LATVIJAS UNIVERSITĀTE
FIZIKAS UN MATEMĀTIKAS FAKULTĀTE
MATEMĀTISKĀS ANALĪZES KATEDRA

# BALANSĒTĀS K TUVĀKO KAIMIŅU IMPUTĀCIJAS METODES UZLABOJUMU PĀRBAUDE

BAKALAURA DARBS

Autors: **Ruāna Pavasare**
Studenta apliecības nr. rp13016
Darba vadītājs: Dr. math. Nataļja Budkina
Konsultants: Aleksis Jurševskis

Rīga, 2017

# ANOTĀCIJA

Bakalaura darba pētāmā problēma ir salīdzināt dažādas imputācijas metodes izlašu novērtējumu problemātikā.

Darbs ir veltīts balansētās k tuvāko kaimiņu imputācijas metodes uzlabojumu analīzei. Uzlabojumā donoru izvēle ir aplūkota kā izlases problēma un tiek izmantota kalibrācija un balansēta izlase.

Darbā ir izstrādāts metodes algoritms un salīdzināti tās rezultāti ar citām izvēlētām imputācijas metodēm. Salīdzināšanai tiek izmantota tuvākā kaimiņa metode, k tuvāko kaimiņu metode, imputācija ar vienkārša gadījuma izlasi ar atkārtojumiem un vienkārša gadījuma izlase bez atkārtojumiem. Metodes tiek salīdzinātas vairākas reizes simulējot iztrūkšanu un atkārtoti imputējot.

Aprēķiniem tiek izmantota programma R un izmantots *iris* datu masīvs.

Atslēgas vārdi: balansēta izlase, kalibrācija, tuvākie kaimiņi, nerespondence, imputācijas

# ANOTATION

The research problem of the bachelor's thesis is to compare several imputation methods in problems of sampling estimates.

The bachelor's thesis is devoted to the analysis of balanced k nearest neighbour imputation method improvements. In improvements the selection of donors is viewed as a sampling problem and calibration and balanced sampling are used.

In the bachelor's thesis algorithm of method is worked out and its results are compared with other selected imputation methods. For comparing nearest neighbour imputation method, k nearest neighbour imputation method, imputation with simple random sampling with replacement and imputation with simple random sampling without replacement are used. Methods are compared several times simulating absence and repeating imputation.

Simulations are done with the help of program R by using iris flower data set.

Keywords: balanced sampling, calibration, nearest neighbours, nonresponse, imputations

# Saturs

# IEVADS

Izlases apsekojumos viena no lielākajām problēmām ir neatbildētība, ar ko jāsaskaras, veicot datu sagatavošanu. Parasti praksē ir reti sastopama situācija, kad visi respondenti ir pilnībā atbildējuši. Trūkstošās vērtības ietekmē gan rādītāju novērtējumus, gan arī to, ka uzreiz nav iespējams pielietot standarta metodes, lai analizētu datus, jo paredzētais datu daudzums ir samazinājies. Trūkstošās vērtības var arī rasties apstrādājot datu masīvu.

Darba mērķis ir pārbaudīt balansētās k tuvāko kaimiņu imputācijas metodes uzlabojumus. Metodes uzlabojumi balstās uz to, ka donoru izvēle tiek apskatīta kā izlases problēma un tiek izmantota kalibrācija un balansēta izlase, lai trūkstošās vērtības tiktu aizstātas ar līdzīgu vērtību jau no esošajām novērotajām vērtībām.

Uzlabotās metodes algoritms tiek izveidots programmā R. Lai veiktu metodes uzlabojumu pārbaudi programmā R tiek izveidotas arī citas imputācijas metodes. Salīdzināšanai tiek izmantota tuvākā kaimiņa metode, k tuvāko kaimiņu metode, imputācija ar vienkārša gadījuma izlasi ar atkārtojumiem un ar vienkārša gadījuma izlasi bez atkārtojumiem.

Darbs sastāv no ievada, 5 nodaļām ar apakšnodaļām, secinājumiem, izmantotās literatūras saraksta ar 5 avotiem un pielikuma. Darba apjoms ir 33 lappuses, darbā attēlotas 3 tabulas un 1.attēls. 1. nodaļā ir aprakstīta uzlabotās balansētās k tuvāko kaimiņu imputācijas metodes teorētiskā daļa. 2. nodaļā ir īsi aprakstīta k tuvāko kaimiņu imputācijas metode un 3. nodaļā ir īss apraksts vienkārtīgajai imputācijai ar vienkāršu gadījuma izlasi. Imputāciju iztrūkšanas šablonu apraksts, kuri mehāniski rada datu iztrūkšanu, atrodams 4.nodaļā un izmantotie dati, Montekarlo novērtējumu aprēķins, iegūtie rezultāti atrodami 5.nodaļā. Secinājumos salīdzina metodes. Pielikumā atrodamas programmā R izstrādātās metodes un novērtējumu aprēķinu kodi.

Darba uzdevums ir izstrādāt uzlabotās metodes algoritmu un salīdzināt tās iegūtos rezultātus ar citām izvēlētām imputācijas metodēm. Metožu salīdzināšanai savā starpā izmantot aprēķinus ar Montekarlo novērtējumiem un secināt, vai dotā imputācijas metode ar uzlabojumiem dod labākus novērtējumus salīdzinājumā ar izvēlētajām metodēm.

# 1. BALANSĒTĀ K TUVĀKO KAIMIŅU IMPUTĀCIJAS METODE

Darbs lielā mērā balstās uz publikāciju [1]. Tuvākā kaimiņa metode ir *Hot-deck* metodes apakšmetode. Tā ir jauna *hot-deck* imputācijas metode, kura nosaukta par balansēto k tuvāko kaimiņa imputācijas metodi (b$k$NNI). Galvenā šīs metodes iezīme ir tā, ka donoru atlase tiek aplūkota kā izlases problēma un tajā izmanto kalibrāciju un balansētu izlasi, kas ļauj izvēlēties donorus tā, lai populācijas novērtējumi būtu tādi paši kā gadījumā, ja neiztrūktu neviena datu punkta. Šī metode ir neparametriska. Metodē donorus izvēlas no saņēmēja kaimiņiem un katram recipientam donors tiek izvēlēts starp tā k tuvākajiem kaimiņiem. Vienas vienības trūkstošā vērtība tiek aizpildīta ar kādas citas līdzīgas vienības novēroto vērtību, kā arī šīs metodes jauninājums nebalstās tikai uz to faktu, ka donoru atlasē tiek izmantota balansēta izlase, bet gan arī uz faktu, ka tas ir saistīts ar neparametrisku donoru atlasi.

Metode sastāv no diviem soļiem:

1. Tiek iegūta varbūtību imputācijas matrica $\varphi^{[bk]}$
2. Tiek ģenerēta imputācijas matrica $\phi^{[bk]}$

## 1.1. Apzīmējumi un jēdzieni

Uzskata, ka ir galīga populācija $U = \{1,2,...,i,...,N\}$ un pieņem, ka interesējošie mainīgie ir $y = (y_1,y_2,...,y_i,...,y_N)^T$. Nejauša izlase $S$ apjomā $n$ ir iegūta no $U$ ar dotu izlases dizainu $p(\cdot)$. Ar $\pi_i = Pr(i \in S)$ apzīmē pirmās kārtas iekļaušanas varbūtības vienībai $i$ un ar $d_i = 1/\pi_i$ apzīmē Horvica-Tomsona svarus. $Q$ papildus mainīgo vektors $x_i = (x_{i1}, x_{i2},...,x_{iQ})^T$ ir zināms katrai vienībai $i$ izlasē $S$. Respondentu apakšizlase $S_r \subset S$ ir iegūta no $S$ ar parasti nezināmu nosacīto sadalījumu $q(S_r \mid S)$. Interesējošā mainīgā $y_i$ vērtības ir zināmas tikai vienībām no $S_r$. Ar $S_m = S \backslash S_r$ apzīmē $S$ un $S_r$ starpību, tātad, $S_m$ sastāv no vienībām ar trūkstošajām vērtībām (nerespondentiem). Šo apakšgrupu izmērs ir $n_r$ un $n_m$ ar $n_r + n_m = n$.

