

Doner based imputation methods

with applications in R

Aksel Thomsen (akt@dst.dk)

Statistics Denmark

2020/12/7

Outline

1. Theory

- Imputation in general (recap)
- Donor-based imputation methods

2. Practical

- Apply methods
- How to in R

Theory

General theory

- Item or **partiel** non-response

General theory

- Item or **partiel** non-response
- Donor vs model based
- Stochastic or deterministic
- Hot- or cold-deck

General theory

- Item or **partiel** non-response
- Donor vs model based
- Stochastic or deterministic
- Hot- or cold-deck
- Deductive (logical) imputation

Donor imputation

Two general approaches:

1. Nearest neighbor
 - KNN
 - Distance in multidimensional space
 - Predictive mean matching
2. Random draws (stratified)

Donor imputation

When is a donor good enough?

And can different donors be used for the same observations?

Donor imputation

When is a donor good enough?

And can different donors be used for the same observations?

3 cases:

1. Complete:

- All imputed variables complete for doner
- Same donor for all variables

2. Univariate

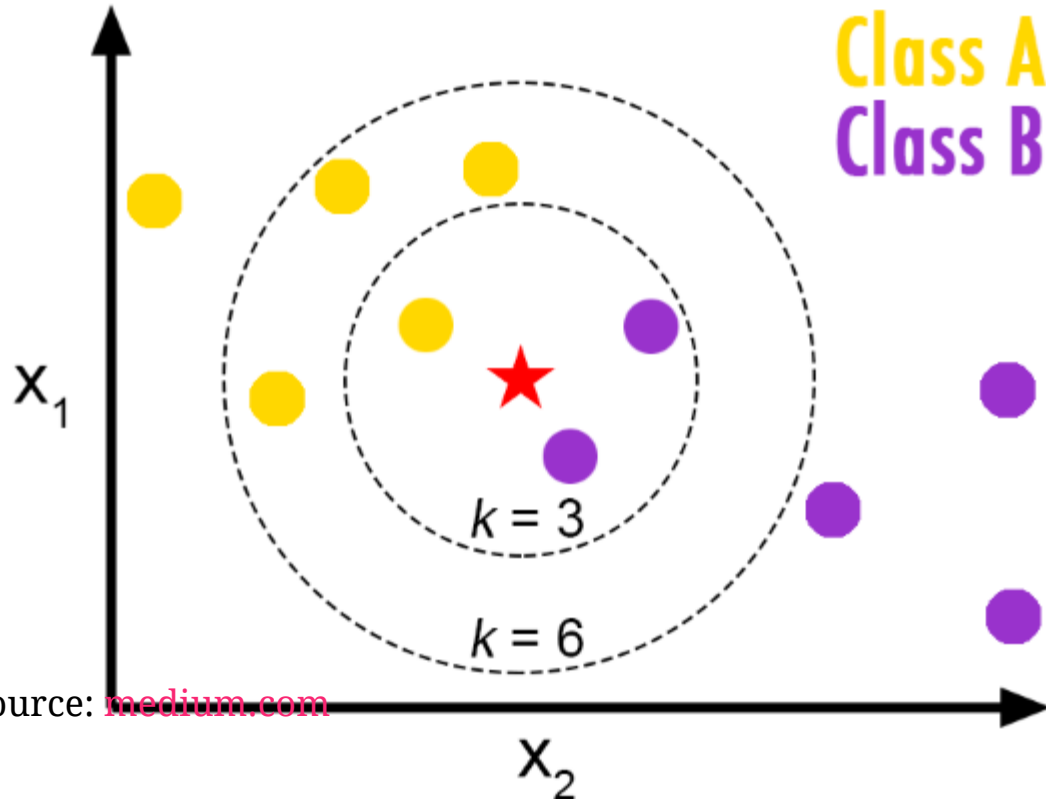
- Variables are imputed one by one
- Seperate donors for each variable

3. Multivariate

- Donor pool for each missingness value
- Same donor for all variables

KNN

- Find the K nearest neighbors
 - $K = 1$: Pure donor imputation
 - $K > 1$: "Average" of the donors

