Visualisation and Imputation of Missing Values

Alexander Kowarik (Statistics Austria), Matthias Templ (ZHAW Winterthur)

July 2017

Outline / R Package

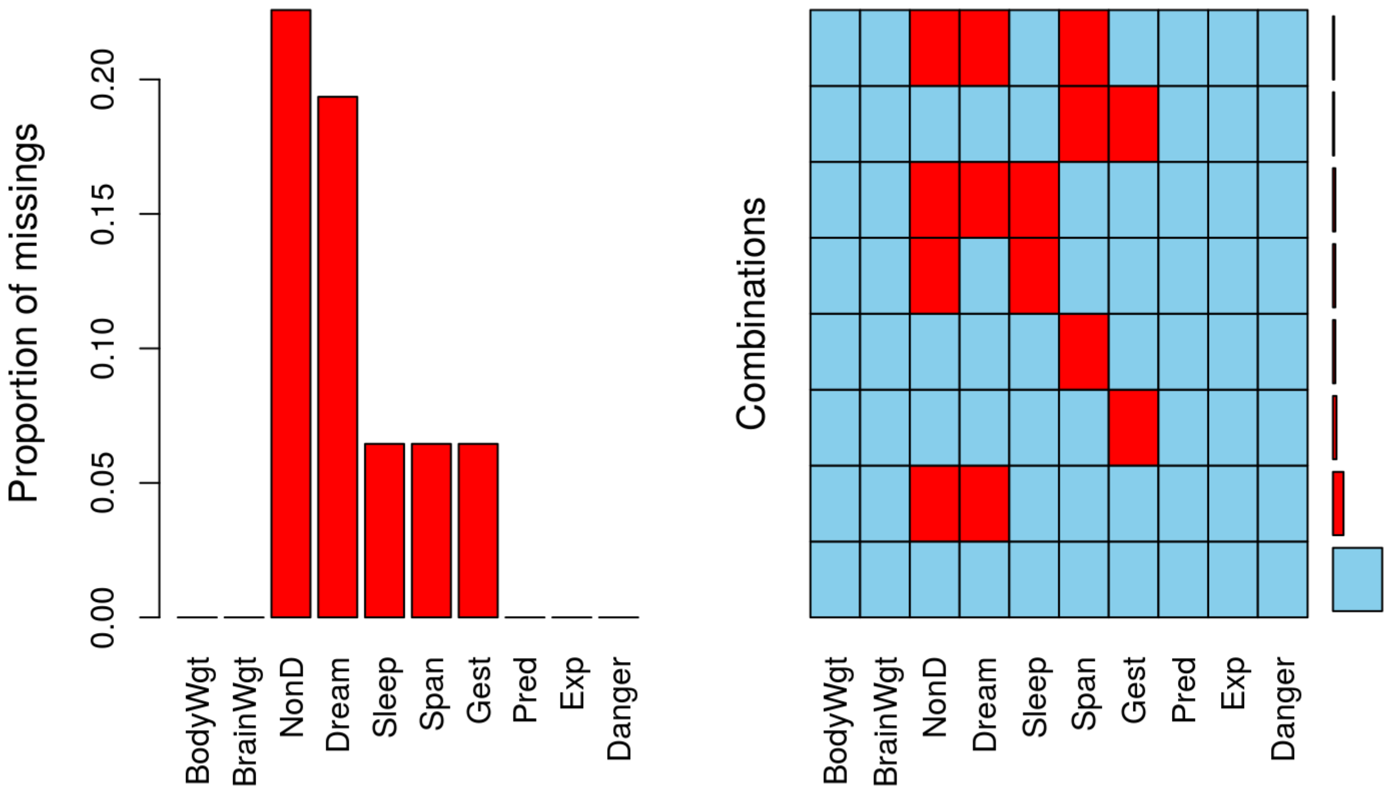
- Content:
  - Tools for visualization of missing data structures (and imputed values)
  - Tools for imputation
- Current CRAN version 4.7.0
- Development version and issue tracking on github https://github.com/statistikat/VIM
- This presentation and the R code https://github.com/alexkowa/VIM_ISI2017
- JSS paper on imputation of missing values with VIM, Kowarik, Templ
- Advances in Data Analysis and Classification paper on visualization with VIM, Templ, Alfons, Filzmoser

Visualisation of Missing Data

- Always important: knowledge about the structure of missing values. Visualisation vs statistical tests.
- literature with focus on visualization of missing data is sparse
- only a few visualization tools missing data
- R package VIM supports the visualization (also with a GUI).

