Poisson rate function to avoid error. By default is equal to 0.0001. It's used in the pseudo gaussian values calculation YGauss, see YGauss_calculation and ELBO calculation, see ELBO_calculation.
ss_start_time	time recorded before the preprocessing step starts. Generated using Sys.time function. By default is NULL.
outputDir	output directory name where to save results. By default results are saved in the current directory.

Details

This function is called after target dataset is prepared (using TargetDataPreparation) and parameters initialized (using initTransferLearningParamaters).

TL_param is a named list of the initialized parameters and data objects for transfer learning. It contains :

1. YTrg: a named list of matrices. Each matrix corresponds to the target dataset.
2. Fctrzn_Lrn_W0: a named list of vectors. Each vector contains the features mean weight matrix calculated for the learning dataset using MOFA.
3. Fctrzn_Lrn_W: a named list of matrices. Each matrix contains the weights matrix calculated for the learning dataset using MOFA.
4. Fctrzn_Lrn_WSq: a named list of matrices. Each matrix contains the squared weights matrix calculated for the learning dataset using MOFA.
5. Tau: a named list of matrices. Each matrix contains the Tau values matrix calculated for the learning dataset using MOFA.
6. TauLn: a named list of matrices. Each matrix contains the TauLn values matrix calculated for the learning dataset using MOFA.

Names of each list should be identical (e.g. `c("mRNA", "miRNA", "DNAme", "SNV")`) and so each element corresponds to each omic data.

To create the TL_param variable, see the initTransferLearningParamaters function.

Value

a named list of results. It contains

1. YTrgSS list of matrices of target dataset
2. YGauss list of matrices of pseudo gaussian target dataset
3. ZMu_0 list of ZMu intercepts matrices
4. ZMu list of ZMu
5. Fctrzn_Lrn_W0 list of learning features mean weight matrix
6. Fctrzn_Lrn_W list of learning weights matrix
7. ELBO numeric value of ELBO
8. VarExpl variance explained by each target dataset
9. ss_start_time time when start the analysis (i.e. before the preprocessing step)
10. ss_fit_start_time time when start the transfer learning analysis
11. ss_end_time time when finish the transfer learning.

Results are also saved into TL_data.rds file.

Examples

```
data("TL_param", package = "MOTL")

ss_start_time <- Sys.time()
minFactors <- 13
StartDropFactor <- 1
FreqDropFactor <- 1
StartELBO <- 1
FreqELBO <- 5
```

```
DropFactorTH <- 0.01
MaxIterations <- 10
MinIterations <- 2
ConvergenceIts <- 2
ConvergenceTH <- 0.0005
PoisRateCstnt <- 0.0001

TL_data <- transferLearning_function(TL_param, MaxIterations,
                                     MinIterations, minFactors,
                                     views, likelihoods, Fctrzn,
                                     StartDropFactor, FreqDropFactor,
                                     StartELBO, FreqELBO,
                                     DropFactorTH, ConvergenceIts,
                                     ConvergenceTH,
                                     CenterTrg, PoisRateCstnt = 0.0001,
                                     ss_start_time = NULL,
                                     outputDir = "./")
```

`Trg` *Target datasets*

Description

Contains a list of target datasets used in different step in the transfer learning workflow and there associated metadata.

Usage

```
data("Trg")
```

Format

YTrg_list list of the target dataset - Samples are in columns and features are in rows.

- `mRNA`**:** data stored into a `SummarizedExperiment` object
- `miRNA`**:** data stored into a `SummarizedExperiment` object
- `DNAme`**:** data stored into a `SummarizedExperiment` object
- `SNV`**:** data stored into a `matrix`

Trg_meta list of metadata:

- **Five** `character` smpls, ftrs_mRNA, ftrs_miRNA, ftrs_DNAme, ftrs_SNV
- **Four** `data.frame` with three variables (brcds, submittor, prjct) - brcds_mRNA, brcds_miRNA, brcds_DNAme, brcds_SNV
- **Six** `integer` Seed, ElbowK_Total, ElbowK_mRNA, ElbowK_miRNA, ElbowK_DNAme, ElbowK_SNV
- **One** `logical` if_vst

Four `numeric` PCVarPrcnt_mRNA, PCVarPrcnt_miRNA, PCVarPrcnt_DNAme, PCVarPrcnt_SNV

brcds_SS a list of 4 list of `data.frame` with the sample names.

YTrg_prep list of the prepared input target dataset (mRNA, miRNA, DNAme, SNV). Samples are in columns and features are in rows.