## 1.2. Nejaušā k tuvākā kaimiņa imputācijas metode

Nerespondentu vienības $j \in S_m$ k tuvākie kaimiņi ir definēti kā $k$ vislīdzīgākās respondentu vienības $i \in S_r$

$$knn(j) = \{i \in S_r \mid rank(d(i,j)) \leq k\}, \tag{1}$$

kur $d(\cdot,\cdot)$ ir Mahalanobis distance aprēķināta ar palīgmainīgajiem, kuri nav konstantes

$$d(i,j) = \{(x_i - x_j)^T \sum^{-1} (x_i - x_j)\}^{1/2}, \tag{2}$$

kur $\sum$ ir variācijas-kovariācijas nenoteiktu palīgmainīgo matrica. Variācijas-kovariācijas matrica ir kvadrātiska matrica, kura satur dispersijas un kovariācijas, kas saistītas ar dažādiem mainīgajiem. Elementi, kuri atrodas uz diognāles satur mainīgo dispersijas un elementi, kuri atrodas ārpus diognāles satur kovariācijas starp visiem iespējamajiem mainīgo pāriem. Matrica ir simetriska.

Nejaušā k tuvākā kaimiņa imputācijas metode ($k$NNI) sastāv no trūkstošo vērtību $j \in S_m$ aizstāšanas ar kādu vienu no k tuvāko kaimiņu vērtībām. Donori ir nejauši izvēlēti ar vienādām varbūtībām. Rezultāti tiek iegūti kā $\varphi^{[k]} = (\varphi_{ij}^{[k]})$, $(i,j) \in S_r \times S_m$ ar

$$\varphi_{ij}^{[k]} = \begin{cases} \frac{1}{k} & \text{ja i pieder j k tuvākajiem kaimiņiem} \\ 0 & \text{citādi.} \end{cases} \tag{3}$$

$k$NNI un varbūtību imputācijas matrica ir saistīti, kur matrica satur tieši k ne nulles koeficientus katrā kolonnā un visi šie ne nulles koeficienti ir vienādi ar $1/k$.

## 1.3. Kalibrācija

Varbūtību imputācijas matrica $\varphi^{[bk]}$ tiek iegūta ar kalibrācijas palīdzību. Uzskata, ka $Q$ palīgmainīgo vektors $x_i = (x_{i1}, x_{i2},...,x_{iQ})^T$ ir zināms katrai populācijas $U$ vienībai. Kalibrācijas mērķis ir atrast kalibrācijas svarus $w_i$ $i \in S$ tik tuvu cik iespējams sākuma dizaina svariem $d_i = 1/\pi_i$ attiecībā uz kalibrācijas vienādojumu

$$\sum_{i \in S} w_i x_i = \sum_{i \in U} x_i = X \tag{4}$$

Vairākas distances funckcijas ir izteiktas kā vidējo mērījumu distance starp sākuma dizaina svariem $d_i = 1/\pi_i$ un beigu svariem $w_i$. Katra distance nodrošina īpašu formu

beigu svariem $w_i$. Izlases balansēšanas metode ir iegūta kalibrācijā, ņemot vērā distances funkciju $G(\cdot,\cdot)$, kura dota formā

$$G(w_i,d_i) = w_i \log\left(\frac{w_i}{d_i}\right) - w_i + d_i \tag{5}$$

Tas noved pie beigu svariem $w_i = d_i \exp(\lambda^T x_i)$, kur vektors $\lambda = (\lambda_1, \lambda_2,...,\lambda_Q)^T$ ir atrisinājums kalibrācijas vienādojumam

$$\sum_{i\in S} d_i \exp(\lambda^T x_i) x_i = \sum_{i\in U} x_i. \tag{6}$$

## 1.4. Varbūtību imputācijas matricas iegūšana

Varbūtību imputācijas matricas $\varphi^{[bk]}$ iegūšanas procedūra. Matricai ir jāapmierina

$$\varphi_{ij}^{[bk]} \neq 0 \quad \text{tikai ja} \quad i \in \text{knn}(j), \tag{7}$$

Imputāciju matricai ir jāapmierina vienādojumi $\sum_{i\in S_r} \varphi_{ij} = 1$ katram $j \in S_m$ un $\varphi_{ij} \geq 0$ katram $(i,j) \in S_r \times S_m$.

Ierobežojums $E_I(\hat{X}_I) = \hat{X}$ nosacīti attiecas uz izlases mehānismu un nerespondences mehānismu, jo imputācijas mehānisms nodrošina nenovirzītu kopējo imputēto novērtējumu priekš palīgmainīgajiem. Vienādojums $E_I(\hat{X}_I) = \hat{X}$ ir ekvivalents

$$\sum_{j\in S_m} d_j \sum_{i\in S_r} \varphi_{ij}^{[bk]} x_i = \sum_{j\in S_m} d_j x_j. \tag{8}$$

Tātad, imputāciju varbūtības matricai $\varphi^{[bk]} = (\varphi_{ij}^{[bk]})$ ir vienlaicīgi jāapmierina vienādojumi

$$\varphi_{ij}^{[bk]} \neq 0 \quad \text{tikai ja} \quad i \in \text{knn}(j), \tag{9}$$

$$\sum_{i\in S_r} \varphi_{ij}^{[bk]} = 1 \quad \text{katram} \quad j \in S_m, \tag{10}$$

$$\varphi_{ij}^{[bk]} \geq 0 \quad \text{katram} \quad (i,j) \in S_r \times S_m, \tag{11}$$

$$\sum_{j\in S_m} d_j \sum_{i\in S_r} \varphi_{ij}^{[bk]} x_i = \sum_{j\in S_m} d_j x_j. \tag{12}$$

Varbūtību imputācijas matricas $\varphi^{[bk]}$ eksistenci nodrošina augstāk minētie nosacījumi. Zemāk aprakstītais algoritms parāda, kā iegūt varbūtību imputācijas matricu $\varphi^{[bk]}$. Galvenā algoritma ideja ir aprēķināt imputācijas matricu, vienlaicīgi apmierinot vienādojumus (9), (10), (11) un (12). Algoritms tiek sākts ar $\varphi^{[k]}$ matricu. Visos soļos nulles koeficients paliek nemainīgs, kas nozīmē, ka vienādojums (9) ir apmierināts. Pēc tam kalibrēšana un normalizēšana tiek mainītas. Kalibrēšana nodrošina matricas $\varphi(2\ell)$, kad $\ell \geq 1$, atbilstību vienādojumiem (11) un (12). Taču šī matrica $\varphi(2\ell)$ nav varbūtību imputācijas matrica, jo tā neapmierina vienādojumu (10). Normalizācija nodrošina matricas $\varphi(2\ell + 1)$ kad $\ell \geq 1$ atbilstību vienādojumiem (10) un (11), bet nav obligāti nepieciešams, lai atbilstu vienādojumam (12). Algoritms apstājas, kad normalizācijas matrica $\varphi(2\ell + 1)$ aptuveni apmierina vienādojumu (12). Matrica $\varphi^{[bk]}$ tiek uzskatīta par pēdējo $\varphi(2\ell + 1)$. Attiecīgi $\varphi^{[bk]}$ vienlaicīgi apmierina vienādojumus (9), (10), (11) un (12).

**Algoritms 1** Varbūtību imputācijas matricas $\varphi^{[bk]}$ iegūšana

---

- *Inicializācija*

  Uzdod $\varphi(1) = \varphi^{[k]}$, varbūtību imputācijas matrica attiecināta uz $k$NNI, kura definēta izteiksmē (3).

- *Iterācija*

  Atkārtot, kad $\ell = 1,2,...$

  - *Kalibrācija*

    Kad $i \in S_r$ aprēķina sākuma svarus $\tilde{d}_i = \sum_{j \in S_m} d_j \varphi(2\ell - 1)$ un iegūst kalibrētus svarus $w_i = \tilde{d}_i \exp(\lambda^T x_i)$, izmantojot grābšanas metodi. Kalibrācijas vienādojums ir

    $$\sum_{i \in S_r} w_i x_i = \sum_{j \in S_m} d_j x_j.$$

  - Matricu $\varphi(2\ell)$ definē kā $\varphi(2\ell)_{ij} = \varphi(2\ell - 1)_{ij} \exp(\lambda^T x_i)$.

  - *Normalizācija*

    Matricu $\varphi(2\ell + 1)$ definē kā $\varphi(2\ell + 1)_{ij} = \frac{\varphi(2\ell)_{ij}}{\sum_{j \in S_m} \varphi(2\ell)_{ij}}$.

  - *Beigu kritērijs*

    Ja

    $$\max_{1 \le q \le Q} \left| \frac{\sum_{j \in S_m} \sum_{i \in S_r} d_j \varphi(2\ell + 1)_{ij} x_{iq} - \sum_{j \in S_m} d_j x_{jq}}{\sum_{j \in S_m} d_j x_{jq}} \right| \le tol$$

    kādai mazai fiksētai tolerancei $tol$, tad

    * tad $\varphi^{[bk]} = \varphi(2\ell + 1)$,

    * Apstāties.

