Tartu Ülikool
Matemaatika-informaatikateaduskond
Matemaatilise statistika instituut

Ivo Adermann

Andmete puudumise struktuuri määramise testid

Bakalaureusetöö (9 EAP)

Juhendaja: Ene Käärik

Tartu 2015

Andmete puudumise struktuuri määramise testid

Antud töö eesmärk on anda ülevaade andmete puudumise struktuuridest ning kolmest testist, mille abil etteantud andmestike korral puudumise struktuure üksteisest eristada. Vaatluse all on järgnevad testid: Little'i test, t-testide test ja MM-test.

Igat testi testiti nullhüpoteesi ja sisuka hüpoteesi tingimustes ning mõlemal juhul erineva suuruse ja ehitusega andmestike korral. Nullhüpoteesi simulatsioonides muudeti andmestiku pikkust, tunnuste arvu ja konstantset (ning võrdset) puudumise tõenäosust. Sisuka hüpoteesi simulatsioonides muudeti lisaks andmestiku mõõtmetele ka täpset puudumise struktuuri ennast. Kõik simulatsioonid sooritati R-is.

MM-test osutus kasutuks. Ülejäänud kahest testist töötas Little'i test paremini väiksemate ning t-testide test suuremate andmestike korral. Kiireim ning samuti kõige töökindlam test oli t-testide test, mida võib neist kolmest kõige õnnestunumaks testiks pidada.

Märksõnad: simulatsioon, puuduvad andmed, andmeanalüüs.

Tests to Determine the Missingness Mechanism in the Data

The purpose of given thesis is to give an overview on different possible missingness mechanisms and tests that can determine whether missingness depends on the variables in the data. Three following tests were chosen for the study: Little's test, test of t-tests and MM-test.

Each of them was tested under null hypothesis and alternative hypothesis, using data with different number of rows and columns. In null hypothesis' tests, the constant (and equal) missingness probability was also changed; in alternative hypothesis' tests, the missingness mechanism was also changed. Simulations were conducted in R.

MM-test was of no use. Of other two, Little's test worked better on smaller and test of t-tests on larger data sets. The fastest and the most reliable test was also test of t-tests which can therefore be named the most succesful test out of these three.

Keywords: simulation, missing data, data processing.

Sisukord

Sissejuhatus

Tänapäevastes andmestikes on küllaltki tavaline, et mõnel tunnusel puuduvad mitme vastaja korral väärtused. Selle põhjuseks võib olla inimlik eksitus, kuid sagedamini pole vastavaid andmeid lihtsalt õnnestunud kätte saada ja seda tihtipeale seetõttu, et inimesed pole mingisugusel põhjusel soovinud neid andmeid teistega jagada. Hiljem, andmeid analüüsides, on aga oluline teada, mis põhjustel tühjad lahtrid andmestikus tühjad on.

Tuntud USA statistik Donald R. Rubin tutvustas 1976. aastal puudumiste tüpoloogiat, mida senini puuduvatest andmetest rääkides kasutatakse. Andmete puudumise struktuurid on jagatud kolmeks: täiesti juhuslik puudumine (*missing completely at random*, edaspidi MCAR), juhuslik puudumine (*missing at random*, edaspidi MAR) ning mittejuhuslik puudumine (*not missing at random*, edaspidi NMAR). Antud töös võrreldakse teste, mis testivad, kui tõenäoline on, et etteantud andmestikus on puudumise struktuuriks MCAR. Kui see on piisavalt vähetõenäoline, otsustavad testid, et puudumise struktuur on MAR.

Kõikvõimalike andmeanalüüsi meetodite korral on struktuur MCAR kahtlemata kõige lihtsam variant. Kuna sellisel juhul on igal tabeli real võrdne tõenäosus omada mõnd puuduvat väärtust, võib puuduvaid andmeid sisaldavad read andmestikust lihtsalt kõrvale jätta. Et aga selgitada välja, kas puudumise struktuur siiski on MCAR ja mitte MAR, tuleb kasutada mingisugust testi. Antud töös võrreldakse neist järgmist kolme: Little'i testi, t-testide testi ja üht uuemat testi: R-i paketis „MissMech" olevat „TestMCARNormality" testi, mida siin edaspidi MM-testiks nimetatakse.

Kolme eelnimetatud testi võrreldi erinevates situatsioonides. Nende situatsioonide (teisisõnu andmestike) loomine ja neil testide kasutamine toimusid arvukates R-i simulatsioonides. Erinevates nullhüpoteesi (struktuuri MCAR) simulatsioonides muudeti andmestiku pikkust, laiust (tunnuste arvu) ning puudumiste tõenäosust, sisuka hüpoteesi (struktuuri MAR) simulatsioonides aga andmestiku pikkust, laiust ning seda, kuidas ülejäänud andmete puudumise tõenäosused täpselt andmestiku kahe esimese veeru väärtustest sõltuvad.

Kõikvõimalikke situatsioone, mille korral testide töötamist uurida, saab kahtlemata suurel hulgal välja mõelda. Antud töös tuli kõigi nende hulgast mingisugune osa välja valida. Simulatsioonid, eriti suuremate andmestike korral, võtavad paraku palju aega ning just see aeg pani piirid antud töö mahule.

