ExposomeX handbook (v1.0.0)

ExposomeX: An integrated platform to expedite the discovery of "Exposure-Biology-Disease" nexus

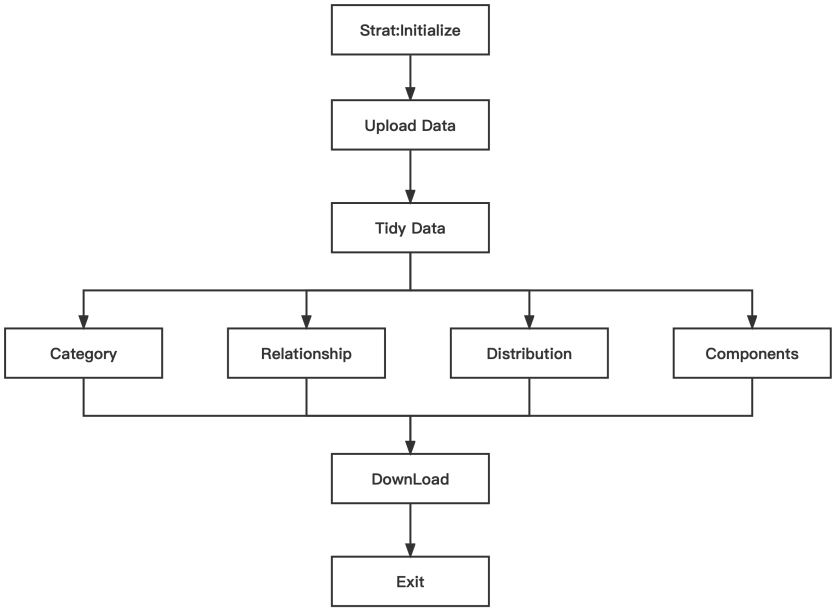
Bin Wang, Mingiang Fang, and ExposomeX team

2024-6-1

Author list

- Bin Wang, School of Public Health, Peking University
- Mingliang Fang, Department of Environmental Science and Engineering, Fudan University
- Changxin Lan, School of Public Health, Peking University
- Guohuan Zhang, School of Public Health, Peking University
- Mengyuan Ren, School of Public Health, Peking University
- Tianxiang Wu, School of Public Health, Peking University
- Ning Gao, School of Public Health, Peking University
- Weinan Lin, School of Public Health, Peking University
- Yanqiu Feng, School of Public Health, Peking University
- Yuting Wang, School of Public Health, Peking University

Contents



1 Introduction of "ExposomeX" platform

1.1 Overview and perspective

Exposome has become the hotspot of next-generation health studies. To date, there is no available effective platform to standardize the analysis of exposomic data. In the present study, we aim to propose one new framework of exposomic analysis and build up one novel integrated platform "ExposomeX" to expediate the discovery of the "Exposure-Biology-Disease" nexus. We have developed six major functions including exposome database search (EDB), biological link (EBIO), statistical analysis (ESTAT), mass spectrometry data processing (EMS), meta-analysis (EMETA), and data visualization (EVIZ). Using ExposomeX, we can effectively analyze the multiple-dimensional exposomic data and investigate the "Exposure-Biology-Disease" nexus by exploring the association strength, understanding the statistical and biological mechanisms, enhancing the prediction performance, and automatically conducting meta-analysis based on high-quality literature databases and publications. The performance of ExposomeX has been well validated by re-analyzing three typical multi-omic datasets. Additionally, ExposomeX can efficiently help discover new associations, as well as the relevant in-depth biological pathways via protein, protein-protein interaction, gene, and gene ontology network analysis. User can also use these packages by web-interaction, see: http://www.exposomex.cn. In sum, we have proposed a novel framework for standardized exposomic analysis, which can be accessed using both R and online interactive platform. All the modules will keep updating. Please the user tutorials at: http://www.exposomex.cn/#/toturial.

This platform is founded by Bin Wang (Peking University, binwang@pku.edu.cn) and Mingliang Fang (Fudan University, mlfang@fudan.edu.cn). Great thanks should be given to the core development members for their significant contributions to the individual 14 R packages including Bin Wang (StatTidy, EDB, StatCros, StatMix, StatPanel, and StatIP), Mingliang Fang (EMS and EBIO), Yanqiu Feng (StatDesc), Ning Gao (EVIZ), Guohuan Zhang and Yuting Wang (StatMO), Mengyuan Ren (StatMedt), Changxin Lan (StatSurv), and Weinan Lin (EMETA). Special credits to Changxin Lan for making the codes into R packages of all modules, Ning Gao for providing help to most of the visualization funcitons, Weinan Lin and Tianxiang Wu for tidying the IDs of chemicals, proteins, and diseases. The other contributors are acknowledged at http://www.exposomex.cn/#/about. Welcome to contact ExposomeX development team by E-mail: exposomex@gmail.com

1.2 Methodology

1.3 Basics

Initialize package

Make sure that the required packages is already installed.

```
# The following two packages should be installed in advance
# devtools::install_github("ExposomeX/exposomex", force = TRUE)
# install.packages("pacman")
pacman::p_load(
  "exposomex",
  "tidyverse"
  )
```

At first, you need to initialize the calculation environment using a series of initialization functions, e.g., InitStatCros, InitStatMo, InitStatTidy, InitEVIZ, InitEBIO, etc. ere, we use the module "StatPanel" for panel data analysis for example. The detailed information about the functions and returned value will be introduced in the following chapters.

```
res = InitStatPanel()
res
```

```
## <eSet>
##   Public:
##     AddCommand: function (x)
##     AddLog: function (x)
##     clone: function (deep = FALSE)
##     EpiDesign: NULL
##     ExcecutionLog: Complete initializing the ExpoStat module.2024.06.21 10. ...
##     Expo: list
##     FileDirIn: NULL
##     FileDirOut: /home/ubuntu/ExposomeX/R_Exposome_1.0/output_104420.6226 ...
##     PID: 104420.622618CLPXLN
##     RCommandLog: eSet <- InitPanel()
##     VarsDel: NULL
```

Here, we can see that the returned value "res" is an R6 object. It contains an unique program ID of res$PID (e.g., "100737GJMWJA"), which is random generated by the system. Users need use it in the following step for further data process.

Upload data

```
res1 <- LoadStatPanel(PID = res$PID,
                      UseExample = "example#1")

res1$Expo$Voca %>%
  dplyr::slice(1:20)
```

```
## # A tibble: 20 x 5
##    SerialNo SerialNo_Raw FullName                              GroupName Type
##    <chr>    <chr>        <chr>                                 <chr>     <chr>
##  1 Y1       Y1           TL_cat2                               Outcome   cate
##  2 Y2       Y2           TL                                    Outcome   cont
##  3 C1       C1           cMCs_NFkB_unstim                      immunome  cont
##  4 C2       C2           CCR2poscMCs_STAT3_unstim              immunome  cont
##  5 C3       C3           MDSCs_STAT3_unstim                    immunome  cont
##  6 X1       X1           DCs_S6_unstim                         immunome  cont
##  7 X2       X2           DCs_MAPKAPK2_unstim                   immunome  cont
##  8 X3       X3           DCs_NFkB_unstim                       immunome  cont
##  9 X4       X4           CCR2negncMCs_NFkB_unstim              immunome  cont
## 10 X5       X5           NK_STAT5_unstim                       immunome  cont
## 11 X6       X6           CD69negCD56loCD16negNK_S6_unstim      immunome  cont
## 12 X7       X7           CD69posCD56loCD16negNK_S6_unstim      immunome  cont
## 13 X8       X8           CD4Tcells_ERK_unstim                  immunome  cont
## 14 X9       X9           CD4Tcells_STAT6_unstim                immunome  cont
## 15 X10      X10          CD4posTnaive_MAPKAPK2_unstim          immunome  cont
## 16 X11      X11          CCR5posCCR2posCD4Tem_ERK_unstim       immunome  cont
## 17 X12      X12          Th1_ERK_unstim                        immunome  cont
## 18 X13      X13          CD8Tmem_MAPKAPK2_unstim               immunome  cont
## 19 X14      X14          CD8Tem_S6_unstim                      immunome  cont
## 20 X15      X15          CD8Tem_IkB_unstim                     immunome  cont
```

```
res1$Expo$Data %>%
  dplyr::select(SampleID:X2) %>%
  dplyr::slice(1:20)
```

```
## # A tibble: 20 x 9
##    SampleID SubjectID    Y1    Y2     C1    C2    C3    X1      X2
##    <chr>        <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>   <dbl>
##  1 Tr1              1     1   106 0.0473 0.623 0.593 0.234 0
##  2 Tr2              1     0    56 0.0392 0.276 0.186 0.189 0
##  3 Tr3              1     0    42 0.0526 0.423 0.343 0.225 0
##  4 Tr4             16     1    66 0.0860 0.557 0.608 0.290 0
##  5 Tr5             16     0    33 0.109  0.405 0.385 0.322 0
##  6 Tr6             16     0    13 0.0886 0.433 0.458 0.307 0.0103
##  7 Tr7             28     1    68 0.0826 0.509 0.536 0.291 0.0168
##  8 Tr8             28     0    40 0.0979 0.382 0.392 0.328 0.00480
##  9 Tr9             28     0    19 0.0904 0.346 0.338 0.305 0.00966
## 10 Tr10            36     1   104 0.122  0.378 0.529 0.280 0.0840
## 11 Tr11            36     0    65 0.116  0.369 0.505 0.257 0.0744
## 12 Tr12            36     0    37 0.178  0.534 0.721 0.331 0.102
## 13 Tr13            41     0    78 0.0911 0.231 0.202 0.268 0
## 14 Tr14            41     0    23 0.0835 0.305 0.317 0.271 0
## 15 Tr15            46     0    53 0.110  0.348 0.345 0.373 0.0417
## 16 Tr16            46     0    25 0.0912 0.491 0.427 0.321 0.0453
## 17 Tr17            46     0    14 0.0409 0.256 0.322 0.310 0.0600
## 18 Tr18             3     1   103 0.120  0.287 0.303 0.356 0.0638
## 19 Tr19             3     0    75 0.0997 0.280 0.316 0.274 0.0340
## 20 Tr20             3     0    41 0.131  0.299 0.337 0.343 0.0828
```

Tidy data

```
res2 <- TidySataPanel(PID = res$PID,
                      OutPath = "default",
                      TransDummyVars = "default")
```

Modeling

```
res3 = PanelAsso(PID = res$PID,
                 OutPath = "default",
                 VarsY = "Y1",
                 VarsX = "X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,X11,X12",
                 VarsN = "multiple.factor",
                 VarsRandomIpt = "SubjectID",
                 VarsRandomSlp = "none",
                 Covariates = "all.c")

res3$Y1_multiple.factor
```

```
## # A tibble: 12 x 6
##    vars      beta    se p.value beta_lcl beta_hcl
##    <chr>    <dbl> <dbl>   <dbl>    <dbl>    <dbl>
##  1 X1      0.223   3.26  0.945     -6.16    6.61
##  2 X2      7.83   10.7   0.466    -13.2    28.9
##  3 X3     -1.40    5.71  0.806    -12.6     9.80
##  4 X4     -7.90    3.68  0.0318   -15.1    -0.688
##  5 X5      0.0269  4.78  0.996     -9.35    9.40
##  6 X6     -2.37    7.19  0.742    -16.5    11.7
##  7 X7     -0.640   4.87  0.896    -10.2     8.91
##  8 X8     49.1    27.5   0.0736    -4.69  103.
##  9 X9    -37.5    15.1   0.0132   -67.1    -7.83
## 10 X10    12.1     6.97  0.0826    -1.56   25.8
## 11 X11    12.9     8.60  0.134     -3.96   29.7
## 12 X12     1.87    6.87  0.785    -11.6    15.3
```

Visualize model

```
res4 = VizPanelAsso(PID = res$PID,
                    OutPath = "default",
                    VarsY = "Y1",
                    VarsN = "multiple.factor" ,
                    EffectThr = 0.5,
                    Layout = "forest",
                    Brightness = "light",
                    Palette = "default1")
res4$forest_Y1_multiple.factor_forest_light_default1
```

```
## [[1]]
## TableGrob (1 x 1) "arrange": 1 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
##
## attr(,"class")
## [1] "arrangelist" "list"
```

```
FuncExit(PID = res$PID)
```

```
## [1] "Success to exit. Thanks for using ExposomeX platform!"
```

2 EDB module

2.1 EDb function

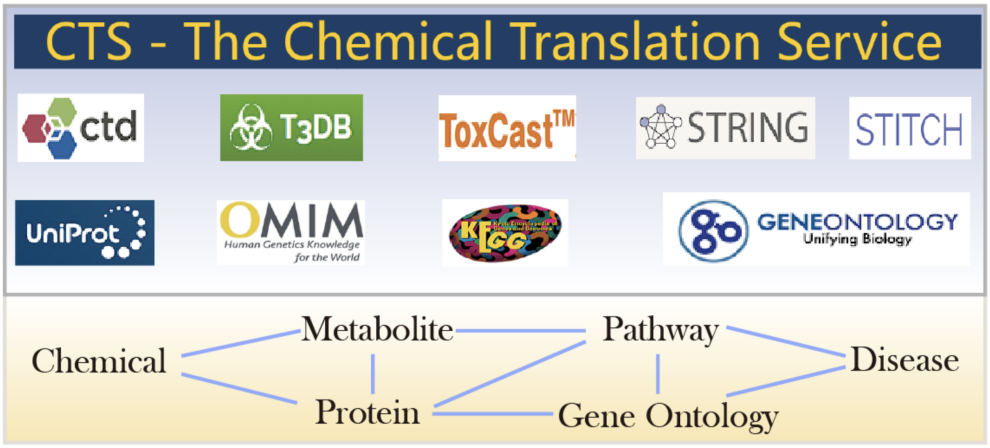
2.1.1 Application domain

EDB module is designed as a convenient tool to explore the data, as well as facilitating to find the biological relationship between exposure and diseases from the perspective of bioinformatics. It provides two main functions for users : EDBNode, EDBLink.

- EDBNode: Search the related information of exposure, metabolite, protein, enzyme, disease, GO, and gene
- EDBLink: Explore the potential nexuses between exposure, protein, phenotype, and disease. Nexus direction from LinkFrom (class A) to LinkTo (class B)

2.1.2 Theory

This module adopts the most frequently-used and authoritative databases, e.g., T3DB, CTD, ToxCast, StringDB, STITCH, KEGG, and GO.It can be summaried by the flow chart below.



2.1.3 Work pipeline

Initialize package

Make sure that the required package is already installed.

```
# The package "exposomex" should be installed in advance
# devtools::install_github("ExposomeX/exposomex", force = TRUE)
library(exposomex)
```

At first, you need to initialize the calculation environment using a series of initialization functions, e.g., InitStatCros, InitStatMO, InitStatTidy, InitEViz, InitEBIO, etc. Here, we use the “EDB” module for example. The detailed information about the functions and returned value will be introcuced in the following chapters.

```
res = InitEDB()
res
```

```
## <eSet>
##   Public:
##     AddCommand: function (x)
##     AddLog: function (x)
##     clone: function (deep = FALSE)
##     Db: list
##     ExcecutionLog: Complete initializing the ExpoStat module.2024.06.23 12. ...
##     FileDirOut: /home/ubuntu/ExposomeX/R_Exposome_1.0/output_124822.0184 ...
##     PID: 124822.018491GZSHJX
##     RCommandLog: eSet <- InitDB()
```

Here, we can see that the returned value "res" is an R6 object. It contains an unique program ID of res$PID (e.g., "100737GJMWJA"), which is random generated by the system. Users need to use it in the following step for further data process.

Upload data

After initializing the calculation environment, the second step is to upload local data file for EDB Module. `LoadEDB` is provided for this. It has three parameters, PID, UseExample and DataPath. PID is Program ID, which must be the same with the PID generated by `InitEDB`. UseExample is a character indicates whether uses example data for analyses, available option include "example#1" for using example data1 and "default" for using data uploaded. DataPath refer to the input file directory, e.g. "D:/test/eg_edb.xlsx". It should be noted that the slash symbol is `/`, not `\`. For convenience, here we use example data one for the following step.

```
res1 = LoadEDB(PID=res$PID,
               UseExample="example#1")
```

ExpoNode

EDBNode provides the functions to explain the keyword in ExposomeX platform. Please attention, PID must be got from the return result of InitEDB(). EDBNode can only run successfully after successfully running InitEDB and LoadEDB functions.

```
res2 = EDBNode(PID=res$PID,OutPath = "default",Class = "GO",Nodes = "default")
res2
```

```
## # A tibble: 7 x 7
##   go.id      label                    go.name ontology definition source version
##   <chr>      <chr>                    <chr>   <chr>    <chr>      <chr>  <chr>
## 1 GO:0000001 mitochondrion inheritan~ mitoch~ BP       The distr~ go.db  v2024.~
## 2 GO:0004857 enzyme inhibitor activi~ GO:004~ MF       Binds to ~ go.db  v2024.~
## 3 GO:0004857 enzyme inhibitor activi~ metall~ MF       Binds to ~ go.db  v2024.~
## 4 GO:0004866 endopeptidase inhibitor~ alpha-~ MF       Binds to ~ go.db  v2024.~
## 5 GO:0004866 endopeptidase inhibitor~ endopr~ MF       Binds to ~ go.db  v2024.~
## 6 GO:0004866 endopeptidase inhibitor~ protei~ MF       Binds to ~ go.db  v2024.~
## 7 GO:0030414 peptidase inhibitor act~ protea~ MF       Binds to ~ go.db  v2024.~
```

ExpoLink

ExpoLink provides the functions to find the nexuses between the keywords in ExposomeX platform. Please attention, PID must be got from the return result of InitEDB(). EDBLink can only run successfully after successfully running InitEDB and LoadEDB functions.

```
res2 = EDBLink(PID=res$PID,OutPath ="default",ClassA = "Exposure",ClassB = "Protein",LinkFrom = "defaul
res2
```

```
## # A tibble: 1 x 12
##   EXE     EXP   remarks source version cid   exposure.preferred.n~1 cas.rn ensp
##   <chr>   <chr> <chr>   <chr>  <chr>   <chr> <chr>                  <chr>  <chr>
## 1 EX:E00~ EX:P~ affect~ ctd^2~ v2024.~ 23978 copper                 7440-~ 9606~
## # i abbreviated name: 1: exposure.preferred.name
## # i 3 more variables: pro.preferred.name <chr>, protein.name <chr>,
## #   uniprot <chr>
```

After all the analysis is done, please run the "FuncExit()" function to delete the data uploaded to the server.

```
FuncExit(PID = res$PID)
```

```
## [1] "Success to exit. Thanks for using ExposomeX platform!"
```

3 EBIO module

3.1 EBio function

3.1.1 Application domain

EBIO module is designed to find the biological relationships between exposure factors and health outcome. This module adopts the most frequently-used and authoritative databases, e.g., T3DB, CTD, ToxCast, StringDB, STITCH, KEGG, and GO.

3.1.2 Theory

The one-layer link modes include: (1) Exposure-GO-Disease (EGoD) supported by two curated databases of "DB_Exposure_GO" and "DB_GO_Disease". Pathway enrichment analysis was used to figure out the GO pathways of each omics data. For enrichment analysis of proteomics data, StringDB database was used as the background. For the metabolites, KEGG pathways and their corresponding metabolites adopted for enrichment analysis. Fisher's Exact Test was used as the algorithm for the pathway enrichment. The GO pathways from exposure-disease links and various omics enrichment were compared, and the matched pathways were highlighted; (2) Exposure-Gene-Disease (EGeD) supported by two curated databases of "DB_Exposure_Gene" and "DB_Gene_Disease"; (3) Exposure-Protein-Disease (EPD) supported by two curated databases of "DB_Exposure_Protein" and "DB_Protein_Disease"; (4) Exposure-Pathway-Disease (EPaD) supported by two curated databases of "DB_Exposure_Pathway" and "DB_Pathway_Disease". In the current version, the two-layer link mode only has Exposure-Protein-Protein-Disease (EPPD) supported by three databases of "DB_Exposure_Protein", "DB_Protein _Disease", and "DB_Protein_Protein".

3.1.3 Work pipeline

Users can easily get the modeling results and their visualization plots of high quality by following the detailed instructions in each step. The biological relationships between exposures and health outcome are interpreted from the perspectives of protein-protein interaction (PPI), gene ontology (GO), protein, gene, pathway.

```
library(tidyverse)
# devtools::install_github("ExposomeX/exposomex", force = TRUE)
library(exposomex)
```

```
res = InitEBIO()

res1 = LoadEBIO(PID=res$PID,
                  UseExample="example#1")
res1$Expo$Data
```

```
## # A tibble: 221 x 7
##    SerialNo FullName GroupName DiseaseID   ExposureID MetabolomeID ProteomeID
##    <chr>    <chr>    <chr>     <chr>       <chr>      <chr>        <chr>
##  1 Y1       CSNU     disease   OMIM:220100 <NA>       <NA>         <NA>
##  2 Y2       SCZD     disease   OMIM:181500 <NA>       <NA>         <NA>
##  3 Y3       TGCT     disease   OMIM:273300 <NA>       <NA>         <NA>
##  4 Y4       RASJ     disease   OMIM:604302 <NA>       <NA>         <NA>
##  5 Y5       HH3      disease   OMIM:244200 <NA>       <NA>         <NA>
##  6 Y6       DECRD    disease   OMIM:616034 <NA>       <NA>         <NA>
##  7 Y7       PMDS1    disease   OMIM:261550 <NA>       <NA>         <NA>
```

```
##  8 Y8       CRMCC2   disease   OMIM:617341 <NA>       <NA>         <NA>
##  9 Y9       FSHD1    disease   OMIM:158900 <NA>       <NA>         <NA>
## 10 Y10      RA       disease   OMIM:180300 <NA>       <NA>         <NA>
## # i 211 more rows
```

```
res2 = EBIOConvToExpoID(PID = res$PID,
                        OutPath = "default")
res2
```

```
## # A tibble: 228 x 8
##    SerialNo FullName GroupName DiseaseID   ExposureID MetabolomeID ProteomeID
##    <chr>    <chr>    <chr>     <chr>       <chr>      <chr>        <chr>
##  1 Y1       CSNU     disease   OMIM:220100 <NA>       <NA>         <NA>
##  2 Y2       SCZD     disease   OMIM:181500 <NA>       <NA>         <NA>
##  3 Y3       TGCT     disease   OMIM:273300 <NA>       <NA>         <NA>
##  4 Y4       RASJ     disease   OMIM:604302 <NA>       <NA>         <NA>
##  5 Y4       RASJ     disease   OMIM:604302 <NA>       <NA>         <NA>
##  6 Y5       HH3      disease   OMIM:244200 <NA>       <NA>         <NA>
##  7 Y5       HH3      disease   OMIM:244200 <NA>       <NA>         <NA>
##  8 Y6       DECRD    disease   OMIM:616034 <NA>       <NA>         <NA>
##  9 Y7       PMDS1    disease   OMIM:261550 <NA>       <NA>         <NA>
## 10 Y7       PMDS1    disease   OMIM:261550 <NA>       <NA>         <NA>
## # i 218 more rows
## # i 1 more variable: EX <chr>
```

```
res3 = EBIOBiolink(PID = res$PID,
                   OutPath = "default",
                   Mode = "EPPD",
                   MetBiospec = "blood")
res3$Edges
```

```
## # A tibble: 851 x 9
##    Source       Target Source preferred nam~1 Target preferred nam~2 Interaction
##    <chr>        <chr>  <chr>                  <chr>                  <chr>
##  1 EX:E0000002~ EX:P0~ 2-hydroxypropane-1,2,~ NME8                   score>500
##  2 EX:E0000002~ EX:P0~ 2-hydroxypropane-1,2,~ SEMA4G                 score>500
##  3 EX:E0000002~ EX:P0~ 2-hydroxypropane-1,2,~ MIF                    binding
##  4 EX:E0000002~ EX:P0~ 2-hydroxypropane-1,2,~ SNRPD3                 score>500
##  5 EX:E0000002~ EX:P0~ 2-hydroxypropane-1,2,~ SLC25A1                score>500
##  6 EX:E0000002~ EX:P0~ 2-hydroxypropane-1,2,~ TXN2                   score>500
##  7 EX:E0000002~ EX:P0~ 2-hydroxypropane-1,2,~ TGFB1                  score>500
##  8 EX:E0000002~ EX:P0~ 2-hydroxypropane-1,2,~ NOD1                   score>500
##  9 EX:E0000002~ EX:P0~ 2-hydroxypropane-1,2,~ CCL2                   affects^co~
## 10 EX:E0000002~ EX:P0~ 2-hydroxypropane-1,2,~ YWHAH                  score>500
## # i 841 more rows
## # i abbreviated names: 1: 'Source preferred name', 2: 'Target preferred name'
## # i 4 more variables: 'Source group' <chr>, 'Target group' <chr>,
## #   Database <chr>, 'Edge type' <dbl>
```

```
res3$Nodes
```

```
## # A tibble: 327 x 5
```

```
##    EXID          `Group name` `Preferred name`                    ID    Synonyms
##    <chr>         <chr>        <chr>                               <chr> <chr>
##  1 EX:E000000296 Exposure     2-hydroxypropane-1,2,3-tricarboxyl~ 77-9~ citric ~
##  2 EX:P000220    Protein_1    NME8                                C9JG~ 51314^A~
##  3 EX:P000287    Protein_1    SEMA4G                              E5RG~ 57715^A~
##  4 EX:P000318    Protein_1    MIF                                 I4AY~ 4GUM^1C~
##  5 EX:P000324    Protein_1    SNRPD3                              P623~ 1D3B^3C~
##  6 EX:P000328    Protein_1    SLC25A1                             P530~ 6576^A8~
##  7 EX:P000368    Protein_1    TXN2                                F8WD~ 1UVZ^1W~
##  8 EX:P000623    Protein_1    TGFB1                               A0A4~ 1KLA^1K~
##  9 EX:P000687    Protein_1    NOD1                                A0A0~ 10392^2~
## 10 EX:P000769    Protein_1    CCL2                                J3KR~ 1DOK^1D~
## # i 317 more rows
```

```
res4 = EBIOBiolink(PID = res$PID,
               OutPath = "default",
               Mode = "EGoD",
               MetBiospec = "blood")
res4$Edges
```

```
## # A tibble: 1,640 x 9
##    Source       Target Source preferred nam~1 Target preferred nam~2 Interaction
##    <chr>        <chr>  <chr>                  <chr>                  <chr>
##  1 EX:E0001316~ GO:00~ nitric oxide           anoikis                association
##  2 EX:E0001316~ GO:00~ nitric oxide           apoptotic DNA fragmen~ association
##  3 EX:E0001316~ GO:00~ nitric oxide           apoptotic process      association
##  4 EX:E0001316~ GO:00~ nitric oxide           cell death             association
##  5 EX:E0001316~ GO:00~ nitric oxide           cellular senescence    association
##  6 EX:E0001316~ GO:00~ nitric oxide           flagellated sperm mot~ association
##  7 EX:E0001316~ GO:00~ nitric oxide           gene expression        association
##  8 EX:E0001316~ GO:00~ nitric oxide           glomerular filtration  association
##  9 EX:E0001316~ GO:00~ nitric oxide           histone H3-K9 dimethy~ association
## 10 EX:E0001316~ GO:00~ nitric oxide           histone H3-K9 methyla~ association
## # i 1,630 more rows
## # i abbreviated names: 1: `Source preferred name`, 2: `Target preferred name`
## # i 4 more variables: `Source group` <chr>, `Target group` <chr>,
## #   Database <chr>, `Edge type` <dbl>
```

```
res4$Nodes
```

```
## # A tibble: 354 x 5
##    EXID          `Group name` `Preferred name`            ID            Synonyms
##    <chr>         <chr>        <chr>                       <chr>         <chr>
##  1 EX:E000131640 Exposure     nitric oxide                10102-43-9^[~ nitric ~
##  2 GO:0043276    GO           anoikis                     GO:0043276    detachm~
##  3 GO:0006309    GO           apoptotic DNA fragmentation GO:0006309    GO:0008~
##  4 GO:0006915    GO           apoptotic process           GO:0006915    GO:0006~
##  5 GO:0008219    GO           cell death                  GO:0008219    acciden~
##  6 GO:0090398    GO           cellular senescence         GO:0090398    ^cellul~
##  7 GO:0030317    GO           flagellated sperm motility  GO:0030317    GO:0097~
##  8 GO:0010467    GO           gene expression             GO:0010467    ^gene e~
##  9 GO:0003094    GO           glomerular filtration       GO:0003094    ^glomer~
## 10 GO:0036123    GO           histone H3-K9 dimethylation GO:0036123    histone~
## # i 344 more rows
```

```
res5 = EBIOBiolink(PID = res$PID,
               OutPath = "default",
               Mode = "EPD",
               MetBiospec = "blood")
res5$Edges
```

```
## # A tibble: 83 x 9
##    Source       Target Source preferred nam~1 Target preferred nam~2 Interaction
##    <chr>        <chr>  <chr>                  <chr>                  <chr>
##  1 EX:E0000002~ EX:P0~ 2-hydroxypropane-1,2,~ MIF                    binding
##  2 EX:E0000002~ EX:P0~ 2-hydroxypropane-1,2,~ DECR1                  score>500
##  3 EX:E0000002~ EX:P0~ 2-hydroxypropane-1,2,~ CHI3L1                 score>500
##  4 EX:E0000002~ EX:P0~ 2-hydroxypropane-1,2,~ PRODH                  score>500
##  5 EX:E0000002~ EX:P0~ 2-hydroxypropane-1,2,~ RGS4                   score>500
##  6 EX:E0000005~ EX:P0~ arsorite               MIF                    affects^bi~
##  7 EX:E0000005~ EX:P0~ arsorite               PROKR2                 score>500
##  8 EX:E0000005~ EX:P0~ arsorite               AMH                    affects^bi~
##  9 EX:E0000005~ EX:P0~ arsorite               TBXT                   score>500
## 10 EX:E0000005~ EX:P0~ arsorite               PRICKLE1               score>500
## # i 73 more rows
## # i abbreviated names: 1: 'Source preferred name', 2: 'Target preferred name'
## # i 4 more variables: 'Source group' <chr>, 'Target group' <chr>,
## #   Database <chr>, 'Edge type' <dbl>
```

```
res5$Nodes
```

```
## # A tibble: 54 x 5
##    EXID          'Group name'   'Preferred name'                   ID    Synonyms
##    <chr>         <chr>          <chr>                              <chr> <chr>
##  1 EX:E000000296 Exposure       2-hydroxypropane-1,2,3-tricarbox~ 77-9~ citric ~
##  2 EX:P000318    Protein        MIF                                I4AY~ 4GUM^1C~
##  3 EX:P000565    Protein        DECR1                              E5RF~ 1.3.1.1~
##  4 EX:P001808    Protein        CHI3L1                             H0Y3~ B2R7B0^~
##  5 EX:P008893    Protein        PRODH                              A0A0~ 1.5.5.2~
##  6 EX:P014634    Protein        RGS4                               E9PR~ 5999^A0~
##  7 EX:E000000516 Exposure       arsorite                           1550~ arsorit~
##  8 EX:P000449    Protein (prot) PROKR2                             Q8NF~ 128674^~
##  9 EX:P000601    Protein        AMH                                P039~ 268^AAA~
## 10 EX:P004593    Protein        TBXT                               H0YM~ 5QRF^5Q~
## # i 44 more rows
```

```
res6 = EBIOBiolink(PID = res$PID,
               OutPath = "default",
               Mode = "EGeD",
               MetBiospec = "blood")
res6$Edges
```

```
## # A tibble: 56,562 x 9
##    Source       Target Source preferred nam~1 Target preferred nam~2 Interaction
##    <chr>        <chr>  <chr>                  <chr>                  <chr>
##  1 EX:E0000002~ EX:G1~ 2-hydroxypropane-1,2,~ BAX                    affects^co~
##  2 EX:E0000002~ EX:G1~ 2-hydroxypropane-1,2,~ BCL2                   affects^co~
```

```
##  3 EX:E0000002~ EX:G1~ 2-hydroxypropane-1,2,~ CCNB1                affects^co~
##  4 EX:E0000002~ EX:G1~ 2-hydroxypropane-1,2,~ CDK2                 affects^co~
##  5 EX:E0000002~ EX:G1~ 2-hydroxypropane-1,2,~ ANGPTL3              increases^~
##  6 EX:E0000005~ EX:G0~ arsorite               eif3s10              decreases^~
##  7 EX:E0000005~ EX:G0~ arsorite               foxg1b               increases^~
##  8 EX:E0000005~ EX:G0~ arsorite               cart1                increases^~
##  9 EX:E0000005~ EX:G1~ arsorite               CYorf15A             decreases^~
## 10 EX:E0000005~ EX:G1~ arsorite               ABCA3                affects^bi~
## # i 56,552 more rows
## # i abbreviated names: 1: `Source preferred name`, 2: `Target preferred name`
## # i 4 more variables: `Source group` <chr>, `Target group` <chr>,
## #   Database <chr>, `Edge type` <dbl>
```

```
res6$Nodes
```

```
## # A tibble: 11,174 x 5
##    EXID          `Group name` `Preferred name`                    ID    Synonyms
##    <chr>         <chr>        <chr>                               <chr> <chr>
##  1 EX:E000000296 Exposure     2-hydroxypropane-1,2,3-tricarboxyl~ 77-9~ citric ~
##  2 EX:G11389563  Gene         BAX                                 581   BCL2L4^~
##  3 EX:G11389573  Gene         BCL2                                596   Bcl-2^P~
##  4 EX:G11389813  Gene         CCNB1                               891   CCNB^MI~
##  5 EX:G11389927  Gene         CDK2                                1017  CDKN2^p~
##  6 EX:G11399518  Gene         ANGPTL3                             27329 ANG-5^A~
##  7 EX:E000000516 Exposure     arsorite                            1550~ arsorit~
##  8 EX:G07715188  Gene         eif3s10                             3275~ wu:fi15~
##  9 EX:G07719167  Gene         foxg1b                              4058~ zgc:859~
## 10 EX:G07726778  Gene         cart1                               5628~ fj47d09~
## # i 11,164 more rows
```

```
res7 = EBIOBiolink(PID = res$PID,
               OutPath = "default",
               Mode = "EPaD",
               MetBiospec = "blood")
res7$Edges
```

```
## # A tibble: 8,598 x 9
##    Source       Target Source preferred nam~1 Target preferred nam~2 Interaction
##    <chr>        <chr>  <chr>                  <chr>                  <chr>
##  1 EX:E0000002~ KEGG:~ 2-hydroxypropane-1,2,~ carbon metabolism      0.00147
##  2 EX:E0000002~ KEGG:~ 2-hydroxypropane-1,2,~ egfr tyrosine kinase ~ 0.000338
##  3 EX:E0000002~ KEGG:~ 2-hydroxypropane-1,2,~ platinum drug resista~ 2.63e-06
##  4 EX:E0000002~ KEGG:~ 2-hydroxypropane-1,2,~ hif-1 signaling pathw~ 1.36e-05
##  5 EX:E0000002~ KEGG:~ 2-hydroxypropane-1,2,~ foxo signaling pathway 0.00263
##  6 EX:E0000002~ KEGG:~ 2-hydroxypropane-1,2,~ cell cycle             0.00205
##  7 EX:E0000002~ KEGG:~ 2-hydroxypropane-1,2,~ p53 signaling pathway  1.97e-06
##  8 EX:E0000002~ KEGG:~ 2-hydroxypropane-1,2,~ pi3k-akt signaling pa~ 8.15e-06
##  9 EX:E0000002~ KEGG:~ 2-hydroxypropane-1,2,~ apoptosis              6.51e-05
## 10 EX:E0000002~ KEGG:~ 2-hydroxypropane-1,2,~ ferroptosis            0.00286
## # i 8,588 more rows
## # i abbreviated names: 1: `Source preferred name`, 2: `Target preferred name`
## # i 4 more variables: `Source group` <chr>, `Target group` <chr>,
## #   Database <chr>, `Edge type` <dbl>
```

```
res7$Nodes
```

```
## # A tibble: 1,115 x 5
##    EXID          `Group name` `Preferred name`                    ID    Synonyms
##    <chr>         <chr>        <chr>                               <chr> <chr>
##  1 EX:E000000296 Exposure     2-hydroxypropane-1,2,3-tricarboxyl~ 77-9~ citric ~
##  2 KEGG:hsa01200 Pathway      carbon metabolism                   KEGG~ carbon ~
##  3 KEGG:hsa01521 Pathway      egfr tyrosine kinase inhibitor res~ KEGG~ egfr ty~
##  4 KEGG:hsa01524 Pathway      platinum drug resistance            KEGG~ platinu~
##  5 KEGG:hsa04066 Pathway      hif-1 signaling pathway             KEGG~ hif-1 s~
##  6 KEGG:hsa04068 Pathway      foxo signaling pathway              KEGG~ foxo si~
##  7 KEGG:hsa04110 Pathway      cell cycle                          KEGG~ cell cy~
##  8 KEGG:hsa04115 Pathway      p53 signaling pathway               KEGG~ p53 sig~
##  9 KEGG:hsa04151 Pathway      pi3k-akt signaling pathway          KEGG~ pi3k-ak~
## 10 KEGG:hsa04210 Pathway      apoptosis                           KEGG~ apoptos~
## # i 1,105 more rows
```

```
FuncExit(PID = res$PID)
```

```
## [1] "Success to exit. Thanks for using ExposomeX platform!"
```

4 EMETA module

4.1 EMeta funtion

4.1.1 Application domain

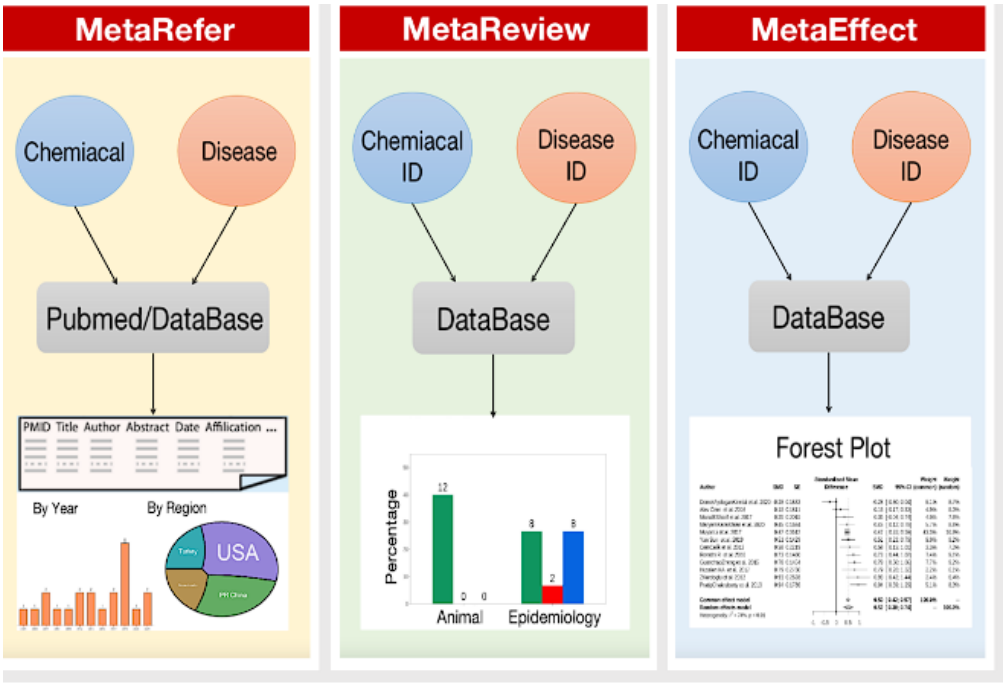
Meta-analyses can be performed when there are multiple scientific studies addressing the same question, with each individual study reporting measurements that are expected to have some degree of error. The aim then is to use approaches from statistics to derive a pooled estimate closest to the unknown common truth based on how this error is perceived. Meta-analytic results are considered the most trustworthy source of evidence by the evidence-based medicine literature.

The EMETA module mainly provides users preliminary information retrieval and screening for meta-analysis. It can also summarize the conclusions and effect values of previous studies (available in our meta database), which can help users form a preliminary view of the research topic and further validate the result from the present study. It provides four main functions for users : MetaRef, VizRefer, MetaRev, and MetaAsso.

- MetaRef: Auto-search all the relative publications from the our meta database and PubMed based on the defined keyword, which has been summarized by the machine learning method and further conveyed to the users.
- VizRefer: VizRefer provides the visual function of papers information searchced or downloaded by `MetaRef`, showing the year and region distribution directly.
- MetaRev: We have been building a standardized database to summarize the up-to-now knowledge about the relationship between environmental exposure and specific diseases. For each topic, the viewpoint from the animal and epidemiological studies are comprehensively summarized by well-trained researchers.
- MetaAsso: For some issues, if their epidemiological studies have been well-conducted, e.g. association between PM2.5 exposure and mortality, the meta-analysis can be conducted to summarize the effect value [e.g. odds ratio (OR), relative risk (RR), hazardous risk (HR)]. The related publications are also summarized in the same database with the "MetaReiew".

4.1.2 Theory

The "EMETA" module is based on "A 24-step guide on how to design, conduct, and successfully publish a systematic review and meta-analysis in medical research",which can be summaried by the flow chart below.



Muka T, Glisic M, Milic J, Verhoog S, Bohlius J, Bramer W, Chowdhury R, Franco OH. A 24-step guide on how to design, conduct, and successfully publish a systematic review and meta-analysis in medical research. Eur J Epidemiol. 2020 Jan;35(1):49-60. doi: 10.1007/s10654-019-00576-5. Epub 2019 Nov 13. PMID: 31720912.

4.1.3 Work pipeline

Initialize package

Make sure that the required packages is already installed.

```
# The package "exposomex" should be installed in advance
# devtools::install_github("ExposomeX/exposomex", force = TRUE)
library(exposomex)
```

At first, you need to initialize the calculation environment using a series of initialization functions, e.g., InitStatCros, InitStatMO, InitStatTidy, InitEViz, InitEBIO, etc. Here, we use the package “exmeta” for meta analysis for example. The detailed information about the functions and returned value will be introcuced in the following chapters.

```
res <- InitEMETA()
res
```

```
## <eSet>
##   Public:
##     AddCommand: function (x)
##     AddLog: function (x)
##     clone: function (deep = FALSE)
##     ExcecutionLog: Complete initializing the ExpoMeta module.2024.06.23 12. ...
##     Expo: list
```

```
##     FileDirIn: NULL
##     FileDirOut: /home/ubuntu/ExposomeX/R_Exposome_1.0/output_125355.3416 ...
##     PID: 125355.341641BKKGBG
##     RCommandLog: eSet <- InitMeta(PID = Any ID your like, FileDirOut = An ...
```

Here, we can see that the returned value "res" is an R6 object. It contains an unique program ID of res$PID (e.g., "100737GJMWJA"), which is random generated by the system. Users need to use it in the following step for further data process.