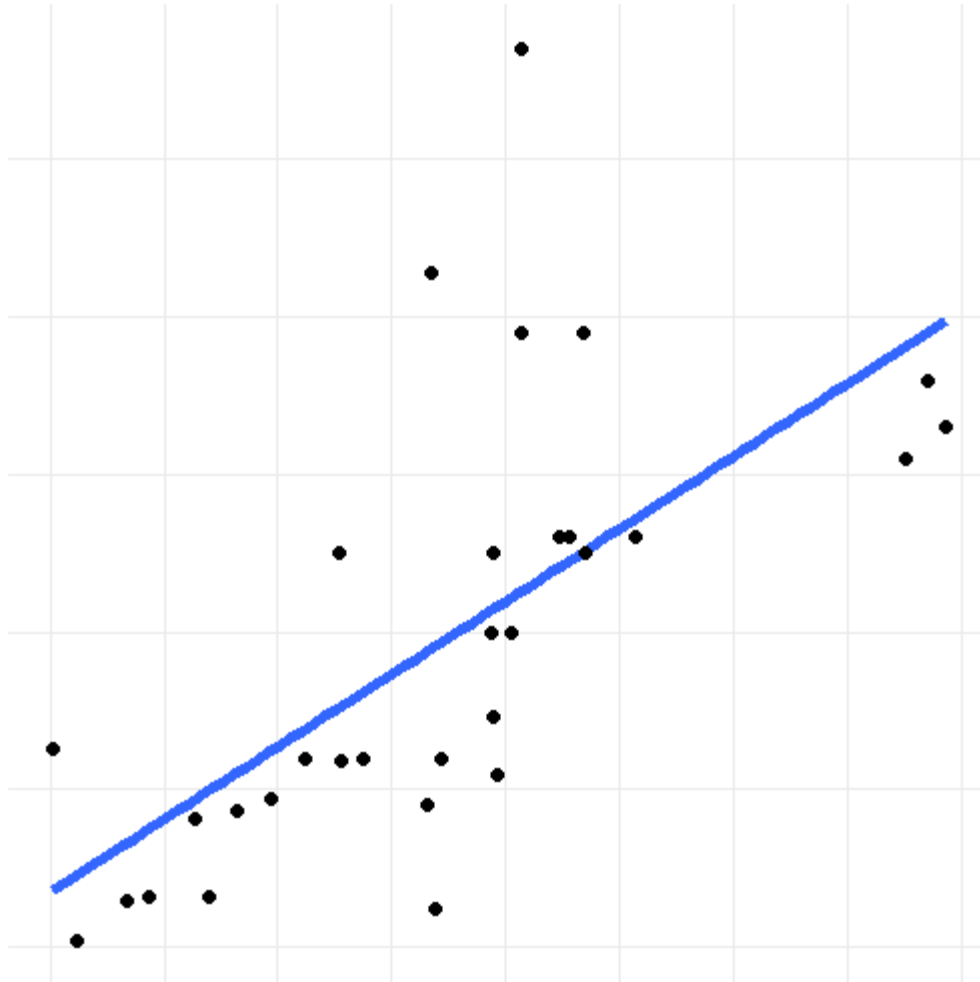


Source: medium.com

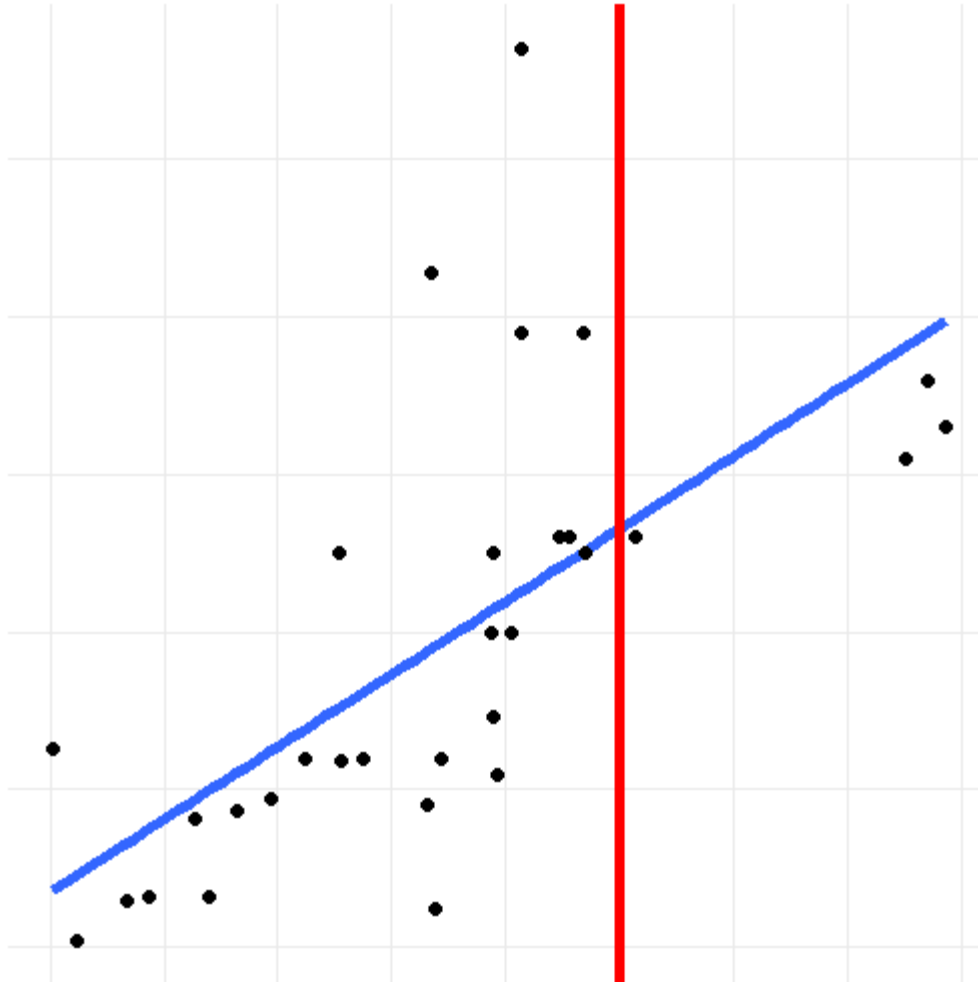
Predictive mean matching

- Mix between model and donor based imputation
- Method:
 1. Estimate a model predicting the missing variable(s)
 2. Form predictions for all observation
 3. Donor is the observation with the closest predicted value
- From here a KNN with $K = 1$
- A way to redefine a multidimensional problem into a one dimensional problem

Example: Linear prediction (1/2)



Example: Linear prediction (2/2)



Random draws

- Sequential or **random**
- **With** or without replacement or maximum donations per donor

Practical

Simulated LFS

```
library(tidyverse)
lfs <- read_csv("example.csv", col_types = "inffnn") %>%
  as.data.frame()

head(lfs)
```

id	age	gender	region	employed	hours
1	64	F	W	1	40
2	77	M	S	0	NA
3	83	F	S	NA	NA
4	24	F	W	1	40
5	65	F	N	1	40
6	42	M	E	0	NA


```
summary(lfs)
```

```
##           id           age      gender  region      employed
##  Min.      : 1.0    Min.      :18.00   F:259    W:164    Min.      :0.0000
## 1st Qu.:125.8    1st Qu.:34.00   M:241    S:132    1st Qu.:0.0000
## Median :250.5    Median :47.00           N: 81    Median :1.0000
## Mean   :250.5    Mean   :49.55           E:123    Mean   :0.6526
## 3rd Qu.:375.2    3rd Qu.:65.00           3rd Qu.:1.0000
## Max.    :500.0    Max.    :90.00           Max.    :1.0000
##                                     NA's     :51
##           hours
##  Min.      :20.00
## 1st Qu.:39.00
## Median :40.00
## Mean   :37.24
## 3rd Qu.:40.00
## Max.    :40.00
## NA's     :207
```

```
summary(lfs)
```

```
##           id           age      gender  region      employed
##  Min.      : 1.0    Min.      :18.00   F:259    W:164    Min.      :0.0000
## 1st Qu.:125.8    1st Qu.:34.00   M:241    S:132    1st Qu.:0.0000
## Median :250.5    Median :47.00           N: 81    Median :1.0000
## Mean   :250.5    Mean   :49.55           E:123    Mean   :0.6526
## 3rd Qu.:375.2    3rd Qu.:65.00           3rd Qu.:1.0000
## Max.    :500.0    Max.    :90.00           Max.    :1.0000
##                                     NA's    :51
##           hours
##  Min.      :20.00
## 1st Qu.:39.00
## Median :40.00
## Mean   :37.24
## 3rd Qu.:40.00
## Max.    :40.00
## NA's    :207
```