1 Andmete puudumise struktuurid

Arvatavasti on iga statistik kokku puutunud olukorraga, kus analüüsitavas andmestikus on mõni tühi lahter. Kui see probleemi valmistab, on lihtsaim variant puuduvaid andmeid sisaldavad read vaatluse alt välja jätta, nagu poleks neid olemas olnudki. Leiduvad kaks tüüpilist põhjust, miks niimoodi alati toimida ei tohiks. Esiteks võib andmestikku alles jääda liiga vähe andmeid, kui neid alguseski kuigi palju ei olnud või kui küllaltki arvukates ridades mõni tühi lahter on. Teiseks võib juhtuda, et üksnes allesjäänud andmeid analüüsides saadakse (liiga suure) nihkega hinnangud, sest andmete puudumise struktuur polnud täiesti juhuslik.

Tähistagu Y $(n \times m)$ andmetabelit, kus võib esineda ka puuduvaid väärtuseid. Vastavalt sellele, missugused andmed tabelis olemas on ja millised mitte, on algne andmestik jagatud kaheks: $Y = (Y_{obs}, Y_{mis})$, kus Y_{obs} tähistab vaadeldud osa ning Y_{mis} puuduvat osa andmestikust. Lisaks tähistagu M $(n \times m)$ tabelit puudumisindikaatoritega, kus 1 tähistab puuduvat vaatlust ning 0 olemasolevat vaatlust. Puudumisindikaatorite jaotust tähistagu f(M). Neid tähiseid kasutades defineeris Rubin (1976) puudumiste struktuurid.

Definitsioon 1. Andmestikus on puudumise struktuur täiesti juhuslik (MCAR) parajasti siis, kui kehtib f$(M|Y) =$ f(M).

Teisisõnu on tegemist struktuuriga MCAR, kui tõenäosus, et tabelis mõni konkreetne väärtus puudub, ei sõltu ei selle arvu enda ega ka ühegi teise tunnuse väärtusest. See on kõige hõlpsamini mõistetav ja ka edasisteks analüüsideks kõige lihtsam puudumise struktuur. Struktuur MCAR võib tekkida näiteks siis, kui halva käekirja tõttu ei õnnestu mõnd numbrit välja lugeda ja seetõttu jäetakse vastav lahter andmestikus tühjaks või kui küsitlustiku täitja poole täitmise pealt kiireloomulise kõne saab ja seejärel kohe poolikult täidetud ankeedi tagastab.

Struktuuriga MCAR võib tegemist olla ka siis, kui inimene meelega mõnele küsimusele vastamata jätab. Kui keegi jätab küsimusele vastamata näiteks seepärast, et ta on tugevalt usklik (ja kui ühegi tema küsimustiku küsimuse vastus tema usulisest kuuluvusest ei sõltu), on tegemist ikkagi puudumise struktuuriga MCAR, sest tema mittevastamine pole põhjustatud ühegi küsimuse vastusest.

Definitsioon 2. Andmestikus on puudumise struktuur juhuslik (MAR) parajasti siis, kui kehtib f$(M|Y) =$ f$(M|Y_{obs})$.

Teisisõnu on tegemist struktuuriga MAR, kui tõenäosus, et tabelis mõni konkreetne väärtus puudub, ei sõltu selle arvu enda ega ühegi teise puuduva tunnuse väärtusest, kuid sõltub mõne kolmanda, olemasoleva tunnuse väärtusest. Üks näide struktuurist MAR on järgmine: õpetaja küsib sumisevalt klassilt, kui vanad nad olid esimest korda loomaaias käies, ja jagab seejärel kõigile paberilehed, kuhu ootab kahe küsimuse vastuseid. Esimene küsimus on: „Kas sa kuulsid, mida ma teilt tahvli ees just küsisin?“ Teine küsimus on „Mis on sinu vastus sellele küsimusele?“

Vastamise tõenäosus viimasele küsimusele ei sõltu kindlasti selle õigest vastusest: vaevalt et keegi häbeneb oma vanust esimest korda loomaaeda külastades. Küll aga sõltub sellele vastamise tõenäosus esimese küsimuse vastusest: kui see oli eitav, ei tea õpilane teist küsimust ega saa sellele ka vastata.

Kõige halvem variant andmete analüüsijale on struktuur NMAR. Rubini definitsiooni järgi on puudumise struktuur NMAR parajasti siis, kui see pole MCAR ega MAR. Teisisõnu on tegemist struktuuriga NMAR, kui tõenäosus, et tabelis mõni konkreetne väärtus puudub, sõltub selle arvu enda või mõne teise puuduva tunnuse mitteteadaolevast väärtusest. Klassikaline näide struktuurist NMAR on lihtne küsitlus inimeste sissetulekute kohta: kui küsimustikus on selline küsimus, on väga suure ja ka väga väikese sissetulekuga inimeste korral kahtlemata suurem tõenäosus, et nad jätavad sellele küsimusele vastamata.

Kui seejärel tekitada uus andmetabel, kust on välja jäetud kõik read, kus esialgses andmestikus mõni tühi lahter oli, kerkiksid mitmed probleemid. On võimalik, et tunnuse keskväärtust see väga palju ei mõjutakski, kui kaks äärmust teineteist ära peaksid tasakaalustama, kuid tunnuse dispersioon tuleks säärase andmestiku pealt arvutatuna kindlasti suure nihkega. Samuti jääks tunnuse jaotusest vale mulje, kui vaadata uue andmestiku põhjal joonistatud graafikuid.

Kas puuduvate andmete struktuur on või ei ole NMAR, on pelgalt andmestikku vaadates võimatu öelda, sest ei saa uurida millegi sõltuvust millestki sellisest, mille kohta mingisugust infot ei ole. Üks võimalust teada saada, kas andmete puudumise struktuur on NMAR, on uurida varasemate sarnaste uuringute infot või kasutada mingisugust muud nende kohta teada olevat taustinfot. Struktuuri NMAR ära tundmine pole aga antud töö eesmärk – kõigi siin käsitletavate testide eesmärk on teha vahet puudumise struktuuride MCAR ning MAR vahel.