Upload data

After initializing the calculation environment, the second step is to upload local data file for EMETA Module. `LoadEMETA` is provided for this. It has three parameters, PID, UseExample and DataPath. PID is Program ID, which must be the same with the PID generated by `InitEMETA`. UseExample is a character indicates whether uses example data for analyses, available option include "example#1" for using example data1 and "default" for using data uploaded. DataPath refer to the input file directory, e.g. "D:/test/eg_meta.xlsx". It should be noted that the slash symbol is `/`, not `\`. For convenience, here we use example data one for the following step.

```
res1 <- LoadEMETA(res$PID,
                  UseExample = "example#1")
```

MetaRef

MetaRef provides the functions of paper retrieval and relevance sorting, returning the information to the user based on keywords. Please attention, PID must be got from the return result of InitEMETA(). MetaRef can only run successfully after successfully running InitEMETA and LoadEMETA functions.

Two modes are provided in MetaRef. "Search" for paper retrieval by keywords VarX/VarY/VarM/YearFrom/YearEnd, and "Download" for downloading information (main information only) for specified PMID. Set the Mode parameter as you like. OutPath refers to the output file directory, e.g. "D:/output". If "default", the current working directory will be set. It should be noted that the slash symbol is `/`, not `\`.

VarX/VarY/VarM/YearFrom/YearEnd/PMID parameters can be "default" or a character, run `?MetaRef` to see more details. For convenience, we set them "default" in this example.

```
res2 <- MetaRef(PID = res$PID,
                    OutPath = "default",
                    Mode = "search",
                    VarX = "default",
                    VarY = "default",
                    VarM = "default",
                    YearFrom = "default",
                    YearEnd = "default",
                    PMID = "default")
```

VizRefer

VizRefer function can visualize the articles' main information after MetaRef function. Which has only two parameters, PID and OutPath. PID is Program ID, which must be the same with the PID generated by `InitEMETA` and OutPath refers to the output file directory.

```
res3 <- VizRefer(PID = res$PID,
                 OutPath = "default")
```

MetaRev

MetaRev provides the functions of literature review. In the publishd papers, how many recorded that X is a protective/risky factor for Y? This questions can be solved by MetaRev function. (It can only search the papers available in our database).

MetaRev has four parameters, PID, CID, DID and OutPath. PID and OutPath are the same as introduced above. CID can be "default" or a chemical ID character (separate different values by ","). If "default", the function will use the Chemical_ID values in the file loaded by LoadEMETA. If a character (separate different values by ","), the function will use the chemical ID in the character instead. Chemical_ID refers to the target chemical ID which can be inchikey (eg. JIAARYAFYJHUJI-UHFFFAOYSA-L), cas.rn (eg. 7784-42-1) or our EXC ID (eg. EX:C01631). DID can be "default" or a disease ID character (separate different values by ","). If "default", the function will use the Disease_ID values in the file loaded by LoadEMETA. If a character (separate different values by ","), the function will use the disease ID in the character instead. Disease_ID refers to the target disease ID which can be MESH ID (format like MESH:D006973), OMIM ID (format like OMIM:182940) or our EXD ID (eg. EX:D16243).

```
res2 <- MetaRev(PID = res$PID,
                CID = "default",
                DID = "default",
                OutPath = "default")
```

MetaAsso

MetaAsso provides the functions of effect value pooling. In the publishd papers, what is the effect value between X and Y? This questions can be solved by MetaAsso function. It can provide the combined results of fixed effect model and random effect model. (It can only search the papers available in EMETA database DB_Meta).

MetaAsso has four parameters, PID, CID, DID and OutPath, which are all the same as MetaRev.

```
res2 <- MetaAsso(PID = res$PID,
                 CID = "default",
                 DID = "default",
                 OutPath = "default")
```

After all the analysis is done, please run the "FuncExit()" function to delete the data uploaded to the server.

```
FuncExit(PID = res$PID)
```

```
## [1] "Success to exit. Thanks for using ExposomeX platform!"
```

5 EVIZ module

5.1 EViz funtion

5.1.1 Application domain

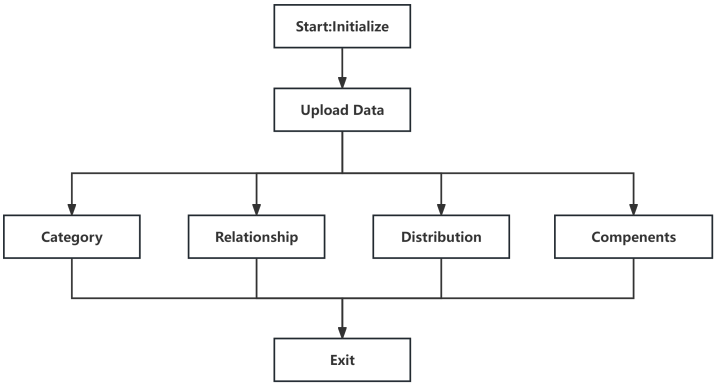
The *EVIZ* module is designed for the data visualization of different statistical and biological analysis in a user friendly and easy way, including four typical classes of visualization. The visualization of the high dimension data is also very useful for the users in the field. The data visualization of different statistical analysis as well as the biological interaction can save much time for the data interpretation. Here, we mainly classify all the visualization task into four types, including category (distinguishing the characteristics of various groups), relationship (statistical relationship between various features), distribution (the distribution characters of the single or multiple factors), and components (the inclusion relationship between the part components and the whole). For each type, users only need to provide the original dataset by following the template and the target types, the *EVIZ* module can generate various potential visualization types.

5.1.2 Theory

The *EVIZ* module is based on "ggplot2" package and its extension packages used for four typical classes of data visualization.Accurate and beautiful visualization can make readers understand the paper presentation easily.The main functions of the visualization include:

(1) To show the data truthfully,accurately and comprehensively;
(2) To carry more information in a smaller space;
(3) To reveal the essence,relationship and rule of data.

5.1.3 Work pipeline



5.1.4 Use example

Initialize module

Make sure that the packages needed is already installed.

```r
# The following packages should be installed in advance

library(tidyverse)

# devtools::install_github("ExposomeX/exposomex", force = TRUE)
library(exposomex)
```

At first, you need to initialize the calculation environment using a series of initialization functions, e.g., InitStatCros, InitStatMo, InitStatTidy, InitEVIZ, InitEBIO, etc. Here, we use the module "EVIZ" for data visualization for example. The detailed information about the functions and returned value will be introcuced in the following chapters.

```r
res = InitEVIZ()
res
```

```
## <eSet>
##   Public:
##     AddCommand: function (x)
##     AddLog: function (x)
##     clone: function (deep = FALSE)
##     ExcecutionLog: Complete initializing the Exoverse module.2024.06.23 12. ...
##     Expo: list
##     ExpoDel: list
##     FileDirIn: NULL
##     FileDirOut: /home/ubuntu/ExposomeX/R_Exposome_1.0/output_125914.4168 ...
##     PID: 125914.416805FDSNSJ
##     RCommandLog: eSet <- InitVisual(PID = Any ID your like, FileDirOut =  ...
```

Here, we can see that the returned value "res" is an R6 object. It contains an unique program ID of res$PID (e.g., "100737GJMWJA"), which is random generated by the system. Users need use it in the following step for further data process.

Upload data

Secondly, you need to load data file for visualization. The PID Parameter, you can enter res$PID which is random generated by the system when you run the InitEVIZ function. If you want to try this package at first, you can input the UseExample Parameter with "example#1" to use our example data. Or you can enter *UseExample = "default"* to use your own data, you can enter *DataPath = " "* and *VocaPath = " "* to choose you own input data file and vocabulary file directories. The demand of these Parameters can see in the help of *LoadEVIZ* function.

You should note that the format of the input data file and vocabulary file is ruled. You can see the demand of the data format by visiting the following website http://www.exposomex.cn.

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitEVIZ*.
UseExample
chr. Method of uploading data. If "default", user should upload their own data files,or use "example#1" provided by this module.
DataPath
chr. Input data file directory, e.g. "D:/test/eg_eviz_data.xlsx". It should be noted that the slash symbol

is “/”, not “\”.
VocaPath
chr. Input vocabulary file directory, e.g. “D:/test/eg_eviz_voca.xlsx”. It should be noted that the slash symbol is “/”, not “\”.

```
res1 <- LoadEVIZ(res$PID,
                 UseExample = "example#1")

res1$Expo$Voca %>%
  dplyr::slice(1:20)
```

```
## # A tibble: 20 x 5
##    SerialNo SerialNo_Raw FullName GroupName Type
##    <chr>    <chr>        <chr>    <chr>     <chr>
##  1 Y1       Y1           Y_cont   Outcome   cont
##  2 Y2       Y2           Y_disc   Outcome   cate
##  3 X1       X1           TE_1     Chemical  cate
##  4 X2       X2           TE_2     Chemical  cate
##  5 X3       X3           TE_3     Chemical  cont
##  6 X4       X4           TE_4     Chemical  cont
##  7 X5       X5           CH1      Chemical  cont
##  8 X6       X6           CH2      Chemical  cont
##  9 X7       X7           CH3      Chemical  cont
## 10 X8       X8           CH4      Chemical  cont
## 11 X9       X9           CH5      Chemical  cont
## 12 X10      X10          CH6      Chemical  cont
## 13 X11      X11          CH7      Chemical  cont
## 14 X12      X12          CH8      Chemical  cont
## 15 X13      X13          CH9      Chemical  cont
## 16 X14      X14          CH10     Chemical  cont
## 17 X15      X15          CH11     Chemical  cont
## 18 X16      X16          CH12     Chemical  cont
## 19 X17      X17          CH13     Chemical  cont
## 20 X18      X18          CH14     Chemical  cont
```

```
res1$Expo$Data %>%
  dplyr::select(Y1:X10) %>%
  dplyr::slice(1:20)
```

```
## # A tibble: 20 x 12
##       Y1    Y2    X1    X2    X3    X4    X5       X6    X7    X8    X9   X10
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl> <dbl> <dbl>
##  1  -101     1     1     1  2.00  1.87 15.7  15.6      7.76 10.2  24.2  14.7
##  2   -51     0     1     1  1.69  1.18  8.80 13.3     10.1  11.6   3.39 13.7
##  3   -37     0     1     1  1.52  1.95  6.35 22.8      8.54  9.52 15.8  16.8
##  4   -61     1     1     1  1.91  1.29  6.49  7.13    14.2  14.9  11.3  12.0
##  5   -28     0     1     2  1.04  1.68  8.75 11.9     11.0  16.4   1.43 16.3
##  6    -8     0     1     1  1.26  1.17  7.30 22.8     11.3  12.7   8.37 17.9
##  7   -63     1     1     1  1.46  1.08 19.1   5.78     7.66  7.73 17.1   4.74
##  8   -35     0     1     1  1.30  1.09  6.72 13.3     11.3   8.25 13.3  16.8
##  9   -14     0     1     1  1.46  1.23  6.80 14.3     14.5  11.1  11.4  20.7
## 10   -99     1     1     1  1.92  1.85 18.9   2.83     6.26  6.04  8.03  6.35
## 11   -60     0     1     1  1.73  1.07  9.35  3.29     3.43 10.3   8.45  6.04
```

```
## 12   -32     0     1     1  1.98  1.06  5.62  1.49     6.75  6.86 28.0   4.68
## 13   -73     0     1     1  1.45  1.53  6.82  8.72     7.71  9.47 22.3   3.65
## 14   -18     0     1     1  1.62  1.58  6.66 11.9     12.6   8.97  9.23  6.04
## 15   -48     0     1     1  1.85  1.94  7.10  6.84    11.7   8.60  9.31  8.47
## 16   -20     0     1     1  1.47  1.48  5.42  9.52     7.10 12.5   2.52 12.9
## 17    -9     0     1     2  1.95  1.18  8.58  9.98     6.32 49.1   8.52  7.57
## 18   -98     1     1     1  1.70  1.98  7.94  0.00937  4.66  8.49 24.2   5.24
## 19   -70     0     1     1  1.87  1.48 18.1   3.32     2.44 -2.27 16.0   7.10
## 20   -36     0     1     1  1.84  1.94 18.8   6.59    10.6   7.07 10.0  10.8
```

Here, we can see that the returned value "res1" is an R6 object. It contains two data frames which are your input data. Users need use it in the following step for further data visualization.

By now, we have prepared our dataset ready for the following visualization.

Category Visualization

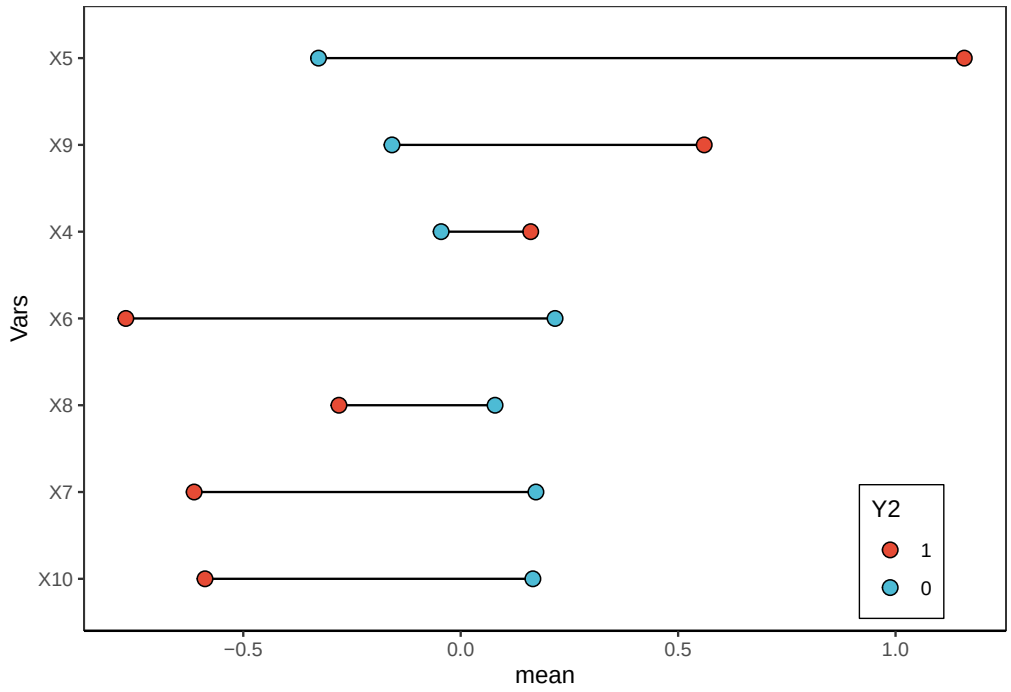
A category comparison chart generally contains two types of data: numerical and categorical, often used to compare the size of data. The input data must contain a categorical variable to use this EVIZCateDot function. We have a function to visualize data via dot plot.

The *Vars* Parameter is necessarily needed, other Parameters have default option. Or users can choose to enter other options.

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitEVIZ*.
OutPath
chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.
Vars
chr. Specifying the variables.Available options include:"all.x", all independent variables; or input a character string specifying the variables,separated by comma "," without space(e.g."X4,X5,X6,X7").No more than 50 variables be entered is recommended (< 50 variables).
Parameter
chr. Specifying which parameter of the data to be the ordinate of the output plot. Available options include: "mean", "median", "min", "max", "mad" or "sd".Default option is "mean".
Brightness
chr. Visualization brightness. Available options include "light" and "dark".
Palette
chr. Visualization palette. Available options include "default1", "default2" and several journal preference styles including "cell", "nature", "science", "lancet", "nejm", and "jama".

```
res2 = EVIZCateDot(PID = res$PID,
                   OutPath = "default",
                   Vars = "X4,X5,X6,X7,X8,X9,X10",
                   Parameter = "mean",
                   Brightness = "light",
                   Palette = "default1")

res2$light_default1
```



Here, we can see that the returned value "res2" is a ggplot2 plot. The dot plot is used to display the relative position of two data points in the same time period, or compare the difference between the two categorical variables. The vertical coordinates are ordered by dividing the absolute value of the difference by the mean value.

Relationship Visualization

Data relational chart includes relatioanal, hierarchical and network relational charts, which respectively show the relationships between two or more variabels, the hierarchical relationships between data individuals and the visualization of relational data without hierarchical structures.

Network Visualization —— EVIZRelatNetwork is used to Visualize data via network plot.

The *VarsY* Parameter is necessarily needed, other Parameters have default option. Or users can choose to enter other options.

The details for each parameter are listed below:

PID

chr. Program ID. It must be the same with the PID generated by *InitEVIZ*.

OutPath

chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.

VarsY

chr. Specifying the dependent variables(e.g."Y2").

VarsX

chr. Specifying the independent variables. Available options include: "all.x", all independent variables; or input a character string specifying the variables,separated by comma "," without space(e.g."X4,X5,X6,X7,X8").

Family

chr. Specifying the data distribution in order to determine the visualization method. Available options include:"gaussian","poisson" and "logistic".Notice that the family are determined by data type of an

outcome, or the plot can not be visualized.

Layout

chr. Visualization layout. Available options include “force-directed” and “degree-circle”.

CutOff

num. Partial outcomes to visualize which is determined by correlation coefficient r. The range must between 0 and 1.

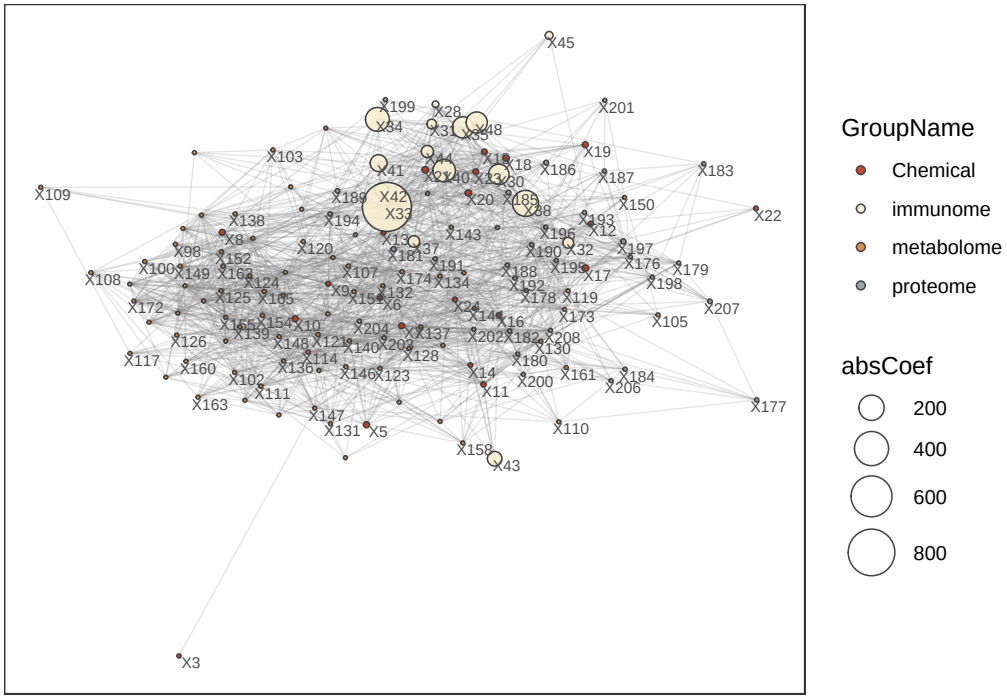
Brightness

chr. Visualization brightness. Available options include “light” and “dark”.

Palette

chr. Visualization palette. Available options include “default1”, “default2” and several journal preference styles including “cell”, “nature”, “science”, “lancet”, “nejm”, and “jama”.

```
res3 = EVIZRelatNetwork(PID = res$PID,
                        OutPath = "default",
                        VarsY = "Y1",
                        VarsX = res1$Expo$Voca$SerialNo[c(3:26,30:50,100:210)],
                        Family = "gaussian",
                        Layout = "force-directed",
                        CutOff = 0.8,
                        Brightness = "light",
                        Palette = "default1")
res3$light_default1
```



Here, we can see that the returned value “res3” is a ggplot2 plot. The network plot is used to visualize the relationship between the input variables.The point color is defined by the coefficient which is calculate by the generalized linear model, the point size is defined by the p-value which is determined by the relationship between the input variables.

Edge Bundling Visualization —— EVIZRelatEdgeBundling is used to Visualize data via edge bundling plot.

The *VarsY* Parameter is necessarily needed, other Parameters have default option. Or users can choose to enter other options.

The details for each parameter are listed below:

PID

chr. Program ID. It must be the same with the PID generated by *InitEVIZ*.

OutPath

chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.

VarsY

chr. Specifying the dependent variables(e.g."Y2").

VarsX

chr. Specifying the independent variables. Available options include: "all.x", all independent variables; or input a character string specifying the variables,separated by comma "," without space(e.g."X4,X5,X6,X7").

Family

chr. Specifying the data distribution in order to determine the visualization method. Available options include:"gaussian","poisson" and "logistic".Notice that the family are determined by data type of an outcome, or the plot can not be visualized.

SizeFor

chr. Parameter to represent the size of the points in the output plot. Available options include "pvalue" and "beta".The default option is "pvalue".

CutOff

num. Partial outcomes to visualize which is determined by correlation coefficient r. The range must between 0 and 1.

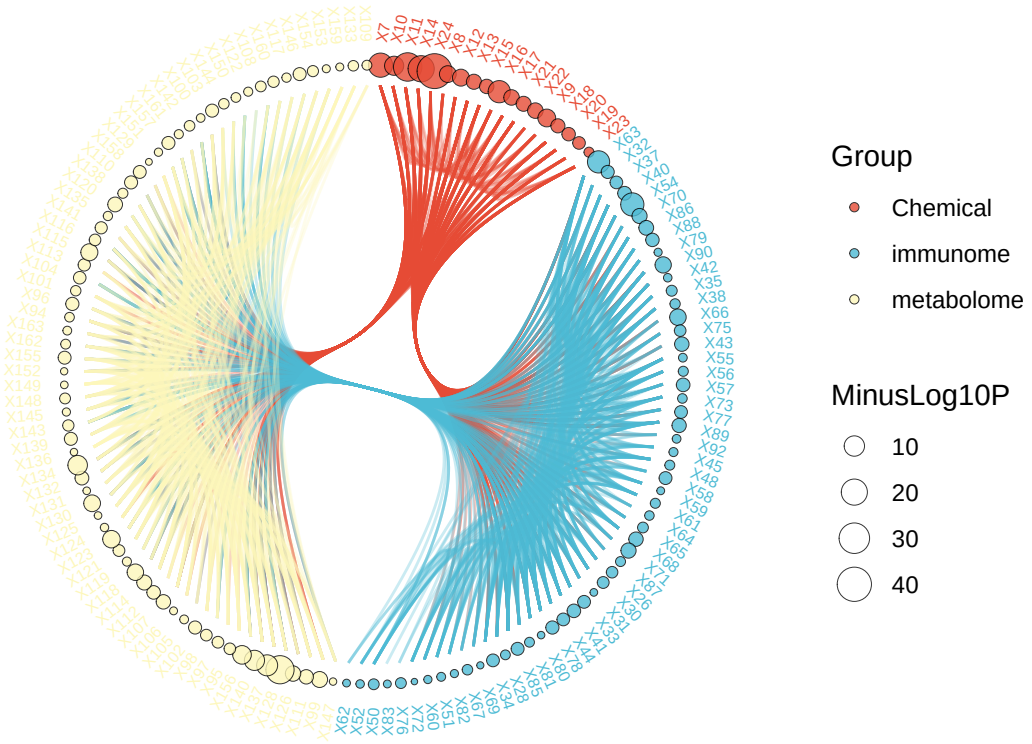
Brightness

chr. Visualization brightness. Available options include "light" and "dark".

Palette

chr. Visualization palette. Available options include "default1", "default2" and several journal preference styles including "cell", "nature", "science", "lancet", "nejm", and "jama".

```
res4 = EVIZRelatEdgeBundling(PID = res$PID,
                             OutPath = "default",
                             VarsY = "Y1",
                             VarsX = "all.x",
                             Family = "gaussian",
                             SizeFor = "pvalue",
                             Brightness = "light",
                             Palette = "default1")
res4$light_default1
```



Here, we can see that the returned value "res4" is a ggplot2 plot. The edge bundling plot is used to bundle the edges closely in order to reduce complexity. The point color is defined by the group of the input independent variables, the point size is defined by the p-value which is determined by the relationship between the input variables.

Heatmap Visualization —— EVIZRelatHeatmap is used to Visualize data via heatmap plot.

The *Vars* Parameter are necessarily needed, other Parameters have default option. Or users can choose to enter other options.

The details for each parameter are listed below:

PID

chr. Program ID. It must be the same with the PID generated by *InitEVIZ*.

OutPath

chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.

Vars

chr. Specifying the independent variables. Available options include: "all.x", all independent variables; or input a character string specifying the variables,separated by comma "," without space(e.g."X4,X5,X6,X7").No more than 200 variables be entered is recommended (< 200 variables).

Method

chr. Method to calculate the correlation. Default option is "spearman".

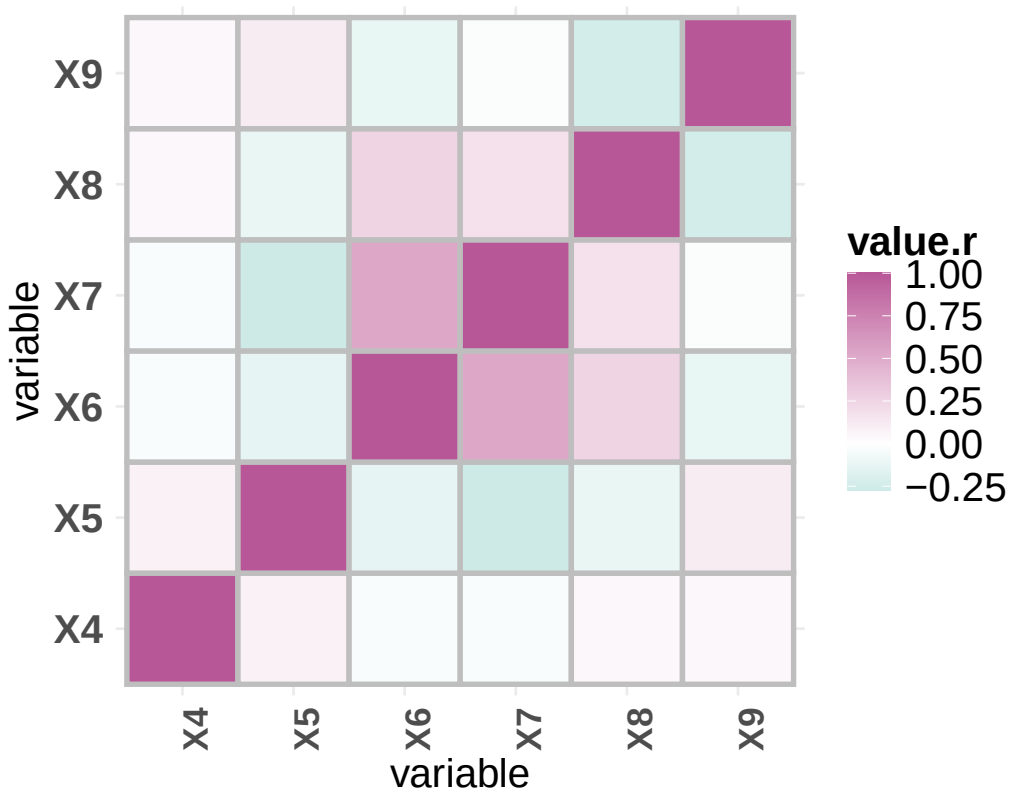
Brightness

chr. Visualization brightness. Available options include "light" and "dark".

Palette

chr. Visualization palette. Available options include "default1", "default2" and several journal preference styles including "cell", "nature", "science", "lancet", "nejm", and "jama".

```
res5 = EVIZRelatHeatmap(PID = res$PID,
                        OutPath = "default",
                        Vars = "X4,X5,X6,X7,X8,X9",
                        Method = "spearman",
                        Brightness = "light",
                        Palette = "default1")

res5$light_default1
```



Here, we can see that the returned value "res5" is a ggplot2 plot. The heatmap plot is used to display data in color changes as a matrix. The point color is defined by the coefficient r which is determined by the relationship between the input variables.

Matrix Visualization —— EVIZRelatMatrix is used to Visualize data via matrix plot.

The *Vars* Parameter are necessarily needed, other Parameters have default option. Or users can choose to enter other options.

The details for each parameter are listed below:

PID

chr. Program ID. It must be the same with the PID generated by *InitEVIZ*.

OutPath

chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.

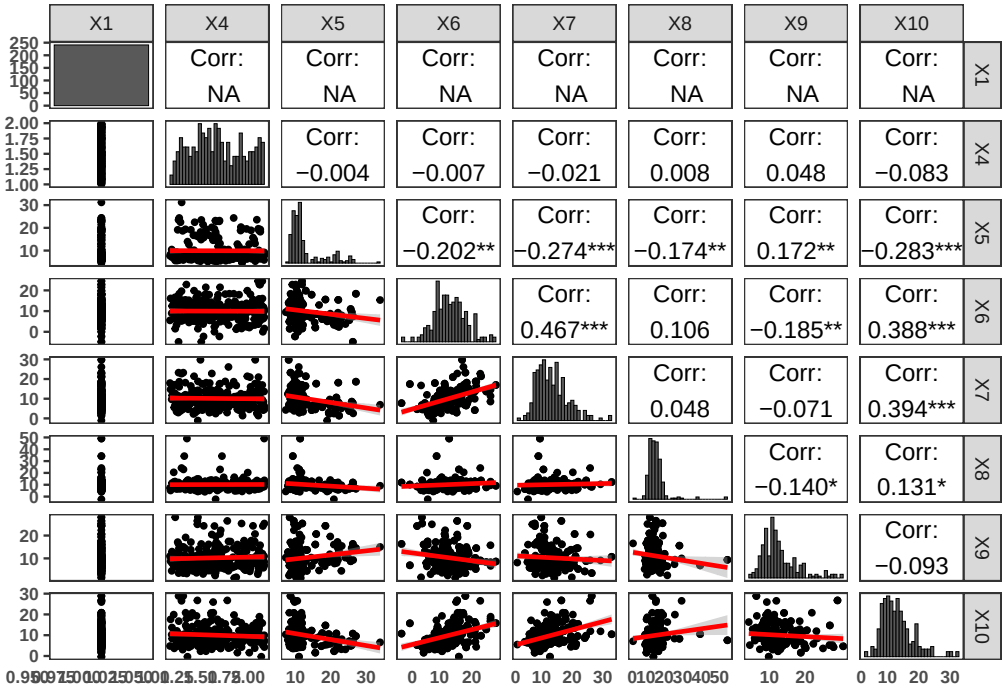
Vars

chr. Specifying the independent variables. Available options include: "all.x", all independent

variables; or input a character string specifying the variables,separated by comma "," without space(e.g."X4,X5,X6,X7").No more than 20 variables be entered is recommended (< 20 variables).
Method

chr. Method to calculate the correlation. Default option is "spearman".

```
res6 = EVIZRelatMatrix(PID = res$PID,
                       OutPath = "default",
                       Vars = "X1,X4,X5,X6,X7,X8,X9,X10",
                       Method = "spearman")

res6
```



Here, we can see that the returned value "res6" is a plot. The matrix plot is used to make a matrix of plots with a given data set.

Distribution Visualization

Data distribution type chart mainly shows the values in the data set and their frequency or distribution rule. Generally, the horizontal axis represents the data type and the vertical axis represents the distribution.We have a EVIZDistrSierra function to visualize data via sierra plot.

The *Vars* Parameter is necessarily needed, other Parameters have default option. Or users can choose to enter other options.

The details for each parameter are listed below:

PID

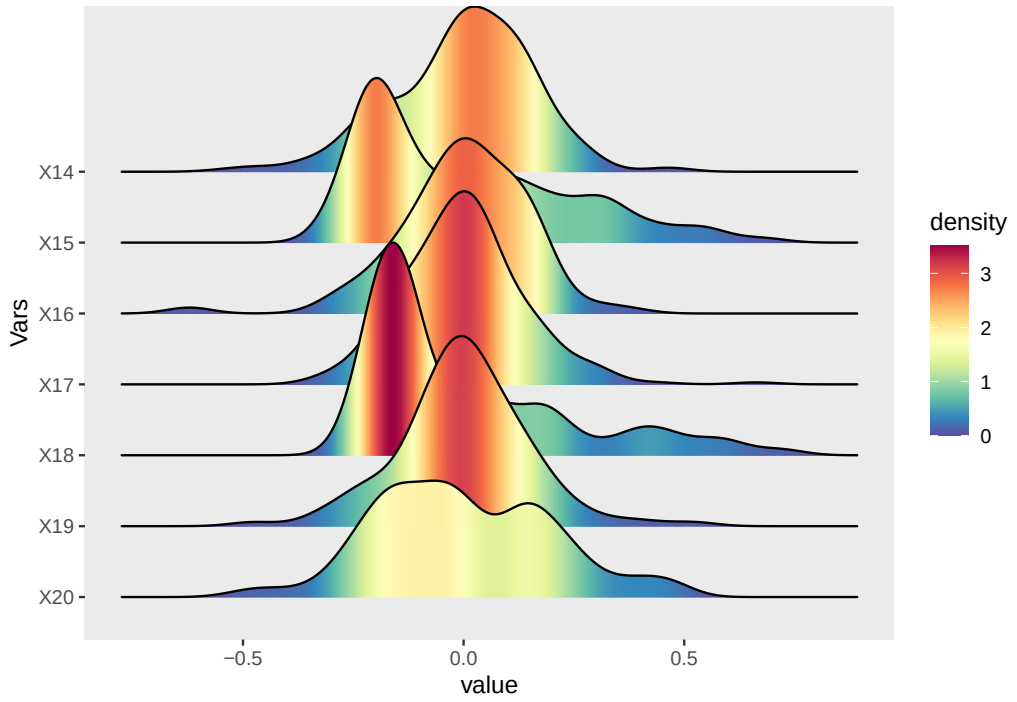
chr. Program ID. It must be the same with the PID generated by *InitEVIZ*.

OutPath

chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If

"default", the current working directory will be set.
Vars
chr. Specifying the variables.Available options include: "all.x", all independent variables; or input a character string specifying the variables,separated by comma "," without space(e.g."X4,X5,X6").No more than 20 variables be entered is recommended (< 20 variables).
Brightness
chr. Visualization brightness. Available options include "light" and "dark".
Palette
chr. Visualization palette. Available options include "default1", "default2" and several journal preference styles including "cell", "nature", "science", "lancet", "nejm", and "jama".

```
res7 = EVIZDistrSierra(PID = res$PID,
                       OutPath = "default",
                       Vars = "X14,X15,X16,X17,X18,X19,X20",
                       Brightness = "light",
                       Palette = "default1")
res7$light_default1
```



Here, we can see that the returned value "res7" is a ggplot2 plot. The sierra plot is used to visualize the kernel density estimation of data.

Components Visualization

Local integrity chart can show the proportion information of the local component and the whole. We have a EVIZCompoDendrogram function to visualize data via dendrogram plot.

The *Vars* Parameter is necessarily needed, other Parameters have default option. Or users can choose to enter other options.

The details for each parameter are listed below:

PID

chr. Program ID. It must be the same with the PID generated by *InitEVIZ*.

OutPath

chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.

Vars

chr. Specifying the variables.Available options include: "all.x", all independent variables; or input a character string specifying the variables,separated by comma "," without space(e.g."X4,X5,X6,X7").No more than 100 variables be entered is recommended (< 100 variables).

Parameter

chr. Specifying which parameter of the data to be the ordinate of the output plot. Available options include: "mean", "median", "min", "max", "mad" or "sd".Default is "mean".

DistMethod

chr.The distance measure. This must be one of "euclidean", "maximum" or "manhattan".Default is "euclidean".

ClusterMethod

chr.The agglomeration method. This should be one of "ward.D", "ward.D2" or "single".Default is "ward.D".

ClusterNum

num. The number of groups for cutting the tree.Default is 4.

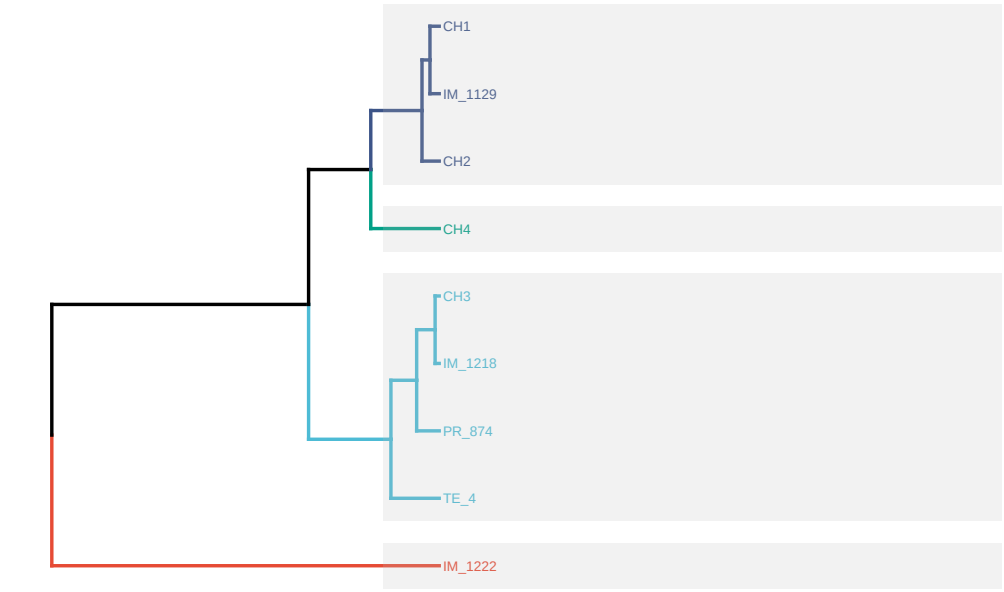
Brightness

chr. Visualization brightness. Available options include "light" and "dark".

Palette

chr. Visualization palette. Available options include "default1", "default2" and several journal preference styles including "cell", "nature", "science", "lancet", "nejm", and "jama".

```
res8 = EVIZCompoDendrogram(PID = res$PID,
                           OutPath = "default",
                           Vars = "X4,X5,X6,X7,X8,X61,X66,X67,X200",
                           Parameter = "median",
                           DistMethod = "euclidean",
                           ClusterMethod = "ward.D2",
                           ClusterNum = "4",
                           Brightness = "light",
                           Palette = "default1")
res8$light_default1
```



Here, we can see that the returned value "res8" is a ggplot2 plot. Here we show one of them. The dendrogram plot is used to plot beautiful dendrograms.

```
FuncExit(PID = res$PID)
```

```
## [1] "Success to exit. Thanks for using ExposomeX platform!"
```

6 ESTAT module

6.1 StatTidy function

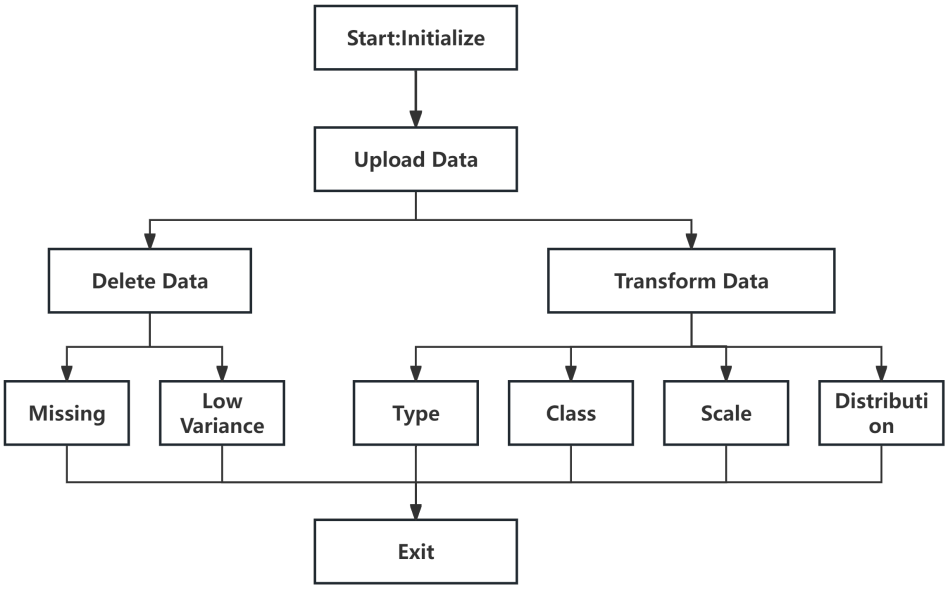
6.1.1 Application domain

The *StatTidy* function is designed to analyze the cross-sectional data from exposome-wide association study (EWAS). This data structure can be obtained from the epidemiological designs of cross-section, case-control, and cohort.

6.1.2 Theory

Users can easily get the modeling results and their visualization plots with high quality by following the detailed instructions in each step. For association, the model is chosen according to the data type of health outcome; while for prediction, most of the frequently-used models are evaluated for users' reference.