---

## 1.5. Donora izvēle

Šīs procedūras galvenā ideja ir katram recipientam izvēlēties donoru starp respondentiem, apmierinot nosacījumus. Mērķis ir uzģenerēt matricu $\phi^{[bk]}$ tādu, ka

$$\sum_{i \in S_r} \frac{d_j \varphi_{ij}^{[bk]} x_i}{\varphi_{ij}^{[bk]}} \phi_{ij}^{[bk]} = \sum_{i \in S_r} d_j \varphi_{ij}^{[bk]} x_i \quad \text{katram} \quad j \in S_m, \tag{13}$$

$$\mathrm{E}_I(\phi_{ij}^{[bk]}) = \varphi_{ij}^{[bk]} \quad \text{katram} \quad (i,j) \in S_r \times S_m. \tag{14}$$

Bieži vien matrica $\phi_{ij}^{[bk]}$ neatbilst augstāk minētajiem vienādojumiem un nosacījumi neizpildās. Taču, kad ģenerē matricu ar zemāk aprakstīto procedūru, kamēr tiek atrasts atrisinājums, ierobežojumi ir atvieglotāki.

Uzskata, ka populācija ir

$$\dot{U} = \{(i,j) | i \in S_r, j \in S_m\}.$$

Populācija ir sadalīta $n_m$ stratās $\dot{U}$, $j \in S_m$, kur $\dot{U} = \{(i,j) | i \in S_r\}$. Katra strata atbilst vienam recipientam. Tad, katrā stratā tiks izvēlēta tieši viena vienība, nodrošinot katram recipientam vienu donoru. Katrai vienībai $(i,j) \in \dot{U}$ uzskata, ka sākuma iekļaušanas varbūtības ir

$$\dot{\pi}_{(i,j)} = \varphi_{ij}^{[bk]},$$

un papildus mainīgie

$$\dot{\mathrm{x}}_{(i.j)} = d_j \varphi_{ij}^{[bk]} \mathrm{x}_i.$$

Matricā $\phi^{[bk]}$ ir 1, ja respondents $i$ ir donors nerespondentam $j$ un 0 otrā gadījumā, kad respondents nav donors, uzskata

$$\mathbb{1}_{(i,j) \in S} = \phi_{ij}^{[bk]}, \quad \text{kur}$$

$\mathbb{1}_{(i,j) \in S}$ ir indikātora funkcija, kura pieņem vērtību 1, ja vienība $(i,j)$ pieder S. Tas nozīmē, ka respondentam $i \in S_r$ ir jāimputē nerespondenta $j \in S_m$ trūkstošā vērtība, ja vienība $(i,j)$ ir izvēlēta izlasē. Problēma vienādojumos (13) un (14) var tikt pārrakstīta

$$\sum_{(i,j) \in \dot{U}_j} \frac{\dot{\mathrm{x}}_{(i,j)}}{\dot{\pi}_{(i,j)}} \mathbb{1}_{(i,j) \in S} = \sum_{(i,j) \in \dot{U}_j} \mathrm{x}_{(i,j)} \quad \text{katram} \quad j \in S_m, \tag{15}$$

$$\mathrm{E}_I(\mathbb{1}_{(i,j)\in S}) = \dot{\pi}_{(i,j)} \quad \text{katram} \quad (i,j) \in \dot{U}. \tag{16}$$

Šī ir tipiska problēma stratificēti balansētai izlasei, kur ir tikai viena vienība katrā stratā, jo katrs respondents saņem tieši tikai vienu vērtību. Matricas $\phi^{[bk]}$ iegūšanas procedūra ir zemāk aprakstītajā algoritmā.

---

**Algoritms 2** Imputācijas matricas $\phi^{[bk]}$ iegūšana

---

- *Stratificēta balansēta izlase*

  Izvēlas stratificētu balansētu izlasi $S$ populācijā $\dot{U} = \{(i,j)|i \in S_r, j \in S_m\}$ ar metodi

$$\sum_{(i,j)\in\dot{U}_j} \frac{\dot{\mathrm{x}}_{(i,j)}}{\dot{\pi}_{(i,j)}} \mathbb{1}_{(i,j)\in S} = \sum_{(i,j)\in\dot{U}_j} \mathrm{x}_{(i,j)} \quad \text{katram} \quad j \in S_m,$$

$$\mathrm{E}_I(\mathbb{1}_{(i,j)\in S}) = \dot{\pi}_{(i,j)} \quad \text{katram} \quad (i,j) \in \dot{U},$$

  kur

  - $\dot{\mathrm{x}}_{(i.j)} = d_j \varphi_{ij}^{[bk]} \mathrm{x}_i$ ir balansēto mainīgo vektors saistīts ar vienībām $(i,j) \in S_r \times S_m$,
  - $\dot{\pi}_{(i,j)} = \varphi_{ij}^{[bk]}$ ir iekļaušanas varbūtības pievienotas vienībām $(i,j) \in S_r \times S_m$,
  - $\dot{U} = \{(i,j)|i \in S_r, j \in S_m\}$ ir pievienotās stratas vienībām $(i,j) \in S_r \times S_m$.

- *Imputāciju matrica*

  Matricu $\phi^{[bk]}$ definē kā $\phi^{[bk]} = \mathbb{1}_{(i,j)\in S}$.

---

# 2. TUVĀKĀ KAIMIŅA METODE

Balstoties uz [2] literatūras avotu, tuvākā kaimiņa metode ir *Hot-deck* metodes apakšmetode, kur izpildās viens konkrēts nosacījums - tiek izmantota "blakus esošā" elementa vērtība. *Hot-deck* un tuvākais kaimiņš dod vienādus rezultātus pie noteiktiem nosacījumiem. *Hot-deck* metodē trūkstošās vērtības tiek imputētas no tā paša datu kopuma, kad tiek izmantoti tikko savāktie un apkopotie dati. Pastāv arī *Cold-deck* metode, kura tāpat kā *Hot-deck* ir donoru tipa metode, kad trūkstošo datu aizvietošanai tiek izmantotas reālas vērtības. Vērtības *Cold-deck* metodei iegūst no iepriekšējajiem apsekojumiem jeb pagātnes datiem.

Jau no metodes nosaukuma skaidrs, ka nerespondentu atbilžu vietās tiek imputēti dati no citiem donoriem, kuriem ir līdzīgas pazīmes ar nerespondējošajiem elementiem. Tiek imputēta tā elementa atbilde, kura pēc pazīmēm ir vistuvākā nerespondentam. Metodes būtība ir tāda, ka datu masīvs tiek sakārtots pēc konkrētām pazīmēm un tad, kad tiek atrasta trūkstošā vērtība, tā tiek imputēta no iepriekšējām vai nākamajām vērtībām.

Šī metode darbojas labi un ir plaši pielietojama. Tuvākā kaimiņa imputācijas veidu var lietot, kad zināma papildinformācija par elementiem (respondentiem), piemēram, vecums, dzimums un dzīvesvieta. Imputējamo donora elementu izvēlas pēc kādas no pieejamajām pazīmēm. Izvēlas to elementu, kurš pēc attiecīgās pazīmes ir vistuvākais nerespondentam. Ja imputācijas klases ietvaros tiek izraudzīta vērtība pēc gadījuma principa, tad pielietojamā metode ir tieši tāda pati kā *Hot-deck* metode.

Tuvākā kaimiņa metodes ierobežojumi:

1. Dažreiz ir problēma izvēlēties tādu mēru, ar ko tiek atrasts "tuvākais kaimiņš".

2. Skaitļošanas sarežģītība.

3. Ar labu apkārtējo informāciju, ne tikai mainīgā vērtības, bet arī ieraksta līmenī var iegūt labus rezultātus. Risks ir daudz lielāks, ja ārējā informācija nav tik laba. Bet joprojām ir labākā no reālo skaitļu metodēm – kad tiek lietots liels datu masīvs.

# 3. VIENKĀRTĪGĀ IMPUTĀCIJA AR VIENKĀRŠU GADĪJUMA IZLASI

Šajā nodaļā tiek izmantots [2] literatūras avots. Imputācija ar reālām vērtībām ir viena no vienkārtīgo imputāciju apakšgrupas. Tā sastāv no metodēm, kas imputāciju veic ar reālām vērtībām jeb donoriem. Donora imputētā vērtība vienmēr ir reāla, tā nav sarēķināta vai novērtēta. Donori tiek izmantoti gan no imputējamā mainīgā citām vērtībām, gan no kādiem citiem datiem. Donora imputācija ir imputācijā pielietojamā metode. Donors izmanto jau esošās vērtības, lai aizstātu kādu citu trūkstošo mainīgo. Tā vērtība tiek nokopēta un ierakstīta saņēmējam vajadzīgajā vietā.