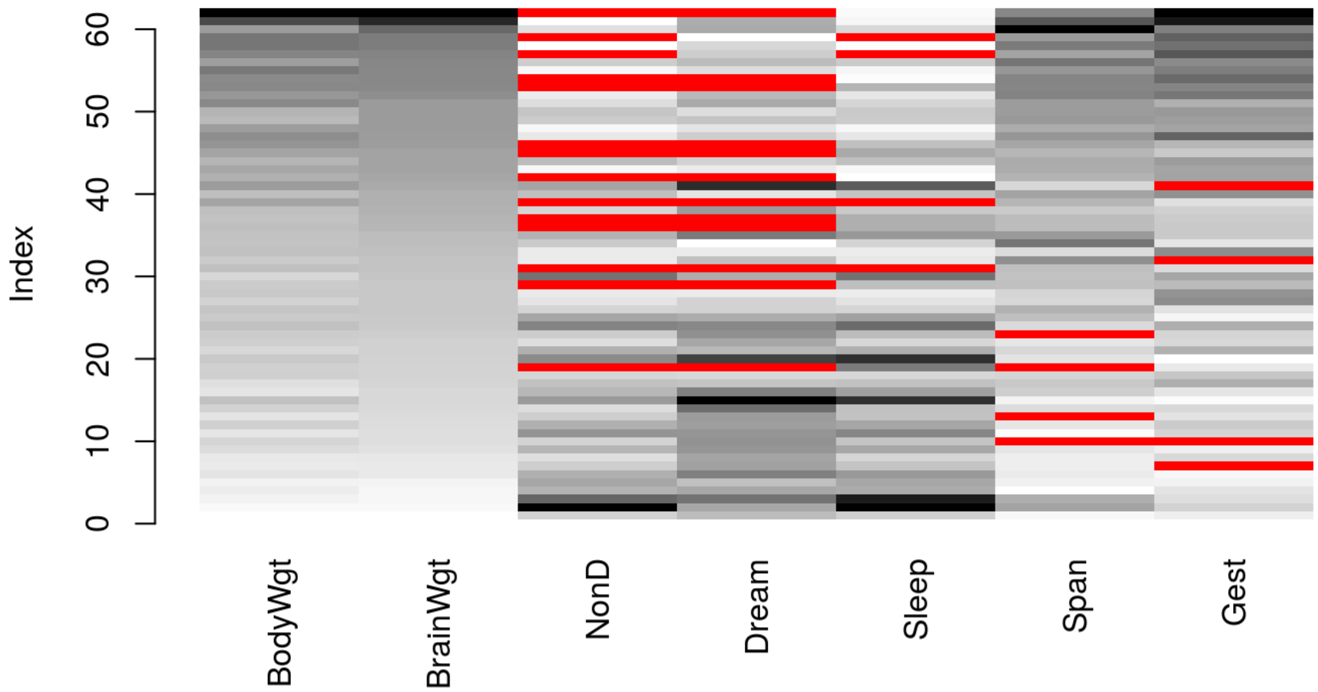
Aggregation Plots

```
aggr(sleep)
```



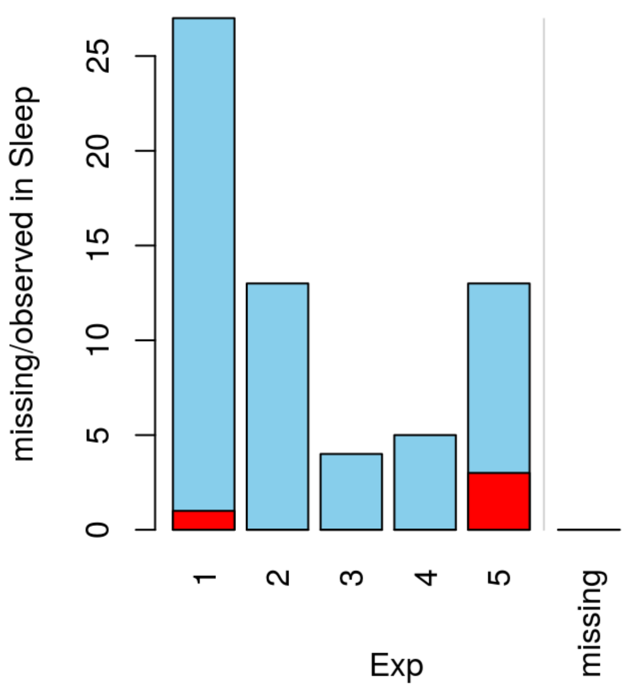
Missing Values in Matrix Form

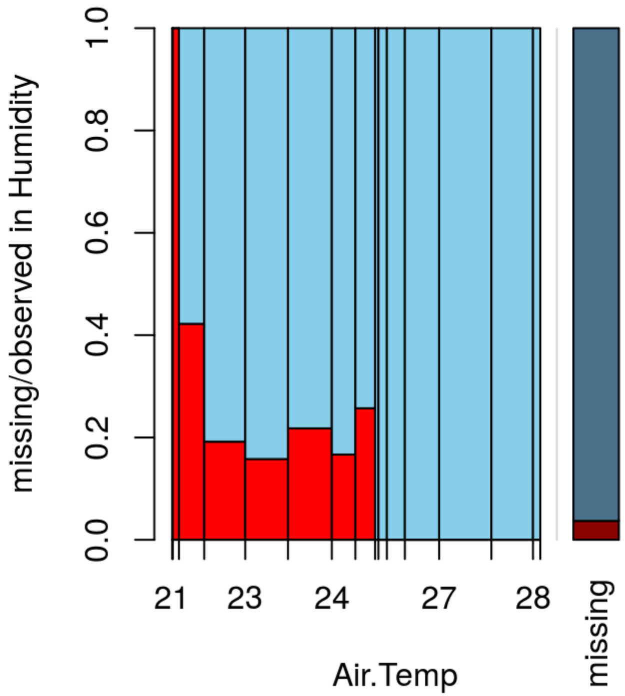
```r
matrixplot(x, sortby = "BrainWgt")
```



Univariate Plots

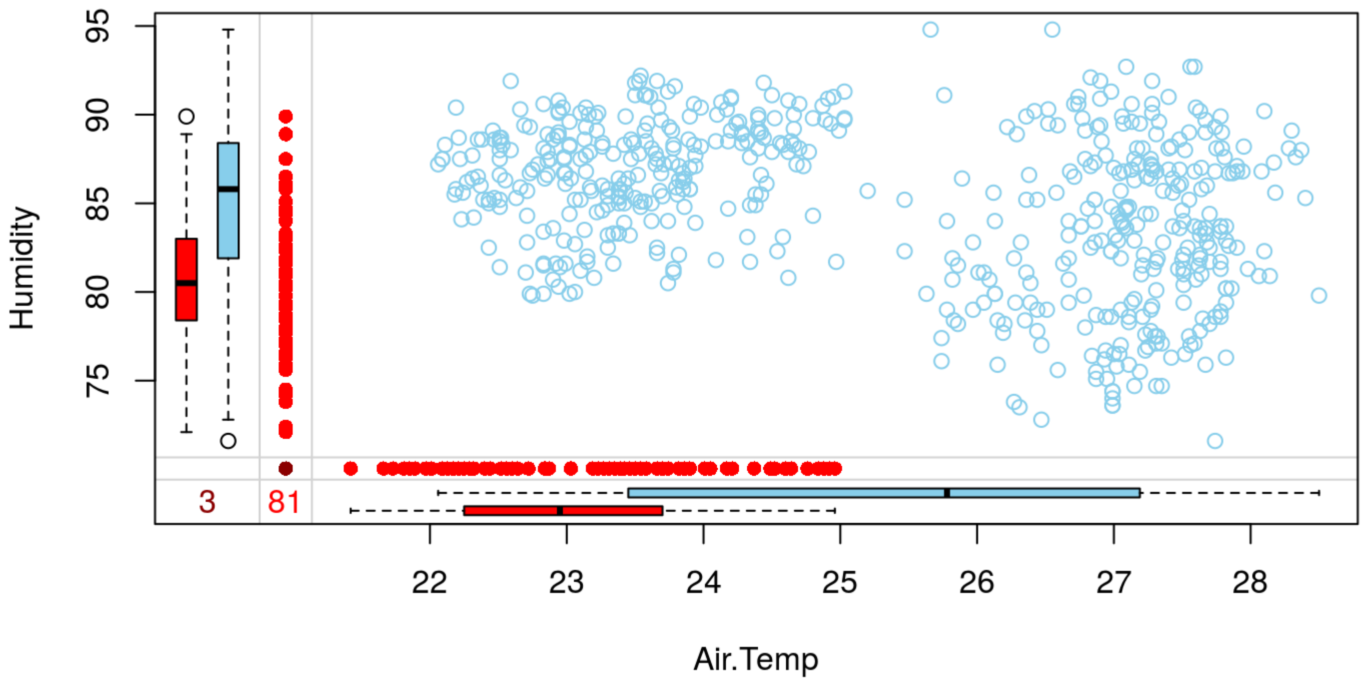
```
par(mfrow=c(1,2)); histMiss(x2); spineMiss(x3)
```