`VarExplFun` *Calculate the variance explained by each factor for each view*

Description

Calculate the variance explained by each factor for each view

Usage

```
VarExplFun(views, YGauss, ZMu_0, Fctrzn_Lrn_W0, ZMu, Fctrzn_Lrn_W)
```

Arguments

`views`	list of view names
`YGauss`	list of pseudo Y value matrices
`ZMu_0`	vector of coefficients for weight intercepts
`Fctrzn_Lrn_W0`	list of factorized learning set weight intercept matrices
`ZMu`	matrix of Z values
`Fctrzn_Lrn_W`	list of factorized learning set weight matrices

Value

variance explained matrix

Examples

```
VarExpl <-
    VarExplFun(views, YGauss, ZMu_0, Fctrzn_Lrn_W0, ZMu, Fctrzn_Lrn_W)
```

W0_calculation *Initialization of feature weight intercept values*

Description

This function loads or calculates the weight intercept values.

Usage

```
W0_calculation(view, CenterTrg, Fctrzn, LrnFctrnDir)
```

Arguments

view	a character of current view name data
CenterTrg	if FALSE, use the estimated feature weight intercept from the EstimatedIntercepts.rds file. If TRUE, don't use the estimated feature weight intercept.
Fctrzn	learning dataset factorization model object (from MOFA)
LrnFctrnDir	directory where the extimated intercepts file is.

Details

The weight intercept values can be load from the EstimatedIntercepts.rds file. This file can be created using the intercepts_calculation function.

The weight intercept values can also be initialized using the weight matrix. The weight matrix is set to zero.

Value

a feature weight intercept values matrix for the current data

Examples

```
data("Lrn", package = "MOTL")

Fctrzn <- Lrn$Fctrzn

W0_mRNA = W0_calculation(view = "mRNA",
                         CenterTrg = TRUE,
                         Fctrzn = Fctrzn,
                         LrnFctrnDir = LrnFctrnDir)
```

WSq_calculation *Initialization of the squared weight values*

Description

This function load or calculate the squares weight values.

Usage

```
WSq_calculation(view, Fctrzn, LrnFctrnDir, LrnSimple = TRUE)
```

Arguments

view	a character of current view name data
Fctrzn	learning dataset factorization model object (from MOFA)
LrnFctrnDir	directory where WSq values are saved
LrnSimple	if TRUE, calculates the squared weight values WSq. If FALSE, imports values from a .csv file. By default is set to "TRUE". $E[W^2]$ using the weight values W $E[W]$. If FALSE, load squared weight values from a file. By default, it's set to TRUE.

Details

The squared weight values can be load from a .csv file. This file can be created during the factorization of the learning data. See the documentation to learn how to create this file. The file name should follow this format: WSq_mRNA.csv.

The squared weight valued can also be calculated using the weight values calculated during the factorization of the learning data. These values are saved in the Fctrzn variable. See the documentation.

Value

the squared weight matrix for the current view

Examples

```
library("MOFA2")

data("Lrn", package = "MOTL")

Fctrzn <- Lrn$Fctrzn
likelihoodsLrn <- get_default_model_options(Fctrzn)$likelihoods

WSq_mRNA = WSq_calculation(view = "mRNA",
                           Fctrzn = Fctrzn,
                           LrnFctrnDir = LrnFctrnDir,
                           LrnSimple = TRUE)
```

YGauss_calculation *Initialize or update pseudo Y values (YGauss)*

Description

For gaussian data, Y values (observed data) are centered (if CenterTrg = TRUE) and will not change. For non gaussian, Y values are transformed and change after each update of Z matrix. The Y pseudo values are centered at each step if CenterTrg = TRUE. For gaussian data this is done for each iteration It>=0, for others it is done for each iteration, exept the first one It>0.

Usage

```
YGauss_calculation(
  view,
  likelihoods,
  YTrg,
  Zeta,
  Tau,
  CenterTrg,
  PoisRateCstnt
)
```

Arguments

view	a character of current view name data
likelihoods	a named list of data types. The list can contain gaussian, poisson or bernoulli depending of the data type. Names are the view names.
YTrg	current data matrix
Zeta	list of Zeta matrices
Tau	list of Tau matrices
CenterTrg	if FALSE, use the estimated feature weight intercept from the EstimatedIntercepts.rds file. If TRUE, use the feature weight means.
PoisRateCstnt	small constant added when transforming Poisson data to avoid errors

Value

pseudo Y values for the current view

Examples

```
data("TL_param", package = "MOTL")

view <- "mRNA"
likelihoods <- c("mRNA" = "gaussian", "miRNA" = "gaussian",
                 "DNAme" = "gaussian", "SNV" = "bernoulli")
CenterTrg <- FALSE
YTrg <- TL_param$YTrg
Zeta <- TL_param$Zeta
PoisRateCstnt <- 0.0001

YGauss <- YGauss_calculation(view = view,
                             likelihoods = likelihoods,
                             YTrg, Zeta, Tau, CenterTrg, PoisRateCstnt)
```

`Zeta_calculation` *Calculate the Zeta matrix for the current data view*

Description

For the current data view, calculate the Zeta matrix `Zeta`.