Izmantojot reālās donora vērtības, nevar rasties pārsteidzīgas vērtības, jo tiek imputēta esošā donora vērtība. Var būt 2 metodes, kad donors tiek izraudzīts ar gadījuma palīdzību:

1. Vienkārša gadījuma izlase ar elementu atkārtošanos. Šajā gadījumā izraudzītā vērtība, kas tiek ievietota trūkstošās vērtības vietā, var tikt izvēlēta arī kāda cita trūkstošā mainīgā vērtības vietā.

2. Vienkārša gadījuma izlase bez elementu atkārtošanās. Izraugās konkrēto donoru, bet no tālākām vērtību imputācijām šo vērtību izslēdz. Ja trūkstošo vērtību skaits pārsniedz 50%, tad šajā gadījumā visas trūkstošās vērtības nav iespējams imputēt.

Donoru bieži izraugās faktora tipa mainīgajiem, it sevišķi tajos gadījumos, kad vienā rindā trūkst vairāki ieraksti. Tas ir, piemēram, gadījumos, kad izlases apsekojumā respondenti nav atbildējuši uz daļu no jautājumiem.

# 4. IMPUTĀCIJAS IZTRŪKŠANAS ŠABLONI

Šajā nodaļā tiek izmantots [3] literatūras avots. Tie ir mehānismi, kas rada datu iztrūkšanu. Šī tēma attiecas uz mehānismiem, kas noved pie datu iztrūkšanas, kā arī uz jautājumu par to vai fakts, ka ir trūkstošie mainīgie ir saistīts ar papildmainīgo kopu. Datu iztrūkšanas mehānismiem ir būtiska nozīme kopš trūkstošo datu metožu īpašības ir ļoti spēcīgi atkarīgas no atkarīgo mainīgo rakstura šajos mehānismos. Tiek apstrādāti trūkstošo datu rādītāji kā gadījuma lielumi un viņiem tiek piešķirts sadalījums.

Apzīmē pilno datu kopu $Y = (y_{ij})$ un trūkstošo datu indikātora matricu $M = (M_{ij})$, tādu, kur $M_{ij} = 1$, ja $y_{ij}$ nav zināms un $M_{ij} = 0$, ja $y_{ij}$ ir zināms. Trūkstošo datu mehānisms ir raksturots ar $M$ nosacīto sadalījumu dotam $Y$, teiksim $f(M|Y, \phi)$, kur $\phi$ satur nezināmus parametrus. Ja iztrūkšana nav atkarīga no $Y$ datu vērtībām, trūkstošajām vai novērotajām, tad

$$f(M|Y, \phi) = f(M|\phi) \quad \text{visiem} \quad Y,\phi, \tag{17}$$

dati tiek saukti ar pilnībā nejaušu iztrūkšanu (MCAR - *missing completely at random*) - jāņem vērā, ka šis pieņēmums nenozīmē to, ka šablons pats par sevi ir nejaušs, bet drīzāk, ka iztrūkšana nav atkarīga no datu vērtībām. Ar $Y_{obs}$ apzīmē novērotās komponentes vai ierakstus $Y$ un ar $Y_{mis}$ trūkstošās komponentes. Pieņēmums ir mazāk ierobežojošs nekā MCAR ir, kad iztrūkšana ir atkarīga tikai no novērotajām konponentēm $Y_{obs}$ no $Y$ un ne no komponentēm, kas trūkst. Tas ir,

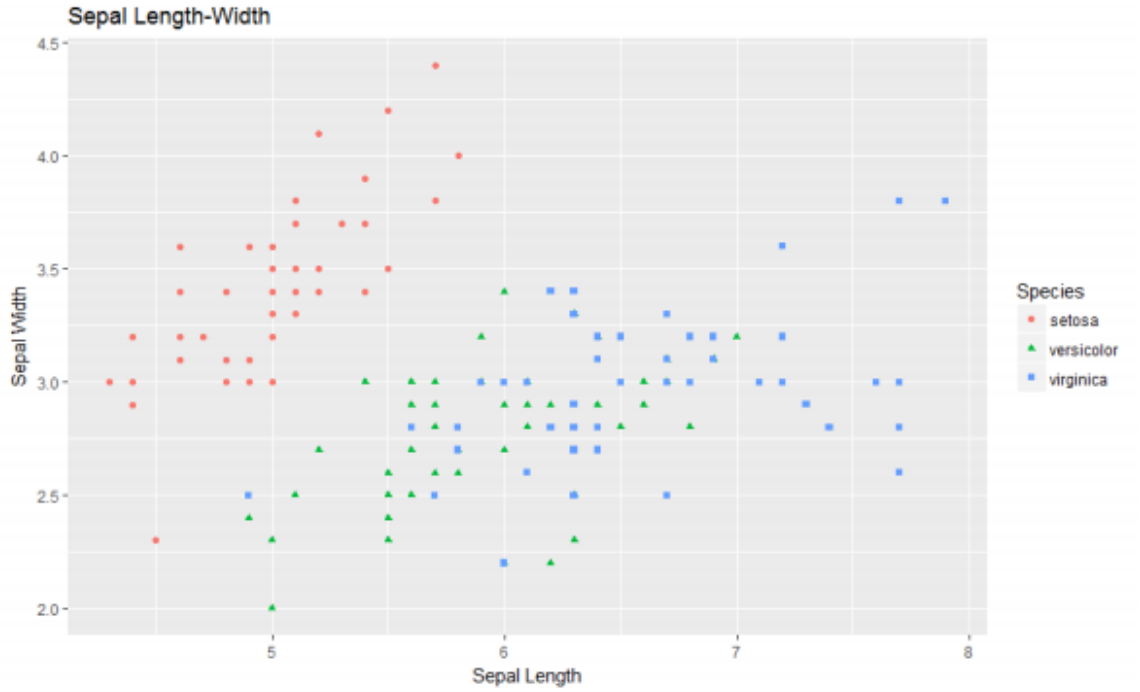
$$f(M|Y, \phi) = f(M|Y_{obs}, \phi) \quad \text{visiem} \quad Y_{mis}, \phi. \tag{18}$$

Tad trūkstošo datu mehānisms tiek saukts par nejaušu iztrūkšanu (MAR - *missing at random*). Mehānisms tiek saukts par nenejaušu iztrūkšanu (NMAR - *not missing at random*), ja $M$ sadalījums ir atkarīgs no trūkstošajām vērtībām datu matricā $Y$.

# 5. REZULTĀTI

## 5.1. Dati

Rezultātu iegūšanai tiek izmantota programmā R [5] iebūvētā datu tabula *iris*. *Iris* datu - īrisa sugu attēlojums grafiski:




1. att. *Iris* datu attēlojums grafiski

*Iris* datu kopa apraksta īrisa sugu kauslapas un ziedlapas garumu un platumu, kopa sastāv no 150 ierakstiem. Tā sastāv no:

- **y:** īrisa suga (Species)
- $x^1$ : īrisa sugas kauslapas garums (Sepal.length)
- $x^2$ : īrisa sugas kauslapas platums (Sepal.width)
- $x^3$ : īrisa sugas ziedlapas garums (Petal.length)
- $x^4$ : īrisa sugas ziedlapas platums (Petal.width)

Korelācija starp **y** un mainīgajiem $x^1$, $x^2$, $x^3$ un $x^4$ ir 0.78, -0.43, 0.95 un 0.96. Populācijas lielums ir $N = 150$. Rezultāti tika pārbaudīti 3 gadījumos:

1. gadījums, kad no datu kopas tika izvēlēti visi četri papildmainīgie ($x^1$, $x^2$, $x^3$, un $x^4$),

2. gadījums, kad no datu kopas tika izvēlēti 3 papildmainīgie ($x^1$, $x^3$, un $x^4$), kuri visstiprāk korelē ar **y**,

3. gadījums, kad no datu kopas tika izvēlēts tikai papildmainīgais ($x^3$), kurš vismazāk korelē ar **y**.

$R^2$ tiek iegūts no lineārās regresijas modeļa $y = \beta_0 + \beta_1 x_1 + \beta_p x_p + \varepsilon$, kur $y$ ir atkarīgais mainīgas, $\beta$ ir regresoru koeficients, $x_1$,...,$x_n$ neatkarīgie mainīgie un kļūdas ir normāli sadalītas un dispersija ir konstanta. [4] No lineārās regresijas modeļa programmā R iegūst, ka $R^2$ ir 0.93.