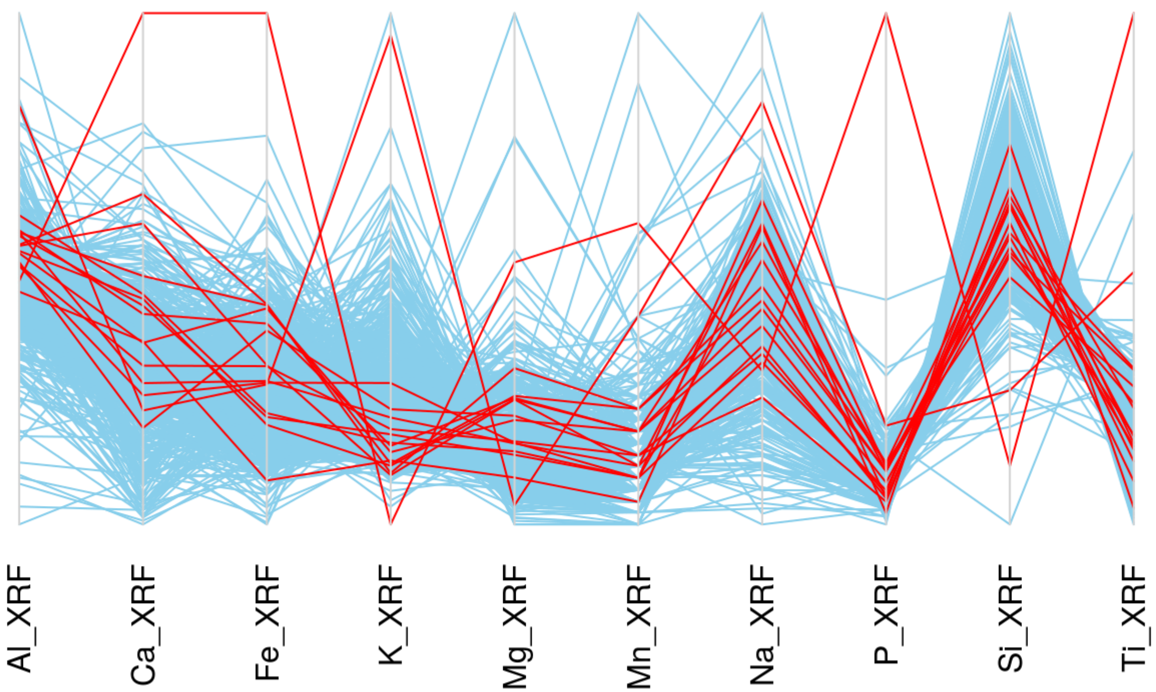
Bivariate Plots

```
marginplot(x3)
```



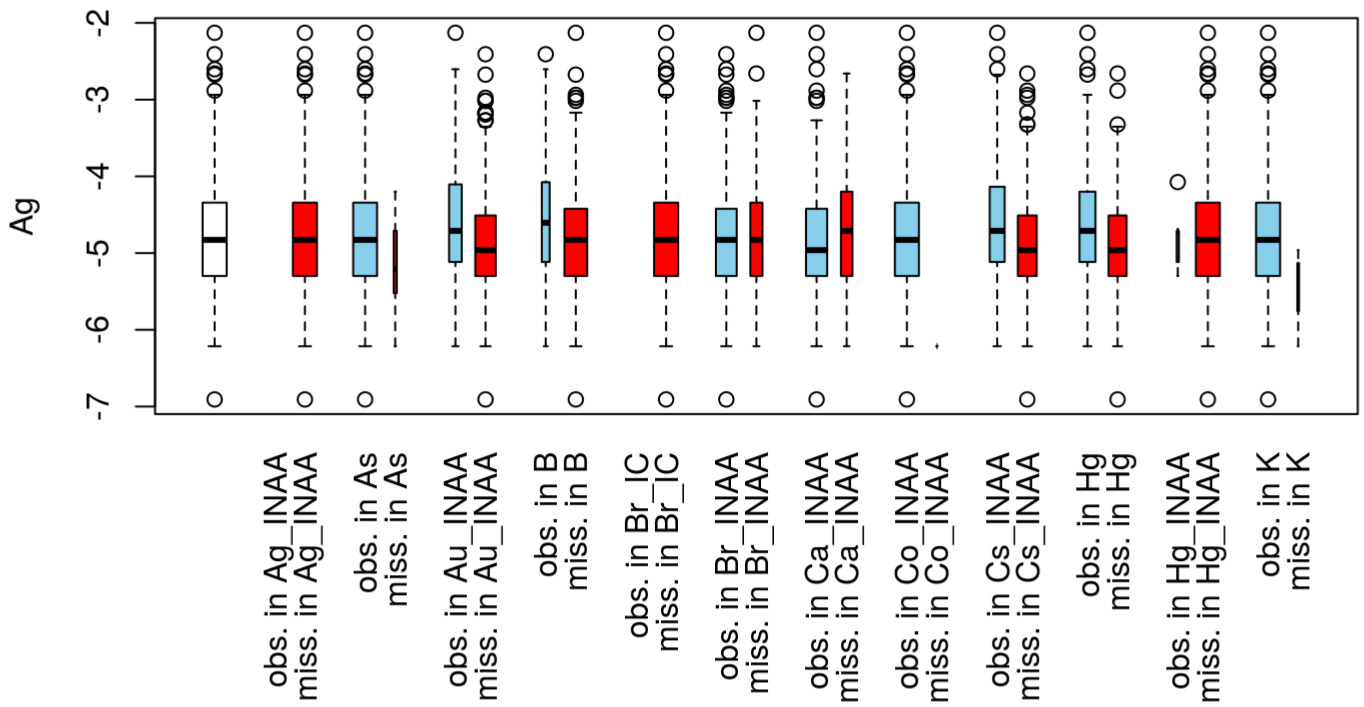
Multivariate Plots

```
parcoordMiss(x4,plotvars=2:11, interactive = FALSE)
```



