Characterization of missing values in untargeted MS-based metabolomics data and evaluation of missing data handling strategies

Example code for data simulation and imputation

K.T. Do, S. Wahl*, J. Raffler, S. Molnos, M. Laimighofer, J. Adamski, K. Suhre, K. Strauch, A. Peters, C. Gieger, C. Langenberg, I.D. Stewart, F.J. Theis, H. Grallert, G. Kastenmueller, J. Krumsiek*

2018-08-21

Simulation of incomplete data

Load required packages and functions

```
require(mvtnorm)
require(parallel)
require(corpcor)

source("gen.data.met.r")
```

Parameter settings

```
n <- 250
naux <- 5
p <- 2+2*naux
corr <- 0.2
pcor12 <- 0.09977889 # corresponding to Pearson correlation corr = 0.2
pcoraux <- 0.3165772
```

- n is the number of observations.
- naux is the number of auxiliary metabolites for each metabolite of interest.
- p is the number of variables to be created; in this case, 2 metabolites of interest and 5 auxiliary metabolites for each of them are created.
- corr is the Pearson correlation to achieve between the 2 metabolites of interest.
- pcor12 is the partial correlation between the 2 metabolites of interest such that the set Pearson correlation ($corr = 0.2$) is achieved in the overall correlation matrix.
- pcoraux is the partial correlation of each metabolite of interest with its 5 auxiliary metabolites. In this case, the value was determined numerically based on our real dataset (KORA F4) such that the 5 auxiliary metabolites achieve an adjusted R^2 of 50% for the metabolite of interest on average to resemble real data example.

Generate correlation matrix based on specified parameters

```
Pcor=rbind(c(0,pcor12,rep(pcoraux,naux),rep(0,naux)),
           c(pcor12,0,rep(0,naux),rep(pcoraux,naux)),
          matrix(rep(c(pcoraux,0,rep(0,naux),rep(0,naux)),each=naux),nrow=naux),
         matrix(rep(c(0,pcoraux,rep(0,naux),rep(0,naux)),each=naux),nrow=naux))
diag(Pcor) <- 1

Cor <- pcor2cor(Pcor)
```

- `Pcor` is the generated partial correlation matrix.
- `Cor` is the Pearson correlation matrix generated from the partial correlation matrix using function `pcor2cor` from package `corpcor`.

Generate artificial, complete dataset

Based on the parameters defined above, the function `gen.complete.data` generates a list `datCom`, which stores the artificial data in `datCom$x` and runday information in `datCom$rundays`. `datCom$x` is a data.frame with 250 observations and 13 variables. The first and second column correspond to the two main metabolites of interest, columns 3-12 correspond to the auxiliary metabolites, and the last column stores the phenotype (see Supporting Information S3).

```
datCom <- gen.complete.data(n=n,
                            p=p,
                            Cor=Cor,
                            n.rundays=ceiling(n/34))
```

- `n` is the number of observations.
- `p` is the number of variables.
- `Cor` is correlation matrix of the variables.
- `n.rundays` is the number of rundays. In this case, it is specified such that each runday comprises measurements of 34 samples, the everage number observed for the KORA F4 dataset.

Generate artificial, incomplete data by introducing missingness

The function `gen.miss` introduces missing values into the generated artificial data matrix, thereby producing the incomplete data matrix `dat`.

```
dat <- gen.miss(datx=datCom$x,
                incoms=c(0.1,0.1,rep(0,naux*2+1)),
                mech="trend.runday",
                betas1 = rep(-10,2),
        strengths.incom.vary=rep(0.5/4^2,2),
                strengths.beta1.vary=rep(25,2),
        cors.incom=rep(0,2),
                rundays=datCom$rundays)

dat <-  data.frame(dat,rundays=datCom$rundays)
datCom <- data.frame(datCom$x,rundays=datCom$rundays)
```

- `datx` is the complete data set (output from `gen.complete.data`).
- `incoms` is the vector of the proportion (percentage) of incomplete entries to be introduced into the variables; in this example, 10% of the values of the two main metabolites will be set to missing, while all remaining variables will remain complete.
- `mech` is the missingness mechanism to be introduced into the data (see Supporting Information S3); in this case, runday-specific probabilistic LOD is used.
- `betas1` is a vector which specifies the strength of dependency of missingness from the concentration values; in this case, we specify a strong, negative dependency guided by observations from the real dataset (see parameter β_1 in Table S8 of Supporting Information S3).