## 5.2. Simulāciju iestatījumi

Simulācijas gaita balstās uz Haslera [1] publikāciju. Tiek uzskatīts, ka ir pilna populācija, kas nozīmē, ka $\pi_i = d_i = 1$ katrai vienībai $i$ no populācijas $U$ = 1,2,...,$N$. Izlase sakrīt ar populāciju, tas ir, $S = U$. Simts respondentu kopas veidotas no 100 atbildētības indikātoru vektoru R ģenerēšanas. Katra komponente $r_i$, $i \in U$ no $R$ ģenerēta no Bernulli sadalījuma ar parametru

$$\theta_i = \frac{1}{1 + \exp(1 - \beta x_{il})},$$

kur $\beta$ ir pozitīvs koeficients, kurš izmantots, lai sasniegtu vidējo atbildētības līmeni 70% (MAR), $x_{il}$ ir mainīgā $x^l$ vērtība vienībai $i \in U$ un $l$ = 1,2,3,4.

Katrai respondentu kopai 100 imputācijas tiek veiktas ar sekojošām metodēm:

- NNI: tuvākais kaimiņš,
- SRS: imputācija ar vienkārša gadījuma izlasi ar atkārtojumiem, donori nejauši izvēlēti ar atkārtojumiem no respondentu kopas,
- SRSWOR: imputācija ar vienkārša gadījuma izlasi bez atkārtojumiem. Tas pats, kas SRS, izņemot to, ka donori izvēlēti bez atkārtojumiem no respondentu kopas,
- $k$NNI: $k$-tuvākie kaimiņi,
- b$k$NNI: balansētie $k$-tuvākie kaimiņi,

ar $k$=20. Katrai imputācijai tika novērtēti kopējie rādītāji interesējošajam imputācijas mainīgajam.

## 5.3. Salīdzinājuma mērījumi

Salīdzinājuma mērījumi balstās uz Haslera [1] publikāciju. Lai novērtētu imputētā novērtējuma $\hat{\theta}_I$ novirzi priekš parametra $\theta$, tiek izmantota Montekarlo relatīvā novirze RB. Tā ir definēta kā

$$\mathrm{RB}(\hat{\theta}_I) = \frac{\hat{\theta}_I^* - \theta}{\theta},$$

kur

$$\hat{\theta}_I^* = \frac{1}{M_R}\frac{1}{M_I}\sum_{r=1}^{M_R}\sum_{i=1}^{M_I}\hat{\theta}_I^{r,i},$$

$M_R = 100$ ir respondentu kopu ģenerētais skaits, $M_I$ ir katrai respondentu kopai veiktais imputāciju skaits un $\hat{\theta}_I^{r,i}$ ir iegūts novērtējums priekš $r$-tās ģenerētās respondentu kopas $i$-tās imputācijas. Tāpēc $\hat{\theta}_I^*$ apjoms attēlo parametra $\theta$ novērtētās vērtības vidējo vērtību kopumā $M_R M_I$ simulācijās. Imputētā novērtējuma $\hat{\theta}_I$ mainīgums mērīts caur Montekarlo relatīvo vidējo kvadrātisko kļūdu (RRMSE), kura definēta, kā

$$\mathrm{RRMSE}(\hat{\theta}_I) = \frac{\sqrt{\mathrm{MSE}(\hat{\theta}_I)}}{\theta},$$

kur

$$\mathrm{MSE}(\hat{\theta}_I) = \frac{1}{M_R}\frac{1}{M_I}\sum_{r=1}^{M_R}\sum_{i=1}^{M_I}(\hat{\theta}_I^{r,i} - \theta)^2.$$

Montekarlo relatīvā imputāciju novirze (RRIV), vai relatīvā imputāciju standartkļūda, imputētajam novērtējumam $\hat{\theta}_I$ rēķināta, lai aprēķinātu mērījumu dispersijas summu dēļ imputācijām. Tā ir definēta kā

$$\mathrm{RRIV}(\hat{\theta}_I) = \frac{\sqrt{\mathrm{IV}(\hat{\theta}_I)}}{\theta},$$

kur

$$\mathrm{IV}(\hat{\theta}_I) = \frac{1}{M_R}\sum_{r=1}^{M_R}\frac{1}{M_I - 1}\sum_{i=1}^{M_I}(\hat{\theta}_I^{r,i} - \hat{\theta}_I^{r})^2,$$

un

$$\hat{\theta}_I^{r} = \frac{1}{M_I}\sum_{i=1}^{M_I}\hat{\theta}_I^{r,i}$$

apzīmē $\theta$ vidējo novērtēto vērtību piekš $r$-tās respondentu kopas.

## 5.4. Simulāciju rezultāti

5.4.1. tabula. **Montekarlo novērtējumi 1.gadījumam, kad izvēlēti visi papildmainīgie**

| | | Montekarlo novērtējumi | | |
|---|---|---|---|---|
| Interesējošais parametrs | Metode | RB | RRMSE | RRIV |
| Total | NNI | 0.000 | 0.004 | 0.000 |
| | SRS | 0.014 | 0.101 | 0.063 |
| | SRSWOR | -0.149 | 0.179 | 0.046 |
| | $k$NNI | 0.004 | 0.020 | 0.007 |
| | b$k$NNI | 0.007 | 0.039 | 0.027 |

Tabulā redzami iegūtie Montekarlo novērtējuma mērījumi piecām imputāciju metodēm. Iegūtie rezultāti neapstiprina to, ka piedāvātā metode (b$k$NNI) dotu labāku rezultātu, it īpaši, kad ir stipra lineārā sakarība ($R^2 \approx 0.93$) starp interesējošo mainīgo un papildus mainīgajiem. Rezultāti 1.tabulā parāda to, ka NNI metode pārspēj pārējās imputācijas metodes, kuras ir uzskaitītas rezultātu tabulā. Tā uzrāda vismazāko RRMSE un RRIV interesējošajam parametram.

5.4.2. tabula. **Montekarlo novērtējumi 2.gadījumam, kad izvēlēti visstiprāk ar y korelējošie papildmainīgie**

| | | Montekarlo novērtējumi | | |
|---|---|---|---|---|
| Interesējošais parametrs | Metode | RB | RRMSE | RRIV |
| Total | NNI | 0.000 | 0.000 | 0.000 |
| | SRS | 0.016 | 0.103 | 0.064 |
| | SRSWOR | -0.149 | 0.179 | 0.047 |
| | $k$NNI | 0.000 | 0.000 | 0.000 |
| | b$k$NNI | 0.003 | 0.026 | 0.020 |

5.4.3. tabula. **Montekarlo novērtējumi 3.gadījumam, kad izvēlēts visvājāk ar y korelējošais mainīgais**

| | | Montekarlo novērtējumi | | |
|---|---|---|---|---|
| Interesējošais parametrs | Metode | RB | RRMSE | RRIV |
| Total | NNI | 0.007 | 0.091 | 0.020 |
| | SRS | 0.015 | 0.102 | 0.063 |
| | SRSWOR | -0.149 | 0.179 | 0.047 |
| | $k$NNI | 0.028 | 0.090 | 0.008 |
| | b$k$NNI | 0.000 | 0.081 | 0.050 |

Turklāt 5.4.1.tabula un 5.4.2.tabula parāda to, ka tuvākā kaimiņa principam un balansēšanas principam ir ietekme uz kopējo imputācijas dispersiju. Šī ietekme ir atkarīga no sakarību stipruma starp interesējošo mainīgo un papildus mainīgajiem. Tiešām, 2.gadījumā, kad ir stipra lineārā sakarība starp mainīgajiem, kopējā rādītāja RRIV ir 0.064, kura samazina $k$NNI uz 0 (kaimiņa princips) un b$k$NNI uz 0.020 (kaimiņa princips un balansēšanas princips) turpretim, 3.gadījumā, kad ir vāja lineārā sakarība, šie skaitļi ir 0.063, 0.008 un 0.050. Kā arī, 1.gadījumā, kad tiek izmantoti visi mainīgie, rezultāti priekš b$k$NNI ir sliktāki nekā 2.gadījumā, kad tiek izmantoti mainīgie, kuri stipri korelē ar interesējošo mainīgo.

Rezultāti arī 2.gadījumā un 3.gadījumā parāda to, ka donoru izvēle bez atkārtošanās (SRSWOR) starp respondentiem izraisa mazāku RB un RRIV nekā izvēloties donorus ar atkārtošanos (SRS).