6.1.3 Work pipeline



6.1.4 Use example

Initialize package

Make sure that the packages needed is already installed.

```r
# The following package should be installed in advance

library(tidyverse)

# devtools::install_github("ExposomeX/exposomex", force = TRUE)
library(exposomex)
```

At first, you need to initialize the calculation environment using a series of initialization functions, e.g., InitStatCros, InitStatMo, InitStatTidy, InitStatTidy, InitEBIO, etc. Here, we use the function "StatTidy" for tidying data for example. The detailed information about the functions and returned value will be introcuced in the following chapters.

```r
res = InitStatTidy()
res
```

```
## <eSet>
##   Public:
##     AddCommand: function (x)
##     AddLog: function (x)
##     clone: function (deep = FALSE)
##     EpiDesign: NULL
##     ExcecutionLog: Complete initializing the ExpoStat module.2024.06.23 13. ...
##     Expo: list
##     FileDirIn: NULL
##     FileDirOut: /home/ubuntu/ExposomeX/R_Exposome_1.0/output_130640.1281 ...
##     PID: 130640.128142PRHSMZ
##     RCommandLog: eSet <- Init()
##     VarsDel: NULL
```

Here, we can see that the returned value "res" is an R6 object. It contains an unique program ID of res$PID (e.g., "100737GJMWJA"), which is random generated by the system. Users need use it in the following step for further data process.

Upload data

Secondly, you need to load data file for tidy. The PID Parameter, you can enter res$PID which is random generated by the system when you run the InitStatTidy function. If you want to try this package at first, you can input the UseExample Parameter with "example#1" to use our example data. Or you can enter *UseExample = "default"* to use your own data, you can enter *DataPath = " "* and *VocaPath = " "* to choose you own input data file and vocabulary file directories. The demand of these Parameters can see in the help of *LoadStatTidy* function.

You should note that the format of the input data file and vocabulary file is ruled. You can see the demand of the data format by visiting the following website http://www.exposomex.cn.

The details for each parameter are listed below:

PID

chr. Program ID. It must be the same with the PID generated by *InitStatTidy*.

UseExample

chr. Method of uploading data. If "default", user should upload their own data files,or use "example#1" provided by this module.

DataPath

chr. Input data file directory, e.g. “D:/test/eg_stattidy_data.xlsx”. It should be noted that the slash symbol is “/”, not “\”.

VocaPath

chr. Input vocabulary file directory, e.g. “D:/test/eg_stattidy_voca.xlsx”. It should be noted that the slash symbol is “/”, not “\”.

```
res1 <- LoadStatTidy(res$PID,
                     UseExample = "example#1")

res1$Expo$Voca %>%
  dplyr::slice(1:20)
```

```
## # A tibble: 20 x 4
##    SerialNo SerialNo_Raw Type    Lod
##    <chr>    <chr>        <chr> <dbl>
##  1 Y1       Y1           cate   NA
##  2 Y2       Y2           cont   NA
##  3 C1       C1           cont   NA
##  4 C2       C2           cont   NA
##  5 C3       C3           cate   NA
##  6 C4       C4           cate   NA
##  7 C5       C5           cate   NA
##  8 C6       C6           cate   NA
##  9 X1       X1           cont    0.5
## 10 X2       X2           cate    0.5
## 11 X3       X3           cate    0.5
## 12 X4       X4           cont    0.5
## 13 X5       X5           cont    0.5
## 14 X6       X6           cont    0.5
## 15 X7       X7           cont    0.5
## 16 X8       X8           cont    0.5
## 17 X9       X9           cont    5
## 18 X10      X10          cont    5
## 19 X11      X11          cont    5
## 20 X12      X12          cont    5
```

```
res1$Expo$Data %>%
  dplyr::select(Y1:X4) %>%
  dplyr::slice(1:20)
```

```
## # A tibble: 20 x 12
##       Y1    Y2    C1    C2    C3    C4    C5    C6    X1    X2    X3    X4
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1     1  -101  26.9  25.4     3     0     1     3 NA        1     1  1.47
##  2     0   -51  30.9  23.9     1     1     2     2  1.23     1     1  1.03
##  3     0   -37  25.8  23.0     2     3     1     2  1.18     0     1  1.23
##  4     1   -61  38.0  21.2     1     2     2     3  1.90     1     1 NA
##  5     0   -28  31.6  19.5     1     0     2     2 NA        1     2  1.31
##  6     0    -8  25.9  20.8     3     3     1     2  1.99     1     1  1.29
##  7     1   -63  32.4  27.0     2     3     2     1  1.69     1     1  1.49
##  8     0   -35  33.7  22.1     2     0     2     3  1.31     1     1  1.48
##  9     0   -14  32.9  19.8     2     2     1     2  1.84     1     1  1.29
## 10     1   -99  28.5  29.6     1     3     2     2  1.40     1     1  1.92
```

```
## 11     0   -60  37.6  25.3     3     3     1     3 NA        1     1  1.76
## 12     0   -32  31.9  23.3     3     0     1     1  1.17     1     1  1.03
## 13     0   -73  26.9  27.2     3     2     2     3  1.29     1     1 NA
## 14     0   -18  18.9  26.7     2     0     2     3  1.58     1     1  1.23
## 15     0   -48  35.6  22.1     2     2     2     3  1.97     1     1  1.20
## 16     0   -20  29.8  30.6     2     0     2     3  1.53     1     1  1.83
## 17     0    -9  29.9  23.2     1     0     1     2 NA        1     2  1.26
## 18     1   -98  34.7  19.7     3     0     1     2  1.96     1     1  1.07
## 19     0   -70  34.1  23.6     3     0     1     1  1.69     1     1  1.43
## 20     0   -36  33.0  24.6     1     1     2     3  1.07     1     1  1.99
```

Here, we can see that the returned value "res1" is an R6 object. It contains two data frames which are your input data. Users need use it in the following step for further data tidy.

By now, we have prepared our dataset ready for the following tidy.

Missing data imputation

The *Vars* Parameter is necessarily needed, other Parameters have default option. Or users can choose to enter other options.

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitStatTidy*.
OutPath
chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.
Vars
chr. Specifying the variables.Available options include:"all.x", all independent variables; or input a character string specifying the variables,separated by comma "," without space(e.g."X4,X5,X6,X7").
Method
chr. Methods used for imputation. Available options include "lod" or "cart" methods. For "lod" method, limit of detection (LOD) should be included in the "Vocabulary" file.

```
res2 = TidyTransImput(PID=res$PID,
                      OutPath = "default",
                      Vars="all.x",
                      Method="lod")
```

Delete variables with missing values

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitStatTidy*.
OutPath
chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.

```
res3 = TidyDelMiss(PID=res$PID,
                   OutPath = "default")
```

Delete variables with low variance

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitStatTidy*.
OutPath
chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.

```
res4 = TidyDelNearZeroVar(PID=res$PID,
                          OutPath = "default")
```

Transform data type

The *Vars* Parameter is necessarily needed, other Parameters have default option. Or users can choose to enter other options.

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitStatTidy*.
OutPath
chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.
Vars
chr. Specifying the variables.Available options include:"all.x", all independent variables; or input a character string specifying the variables,separated by comma "," without space(e.g."X4,X5,X6,X7").
To
chr. Indicate the type of the chosen variables to be transformed into.Available options include "integer", "numeric", "character", "factor", "logical", and "date".

```
res5 = TidyTransType(PID=res$PID,
                     OutPath = "default",
                     Vars="Y1",
                     To="factor")
```

Classify variables into various groups

The *Vars* Parameter is necessarily needed, other Parameters have default option. Or users can choose to enter other options.

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitStatTidy*.
OutPath
chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.
Vars
chr. Specifying the variables.Available options include:"all.x", all independent variables; or input a character string specifying the variables,separated by comma "," without space(e.g."X4,X5,X6,X7").
LevelTo
num. The number of levels to convert variables to.

```
res6 = TidyTransClass(PID=res$PID,
                      OutPath = "default",
                      Vars="X1",
                      LevelTo="4")
```

Scale variables

The *Vars* Parameter is necessarily needed, other Parameters have default option. Or users can choose to enter other options.

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitStatTidy*.
OutPath
chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.
Vars
chr. Specifying the variables.Available options include:"all.x", all independent variables; or input a character string specifying the variables,separated by comma "," without space(e.g."X4,X5,X6,X7").
Method
chr. Scaling methods. Available options include "normal" and "range".
Direct
chr. Direction to be transformed, Available options include "positive" and "negative".
RangeLow
chr. Lower limit for range method.
RangeUpper
chr. Upper limit for range method. It should be greater than the lower limit.

```
res7 = TidyTransScale(PID=res$PID,
                      OutPath = "default",
                      Vars="X4,X5",
                      Method="normal")
```

Transform variable distribution

The *Vars* Parameter is necessarily needed, other Parameters have default option. Or users can choose to enter other options.

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitStatTidy*.
OutPath
chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.
Vars
chr. Specifying the variables.Available options include:"all.x", all independent variables; or input a character string specifying the variables,separated by comma "," without space(e.g."X4,X5,X6,X7").
Method
chr. Method used to transform distribution of target variables. Available options include:
"ln", natural logarithm;
"log2", binary logarithm or logarithm base 2;
"log10" logarithm base 10.

```
res8 = TidyTransDistr(PID=res$PID,
                      OutPath = "default",
                      Vars="X6,X7",
                      Method="log10")
```

```
FuncExit(PID = res$PID)
```

```
## [1] "Success to exit. Thanks for using ExposomeX platform!"
```

6.2 StatDesc function

6.2.1 Application domain

Data statistics is designed for the statistical description of data. Before selecting the statistical algorithm to build the model, it is necessary to understand the distribution of the data and the socio-demographic characteristics of the population. The results of the statistical description provide the basis for selecting the statistical algorithm to build the model and the potential confounding to be adjusted during the modeling process.

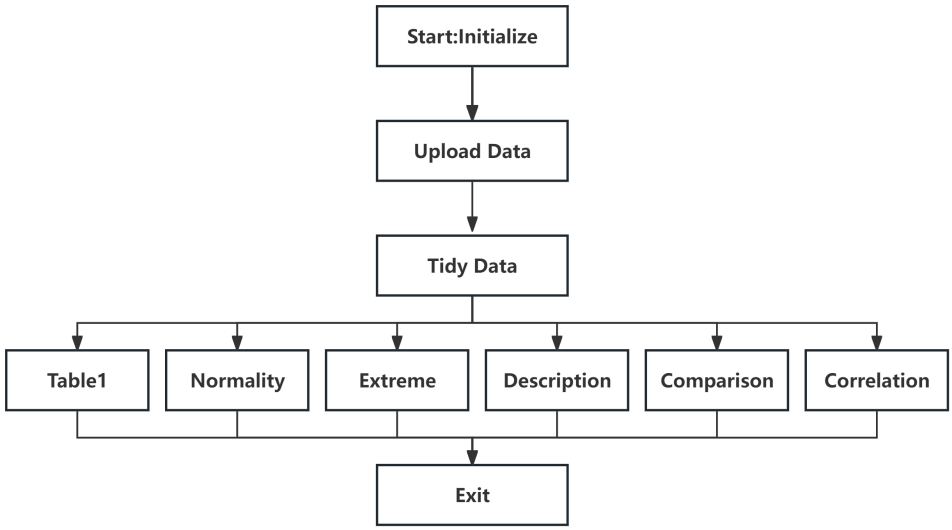
The StatDesc function provides functions for creating the basic descriptive statistics table, testing normality distribution of variable, screening extreme values, comparing the size between/among groups, and calculating correlation coefficient between variables. The visualization of the statistical results are also provided. It provides six main functions for users : DescTable1, DescNorm, DescExtre, DescSize, DescComp, and DescCorr.

- DescTable1: Create a statistical description table of demographic characteristics for three common types of epidemiological studies(i.e., cohort, case control, and cross sectional).
- DescNorm: Normality test for continuous variables, and visualizations of whether variables conform to normal distribution are provided.
- DescExtre: Calculating The extreme value of the variable and output the result. The visualization of the result shows the frequency of the extreme value in each subject.
- DescSize: Calculate the mean, standard deviation, median and quartile of continuous variables, and the count and frequency of discrete variables.
- DescComp: Performing inter-group comparison for continuous variables and visualize whether there is a statistical difference between the different groups.
- StatCorr: Calculate the correlation coefficient between each variable, output the correlation coefficient and P value, and visualize the results.

6.2.2 Theory

The different “StatDesc” function is based on different statistics test. Users can easily get the modeling results and their visualization plots with high quality by following the detailed instructions in each step. Only typical statistical methods are included in our module for convenience. For example, the statistical method of normality test is Shapiro-Wilk test, the statistical method of inter-group comparison is Wilcoxon rank test, and the statistical method of calculating the correlation coefficient between each variable is Pearson correlation coefficient or Spearman rank correlation coefficient.

6.2.3 Work pipeline



6.2.4 Use example

Initialize package

Make sure that the required packages is already installed.

```
library(tidyverse)
library(gridExtra)
# devtools::install_github("ExposomeX/exposomex", force = TRUE)
library(exposomex)
```

At first, you need to initialize the calculation environment using a series of initialization functions, e.g., InitStatCros, InitStatMo, InitStatTidy, InitEVIZ, InitEBIO, etc. Here, we use the module "StatDesc" for data description for example. The detailed information about the functions and returned value will be introcuced in the following chapters.

```
res <- InitStatDesc()
res
```

```
## <eSet>
##   Public:
##     AddCommand: function (x)
##     AddLog: function (x)
##     clone: function (deep = FALSE)
##     EpiDesign: NULL
##     ExcecutionLog: Complete initializing the StatDesc module.2024.06.23 13. ...
##     Expo: list
##     FileDirIn: NULL
```

```
##     FileDirOut: /home/ubuntu/ExposomeX/R_Exposome_1.0/output_131718.9934 ...
##     PID: 131718.993465WURQCC
##     RCommandLog: eSet <- InitDesc(PID = Any ID your like, FileDirOut = An ...
##     VarsDel: NULL
```

Here, we can see that the returned value "res" is an R6 object. It contains an unique program ID of res$PID (e.g., "100737GJMWJA"), which is random generated by the system. Users need to use it in the following step for further data process.

Upload data

After initializing the calculation environment, the second step is to upload local data file for StatDesc Module. `LoadStatDesc` is provided for this. It has four parameters, PID, UseExample, DataPath and VocaPath. If you want to try this package at first, you can input the UseExample Parameter with "example#1" to use our example data. Or you can enter *UseExample = "default"* to use your own data, you can enter *DataPath = " "* and *VocaPath = " "* to choose you own input data file and vocabulary file directories. The demand of these Parameters can see in the help of *LoadStatDesc* function.

You should note that the format of the input data file and vocabulary file is ruled. You can see the demand of the data format by visiting the following website http://www.exposomex.cn.

The details for each parameter are listed below:

PID
chr. Program ID, which must be the same with the PID generated by `InitStatDesc`.
UseExample
chr.A character indicates whether uses example data for analyses, available option include "example#1" for using example data1 and "default" for using data uploaded.
DataPath
chr. Input file directory, e.g. "D:/test/eg_data.xlsx". It should be noted that the slash symbol is `/`, not \.
VocaPath
chr. Input vocabulary file directory, e.g. "D:/test/eg_voca.xlsx". It should be noted that the slash symbol is "/", not "".

```
res1 <- LoadStatDesc(PID = res$PID,
                     UseExample = "example#1")

res1$Expo$Voca %>%
  dplyr::slice(1:20)
```

```
## # A tibble: 20 x 6
##    SerialNo SerialNo_Raw FullName Type  GroupName    Lod
##    <chr>    <chr>        <chr>    <chr> <chr>      <dbl>
##  1 Y1       Y1           Y_disc   cate  Outcome     NA
##  2 Y2       Y2           Y_cont   cont  Outcome     NA
##  3 C1       C1           Cov_1    cont  Demography  NA
##  4 C2       C2           Cov_2    cont  Demography  NA
##  5 C3       C3           Cov_3    cate  Demography  NA
##  6 C4       C4           Cov_4    cate  Demography  NA
##  7 C5       C5           Cov_5    cate  Demography  NA
##  8 C6       C6           Cov_6    cate  Demography  NA
##  9 X1       X1           TE_1     cont  Chemical     0.5
## 10 X2       X2           TE_2     cate  Chemical     0.5
```

```
## 11 X3       X3          TE_3     cate  Chemical     0.5
## 12 X4       X4          TE_4     cont  Chemical     0.5
## 13 X5       X5          TE_5     cont  Chemical     0.5
## 14 X6       X6          TE_6     cont  Chemical     0.5
## 15 X7       X7          TE_7     cont  Chemical     0.5
## 16 X8       X8          TE_8     cont  Chemical     0.5
## 17 X9       X9          CH1      cont  Chemical     5
## 18 X10      X10         CH2      cont  Chemical     5
## 19 X11      X11         CH3      cont  Chemical     5
## 20 X12      X12         CH4      cont  Chemical     5
```

```
res1$Expo$Data %>%
  dplyr::select(SampleID:X2) %>%
  dplyr::slice(1:20)
```

```
## # A tibble: 20 x 12
##    SampleID SubjectID    Y1    Y2    C1    C2    C3    C4    C5    C6    X1
##    <chr>    <chr>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 Tr1      S1            1  -101  26.9  25.4     3     0     1     3 NA
##  2 Tr2      S2            0   -51  30.9  23.9     1     1     2     2  1.23
##  3 Tr3      S3            0   -37  25.8  23.0     2     3     1     2  1.18
##  4 Tr4      S4            1   -61  38.0  21.2     1     2     2     3  1.90
##  5 Tr5      S5            0   -28  31.6  19.5     1     0     2     2 NA
##  6 Tr6      S6            0    -8  25.9  20.8     3     3     1     2  1.99
##  7 Tr7      S7            1   -63  32.4  27.0     2     3     2     1  1.69
##  8 Tr8      S8            0   -35  33.7  22.1     2     0     2     3  1.31
##  9 Tr9      S9            0   -14  32.9  19.8     2     2     1     2  1.84
## 10 Tr10     S10           1   -99  28.5  29.6     1     3     2     2  1.40
## 11 Tr11     S11           0   -60  37.6  25.3     3     3     1     3 NA
## 12 Tr12     S12           0   -32  31.9  23.3     3     0     1     1  1.17
## 13 Tr13     S13           0   -73  26.9  27.2     3     2     2     3  1.29
## 14 Tr14     S14           0   -18  18.9  26.7     2     0     2     3  1.58
## 15 Tr15     S15           0   -48  35.6  22.1     2     2     2     3  1.97
## 16 Tr16     S16           0   -20  29.8  30.6     2     0     2     3  1.53
## 17 Tr17     S17           0    -9  29.9  23.2     1     0     1     2 NA
## 18 Tr18     S18           1   -98  34.7  19.7     3     0     1     2  1.96
## 19 Tr19     S19           0   -70  34.1  23.6     3     0     1     1  1.69
## 20 Tr20     S20           0   -36  33.0  24.6     1     1     2     3  1.07
## # i 1 more variable: X2 <dbl>
```

Here, we can see that the returned value "res1" is an R6 object. It contains two data frames which are your input data. Users need use it in the following step for further data process.

By now, we have prepared our dataset ready for the following calculation.

Tidy data

After initializing the calculation environment, and upload local data file for StatDesc Module. This function includes deleting missing values and low variation variables.Users can also use other tidy functions in the StatTidy Module.

The details for each parameter are listed below:

PID

chr. Program ID. It must be the same with the PID generated by *InitStatDesc*.

OutPath

chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.

```
res2 <- TidyStatDesc(PID = res$PID,
                     OutPath = "default")
```

DescTable1

DescTable1 provides sociodemographic information tables for three common epidemiological studies, returning the information to the user based on epidemiological design. EpiDesign/VarsY/VarsC/Missing parameters must be entered. Attention please, PID must be got from the return result of InitStatDesc(). DescTable1 can only run successfully after successfully running InitStatDesc and LoadStatDesc functions.

EpiDesign/VarsY/VarsC/Missing parameters can be a character, run `?DescTable1` to see more details.

```
res3 <- DescTable1(PID = res$PID,
                   OutPath = "default",
                   EpiDesign = "case.control",
                   VarsY = "Y1",
                   VarsC = "C1,C2,C3,C4,C5",
                   Missing = "no")
#res3$Stat_Table1
```

DescNorm

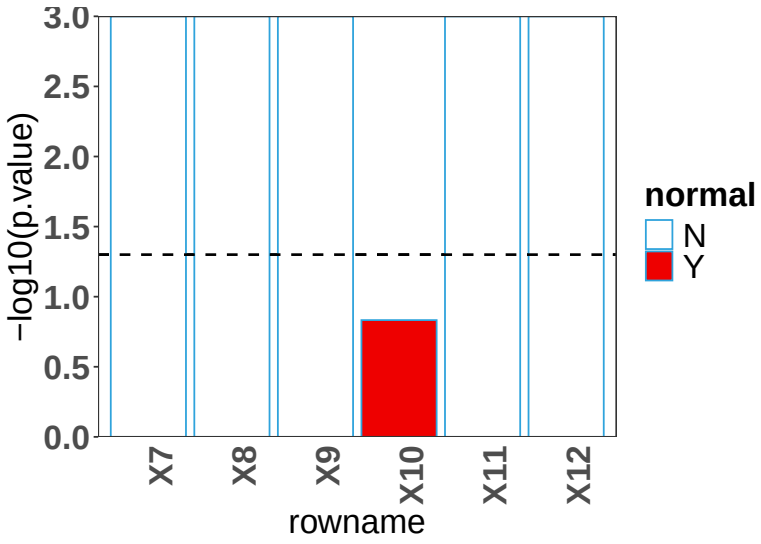
DescNorm provides function of normality test for continuous variable.For the time being, only the typical Shapiro-Wilk test is provided to test the normality of variables, and other tests will be added gradually in the future. Vars/Method/Layout/Brightness/Palette parameters must be entered. Attention please, PID must be got from the return result of InitStatDesc(). DescNorm can only run successfully after successfully running InitStatDesc and LoadStatDesc functions.

Vars/Method/Layout/Brightness/Palette parameters can be a character, run `?DescNorm` to see more details.

```
res4 <- DescNorm(PID = res$PID,
                 OutPath = "default",
                 Vars =  "X7,X8,X9,X10,X11,X12",
                 Method = "shapiro.test",
                 Layout = "column",
                 Brightness = "light",
                 Palette = "default2")
res4$normtable
```

```
## $DescNorm_by_shapiro.test
## # A tibble: 6 x 5
##   Group Vars      P Method       Normal
##   <chr> <fct> <dbl> <chr>        <chr>
## 1 all   X7    3     shapiro.test N
## 2 all   X8    3     shapiro.test N
## 3 all   X9    3     shapiro.test N
## 4 all   X10   0.833 shapiro.test Y
## 5 all   X11   3     shapiro.test N
## 6 all   X12   3     shapiro.test N
```

```
res4$NormPlot$VizDescNorm_Column_light_default2
```



DescExtre

DescExtre provides the functions of calculating The extreme value for the continuous variable. The two parameters 'LimitLow' and 'LimitUpper' can be set to adjust the percentile of the variable. Vars/LimitLow/LimitUpper/Layout/Brightness/Palette parameters must be entered. Attention please, PID must be got from the return result of InitStatDesc(). DescExtre can only run successfully after successfully running InitStatDesc and LoadStatDesc functions.

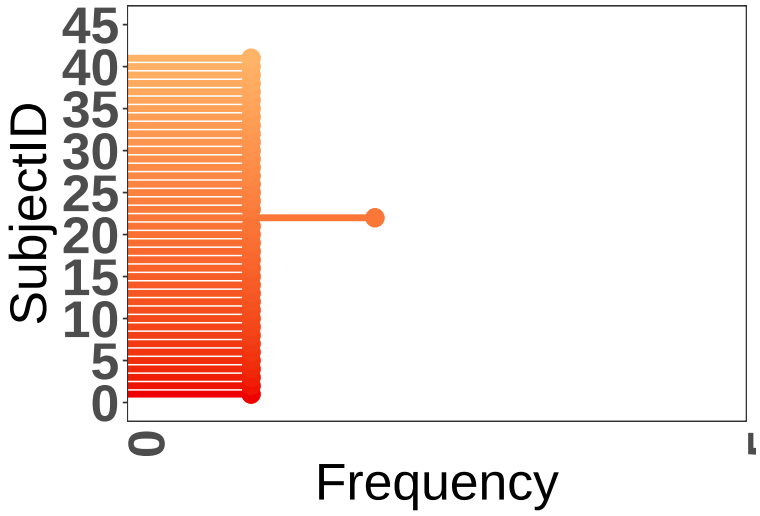
Vars/LimitLow/LimitUpper/Layout/Brightness/Palette parameters can be a character, run `?DescExtre` to see more details.

```
res5 <- DescExtre(PID = res$PID,
                  OutPath = "default",
                  Vars = "X5,X6,X7,X8,X9",
                  LimitLow = 0.025,
                  LimitUpper = 0.975,
                  Layout = "column.points",
                  Brightness = "light",
                  Palette = "default1")
res5$Extretable
```

```
## # A tibble: 42 x 4
##    SerialNo value SubjectID SampleID
##    <chr>    <dbl> <chr>     <chr>
##  1 X7       -5    S106      Tr106
##  2 X7       -4.91 S150      Tr150
##  3 X7       -4.86 S102      Tr102
##  4 X7       -4.85 S194      Te70
##  5 X7       -4.69 S224      Te120
##  6 X7       -4.68 S210      Te98
##  7 X7       -4.59 S5        Tr5
##  8 X7        4.47 S26       Tr26
##  9 X7        4.53 S17       Tr17
```

```
## 10 X7        4.54 S161      Te20
## # i 32 more rows
```

```
res5$ExtrePlot$VizDescExtre_ColumnPoints_light_default1
```



DescSize

DescSize provides the description of the size of a variable. Different types of variables have different size descriptions.The module calculates the mean, standard deviation, median, quartile and other information of continuous variables, the count and frequency of discrete variables. The visualization of this part is different from the visualization of the result statistics of other functions. This part only provides the box diagram and violin diagram to represent the median quartile of continuous variables. Vars/VarsBy/Layout/Brightness/Palette parameters must be entered. Attention please, PID must be got from the return result of InitStatDesc(). StatDesc can only run successfully after successfully running InitStatDesc and LoadStatDesc functions.

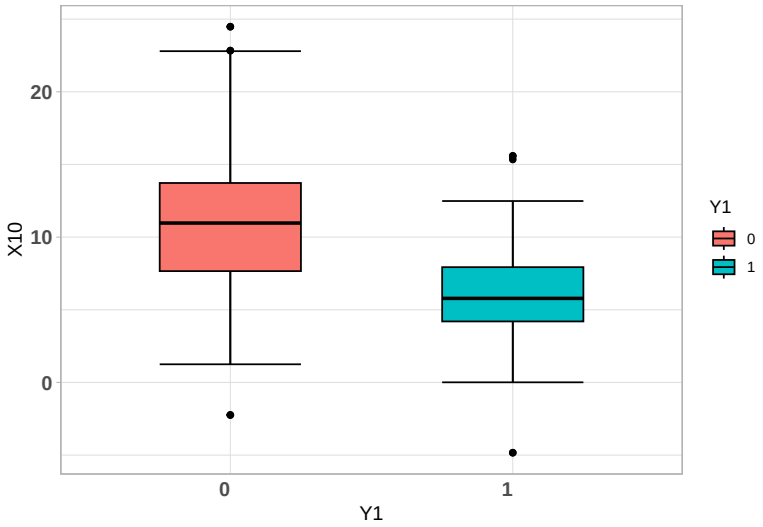
Vars/VarsBy/Layout/Brightness/Palette parameters can be a character, run `?StatDesc` to see more details.

```
res6 <- DescSize(PID = res$PID,
                 OutPath = "default",
                 Vars = "X5,X6,X7,X8,X9,X10,X11",
                 VarsBy = "Y1",
                 Layout = "box",
                 Brightness = "light",
                 Palette = "default1")
res6$Desctable
```

```
## $`/Stat_DescNum_by_Y1`
##    varsby level variable   n    min    max median     q1     q3   iqr   mad
## 1      Y1     1       X7  53 -4.859  5.000 -0.258 -2.762  1.983 4.745 3.477
## 2      Y1     0       X7 188 -5.000  4.779  0.236 -2.270  2.419 4.690 3.684
## 3      Y1     1       X8  53 -4.179  5.000 -0.075 -1.760  3.117 4.877 3.291
## 4      Y1     0       X8 188 -5.000  4.984 -0.371 -2.381  2.276 4.657 3.434
## 5      Y1     1       X9  53 -4.617  5.000 -0.615 -3.245  0.653 3.898 1.956
## 6      Y1     0       X9 188 -5.000  2.434 -3.876 -4.196 -3.474 0.722 0.554
```

```
## 7      Y1     1      X10  53 -4.834 15.587  5.784  4.199  7.933 3.734 3.091
## 8      Y1     0      X10 188 -2.238 24.479 10.968  7.661 13.725 6.065 4.432
## 9      Y1     1      X11  53 -1.092 22.812  6.583  4.599  7.757 3.158 2.711
## 10     Y1     0      X11 188  2.437 29.755 10.319  7.038 13.655 6.617 4.933
##      mean    sd    se    ci
## 1  -0.265 2.823 0.388 0.778
## 2   0.097 2.826 0.206 0.407
## 3   0.433 2.798 0.384 0.771
## 4  -0.151 2.839 0.207 0.408
## 5  -0.780 2.244 0.308 0.618
## 6  -3.586 1.201 0.088 0.173
## 7   6.041 4.099 0.563 1.130
## 8  11.101 4.835 0.353 0.696
## 9   6.970 4.136 0.568 1.140
## 10 11.015 5.058 0.369 0.728
```

```
res6$DescPlot$VizDescSize_Box_X10_By_Y1_light_default1
```



DescComp

DescComp provides the function of size comparison between groups. For the time being, only the typical Wilcoxon rank test is provided to test the difference between groups, and other tests will be added gradually in the future. Task/Vars/VarsBy/Method/Layout/Brightness/Palette parameters must be entered. Attention please, PID must be got from the return result of InitStatDesc(). StatDesc can only run successfully after successfully running InitStatDesc and LoadStatDesc functions.

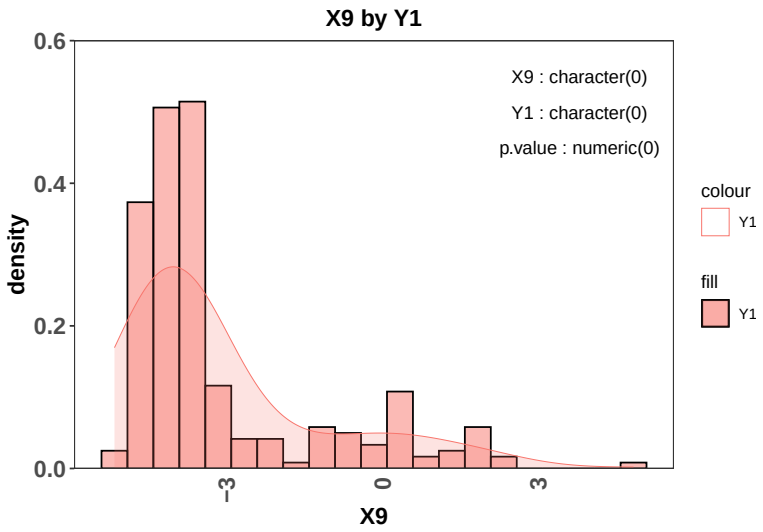
Task/Vars/VarsBy/Method/Layout/Brightness/Palette parameters can be a character, run `?DescComp` to see more details.

```
res7 <- DescComp(PID = res$PID,
                 OutPath = "default",
                 Task = "mean",
                 Vars = "X7,X8,X9",
                 VarsBy = "Y1",
                 Method = "wilcox",
                 Layout = "density",
```

```
                Brightness = "light",
                Palette = "default1")
res7$Comptable
```

```
## NULL
```

```
res7$CompPlot$VizDescComp_Density_X9_by_Y1
```



DescCorr

DescCorr provides the function of calculating of correlation coefficient between variables. For the time being, there are two typical statistics method Pearson correlation coefficient or Spearman rank correlation coefficient is provided to calculating of correlation, and statistics method will be added gradually in the future. VarsX/VarsBy/Method/Layout/Brightness/Palette parameters must be entered. Attention please, PID must be got from the return result of InitStatDesc(). StatDesc can only run successfully after successfully running InitStatDesc and LoadStatDesc functions.

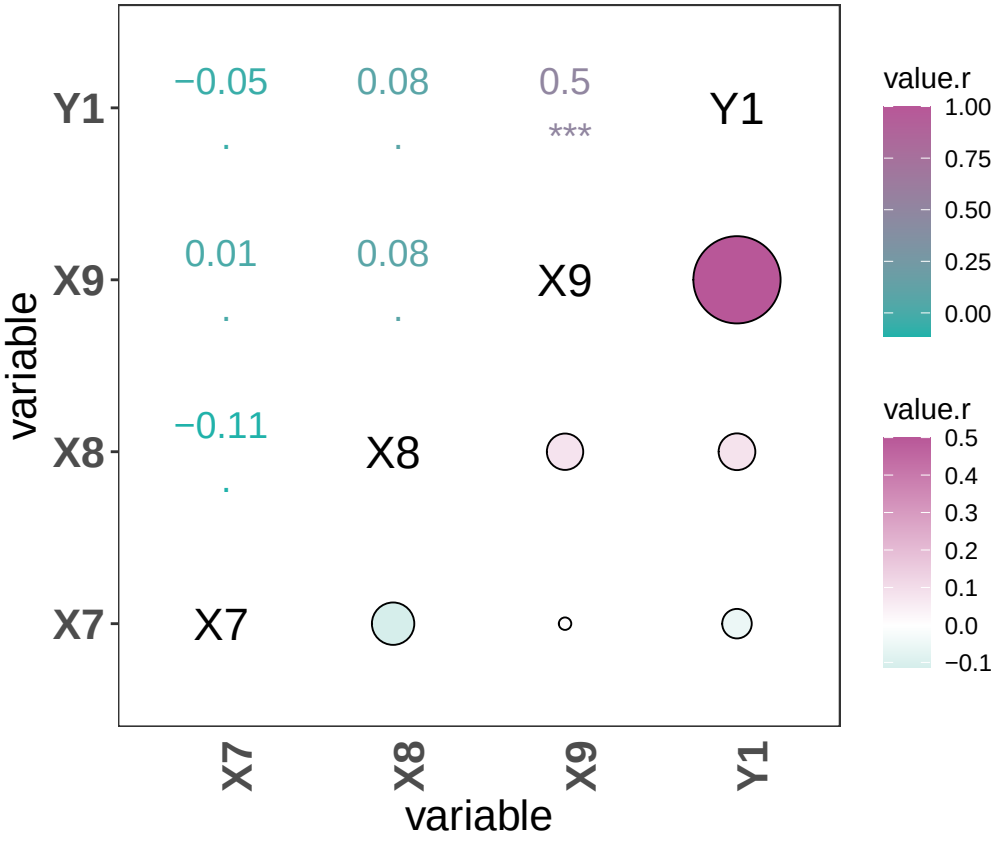
VarsX/VarsBy/Method/Layout/Brightness/Palette parameters can be a character, run `?DescCorr` to see more details.

```
res8 = DescCorr(PID = res$PID,
                OutPath = "default",
                VarsX = "X5,X6,X7,X8,X9",
                VarsY = "Y1",
                VarsBy = "Y1",
                Method = "spearman",
                Layout = "bubble",
                Brightness = "light",
                Palette = "default1")
res8$Corrtable
```

```
## $DescCorr_spearman
## # A tibble: 8 x 7
##   Group Terms Vars        X7      X8       X9      Y1
##   <chr> <chr> <chr>    <dbl>   <dbl>    <dbl>   <dbl>
```

```
## 1 all   r     X7     1        -0.113   0.00767 -0.0524
## 2 all   r     X8    -0.113     1        0.0816   0.0834
## 3 all   r     X9     0.00767  0.0816   1        0.500
## 4 all   r     Y1    -0.0524   0.0834   0.500    1
## 5 all   P     X7    NA        0.0809   0.906    0.418
## 6 all   P     X8     0.0809   NA        0.207    0.197
## 7 all   P     X9     0.906    0.207    NA       0
## 8 all   P     Y1     0.418    0.197    0        NA
```

```
res8$CorrPlot$VizDescCorr_Bubble_light_default1
```



After all the analysis is done, please run the "FuncExit()" function to delete the data uploaded to the server.

```
FuncExit(PID = res$PID)
```

```
## [1] "Success to exit. Thanks for using ExposomeX platform!"
```

6.3 StatCros function

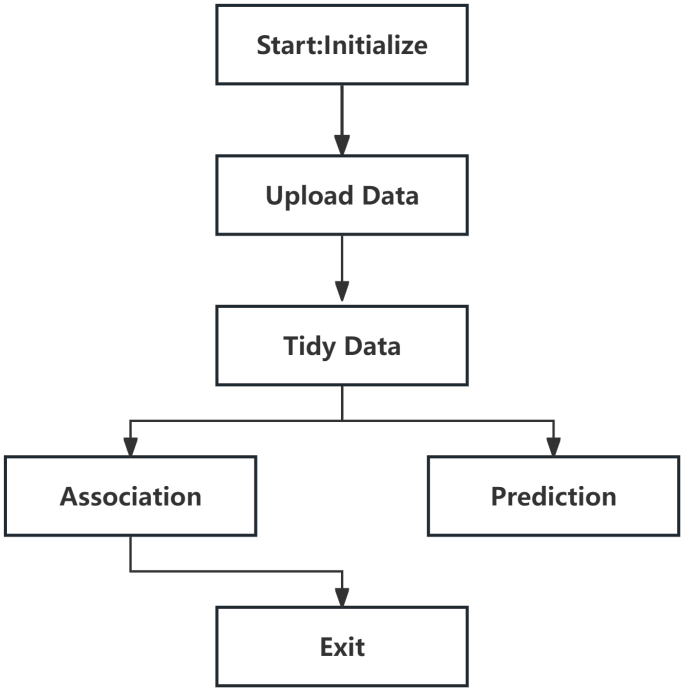
6.3.1 Application domain

StatCros module was designed to analyze the cross-sectional data from exposome-wide association study (EWAS). This data structure can be obtained from the epidemiological designs of cross-section, case-control, and cohort.

6.3.2 Theory

Users can easily get the modeling results and their visualization plots with high quality by following the detailed instructions in each step. For association, the model is chosen according to the data type of health outcome; while for prediction, most of the frequently-used models are evaluated for users' reference.

6.3.3 Work pipeline



6.3.4 Use example

Initialize package

Make sure that the required packages is already installed.

```
library(tidyverse)
# devtools::install_github("ExposomeX/exposomex", force = TRUE)
library(exposomex)
```

At first, you need to initialize the calculation environment using a series of initialization functions, e.g., InitStatCros, InitStatMo, InitStatTidy, InitEVIZ, InitEBIO, etc. Here, we use the module "StatCros" for cross-sectional data analysis for example. The detailed information about the functions and returned value will be introcuced in the following chapters.

```
res = InitStatCros()
res
```

```
## <eSet>
##   Public:
##     AddCommand: function (x)
##     AddLog: function (x)
##     clone: function (deep = FALSE)
##     EpiDesign: NULL
##     ExcecutionLog: Complete initializing the ExpoStat module.2024.06.21 00. ...
##     Expo: list
##     FileDirIn: NULL
##     FileDirOut: /home/ubuntu/ExposomeX/R_Exposome_1.0/output_005154.2732 ...
##     PID: 005154.273217NJUQGU
##     RCommandLog: eSet <- InitCros(PID = Any ID your like, FileDirOut = An ...
##     VarsDel: NULL
```

Here, we can see that the returned value "res" is an R6 object. It contains an unique program ID of res$PID (e.g., "100737GJMWJA"), which is random generated by the system. Users need use it in the following step for further data process.

Upload data

Secondly, you need to load data file for data process. The PID Parameter, you can enter res$PID which is random generated by the system when you run the InitStatCros function. If you want to try this package at first, you can input the UseExample Parameter with "example#1" to use our example data. Or you can enter *UseExample = "default"* to use your own data, you can enter *DataPath = " "* and *VocaPath = " "* to choose you own input data file and vocabulary file directories. The demand of these Parameters can see in the help of *LoadStatCros* function.

You should note that the format of the input data file and vocabulary file is ruled. You can see the demand of the data format by visiting the following website http://www.exposomex.cn.

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitStatCros*.
UseExample
chr. Method of uploading data. If "default", user should upload their own data files,or use "example#1" provided by this module.
DataPath
chr. Input data file directory, e.g. "D:/test/eg_statcros_data.xlsx". It should be noted that the slash symbol is "/", not "\".
VocaPath

chr. Input vocabulary file directory, e.g. "D:/test/eg_statcros_voca.xlsx". It should be noted that the slash symbol is "/", not "\".

```
res1 <- LoadStatCros(PID=res$PID,
                     UseExample = "example#1")

res1$Expo$Voca %>%
  dplyr::slice(1:20)
```

```
## # A tibble: 20 x 6
##    SerialNo SerialNo_Raw FullName GroupName  Type    Lod
##    <chr>    <chr>        <chr>    <chr>      <chr> <dbl>
##  1 Y1       Y1           Y_cont   Outcome    cont   NA
##  2 Y2       Y2           Y_disc   Outcome    cate   NA
##  3 C1       C1           Cov_1    Demography cont   NA
##  4 C2       C2           Cov_2    Demography cont   NA
##  5 C3       C3           Cov_3    Demography cate   NA
##  6 C4       C4           Cov_4    Demography cate   NA
##  7 C5       C5           Cov_5    Demography cate   NA
##  8 C6       C6           Cov_6    Demography cate   NA
##  9 X1       X1           TE_1     Chemical   cont    0.5
## 10 X2       X2           TE_2     Chemical   cont    0.5
## 11 X3       X3           TE_3     Chemical   cate    0.5
## 12 X4       X4           TE_4     Chemical   cont    0.5
## 13 X5       X5           TE_5     Chemical   cont    0.5
## 14 X6       X6           TE_6     Chemical   cont    0.5
## 15 X7       X7           TE_7     Chemical   cont    0.5
## 16 X8       X8           TE_8     Chemical   cont    0.5
## 17 X9       X9           CH1      Chemical   cont    5
## 18 X10      X10          CH2      Chemical   cont    5
## 19 X11      X11          CH3      Chemical   cont    5
## 20 X12      X12          CH4      Chemical   cont    5
```

```
res1$Expo$Data %>%
  dplyr::select(SampleID:X2) %>%
  dplyr::slice(1:20)
```

```
## # A tibble: 20 x 13
##    SampleID SubjectID Group    Y1    Y2    C1    C2    C3    C4    C5    C6
##    <chr>    <chr>     <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 Tr1      S1        train  -101     1  26.9  25.4     3     0     1     3
##  2 Tr2      S2        train   -51     0  30.9  23.9     1     1     2     2
##  3 Tr3      S3        train   -37     0  25.8  23.0     2     3     1     2
##  4 Tr4      S4        train   -61     1  38.0  21.2     1     2     2     3
##  5 Tr5      S5        train   -28     0  31.6  19.5     1     0     2     2
##  6 Tr6      S6        train    -8     0  25.9  20.8     3     3     1     2
##  7 Tr7      S7        train   -63     1  32.4  27.0     2     3     2     1
##  8 Tr8      S8        train   -35     0  33.7  22.1     2     0     2     3
##  9 Tr9      S9        train   -14     0  32.9  19.8     2     2     1     2
## 10 Tr10     S10       train   -99     1  28.5  29.6     1     3     2     2
## 11 Tr11     S11       train   -60     0  37.6  25.3     3     3     1     3
## 12 Tr12     S12       train   -32     0  31.9  23.3     3     0     1     1
## 13 Tr13     S13       train   -73     0  26.9  27.2     3     2     2     3
```

```
## 14 Tr14     S14       train   -18     0  18.9  26.7     2     0     2     3
## 15 Tr15     S15       train   -48     0  35.6  22.1     2     2     2     3
## 16 Tr16     S16       train   -20     0  29.8  30.6     2     0     2     3
## 17 Tr17     S17       train    -9     0  29.9  23.2     1     0     1     2
## 18 Tr18     S18       train   -98     1  34.7  19.7     3     0     1     2
## 19 Tr19     S19       train   -70     0  34.1  23.6     3     0     1     1
## 20 Tr20     S20       train   -36     0  33.0  24.6     1     1     2     3
## # i 2 more variables: X1 <dbl>, X2 <dbl>
```

Here, we can see that the returned value "res1" is an R6 object. It contains two data frames which are your input data. Users need use it in the following step for further data process.