Multiple Plots

pbox(x5)



Donor Imputation - hotdeck

- Random (within group)
- Sequential (within group)

```
hotdeck(data, variable = NULL, ord_var = NULL,
  domain_var = NULL, makeNA = NULL, NAcond = NULL,
  impNA = TRUE, donorcond = NULL, imp_var = TRUE,
  imp_suffix = "imp")
```

- *data* - data.frame
- *variable* - variables to be imputed
- *ord_var* - variables to sort by
- *domain_var* - variables to build imputation classes
- a random sort variable is always be added

Donor Imputation - kNN

- kNN imputation based on an extended Gower distance
- different (customized/weighted) possibilities for the aggregation step
- Weighting of distance variables

```
kNN(data, variable=colnames(data), metric=NULL, k=5,
    dist_var=colnames(data),weights=NULL, numFun = median,
    catFun=maxCat,makeNA=NULL,NAcond=NULL, impNA=TRUE,
    donorcond=NULL,mixed=vector(),mixed.constant=NULL,trace=FALSE,
    imp_var=TRUE,imp_suffix="imp",addRandom=FALSE,useImputedDist=TRUE,
    weightDist=FALSE)
```

- *dist_var* - variables used for distance combination
- *weights* - weights for distance computation
- *numFun*, *catFun* - aggregation function for numerical or categorical target variables (*sampleCat*, *maxCat*).
- *addRandom* - add a random variable to the distance computation (very low weight)

Donor Imputation - matchImpute

Random within groups imputation, grouping variables are dropped sequentially in case all values are missing in a group.

```
matchImpute(data,
  variable = colnames(data)[!colnames(data) %in% match_var],
  match_var, imp_var = TRUE, imp_suffix = "imp")
```