Partial non-response!

```
lfs %>% count(employed, is.na(hours))
```

employed	is.na(hours)	n
0	TRUE	156
1	FALSE	265
1	TRUE	28
NA	FALSE	28
NA	TRUE	23

```
lfs %>% count(employed, is.na(hours))
```

employed	is.na(hours)	n
0	TRUE	156
1	FALSE	265
1	TRUE	28
NA	FALSE	28
NA	TRUE	23

Routing: employed = 0 => hours not asked (NA is valid)

```
lfs %>% count(employed, is.na(hours))
```

employed	is.na(hours)	n
0	TRUE	156
1	FALSE	265
1	TRUE	28
NA	FALSE	28
NA	TRUE	23

Routing: employed = 0 => hours not asked (NA is valid)

Logical imputation: hours answered => the person is employed

```
lfs %>% count(employed, is.na(hours))
```

employed	is.na(hours)	n
0	TRUE	156
1	FALSE	265
1	TRUE	28
NA	FALSE	28
NA	TRUE	23

Routing: employed = 0 => hours not asked (NA is valid)

Logical imputation: hours answered => the person is employed

51 missing cells left

Imputation

- R package to make imputations easy, covers:
- **Model based** (optionally add [non-]parametric random residual)
 - linear regression
 - robust linear regression
 - ridge/elasticnet/lasso regression
 - CART models (decision trees)
 - Random forest
- **Multivariate** imputation
 - Imputation based on the expectation-maximization algorithm
 - missForest (=iterative random forest imputation)
- **Donor** imputation (including various donor pool specifications)
 - k-nearest neighbour (based on gower's distance)
 - sequential hotdeck (LOCF, NOCB)
 - random hotdeck
 - Predictive mean matching
- **Other**
 - (groupwise) median imputation (optional random residual)
 - Proxy imputation: copy another variable or use a simple transformation to compute imputed values.
 - Apply trained models for imputation purposes.

Imputation strategy

1. Deductive: If answered hours, then the person is employed.
2. Two step donor imputation:
 1. Employment: Predictive mean matching
 2. Hours: Random hot-deck donor

Step 1

```
library(simputation)

lfs_imp <- lfs %>%
  impute_proxy(formula = employed ~ hours > 0)
```

Step 1

```
library(simputation)

lfs_imp <- lfs %>%
  impute_proxy(formula = employed ~ hours > 0)
```

```
lfs_imp %>% count(employed, is.na(hours))
```

employed	is.na(hours)	n
0	TRUE	156
1	FALSE	293
1	TRUE	28
NA	TRUE	23

Step 2

```
lfs_imp <- lfs %>%  
  impute_proxy(formula = employed ~ hours > 0) %>%  
  impute_pmm(formula = employed ~ age + gender + region)
```

Step 2

```
lfs_imp <- lfs %>%  
  impute_proxy(formula = employed ~ hours > 0) %>%  
  impute_pmm(formula = employed ~ age + gender + region)
```

```
lfs_imp %>% count(employed, is.na(hours))
```

employed	is.na(hours)	n
0	TRUE	165
1	FALSE	293
1	TRUE	42

Step 3

```
lfs_imp <- lfs %>%  
  impute_proxy(formula = employed ~ hours > 0) %>%  
  impute_pmm(formula = employed ~ age + gender + region) %>%  
  impute_rhd(formula = hours ~ age + gender + region | employed)
```