Rezultāti apstiprina to, ka piedāvātās metodes izpildījums balstās uz lineārās sakarības stiprumu, kas regulē datus. Tas arī rezultātu 5.4.2. tabulā un 5.4.3. tabulā ir redzams, ka 3.gadījumā, kad šī lineārā sakarība ir daudz vājāka nekā 2.gadījumā, piedāvātā metode uzrāda sliktākus RRMSE un RRIV mērījumus salīdzinot ar 2.gadījumu. Piedāvātā metode arī uzrāda labākus mērījumus, nekā 1.gadījumā, kad tiek izmantoti visi papildmainīgie, iekļaujot gan vāji, gan stipri korelējošos mainīgos. Taču, piedāvātā metode salīdzinājumā ar citām, uz *iris* datiem, neuzrāda labāko rezultātu. Šajā gadījumā nepārprotami NNI un $k$NNI metodes pārspēj pārējās izvēlētās metodes, kad ir izmantoti stipri korelējošie mainīgie.

# SECINĀJUMI

Darba mērķis ir sasniegts. Ir pārbaudīti un salīdzināti uzlabotās imputācijas metodes iegūtie rezultāti ar citām metodēm un ir izdarīti secinājumi vai uzlabotā metode uz izmantotajiem datiem uzrāda labāku novērtējumu.

Pēc visu metožu apskata un iegūtajiem rezultātiem secināms, ka uz izmantotajiem *iris* datiem uzlabotā metode neuzrāda labāko rezultātu salīdzinājumā ar pārējām metodēm. Kopumā vislabāko rezultātu uzrāda NNI, gadījumos, kad tiek izmantoti visi papildus mainīgie un tikai, kad tiek izvēlēti papildus mainīgie, kuriem ir stiprāka korelācija ar interesējošo mainīgo y. Rezultātu analizēšanā secināms, ka uzlabotā metode lielā mērā ir atkarīga no papildus mainīgo izvēles, kā tas redzams gadījumā, kad izvēlētajiem papildus mainīgajiem ir stipra korelācija ar interesējošo mainīgo y un, kad tiek izvēlēts papildus mainīgas ar visvājāko korelāciju.

Tiek secināts, ka pētījums ar *iris* datiem neapstiprina publikācijas autoru rezultātus, kad izmanto papildmainīgos, kad ir stipra korelācija ar interesējošo mainīgo, gan gadījumā, gan izmanto papildmainīgos ar vāju korelāciju un, kas šajā gadījumā ar *iris* datu masīvu neizpildās.

Tā kā 2.gadījumā, kad izmantoti papildus mainīgie ar visstiprāko korelāciju, NNI un $k$NNI pārliecinoši uzrāda vislabākos novērtējumus, tad iespējams, ka britu statistiķa un biologa Ronalda Fišera izveidotajā *iris* datu masīvā īrisa sugas izveidojušas izteiktus klāsterus, ka uzlabot NNI un $k$NNI gandrīz nav iespējams.

Imputācijas ar vienkāršu gadījuma izlasi ar atkārtojumiem un bez atkārtojumiem nav īsti labi pielietojamas šāda veida imputācijām, jo tās uzrāda vissliktākos novērtējumus.

# IZMANTOTĀ LITERATŪRA UN AVOTI

## Literatūras saraksts


[1] Hasler, Caren, and Yves Tillé. "Balanced $k$-nearest neighbor imputation." arXiv preprint arXiv:1501.07622 (2015).

[2] Budkina N. Izlases apsekojumi. 2015. (Lekciju materiāli)

[3] Little, Roderick JA, and Donald B. Rubin. Statistical analysis with missing data. John Wiley & Sons, 2014.

[4] Yan, Xin, and Xiaogang Su. Linear regression analysis: theory and computing. World Scientific, 2009.

[5] R Core Team (2017). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

# PIELIKUMS

## BALANSĒTĀ K TUVĀKO KAIMIŅU METODE

```
library(LaplacesDemon)
library(data.table)
library(sampling)
library(biotools)


bknni_imputation_function <- function(full_data, tolerance, k,
    columns_of_auxiliary_variables) {
  # respondent
  Sr <- subset(full_data,
               !is.na(full_data[,colnames(full_data)[colSums(is.na(full_data)) >
                   0]]))

  # nonrespondent
  Sm <- subset(full_data,
               is.na(full_data[,colnames(full_data)[colSums(is.na(full_data)) >
                   0]]))

  nr <- nrow(Sr)
  nm <- nrow(Sm)
  n <- sum(nr,nm)
  cov <- cov(full_data[,columns_of_auxiliary_variables])

  ### psii(1)
  psi_1 <- D2.dist(full_data[,columns_of_auxiliary_variables], cov)
  psi_1 <- as.matrix(sqrt(psi_1))
  psi_1 <- psi_1[-c(as.numeric(row.names(Sm))), -c(as.numeric(row.names(Sr)))]



  for (i in 1:ncol(psi_1)){
    for (j in 1:k){
      if (sort(psi_1[,i])[1:k][j] %in% psi_1[,i]){
        psi_1[sort(psi_1[,i])[1:k][j] == psi_1[,i], i] <- 1/k
      }
    }
  }
  psi_1[psi_1 != 1/k] <- 0

  ### Calibration
  Xs <- full_data[row.names(Sr), columns_of_auxiliary_variables]
  total <- colSums(full_data[row.names(Sm), columns_of_auxiliary_variables])
  repeat {
    d <- vector()
    for (i in 1:nrow(psi_1)) {
      d[i] <- sum(psi_1[i,], na.rm = TRUE)
    }

    w <- calib(Xs, d, total, method = "raking")
```

```
    exp <- w/d


    ### psii(2)
    psi_2 <- data.frame(nrow = nrow(Sr), ncol = ncol(Sm))

    for (j in 1:ncol(psi_1)) {
      for (i in 1:nrow(psi_1)) {
        psi_2[i,j] <- psi_1[i,j]*exp[i]
      }
      psi_2[,] <- lapply(psi_2,function(x){ x[is.nan(x)]<-0;return(x)})
    }

    row.names(psi_2) <- row.names(Sr)
    colnames(psi_2) <- row.names(Sm)

    ### normalization - psii(3)
    psi_3 <- data.frame(nrow = nrow(Sr), ncol = ncol(Sm))

    for (j in 1:ncol(psi_2)) {
      for (i in 1:nrow(psi_2)) {
        psi_3[i,j] <- psi_2[i,j]/(sum(psi_2[,j]))
      }
      psi_3[,] <- lapply(psi_3,function(x){ x[is.nan(x)]<-0;return(x)})
    }
    row.names(psi_3) <- row.names(Sr)
    colnames(psi_3) <- row.names(Sm)

    ### criterion

    vectors_q_in_criterion <- vector()
    summ <- vector()
    for (q in 1:length(columns_of_auxiliary_variables)) {
      for (j in 1:nrow(Sm)) {
        summ[j] <- sum(psi_3[,j]*Sr[,q])
        sum_1 <- sum(summ)
        sum_2 <- sum(Sm[,q])
        vectors_q_in_criterion[q] <- (sum_1 - sum_2)/(sum_2)
      }
    }
    max(abs(vectors_q_in_criterion))

    psi_1 <- psi_3

    if (max(abs(vectors_q_in_criterion)) <= tolerance) break }
  return(psi_1)
}


visu_imp_funkcija <- function(datas){
  iteracijas_1_100 <- list()

  for (k in 1:100) {
```

```
  iteracijas_1_100[[k]] <- bknni_imputation_function(data.frame(datas), 0.5,
     20, c(1,2,3,4))
}

matrica_1_100 <- list()
for (j in 1:100) {
  psi_3 <- data.frame(iteracijas_1_100[[j]])
  psi_1_0 <- psi_3
  for (i in 1:ncol(psi_3)) {
    psi_1_0[sample(as.numeric(row.names(psi_3[which(psi_3[,i] != "0"),])),
                size = 1,
                prob = psi_3[which(psi_3[,i] != "0"),i]) == row.names(psi_3),
                     i] <- "1"
  }
  psi_1_0[psi_1_0 != "1"] <- 0
  matrica_1_100[[j]] <- psi_1_0
}

for (k in 1:100) {
  for (i in 1:nrow(matrica_1_100[[k]])) {
    for (j in 1:ncol(matrica_1_100[[k]])) {
      if (matrica_1_100[[k]][i,j] == 1) {
        matrica_1_100[[k]][i,j] <- rownames(matrica_1_100[[k]][i,])
      }
    }
  }
}

for (k in 1:100) {
  cols.num <- c(1:ncol(matrica_1_100[[k]]))
  matrica_1_100[[k]][cols.num] <- sapply(matrica_1_100[[k]][cols.num],as.
     numeric)
  sapply(matrica_1_100[[k]], class)

  matrica_1_100[[k]] <- matrica_1_100[[k]][!rowSums(matrica_1_100[[k]]) ==
     0,]
}

# donori sanemejiem
donori <- list()
for (k in 1:100) {
  don <- vector()
  m <- 1
  for (j in 1:ncol(matrica_1_100[[k]])) {
    don[m] <- max(matrica_1_100[[k]][,j])
    m <- m + 1
  }
  donori[[k]] <- don
}

y <- list()
for (k in 1:100) {
  sanemeji <- data.frame(c(colnames(matrica_1_100[[k]])))
```

```
  y[[k]] <- cbind(sanemeji = sanemeji, donori = data.frame(donori[[k]]))
  colnames(y[[k]]) <- c("sanemeji", "donori")
  y[[k]]$sanemeji <- as.numeric(gsub("\\X","",y[[k]]$sanemeji))
}


donoru_vektors <- list()
k <- 1
for (k in 1:100) {
  vektors <- vector()
  l <- 0
  for (i in 1:150) {
    if (is.na(datas$Species[i])) {
      l <- l + 1
      vektors[i] <- y[[k]]$donori[l]
    } else {
      vektors[i] <- "0"
    }
  }
  donoru_vektors[[k]] <- vektors
  k <- k + 1
}

iteracijas_100_imp <- list()
k <- 1
for (k in 1:100) {
  iteracijass <- data.table(datas)
  iteracijas_100_imp[[k]] <- iteracijass[,imputacijas := data.frame(
     donoru_vektors[[k]])]
  k <- k + 1
}

vektoru_lists <- list()
l <- 1
k <- 1
for (k in 1:100) {
  vec <- vector()
  for (i in 1:150) {
    vec[i] <- ifelse((iteracijas_100_imp[[k]]$imputacijas[i]) %in% rownames(
        iteracijas_100_imp[[k]]),
                     iteracijas_100_imp[[k]]$Species[as.numeric(as.vector(
                        iteracijas_100_imp[[k]]$imputacijas[i]))],0)
  }
  vektoru_lists[[k]] <- vec
  k <- k + 1
}

iteracijas_100_vertibas <- list()
k <- 1
for (k in 1:100) {
  iteracijass <- data.table(iteracijas_100_imp[[k]])
  iteracijas_100_vertibas[[k]] <- iteracijass[,imputacijas_vertibas := data.
     frame(vektoru_lists[[k]])]
  k <- k + 1
```

```
  }

  # pievieno kolonnu ar nosaukumiem, kadi tika imputeti prieks NA
  k <- 1
  for (k in 1:100) {
    iteracijas_100_vertibas[[k]][,imp_sugas := ifelse(iteracijas_100_vertibas[[
        k]]$imputacijas_vertibas == "1", "setosa",
                                      ifelse(iteracijas_100_vertibas[[k]]
                                          $imputacijas_vertibas == "2", "
                                          versicolor",
                                      ifelse(iteracijas_100_vertibas[[k]]
                                          $imputacijas_vertibas == "3", "
                                          virginica",0)))]
    k <- k + 1
  }

  # pievieno kolonnu ar visu (jau esosas vertibas, gan imputeetas)
  k <- 1
  for (k in 1:100) {
    iteracijas_100_vertibas[[k]][,viss_ar_imp := ifelse(iteracijas_100_vertibas
        [[k]]$imputacijas_vertibas == "1", "setosa",
                                       ifelse(iteracijas_100_vertibas[[k]]
                                           $imputacijas_vertibas == "2", "
                                           versicolor",
                                       ifelse(iteracijas_100_vertibas[[k]]
                                           $imputacijas_vertibas == "3", "
                                           virginica",
                                       as.character(iteracijas_100_vertibas[[k
                                           ]]$Species))))]
    k <- k + 1
  }
  return(iteracijas_100_vertibas)
}
```