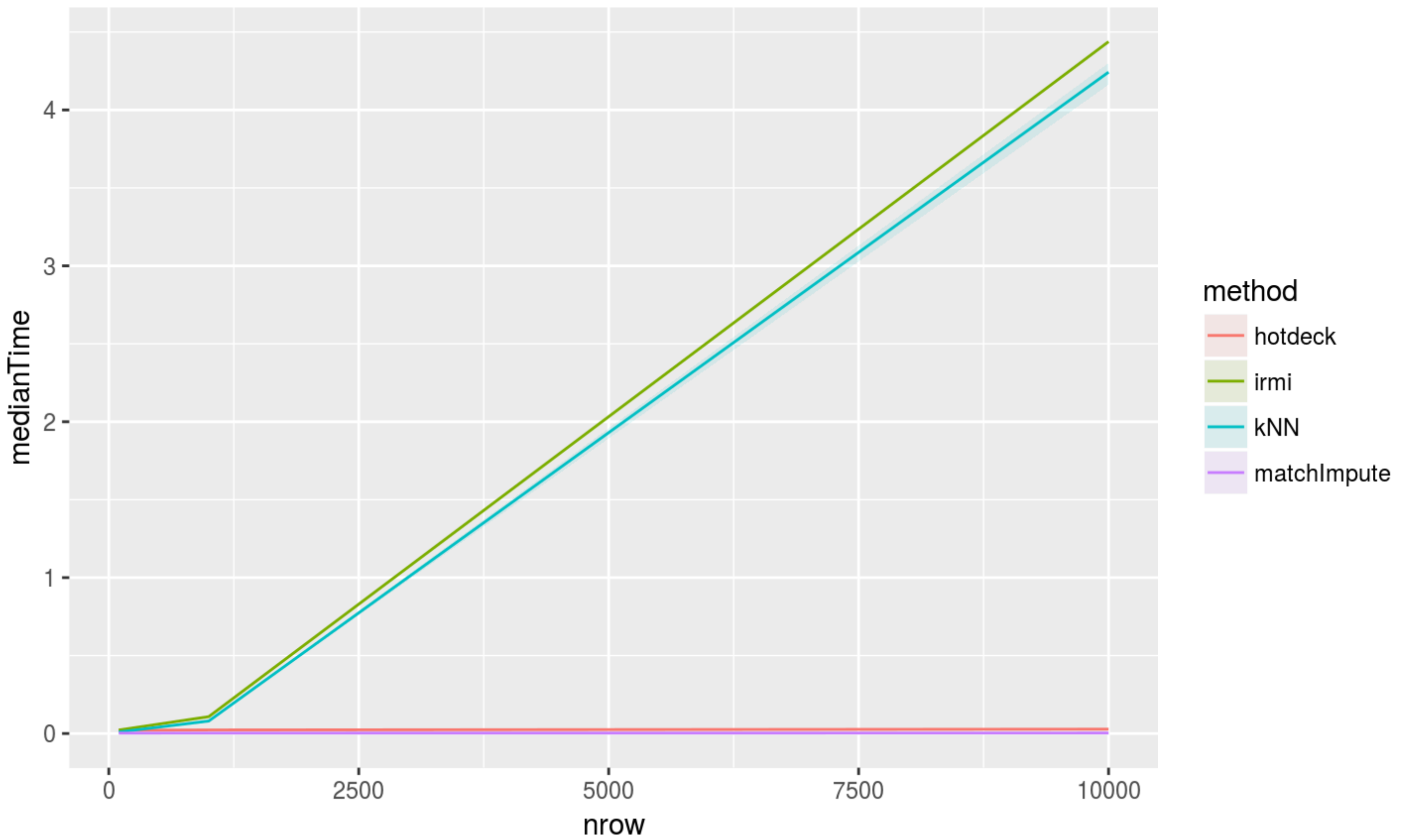
- *match_var* variables to build groups

Iterative (Robust) Regression Imputation (1)

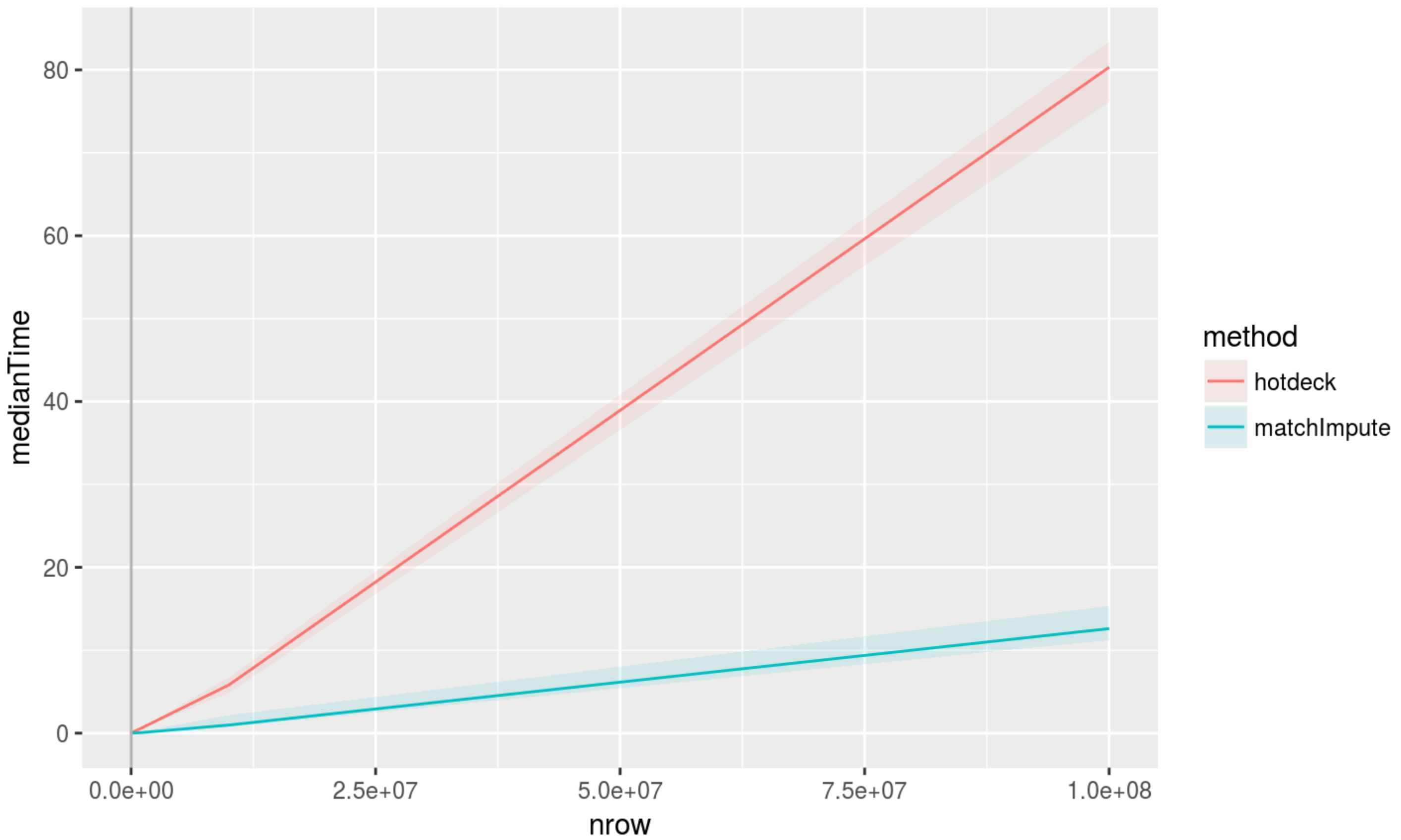
```
irmi(x, eps = 5, maxit = 100, mixed = NULL,
  mixed.constant = NULL, count = NULL, step = FALSE ,
  robust = FALSE , takeAll = TRUE, noise = TRUE,
  noise.factor = 1, force = FALSE , robMethod = "MM",
  force.mixed = TRUE, mi = 1, addMixedFactors = FALSE ,
  trace = FALSE , init.method = "kNN")
```

- *robust* - robust or non-robust
- *step* - *stepAIC* in every iteration
- *mixed* - column indices of semi-continuous variables
- *count* - column indices of count variables (Poisson)
- *noise* - add a random error to the imputed value
- *mi* - number of imputations $\Rightarrow$ multiple imputation