2 Testid

Tänapäeval leidub hulganisti teste, millega etteantud andmestiku korral teha vahet struktuuride MCAR ja MAR vahel (Enders 2010, lk 18). Neist said antud töö tarvis välja valitud 3: maineka USA statistiku Roderick J. A. Little'i 1988. aastal avaldatud artiklis tutvustatud test, samas artiklis mainitud sisuliselt väga lihtne t-testide test ning 2014. aastal avaldatud R-i paketis „MissMech" olev MM-test. Kõik kolm testi kontrollivad järgnevat hüpoteeside paari:

H_0: Andmestikus on puuduvate andmete struktuur MCAR.
H_1: Andmestikus on puuduvate andmete struktuur MAR.

2.1 Little'i test

Järgnev peatükk põhineb Little'i eelmainitud 1988. aastal avaldatud artiklil.

Tähistagu Y $(n \times m)$ andmetabelit, kus on n rida ning m veergu. Andmestikus võib esineda ka puuduvaid väärtuseid. Kõik andmestiku n objekti jagatakse J gruppi nii, et igas grupis on ühesuguse puudumismustriga objektid. Ühe grupi moodustavad näiteks kõik read, kus pole ühtegi puuduvat väärtust, teise grupi need andmestiku read, kus puudub vaid 1. tunnuse väärtus jne. Edasi tegeldakse vaid nende mustritega, kus on vähemalt u objekti. Antud töös võeti $u = 2$.

Arv m_j tähistagu j. mustris olevate objektide arvu. $\sum m_j = n$. Arv p_j tähistagu j. mustris olevate tunnuste arvu. Vektor $y_{obs,i}$ tähistagu j. mustri i. elementi, mis on $(1 \times p_j)$ vektor ning vektor $\bar{y}_{obs,j} \equiv m_j^{-1} \sum y_{obs,i}$ ehk vektor $\bar{y}_{obs,j}$ koosneb j. mustris esindatud tunnuste keskväärtuste hinnangutest üle üksnes j. mustri. Analoogne vektor $\hat{\mu}_{obs,j}$ koosneb samuti j. mustris esindatud tunnuste keskväärtuste hinnangutest, aga üle kogu andmestiku.

Tähistagu $\sum_{obs,j}$ j. mustris esindatud tunnuste $(p_j \times p_j)$ kovariatsioonimaatriksit ning $\widetilde{\sum}_{obs,j}$ sellesama maatriksi hinnangut. Sümbol $\widetilde{\sum}_{obs,j}^{-1}$ tähistab seega j. mustris esindatud tunnuste kovariatsioonimaatriksi hinnangu pöördmaatriksit, mis on samuti mõõtmetega $(p_j \times p_j)$.

Neid tähistusi kasutades pakkus Little, testimaks, kas andmestikus on puuduvate andmete struktuur MAR või MCAR, välja testi järgmise teststatistikuga:

$$d^2 = \sum_{j=1}^{J} m_j (\bar{y}_{obs,j} - \hat{\mu}_{obs,j}) \widetilde{\sum}_{obs,j}^{-1} (\bar{y}_{obs,j} - \hat{\mu}_{obs,j})^T.$$

Statistik d^2 on asümptootiliselt hii-ruut jaotusega, vabadusastmetega $\sum p_j - m$. Nullhüpoteesile vastav struktuur MCAR lükatakse ümber struktuuri MAR kasuks olulisuse nivool α, kui statistiku d^2 väärtus tuleb suurem kui vastava hii-ruut jaotuse $(1-\alpha)$-kvantiil.

Ilma tegurita $\tilde{\sum}_{obs,j}^{-1}$ mõõdaks Little'i test erinevust iga tunnuse üldkeskväärtuse ning tema mustrite siseste keskväärtuste vahel, arvestamata seda, et tunnused võivad omavahel korreleeritud olla. Sellega läbi korrutamine võimaldab aga arvesse võtta ka tunnustevahelisi korrelatsioone. Nullhüpoteesi olukorras langeksid vektorid $\bar{y}_{obs,j}$ ja $\hat{\mu}_{obs,j}$ kokku ehk statistiku d^2 väärtus püsiks nullilähedane, mistõttu jäädakski õigustatult nullhüpoteesi juurde.

Testi ainsaks praktiliselt keeruliseks sammuks on *j* kovariatsioonimaatriksi leidmine. Nagu ka Little oma artiklis soovitas, tehti seda antud töös *expectation-maximization* algoritmi abil (edaspidi EM algoritm). R-is sobib selle jaoks kasutada paketis „norm" asuvat „em.norm" käsku. See käsk eeldab, et kõik andmed pärineksid normaaljaotustest just nagu antud töös.

EM algoritmi tasub kasutada, kui on vaja hinnata mitme parameetri väärtuseid, mis üksteisest sõltuvad. Kõigepealt antakse igale parameetrile mingisugune algväärtus ning seejärel hakatakse ükshaaval parameetritele uusi hinnanguid suurima tõepära meetodil omistama, eeldades, et ülejäänud parameetrite väärtused on korrektsed. Uute hinnangute leidmine lõpetatakse, kui ühegi parameetri väärtus viimasel ringil enam kuigi palju ei muutunud.