### K TUVĀKO KAIMIŅU METODE

```
library(caret)
library(data.table)

knn_function <- function(data) {

  iteracijas_knn <- data.table(data)
  iteracijas_knn[,id := c(1:150)]
  iris_train <- iteracijas_knn[!is.na(iteracijas_knn$Species),]
  iris_test <- iteracijas_knn[is.na(iteracijas_knn$Species),]

  iris_fit <- knn3(Species ~ .-id, k = 20, data = iris_train)

  #predict(iris_fit, newdata = iris_test, type = "class")

  res <- predict(iris_fit, newdata = iris_test, type = "class")
  iris_test[, res := res]
  iris_test[, imp := ifelse(res == "setosa", 1, ifelse(res == "versicolor", 2,
      ifelse(res == "virginica", 3,0)))]
```

```
  iteracijas_knn[, varbutibas := 0]

  iteracijas_knn <- data.table(iteracijas_knn)
  for (i in 1:nrow(iris_test)) {
    if (iris_test$id[i] %in% iteracijas_knn$id) {
      iteracijas_knn[iris_test$id[i] == iteracijas_knn$id, 7] <- iris_test$imp[i
          ]
    }
  }

  iteracijas_knn[, varbutibas := ifelse(varbutibas == 1,"setosa",
                                  ifelse(varbutibas == 2, "versicolor",
                                        ifelse(varbutibas == 3, "virginica",
                                           0)))]

  iteracijas_knn[, viss_ar_imp := ifelse(varbutibas == "setosa", "setosa",
                              ifelse(varbutibas == "versicolor", "versicolor",
                                    ifelse(varbutibas == "virginica", "virginica
                                        ",
                                           as.character(Species))))]
  return(iteracijas_knn)

}

### 100 reizes ēimput, katram MAR
knn_imputation_100_f <- function(iteracijas) {
  knn_imputation_100 <- list()
  for (k in 1:100) {
    knn_imputation_100[[k]] <- knn_function(iteracijas)
  }
  return(knn_imputation_100)
}
```

### TUVĀKO KAIMIŅU METODE

```
library(caret)
library(data.table)

knn_function <- function(data) {

  iteracijas_knn <- data.table(data)
  iteracijas_knn[,id := c(1:150)]
  iris_train <- iteracijas_knn[!is.na(iteracijas_knn$Species),]
  iris_test <- iteracijas_knn[is.na(iteracijas_knn$Species),]

  iris_fit <- knn3(Species ~ .-id, k = 1, data = iris_train)

  #predict(iris_fit, newdata = iris_test, type = "class")

  res <- predict(iris_fit, newdata = iris_test, type = "class")
  iris_test[, res := res]
  iris_test[, imp := ifelse(res == "setosa", 1, ifelse(res == "versicolor", 2,
     ifelse(res == "virginica", 3,0)))]
  iteracijas_knn[, varbutibas := 0]
```

```
  iteracijas_knn <- data.table(iteracijas_knn)
  for (i in 1:nrow(iris_test)) {
    if (iris_test$id[i] %in% iteracijas_knn$id) {
      iteracijas_knn[iris_test$id[i] == iteracijas_knn$id, 7] <- iris_test$imp[i
          ]
    }
  }

  iteracijas_knn[, varbutibas := ifelse(varbutibas == 1,"setosa",
                                        ifelse(varbutibas == 2, "versicolor",
                                               ifelse(varbutibas == 3, "virginica",
                                                      0)))]

  iteracijas_knn[, viss_ar_imp := ifelse(varbutibas == "setosa", "setosa",
                                ifelse(varbutibas == "versicolor", "versicolor",
                                       ifelse(varbutibas == "virginica", "virginica
                                           ",
                                              as.character(Species))))]
  return(iteracijas_knn)

}

### 100 reizes ēimput, katram MAR
knn_imputation_100_f <- function(iteracijas) {
  knn_imputation_100 <- list()
  for (k in 1:100) {
    knn_imputation_100[[k]] <- knn_function(iteracijas)
  }
  return(knn_imputation_100)
}
```

## PROGRAMMAS R KODS IMPUTĀCIJAI AR VIENKĀRŠA GADĪJUMA IZLASI AR ATKĀRTOJUMIEM

```
srs_imputation <- function(data) {
  k <- 1
  list_100 <- list()
  for (k in 1:100) {
    ite <- data
    don <- ite[!is.na(ite$Species),]
    for (i in 1:nrow(ite)) {
      if (is.na(ite$Species[[i]])) {
        ite[i,6] <- sample(as.vector(don$Species),1)
      } else {
        ite[i,6] <- ite[i,5]
      }
    }
    list_100[[k]] <- ite
    k <- k + 1
  }

imputacijas <- list()
for (i in 1:100) {
```

```
  vet <- data.table(list_100[[i]])
  imputacijas[[i]] <- vet[, viss_ar_imp := ifelse(vet$V6 == "1", "setosa",
                                          ifelse(vet$V6 == "2", "versicolor",
                                                 ifelse(vet$V6 == "3", "
                                                    virginica",
                                                      as.character(vet$V6))))]
}
 return(imputacijas)
}
```