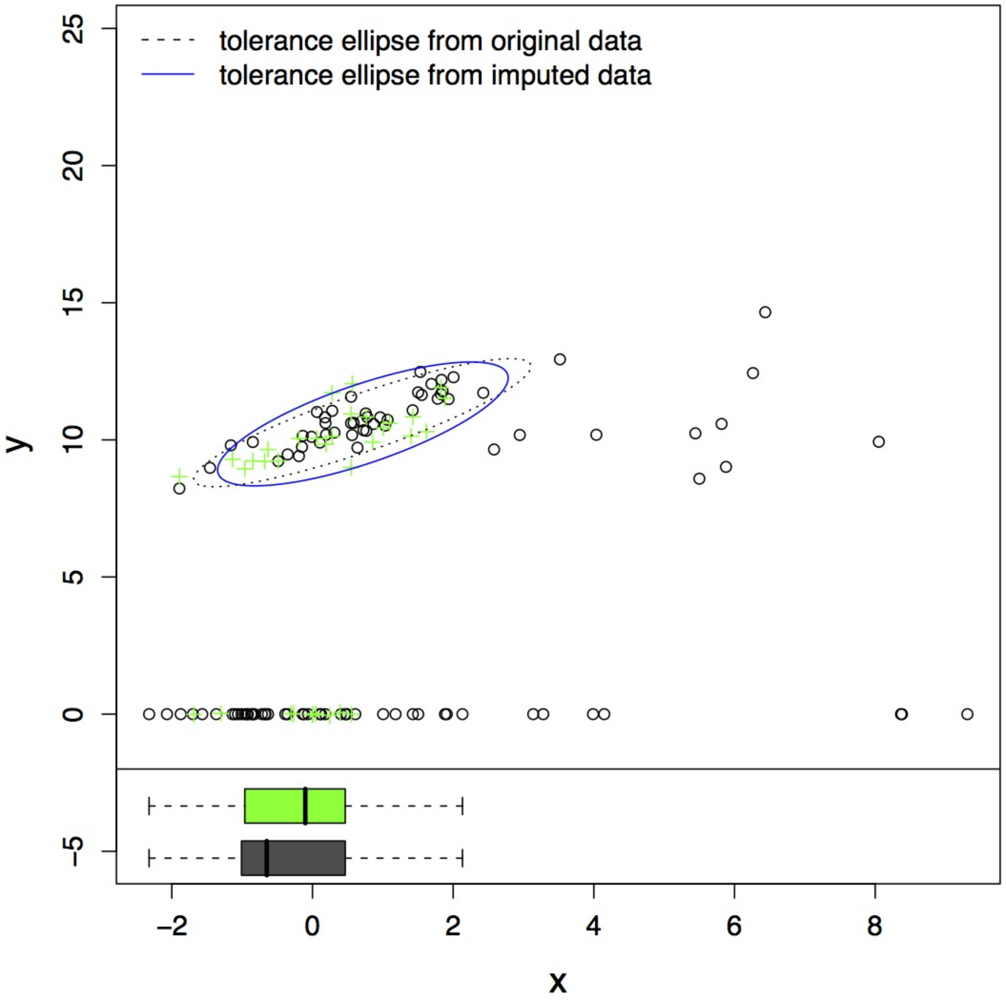
Imputation Benchmarking (1)



Imputation Benchmarking (2)



Iterative Robust Regression Imputation (2)



One more thing: simputation

- Great package by Mark van der Loo
- A lot of different imputation methods including methods kNN and hotdeck from VIM

```
sleepImp <- sleep %>% hotdeck(variable="NonD",domain_var="Danger") %>%
  kNN(variable="Dream",dist_var=c("BodyWgt","BrainWgt"))

sleepImp <- sleep %>% impute_shd( NonD~Danger,backend="VIM") %>%
  impute_knn(Dream~BodyWgt+BrainWgt, backend="VIM")
```

Thank you

Feedback always welcome:

- alexander.kowarik@statistik.gv.at
- https://github.com/statistikat/VIM
- Twitter: Alexkvienna

Simulation of public-use files from complex survey and population data

Matthias Templ (ZHAW Winterthur), Alexander Kowarik (Statistics Austria)

July 2017

Why synthetic populations?

- **comparison of methods**, e.g. in design-based simulation studies
- **policy modelling** on individual level (e.g health planning, climate change, demographic change, economic change, ...)
- **teaching** (e.g. teaching of survey methods)
- creation of public-/scientific-use files with (very) **low disclosure risk**
- data availability is often a problem (legal issues, costs,...)

Remark: We always can draw samples from a population. To generate a population is a more general approach.

Properties of close-to-reality data

- actual sizes of regions and strata need to be reflected
- marginal distributions and interactions between variables should be represented correctly
- hierarchical and cluster structures have to be preserved
- data confidentiality must be ensured
- pure replication of units from the underlying sample should be avoided
- sometimes some marginal distributions must exactly match known values
- calibration: certain marginal distributions should be exactly the same as known from other data sources

Available information

- choice of methods depends on available information:
  - census
  - survey samples
  - aggregated information from samples
  - known marginal distributions from population

Model-based approach

- In general, the procedure consists of four steps:
- setup of the household structure (with additional variables)
- simulation of categorical variables
- simulation of continuous variables
- the splitting continuous variables into components
- Stratification: allows to account for heterogenities (e.g. regional differences)

Model-based approach - the basic structure file

- **direct**: estimation of the population totals for each combination of stratum and household size using the Horvitz-Thompson estimator
- **multinom**: estimation of the conditional probabilities within the strata using a multinomial log-linear model and random draws from the resulting distributions
- **distribution**: random draws from the observed conditional distributions within the strata

Example of variables spanning the basic structure: age × region × sex
($\forall$ strata & households)

Model-based approach - fitting

$$
\textit{sample} \quad \boldsymbol{S} = \begin{array}{c}
\begin{array}{ccccccc}
\multicolumn{4}{c}{\overbrace{\hspace{10em}}^{\text{“predictors”}}} & \overbrace{\hspace{3em}}^{\text{response}} & \multicolumn{2}{c}{\overbrace{\hspace{5em}}^{\text{rest}}}
\end{array} \\
\begin{pmatrix}
x_{1,1} & x_{1,2} & \cdots & x_{1,j} & x_{1,j+1} & x_{1,j+2} & \cdots \\
x_{2,1} & x_{2,2} & \cdots & x_{2,j} & x_{2,j+1} & x_{2,j+2} & \cdots \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
x_{n,1} & x_{n,2} & \cdots & x_{n,j} & x_{n,j+1} & x_{n,j+2} & \cdots
\end{pmatrix}
\end{array}
$$

$\longrightarrow$ design matrix to model $\boldsymbol{x}_{j+1}$ (account for interactions, etc.).