By now, we have prepared our dataset ready for the following calculation.

Tidy data

Tidy the data for following modeling, including deleting missing variables, deleting low variation variables, and transforming dummy variables.

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitStatCros*.
OutPath
chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.
TransDummyVars
chr. Variables to be transformed as dummy variables. It should be noted that there is fixed format for the entering characters separated with comma and without space, e.g., "X1,X2,X3". If "default", all the factor variables will be transformed into dummy ones. These variables need to be transformed as factor ones in previous transform step using TransType function.

```
res2 <- TidyStatCros(PID=res$PID,
                     OutPath="default",
                     TransDummyVars="default")
```

6.4.4 Model: Association

Association model —— CrosAsso is used for association analysis for cross-sectional data.

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitStatCros*.
OutPath
chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.
Linear
lgl. T (or TRUE) and F (or FALSE). Whether the relationship between variables is linear.
EpiDesign
chr. Epidemiological design of the study, including "cohort" "case.control" and "cross.sectional". It doesn't affect the modeling, but the format of the output file. For the three designs, the effect values are usually indicated by RR (relative risk) of cohort, OR (odds ratio) of case-control, and beta value of cross-sectional.

VarsY

chr. Outcome variable used for modeling. Only one variable can be entered.

VarsX

chr. Exposure variable used for modeling. The default option is "all.x" (All exposure variables are included). Users can also choose available variables. It should be noted that there is fixed format for the entering characters separated with comma and without space, e.g., "X1,X2,X3".

VarsN

chr. Choose the single factor or multiple factor model. Available options include "single.factor" and "multiple.factor".

VarsSel

lgl. T (or TRUE) and F (or FALSE). Whether to select the significant variable for the final model. Available options.

VarsSelThr

num. If "VarsSel" = T, provide the selection threshold of the P-value. Three value can be chosen, i.e. 0.05, 0.1, and 0.2.

Covariates

chr. Covariates used for modeling. The default option is "all.c" (All covariates are included).

Family

chr. The link function for the regression model according the data type of outcomes, including "gaussian" for continuous variable, "binomial" for binary variable, and "poisson" for counting variable.

RepMsr

lgl. T (or TRUE) and F (or FALSE). Whether existing repeated measurement of the subjects.

Corstr

chr. If "RepMsr" = T, the generalized estimating equations (GEE) will be used. For GEE, three correlation structure options are "exchangeable" "ar1" "unstructured".

```
res3 = CrosAsso(PID=res$PID,
                OutPath = "default",
                Linear = T,
                EpiDesign = "cohort",
                VarsY = "Y2",
                VarsX = res2$Expo$Voca$SerialNo[20:60],
                VarsN = "single.factor" ,
                VarsSel = F,
                VarsSelThr = 0.1,
                Covariates = "all.c",
                Family = "binomial",
                RepMsr = F,
                Corstr = "ar1")

res3$Y2_single.factor_cohort_binomial
```

```
## # A tibble: 41 x 13
##    SerialNo Vars.dummy beta.value    ci_l     ci_h     p.value std.error formula
##    <chr>    <chr>           <dbl>   <dbl>    <dbl>       <dbl>     <dbl> <chr>
##  1 X9       X9              2.13   1.33    2.93    0.000000192     0.409 Y2~C1+~
##  2 X10      X10            -1.35  -2.01   -0.690   0.0000596       0.336 Y2~C1+~
##  3 X11      X11            -1.04  -1.66   -0.412   0.00116         0.320 Y2~C1+~
##  4 X12      X12            -0.831 -1.65   -0.00797 0.0478          0.420 Y2~C1+~
##  5 X13      X13             0.572  0.159   0.985   0.00668         0.211 Y2~C1+~
##  6 X14      X14            -1.41  -2.15   -0.660   0.000222        0.381 Y2~C1+~
##  7 X15      X15            -1.58  -2.22   -0.945   0.00000119      0.326 Y2~C1+~
##  8 X16      X16            -0.808 -1.28   -0.336   0.000800        0.241 Y2~C1+~
##  9 X17      X17             0.365 -0.0630  0.794   0.0946          0.219 Y2~C1+~
```

```
## 10 X18      X18            -1.48  -2.08   -0.875   0.00000144      0.306 Y2~C1+~
## # i 31 more rows
## # i 5 more variables: RR <dbl>, RR_CI_L <dbl>, RR_CI_H <dbl>, FullName <chr>,
## #   GroupName <chr>
```

Here, we can see that the returned value "res3" contains a result table which includes the association analysis results.

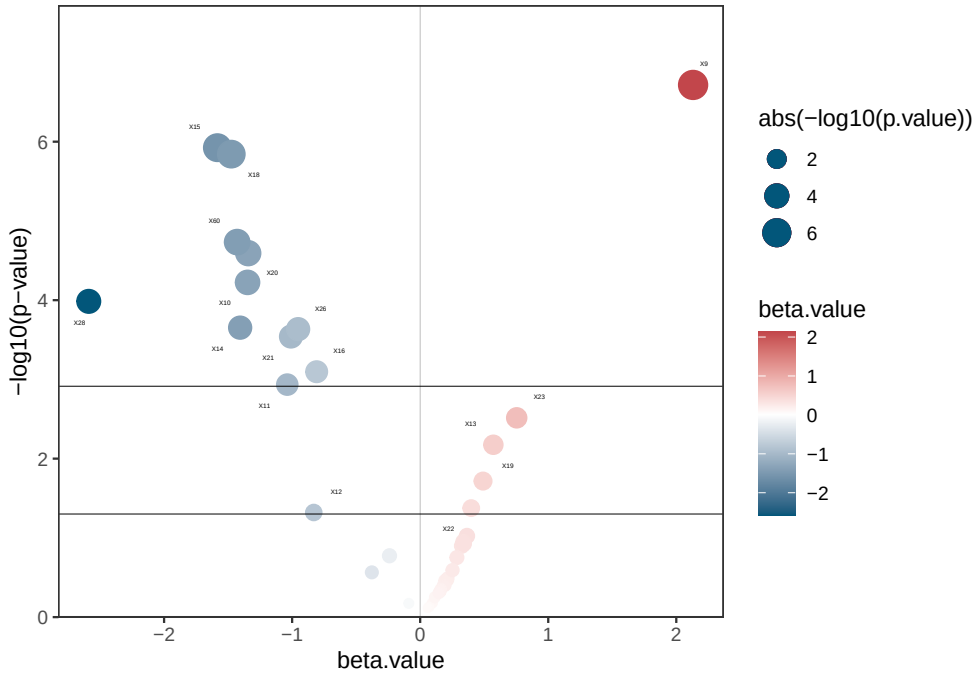
Visualize association —— VizCrosAsso is used to visualize association analysis.

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitStatCros*.
OutPath
chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.
VarsY
chr. Outcome variable used for modeling. Only one variable can be entered.
VarsN
Choose the single factor or multiple factor model. Available options include "single.factor" and "multiple.factor".
Layout
chr. Visualization layout. Available options include "forest" and "volcano".
Brightness
chr. Visualization brightness. Available options include "light" and "dark".
Palette
chr. Visualization palette. Available options include "default1", "default2" and several journal preference styles including "cell", "nature", "science", "lancet", "nejm", and "jama".
ColorFor
chr. Parameter to represent the color of the points in the output plot. Available options include "p.value" and "beta.value".
SizeFor
chr. Parameter to represent the size of the points in the output plot. Available options include "p.value" and "beta.value".

```
res4 = VizCrosAsso(PID=res$PID,
                   OutPath="default",
                   VarsY="Y2",
                   VarsN="single.factor",
                   Layout="volcano",
                   Brightness="dark",
                   Palette="default1",
                   ColorFor="beta.value",
                   SizeFor="p.value")
res4
```

```
## $Y2_single.factor_volcano_dark_default1_beta.value_p.value
```



Model: Prediction

Prediction model —— CrosPred is used to build prediction models.

The details for each parameter are listed below:

PID

chr. Program ID. It must be the same with the PID generated by *InitStatCros*.

OutPath

chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.

VarsY

chr. Outcome variable used for modeling. Only one variable can be entered.

VarsX

chr. Exposure variable used for modeling. The default option is "all.x" (All exposure variables are included). Users can also choose available variables. It should be noted that there is fixed format for the entering characters separated with comma and without space, e.g., "X1,X2,X3".

PredType

chr. Prediction type of the outcome variable, including "response" for the actual values and "prob" for outcome with binary variable.

VarsSel

lgl. Whether to select the significant variable for the final model.Available options include T (or TRUE) and F (or FALSE).

VarsSelThr

num. If "VarsSel" = T, provide the selection threshold of the P-value. Three value can be chosen, i.e. 0.05, 0.1, and 0.2.

Covariates

chr. Covariates used for modeling. The default option is "all.c" (All covariates are included).

RsmpMethod

chr. Four resampling methods options for internal validation, including "cv" (i.e.,Cross validation) , "loo" (i.e., eave-one-out), "bootstrap", and "holdout".

Folds
num. Folds of Cross-validation resampling. It is ranging 2-10.
Ratio
num. Ratio of Bootstrap resampling. It is ranging 0.4-0.9.
Repeats
num. Number of Bootstrap resampling. It is ranging 2-20.

```
res5 = CrosPred(PID=res$PID,
                OutPath = "default",
                VarsY = "Y2",
                VarsX = res2$Expo$Voca$SerialNo[20:60],
                PredType = "prob",
                VarsSel = F,
                VarsSelThr = 0.1,
                Covariates = "all.c",
                RsmpMethod = "cv" ,
                Folds = 5 ,
                Ratio = 0.667,
                Repeats = 5)

res5$Y2_cv
```

```
## # A tibble: 12 x 17
##    learner_id          ce_rsmp_train_mean ce_rsmp_test_mean auc_rsmp_mean
##    <chr>                            <dbl>             <dbl>         <dbl>
##  1 classif.cforest                 0.0683             0.147         0.951
##  2 classif.earth                   0.0367             0.167         0.857
##  3 classif.gbm                     0.01               0.12          0.943
##  4 classif.glmboost                0.0717             0.107         0.962
##  5 classif.kknn                    0.045              0.187         0.863
##  6 classif.ksvm                    0.0317             0.107         0.935
##  7 classif.lda                     0.0267             0.12          0.904
##  8 classif.multinom                0                  0.173         0.820
##  9 classif.naive_bayes             0.122              0.153         0.916
## 10 classif.nnet                    0.152              0.213         0.753
## 11 classif.ranger                  0                  0.153         0.936
## 12 classif.rpart                   0.0883             0.187         0.855
## # i 13 more variables: ce_rsmp_train_sd <dbl>, ce_rsmp_test_sd <dbl>,
## #   auc_rsmp_sd <dbl>, ce_train <dbl>, acc_train <dbl>, sens_train <dbl>,
## #   spec_train <dbl>, auc_train <dbl>, ce_test <dbl>, acc_test <dbl>,
## #   sens_test <dbl>, spec_test <dbl>, auc_test <dbl>
```

Here, we can see that the returned value "res5" contains a result table which includes the prediction performance evaluation.

Visualize prediction performance —— VizCrosPred is used to visualize prediction analysis.

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitStatCros*.
OutPath
chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If

“default”, the current working directory will be set.

VarsY

chr. Outcome variable used for modeling. Only one variable can be entered.

Layout

chr. Visualization layout. Available options include “bar” and “roc”.

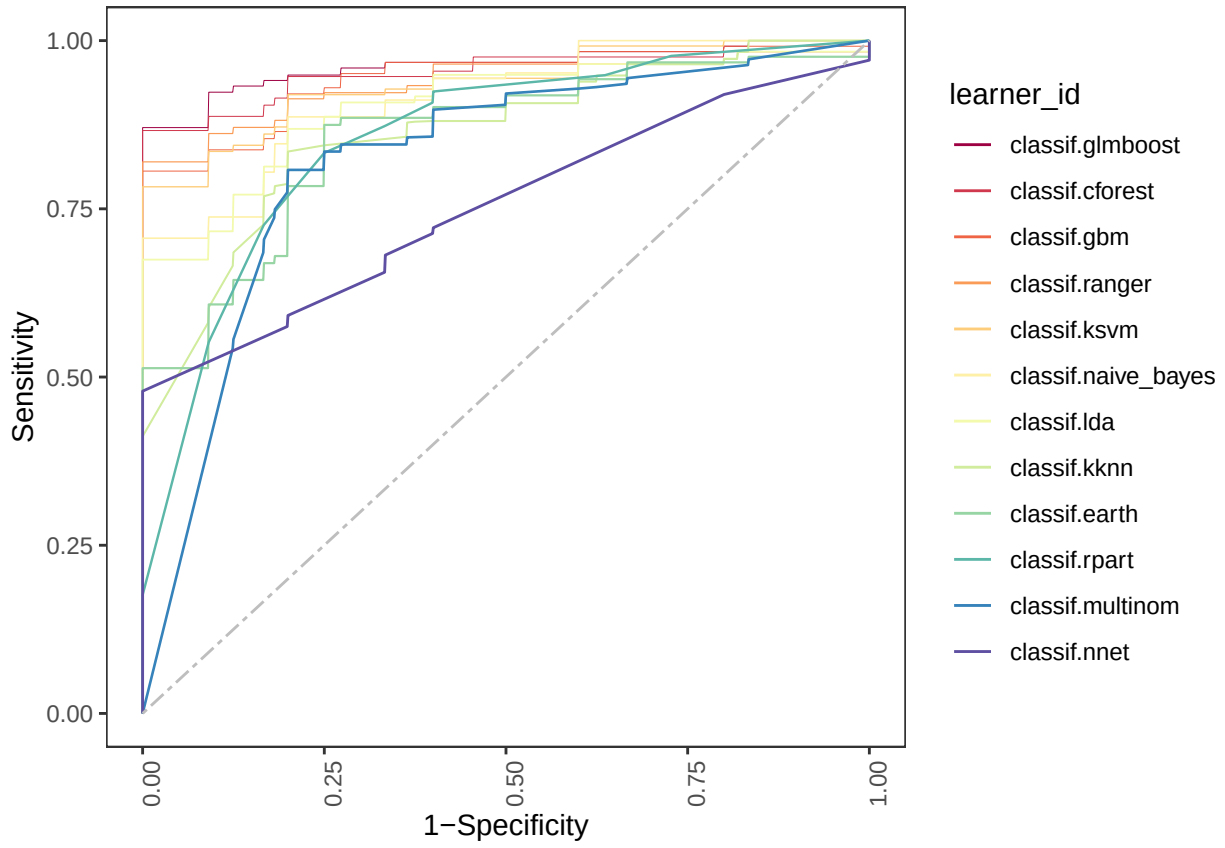
Brightness

chr. Visualization brightness. Available options include “light” and “dark”.

Palette

chr. Visualization palette. Available options include “default1”, “default2” and several journal preference styles including “cell”, “nature”, “science”, “lancet”, “nejm”, and “jama”.

```
res6 = VizCrosPred(PID=res$PID,
                   OutPath="default",
                   VarsY="Y2",
                   Layout="roc",
                   Brightness="light",
                   Palette="default1")
res6$Y2_roc_light_default1
```




```
FuncExit(PID = res$PID)
```

```
## [1] "Success to exit. Thanks for using ExposomeX platform!"
```

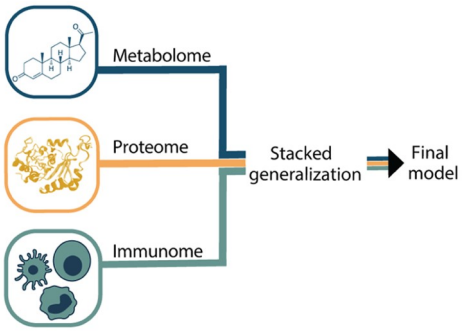
6.4 StatMO function

6.4.1 Application domain

Multi-omics research is a method of exploring the interactions between multiple substances in biological systems, including genomics, proteomics, metabolomics, etc., which together affect the phenotype and traits of organisms. With the rapid development of high-throughput sequencing and multiomics, we can develop more comprehensive predictive models and identify important features by integrating multi-omic data, which facilitates a deeper understanding of the complexity of biology. ***StatMO*** package is designed to integrate the multi-omic data. It mainly aims to construct various stacked generalization models to predict the response or probability of outcome incidence, as well as providing the statistical explanation. Additionly, the package can provide visualization plots with high quality of the final calculation results to make it easier for users to understand.

6.4.2 Theory

The theory of the package ***StatMO*** is based on "Integrated trajectories of the maternal metabolome, proteome, and immunome predict labor onset". And the code for machine learning and visualization of the results are based on "mlr3" package and "ggplot2" package.



Stelzer IA, Ghaemi MS, Han X, et al. Integrated trajectories of the maternal metabolome, proteome, and immunome predict labor onset. Sci Transl Med. 2021;13(592):eabd9898. doi:10.1126/scitranslmed.abd9898

6.4.3 Work pipeline

Initialize package

Make sure that the required packages is already installed.

```
library(tidyverse)
# devtools::install_github("ExposomeX/exposomex", force = TRUE)
library(exposomex)
```

The first step,you need to initialize the environment using "InitStatMO".

```
res = InitStatMO()
```

Here, we can see that the returned value "res" is an R6 object. It contains an unique program ID of res$PID (e.g., "100737GJMWJA"), which is random generated by the system. Users need use it in the following step for further data process.

Upload data

```
res1 = LoadStatMO(PID = res$PID,
              UseExample="example#1")
```

Tidy data

```
res2 = TidyStatMO(PID=res$PID,
                  OutPath = "default",
                  Vars = "",
                  To = "")
```

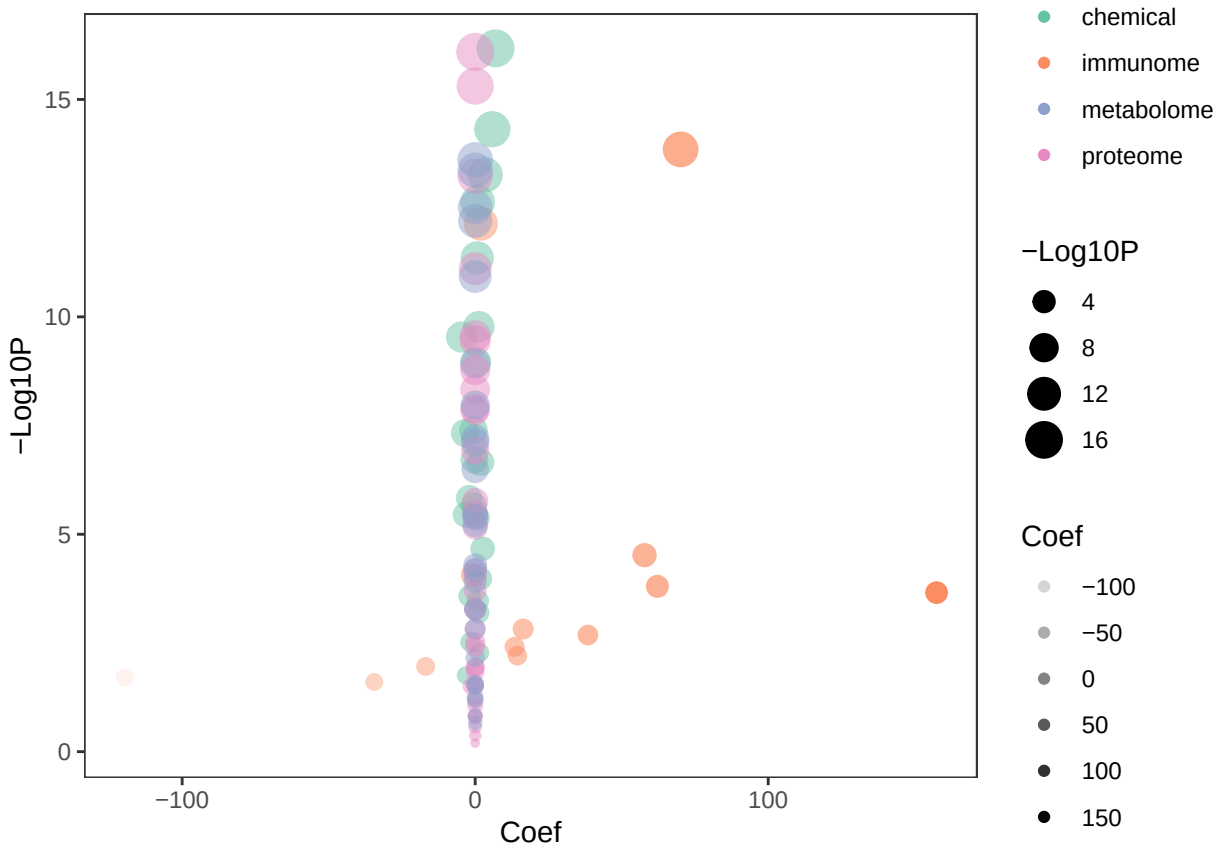
Modeling This step is the most critical part of the entire module. we will build multiomics model with function "MulOmicsCros".You can select one or more arbitrary learning methods in parameter SG_Lrns. Here we choose lasso(least absolute shrinkage and selection operator) and rf(random forest) as examples. The calculation time depends on the characteristics of your data, the number of learning methods, and the tuning method. For parameter TuneMethod, the default option can provide faster calculations but less accurate results than other autotune methods. If you want to train a better model, choose other auto-tune method and increase the number of tuning times.

```
res3 = MulOmicsCros(PID=res$PID,
                    OutPath = "default",
                    OmicGroups = "proteome,chemical,immunome,metabolome",
                    VarsY = "Y1",
                    VarsC = "all.c",
                    TuneMethod = "default",
                    TuneNum = 20,
                    RsmpMethod = "cv",
                    Folds = 5,
                    Ratio = 0.67,
                    Repeats = 5,
                    VarsImpThr = 0.85,
                    SG_Lrns ="lasso,rf")
res3$MulOmics$SGplot
```

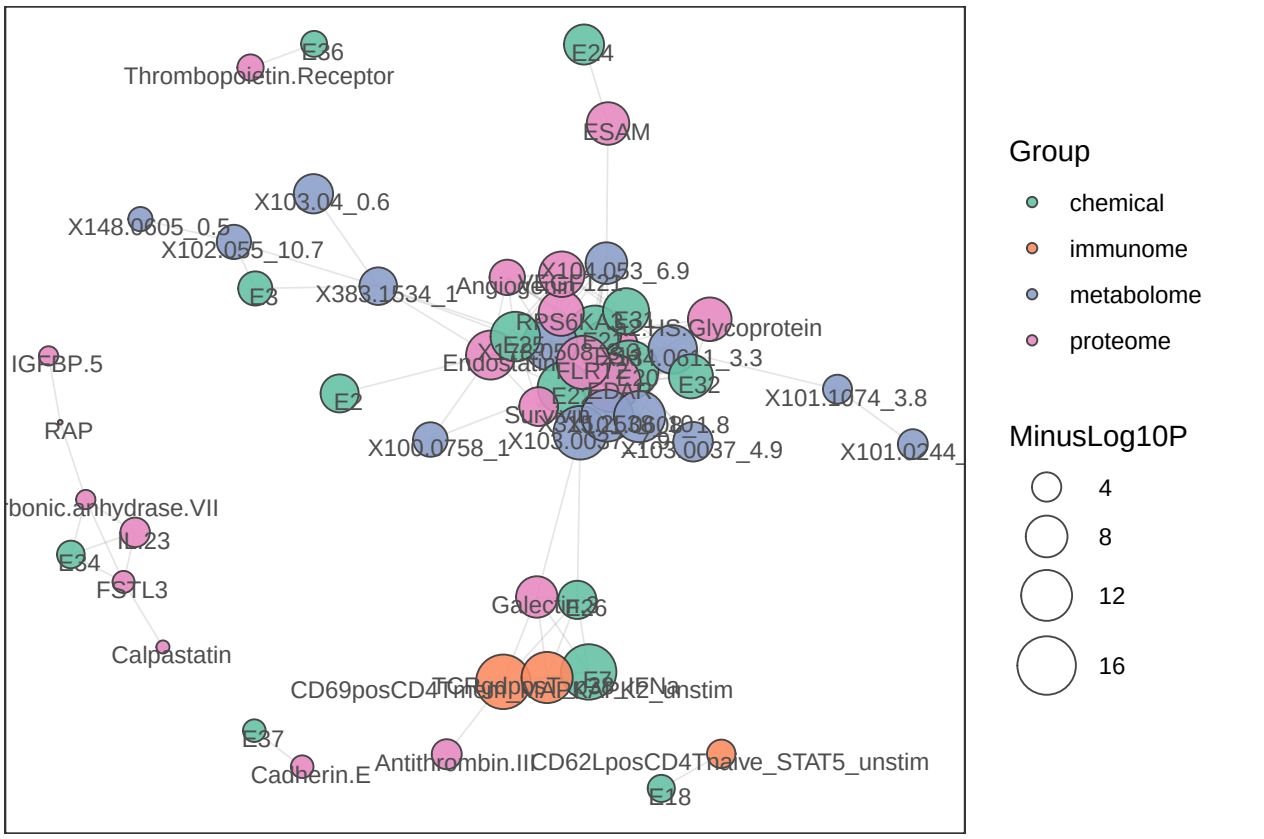
```
## NULL
```

Visualize model This step will visualize multi-omic data with function "VizMultOmic". You can get different styles of images by selecting different parameters.

```
res4 = VizMultOmic(PID=res$PID,
                   OutPath = "default",
                   VarsY = "Y1",
                   NodeNum=100,
                   EdgeThr= 0.45,
                   Layout = "force-directed",
                   Brightness = "light",
                   Palette = 'default1')
res4$Nodeplot$lasso
```




```
res4$Networkplot$lasso
```




```
FuncExit(PID = res$PID)
```

```
## [1] "Success to exit. Thanks for using ExposomeX platform!"
```

6.5 StatPanel function

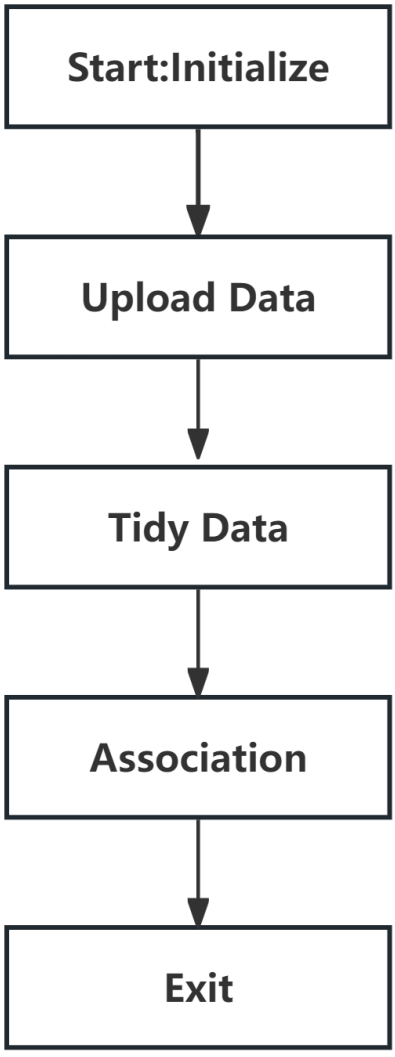
6.5.1 Application domain

StatPanel function is designed to conduct the analysis of the panel data. It mainly aims to evaluate the associations between exposure factors and the health outcome.

6.5.2 Theory

Users can easily get the modeling results and their visualization plots with high quality by following the detailed instructions in each step. Linear mixed effect model is adopted to evaluate the association.

6.5.3 Work pipeline



6.5.4 Use example

Initialize package

Make sure that the required packages is already installed.

```
library(tidyverse)
library(gridExtra)
# devtools::install_github("ExposomeX/exposomex", force = TRUE)
library(exposomex)
```

At first, you need to initialize the calculation environment using a series of initialization functions, e.g., InitStatPanel, InitStatMo, InitStatTidy, InitEVIZ, InitEBIO, etc. Here, we use the "StatPanel" for panel data analysis for example. The detailed information about the functions and returned value will be introcuced in the following chapters.

```
res = InitStatPanel()
res
```

```
## <eSet>
##   Public:
##     AddCommand: function (x)
##     AddLog: function (x)
##     clone: function (deep = FALSE)
##     EpiDesign: NULL
##     ExcecutionLog: Complete initializing the ExpoStat module.2024.06.23 15. ...
##     Expo: list
##     FileDirIn: NULL
##     FileDirOut: /home/ubuntu/ExposomeX/R_Exposome_1.0/output_152744.3086 ...
##     PID: 152744.308686AGPBBR
##     RCommandLog: eSet <- InitPanel()
##     VarsDel: NULL
```

Here, we can see that the returned value "res" is an R6 object. It contains an unique program ID of res$PID (e.g., "100737GJMWJA"), which is random generated by the system. Users need use it in the following step for further data process.

Upload data

Secondly, you need to load data file for data process. The PID Parameter, you can enter res$PID which is random generated by the system when you run the InitStatPanel function. If you want to try this package at first, you can input the UseExample Parameter with "example#1" to use our example data. Or you can enter *UseExample = "default"* to use your own data, you can enter *DataPath = " "* and *VocaPath = " "* to choose you own input data file and vocabulary file directories. The demand of these Parameters can see in the help of *LoadStatPanel* function.

You should note that the format of the input data file and vocabulary file is ruled. You can see the demand of the data format by visiting the following website http://www.exposomex.cn.

The details for each parameter are listed below:

PID
: chr. Program ID. It must be the same with the PID generated by *InitStatPanel*.

UseExample

chr. Method of uploading data. If “default”, user should upload their own data files,or use “example#1” provided by this module.

DataPath

chr. Input data file directory, e.g. “D:/test/eg_statcros_data.xlsx”. It should be noted that the slash symbol is “/”, not “\”.

VocaPath

chr. Input vocabulary file directory, e.g. “D:/test/eg_statcros_voca.xlsx”. It should be noted that the slash symbol is “/”, not “\”.

```
res1 <- LoadStatPanel(PID = res$PID,
                      UseExample = "example#1")

res1$Expo$Voca %>%
  dplyr::slice(1:20)
```

```
## # A tibble: 20 x 5
##    SerialNo SerialNo_Raw FullName                          GroupName Type
##    <chr>    <chr>        <chr>                             <chr>     <chr>
##  1 Y1       Y1           TL_cat2                           Outcome   cate
##  2 Y2       Y2           TL                                Outcome   cont
##  3 C1       C1           cMCs_NFkB_unstim                  immunome  cont
##  4 C2       C2           CCR2poscMCs_STAT3_unstim          immunome  cont
##  5 C3       C3           MDSCs_STAT3_unstim                immunome  cont
##  6 X1       X1           DCs_S6_unstim                     immunome  cont
##  7 X2       X2           DCs_MAPKAPK2_unstim               immunome  cont
##  8 X3       X3           DCs_NFkB_unstim                   immunome  cont
##  9 X4       X4           CCR2negncMCs_NFkB_unstim          immunome  cont
## 10 X5       X5           NK_STAT5_unstim                   immunome  cont
## 11 X6       X6           CD69negCD56loCD16negNK_S6_unstim  immunome  cont
## 12 X7       X7           CD69posCD56loCD16negNK_S6_unstim  immunome  cont
## 13 X8       X8           CD4Tcells_ERK_unstim              immunome  cont
## 14 X9       X9           CD4Tcells_STAT6_unstim            immunome  cont
## 15 X10      X10          CD4posTnaive_MAPKAPK2_unstim      immunome  cont
## 16 X11      X11          CCR5posCCR2posCD4Tem_ERK_unstim   immunome  cont
## 17 X12      X12          Th1_ERK_unstim                    immunome  cont
## 18 X13      X13          CD8Tmem_MAPKAPK2_unstim           immunome  cont
## 19 X14      X14          CD8Tem_S6_unstim                  immunome  cont
## 20 X15      X15          CD8Tem_IkB_unstim                 immunome  cont
```

```
res1$Expo$Data %>%
  dplyr::select(SampleID:X2) %>%
  dplyr::slice(1:20)
```

```
## # A tibble: 20 x 9
##    SampleID SubjectID    Y1    Y2     C1    C2    C3    X1      X2
##    <chr>        <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>   <dbl>
##  1 Tr1              1     1   106 0.0473 0.623 0.593 0.234 0
##  2 Tr2              1     0    56 0.0392 0.276 0.186 0.189 0
##  3 Tr3              1     0    42 0.0526 0.423 0.343 0.225 0
##  4 Tr4             16     1    66 0.0860 0.557 0.608 0.290 0
##  5 Tr5             16     0    33 0.109  0.405 0.385 0.322 0
##  6 Tr6             16     0    13 0.0886 0.433 0.458 0.307 0.0103
```

```
##  7 Tr7             28     1    68 0.0826 0.509 0.536 0.291 0.0168
##  8 Tr8             28     0    40 0.0979 0.382 0.392 0.328 0.00480
##  9 Tr9             28     0    19 0.0904 0.346 0.338 0.305 0.00966
## 10 Tr10            36     1   104 0.122  0.378 0.529 0.280 0.0840
## 11 Tr11            36     0    65 0.116  0.369 0.505 0.257 0.0744
## 12 Tr12            36     0    37 0.178  0.534 0.721 0.331 0.102
## 13 Tr13            41     0    78 0.0911 0.231 0.202 0.268 0
## 14 Tr14            41     0    23 0.0835 0.305 0.317 0.271 0
## 15 Tr15            46     0    53 0.110  0.348 0.345 0.373 0.0417
## 16 Tr16            46     0    25 0.0912 0.491 0.427 0.321 0.0453
## 17 Tr17            46     0    14 0.0409 0.256 0.322 0.310 0.0600
## 18 Tr18             3     1   103 0.120  0.287 0.303 0.356 0.0638
## 19 Tr19             3     0    75 0.0997 0.280 0.316 0.274 0.0340
## 20 Tr20             3     0    41 0.131  0.299 0.337 0.343 0.0828
```

Here, we can see that the returned value "res1" is an R6 object. It contains two data frames which are your input data. Users need use it in the following step for further data process.

By now, we have prepared our dataset ready for the following calculation.

Tidy data

Tidy the data for following modeling, including deleting missing variables, deleting low variation variables, and transforming dummy variables.

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitStatPanel*.
OutPath
chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.
TransDummyVars
chr. Variables to be transformed as dummy variables. It should be noted that there is fixed format for the entering characters separated with comma and without space, e.g., "X1,X2,X3". If "default", all the factor variables will be transformed into dummy ones.

```
res2 <- TidySataPanel(PID = res$PID,
                      OutPath = "default",
                      TransDummyVars = "default")
```

Model: Association

Association model —— PanelAsso is used for association analysis for panel data.

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitStatPanel*.
OutPath
chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.
VarsY

chr. Outcome variable used for modeling. Only one variable can be entered.

VarsX

chr. Exposure variable used for modeling. The default option is "all.x" (All exposure variables are included). Users can also choose available variables. It should be noted that there is fixed format for the entering characters separated with comma and without space, e.g., "X1,X2,X3".

VarsN

chr. Choose the single factor or multiple factor model. Available options include "single.factor" and "multiple.factor".

VarsRandomIpt

chr. Random intercept variable for the linear mixed-effect model. The default is "SubjectID".

VarsRandomSlp

chr. Random slope variable for the linear mixed-effect model. The default is "none". It should be noted that there is fixed format for the entering characters separated with comma and without space, e.g., "X1,X2,X3".

Covariates

chr. Covariates used for modeling. The default option is "all.c" (All covariates are included).

```
res3 = PanelAsso(PID = res$PID,
                 OutPath = "default",
                 VarsY = "Y1",
                 VarsX = "X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,X11,X12",
                 VarsN = "multiple.factor",
                 VarsRandomIpt = "SubjectID",
                 VarsRandomSlp = "none",
                 Covariates = "all.c")

res3$Y1_multiple.factor
```

```
## # A tibble: 12 x 6
##    vars      beta    se p.value beta_lcl beta_hcl
##    <chr>    <dbl> <dbl>   <dbl>    <dbl>    <dbl>
##  1 X1      0.223   3.26  0.945     -6.16    6.61
##  2 X2      7.83   10.7   0.466    -13.2    28.9
##  3 X3     -1.40    5.71  0.806    -12.6     9.80
##  4 X4     -7.90    3.68  0.0318   -15.1    -0.688
##  5 X5      0.0269  4.78  0.996     -9.35    9.40
##  6 X6     -2.37    7.19  0.742    -16.5    11.7
##  7 X7     -0.640   4.87  0.896    -10.2     8.91
##  8 X8     49.1    27.5   0.0736    -4.69  103.
##  9 X9    -37.5    15.1   0.0132   -67.1    -7.83
## 10 X10    12.1     6.97  0.0826    -1.56   25.8
## 11 X11    12.9     8.60  0.134     -3.96   29.7
## 12 X12     1.87    6.87  0.785    -11.6    15.3
```

Here, we can see that the returned value "res3" contains a result table which includes the association analysis results.

Visualize association —— VizPanelAsso is used to visualize association analysis.

The details for each parameter are listed below:

PID

chr. Program ID. It must be the same with the PID generated by *InitStatPanel*.

OutPath

chr. Output file directory, e.g. “D:/test”. It should be noted that the slash symbol is “/”, not “\”. If “default”, the current working directory will be set.

VarsY

chr. Outcome variable used for modeling. Only one variable can be entered.

VarsN

Choose the single factor or multiple factor model. Available options include “single.factor” and “multiple.factor”.

EffectThr

num. The threshold to distinguish the effect value of the volcano plot. It should be determined according to the range of the effect value (i.e. beta value), as well as the users’ preference.

Layout

chr. Visualization layout. Available options include “forest” and “volcano”.

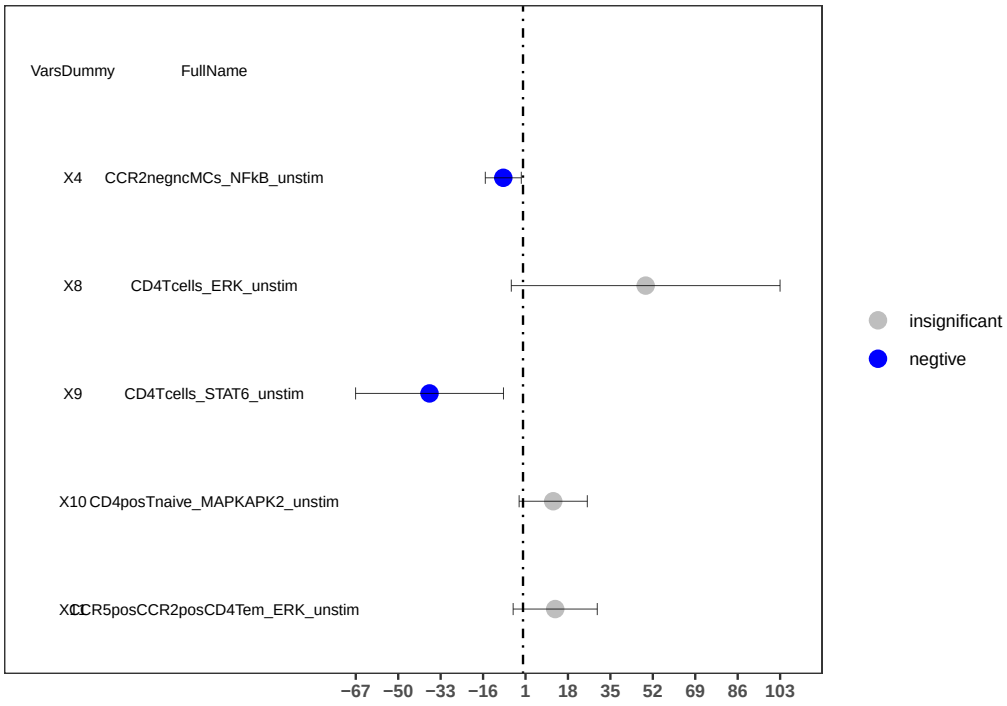
Brightness

chr. Visualization brightness. Available options include “light” and “dark”.

Palette

chr. Visualization palette. Available options include “default1”, “default2” and several journal preference styles including “cell”, “nature”, “science”, “lancet”, “nejm”, and “jama”.

```
res4 = VizPanelAsso(PID = res$PID,
                    OutPath = "default",
                    VarsY = "Y1",
                    VarsN = "multiple.factor" ,
                    EffectThr = 0.5,
                    Layout = "forest",
                    Brightness = "light",
                    Palette = "default1")
res4$forest_Y1_multiple.factor_forest_light_default1
```



```
FuncExit(PID = res$PID)
```

```
## [1] "Success to exit. Thanks for using ExposomeX platform!"
```

6.6 StatSurv function

6.6.1 Application domain

StatSurv is designed to conduct the survival analysis of the censored data. It mainly aims to evaluate the associations between exposure factors and health outcome, as well as predicting the survival probability. Users can easily get the modeling results and their visualization plots with high quality by following the detailed instructions in each step.