## PROGRAMMAS R KODS IMPUTĀCIJAI AR VIENKĀRŠA GADĪJUMA IZLASI AR ATKĀRTOJUMIEM

```
srswor_imputation <- function(data) {
  k <- 1
  list_100 <- list()
  for (k in 1:100) {
    ite <- data
    don <- ite[!is.na(ite$Species),]
    a <- as.integer(don$Species)

    for (i in 1:nrow(ite)) {

      if (is.na(ite$Species[[i]])) {
        b <- sample(a,1)
        ite[i,6] <- as.integer(b)
        a <- a[-b]
      } else {
        ite[i,6] <- as.integer(ite[i,5])
      }
    }
    list_100[[k]] <- ite
    k <- k + 1
  }

  imputacijas <- list()
  for (i in 1:100) {
    vet <- data.table(list_100[[i]])
    imputacijas[[i]] <- vet[, viss_ar_imp := ifelse(vet$V6 == "1", "setosa",
                                            ifelse(vet$V6 == "2", "versicolor
                                                ",
                                                  ifelse(vet$V6 == "3", "
                                                      virginica",
                                                        as.character(vet$V6)))
                                                          )]
  }
  return(imputacijas)
}
```

## TRŪKSTOŠO VĒRTĪBU VEIDOŠANA

```
data(iris)
```

```
library(LaplacesDemon)
library(data.table)

beta_vector <- c(rep(0.011111, 4))
x <- list()
betaxil <- list()
teta <- list()
for (i in 1:150) {
  x[[i]] <- as.vector(iris[i, 1:4])
  betaxil[[i]] <- sum(x[[i]] * beta_vector)
  teta[[i]] <- 1/(1 + exp(1 - betaxil[[i]]))
}

df_theta <- data.frame(matrix(unlist(teta), nrow = 150, byrow = T))
colnames(df_theta) <- c("tetas")
mean(as.vector(df_theta[,1])) ### 0.3003178

# izveido 100 MAR
iteracijas <- list()
for (i in 1:100) {
  data(iris)
  mar <- rbern(dim(iris)[1], prob = as.vector(df_theta[,1]))
  iris[mar == 1, 5] <- NA
  iteracijas[[i]] <- iris
}

save(iteracijas, file = "data/0-iris-100-MAR.Rdata")
```

## IMPUTĒŠANA

```
rm(list = ls())
library(LaplacesDemon)
library(data.table)

source("impute-bknni.R")
source("impute-knni.R")
source("impute-srs.R")
source("impute-srswor.R")
source("impute-nni.R")

load("data/0-iris-100-MAR.Rdata")

katram_mar_imp_100_bknni <- list()
for (k in 1:100) {
  katram_mar_imp_100_bknni[[k]] <- visu_imp_funkcija(iteracijas[[k]])
}

katram_mar_imp_100_knni <- list()
for (k in 1:100) {
  katram_mar_imp_100_knni[[k]] <- knn_imputation_100_f(iteracijas[[k]])
}

katram_mar_imp_100_nni <- list()
```

```
for (k in 1:100) {
  katram_mar_imp_100_nni[[k]] <- knn_imputation_100_f(iteracijas[[k]])
}

katram_mar_imp_100_srs <- list()
for (k in 1:100) {
  katram_mar_imp_100_srs[[k]] <- srs_imputation(iteracijas[[k]])
}

katram_mar_imp_100_srswor <- list()
for (k in 1:100) {
  katram_mar_imp_100_srswor[[k]] <- srswor_imputation(iteracijas[[k]])
}

save(katram_mar_imp_100_bknni, file = "data/1-katram_mar_imp_100_bknni.Rdata")
save(katram_mar_imp_100_knni, file = "data/1-katram_mar_imp_100_knni.Rdata")
save(katram_mar_imp_100_srs, file = "data/1-katram_mar_imp_100_srs.Rdata")
save(katram_mar_imp_100_nni, file = "data/1-katram_mar_imp_100_nni.Rdata")
save(katram_mar_imp_100_srswor, file = "data/1-katram_mar_imp_100_srswor.Rdata
    ")
```

## METRIKU APRĒĶINĀŠANA

```
library(data.table)

metrics <- function(katram_mar_imp_100) {

  ######### RB
  summa_setosas <- vector()
  for (m in 1:100) {
    setosas_1 <- vector()
    for (k in 1:100) {
      a <- katram_mar_imp_100[[m]][[k]][,"viss_ar_imp"]
      setosas_1[k] <- nrow(a[a$viss_ar_imp == "setosa",])
    }
    summa_setosas[m] <- sum(setosas_1)
    m <- m + 1
  }

  rb <- ((sum(summa_setosas)/10000) - 50)/50



  ################### RRMSE

  mse_sum <- vector()
  for (m in 1:100) {
    setosas_mse <- vector()
    for (k in 1:100) {
      a <- katram_mar_imp_100[[m]][[k]][,"viss_ar_imp"]
      setosas_mse[k] <- (nrow(a[a$viss_ar_imp == "setosa",]) - 50)^2
    }
    mse_sum[m] <- sum(setosas_mse)
    m <- m + 1
```

```
  }
  mse <- sum(mse_sum)/10000
  rrmse <- sqrt(sum(mse_sum)/10000)/50


  ### RRIV

  setosas_sum <- vector()
  for (i in 1:100) {
    setosas <- vector()
    for (k in 1:100) {
      a <- katram_mar_imp_100[[i]][[k]][,"viss_ar_imp"]
      setosas[k] <- nrow(a[a$viss_ar_imp == "setosa",])
    }
    setosas_sum[i] <- sum(setosas)
    i <- i + 1
  }


  teta_hat_r <- vector()
  for (k in 1:100) {
    teta_hat_r[k] <- setosas_sum[k]/100
  }


  areja_summa <- vector()
  for (i in 1:100) {
    iekseja_summa <- vector()
    for (k in 1:100) {
      a <- katram_mar_imp_100[[i]][[k]][,"viss_ar_imp"]
      pirmais <- nrow(a[a$viss_ar_imp == "setosa",])
      iekseja_summa[k] <- (pirmais - teta_hat_r[i])^2
    }
    areja_summa[i] <- sum(iekseja_summa)/99
    i <- i + 1
  }

  rriv <- sqrt(sum(areja_summa)/100)/50

  bknni_metrikas <- data.frame(rb, rrmse, rriv)
  return(bknni_metrikas)
}

library(data.table)

source("metrics.R")

load("data/1-katram_mar_imp_100_bknni.Rdata")
load("data/1-katram_mar_imp_100_knni.Rdata")
load("data/1-katram_mar_imp_100_nni.Rdata")
load("data/1-katram_mar_imp_100_srs.Rdata")
load("data/1-katram_mar_imp_100_srswor.Rdata")

metrics_bknni <- metrics(katram_mar_imp_100_bknni)
```

```
metrics_knni <- metrics(katram_mar_imp_100_knni)
metrics_nni <- metrics(katram_mar_imp_100_nni)
metrics_srs <- metrics(katram_mar_imp_100_srs)
metrics_srswor <- metrics(katram_mar_imp_100_srswor)

metrikas <- rbind(bknni = metrics_bknni,
                  nni = metrics_nni,
                  knni = metrics_knni,
                  srs = metrics_srs,
                  srswor = metrics_srswor)


sink("metrikas.txt")
metrikas
sink()

round(metrikas, 3)

## grafiskais datu attelojums

scatter <- ggplot(data=iris, aes(x = Sepal.Length, y = Sepal.Width))
scatter + geom_point(aes(color=Species, shape=Species)) +
  xlab("Sepal Length") + ylab("Sepal Width") +
  ggtitle("Sepal Length-Width")
```

Bakalaura darbs ”Balansētās k tuvāko kaimiņu imputācijas metodes uzlabojumu pārbaude” izstrādāts LU Fizikas un matemātikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: Ruāna Pavasare

Rekomendēju darbu aizstāvēšanai

Vadītāja: Dr. math. Nataļja Budkina ( ) 05.06.2017.

Recenzents: Mārtiņš Liberts

Darbs iesniegts Matemātikas nodaļā ___.06.2017.

Dekāna pilnvarotā persona: vecākā metodiķe Dzintra Holsta

Darbs aizstāvēts Valsts pārbaudījuma komisijas sēdē

_____ 06.2017. prot. Nr._____, vērtējums _____

Komisijas sekretāre: asociētā profesore Ingrīda Uļjane