$\longrightarrow$ estimation of the $\boldsymbol{\beta}$'s

Model-based approach - prediction

$$
\textit{population} \quad \boldsymbol{U} = \begin{array}{c}
\overbrace{\hspace{14em}}^{\hat{\beta}\times \text{“pred.”} \approx}\ \overbrace{\hspace{4em}}^{\hat{x}_{j+1}} \\
\begin{pmatrix}
\hat{x}_{1,1} & \hat{x}_{1,2} & \cdots & \hat{x}_{1,j} & \hat{x}_{1,j+1} \\
\hat{x}_{2,1} & \hat{x}_{2,2} & \cdots & \hat{x}_{2,j} & \hat{x}_{1,j+1} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\hat{x}_{N,1} & \hat{x}_{n,2} & \cdots & \hat{x}_{N,j} & \hat{x}_{1,j+1}
\end{pmatrix}
\end{array}
$$

we don't took expected values but draw from predictive distributions

Model-based approach - categorical variables

Estimation of the β's

- **multinom**: estimation of the conditional probabilities using multinomial log-linear models and random draws from the resulting distributions. Can deal with structural zeros.
- **distribution**: random draws from the observed conditional distributions of their multivariate realizations
- **ctree**: for using classification trees
- **ranger**: for using random forest

simCategorical()

Model-based approach - continuous variables

Similar to the categorical case, but models differ.

- **multinom**: categorize first, then draw from the predictive distributions
- **lm**: for using (two-step) regression models combined with random error terms
- **glm's**, e.g. **poisson** for using Poisson regression for count variables
- robust methods
- **ranger**: for using random forest

simContinuous()

Model-based approach - more methods

Components:

- by resampling fractions from survey data (**simComponents()**)

Relations:

- taking relationships between household members into account (**simRelation()**)

Spatial:

- generation of smaller regions given an existing spatial variable and a table (**simSpatialInit()**)

R package simPop

- Templ, Kowarik, and Meindl (2017), Journal of Statistical Software (accepted)
- latest version on CRAN
- development on github
- parallel computing is applied automatically
- efficient implementation

Define the structure

Create an object of class *dataObj* with function **specifyInput()**.

```
inp <- specifyInput(data=origData,
                    hhid="db030",
                    hhsize="hsize",
                    strata="db040",
                    weight="rb050")
```

```
class(inp)
```

```
## [1] "dataObj"
## attr(,"package")
## [1] "simPop"
```

Simulating the basic structural variables

```
synthP <- simStructure(data=inp,
                       method="direct",
                       basicHHvars=c("age", "rb090", "db040"))
class(synthP)
```

```
## [1] "simPopObj"
## attr(,"package")
## [1] "simPop"
```

- output object (*"synthP"*) is of class *simPopObj*
- various functions can be applied to such objects

Simulation of categorical variables

```
synthP <- simCategorical(synthP, additional=c("pl030", "pb220a"),
  method="multinom")
synthP
```

```
##
## --
## synthetic population  of size
##   8182010 x 9
##
## build from a sample of size
## 11725 x 19
## --
##
## variables in the population:
## db030,hsize,age,rb090,db040,pid,weight,pl030,pb220a
```

almost the same for *simContinuous()*

Census information to calibrate

- We add these marginals to the object and calibrate afterwards

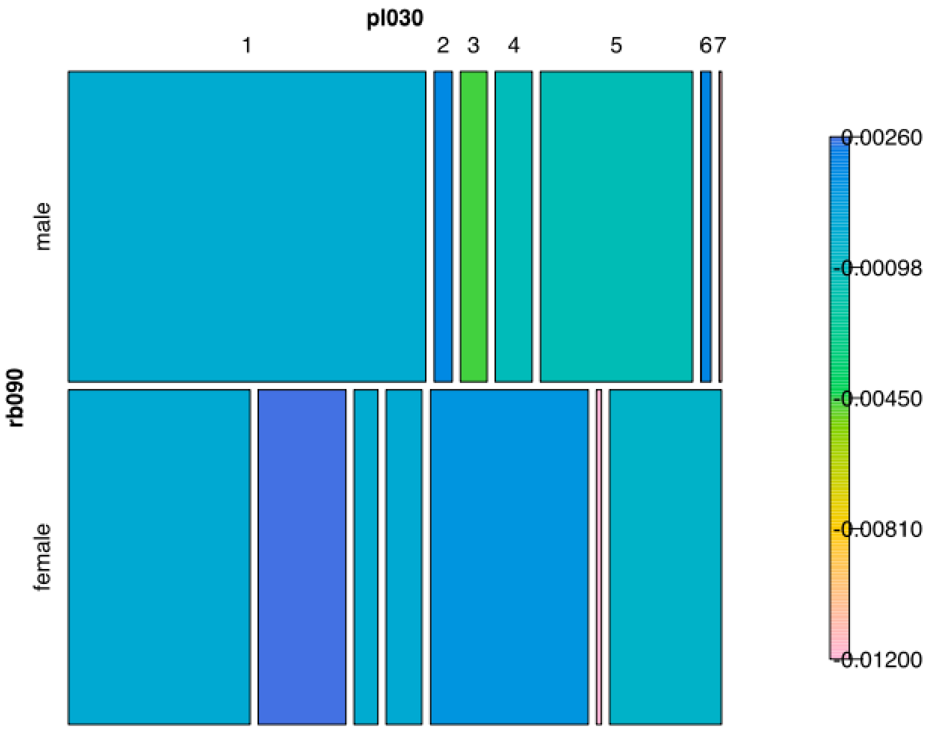
```
synthP <- addKnownMargins(synthP, margins) # add margins
```

```
# calibration using simulated annealing
synthPadj <- calibPop(synthP, split="db040", temp=1,
                      eps.factor=0.00005, maxiter=200,
                      temp.cooldown=0.975,
                      factor.cooldown=0.85,
                      min.temp=0.001, verbose=TRUE)
```

Now: margins of the sample **equals known margins of the population** (not shown here, long computation time.)