6.6.2 Theory

In "SurvAsso()" function, Cox proportional hazards regression model is used to calculate the hazard ratio (HR). In "SurvPred()" function, various machine learners are used to conduct the prediction analysis, including coxph, coxboost, xgboost, ranger, cforest, ctree, nelson, etc., from mlr3 R package.

6.6.3 Work pipeline

Install packages

To use StatSurv, users can install the "exposomex" package.

```
# The following two packages should be installed in advance
library(tidyverse)
# devtools::install_github("ExposomeX/exposomex", force = TRUE)
library(exposomex)
```

Initialize

At first, you need to initialize the calculation environment using initialization function: "InitStatSurv()".

```
res = InitStatSurv()
res
```

```
## <eSet>
##   Public:
##     AddCommand: function (x)
##     AddLog: function (x)
##     clone: function (deep = FALSE)
##     ExcecutionLog: Complete initializing the Exoverse module.2024-06-23 15: ...
##     Expo: list
##     ExpoDel: list
##     FileDirOut: /home/ubuntu/ExposomeX/R_Exposome_1.0/output_153447.0854 ...
##     Func: list
##     PID: 153447.085473XJUWSY
##     RCommandLog: eSet <- InitExpoData(PID = Any ID your like, FileDirOut  ...
```

Here, we can see that the returned value "res" is an R6 object. It contains an unique program ID of res$PID (e.g., "100737GJMWJA"), which is random generated by the system. Users need use it in the following step for further data process.

Upload data

Then, you need to upload data for analysis by runing "LoadStatSurv()" function. The function includes four parameters:

1. PID: chr. Program ID. It must be the same with the PID generated by InitSurv.
2. UseExample: chr. Method of uploading data. If "default", user should upload their own data files, or use "example#1" provided by this module.
3. DataPath: chr. Input data file directory, e.g. "D:/test/eg_Surv_data.xlsx".
4. VocaPath: chr. Input vocabulary file directory, e.g. "D:/test/eg_Surv_voca.xlsx".

Noted that there are several data format requirments for the data and vocabulary file. For data file,the first three columns should be named as "SampleID", "SubjectID", and "Group", respectively. For the "Group" variable, only two values can be used, i.e. "train" and "test". If there is no data for test, all values should be set as "train". For outcome variables, their initials must be set as "Y" and serialized by adding Arabic numerals if needed, e.g., Y1, Y2, Y3. In this module, the survival time (Y1) and status (Y2) must be provided.For exposure variables, their initials must be set as "X" and serialized by adding Arabic numerals if needed, e.g., X1, X2, X3. For covariate variables, their initials must be set as "C" and serialized by adding Arabic numerals if needed, e.g., C1, C2, C3. It should be noted the covariates are not required if users don't have. For vocabulary file, the first two columns must be named as "SerialNo" and "FullName", respectively. The list of SerialNo of outcomes, exposure, and covariates should be the same with the column names of "Data file". The list of the FullName is prepared as users' like.

Here, we use "example#1" data for demonstration:

```
res1 <- LoadStatSurv(PID = res$PID, UseExample = "example#1", DataPath = NULL,
   VocaPath = NULL)
res1$Expo$Data
```

```
## # A tibble: 394 x 33
##    SampleID SubjectID Group    Y1    Y2    C1    C2    C3    X1     X2      X3
##       <dbl>     <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>   <dbl>
##  1        1         1 train   306     2  47.8    74     1     1  0.658  1.37
##  2        2         2 train   455     2  44      68     1     0  0.658  0.695
##  3        3         3 train  1010     1  42      56     1     0  0.658  0.695
##  4        4         4 train   210     2  36.7    57     1     1  0.658 -1.33
##  5        5         5 train   883     2  41.8    60     1     0  1.47   0.695
##  6        6         6 train  1022     1  34.3    74     1     1 -2.59   0.0207
##  7        7         7 train   310     2  35.3    68     2     2 -0.964 -1.33
##  8        8         8 train   361     2  36      71     2     2 -1.78   0.0207
##  9        9         9 train   218     2  36.8    53     1     1 -0.964  0.0207
## 10       10        10 train   166     2  39.7    61     1     2 -0.964 -0.654
## # i 384 more rows
## # i 22 more variables: X4 <dbl>, X5 <dbl>, X6 <dbl>, X7 <dbl>, X8 <dbl>,
## #   X9 <dbl>, X10 <dbl>, X11 <dbl>, X12 <dbl>, X13 <dbl>, X14 <dbl>, X15 <dbl>,
## #   X16 <dbl>, X17 <dbl>, X18 <dbl>, X19 <dbl>, X20 <dbl>, X21 <dbl>,
## #   X22 <dbl>, X23 <dbl>, X24 <dbl>, X25 <dbl>
```

```
res1$Expo$Voca
```

```
## # A tibble: 30 x 5
##    SerialNo SerialNo_Raw FullName                      GroupName Type
##    <chr>    <chr>        <chr>                         <chr>     <chr>
##  1 Y1       Y1           time                          outcome   cont
##  2 Y2       Y2           status                        outcome   cate
##  3 C1       C1           conf                          chemical  cont
##  4 C2       C2           age                           chemical  cont
##  5 C3       C3           sex                           chemical  cate
##  6 X1       X1           ph.ecog                       chemical  cate
```

```
##  7 X2       X2          ph.karno                          chemical  cont
##  8 X3       X3          pat.karno                         chemical  cont
##  9 X4       X4          wt.loss                           chemical  cont
## 10 X5       X5          CD69negCD56loCD16negNK_S6_unstim immunome  cont
## # i 20 more rows
```

Tidy data

```
res2 = TidyStatSurv(PID = res$PID,
                    OutPath = "default",
                    TransDummyVars="default")
```

Association

Now, users can calculate the hazard ratio(HR) of exposures variables by running "SurvAsso()" function. The function includes eight parameters:
1. PID: chr. Program ID. It must be the same with the PID generated by InitSurv.
2. OutPath: chr. Output file directory, e.g. "D:/test". If "default", the current working directory will be set.
3. TimeY: chr. Outcome variable of survival time used for modelling. Only one variable can be entered.
4. EventY: chr. Outcome variable of status used for modelling. Only one variable can be entered.
5. VarsX: chr. Exposure variable used for modeling. The default option is "all.x" (All exposure variables are included). Users can also choose available variables. It should be noted that there is fixed format for the entering characters separated with comma and without space, e.g., "X1,X2,X3"
6. VarsN: chr. Choose the single factor or multiple factor model. Available options include "single.factor" and "multiple.factor"
7. VarsSel: lgl. T (or TRUE) and F (or FALSE). Whether to select the significant variable for the final model. Available options includes T and F.
8. Covariates chr. Covariates used for modeling. The default option is "all.c" (All covariates are included).

```
res3 = SurvAsso(PID = res$PID,
                OutPath = "default",
                TimeY = "Y1",
                EventY= 'Y2',
                VarsX='all.x',
                VarsN="single.factor",
                VarsSel=T,
                Covariates = "all.c")
res3$coxph_res
```

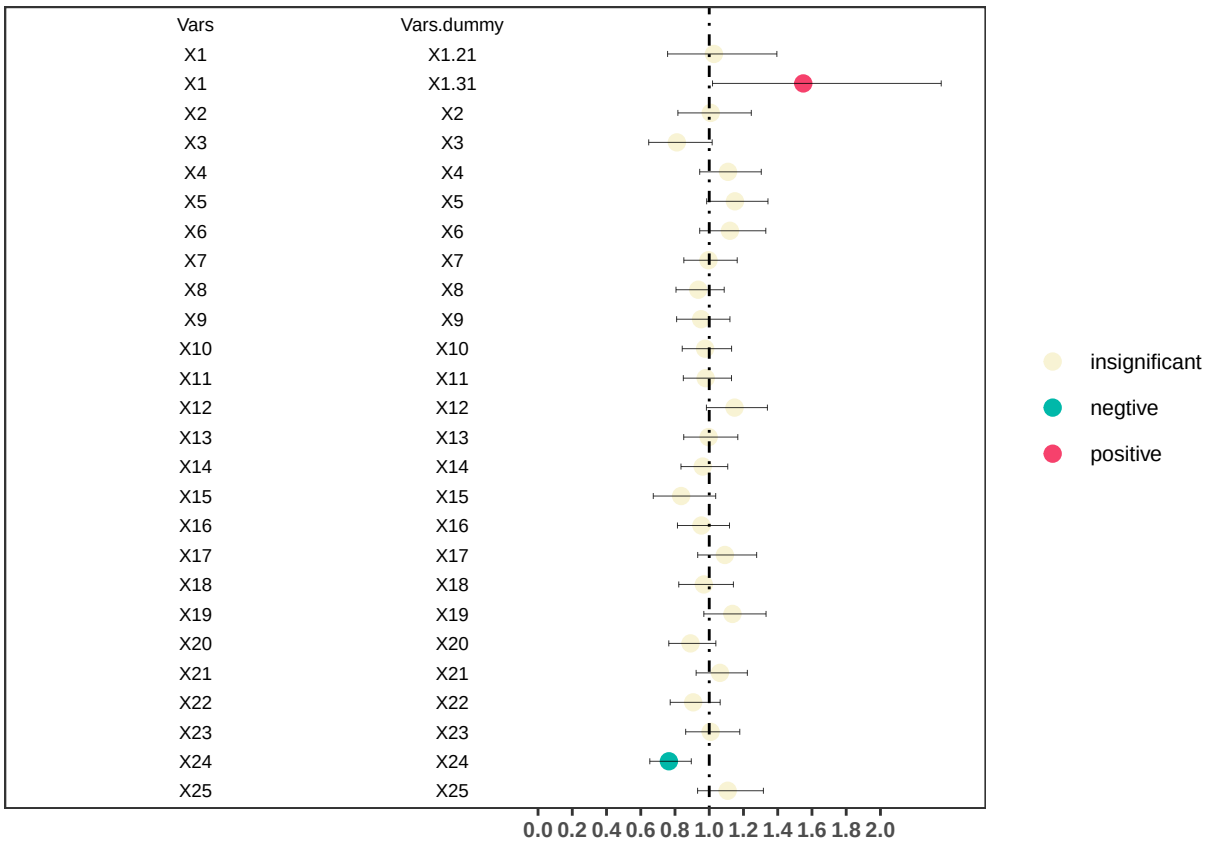
```
## # A tibble: 26 x 8
##    Vars  Vars.dummy    HR  CI_L  CI_H p.value cindex logloss
##    <chr> <chr>      <dbl> <dbl> <dbl>   <dbl>  <dbl>   <dbl>
##  1 X1    X1.21      1.03  0.757  1.40  0.862   0.648    4.70
##  2 X1    X1.31      1.55  1.02   2.35  0.0405  0.652    4.70
##  3 X2    X2         1.01  0.817  1.25  0.936   0.646    4.70
##  4 X3    X3         0.811 0.646  1.02  0.0698  0.648    4.69
##  5 X4    X4         1.11  0.944  1.30  0.207   0.656    4.70
##  6 X5    X5         1.15  0.985  1.34  0.0772  0.649    4.70
##  7 X6    X6         1.12  0.944  1.33  0.192   0.653    4.69
##  8 X7    X7         0.995 0.851  1.16  0.953   0.647    4.70
##  9 X8    X8         0.936 0.806  1.09  0.387   0.643    4.70
```

```
## 10 X9    X9          0.953 0.809  1.12  0.558   0.647    4.70
## # i 16 more rows
```

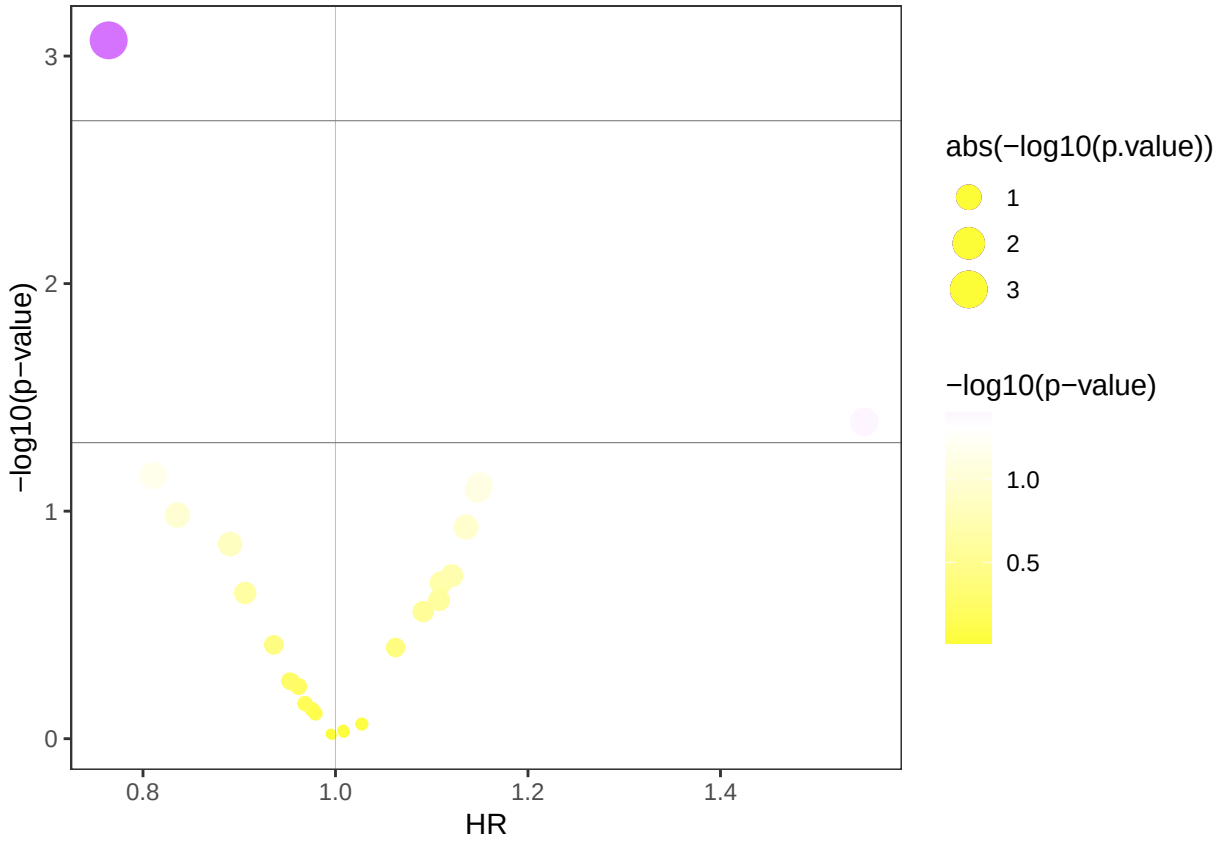
Viz Association

After running association model, users can visualize the results by running "VizSurvAsso()" function. The function includes eight parameters:
1. PID: chr. Program ID. It must be the same with the PID generated by InitSurv.
2. OutPath: chr. Output file directory, e.g. "D:/test". If "default", the current working directory will be set.
3. VarsN: chr. Choose the single factor or multiple factor model. Available options include "single.factor" and "multiple.factor". It must be same as the 'VarsN' in Association function. 4. Layout: chr. Visualization layout. Available options include "forest" and "volcano".
5. Brightness: chr. Visualization brightness. Available options include "light" and "dark".
6. Palette: chr. Visualization palette. Available options include "default1", "default2" and several journal preference styles (i.e., "cell", "nature", "science", "lancet", "nejm", and "jama").
7. ColorFor: chr. Volcano plot dot color. Available options include "p.value" and "hr".
8. SizeFor: chr. Volcano plot dot size. Available options include "p.value" and "hr".

```
res5 = VizSurvAsso(PID = res$PID,
                   VarsN="single.factor",
                   Layout="forest",
                   Brightness= "light",
                   Palette = "default1")
res5$ForestPlot
```



```
res6 = VizSurvAsso(PID = res$PID,
                   VarsN="single.factor",
                   Layout="volcano",
                   Brightness= "light",
                   Palette = "default1",
                   ColorFor= "p.value",
                   SizeFor= "p.value")
res6$VolcanoPlot
```



Prediction

In addition, users can predict the survival curve by "SurvPred()" function. The function includes ten parameters:
1. PID: chr. Program ID. It must be the same with the PID generated by InitSurv.
2. OutPath: chr. Output file directory, e.g. "D:/test". If "default", the current working directory will be set.
3. TimeY: chr. Outcome variable of survival time used for modelling. Only one variable can be entered.
4. EventY: chr. Outcome variable of status used for modelling. Only one variable can be entered. 5. VarsX: chr. Exposure variable used for modeling. The default option is "all.x" (All exposure variables are included). Users can also choose available variables. It should be noted that there is fixed format for the entering characters separated with comma and without space, e.g., "X1,X2,X3" 6. Covariates chr. Covariates used for modeling. The default option is "all.c" (All covariates are included).
7. RsmpMethod: chr. Three resampling methods options for internal validation, including "cv" (i.e.,Cross validation), "bootstrap", and "holdout".
8. Folds: num. Folds of Cross-validation resampling. It is ranging 2-10.
9. Ratio: num. Ratio of Bootstrap resampling. It is ranging 0.4-0.9.
10. Repeats: num. Number of Bootstrap resampling. It is ranging 2-20.

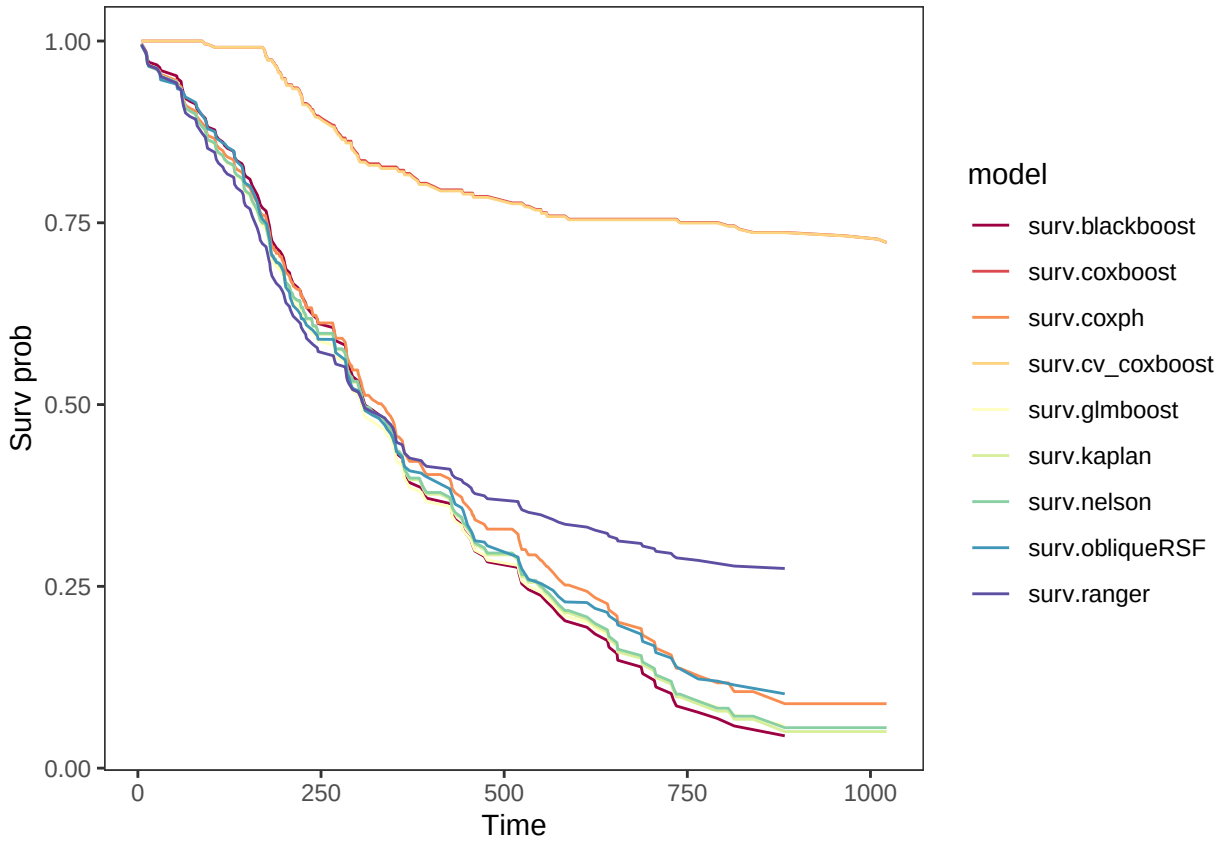
```
res7 = SurvPred(PID = res$PID,
                TimeY = "Y1",
                EventY = 'Y2',
                VarsX ='all.x',
                Covariates = "all.c",
                RsmpMethod ="cv",
                Folds = 3)
res7$bmrout
```

```
## # A tibble: 13 x 7
##    learner_id  cindex_rsmp_train_mean cindex_rsmp_test_mean cindex_rsmp_train_sd
##    <chr>                        <dbl>                <dbl>                 <dbl>
##  1 surv.black~                  0.815                0.600               0.0306
##  2 surv.cfore~                  0.792                0.599               0.0116
##  3 surv.coxbo~                  0.328                0.387               0.0180
##  4 surv.coxph                   0.713                0.589               0.0151
##  5 surv.ctree                   0.566                0.537               0.0575
##  6 surv.cv_co~                  0.404                0.403               0.0369
##  7 surv.gbm                     0.784                0.543               0.0218
##  8 surv.glmbo~                  0.694                0.624               0.0182
##  9 surv.kaplan                  0.5                  0.5                 0
## 10 surv.nelson                  0.5                  0.5                 0
## 11 surv.obliq~                  0.732                0.589               0.0289
## 12 surv.ranger                  0.904                0.572               0.00421
## 13 surv.xgboo~                  0.824                0.533               0.0334
## # i 3 more variables: cindex_rsmp_test_sd <dbl>, cindex_train <dbl>,
## #   cindex_test <dbl>
```

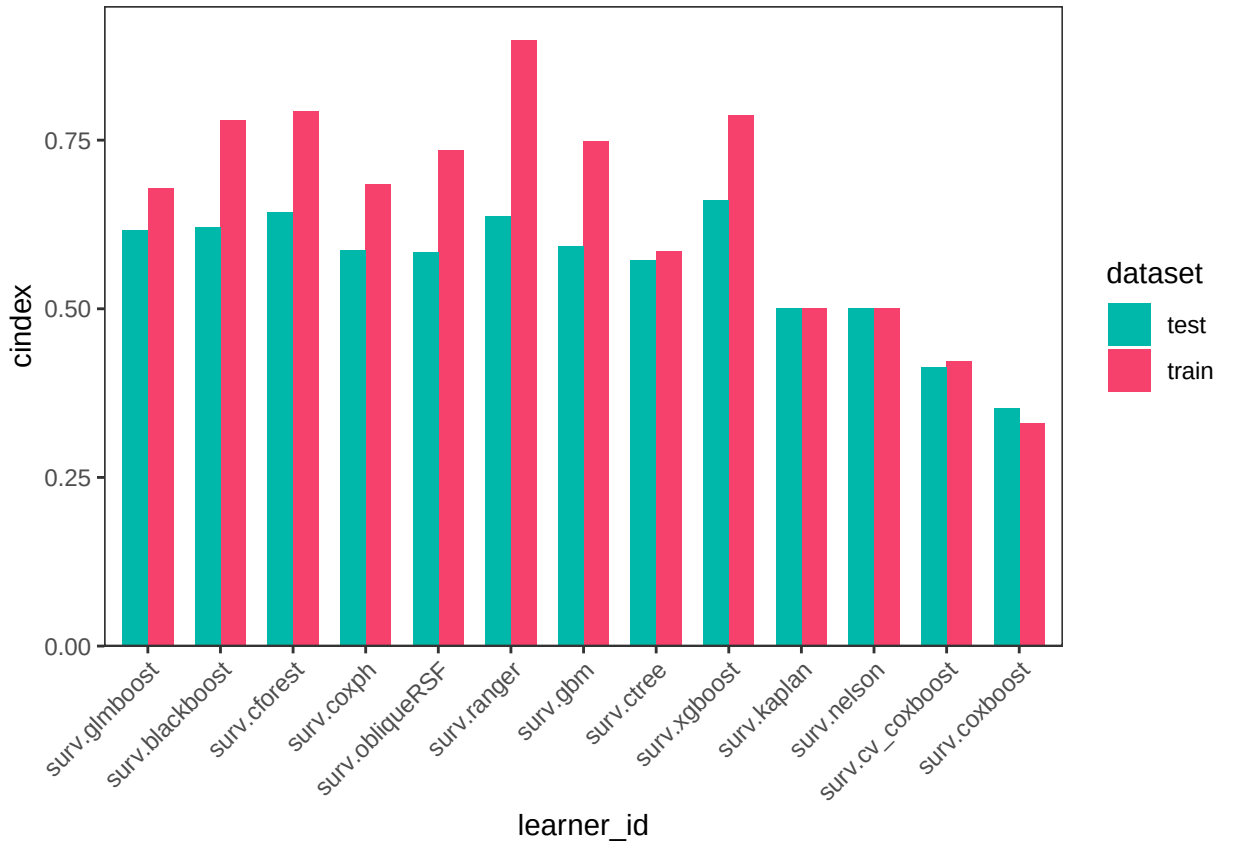
Viz Prediction

After running SurvPred model, users can visualize the results by running "VizSurvPred()". The function includes five parameters:
1. PID: chr. Program ID. It must be the same with the PID generated by InitSurv.
2. OutPath: chr. Output file directory, e.g. "D:/test". If "default", the current working directory will be set.
3. Layout: chr. chr. Visualization layout. Available options include "curve" , "bar" and 'all'. 4. Brightness: chr. Visualization brightness. Available options include "light" and "dark".
5. Palette: chr. Visualization palette. Available options include "default1", "default2" and several journal preference styles (i.e., "cell", "nature", "science", "lancet", "nejm", and "jama").

```
res8 = VizSurvPred(PID = res$PID,
                   Layout="all",
                   Brightness="light",
                   Palette='default1')
res8$PredSurvCurvPlot
```




```
res8$BmrBarPlot
```

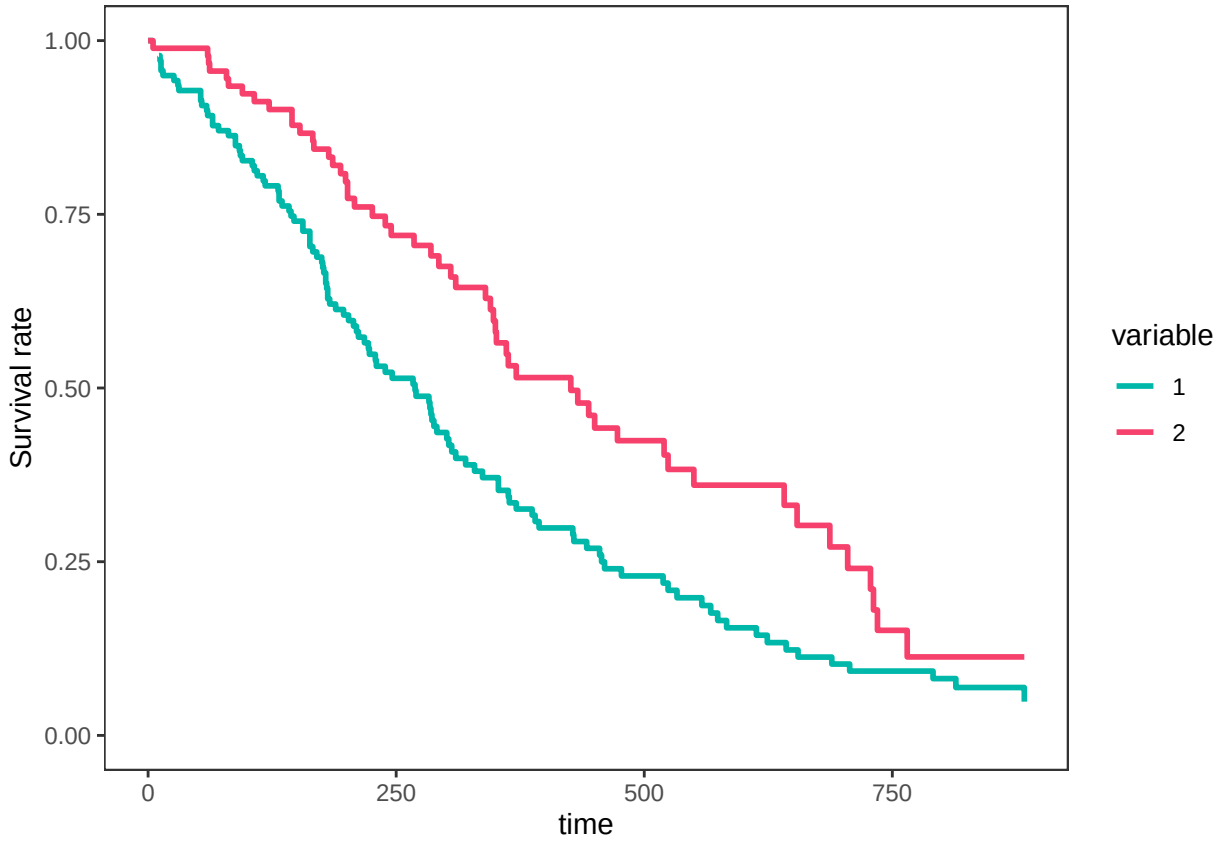


VizSurvCompGroup

We also provide a function to compare the survival curves of two groups. This function does not depend on the previous functions result. Users can run it after "LoadSurv()". The function includes ten parameters:
1. PID: chr. Program ID. It must be the same with the PID generated by InitSurv.
2. OutPath: chr. Output file directory, e.g. "D:/test". If "default", the current working directory will be set.
3. TimeY: chr. Outcome variable of survival time used for modelling. Only one variable can be entered.
4. EventY: chr. Outcome variable of status used for modelling. Only one variable can be entered. 5. VarsG: chr. Grouping variable, must be binary variables. It should be noted that there is fixed format for the entering characters separated with comma and without space, e.g., "X1,X2,X3" 6. Model: chr. Methods to depict the survival curve. Options include 'km' (Kaplan-Meier estimate) and "coxph" (Cox proportional hazards regression mode).
7. VarsAdj: chr. If you choose the cox model, co-variables used for modelling. It should be noted that there is fixed format for the entering characters separated with comma and without space, e.g., "X1,X2,X3".
8. AdjMethod: chr. If you choose the coxph model, method for adjusting model, includes: "average","single","margin" and "conditional".
9. Brightness: chr. Visualization brightness. Available options include "light" and "dark".
10. Palette: chr. Visualization palette. Available options include "default1", "default2" and several journal preference styles (i.e., "cell", "nature", "science", "lancet", "nejm", and "jama").

```
res9 = VizSurvCompGroup(PID = res$PID,
                        TimeY="Y1",
                        EventY="Y2",
                        VarsG="C3",
                        Model="coxph",
```

```
                          VarsAdj='X1,X2,X3,X4,X5',
                          AdjMethod='average',
                          Brightness="light",
                          Palette='default1')
res9$Comp_Coxph_Plot[[1]]
```



Exit

After all the analysis is done, please run the "FuncExit()" function to delete the data uploaded to the server.

```
FuncExit(PID = res$PID)
```

```
## [1] "Success to exit. Thanks for using ExposomeX platform!"
```

6.7 StatMedt fuction

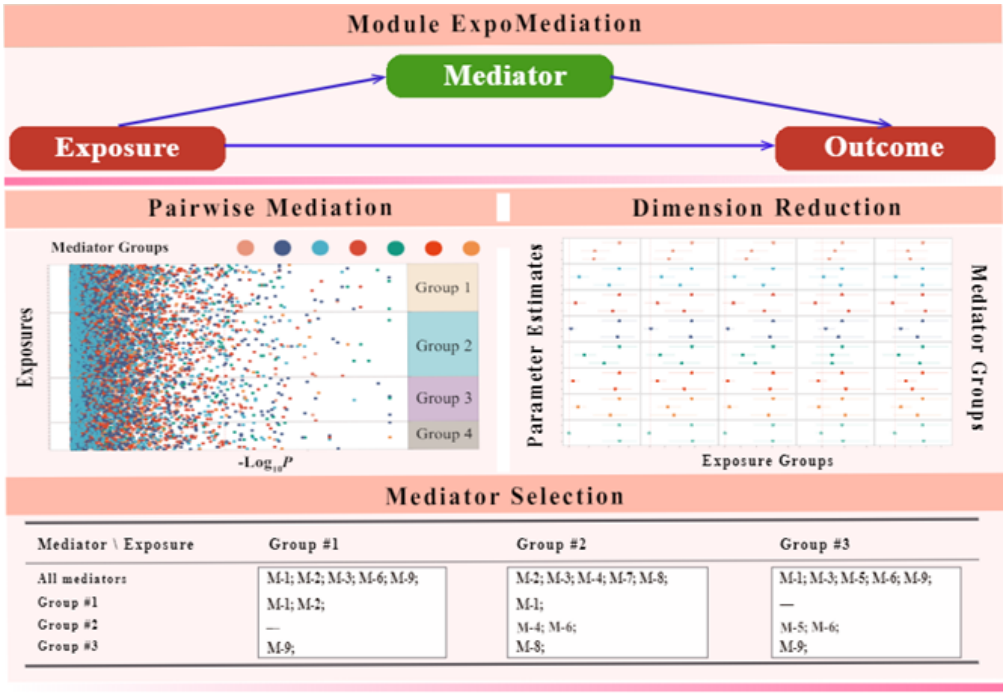
6.7.1 Application domain

The **StatMedt** is designed for estimating and screening potential mediation pathways of indigenous biomarkers (i.e., mediator) between external environmental exposure and health outcome in a user friendly and efficient way. Especially, when the number and category of exposures and mediators are of high dimension. Please see the website (http://www.exposomex.cn/#/expomediation) for more information.

6.7.2 Theory

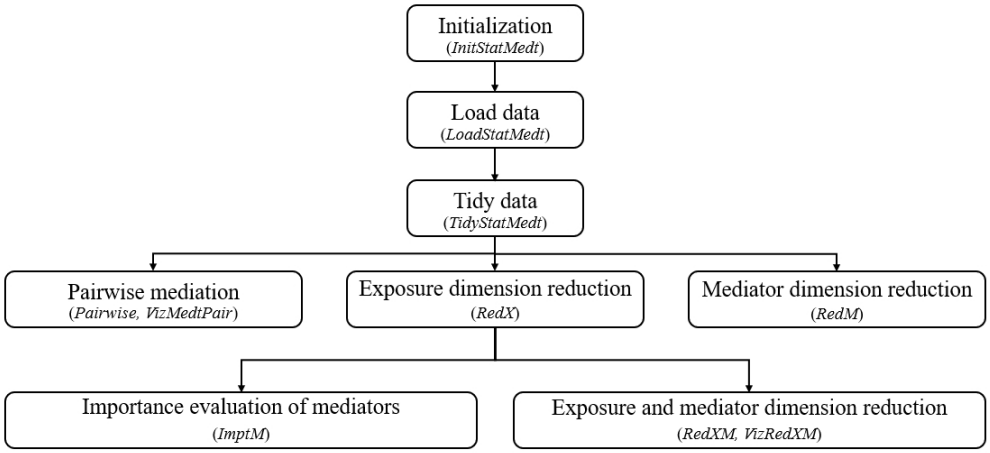
The **StatMedt** is realized by the *ESTAT* module in "*exposomex*" package. The core theory of the *StatMedt* contains two parts: pairwise mediation modelling and exposure/mediator dimension reduction. Pairwise mediation modelling allows the users to estimate the mediation effects of all potential combination of exposure-mediator pairs. For given m exposures and n mediators, an exhaustive rule will be executed and $m * n$ pair-wised modelling are fitted. In most cases or scenarios, the users may deal with datasets with numerous exposures and mediators. Using *StatMedt* might be a good choice to efficiently establish batch mediation modelling.

Dimension reduction for exposure/mediator is also an important module in *StatMedt* module. By providing subgroup information for each exposure/mediator, users can conduct dimension reduction for user-specified variables or for all variables automatically (default). In *StatMedt*, various alternative methods are provided for dimension reduction. Users can choose one that may be suitable for their data or try all possible methods to obtain results for sensitive analyses.



Aung et al., Nat Commun. 2020 Nov 6;11(1):5624. doi: 10.1038/s41467-020-19335-2. PMID: 33159049.

6.7.3 Work pipeline



6.7.4 Use example

Initialize package

Before we start using *StatMedt*, Make sure that the required packages is already installed.

```
# The following packages should be installed in advance
# devtools::install_github("https://github.com/oliverychen/PDM.git")
# devtools::install_github("https://github.com/StoreyLab/qvalue.git")
# install.packages("HIMA")
library(tidyverse)
# devtools::install_github("ExposomeX/exposomex", force = TRUE)
library(exposomex)
```

At first, you need to initialize the calculation environment using a series of initialization functions, e.g., InitStatCros, InitStatMO, InitStatTidy, InitEViz, InitEBIO, etc. Here, we use the function *InitStatMedt* for mediation analysis for example. The detailed information about the functions and returned value will be introduced in the following chapters.

```
res <- InitStatMedt()
res
```

```
## <eSet>
##   Public:
##     AddCommand: function (x)
##     AddLog: function (x)
##     clone: function (deep = FALSE)
##     DataStr: NULL
##     EpiDesign: NULL
##     ExcecutionLog: Complete initializing the mediation module.
##     2024-06-23  ...
##     Expo: list
##     ExpoDel: list
##     FileDirOut: /home/ubuntu/ExposomeX/R_Exposome_1.0/output_162535.1589 ...
##     PID: 162535.158977MDWUUY
##     RCommandLog: eSet <- InitExpoData(PID = Any ID your like, FileDirOut  ...
```

Here, we can see that the returned value "res" is an R6 object. It contains an unique program ID of res$PID, which is random generated by the system. Users need to use it in the following step for further data process.

Upload data

In *StatMedt*, we use the function *LoadStatMedt* to upload data file as well as vocabulary file for the following mediation analyses. Here, we use "example#1" to show the upload step. In *LoadStatMedt*, users can also provide their own data with the *DataPath* and *VocaPath* options, where either data or vocabulary file directory path should be provided. It should be noted that the slash symbol in directory path is "/", not "\". And the format of the input data and vocabulary file is fixed. You can access detailed information by visiting the following website http://www.exposomex.cn.

```
# Load
res1 <- LoadStatMedt(res$PID,
                     UseExample = "example#1")

res1$Expo$Voca %>%
  dplyr::select(SerialNo:DataType) %>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

SerialNo	SerialNo_Raw	FullName	GroupName	SubgroupName	DataType
Y1	Y1	Outcome1	outcome	none	numeric
Y2	Y2	Outcome2	outcome	none	numeric
C1	C1	Race	demography	none	factor
C2	C2	Age	demography	none	numeric
C3	C3	BMI	demography	none	numeric
X1	X1	NAP	chemical	polycyclic_aromatic_hydrocarbons	numeric
X2	X2	ANY	chemical	polycyclic_aromatic_hydrocarbons	numeric
X3	X3	ACE	chemical	polycyclic_aromatic_hydrocarbons	numeric
X4	X4	FLU	chemical	polycyclic_aromatic_hydrocarbons	numeric
X5	X5	PHE	chemical	polycyclic_aromatic_hydrocarbons	numeric

```
res1$Expo$Data %>%
  dplyr::select(SampleID:C2) %>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

SampleID	SubjectID	Group	Y1	Y2	C1	C2
1	1	train	36.08893	1	1	30.41476
2	2	train	37.77839	0	1	35.94014
3	3	train	37.12210	0	2	47.89022
4	4	train	38.31941	0	1	35.02661
5	5	train	35.68648	1	2	32.83606
6	6	train	37.50690	0	1	32.34126
7	7	train	39.01246	0	3	36.94808
8	8	train	38.20825	0	1	38.20296
9	9	train	36.19353	1	1	31.94458
10	10	train	38.56214	0	3	25.56944

Here, we can see that the returned value "res1" is an R6 object. The uploaded data is assigned to *res1ExpoData*, while the vocabulary file assigned to *res1ExpoVoca*.

Tidy data

In *StatMedt*, we use the function *TidyStatMedt* to perform some basic data pre-treatment (tidy) steps before variables were utilized for model establishment. In *TidyStatMedt*, we default delete variables with low variance and provide several data transform options including distribution and scaling transform. In *TidyStatMedt*, the users can specify *LogTransM="log10"* to perform "log10" transform for all numeric exposure and mediator variables simutaneously. Using *RangeLow = 0* and *RangeUpp = 1* can scale all these numeric variables into same scale at certain range. (Be noted that the assigned lower range value should be lower than that of upper range value.)

```
# Tidy
res2 <- TidyStatMedt(PID=res$PID,
                     LogTransM = "log10",
                     RangeLow = 0,
                     RangeUpp = 1)
res2$Expo$Data %>%
  dplyr::select(SampleID, X1:X7) %>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

SampleID	X1	X2	X3	X4	X5	X6	X7
1	0.3968577	0.4911042	0.5180237	0.3640152	0.3253081	0.2700578	0.6425291
2	0.2287671	0.5045529	0.5351105	0.3905166	0.4435101	0.4826007	0.4704535
3	0.3175847	0.4204500	0.4711400	0.2819057	0.3333500	0.3303057	0.5686859
4	0.2568953	0.5269232	0.5570932	0.3324674	0.5005428	0.4377722	0.6354442
5	0.4527545	0.4790360	0.4989746	0.2952093	0.2482096	0.5081859	0.4502595
6	0.6628369	0.7520861	0.7363791	0.6630776	0.6166057	1.0000000	0.5081613
7	0.2696743	0.2462582	0.2790659	0.0485411	0.3755114	0.5365363	0.4596760
8	0.1234730	0.3285827	0.3593810	0.2055361	0.3681568	0.4075867	0.6190422
9	0.3212555	0.3151913	0.4697546	0.4951780	0.4667961	0.3120015	0.5605782
10	0.4205742	0.2815852	0.5220362	0.5035260	0.3179838	0.2186530	0.3384098

Mediation modelling

Pairwise mediation modelling

First, we proposed a pairwise mediation modelling via the function *Pairwise*, where each given exposure (named as the form of "X*,X*"; i.e., "X1,X2") and mediator (named as the form of "M*,M*"; i.e., "M1,M2") will be selected to build mediation models.

In *Pairwise*, several parameters need to be specified. The details for each parameter are listed below:

PID:

A character indicating Program ID. It must be the same with the PID generated by *InitStatMedt*.

VarsY:

A character indicating the outcome variable. The outcome variables should be in the form of "Y*"; i.e., "Y1".Either continuous, binary or count variable is available in function *Pairwise*.

VarsX:

A character indicating the exposure variables. The exposure variables should be in the form of "X*,X*"; i.e., "X1,X2". Users can also specify "default" to include all exposure variables into the function.

VarsM:

A character indicating the mediator variables. The mediator variables should be in the form of "M*,M*"; i.e., "M1,M2". Users can also specify "default" to include all mediator variables into the function.

VarsC:

A character indicating the confounder variables. The confounder variables should be in the form of "C*,C*"; i.e., "C1,C2". Users can also specify "default" to include all confounder variables into the function.

Family:

A character indicating the family of the mediation modellings. Available options include "linear" and "gaussian" for continuous outcome variables, and "binomial" as well as "poisson" options for binary and count outcome variables.

Iter:

A numeric value indicating the number of iteration times. Default is 500. To get a more stable result, a minimum iteration of 1,000 is recommended. To obtain more digits of P value of parameter estimation, a minimum iteration of 5,000 is suggested. Noted that if the number of "Iter", or the number of exposure-mediator pair, is large, the total execution time of the function might be time consumed. Users should be patient for waiting the outputs. The following code generate a set of 40 mediation models (10 exposures * 4 mediators) with 500 iterations for each model. The outcome variable "Y1" is a continuous variable and "linear" family was used.