- strengths.incom.vary is a vector which specifies the variation of missingness across rundays; in this case, we set $(min(incom, 1 - incom)/4)^2$ for moderate variation, which was guided by observations from the real data (see parameter σ^2_{miss} in Table S8 of Supporting Information S3).
- strengths.beta1.vary is a vector which specifies the variation of beta1 across rundays; in this case, we set it to $(\beta_1/2)^2$ for strong variation, which was guided by observations from the real data (see parameter $\sigma^2_{\beta_1}$ in Table S8 of Supporting Information S3).
- cors.incom specifies the correlation of missingness across rundays for the two main metabolites (see parameter ρ_{miss} in Table S8 of Supporting Information S3).
- rundays defines the number of rundays in which each sample was measured.

Imputation and regression- and correlation-based analysis

Load required packages and functions

```
require(mice)
require(mitools)
require(pan)
require(ppcor)
require(tmvtnorm)
source("do.imp.met.r")
```

Perform imputation and evaluations

The function do.imp performs imputation for a given list of methods. The output can either be the imputed data or the already evaluated performance results of the different imputation methods.

```
results <- do.imp(dat,
                  datCom,
                  methods=c("complete", "cca", "min","mean","nie","nie.runday", "u.tsi",
                            "mi.avg.norm", "mi.avg.pmm", "u.tsi.runday","si.norm","si.pmm",
                            "u.tsmi","u.tsmi.runday", "mi.norm","mi.pmm", "si.pan.incl.runday",
                            "mi.pan.incl.runday", "knn.variable","knn.sample.euc",
                            "knn.sample.euc.sel"),
                  #analysis = c("mse1","mse2","mse3","mse4","corxy","pcorxy","lmx","lmy","logx"),
                  K=c(1,3,5,10,20),
                  min.in.runday=17,
                  return.da1=T)
```

- dat is the incomplete data set generated by the function gen.miss; the two columns correspond to the two main metabolites, the third column contains a continuous phenotype without missing values.
- datCom (optional) is the corresponding complete data set generated by the function gen.complete.data for comparison of the results.
- methods is a vector of imputation methods to be applied (see Supporting Information S4).
  - cca: complete case analyses
  - min: minimum imputation
  - mean: mean imputation
  - nie: Richardson & Ciampi (RC)
  - nie.runday: RC on rundays (RC-R)
  - u.tsi: truncated sampling imputation (ITS)
  - u.tsi.runday: ITS on rundays (ITS-R)

  - si.norm: multivariate single imputation by chained eqations and Bayesian regression as primary model (ICE-norm)
  - si.pmm: ICE and predictive mean matching as primiary model (ICE-pmm)
  - u.tsmi: multiple truncated sampling imputation (MITS)
  - u.tsmi.runday: MITS on rundays (MITS-R)
  - mi.norm: multiple ICE-norm (MICE-norm)
  - mi.pmm: multiple ICE-pmm (MICE-pmm)
  - si.pan.incl.runday: ICE with random intercept per runday (ICE-adjR)
  - mi.pan.incl.runday: MICE with random intercept per runday (MICE-adjR)
  - knn.variable: k-nearest neighbor imputation on variables (KNN-var)
  - knn.sample.euc: k-nearest neighbor imputation on samples based on Euclidian distance (KNN-obs)
  - knn.sample.euc.sel: KNN-obs with variable pre-selection
- `analysis` (optional) is a vector of pervormance evalation methods to be performed.
  - mse1/2/3/4: mean squared error
  - corxy: correlation between the two main metabolites
  - pcorxy: partial correlation between the two main metabolites
  - lmx: linear regression with first metabolite as predictor and second metabolite as response
  - lmy: linear regression with first metabolite as response and second metabolite as predictor
  - logx: logistic regression with phenotype as response and first metabolite as predictor
- `K` is a vector of neighbors to use for k-nearest neighbor imputation
- `min.in.runday` specifies the minimum number of observed values in each runday used as a condition to perform ITS imputation; in this example, we set it to half of the average number of observations per runday observed in the real data.
- `return.da1` is a logical parameter indicating that the imputed data should be returned (instead of the evaluation results).