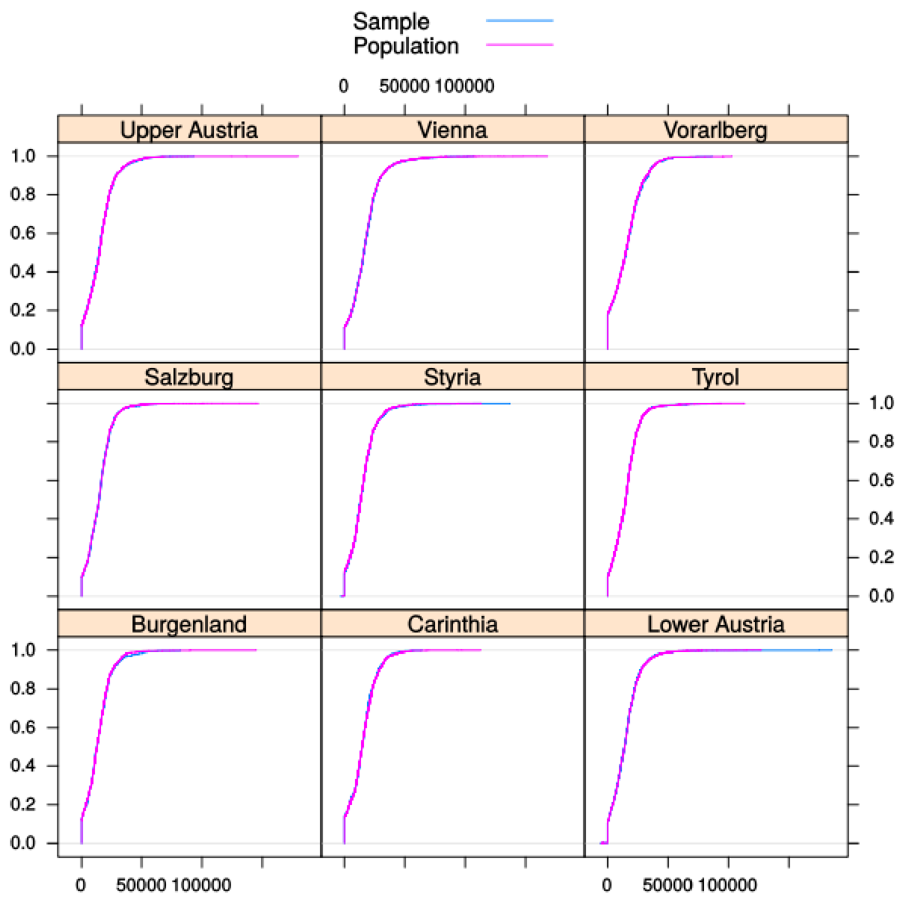
Results

```
tab <- spTable(synthP, select = c("rb090", "pl030"))
spMosaic(tab, method = "color")
```



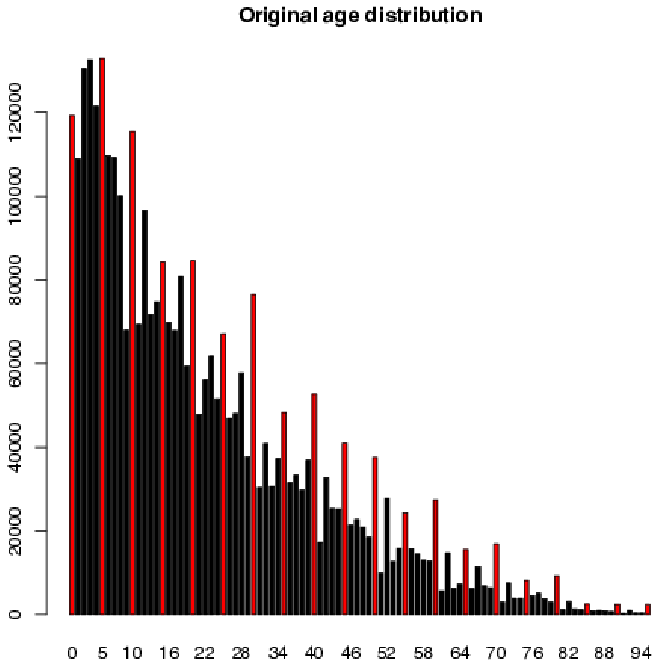
Results

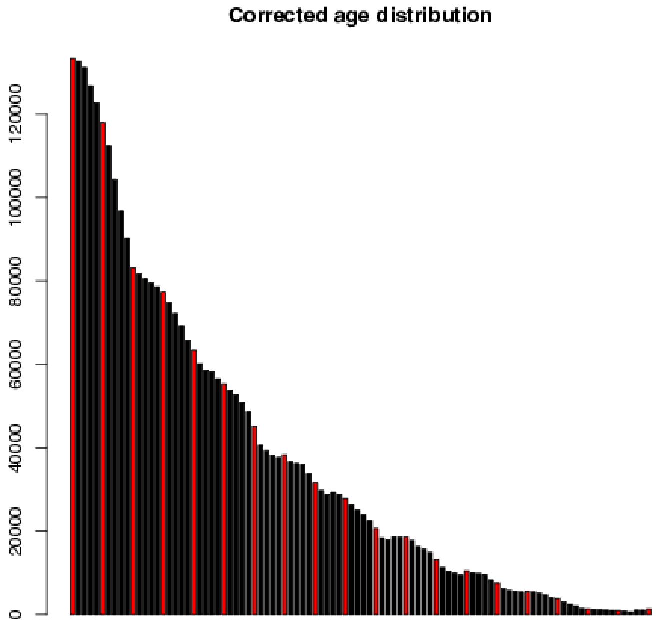
```
spCdfplot(synthPadj, "netIncome", cond="db040", layout=c(3, 3))
```



Other feature of simPop - age heaping

Correct for age heaping using truncated (log-)normal distributions on individual level (function **correctHeap()**)





Conclusions

- Structure of original input data is preserved
- Margins of synthetic populations are calibrated
- The synthetic populations are confidential
- Code of **simPop** is quite efficient
- Many methods are ready to be used