```
# Pairwise mediation modelling
res3 <- Pairwise(PID=res$PID,
                 VarsY = "Y1",
                 VarsX = "X1,X2,X3",
                 VarsM = "M1,M2,M3",
                 VarsC = "C1,C2,C3",
                 Family = "linear",
                 Iter = 50)
res3$MedtPairWise_Stats %>%
  dplyr::select(Pairs:TE_Pvalue) %>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

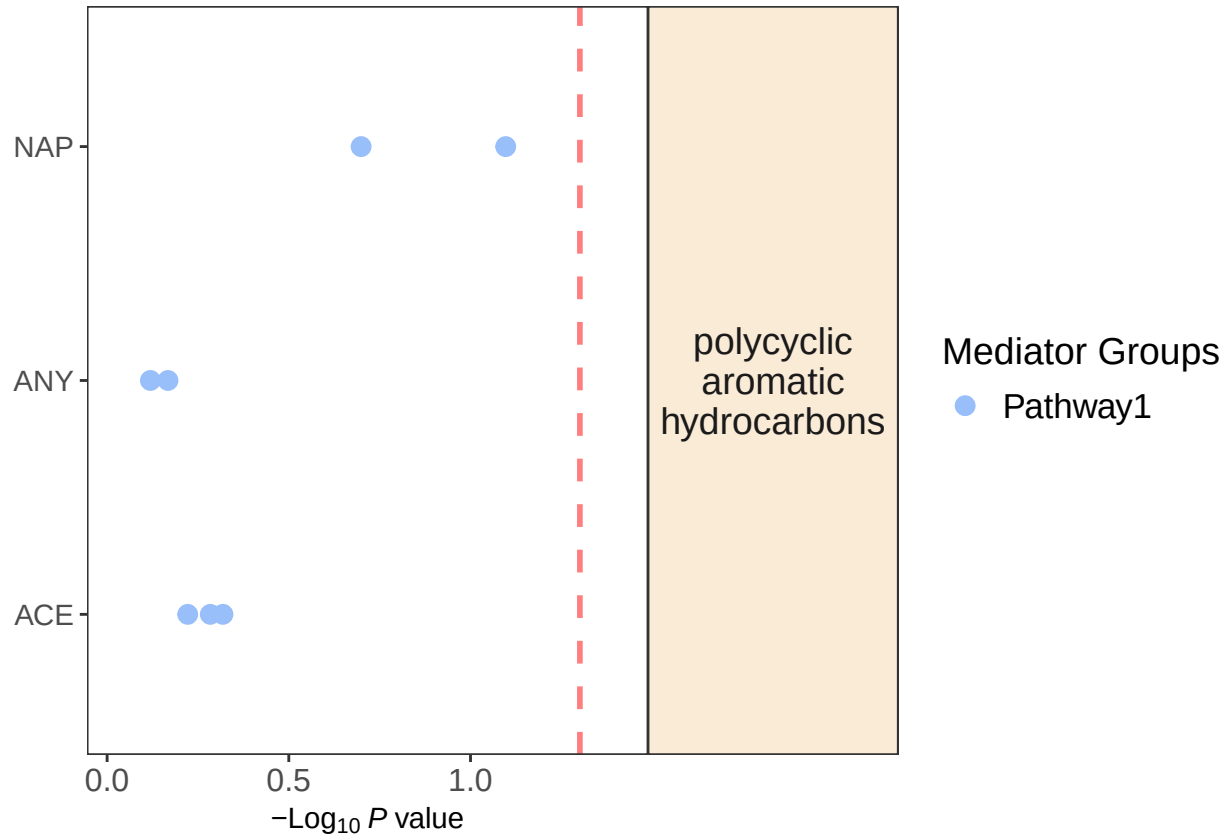
Pairs	TE	TE_LCL	TE_UCL	TE_Pvalue
X1 * M1	-1.834369	-3.254882	-0.2273369	0.04
X1 * M2	-1.472917	-2.869992	-0.0767592	0.04
X1 * M3	-1.807439	-3.083784	-0.0930463	0.08
X2 * M1	-2.906294	-4.411511	-1.4024839	0.00
X2 * M2	-2.929078	-4.405971	-1.5147536	0.00
X2 * M3	-2.878398	-4.831568	-1.4519233	0.00
X3 * M1	-3.337858	-4.987815	-1.4008324	0.00
X3 * M2	-3.126133	-4.342688	-1.5902121	0.00
X3 * M3	-3.096050	-4.769183	-1.2066377	0.00

The *Pairwise* function returns to a R6 object where the pairwise mediation results were stored in *res3$MedtPairWise_Stats*. In this example, we also used function *VizMedtPair* to visualize our pairwise mediation results. Noted that before using function *VizMedtPair*, make sure that the function *Pairwise* have been successfully executed. For details about this function, run "*??exposomex::VizMedtPair*" to see the help mannuals.

```
# Visualize pairwise mediation results
res4 <- VizMedtPair(PID = res$PID,
                    Brightness = "light",
                    Pallette = "default1")

res4$plotdata%>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

Mediator	SubgrpM	NameM	Exposure	SubgrpX	NameX	IE_Pvalue
M1	Pathway1	medi_1	X1	polycyclic aromatic hydrocarbons	NAP	0.08
M1	Pathway1	medi_1	X2	polycyclic aromatic hydrocarbons	ANY	0.88
M1	Pathway1	medi_1	X3	polycyclic aromatic hydrocarbons	ACE	0.48
M2	Pathway1	medi_2	X1	polycyclic aromatic hydrocarbons	NAP	0.04
M2	Pathway1	medi_2	X2	polycyclic aromatic hydrocarbons	ANY	0.68
M2	Pathway1	medi_2	X3	polycyclic aromatic hydrocarbons	ACE	0.52
M3	Pathway1	medi_3	X1	polycyclic aromatic hydrocarbons	NAP	0.20
M3	Pathway1	medi_3	X2	polycyclic aromatic hydrocarbons	ANY	0.76
M3	Pathway1	medi_3	X3	polycyclic aromatic hydrocarbons	ACE	0.60

```
grid::grid.draw(res4$plot[[1]])
```



Exposure dimension reduction

RedX function provides two alternative methods for exposure dimension reduction, including sum method (*mean*) and adaptive elastic net method in *"gcdnet"* package. By default, all exposures will be included for dimension reduction. Meanwhile, *StatMedt* also supports customized pre-defined exposure combinations for dimension reduction step (see detailed information in Vocabulary file format). Afterwards, mediation models will be built between given mediators and shrinkaged exposure variables.

```
# Exposure dimension reduction
res5 <- RedX(PID=res$PID,
             VarsY = "Y1",
             VarsC = "C1",
             #mediators that used for building mediation analyses
             VarsM = "M1,M2,M3",
             Method = "mean",
             Folds = 10,
             Family = "linear",
             Iter = 50)
```

Here, the option *Folds* indicates the folds that used for exposure dimension reduction procedure. For detailed information, see *"gcdnet"* package. *RedX* function returns shrinkaged exposure features, which are stored in $res4MedtRedX_ERSall * and * res4MedtRedX\_ERSlist$, respectively. *RedX* also perform mediation modelling as *Pairwise*, which is stored in *res4$MedtRedX_Stats*.

```
res5$MedtRedX_ERSall %>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
```

```
                align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

ERS_all
0.4792169
0.4947513
0.4386141
0.5022273
0.4547327
0.6714275
0.4011727
0.4583747
0.4805866
0.4473850

```
res5$MedtRedX_ERSlist %>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

polycyclic_aromatic_hydrocarbons	trace_metals
0.4190618	0.5393720
0.4204914	0.5690112
0.3648861	0.5123421
0.4347844	0.5696702
0.3948368	0.5146287
0.6861971	0.6566580
0.2438102	0.5585352
0.2770259	0.6397236
0.4136351	0.5475382
0.4091411	0.4856290

```
res5$MedtRedX_Stats %>%
  dplyr::select(Pairs:TE_Pvalue) %>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

Pairs	TE	TE_LCL	TE_UCL	TE_Pvalue
ERS_all * M1	-7.427839	-10.148338	-4.513628	0
ERS_all * M2	-7.787475	-11.001836	-4.896200	0
ERS_all * M3	-7.507637	-10.747444	-4.623888	0
polycyclic_aromatic_hydrocarbons * M1	-4.070897	-6.334096	-2.307712	0
polycyclic_aromatic_hydrocarbons * M2	-4.059977	-5.943537	-2.051544	0
polycyclic_aromatic_hydrocarbons * M3	-4.130134	-5.848797	-2.457049	0
trace_metals * M1	-5.476709	-9.213678	-2.856374	0
trace_metals * M2	-5.438345	-7.799058	-2.923984	0
trace_metals * M3	-5.185876	-8.659925	-2.622320	0

Mediator dimension reduction

Similarly, *RedM* function provides two alternative methods for mediator dimension reduction, including *mean* method as well as *pdm1* method in *"PDM"* package. By default, all mediators will be included for dimension reduction. *StatMedt* also supports customized pre-defined mediator combinations for dimension reduction step (see detailed information in Vocabulary file format). Afterwards, mediation models will be built between given exposures as well as shrinkaged mediator variables.

```
# Mediator dimension reduction
res6 <- RedM(PID=res$PID,
             VarsY = "Y1",
             VarsX = "X1,X2,X3",
             VarsC = "C1,C2,C3",
             Method = "pdm1",
             Family = "linear",
             Iter = 50)
```

RedM function returns shrinkaged mediator features and perform mediation modelling as *Pairwise*, which is stored in $res6MedtRedM_all * and * res6MedtRedM\_list$.

```
res6$MedtRedM_all %>%
  dplyr::select(Expo:TE_Pvalue) %>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

Expo	Mediator	TE	TE_LCL	TE_UCL	TE_Pvalue
X1	All mediators	-1.71897813198377	-3.05079820055541	0.0820387357317153	0.08
X2	All mediators	-2.78001707324766	-4.33890921650724	-0.673580395885594	0
X3	All mediators	-3.23757205208212	-4.926702443449	-0.964755316674611	0

```
res6$MedtRedM_list %>%
  dplyr::select(Expo:TE_Pvalue) %>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

Expo	Mediator	TE	TE_LCL	TE_UCL	TE_Pvalue
X1	Pathway1	-1.71461098321024	-3.1722874208893	0.0256744570016948	0.08
X2	Pathway1	-2.78138002215819	-4.4736670847764	-0.726199414054782	0
X3	Pathway1	-3.23846607691902	-5.06504068763376	-1.05554490906197	0
X1	Pathway2	-1.71592983267204	-3.13565963440105	0.0573773629587472	0.08
X2	Pathway2	-2.77929651021231	-4.42722150124256	-0.732631148648331	0
X3	Pathway2	-3.23844570788077	-5.02352870399842	-1.02112770892764	0

Exposure and mediator dimension reduction

RedXM function provides two alternative methods for mediator dimension reduction, including *mean* method as well as *pdm1* method in *PDM* package. By default, all mediators will be included for dimension reduction. Afterwards, mediation models will be built between shrinkaged exposure variables as well as shrinkaged mediator variables. Prior to using *RedXM*, make sure that *RedX* function has been executed to obtain shrinkaged exposure variables.

```
# Exposure and mediator dimension reduction
res7 <- RedXM(PID=res$PID,
              VarsY = "Y1",
              VarsC = "C1,C2,C3",
              Method = "pdm1",
              Family = "linear",
              Iter = 50)
```

The *RedXM* function returns similar results as those in *RedM*. The users can check the results in $res7MedtRedXM_all * and * res7MedtRedXM\_list$.

```
res7$MedtRedXM_all %>%
  dplyr::select(Expo:TE_LCL) %>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

Expo	Mediator	TE	TE_LCL
ERS_all	All mediators	-7.29063280412862	-10.0064439802815
polycyclic_aromatic_hydrocarbons	All mediators	-3.8274806373766	-5.71452036395672
trace_metals	All mediators	-5.28144781123686	-7.96058735599803

```
res7$MedtRedXM_list %>%
  dplyr::select(Expo:TE_LCL) %>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

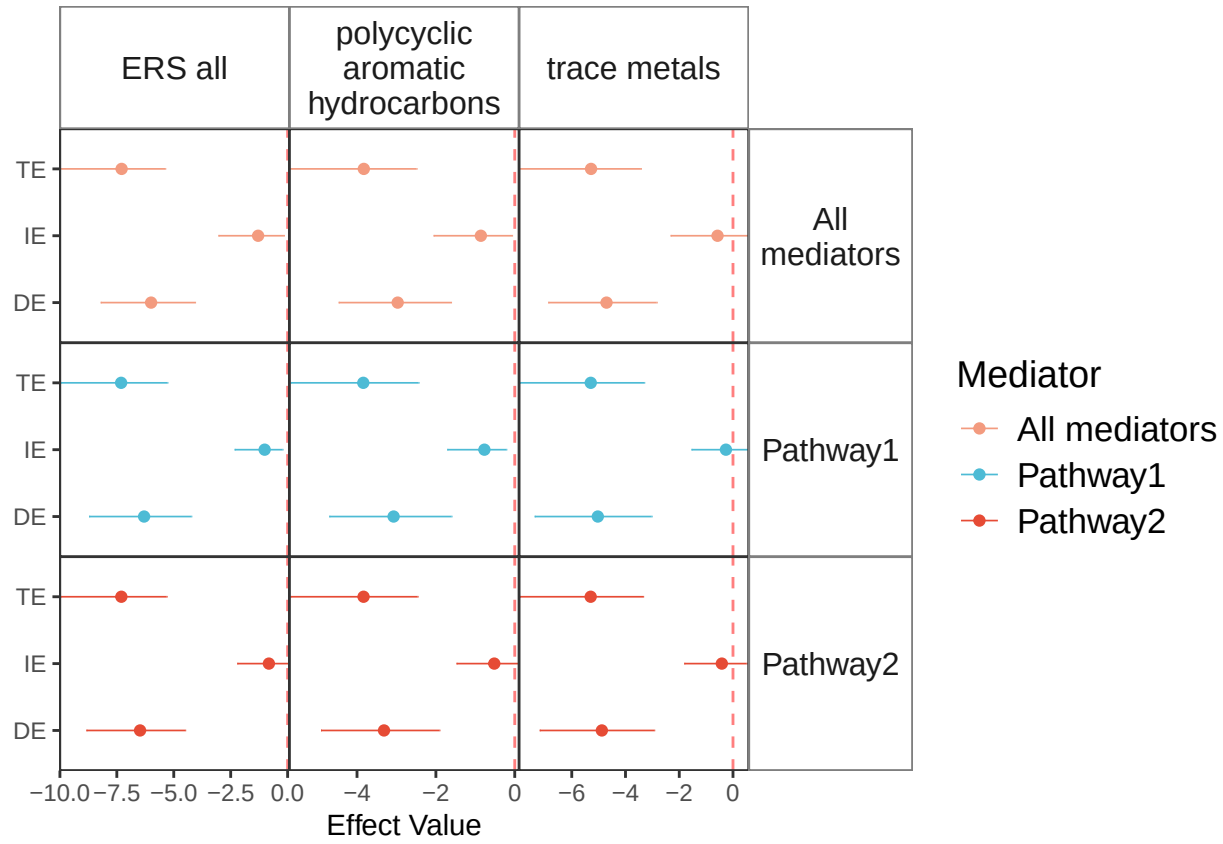
Expo	Mediator	TE	TE_LCL
ERS_all	Pathway1	-7.31172702350687	-9.96656810653492
polycyclic_aromatic_hydrocarbons	Pathway1	-3.84126230206179	-5.68400117426737
trace_metals	Pathway1	-5.29308264658955	-7.92762446713605
ERS_all	Pathway2	-7.30194735626879	-9.97302140074553
polycyclic_aromatic_hydrocarbons	Pathway2	-3.83447961719452	-5.6890803068431
trace_metals	Pathway2	-5.29528065809763	-7.93446227609634

In *StatMedt*, we also provide *VizRedXM* function to draw visualization results of *RedXM*. Noted that before using function *VizRedXM*, make sure that the function *RedXM* have been successfully executed. For details about this function, run "*??exposomex::VizRedXM*" to see the help mannuals.

```
# Exposure and mediator dimension reduction
res8 <- VizRedXM(PID = res$PID,
                    Brightness = "light",
                    Pallette = "nature")

res8$MedtRedXM_plotadta %>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

Expo	Mediator	Effect	Effect_Value	LowerCL	UpperCL
ERS all	Pathway1	IE	-1.0081502	-2.325756	-0.2228192
ERS all	Pathway1	DE	-6.3035769	-8.715190	-4.2280558
ERS all	Pathway1	TE	-7.3117270	-9.966568	-5.2788524
polycyclic aromatic hydrocarbons	Pathway1	IE	-0.7695594	-1.710396	-0.2135985
polycyclic aromatic hydrocarbons	Pathway1	DE	-3.0717029	-4.701916	-1.6037482
polycyclic aromatic hydrocarbons	Pathway1	TE	-3.8412623	-5.684001	-2.4391014
trace metals	Pathway1	IE	-0.2631131	-1.542141	0.5779685
trace metals	Pathway1	DE	-5.0299696	-7.378765	-3.0268923
trace metals	Pathway1	TE	-5.2930826	-7.927625	-3.3029484
ERS all	Pathway2	IE	-0.8225093	-2.208931	0.0789735

```
res8$MedtRedXM_plot
```



Estimation importance of mediators

ImptM function estimates the importance of mediators among given exposures. The estimation procedure is mainly realized by the *hima* function in *HIMA* package. *ImptM* also provides another evaluation method by the *bama* function in *bama* package. The users can search for these packages for further information.

For better comprehension and explanation of the result, we recommend that the parameter "VarsY" and "VarsC" unchanged. *ImptM* also supports shrinkaged exposure variables in VarsX, but *RedX* should be executed before including them into *ImptM*.

```
# Estimation the importance of mediators
res9  <- ImptM(PID = res$PID,
               VarsY = "Y1",
               VarsX = "default",
               VarsC = "default")

res9$MedtImptM_all %>%
  dplyr::select(exposure:pip,Bonferroni.p,BH.FDR) %>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

exposure	mediator	mediator_group	pip	Bonferroni.p	BH.FDR
X1	M1	All mediators	0.008	NA	NA
X1	M2	All mediators	0.004	0.2075003	0.1145398
X1	M3	All mediators	0.004	NA	NA
X1	M4	All mediators	0.002	NA	NA
X1	M5	All mediators	0.004	1.0000000	0.3648106
X1	M6	All mediators	0.002	0.2617725	0.1145398
X1	M7	All mediators	0.000	0.9844047	0.3281349
X1	M8	All mediators	0.000	NA	NA
X1	M9	All mediators	0.002	NA	NA
X1	M10	All mediators	0.004	NA	NA

```
res9$MedtImptM_list %>%
  dplyr::select(exposure:pip,Bonferroni.p,BH.FDR) %>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

exposure	mediator	mediator_group	pip	Bonferroni.p	BH.FDR
X1	M1	Pathway1	0.244	NA	NA
X1	M2	Pathway1	0.108	0.1037501	0.1037501
X1	M3	Pathway1	0.074	NA	NA
X1	M4	Pathway1	0.034	NA	NA
X1	M5	Pathway1	0.048	0.7296212	0.3648106
X2	M1	Pathway1	0.008	NA	NA
X2	M2	Pathway1	0.038	1.0000000	0.8208625
X2	M3	Pathway1	0.188	NA	NA
X2	M4	Pathway1	0.024	1.0000000	0.8208625
X2	M5	Pathway1	0.102	0.5268965	0.5268965

```
res9$MedtImptM_table2 %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

mediator_group	X1	X5	X4	X3	X6
All mediators	M2,M6	M2,M6,M7	NA	NA	NA
Pathway1	M2	M2	M2,M5		
Pathway2	M6	M6	M6	M6	M6

Exit exposome mediation analyses

When finishing your analyses task and saving all the results needed (**That is vital for the users.**), remember run the exit function provided in that package, which will release the memory of R environment and is of helpful for both the users and the ExposomeX platform.

```
FuncExit(PID = res$PID)
```

```
## [1] "Success to exit. Thanks for using ExposomeX platform!"
```

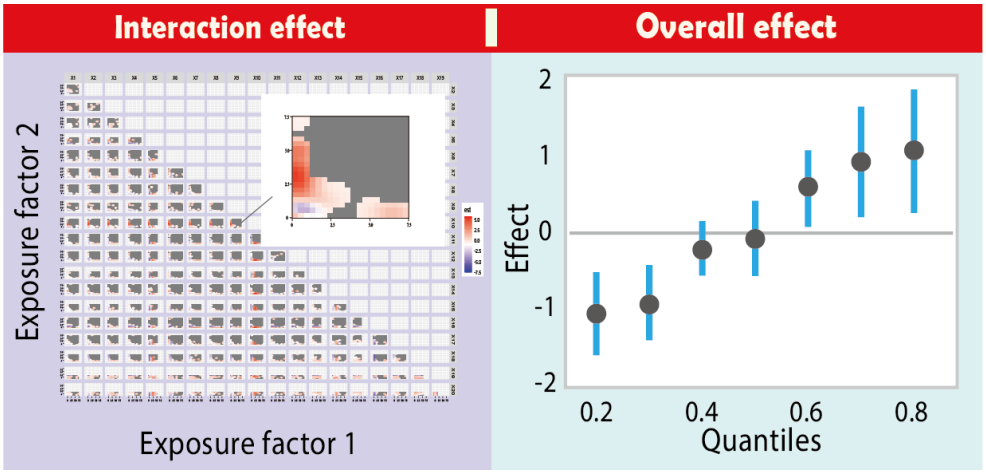
6.8 StatMix function**

6.8.1 Application domain

The **StatMix** function is designed to analyze mixture effect of the various exposure factors. It mainly aims to screen the representative features with high contribution to the health outcome, as well as their potential interaction effect. Users can easily get the modeling results and their visualization plots with high quality by following the detailed instructions in each step. Four frequently used mixture-effect models are provided including multiple linear regression with regulation algorithms (e.g., stepwise, LASSO and elastic network), weighted quantile sum regression (WQSR), Bayesian kernel machine regression (BKMR), and g-computation model. Please see the website (http://www.exposomex.cn/#/statmix) for more information.

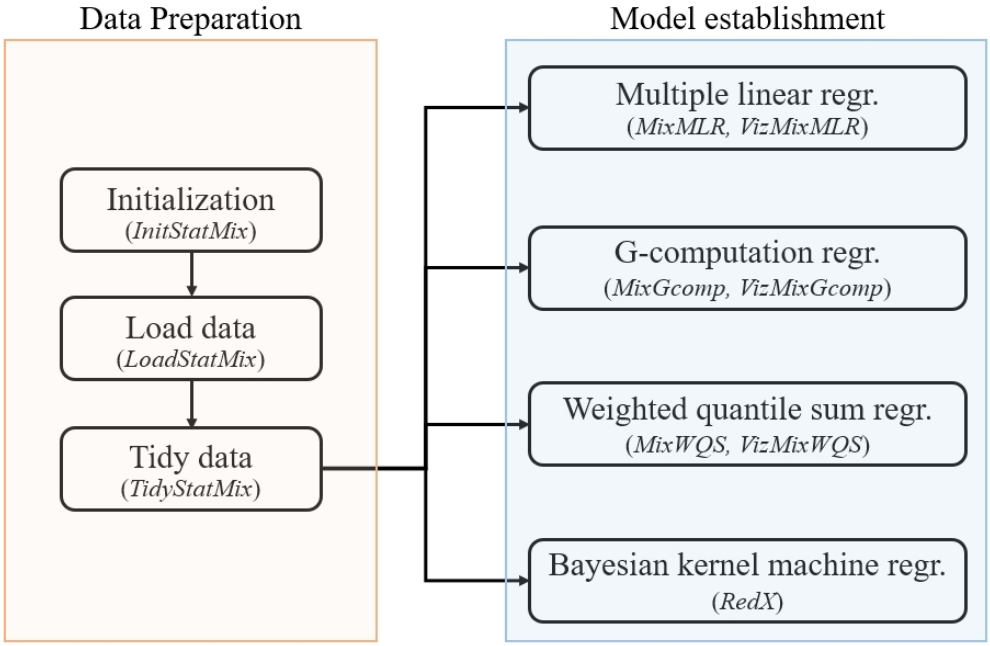
6.8.2 Theory

The **StatMix** is realized by the *ESTAT* module in "*exposomex*" package. The core theory of the **StatMix** contains several famous mixture statistical models that have broad application in environmental epidemiology studies. These models with established statistical algorithms were applied and integrated, providing comprehensive knowledge about the mixture exposures regarding to their overall and potential interaction effects on disease outcomes. In most cases or scenarios, the users may deal with dataset with numerous exposures. Using **StatMix** can be a good choice to efficiently establish batch modelling. Besides, **StatMix** also supports visualization steps for each mixture model, providing easy access for the users to check the modelling results and to compare results between parallel models intuitively.



References: Becker et al., mlr3 book (https://mlr3book.mlr-org.com/) Renzetti et al., 2021, How to use gWQS package (R package vignette of "gWQS"). Jennifer F. Bobb, 2017, Introduction to Bayesian kernel machine regression and the bkmr R package.

6.8.3 Work pipeline



6.8.4 Use example

Initialize package

Before we start using **StatMix**, Make sure that the required packages is already installed.

```
# The following packages should be installed in advance
# devtools::install_github("https://github.com/oliverychen/PDM.git")
# devtools::install_github("https://github.com/StoreyLab/qvalue.git")
# install.packages("HIMA")
library(tidyverse)
# devtools::install_github("ExposomeX/exposomex", force = TRUE)
library(exposomex)
```

First, you need to initialize the calculation environment using a series of initialization functions, e.g., InitStatCros, InitStatMO, InitStatTidy, InitEViz, InitEBIO, etc. Here, we use the function *InitStatMix* for mixture analysis for example. The detailed information about the functions and returned value will be introduced in the following chapters.

```
res <- InitStatMix()
res
```

```
## <eSet>
##   Public:
##     AddCommand: function (x)
##     AddLog: function (x)
##     clone: function (deep = FALSE)
##     EpiDesign: NULL
##     ExcecutionLog: Complete initializing the ExpoStat module.2024.06.23 16. ...
```

```
##     Expo: list
##     FileDirIn: NULL
##     FileDirOut: /home/ubuntu/ExposomeX/R_Exposome_1.0/output_164037.9405 ...
##     PID: 164037.940585YNXGUB
##     RCommandLog: eSet <- InitMix()
##     VarsDel: NULL
```

Here, we can see that the returned value "res" is an R6 object. It contains an unique program ID of res$PID, which is random generated by the system. Users need to use it in the following step for further data process.

Upload data

In **StatMix**, we use the function *LoadStatMix* to upload data file as well as vocabulary file for the following mediation analyses. Here, we use "example#1" to show the upload step. In *LoadStatMix*, users can also provide their own data with the *DataPath* and *VocaPath* options, where either data or vocabulary file directory path should be provided. It should be noted that the slash symbol in directory path is "/", not "\". And the format of the input data and vocabulary file is fixed. You can access detailed information by visiting the following website http://www.exposomex.cn.

```
# Load
res1 <- LoadStatMix(res$PID,
                    UseExample = "example#1")

res1$Expo$Voca %>%
  dplyr::select(SerialNo:DataType) %>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

SerialNo	SerialNo_Raw	FullName	GroupName	DataType
Y1	Y1	Outcome	none	cont
C1	C1	race	none	cate
C2	C2	age	none	cont
C3	C3	tlbmi	none	cont
C4	C4	sg	none	cont
C5	C5	insurance	none	cate
C6	C6	education	none	cate
X1	X1	MEHP	phthalates	cont
X2	X2	MEHHP	phthalates	cont
X3	X3	MEOHP	phthalates	cont

```
res1$Expo$Data %>%
  dplyr::select(SampleID:C2) %>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

SampleID	SubjectID	Group	Y1	C1	C2
1	1	train	13.627221	1	30.41476
2	2	train	4.686458	1	35.94014
3	3	train	8.375273	2	47.89022
4	4	train	5.993156	1	35.02661
5	5	train	18.427371	2	32.83606
6	6	train	60.814057	1	32.34126
7	7	train	5.975616	3	36.94808
8	8	train	2.651296	1	38.20296
9	9	train	7.784187	1	31.94458
10	10	train	13.934884	3	25.56944

Here, we can see that the returned value "res1" is an R6 object. The uploaded data is assigned to *res1ExpoData*, while the vocabulary file assigned to *res1ExpoVoca*.

Tidy data

In **StatMix**, we use the function *TidyStatMix* to perform some basic data pre-treatment (tidy) steps before variables were utilized for model establishment. In *TidyStatMix*, we default delete variables with low variance and provide several data transform options including distribution and scaling transform. In *TidyStatMix*, the users can specify *LogTransM="log10"* to perform "log10" transform for all numeric exposure and mediator variables simutaneously. Using *RangeLow = 0* and *RangeUpp = 1* can scale all these numeric variables into same scale at certain range. (Be noted that the assigned lower range value should be lower than that of upper range value.)

```
# Tidy
res2 <- TidyStatMix(PID=res$PID,
                    LogTransM = "log10",
                    RangeLow = 0,
                    RangeUpp = 1)
res2$Expo$Data %>%
  dplyr::select(SampleID, X1:X7) %>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

SampleID	X1	X2	X3	X4	X5	X6	X7
1	0.3968577	0.4911042	0.5180237	0.3640152	0.3253081	0.2613555	0.3350413
2	0.2287671	0.5045529	0.5351105	0.3905166	0.4435101	0.3762457	0.4712790
3	0.3175847	0.4204500	0.4711400	0.2819057	0.3333500	0.4374336	0.4634929
4	0.2568953	0.5269232	0.5570932	0.3324674	0.5005428	0.3203444	0.5171683
5	0.4527545	0.4790360	0.4989746	0.2952093	0.2482096	0.2643663	0.4115140
6	0.6628369	0.7520861	0.7363791	0.6630776	0.6166057	0.3473278	0.7587697
7	0.2696743	0.2462582	0.2790659	0.0485411	0.3755114	0.1983097	0.2039407
8	0.1234730	0.3285827	0.3593810	0.2055361	0.3681568	0.3076945	0.4751565
9	0.3212555	0.3151913	0.4697546	0.4951780	0.4667961	0.3013361	0.4884304
10	0.4205742	0.2815852	0.5220362	0.5035260	0.3179838	0.3217715	0.3852687

Mixture modelling

In **StatMix** module, four candidate mixture models were provided for use, i.e., multiple linear regression, weighted quantile sum regression (WQS), bayesian kernel machine regression (BKMR), and g-computation

model (G-comp). All of the abovementioned models yield constructing mixture exposure scenario with various advantages. The users can build either model with easy steps and obtain parallel results for comparison.

Multiple linear regression (MLR)

First, we proposed a multiple linear regression modelling with *MixMLR* function. In *MixMLR*, both binary and continous dependent variable can be utilized for model building with *Family* option. *MixMLR* also provides several customized parameters to be specified. The details for each parameter are listed below:

PID:
A character indicating Program ID. It must be the same with the PID generated by *InitStatMix*.
VarsY:
A character indicating the outcome variable. The outcome variables should be in the form of "Y*"; i.e., "Y1".Either continuous, binary or count variable is available in function *MixMLR*.
VarsX:
A character indicating the exposure variables. The exposure variables should be in the form of "X*,X*"; i.e., "X1,X2". Users can also specify "default" to include all exposure variables into the function.
Covariates:
A character indicating the confounder variables. The confounding variables should be in the form of "C*,C*"; i.e., "C1,C2". Users can also specify "default" to include all confounder variables into the function.
SelMethod:
Methods to select the important features to the final model. Options include "Stepwise" (stepwise regression), "LASSO" (Regularization regression of least absolute shrinkage and selection operator), and "ENET" (Regularization regression of elastic net).
PredType:
Prediction type of the outcome variable, including "Response" for the actual values and "Probability" for outcome with binary variable.
Family:
The link function for the regression model according the data type of outcomes, including "Gaussian" for continuous variable, "Binomial" for binary variable, "Multinomial" for categorically ordered variable, and "Poisson" for counting variable.

```
# Multiple linear regression
res3 = MixMLR(PID = res$PID,
              VarsY = "Y1", #continuous dependent variable
              VarsX = "all.x",
              Covariates = "all.c",
              SelMethod = "lasso",# lasso algorithm
              PredType = "response",
              Family = "gaussian")
res3$Y1_lasso_features.sel
```

```
## # A tibble: 1 x 1
##   value
##   <chr>
## 1 X1
```

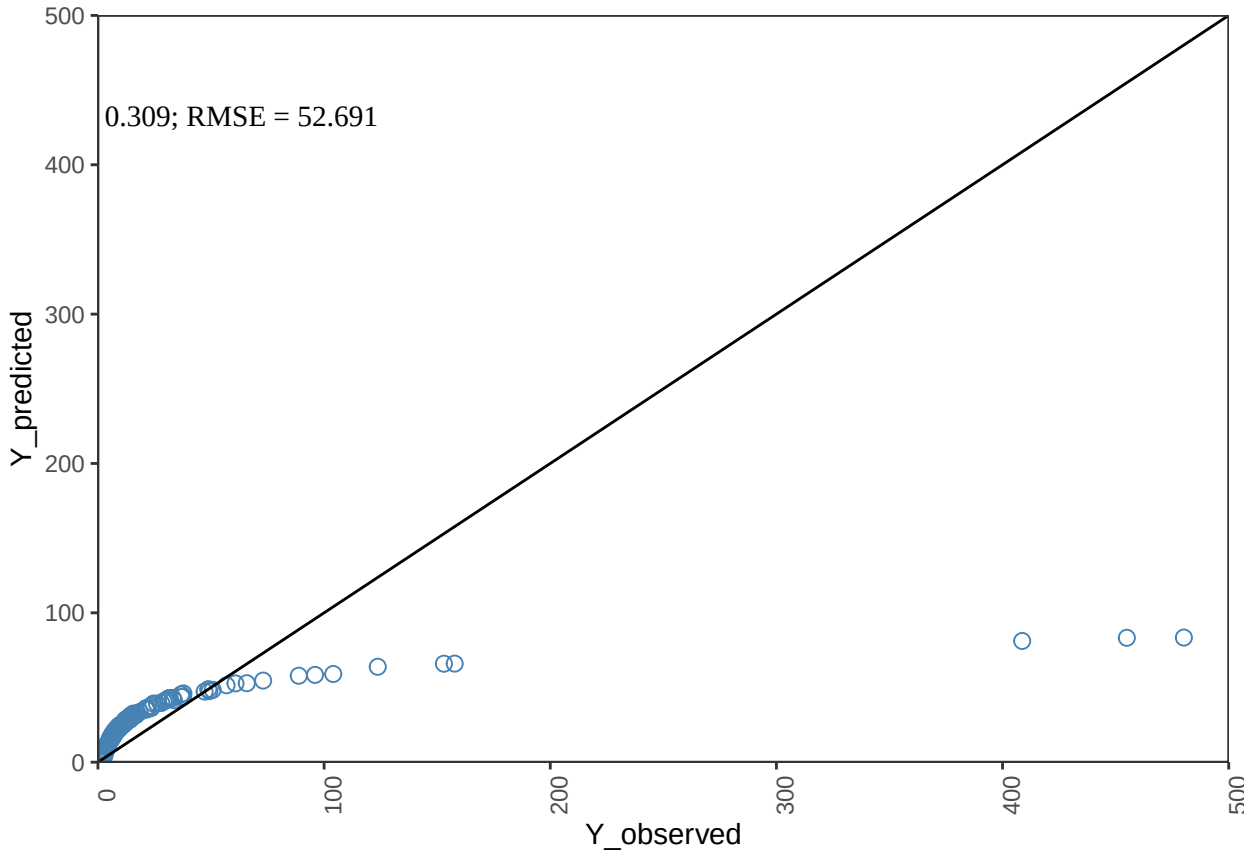
```
res3$Y1_lasso_vip
```

```
## # A tibble: 1 x 3
##   Variable Importance Sign
##   <chr>         <dbl> <chr>
## 1 X1             91.1 POS
```

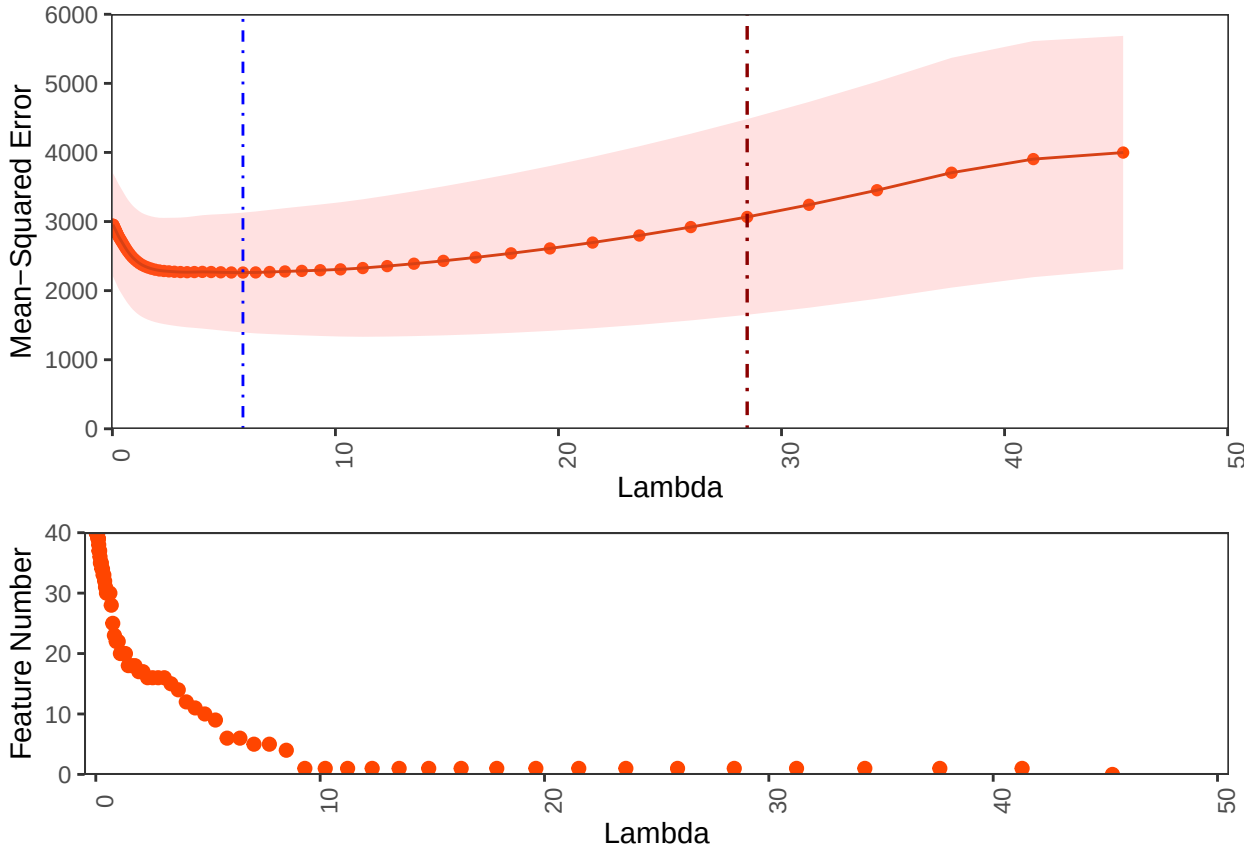
Actually, *MixMLR* function returns a R6 object where the modelling results were stored. In this example, we used lasso algorithm to construct mixture model. The feature selection results were stored in *res3$Y1_lasso_features.sel*, while the feature importance results were in *res3$Y1_lasso_vip*. Noted that in this example, only one feature (i.e., X1) were eventually selected in lasso model, indicating its significant positive effects on the outcome (i.e., Y1).

In **StatMix**, we can also use *VizMixMLR* function to draw plots to visualize model results. Here, we draw the model prediction evaluation and model optimization results. Besides, *VizMixMLR* provides feature importance plot, which is stored in *res4$Y1_lasso_importance_light_default1*.

```
# Draw plots with VizMixMLR
res4 = VizMixMLR(PID=res$PID,
                 VarsY = "Y1",
                 SelMethod = 'lasso',
                 Brightness = "light",
                 Palette = "default1")
res4$Y1_lasso_model.eval_light_default1
```




```
res4$Y1_lasso_opt.curve_light_default1
```



G-computation model (Gcomp)

G-computation models are realized with *"qgcomp"* package. Detailed information is available in https://cran.r-project.org/web/packages/qgcomp/vignettes/qgcomp-vignette.html. Quantile g-computation yields estimates of the effect of increasing all exposures by one quantile simultaneously. Thus, it estimates a "mixture effect" useful in the study of exposure mixtures such as air pollution and water contamination.