2.2 T-testide test

Little selgitas oma artikli alguses, et kõige lihtsam võimalus testimaks, kas ühe tunnuse keskväärtused erinevad vastavalt sellele, kas mingisuguse teise tunnuse väärtus objektil puudub või ei, on jagada esialgse tunnuse väärtused sellele vastavalt kahte gruppi ning nende gruppide keskväärtuseid t-testi abil võrrelda.

Just niisugusel ideel põhinebki t-testide test. Kui andmestikus leidub tunnuste paar, mille korral t-test alternatiivse hüpoteesi kasuks otsustab, võtab ka terve t-testide test vastu sisuka hüpoteesi, muidu mitte.

Et terve t-testide testi olulise nivoo ei ületaks nivood α, võetakse igas üksikus t-testis olulisuse nivooks Bonferroni paranduse järgi

$$\frac{\alpha}{(m-1)(m-2)},$$

sest sooritatav t-testide arv on iga andmestiku korral kuni $(m-1)(m-2)$.

Tähistagu y_k ja y_l kahe tunnuse väärtuste vektoreid andmestikus Y ning tähistagu $y_{i,j}$ andmestiku i. tunnuse j. elementi. Tähistagu igas t-testis $y_{k,obs}$ vaadeldud elementide $y_{k,j}$

hulka, kus teises vaadeldavas tunnuses vaatlusete $y_{l,j}$ väärtused olemas on ja tähistagu $y_{k,mis}$ vaadeldud elementide $y_{k,j}$ hulka, kus teises vaadeldavas tunnuses vaatlused $y_{l,j}$ puuduvad. Siis avaldub sõltumatute valimite ning erinevate dispersioonidega t-testi teststatistik *T* järgmisel kujul:

$$T = \frac{\bar{y}_{k,obs} - \bar{y}_{k,mis}}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}},$$

kus $\bar{y}_{k,obs}$ ja $\bar{y}_{k,mis}$ on kahe võrreldava grupi keskväärtuste hinnangud, s_1^2 ja s_2^2 vastavalt nende gruppide dispersioonide hinnangud ning n_1 ja n_2 nende gruppide suurused. Teststatistik *T* on ligikaudu t-jaotuse ning järgmise vabadusastmete arvuga:

$$df = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{\left(\frac{s_1^2}{n_1}\right)^2}{n_1 - 1} + \frac{\left(\frac{s_2^2}{n_2}\right)^2}{n_2 - 1}} .$$

(vt näiteks Allwood 2008)

Selle lähenemise probleemiks olevat aga Little'i sõnul olnud meetodi aeglus: *m* tunnusega andmestikus tuleks sooritada kuni $m(m-1) \approx m^2$ t-testi.

Näide 1. Kuidas töötab t-testide test?

y_1	y_2
$y_{1,1}$	$y_{2,1}$
$y_{1,2}$	NA
$y_{1,3}$	NA
$y_{1,4}$	$y_{2,4}$
NA	$y_{2,5}$
$y_{1,6}$	$y_{2,6}$
$y_{1,7}$	$y_{2,7}$
NA	$y_{2,8}$
$y_{1,9}$	NA
$y_{1,10}$	$y_{2,10}$
NA	$y_{2,11}$
NA	$y_{2,12}$

Olgu andmestikus puuduvad väärtused tähistatud sümbolitega „NA". Tunnuste y_1 ja y_2 vahel sooritaks t-testide test kaks t-testi: kõigepealt võrdleks test omavahel vektoreid $(y_{1,1}, y_{1,4}, y_{1,6}, y_{1,7}, y_{1,10})$ ja $(y_{1,2}, y_{1,3}, y_{1,9})$ ning seejärel vektoreid $(y_{2,1}, y_{2,4}, y_{2,6}, y_{2,7}, y_{2,10})$ ja $(y_{2,5}, y_{2,8}, y_{2,11}, y_{2,12})$. Teisisõnu jagatakse iga tunnuse väärtused kaheks vastavalt sellele, kas teise tunnuse vastav väärtus puudub või ei puudu.

2.3 MM-test

Järgnev peatükk põhineb Jamshidiani, Jalali ning Janseni 2014. aastal avaldatud artiklil, kus uuriti teste, mille abil eristada puudumise struktuure MAR ja MCAR ning kus pakuti viimaks välja test, mille abil neil vahet teha.

Üks vajalik samm MM-testi jaoks on kõigi andmestiku tühjade lahtrite täitmine, selleks kasutatakse testis mitmest imputeerimist. Selleks on kaks võimalust: võib eeldada tunnuste normaaljaotust või võib seda mitte eeldada. Antud töös sooritatud simulatsioonides kasutati üksnes normaaljaotusega andmeid, mistõttu sooritati ka simulatsioonides mitmene imputeerimine andmete normaaljaotuse eeldusel.

Erinevalt Little'i testist ja t-testide testist, ei võrrelda MM-testis mitte tunnuste keskväärtuseid, vaid kovariatsioone. Nullhüpoteesi olukord on järgmine:

$$\sum_1 = \sum_2 = \ldots = \sum_J .$$

kus $\sum_j$ tähistab puudumismustris j sisalduvate ridade põhjal arvutatud $(m \times m)$ kovariatsioonimaatriksit, sest enne maatriksite arvutamist on algselt tühjad olnud andmestiku lahtrid prognoositud väärtustega täidetud. Puudumismustrid on defineeritud niisamuti kui Little'i testi puhul ning erinevate puudumismustrite arv on endiselt tähistatud tähega *J*.