Usage

```
Zeta_calculation(view, likelihoods, E_ZWSq, E_ZE_W)
```

Arguments

`view`	a character of current view name data (e.g. `mRNA`)
`likelihoods`	a named list of data types. The list can contain `gaussian`, `poisson` or `bernoulli` depending of the data type. Names are the view names.
`E_ZWSq`	expected values of the multiplication of the Z matrix with weight squared W matrix.
`E_ZE_W`	multiplication of the expected values of Z matrix with the expected values of W matrix

Details

For bernoulli data, Zeta is calculated using the expected values of Z matrix and W squared matrix `E_ZWSq`**.** `E_ZWSq` **is**
$Zeta_{nd} = \sqrt{(E[(\sum_k z_{n,k} w_{d,k})^2])}$

For other data type, Zeta is calculated using the expected values of Z matrix and the expected values of W matrix `E_Z`
$Zeta_{nd} = E[\sum_k z_{n,k} w_{d,k}]$, so $Zeta = ZMu\% * \%t(W)$

Value

Zeta matrix for the current data view

Examples

```
data("TL_param", package = "MOTL")

view <- "mRNA"
ZMuSq <- TL_param$ZMuSq
ZMu <- TL_param$ZMu
ZMu_0 <- TL_param$ZMu_0
Fctrzn_Lrn_W0 <- TL_param$Fctrzn_Lrn_W0
Fctrzn_Lrn_W <- TL_param$Fctrzn_Lrn_W
Fctrzn_Lrn_WSq <- TL_param$Fctrzn_Lrn_WSq
likelihoods <- c("mRNA" = "gaussian", "miRNA" = "gaussian",
                 "DNAme" = "gaussian", "SNV" = "bernoulli")

E_ZE_W <- list()
E_Z_SqE_W_Sq <- list()
E_ZSqE_WSq <- list()
E_ZWSq <- list()

E_ZE_W$mRNA <-
    E_ZE_W_update(view, ZMu_0, ZMu, Fctrzn_Lrn_W0, Fctrzn_Lrn_W)
E_Z_SqE_W_Sq$mRNA <-
    E_Z_SqE_W_Sq_update(view, ZMu_0, ZMu, Fctrzn_Lrn_W0, Fctrzn_Lrn_W)
E_ZSqE_WSq$mRNA <-
    E_ZSqE_WSq_update(view, ZMu_0, ZMuSq, Fctrzn_Lrn_W0, Fctrzn_Lrn_WSq)
E_ZWSq$mRNA <-
    E_ZWSq_update(view, E_ZE_W, ZMuSq, E_Z_SqE_W_Sq, E_ZSqE_WSq)

Zeta <- Zeta_calculation(view = "mRNA",
                         likelihoods = likelihoods,
                         E_ZWSq = E_ZWSq,
                         E_ZE_W = E_ZE_W)
```

ZMu_calculation *Z matrix* ZMu *calculation for the current data*

Description

Z matrix ZMu calculation for the current data

Usage

```
ZMu_calculation(view, k, Fctrzn_Lrn_W, Fctrzn_Lrn_W0, Tau, ZMu_0, ZMu, YGauss)
```

Arguments

view	a character of current view name data
k	feature index in the current data
Fctrzn_Lrn_W	list of factorized learning set weight matrices

Fctrzn_Lrn_W0	list of factorized learning set weight intercept matrices
Tau	list of Tau matrices
ZMu_0	vector of coefficients for weight intercepts
ZMu	matrix of Z values
YGauss	list of pseudo Y value matrices

Value

ZMu values for the current view (Z matrix)

Examples

```
data("TL_param", package = "MOTL")

k <- 10
view <- "mRNA"
Fctrzn_Lrn_W <- TL_param$Fctrzn_Lrn_W
Fctrzn_Lrn_W0 <- TL_param$Fctrzn_Lrn_W0
Tau <- TL_param$Tau
ZMu_0 <- TL_param$ZMu_0
ZMu <- TL_param$ZMu
YGauss <- TL_param$YTrg
ZMu <- TL_param$ZMu

ZMu <- ZMu_calculation(view,
                       k,
                       Fctrzn_Lrn_W,
                       Fctrzn_Lrn_W0,
                       Tau,
                       ZMu_0,
                       ZMu,
                       YGauss)
```

ZVar_calculation *Calculation of the Z variances for the current data*

Description

Z variances is calculation using initialized or updated Tau values and the squared weight values WSq values based on the appendix of the MOFA2 paper and Github code

Usage

```
ZVar_calculation(view, Tau, Fctrzn_Lrn_WSq)
```

Arguments

view	a character of current view name data
Tau	list of Tau matrices
Fctrzn_Lrn_WSq	Factorized learning set squared weights

Value

calculated Z variances matrix for the current data

Examples

```
data("TL_param", package = "MOTL")

Tau <- TL_param$Tau
Fctrzn_Lrn_WSq <- TL_param$Fctrzn_Lrn_WSq

ZVar <- ZVar_calculation(view = "mRNA", Tau, Fctrzn_Lrn_WSq)
```

Index