```
# Quantile g-computation
res5 = MixGcomp(PID=res$PID,
                VarsY = "Y1",
                VarsX = "all.x" ,
                Covariates = "all.c",
                Family = "gaussian" ,
                q = 10)
res5$Y1_gaussian_Coef %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

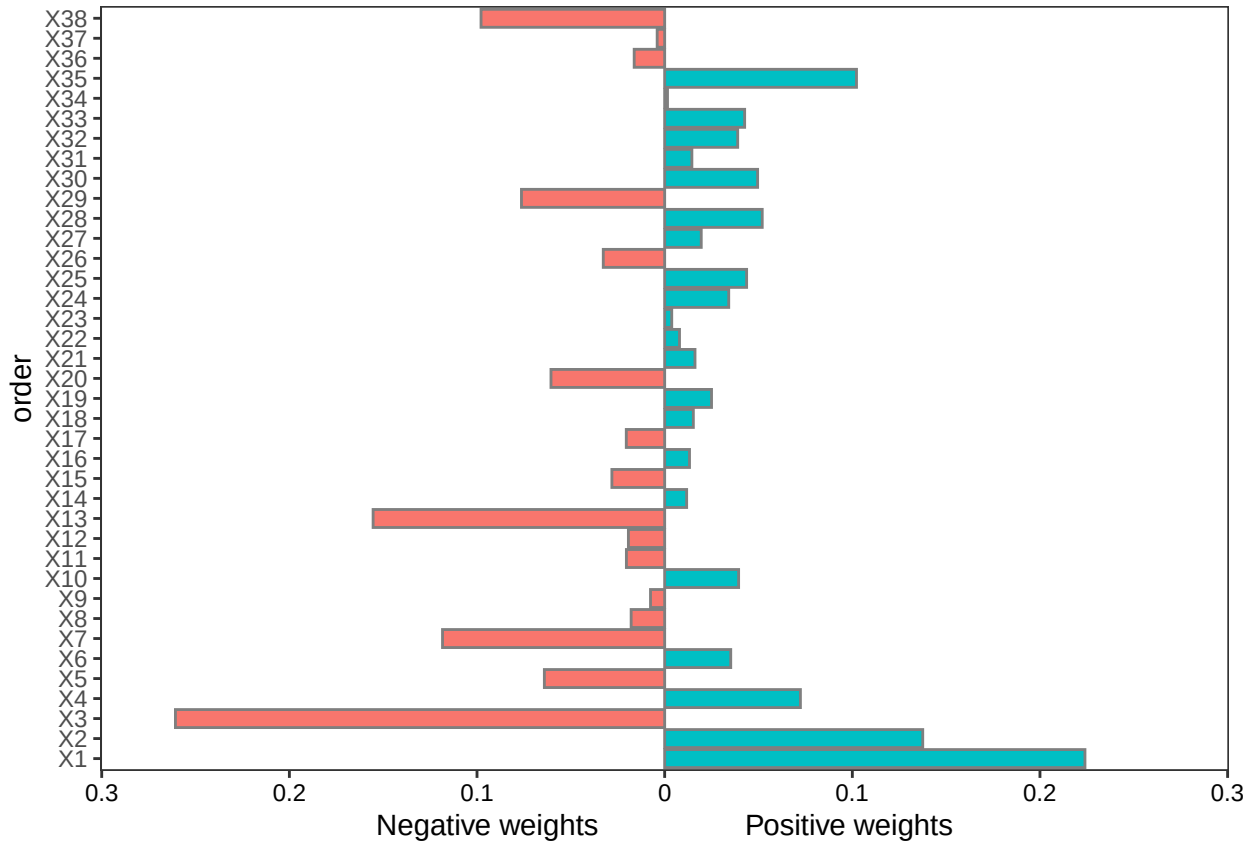
MixEffect	LowerCI	UpperCI	Positive.Size	Negtive.Size	t.stat	P.value
14.59182	0.7342408	28.44939	45.66953	31.07772	2.063812	0.041156

```
res5$Y1_gaussian_Imp %>%
  dplyr::slice(1:10) %>%
  knitr::kable(format = "latex",
               align = "l") %>%
  kableExtra::kable_styling(full_width = F,
                            latex_options = "striped",
                            position = "left",
                            font_size = 10)
```

Varname	Group	Weight
X1	Positive	0.2239659
X2	Positive	0.1375287
X35	Positive	0.1022171
X4	Positive	0.0723409
X28	Positive	0.0519899
X30	Positive	0.0495282
X25	Positive	0.0436662
X33	Positive	0.0426163
X10	Positive	0.0394013
X32	Positive	0.0389109

In this example, the exposure variables were categorized into ten levels according to their quantile values. Mixture effects of all exposures were estimated (i.e., *res5$Y1_gaussian_Coef*), while the relative importance (i.e., weight, *res5$Y1_gaussian_Imp*) was calculated with binary directions. Users can also utilize *VizMixGcomp* to draw feature importance plot.

```
# Draw plots with VizMixGcomp
res6 = VizMixGcomp(PID = res$PID,
                   VarsY = "Y1",
                   Brightness = "dark",
                   Palette = "default1")
res6$Y1_gaussian_Imp_dark_default1
```



Weighted quantile sum (WQS) regression

Weighted quantile sum regression is realized with *"gWQS"* package. Detailed information is available in https://cran.r-project.org/web/packages/gWQS/gWQS.pdf. WQS model also yields estimates of the mixture effect of all exposures. Unlike *gcomp* package, users have to determine the mixture effect direction when they specify model building options. That is, if the mixture effect direction is considered as positive, then "b1_pos = 'F'," should be added when applying *MixWQS* function. For other parameters, users can access detailed information from the above website.

```
# Quantile g-computation
res7 = MixWQS(PID=res$PID,
              VarsY = "Y1",
              VarsX = "all.x" ,
              Covariates = "all.c",
              Family = "gaussian" ,
              VarStrat = "none",
              RatioValidat = 0.3,
              q = 10,
              b=100,
              b1_pos = 'F',
              b1_constr = 'F')
res7$Y1_gaussian_Imp %>%
    dplyr::slice(1:10) %>%
    knitr::kable(format = "latex",
                 align = "l") %>%
    kableExtra::kable_styling(full_width = F,
                              latex_options = "striped",
```

```
                                position = "left",
                                font_size = 10)
```

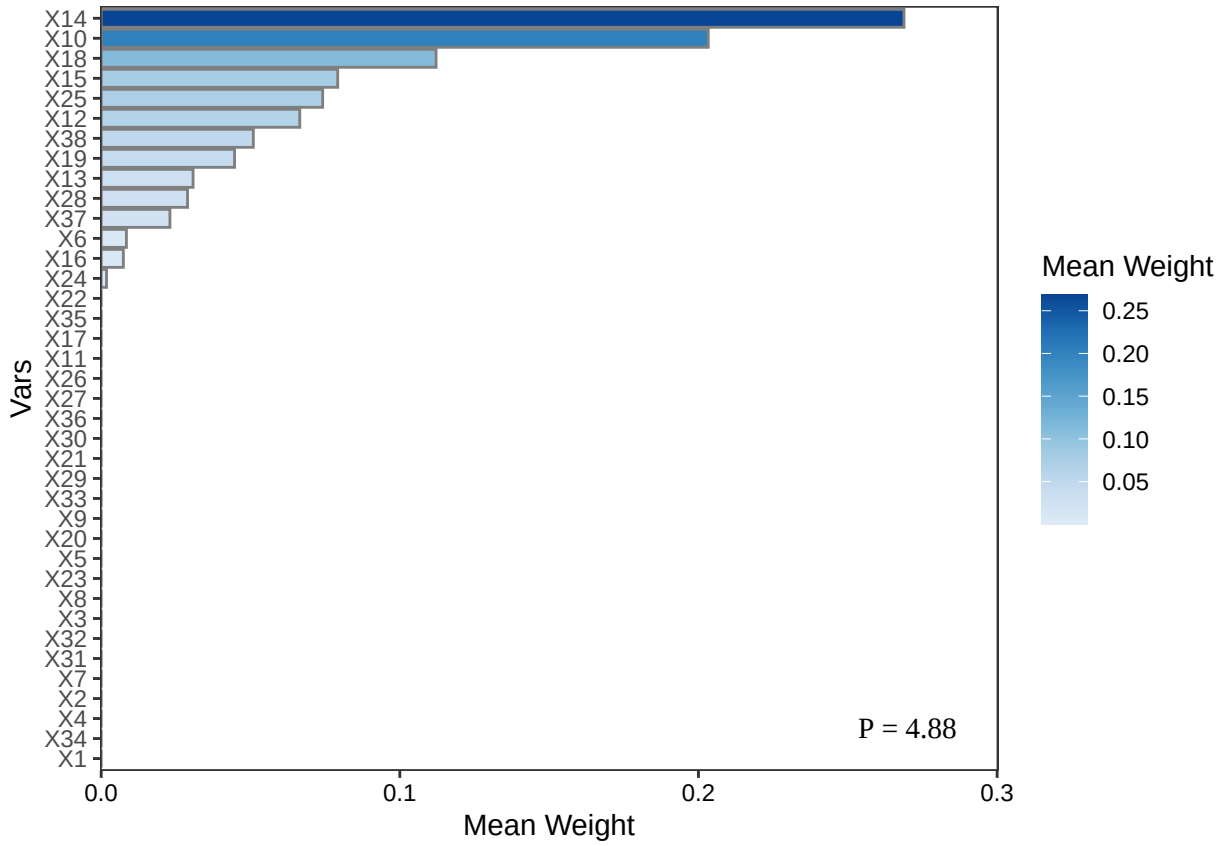
	mix_name	mean_weight
X14	X14	0.2688227
X10	X10	0.2032716
X18	X18	0.1121344
X15	X15	0.0791927
X25	X25	0.0741552
X12	X12	0.0664870
X38	X38	0.0509376
X19	X19	0.0446588
X13	X13	0.0307692
X28	X28	0.0289157

```
res7$Y1_gaussian_Coef %>%
    knitr::kable(format = "latex",
                 align = "l") %>%
    kableExtra::kable_styling(full_width = F,
                                latex_options = "striped",
                                position = "left",
                                font_size = 10)
```

Estimate	Std.Error	t	P
4.880115	3.665781	1.331262	0.1864652

In this example, the exposure variables were categorized into ten levels according to their quantile values. Mixture effects of all exposures were estimated (i.e., *res7$Y1_gaussian_Coef*), while the relative importance (i.e., weight, *res7$Y1_gaussian_Imp*) was calculated. Users can also utilize *VizMixWQS* to draw feature importance plot.

```
# Draw plots with VizMixWQS
res8 = VizMixWQS(PID = res$PID,
                 VarsY = "Y1",
                 Brightness = "dark",
                 Palette = "default1")
res8$Y1_gaussian_Imp_dark_default1
```



Bayesian kernel machine regression (BKMR)

BKMR model is performed with *"bkmr"* package. Detailed information is available in https://jenfb.github.io/bkmr/overview BKMR model the overall mixture effect of all exposures, individual exposure-response relations, and interaction effects between exposures. In **StatMix**, we can use *MixBKMR* to build such models. Noted that the execution of BKMR model can be time-consuming. So, in this example, we use a subset of exposures (i.e., X4-X10) for model fitting with a small iteration of 2000, which is suggested to be no less than 10,000 iterations in formal model construction.

```
# BKMR model
res9 = MixBKMR(PID = res$PID,
               VarsY = "Y1",
               VarsX = "X4,X5,X6,X7,X8,X9,X10",
               Covariates = "all.c",
               Family = "gaussian" ,
               Group = 'F',
               Iter = 2000,
               qfixed = 0.5,
               qsbivar = "default" ,
               qsoverall = "default",
               qsdiff = "default")
res9$Stat_PipUnivar %>%
    knitr::kable(format = "latex",
                 align = "l") %>%
    kableExtra::kable_styling(full_width = F,
                              latex_options = "striped",
                              position = "left",
```

```
                                font_size = 10)
```

variable	PIP
X4	1.000
X5	0.077
X6	0.286
X7	0.272
X8	0.460
X9	0.179
X10	0.148

```
res9$Stat_RiskOverall %>%
    knitr::kable(format = "latex",
                 align = "l") %>%
    kableExtra::kable_styling(full_width = F,
                              latex_options = "striped",
                              position = "left",
                              font_size = 10)
```

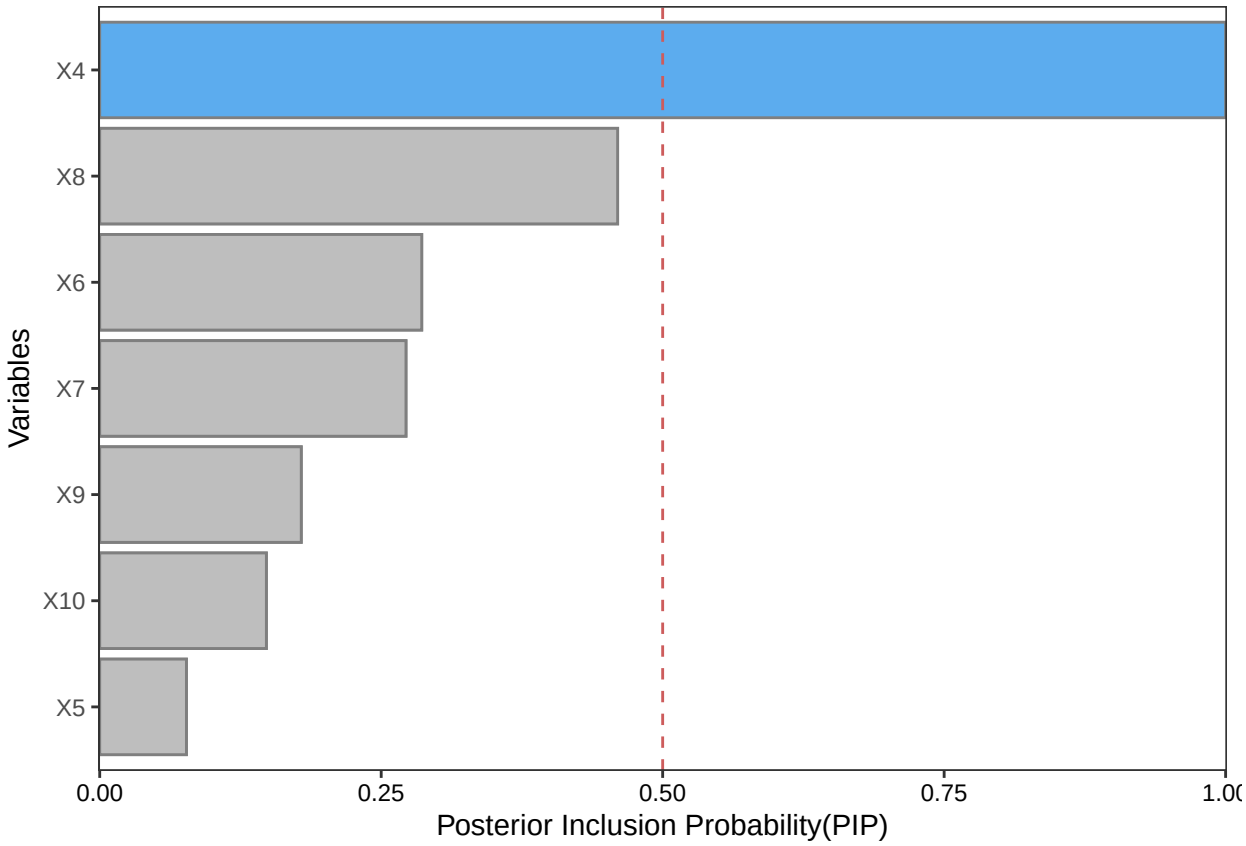
quantile	est	sd
0.25	-8.486775	4.2655923
0.30	-6.826126	3.2965222
0.35	-5.310022	2.5033213
0.40	-3.402295	1.6842014
0.45	-1.547972	0.8163491
0.50	0.000000	0.0000000
0.55	1.711898	1.1143140
0.60	3.317883	2.6676038
0.65	3.649963	4.3444281
0.70	3.556323	4.7987953
0.75	2.803040	5.7531598

The *MixBKMR* function produces various modelling results and they are stored in R6 object (i.e., res9). In this example, the 50% quantile level was utilized as reference level in overall effect estimation with "qfixed = 0.5" option. Mixture effects of all exposures were estimated (i.e., *res9$Stat_RiskOverall*), while the relative importance [i.e., the posterior inclusion probabilities (PIPs), *res9$Stat_PipUnivar*] was calculated. Users can also utilize *VizMixBKMR* to draw plots.

The *VizMixBKMR* function produces series lists of plots. Users can check *res10$BKMRPlot* for detailed results. Here, we mainly display three plots: 1) ranked PIP plot; 2) Bivariate exposure-response relations towards dependent variable; and 3) Overall mixture effects for disease risk estimation.

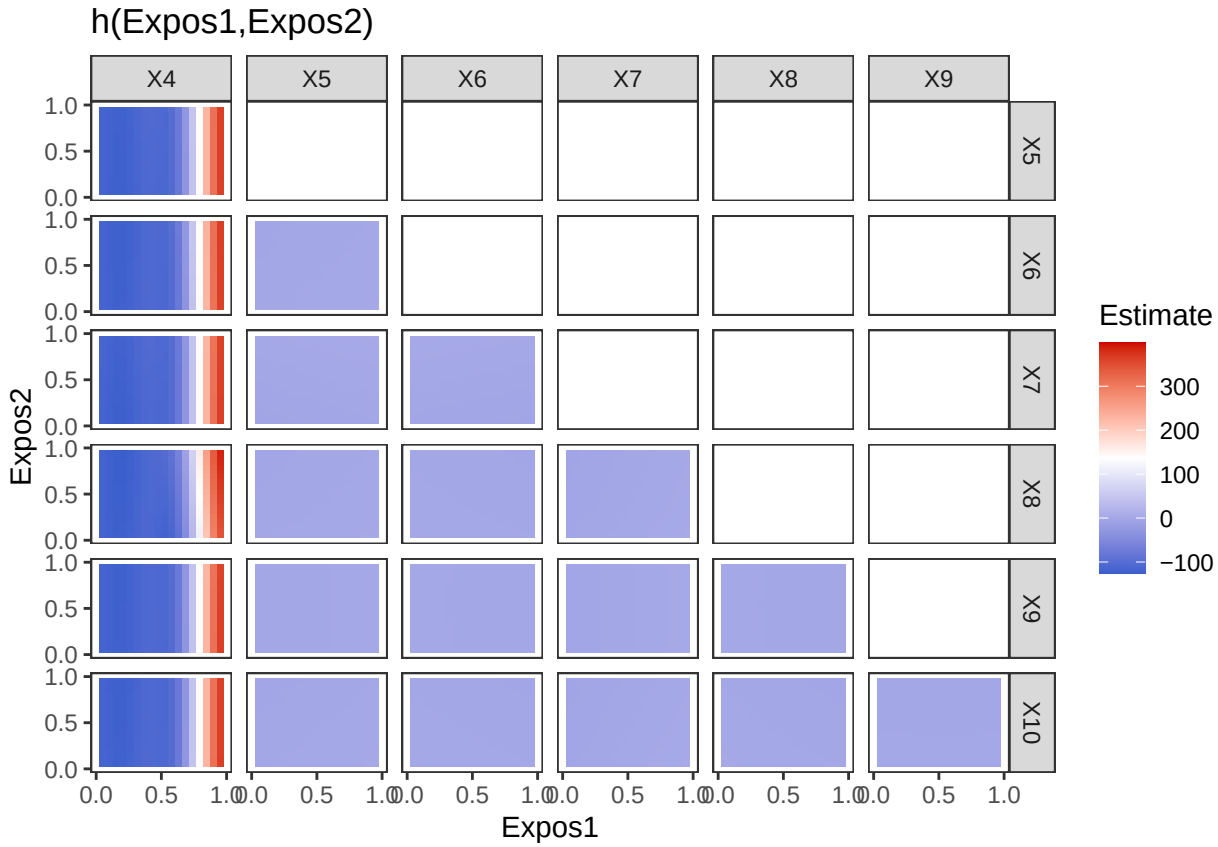
```
# Draw plots with VizMixBKMR
res10 = VizMixBKMR(PID=res$PID,
                   VarsY = "Y1",
                   Brightness = "dark",
                   Palette = "default1")
# PIP plot
res10$BKMRPlot$plot_pip.univar
```

```
## $Y1_gaussian_Imp_dark_default1
```



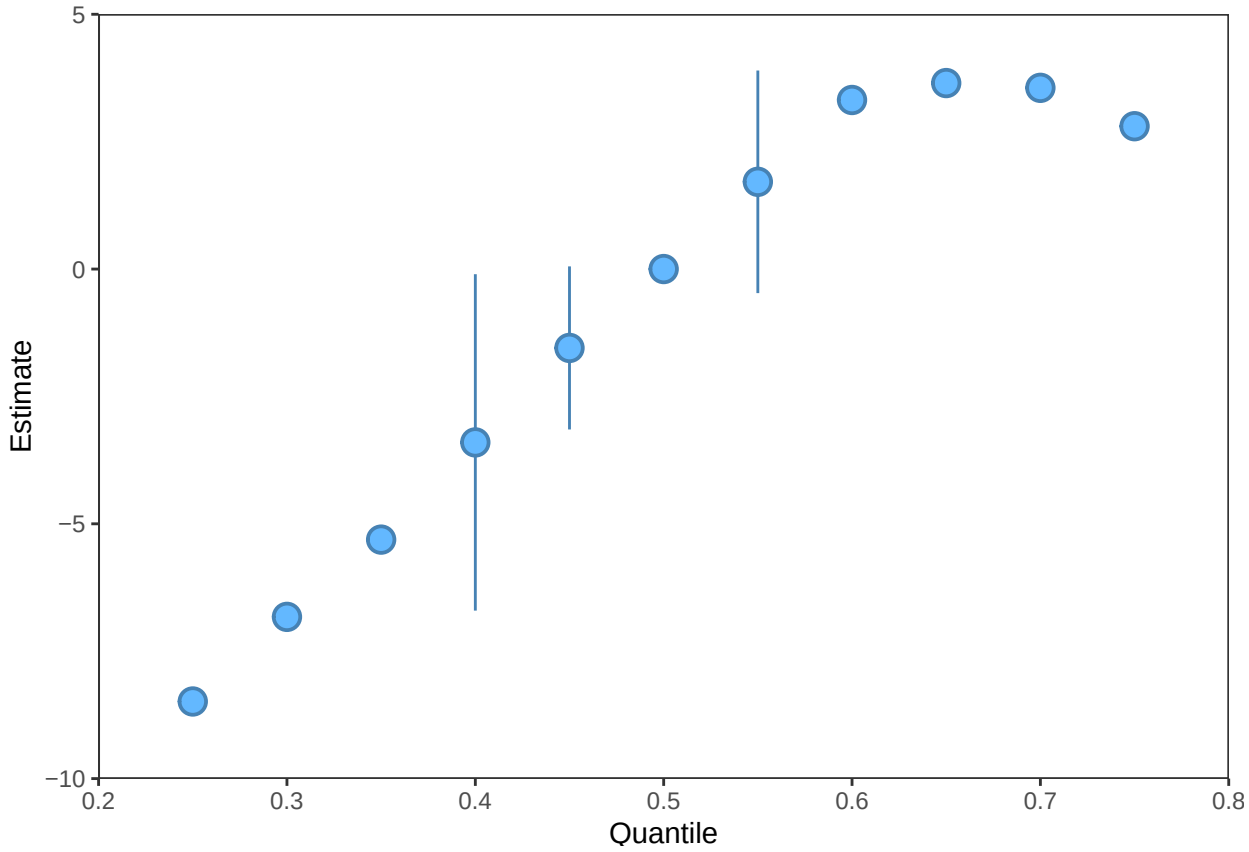

```
# Bivariate plot
res10$BKMRPlot$plot_pred.resp.bivar.all
```

```
## $Y1_gaussian_Imp_dark_default1
```




```
# Overall risk plot
res10$BKMRPlot$plot_risks.overall
```

```
## $Y1_gaussian_Imp_dark_default1
```



Exit exposome StatMix module analyses

When finishing your analyses task and saving all the results needed (**That is vital for the users.**), remember run the exit function provided in that package, which will release the memory of R environment and is of helpful for both the users and the ExposomeX platform.

```
FuncExit(PID = res$PID)
```

```
## [1] "Success to exit. Thanks for using ExposomeX platform!"
```

6.9 StatIP function

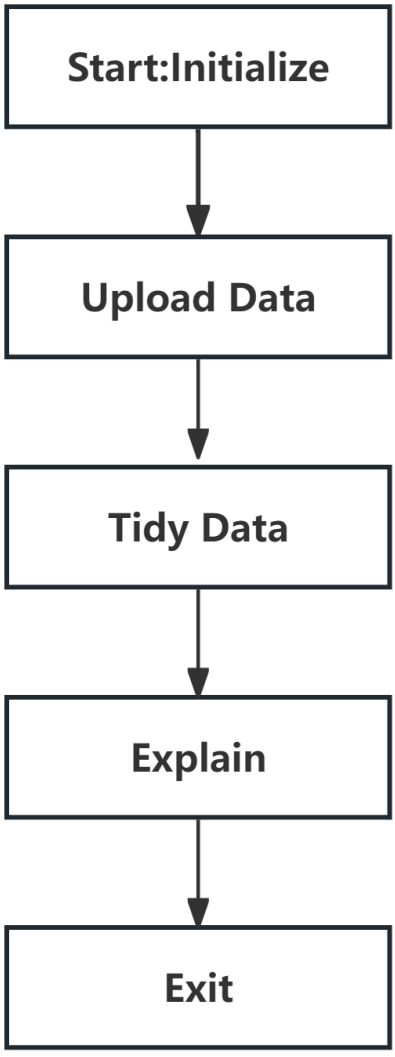
6.9.1 Application domain

StatIP is designed to find the statistical relationships between exposure factors and health outcome.

6.9.2 Theory

Users can easily get the modeling results and their visualization plots with high quality by following the detailed instructions in each step. The statistical interpretations from the perspectives of subjects and whole dataset are both provided.

6.9.3 Work pipeline



6.9.4 Use example

Initialize package

Make sure that the required packages is already installed.

```
library(tidyverse)
library(gridExtra)
# devtools::install_github("ExposomeX/exposomex", force = TRUE)
library(exposomex)
```

At first, you need to initialize the calculation environment using a series of initialization functions, e.g., InitStatIP, InitStatMo, InitStatTidy, InitEVIZ, InitEBIO, etc. Here, we use the "StatIP" for data analysis for example. The detailed information about the functions and returned value will be introcuced in the following chapters.

```
res = InitStatIP()
res
```

```
## <eSet>
##   Public:
##     AddCommand: function (x)
##     AddLog: function (x)
##     clone: function (deep = FALSE)
##     EpiDesign: NULL
##     ExcecutionLog: Complete initializing the ExpoStat module.2024.06.23 17. ...
##     Expo: list
##     FileDirIn: NULL
##     FileDirOut: /home/ubuntu/ExposomeX/R_Exposome_1.0/output_171104.3725 ...
##     PID: 171104.372544WWKJZB
##     RCommandLog: eSet <- InitStatIP(PID = Any ID your like, FileDirOut =  ...
##     VarsDel: NULL
```

Here, we can see that the returned value "res" is an R6 object. It contains an unique program ID of res$PID (e.g., "100737GJMWJA"), which is random generated by the system. Users need use it in the following step for further data process.

Upload data

Secondly, you need to load data file for data process. The PID Parameter, you can enter res$PID which is random generated by the system when you run the InitStatIP function. If you want to try this package at first, you can input the UseExample Parameter with "example#1" to use our example data. Or you can enter *UseExample = "default"* to use your own data, you can enter *DataPath = " "* and *VocaPath = " "* to choose you own input data file and vocabulary file directories. The demand of these Parameters can see in the help of *LoadStatIP* function.

You should note that the format of the input data file and vocabulary file is ruled. You can see the demand of the data format by visiting the following website http://www.exposomex.cn.

The details for each parameter are listed below:

PID
: chr. Program ID. It must be the same with the PID generated by *InitStatIP*.

UseExample

chr. Method of uploading data. If “default”, user should upload their own data files,or use “example#1” provided by this module.

DataPath

chr. Input data file directory, e.g. “D:/test/eg_statcros_data.xlsx”. It should be noted that the slash symbol is “/”, not “\”.

VocaPath

chr. Input vocabulary file directory, e.g. “D:/test/eg_statcros_voca.xlsx”. It should be noted that the slash symbol is “/”, not “\”.

```
res1 <- LoadStatIP(PID = res$PID,
                   UseExample = "example#1")

res1$Expo$Voca %>%
  dplyr::slice(1:20)
```

```
## # A tibble: 12 x 5
##    SerialNo SerialNo_Raw FullName GroupName Type
##    <chr>    <chr>        <chr>    <chr>     <chr>
##  1 Y1       Y1           Y_disc   Outcome   cate
##  2 Y2       Y2           Y_cont   Outcome   cont
##  3 X1       X1           TE_1     Chemical  cont
##  4 X2       X2           TE_2     Chemical  cont
##  5 X3       X3           TE_3     Chemical  cont
##  6 X4       X4           TE_4     Chemical  cont
##  7 X5       X5           TE_5     Chemical  cont
##  8 X6       X6           TE_6     Chemical  cont
##  9 X7       X7           TE_7     Chemical  cont
## 10 X8       X8           TE_8     Chemical  cont
## 11 X9       X9           CH1      Chemical  cont
## 12 X10      X10          CH2      Chemical  cont
```

```
res1$Expo$Data %>%
  dplyr::select(SampleID:X2) %>%
  dplyr::slice(1:20)
```

```
## # A tibble: 20 x 6
##    SampleID SubjectID    Y1    Y2    X1    X2
##    <chr>    <chr>     <dbl> <dbl> <dbl> <dbl>
##  1 Tr1      S1            1  -101  7.76 10.2
##  2 Tr2      S2            0   -51 10.1  11.6
##  3 Tr3      S3            0   -37  8.54  9.52
##  4 Tr4      S4            1   -61 14.2  14.9
##  5 Tr5      S5            0   -28 11.0  16.4
##  6 Tr6      S6            0    -8 11.3  12.7
##  7 Tr7      S7            1   -63  7.66  7.73
##  8 Tr8      S8            0   -35 11.3   8.25
##  9 Tr9      S9            0   -14 14.5  11.1
## 10 Tr10     S10           1   -99  6.26  6.04
## 11 Tr11     S11           0   -60  3.43 10.3
## 12 Tr12     S12           0   -32  6.75  6.86
## 13 Tr13     S13           0   -73  7.71  9.47
## 14 Tr14     S14           0   -18 12.6   8.97
```

```
## 15 Tr15     S15           0   -48 11.7   8.60
## 16 Tr16     S16           0   -20  7.10 12.5
## 17 Tr17     S17           0    -9  6.32 49.1
## 18 Tr18     S18           1   -98  4.66  8.49
## 19 Tr19     S19           0   -70  2.44 -2.27
## 20 Tr20     S20           0   -36 10.6   7.07
```

Here, we can see that the returned value "res1" is an R6 object. It contains two data frames which are your input data. Users need use it in the following step for further data process.

By now, we have prepared our dataset ready for the following calculation.

Tidy data

Tidy the data for following modeling, including deleting missing variables and low variation variables.

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitStatIP*.
OutPath
chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.

```
res2 <- TidyStatIP(PID = res$PID,
                   OutPath = "default")
```

Model: Association

The details for each parameter are listed below:

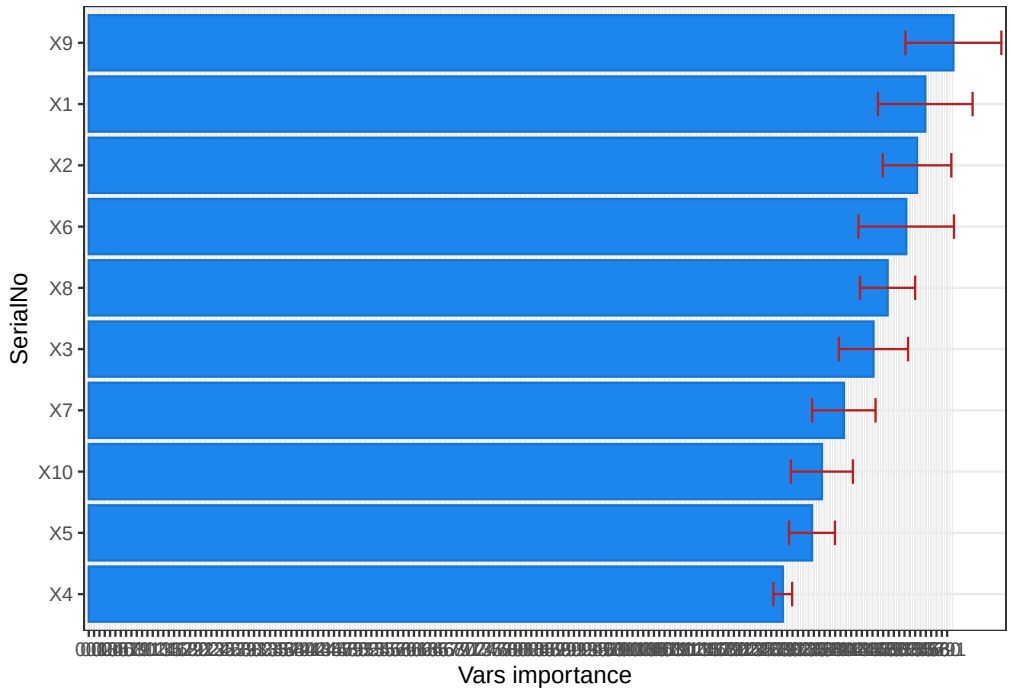
PID
chr. Program ID. It must be the same with the PID generated by *InitStatCros*.
OutPath
chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.
VarsY
chr. Outcome variable used for modeling. Only one variable can be entered.
VarsX
chr. Exposure variable used for modeling. The default option is "all.x" (All exposure variables are included). Users can also choose available variables. It should be noted that there is fixed format for the entering characters separated with comma and without space, e.g., "X1,X2,X3".
LinkModel
chr. Methods to interpret the model. Options include "ranger" (random forest), "glmnet" (elastic net), "svm" (support vector machine), "glm" (linear regression), "gam" (generalized additive model), and "xgboost" (eXtreme gradient boosting).
ObsrPartType
chr. Type of transformation that should be applied for dropout loss. Options include "raw" (drop losses), "ratio" (drop_loss/drop_loss_full_model), and "difference" (drop_loss - drop_loss_full_model).
ObsrPartNum
chr. Number of observations that should be sampled for calculation of variable importance. The default means variable importance will be calculated on whole dataset (no sampling). If "defult", use all Obsrrvations.

ObsrProfType
chr. Type of variable profile. Options include “partial”, “conditional”, and “accumulated”.
ObsrProfNum
int. Number of observations used for calculation of aggregated profiles. By default 100.
ObsrProfVars
chr. Names of variables to be explained. If “all.x”, all “X variable” above are chosen.
ObsrProfGeom
chr. Layout of the explanation profile in dataset level including “aggregates”, “profiles” or “points”.
SubjPredSeq
chr. Subjects which need explanation. Options include “all” (all the subjects), “none” (no subjects), and “other” (copy the subject list by clicking “Available vars”).
SubjPartType
chr. Layout of the explanation profile in subject level. Options include “shap”, “oscillations”, “break_down”, and “none”.

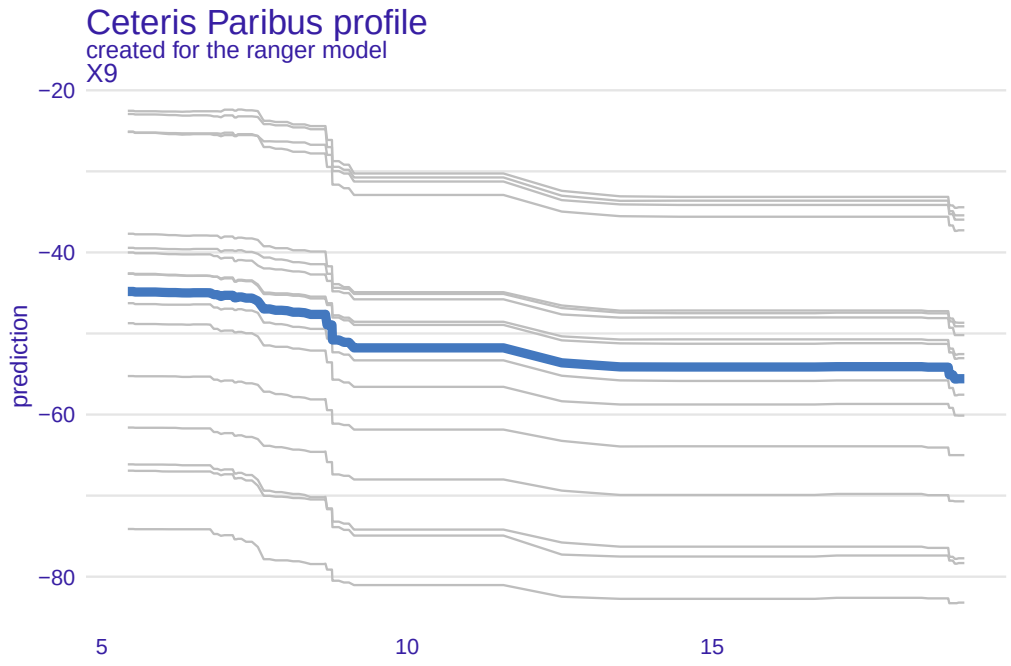
```
res3 = StatIPCros(PID = res$PID,
                  OutPath = "default",
                  VarsY = "Y2",
                  VarsX = "all.x",
                  LinkModel = "ranger",
                  ObsrPartType = "raw",
                  ObsrPartNum = "50",
                  ObsrProfType = "partial",
                  ObsrProfNum = "100",
                  ObsrProfVars = "X9",
                  ObsrProfGeom = "profiles",
                  SubjPredSeq = "S1,S2,S3",
                  SubjPartType = "break_down")

res3$CrosVipTable$Y2_importance_ranger_raw
```

```
## # A tibble: 10 x 5
##    SerialNo FullName GroupName VIP_mean VIP_sd
##    <fct>    <chr>    <chr>        <dbl>  <dbl>
##  1 X4       TE_4     Chemical      13.0  0.176
##  2 X5       TE_5     Chemical      13.6  0.431
##  3 X10      CH2      Chemical      13.8  0.582
##  4 X7       TE_7     Chemical      14.2  0.594
##  5 X3       TE_3     Chemical      14.7  0.648
##  6 X8       TE_8     Chemical      15.0  0.517
##  7 X6       TE_6     Chemical      15.3  0.896
##  8 X2       TE_2     Chemical      15.5  0.643
##  9 X1       TE_1     Chemical      15.7  0.887
## 10 X9       CH1      Chemical      16.2  0.899
```

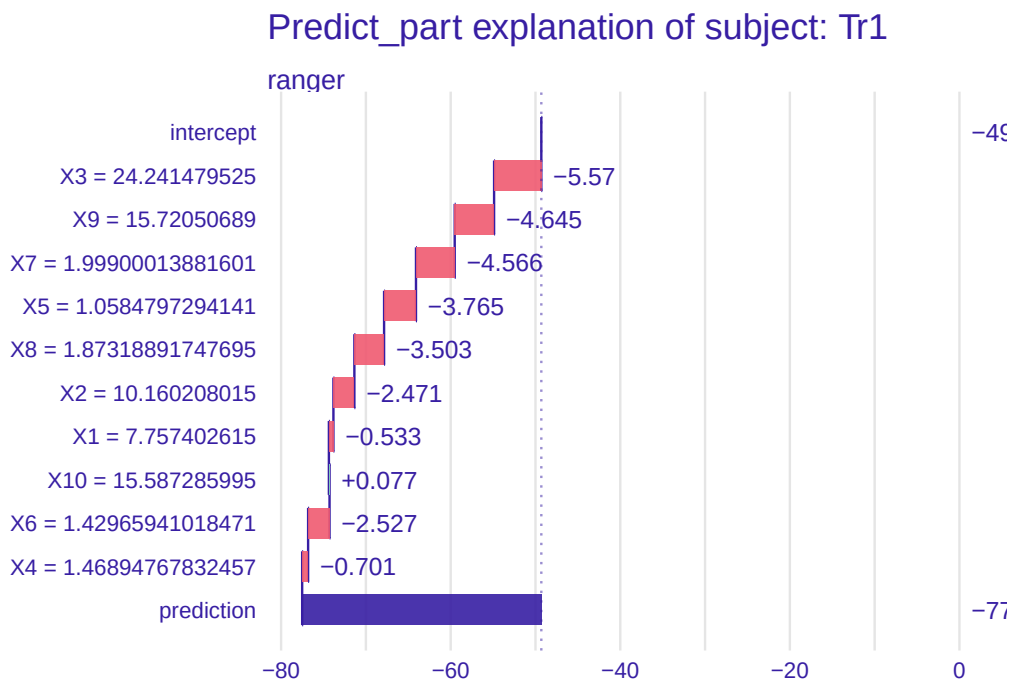
```
res3$CrosVipPlot$Y2_importance_ranger_raw
```




```
res3$CrosProfPlot$Y2_ranger_partial_profiles_by_X9
```




```
res3$CrosSubjPlot$Y2_ranger_break_down_S1
```



Here, we can see that the returned value "res3" contains a result table which includes the association analysis results.

```
FuncExit(PID = res$PID)
```

```
## [1] "Success to exit. Thanks for using ExposomeX platform!"
```

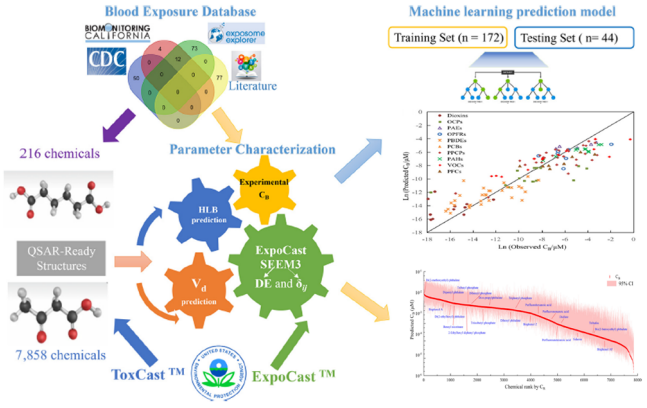
6.10 StatHEP function

6.10.1 Application domain

StatHEP was developed based on our previous study on the prediction of human blood exposome using a random forest model and its application in chemical risk prioritization. Please see the details in Zhao et al., Environ Health Perspect. 2023 Mar;131(3):37009.