Lisaks Little'i testi juures kasutatud tähistustele, tähistagu Y_{ji} *j*. mustri *i*. elementi, mis on $(1 \times m)$ vektor, sest algselt tühjad olnud lahtrid on juba prognoositud väärtustega täidetud ning $\bar{Y}_j$ tähistagu $(1 \times m)$ vektorit *j*. mustri objektide tunnuste keskväärtuste hinnangutest. Lisaks tähistagu *S*, mis on $(m \times m)$ maatriks, kogu andmestiku ühiskovariatsioonimaatriksit (*overall pooled covariance matrix*) ning S^{-1} selle $(m \times m)$ pöördmaatriksit.

$$S = \frac{1}{n-J} \sum_{j=1}^{J} (m_j - 1) \sum_j ,$$

kus maatriks $\sum_j$ tähistab *j*. mustri ridade $(m \times m)$ kovariatsioonimaatriksit (vt näiteks Manly 1994, lk 64).

Nagu Little'i testi puhul, tuli ka MM-testi korral otsustada, mitu objekti peab mustrite grupis vähemalt olema, et gruppi mitte vaatluse alt välja jätta. Antud töös võeti analoogselt Little'i testiga selleks arvuks $u = 2$. Vaikeväärtusena oli selleks arvuks pakutud kuut. Kuna 2 ja 6 on küllaltki erinevad arvud ja kuna MM-test kuigi korralikult ei paistnud töötavat, sai lõpuks ka $u = 6$ puhul testi toimimist uuritud.

MM-testi teststatistikud arvutatakse järgmise eeskirja kohaselt:

Kõigepealt jagatakse objektid *J* gruppi nende puudumismustrite alusel ja liiga väheseid objekte sisaldavad mustrid jäetakse vaatluse alt kõrvale.

Seejärel arvutatakse allesjäänud objektide jaoks teststatistikute F_{ji} väärtused:

$$F_{ji} = \frac{(n - J - m)m_j V_{ji}}{m\left[(m_j - 1)(n - J) - m_j V_{ji}\right]},$$

kus $V_{ji} = \left(Y_{ji} - \bar{Y}_j\right)S^{-1}\left(Y_{ji} - \bar{Y}_j\right)^T$. Nullhüpoteesi kehtides on statistikud F_{ji} F-jaotusega, vabadusastmete arvudega m ning $(n - J - m)$.

Edasi arvutatakse suurused A_{ji} järgmiselt: $A_{ji} = P\left(F > F_{ji}\right)$, kus F on F-jaotusega juhuslik suurus vabadusastmete arvudega m ning $(n - J - m)$.

Nullhüpoteesi kehtides oleksid A_{ji} ühtlasest jaotusest nullist üheni, sest nullhüpoteesi kehtides oleksid F_{ji} sellestsamast F-jaotusest.

Viimaks testitakse, kas eelnev võib tõele vastata. Kui vastav jaotust kontrolliv test jääb olulisuse nivool α nullhüpoteesi ja ühtlase jaotuse juurde, jääb ka terve MM-test olulisuse nivool α nullhüpoteesi ehk struktuuri MCAR juurde; kui ühtlase jaotuse tingimus ümber lükatakse, lükatakse samal olulisuse nivool ka struktuuri MCAR eeldus ümber.

3 Kasutatud andmestike parameetrid

Antud töös kasutati kõigi kirjeldatud testide omavahel võrdlemiseks R-i simulatsioone. Testiti nende 1. liiki vea tegemise tõenäosuseid nullhüpoteesi tingimustes ning 2. liiki vea tegemise tõenäosuseid sisuka hüpoteesi tingimustes. Lisaks märgiti üles iga erineva testi jaoks iga parameetrite kombinatsiooni puhul tööks kulunud aeg. Programmide tööajad sõltusid kahtlemata ka tuvastamatutest müratunnustest, kuid nende mõju peaks olema minimaalne.

Nullhüpoteesi simulatsioonides muudeti järgmisi andmestike parameetreid:

- andmestiku ridade ehk objektide arvu *n* (väärtustega 30, 100, 500 ja 2000);
- andmestiku veergude ehk tunnuste arvu *m* (väärtustega 3, 6 ja 16);
- konstantset puudumise tõenäosust *p* (väärtustega 0,04, 0,1 ja 0,3).

Sisuka hüpoteesi simulatsioonides muudeti järgmisi andmestike parameetreid:

- andmestiku ridade ehk objektide arvu *n* (väärtustega 30, 100, 500 ja 2000);
- andmestiku veergude ehk tunnuste arvu *m* (väärtustega 3, 6 ja 16);
- arvu *v*, mis omandas väärtuseid 1 ja 2 ning näitas, mitme esimese tunnuse väärtustest igal objektil kolmanda tunnuse puudumise tõenäosus sõltub.

Sisuka hüpoteesi olukord oli selline, kus iga objekti esimese kahe tunnuse väärtused olid alati teada. Neist võib mõelda kui mingisugustest küllalt elementaarsetest tunnustest, mida keegi ei varja, nagu inimese sugu või vanus. Kolmanda tunnuse väärtuste puudumise tõenäosused sõltusid aga kahe esimese tunnuse väärtustest vastavail objektidel ega olnud seega konstantsed.

Alternatiivse hüpoteesi andmestikes olid ülejäänud tunnuste väärtuste puudumise tõenäosused konstantsed ja võrdsed ($p = 0{,}15$). Kuna sisuka hüpoteesi andmestike puhul olid vaid 3 esimest tunnust erilised, tundus mõistlikum asendada nullhüpoteesi simulatsioonide tunnuste arvud 6 ja 16 ühe arvuga, sest need variandid poleks omavahel kuigivõrd palju erinenud.