Step 3

```
lfs_imp <- lfs %>%  
  impute_proxy(formula = employed ~ hours > 0) %>%  
  impute_pmm(formula = employed ~ age + gender + region) %>%  
  impute_rhd(formula = hours ~ age + gender + region | employed)
```

```
lfs_imp %>% count(employed, is.na(hours))
```

employed	is.na(hours)	n
0	TRUE	165
1	FALSE	314
1	TRUE	21

```
lfs_imp %>% filter(employed==1, age==21)
```

id	age	gender	region	employed	hours
30	21	M	W	1	NA
79	21	F	S	1	40
97	21	M	S	1	40
265	21	F	W	1	40
357	21	F	S	1	31

```
lfs_imp %>% filter(employed==1, age==21)
```

id	age	gender	region	employed	hours
30	21	M	W	1	NA
79	21	F	S	1	40
97	21	M	S	1	40
265	21	F	W	1	40
357	21	F	S	1	31

No donors in the strata for id = 21


```
lfs_imp %>% filter(employed==1, age==21)
```

id	age	gender	region	employed	hours
30	21	M	W	1	NA
79	21	F	S	1	40
97	21	M	S	1	40
265	21	F	W	1	40
357	21	F	S	1	31

No donors in the strata for id = 21

"Easy" solution => Random donor in 10 year age group

Step 4

```
lfs_imp <- lfs %>%  
  impute_proxy(formula = employed ~ hours > 0) %>%  
  impute_pmm(formula = employed ~ age + gender + region) %>%  
  impute_rhd(formula = hours ~ age + gender + region | employed) %>%  
  mutate(age10 = age %/% 10) %>%  
  impute_rhd(formula = hours ~ age10 | employed) %>%  
  select(-age10)
```

Step 4

```
lfs_imp <- lfs %>%  
  impute_proxy(formula = employed ~ hours > 0) %>%  
  impute_pmm(formula = employed ~ age + gender + region) %>%  
  impute_rhd(formula = hours ~ age + gender + region | employed) %>%  
  mutate(age10 = age %/% 10) %>%  
  impute_rhd(formula = hours ~ age10 | employed) %>%  
  select(-age10)
```

```
lfs_imp %>% count(employed, is.na(hours))
```

employed	is.na(hours)	n
0	TRUE	165
1	FALSE	335

New micro data

```
lfs_imp %>% anti_join(lfs, by = names(lfs_imp)) %>% slice_sample(n=10)
```

id	age	gender	region	employed	hours
402	60	F	N	1	40
187	49	M	W	1	40
166	48	M	E	1	40
247	32	F	S	1	40
153	32	M	W	1	39
437	41	M	S	1	40
82	85	M	N	1	40
443	43	M	W	1	40
112	35	F	W	1	40
127	38	M	N	1	40

Alternative ML solution

```
lfs_mf <- lfs %>%  
  mutate(hours = if_else(employed == 0, 0, hours),  
         employed = as.factor(as.character(employed))) %>%  
  as.data.frame() %>%  
  impute_mf(formula = . ~ id ~ . ~ id) %>%  
  mutate(employed = as.numeric(as.character(employed)),  
         hours = if_else(hours == 0, NA_real_, hours))
```

```
## missForest iteration 1 in progress...done!  
## missForest iteration 2 in progress...done!  
## missForest iteration 3 in progress...done!  
## missForest iteration 4 in progress...done!
```

```
lfs_mf %>% count(employed, is.na(hours))
```

employed	is.na(hours)	n
0	TRUE	156
1	FALSE	344

Questions?
(ressources next slide)

Ressources

Presentation: [GitHub](#)

EU / MEMOBUST: [Handbook on imputation](#)

CRAN Task View: [Official Statistics & Survey Methodology](#)

Mark van der loo: [simputation: Simple Imputation](#)

RStudio: [Tidyverse collection of R packages](#)

Awesome official statistics software: [GSBPM & R packages](#)