6.10.2 Theory

Quantification of all trace organics in the biological fluids seems impossible and costly, regardless of the high individual exposure variability. We hypothesized that the blood concentration of organic pollutants could be predicted via their exposure and chemical properties. Developing a prediction model on the annotation of chemicals in human blood can provide new insight into the distribution and extent of exposures to a wide range of chemicals in humans.



6.10.3 Work pipeline

Initialize package

Make sure that the required packages is already installed.

```
# devtools::install_github("ExposomeX/exposomex", force = TRUE)
library(exposomex)
```

The first step,you need to initialize the environment using “InitStatHEP”.

```
res = InitStatHEP()
```

Here, we can see that the returned value “res” is an R6 object. It contains an unique program ID of res$PID (e.g., “100737GJMWJA”), which is random generated by the system. Users need use it in the following step for further data process.

Upload data

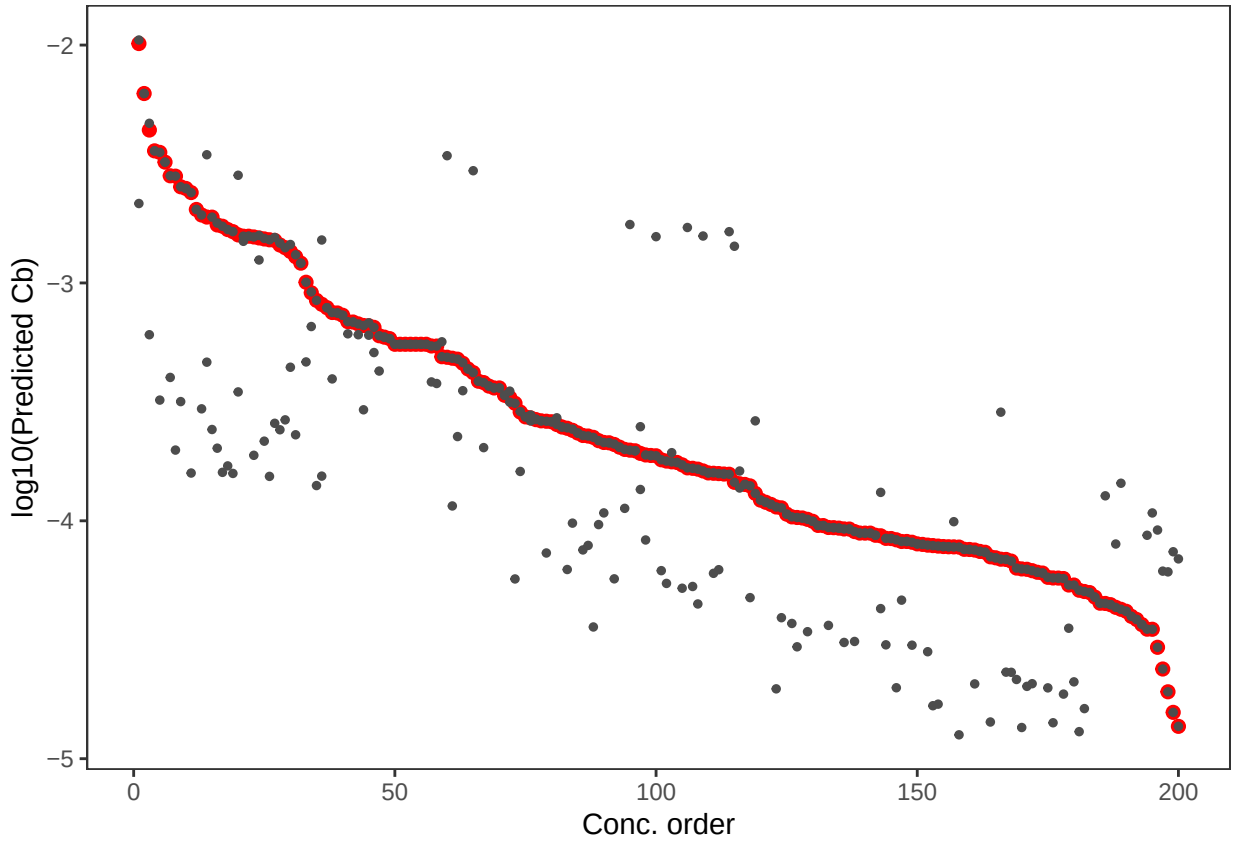
```
res1 = LoadStatHEP(PID = res$PID,
              UseExample="example#1")
```

Modeling This step is the most critical part of the entire module. We will build prediction model with function "HEPPredBlood". You can select whether to perform Mentocaro simulation and the times of Mentocaro simulation.

```
res2 = HEPPredBlood(PID=res$PID,
                    OutPath = "default",
                    MC = T,
                    N = 100)
```

Visualize model This step will visualize prediction data with function "VizHEPPredBlood". You can get different styles of images by selecting different parameters.

```
res3 = VizHEPPredBlood(PID=res$PID,
                       OutPath = "default",
                       MC = T,
                       Layout = "forest",
                       Brightness = "light",
                       Palette = "default1")
res3$eSet$plot_pred
```



```
FuncExit(PID = res$PID)
```

```
## [1] "Success to exit. Thanks for using ExposomeX platform!"
```

10 EMS module

10.1 EMS function

10.1.1 Application domain

EMS module is designed to conduct the analysis of the features from the high-resolution mass spectrometry. It mainly aims to screen and annotate the significant features associated with the health outcomes.

10.1.2 Theory

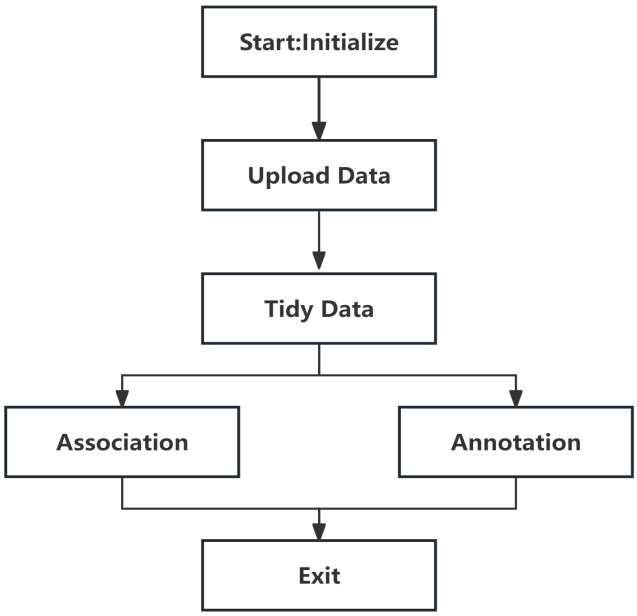
EMS can prioritize and annotate the features that are correlated with the diseases under different epidemiological design from the large exposome data acquired by high resolution mass spectrometry.

Stage-I: outcome directed non-targeted analysis (NTA). In this part, we aimed to develop outcome directed statistical methods to prioritize the mass-charge-ratio (m/z) features that can be linked with the health outcome. Firstly, the peak picking and alignment can be conducted using the "XCMS" R packages and the output table with alignment features was uploaded onto our system. In addition, all the covariates, like age, gender, and behaviour, were all included in the model. The health outcome was included as the dependent variable. According to the exposomic study design, different statistical analyses including linear regression model and more sophisticated machine learning models have been all included. For single-factor method, generalized linear regression is used to obtain the association between each factor and outcome, which are further screened by multiple-hypothesis test correction according to users' need. For multiple-factor method, stepwise, LASSO, and random forest are adopted to screen the features with close relationship with the outcome. In summary, for each analysis model, the features having statistically correlation with the health outcomes will be highlighted and compared. Finally, the highlighted features will be further searched using the above-mentioned mass spectrometry database. Either exogenous or endogenous compounds were included in the NTA analysis by using the chemical annotation database of "DB_NTA" from human blood exposome (20,756 parent and 115,398 metabolite chemicals) and HMDB Database (114,222 compounds). Mass-to-charge conversion of ion adduct for mass-spectrometry (DB_IonAdduct) were provided for users to search compounds in CASRN, formula, m/z, adduct search, and accuracy. A total of 31 adducts for the positive mode (e.g., M+H, M+3H, M+H+2Na, and M+ACN+2H) and 15 adducts for the negative mode (e.g., M-3H, M-2H, M+Na-2H, and M+FA-H) are considered to infer the possible exact molecular weights of the molecular ions, which will be used for peak annotations. The calculation of monoisotopic exact masses of molecular ion adducts were provided in Table S1. The screened significant features warrant more efforts spent to confirm their causal relationship associated with the diseases.

Stage-II: mass spectrometry search and compound annotation. To increase the applicability of mass spectrometry data processing, we have included the MS/MS annotation by incorporating the databases of MassBank of North America (MoNA). In addition, we also included the retention time (RT) prediction tools, which was developed in our previous studies as another filter to increase the reliability of the annotation. The entire workflow was shown in Figure S2. Briefly, the MS raw data was processed using third-party tools such as XCMS and MSDIAL. Subsequently, MS list was generated and correlated with diseases to screen the MS1 features that are of interest. For each MS1, its MS/MS fragments were acquired either data-independent or data-dependent acquisition, which was further annotated against the MS/MS database as mentioned above. Meanwhile, RT prediction was conducted using an input calibration file (e.g., RT of primary metabolites as the demonstration data in this study). Therefore, E-MS has greatly increased the analytical performance by adding the MS, MS/MS, and RT matching scores.

10.1.3 Work pipeline

Users can easily get the modeling results and their visualization plots with high quality by following the detailed instructions in each step. Three methods are adopted to screen the important features, including "Stepwise" "LASSO", and "Random forest".



10.1.4 Use example

Initialize package

Make sure that the required packages is already installed.

```
library(tidyverse)
# devtools::install_github("ExposomeX/exposomex", force = TRUE)
library(exposomex)
```

At first, you need to initialize the calculation environment using a series of initialization functions, e.g., InitStatCros, InitStatMo, InitStatTidy, InitEVIZ, InitEBIO, etc. Here, we use the module "EMS" for analysis of the features from the high-resolution mass spectrometry for example. The detailed information about the functions and returned value will be introcuced in the following chapters.

```
res = InitEMS()
res
```

```
## <eSet>
##   Public:
##     AddCommand: function (x)
##     AddLog: function (x)
##     clone: function (deep = FALSE)
##     DataStr: NULL
##     EpiDesign: NULL
##     ExcecutionLog: Complete initializing the ExpoNontarget module.2024.06.2 ...
##     Expo: list
##     ExpoDel: list
##     FileDirIn: NULL
##     FileDirOut: /home/ubuntu/ExposomeX/R_Exposome_1.0/output_000431.4078 ...
##     ModelBank: spec_tbl_df, tbl_df, tbl, data.frame
##     PID: 000431.407875DCVUJV
##     RCommandLog: eSet <- InitNTA()
```

Here, we can see that the returned value "res" is an R6 object. It contains an unique program ID of res$PID (e.g., "100737GJMWJA"), which is random generated by the system. Users need use it in the following step for further data process.

Upload data

Secondly, you need to load data file for data process. The PID Parameter, you can enter res$PID which is random generated by the system when you run the InitEMS function. If you want to try this package at first, you can input the UseExample Parameter with "example#1" to use our example data. Or you can enter *UseExample = "default"* to use your own data, you can enter *DataPath = " "* and *VocaPath = " "* to choose you own input data file and vocabulary file directories. The demand of these Parameters can see in the help of *LoadEMS* function.

You should note that the format of the input data file and vocabulary file is ruled. You can see the demand of the data format by visiting the following website http://www.exposomex.cn.

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitEMS*.
MS2
lgl. Whether to use MS2 information to annotate the target ion. If use example data, choose "T" to use "example#2" data with MS2 information, choose "F" to use "example#1" data without MS2 information.
UseExample
chr. Method of uploading data. If "default", user should upload their own data files,or use "example#1" provided by this module.
DataPath
chr. Input data file directory, e.g. "D:/test/eg_statcros_data.xlsx". It should be noted that the slash symbol is "/", not "\".
VocaPath
chr. Input vocabulary file directory, e.g. "D:/test/eg_statcros_voca.xlsx". It should be noted that the slash symbol is "/", not "\".

```
res1 <- LoadEMS(PID = res$PID,
                MS2 = T,
                UseExample = "example#2")

res1$Expo$Voca %>%
  dplyr::slice(1:20)
```

```
## # A tibble: 20 x 7
##    SerialNo SerialNo_Raw FullName       Type  ExactMass    RT IonMode
##    <chr>    <chr>        <chr>          <chr>     <dbl> <dbl> <chr>
##  1 Y1       Y1           Y_cont         cont          0   NA   <NA>
##  2 Y2       Y2           Y_disc         cate          0   NA   <NA>
##  3 C1       C1           C1             cont          0   NA   <NA>
##  4 C2       C2           C2             cont          0   NA   <NA>
##  5 X1       X1           X100.0757_1.8  cont        100.   1.8 positive
##  6 X2       X2           X100.0758_1    cont        100.   1   positive
##  7 X3       X3           X100.0758_2.3  cont        100.   2.3 positive
##  8 X4       X4           X100.0758_7.6  cont        100.   7.6 positive
##  9 X5       X5           X101.0243_0.5  cont        101.   0.5 negative
## 10 X6       X6           X101.0244_3.3  cont        101.   3.3 negative
## 11 X7       X7           X101.0244_7.7  cont        101.   7.7 negative
## 12 X8       X8           X101.0607_3.1  cont        101.   3.1 negative
## 13 X9       X9           X101.0608_1.8  cont        101.   1.8 negative
## 14 X10      X10          X101.1074_3.8  cont        101.   3.8 positive
## 15 X11      X11          X102.0197_7.1  cont        102.   7.1 negative
## 16 X12      X12          X102.0373_6.9  cont        102.   6.9 positive
## 17 X13      X13          X102.055_10.7  cont        102.  10.7 positive
## 18 X14      X14          X102.0551_10.3 cont        102.  10.3 positive
## 19 X15      X15          X102.0551_6.9  cont        102.   6.9 positive
## 20 X16      X16          X102.0561_10.2 cont        102.  10.2 negative
```

```
res1$Expo$Data %>%
  dplyr::select(SampleID:X2) %>%
  dplyr::slice(1:20)
```

```
## # A tibble: 20 x 8
##    SampleID SubjectID    Y1    Y2    C1    C2      X1      X2
##    <chr>        <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>   <dbl>
##  1 001_26_A         1  -101     0    NA     7   8122.  36221.
##  2 001_33_B         1   -51     0    NA   779   9570.  26138.
##  3 001_35_C         1   -37     1    30   613   7945.  30597.
##  4 003_25_A         3   -61     0    NA   454 111901. 357396.
##  5 003_30_B         3   -28     1    74   766 115987. 460164.
##  6 003_32_C         3    -8     1    NA   768  14569.  62445.
##  7 005_25_A         5   -63     0    NA   418  26938.  78447.
##  8 005_29_B         5   -35     1    NA   484   8483.  32360.
##  9 005_32_C         5   -14     1    NA   647   7916.  39186.
## 10 006_25_A         6   -99     0    NA   329  15078.  47967.
## 11 006_30_B         6   -60     0    NA   556  11897.  35588.
## 12 006_34_C         6   -32     1    NA    NA  57953. 143940.
## 13 007_28_A         7   -73     0    NA   212  40627. 138281.
## 14 007_36_B         7   -18     1    72   494 114173. 269742.
## 15 008_28_A         8   -48     0    NA   266  41954. 153638.
## 16 008_32_B         8   -20     1    NA   792  50094. 175658.
## 17 008_33_C         8    -9     1    NA   495  10617.  28749.
## 18 012_25_A        12   -98     0    NA   670   7115.  16928.
## 19 012_29_B        12   -70     0    NA    76   8822.  22259.
## 20 012_34_C        12   -36     1    NA   687  10294.  23238.
```

Here, we can see that the returned value "res1" is an R6 object. It contains two data frames which are your input data. Users need use it in the following step for further data process.

By now, we have prepared our dataset ready for the following calculation.

Tidy data

Tidy the data for following modeling, including deleting missing variables, deleting low variation variables, and transforming dummy variables.

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitEMS*.
OutPath
chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.
TransDummyVars
chr. Variables to be transformed as dummy variables. It should be noted that there is fixed format for the entering characters separated with comma and without space, e.g., "X1,X2,X3". If "default", all the factor variables will be transformed into dummy ones.

```
res2 <- TidyEMS(PID = res$PID,
                OutPath = "default",
                TransDummyVars = "default")
```

Users can also use other tidy functions in the StatTidy Module.

Model: Association

Association model —— NtaCros is used for association analysis for input data.

The details for each parameter are listed below:

PID
chr. Program ID. It must be the same with the PID generated by *InitEMS*.
OutPath
chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.
VarsY
chr. Outcome variable used for modeling. Only one variable can be entered.
VarsX
chr. Exposure variable used for modeling. The default option is "all.x" (All exposure variables are included). Users can also choose available variables. It should be noted that there is fixed format for the entering characters separated with comma and without space, e.g., "X1,X2,X3".
VarsN
chr. Choose the single factor or multiple factor model. Available options include "single.factor" and "multiple.factor".
FdrCorrect
lgl. T (or TRUE) and F (or FALSE). Whether to correct the multiple hypotheses by false positive rate (FDR) method.
SelMethod
chr. Methods to select the significant features. Options include "stepwise" (multiple linear regress using stepwise algorithm), "lasso" (multiple linear regress using LASSO regularization algorithm), "random.forest" (random forest).
StepwizeThr

num. Threshold of the P value for stepwise regression to screen important variables. It ranges 0.05-0.25 with the default value of 0.1.

RF_ImpThr

num. Threshold of the total importance for the variables to a random forest model. It ranges 0.5-1.0 with the default value of 0.9.

Covariates

chr. Covariates used for modeling. The default option is "all.c" (All covariates are included).

Family

chr. The link function for the regression model according the data type of outcomes, including "gaussian" for continuous variable, "binomial" for binary variable, and "poisson" for counting variable.

RepMsr

lgl. T (or TRUE) and F (or FALSE). Whether existing repeated measurement of the subjects.

Corstr

chr. If "RepMsr" = T, the generalized estimating equations (GEE) will be used. For GEE, three correlation structure options are "exchangeable" "ar1" "unstructured".

```
res3 = NtaCros(PID = res$PID,
               OutPath = "default",
               VarsY = "Y1",
               VarsX = "all.x",
               VarsN = "single.factor",
               FdrCorrect = "T",
               SelMethod = "lasso",
               StepwizeThr = 0.1,
               RF_ImpThr = 0.9,
               Covariates = "all.c",
               Family = "gaussian",
               RepMsr = "F",
               Corstr = "ar1")

res3$Y1_single.factor
```

```
## # A tibble: 110 x 11
##    SerialNo Vars.dummy Importance FullName      ExactMass beta.value  ci_l  ci_h
##    <chr>    <chr>           <dbl> <chr>             <dbl>      <dbl> <dbl> <dbl>
##  1 X1009    X1009            833. X9_1.9570003       165.      -25.5 -28.7 -22.3
##  2 X1007    X1007            753. X7_1.8209151       197.      -24.9 -28.2 -21.6
##  3 X1011    X1011            748. X11_2.0916213      221.      -24.9 -28.2 -21.6
##  4 X1008    X1008            737. X8_1.8952708       205.      -24.8 -28.1 -21.5
##  5 X1006    X1006            615. X6_1.7661906       165.      -23.7 -27.2 -20.2
##  6 X1010    X1010            561. X10_2.0268264      389.      -23.1 -26.8 -19.5
##  7 X899     X899             244. X207.1032_8.8      207.       18.3  14.0  22.5
##  8 X923     X923             219. X210.0435_3.3      210.      -17.7 -22.0 -13.3
##  9 X631     X631             208. X180.0656_3.3      180.      -17.4 -21.8 -13.0
## 10 X616     X616             195. X178.0508_3.3      178.      -17.0 -21.5 -12.6
## # i 100 more rows
## # i 3 more variables: p.value <dbl>, std.error <dbl>, formula <chr>
```

Here, we can see that the returned value "res3" contains a result table which includes the association analysis results.

Visualize association —— VizNtaCros is used to visualize association analysis.

The details for each parameter are listed below:

PID
: chr. Program ID. It must be the same with the PID generated by *InitEMS*.

OutPath
: chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.

VarsY
: chr. Outcome variable used for modeling. Only one variable can be entered.

VarsN
: Choose the single factor or multiple factor model. Available options include "single.factor" and "multiple.factor".

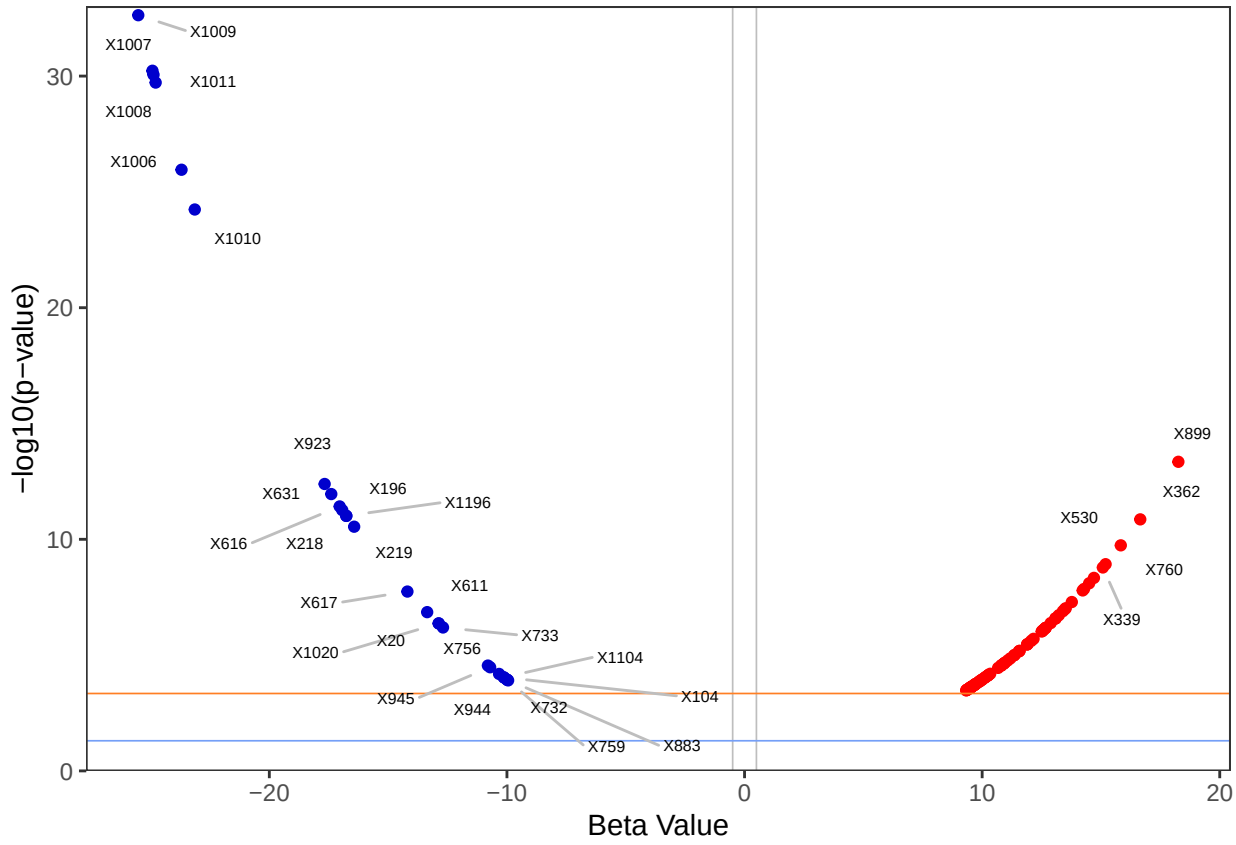
Layout
: chr. Visualization layout. Available options include "forest" and "volcano".

EffectThr
: num. Threshold of the total importance for the variables to a random forest model. It ranges 0.5-1.0 with the default value of 0.9.

Brightness
: chr. Visualization brightness. Available options include "light" and "dark".

Palette
: chr. Visualization palette. Available options include "default1", "default2" and several journal preference styles including "cell", "nature", "science", "lancet", "nejm", and "jama".

```
res4 = VizNtaCros(PID = res$PID,
                  OutPath = "default",
                  VarsY = "Y1",
                  VarsN = "single.factor",
                  Layout = "volcano",
                  EffectThr = 0.5,
                  Brightness = "light",
                  Palette = "default1")
res4$Y1_single.factor_volcano_light_default1
```



Model: Annotation

Annotation model —— NtaAnno is used to annotate the non-targeted features.

The details for each parameter are listed below:

PID

chr. Program ID. It must be the same with the PID generated by *InitEMS*.

OutPath

chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.

VarsY

chr. Outcome variable used for modeling. Only one variable can be entered.

VarsX

chr. Exposure variable used for modeling. The default option is "all.x" (All exposure variables are included). Users can also choose available variables. It should be noted that there is fixed format for the entering characters separated with comma and without space, e.g., "X1,X2,X3".

VarsN

chr. Choose the single factor or multiple factor model. Available options include "single.factor" and "multiple.factor".

FdrCorrect

lgl. T (or TRUE) and F (or FALSE). Whether to correct the multiple hypotheses by false positive rate (FDR) method.

AdductPos

chr. Adducts formed in the positive mode using LC-MS. The default is "all", i.e., all the adducts in positive

mode will be chosen, including "M+3ACN+2H,M+ACN+H,M+NH4,M+2ACN+2H,M+2H,M+3H,M+3Na,M+2Na-H,M+ACN+Na,M+H+Na, M+2K-H,M+H+NH4,2M+K,M+K,2M+NH4,2M+Na,M+2Na,M+DMSO+H,M+2ACN+H,M+M+2H+Na,M+ACN+2H,M+H,2M+H,M+CH3OH+H,M+H+2Na,M+Na,2M+ACN+H,2M+ACN+Na,M+IsoProp+H,M+

AdductNeg

chr. Adducts formed in the negative mode using LC-MS. The default is "all", i.e., all the adducts in positive mode will be chosen, including "M+FA-H,M+Hac-H,M+Br,3M-H,2M+Hac-H,M+K-2H,2M+FA-H,M-H,M-H2O-H,M+Na-2H, M-2H,M+TFA-H,M+Cl,M-3H,2M-H".

AccuracyMS1

num. Accuracy threshold to match the target compounds of MS1 information. The unit is ppm. It is usually set ranging 1-20.

MS2

lgl. T (or TRUE) and F (or FALSE). Whether to use MS2 information to annotate the target ion.

AccuracyMS2

num. Accuracy threshold to match the target compounds of MS2 information. The unit is ppm. It is usually set ranging 1-20.

ScoreMS2

num. The lowest matching score ranging 0-1 using MS2 information. The default is 0.4.

DiffRT

num. Difference of the retention time between the measured and the theoretical prediction. The unit is second (s). It is usually set ranging 60-120.

```
res5 = NtaAnno(PID = res$PID,
               OutPath = "default",
               VarsY = "Y1",
               VarsX = res2$Expo$Voca$SerialNo[3:20],
               VarsN = "single.factor",
               FdrCorrect = "F",
               AdductPos = "M+H",
               AdductNeg = "M-H",
               AccuracyMS1 = 20,
               MS2 = T,
               AccuracyMS2 = 20,
               ScoreMS2 = 0.4,
               DiffRT = 100)

res5$Y1_single.factor_20_20_100
```

```
## # A tibble: 403 x 12
##    SerialNo    RT ExactMass ExactMass_To_Monoisotopic MS1Accuracy Name    SMILES
##    <chr>    <dbl>     <dbl>                     <dbl>       <dbl> <chr>   <chr>
##  1 X1         1.8      100.                      99.1       0.102 N-Meth~ CN1CC~
##  2 X1         1.8      100.                      99.1       0.102 Butyl ~ CCCCN~
##  3 X1         1.8      100.                      99.1       0.102 N,N-Di~ CN(C)~
##  4 X1         1.8      100.                      99.1       0.102 N-Meth~ CN(C=~
##  5 X1         1.8      100.                      99.1       0.102 2,4-Di~ CC1OC~
##  6 X1         1.8      100.                      99.1       0.102 Cyclop~ ON=C1~
##  7 X1         1.8      100.                      99.1       0.102 1-(4,4~ C1CCN~
##  8 X1         1.8      100.                      99.1       0.102 1-(4,5~ C1CCN~
##  9 X1         1.8      100.                      99.1       0.102 1-[2-(~ C1CCN~
## 10 X1         1.8      100.                      99.1       0.102 Isoval~ CC(CC~
## # i 393 more rows
## # i 5 more variables: Monoisotopic_Mass <dbl>, IonMode <chr>, Adduct <chr>,
## #   Group <chr>, MS2Spectrum <chr>
```

```
res5$MS2_Spectum_Matching
```

```
## # A tibble: 12 x 24
##    chemical_name    msLevel rtime precursorMz precursorIntensity collisionEnergy
##    <chr>              <int> <dbl>       <dbl>              <dbl>           <dbl>
##  1 N,N-Dimethylgly~       2  602.        102.            702355.              20
##  2 DL-2-Aminobutyr~       2  602.        102.            702355.              20
##  3 DL-beta-Aminobu~       2  602.        102.            702355.              20
##  4 L-2-Aminobutyri~       2  602.        102.            702355.              20
##  5 N,N-Dimethylgly~       2  602.        102.            702355.              20
##  6 DL-2-Aminobutyr~       2  602.        102.            702355.              20
##  7 DL-beta-Aminobu~       2  602.        102.            702355.              20
##  8 L-2-Aminobutyri~       2  602.        102.            702355.              20
##  9 N,N-Dimethylgly~       2  602.        102.            702355.              20
## 10 DL-2-Aminobutyr~       2  602.        102.            702355.              20
## 11 DL-beta-Aminobu~       2  602.        102.            702355.              20
## 12 L-2-Aminobutyri~       2  602.        102.            702355.              20
## # i 18 more variables: peak_id <chr>, target_precursorMz <dbl>,
## #   target_spectrum_id <chr>, target_spectrum_name <chr>, target_adduct <chr>,
## #   target_ionization <chr>, target_collision_energy_text <chr>,
## #   target_instrument_type <chr>, target_formula <chr>, target_exactmass <dbl>,
## #   target_smiles <chr>, target_inchi <chr>, target_inchikey <chr>,
## #   target_pubchem <chr>, score <dbl>, RTscore <dbl>, RTpred <dbl>,
## #   ms1_accuracy <dbl>
```

Here, we can see that the returned value "res5" contains a result table which includes the annotation information.

Visualize Annotation —— VizNtaAnno is used to visualize annotation information.

The details for each parameter are listed below:

PID

chr. Program ID. It must be the same with the PID generated by *InitEMS*.

OutPath

chr. Output file directory, e.g. "D:/test". It should be noted that the slash symbol is "/", not "\". If "default", the current working directory will be set.

VarsY

chr. Outcome variable used for modeling. Only one variable can be entered.

VarsN

chr. Choose the single factor or multiple factor model. Available options include "single.factor" and "multiple.factor".

AccuracyMS1

num. Accuracy threshold to match the target compounds of MS1 information. The unit is ppm. It is usually set ranging 1-20.

MS2

lgl. T (or TRUE) and F (or FALSE). Whether to use MS2 information to annotate the target ion.

AccuracyMS2

num. Accuracy threshold to match the target compounds of MS2 information. The unit is ppm. It is usually set ranging 1-20.

DiffRT

num. Difference of the retention time between the measured and the theoretical prediction. The unit is second (s). It is usually set ranging 60-120.

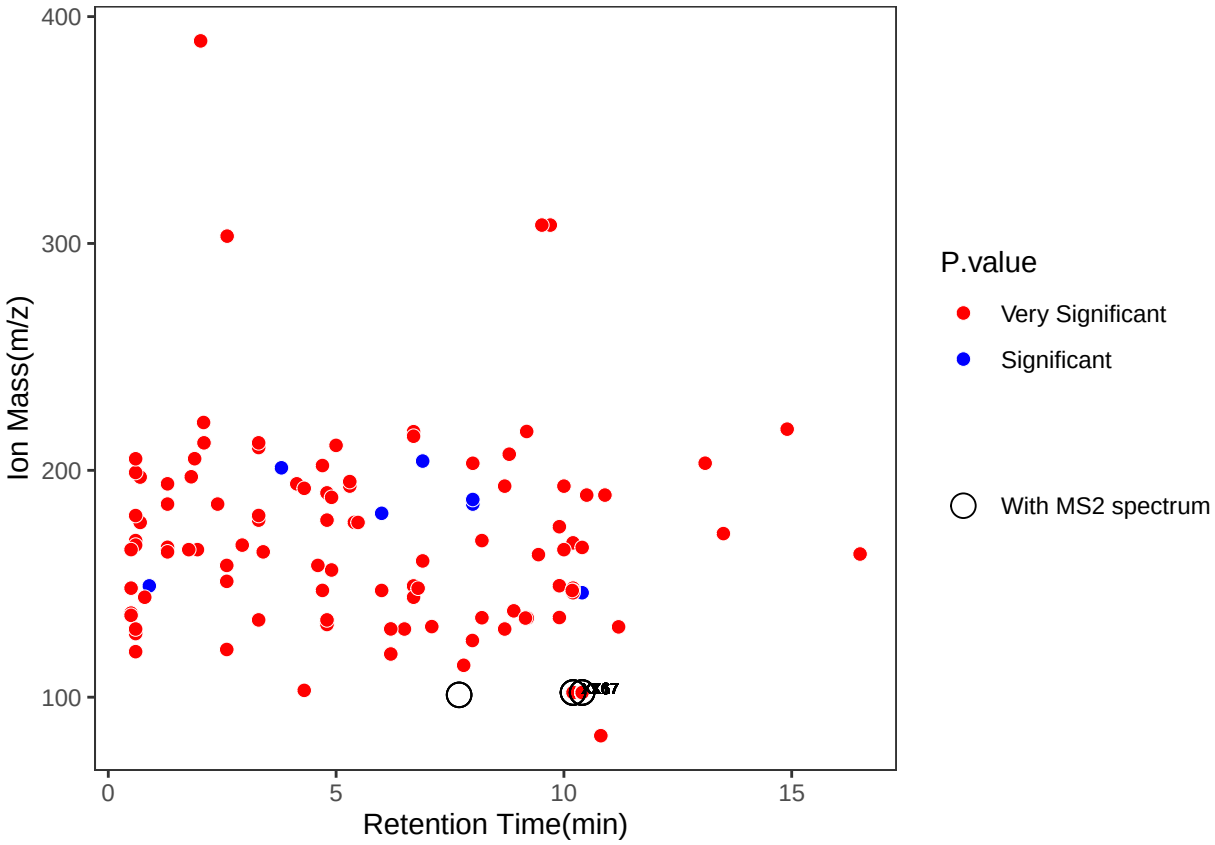
Brightness

chr. Visualization brightness. Available options include “light” and “dark”.

Palette

chr. Visualization palette. Available options include “default1”, “default2” and several journal preference styles including “cell”, “nature”, “science”, “lancet”, “nejm”, and “jama”.

```
res6 = VizNtaAnno(PID = res$PID,
                  OutPath = "default",
                  VarsY = "Y1",
                  VarsN = "single.factor",
                  AccuracyMS1 = 20,
                  Brightness = "light",
                  Palette = "default1",
                  MS2 = T,
                  AccuracyMS2 = 20,
                  DiffRT = 100)
res6$Y1_single.factor_20_20_100_light_default1
```




```
FuncExit(PID = res$PID)
```

```
## [1] "Success to exit. Thanks for using ExposomeX platform!"
```

8 Appendix

8.1 Acknowledgement

8.2 FAQs

A1: The only thing you need to do is to prepare a dataset file together with a vocabulary file that contains additional information about your data. See “data format” in the “Upload page” for detailed requirements. The template can also be downloaded in the “Download” tab.

Q2: Should I need to confirm the correctness of information provided by this module? A2: Yes. We must admit the we cannot make sure the 100% correct about the data provided using this module. For example, a chemical may have several Cas RN due to their different forms or provided by different vendor. Due to the huge data and limited background knowledge of our development team, we mainly adopt the data from the relatively authoritative institutes. However, we are surely to try our best efforts to provide the data with high-quality after careful check before distribution. Users can find the sources from the results.

Q1: What should I prepare to start ESTAT? A1: The only thing you need to do is to prepare a dataset file together with a vocabulary file that contains additional information about your data. See “data format” in the “Upload page” for detailed requirements.The template can also be downloaded in the “Download” tab.

Q2:Some modules can’t provide visualization output plot, why? A2:It is probably due to the reason that the number of the input variables are over the upper limit. For example, too many variables can make the labels in the heatmap of the correlation matrix undistinguished.

Q1: Why I cannot draw the plots successfully when I use categorical variables as outcome variable? A1: The exviz package supports drawing plot regardless of the type of outcome variable. Errors might occur when you use categorical outcome variable and indicate the “Family” parameter as “gaussian” meanwhile, which only functions in situations that the outcome variable is the numeric ones.

Q2: Can I skip some steps of the “Tidy data” part? A2: Yes. But you need to make sure that the data can satisfy the modeling requirement.

Q1: What should I prepare to start StatCros? A1: The only thing you need to do is to prepare a dataset file together with a vocabulary file that contains additional information about your data. See “data format” in the “Upload page” for detailed requirements. The template can also be downloaded in the “Download” tab.

Q2: Should I choose the family of the link function by myself? A2: Yes. The link function for the regression model are determined by data type of an outcome, including “Gaussian” for continuous variable, “Binomial” for binary variable, and “Poisson” for counting variable.

Q1: What should I prepare to start StatMix? A1: The only thing you need to do is to prepare a dataset file together with a vocabulary file that contains additional information about your data. See “data format” in the “Upload page” for detailed requirements. The template can also be downloaded in the “Download” tab.

Q2: Should I choose the family of the link function by myself? A2: Yes. The link function for the regression model are determined by data type of an outcome, including “Gaussian” for continuous variable, “Binomial” for binary variable, and “Poisson” for counting variable.

Q1: What should I prepare to start EMS? A1: The only thing you need to do is to prepare a dataset file together with a vocabulary file that contains additional information about your data. See “data format” in the “Upload page” for detailed requirements. The template can also be downloaded in the “Download” tab.

Q2: Can I skip some steps of the “Tidy” part? A2: Yes. But you need to make sure that the data can satisfy the modeling requirement.

Q1: What should I prepare to start StatPanel? A1: The only thing you need to do is to prepare a dataset file together with a vocabulary file that contains additional information about your data. See “data format” in the “Upload page” for detailed requirements.The template can also be downloaded in the “Download” tab.

Q2: Can I skip some steps of the “Tidy” part? A2: Yes. But you need to make sure that the data can satisfy the modeling requirement.

Q1: What should I prepare to start StatSurv?
A1: The only thing you need to do is to prepare a dataset file together with a vocabulary file that contains additional information about your data. See "data format" in the "Upload page" (http://www.exposomex.cn/#/exposurvival) for detailed requirements. The template can also be downloaded in the "Download" (http://www.exposomex.cn/#/home) tab.

Q2: Can I skip some steps of the "Tidy" part?
A2: Yes. But you need to make sure that the data can satisfy the modeling requirement.

1: What should I prepare to start StatIP? A1: The only thing you need to do is to prepare a dataset file together with a vocabulary file that contains additional information about your data. See "data format" in the "Upload page" for detailed requirements. The template can also be downloaded in the "Download" tab.

Q2: Can I skip some steps of the "Tidy" part? A2: Yes. But you need to make sure that the data can satisfy the modeling requirement.

Q1: What should I prepare to start EBIO? A1: The only thing you need to do is to prepare a dataset file together with a vocabulary file that contains additional information about your data. See "data format" in the "Upload page" for detailed requirements. The template can also be downloaded in the "Download" tab.

Q2: Can I re-produce the network plot using Cytoscape software? A2: Yes. The information of the edges and nodes are provided by the model. You can use the R package to get "RCy3" format file to reproduce the plot. At present, we cannot provide the file format of Cytoscape using the web service. But if you use the R package of ExposomeX, the Cytoscape file can be generated directly by following the instructions.

1. What is DB_Meta ?

- DB_Meta is publication database in the ExpoMeta, containing 29 pieces of information of nearly 20,000 articles. The number of papers is still growing.

2. If I only know the chemical name but without the information on the cas.rn, inchikey or EXC, how can I use the MetaReview or MetaEffect function?

- Chemical name can be transformed into cas.rn, inchikey or EXC through Database Dictionary first in ExpoDB.

3. What is VarX/VarY/VarM/YearFrom/YearEnd/PMID?

- MetaRefer provides two running modes, "Search" and "Download". "Search" for paper retrieval by keywords VarX/VarY/VarM/YearFrom/YearEnd, and "Download" for downloading main informain for specified PMID. VarX refers to the target chemical. VarY refers to the target outcome disease. VarM refers to the target mediating factor. YearFrom and YearEnd limit the time range searched. PMID refers to the papers PMID that are downloaded information when using "Download" mode.

4. What is Chemical_ID/Disease_ID in MetaReview or MetaEffect function?

- Chemical_ID refers to the target chemical ID which can be inchikey (eg. JIAARYAFYJHUJI-UHFFFAOYSA-L), cas.rn (eg. 7784-42-1) or our EXC ID (eg. EX:C01631). Disease_ID refers to the target disease ID which can be MESH ID (format like MESH:D006973), OMIM ID (format like OMIM:182940) or our EXD ID (eg. EX:D16243).

5. What is the difference between "Search" and "Download" modes?

- "Search" for paper retrieval by keywords VarX/VarY/VarM/YearFrom/YearEnd, and "Download" for downloading main information for specified PMID.

8.3 Reference