Simulatsioonide arvud olid eri parameetrite väärtuste kombinatsioonide puhul erinevad. Alati sooritati vähemalt 10000 simulatsiooni, kuid sõltuvalt igale simulatsioonile kuluvast ajast võidi neid ka rohkem sooritada. Kõige väiksemate andmestike korral võttis iga simulatsioon niivõrd vähe aega, et paaril juhul sooritati täpsuse suurendamise eesmärgil koguni 5 miljonit simulatsiooni.

Kõigis simulatsioonides genereeriti kõigi tunnuste väärtused üksteisest sõltumatutest normaaljaotustest. Simulatsioonides kasutati olulisuse nivood 0,05. Algselt oli plaanis läbi viia rohkem erinevaid simulatsioone, sest tunnuseid ja parameetreid, mida saab muuta, on palju. Arvukad simulatsioonid, eriti suurte andmestike jaoks, võtavad aga palju aega, mistõttu tuli langetada mingisugune valik uuritavate parameetrite arvu ning nende väärtuste hulgast.

Kõikvõimalikud kombinatsioonid loetletud parameetrite väärtustest on simulatsioonides esindatud ehk suurele osale reaalsetest andmestikest leiaks nende seast mingil määral sarnase

vaste. Täpsed kasutatud koodid ning saadud tulemused tabelitena on ära toodud töö lõpus lisades.

Iga testi iga parameetrite kombinatsiooni korral arvutati 1. ja 2. liiki vigade tegemiste tõenäosuste punktihinnangutele $\bar{p}$ ka 95-protsendised usaldusvahemikud I_μ. Selleks kasutati järgmist valemit:

$$I_\mu = \left(\bar{p} - t_{\alpha/2}(N-1)\sqrt{\frac{\bar{p}(1-\bar{p})}{N}}, \bar{p} + t_{\alpha/2}(N-1)\sqrt{\frac{\bar{p}(1-\bar{p})}{N}}\right),$$

kus tähega N on tähistatud vastav simulatsioonide arv, mis tagab t-jaotuse vabadusastmete arvu $N-1$. Kuna iga üksiku simulatsiooni tulemusest võib mõelda kui Bernoulli jaotusega juhuslikust suurusest dispersiooni hinnanguga $\bar{p}(1-\bar{p})$, on N sellise suuruse keskmise dispersiooni hinnang

$$s^2 = \frac{\bar{p}(1-\bar{p})}{N},$$

millest ruutjuure võttes saab vea tegemise tõenäosuse standardhälbe hinnangu, mida ongi ülemises valemis kasutatud.

4 Simulatsioonide tulemused

4.1 Little'i test

Täpsed R-i koodid, millega antud töös kõik Little'i testi simulatsioonid sooritati, leiab lisadest numbritega 4-6. Kõigi testide simulatsioonide tulemused leiab tabelite kujul lisadest 8-9.

Nullhüpoteesi kontrollimine

Uuritud kolmest testist töötas Little'i test nullhüpoteesi simulatsioonides kõige stabiilsemalt. Parameetrite (n, m, p) erinevaid kombinatsioone oli $4 \cdot 3 \cdot 3 = 36$. Neist 23 kombinatsioonis $(23/36 \approx 64\%)$ jäi testi 1. liiki vea tegemise tõenäosuse punktihinnang vahemikku (0,045, 0,055). Võrdluseks: t-testide testil oli see näitaja $19/36 \approx 53\%$ ning MM-testil üksnes $1/24 \approx 4\%$.

Nii eelmises kui ka järgnevates lõikudes vaadatakse hinnangute täpsuse mõõtudena eelkõige punktihinnanguid mitte usaldusvahemikke, sest väiksemate andmestike korral sai tehtud märksa rohkem simulatsioone, mistõttu tulid nende puhul ka usaldusvahemikud palju kitsamad kui suurte andmestike korral. Kitsam usaldusvahemik kataks aga väiksema tõenäosusega arvu 0,05 kui laiem usaldusvahemik, mistõttu poleks usaldusvahemike võrdlemine alati kuigivõrd informatiivne.

Ka nende parameetrite väärtuste puhul, mil Little'i test päris täpselt ei töötanud, jäi 1. liiki vea tegemise tõenäosus reeglina siiski enam-vähem mõistlikesse piiridesse. Sellesama punktihinnangu keskmine erinevus ideaalsest nivoost 0,05 üle kõigi 36 parameetrite kombinatsiooni oli 0,0286. Samas t-testide testil oli see näitaja 0,1137 ning MM-testil veel märksa suurem.

Siiski ei möödunud nullhüpoteesi simulatsioonid Little'i testi jaoks probleemideta. Suurimaks probleemiks osutusid parameetrite väärtuste kombinatsioonid (30, 16, 0,1) ning (30, 16, 0,3). Nende puhul ei töötanud test üldse, vaid andis veateate, et singulaarse maatriksi pöördmaatriksit ei saa leida. Tegelikult ei tekkinud neil juhtudel ühtegi puudumismustrit, kus oleks olnud 2 või enam objekti, ning kuna kõik mustrid, kus on vaid 1 objekt, jätab algoritm vaatluse alt välja, ei jäänudki andmeid alles, mille põhjal teststatistiku väärtust arvutada. Teisisõnu ei pruugi Little'i testi kasutada saada, kui andmestiku ridade arv ei ole piisavalt palju suurem tema veergude arvust.

Ülejäänud parameetrite kombinatsioonidest leidusid veel 2, mille korral tegi Little'i test 1. liiki viga lubatust väga palju rohkematel kordadel. Nimelt (100, 16, 0,3) ning (500, 16, 0,3) puhul tehti 1. liiki viga vastavalt 73% ja 18% simulatsioonidest. Tasub tähele panna, et kõige suurema ridade arvu puhul, teisisõnu parameetritega (2000, 16, 0,3), tehti 1. liiki viga vaid 5,9% simulatsioonidest ehk ridade arvu kasvades või veergude arvu vähenedes paistab Little'i test õigemini töötavat.

Little'i test töötas nullhüpteesi simulatsioonides enam-vähem sama ajaga kui t-testide test ja palju kiiremini kui MM-test. Nii Little'i testi kui ka t-testide testi mediaanajaks üle kõigi parameetrite väärtuste kombinatsioonide kujunes umbes 45 sekundit 1000 simulatsiooni kohta. Kõige suuremate andmestike korral töötas Little'i test eriti aeglaselt, mistõttu on tema tööaegade aritmeetiline keskmine 1135 sekundit, kuid t-testide testil vaid 319 sekundit 1000 simulatsiooni kohta.

Kõige suuremate parameetrite vääruste korral (2000, 16, 0,3) kulus Little'i testil 25 sekundit iga simulatsiooni jaoks, t-testide testil aga üksnes 2,4 sekundit. Kuna andmestikud võivad olla ka väga palju suuremad kui 2000-realised, tuleks enne säärasele andmestikule testi valimist täpsemalt uurida, kuidas Little'i testi tööaeg ridade arvu kasvades suureneb. On vägagi võimalik, et märksa lihtsam t-testide test lõpetaks oma töö palju kiiremini.

Sisuka hüpoteesi kontrollimine

Sisuka hüpoteesi simulatsioonides sõltusid testide tulemused andmestike parameetrite väärtustest rohkem kui nullhüpoteesi simulatsioonides.

Tabel 1. Little'i testi sisuka hüpoteesi simulatsioonide tulemused.

n	*m*	*v*	*n_L*	*aeg_L*	*min_L*	*mean_L*	*max_L*
30	3	1	1000000	3	0,4059	**0,4069**	0,4078
30	3	2	1000000	3	0,2318	**0,2327**	0,2335
30	10	1					
30	10	2	100000	26	0,0632	**0,0647**	0,0663
100	3	1	1000000	4	0,9499	**0,9503**	0,9508
100	3	2	1000000	5	0,7186	**0,7195**	0,7204
100	10	1	50000	48	0,1096	**0,1124**	0,1151
100	10	2	50000	57	0,074	**0,0763**	0,0786
500	3	1	250000	10	1	**1**	1
500	3	2	250000	14	1	**1**	1
500	10	1	10000	274	0,5944	**0,604**	0,6136
500	10	2	10000	288	0,282	**0,2909**	0,2998
2000	3	1	100000	40	1	**1**	1
2000	3	2	100000	55	1	**1**	1
2000	10	1	10000	1309	1	**1**	1
2000	10	2	10000	1159	0,9728	**0,9758**	0,9788

n – objektide arv ühes simulatsioonis
m – tunnuste arv igal objektil
v – mitme esimese tunnuse väärtuseid kasutati kolmanda tunnuse väärtuse puudumise tõenäosuse määramiseks igal objektil
n_L – testi vastava parameetrite kombinatsiooni korral kasutatud simulatsioonide arv
aeg_L – testi vastava parameetrite kombinatsiooni korral keskmiselt tuhandeks simulatsiooniks kulunud aeg sekundites

min_L – testi vastava parameetrite kombinatsiooni korral sisuka hüpoteesi vastu võtmise tõenäosuse 95-protsendise usaldusvahemiku alampiir
max_L – testi vastava parameetrite kombinatsiooni korral sisuka hüpoteesi vastu võtmise tõenäosuse 95-protsendise usaldusvahemiku ülempiir
mean_L – testi vastava parameetrite kombinatsiooni korral sisuka hüpoteesi vastu võtmise tõenäosuse punktihinnang

Esimesena tasub tähele panna, et ühe parameetrite kombinatsiooni korral 16-st Little'i test taas ei töötanud. Variandid, kus $n = 30$ ning $m = 10$, on need, kus suhe $m{:}\,n$ on kõige suurem. Nagu juba nullhüpoteesi simulatsioonide tulemusi vaadates näha oli, võib Little'i test mitte töötada, kui ridade arv *n* ei ole märgatavalt suurem veergude arvust *m*.

Praegu õnnestus test parameetrite kombinatsiooni (30, 10, 2) korral kõigil 100000 korral, sest võrreldes (30, 10, 1) juhuga oli seal 3. tunnusel mõnevõrra väiksem puudumise tõenäosus ehk kokkuvõttes oli suurem tõenäosus, et andmestikus leidub mingisugune muster (täpsemini see, kus ühegi tunnuse väärtus ei puudu), mida esineb vähemalt kahel korral.

Järgmisena tasub tähele panna, et testi võimsus kasvab ridade arvu *n* kasvades eranditult kõigi nelja parameetrite *m* ja *v* kombinatsiooni korral. Ka see on mõistetav, sest mida rohkem on andmeid, seda väiksem roll jääb testimisel juhuslikkuse kanda. Sama ei kehti antud töös tunnuste arvu kasvades, sest tunnuste 4 kuni 10 puudumise tõenäosused said genereeritud kõigest sõltumatult ehk neist võib mõelda kui müratunnustest. Vaadates testide tulemusi, kus veergude arv $m = 10$, on näha, kui tihti leiab test andmestikust üles struktuuri MAR tingimuse (otsustab sisuka hüpoteesi kasuks), kui see kehtib vaid ühel tunnusel paljudest.

Kolmandaks tasub tähele panna, et test töötab $v = 1$ korral paremini (mittehalvemini) kui $v = 2$ korral iga parameetrite *n* ja *m* kombinatsiooni korral. Teisisõnu: kui tunnuse puudumise tõenäosus sõltub vähemast arvust parameetritest, peaks Little'i test suurema tõenäosusega õigesti sisuka hüpoteesi kasuks otsustama.

Viiel kõige sobivamal juhul 16-st võttis test eksimatult igas simulatsioonis vastu alternatiivse hüpoteesi. Tõsi küll, t-testide test sai sellega hakkama koguni kuue erineva parameetrite kombinatsiooni puhul. Kui eeldada, et andmestikke, kus puudumise struktuur on MAR, on ligikaudu sama palju kui andmestikke, kus see struktuur on MCAR, ja kui eeldada, et ülejäänud antud töös vaadeldavad parameetrid esinevad andmestikes võrdsete tõenäosustega, saab leida Little'i testi puhul tõenäosused *P_L*, et juhul, kui võeti vastu sisukas hüpotees, oli andmestikus päriselt ka puudumise struktuur MAR.

Tabel 2. Little'i testi ülal mainitud tingimustel sisuka hüpoteesi õigesti vastu võtmise tõenäosused.

n	***m***	***P_L***
30	3	0,885
100	3	0,947
500	3	0,953
2000	3	0,953

Need tõenäosused sai leida vaid veergude arvu $m = 3$ korral, sest suuremate tunnuste arvude puhul ei ühti antud töös sisuka hüpoteesi ja nullhüpoteesi korral kasutatud tunnuste arvud. Lisaks on need tõenäosused veidi eksitavad, sest just väikeste tunnuste arvude korral töötas Little'i test kõige õigemini. Seetõttu võib selle tabeli tulba *P_L* väärtustest mõelda kui vastava ridade arvu *n* korral maksimaalsetest või ideaalsetest õigesti sisuka hüpoteesi vastu võtmise tõenäosustest.

4.2 T-testide test

Täpsed R-i koodid, millega antud töös kõik t-testide testi simulatsioonid sooritati, leiab lisadest numbritega 1-3. Kõigi testide simulatsioonide tulemused leiab tabelite kujul lisadest 8-9.

Nullhüpoteesi kontrollimine

Nagu juba eelmises peatükis selgus, ei ole Little'i pakutud test t-testide testist kiirem. Pigem vastupidi, sest andmestike mõõtmete suurenedes hakkab Little'i test veel aeglasemalt töötama kui t-testide test. Antud töö tulemuste põhjal ei saa kindlasti väita, et Little'i test t-testide testist kiiremini töötaks.

Esimene t-testide testi eelis võrreldes kahe ülejäänud testiga on tema töökindlus: ei tule sooritada ühtegi tehet, mis alati võimalik ei ole. Küll aga võib juhtuda, et pole võimalik sooritada mõnd üksikut t-testi, sest üks kahest võrreldavast grupist sisaldab vähem kui 2 elementi. Sellisel juhul jäetakse see t-test lihtsalt tegemata ning minnakse järgmise juurde.

Hirm, et juhul, kui selliseid t-teste liiga palju on, jääb vähem võimalusi sisuka hüpoteesi kasuks otsustada, mistõttu jäädakse liiga tihti nullhüpoteesi juurde, ei ole põhjendatud, sest kõigi parameetrite kombinatsioonide korral, kus märgatavas koguses t-teste sooritamata jäi, tehti 1. liiki viga liiga palju kordi, mitte liiga vähe.

On võimalik, et t-testide testi suurim probleem on tema suur ebatäpsus väikeste andmestike korral. Nullhüpoteesi simulatsioonides kõikusid tema 1. liiki vea tegemise tõenäosused ridade arvu $n = 30$ korral 10 protsendist 87 protsendini. Kui jätta kõrvale 2 parameetrite väärtuste kombinatsiooni, kus Little'i test nullhüpoteesi olukorras üldse ei töötanud, eksis t-testide test kõigi 30-realiste andmestike korral Little'i testist vähemalt kaks korda rohkematel kordadel. See tekitab idee väikeste andmestike korral Little'i testi eelistada ning kui Little'i test peaks mitte töötama, saab tagavaravõimalusena t-testide testi poole pöörduda.

Olukord on mõnevõrra parem, kui uurida 100-realisi andmestikke. Olenemata tunnuste arvust *m*, jäi parajasti puudumise tõenäosuse $p = 0{,}3$ korral t-testide testi 1. liiki vea tegemise tõenäosus 0,04 ja 0,06 vahele. See on seletatav sellega, et t-testid ei tööta soovitud täpsusega, kui vähemalt üks kahest võrreldavast vektorist sisaldab väikest arvu elemente. Kui puudumise

tõenäosus p on 0,3, on väiksemate elementide arvudega vektorid keskmiselt $100 \cdot 0{,}3 \cdot 0{,}7 = 21$ elementi sisaldavad, $p = 0{,}1$ korral aga ainult $100 \cdot 0{,}1 \cdot 0{,}9 = 9$ ning $p = 0{,}04$ korral keskmiselt üksnes 3,84 elementi sisaldavad. Võrdluseks: Little'i testil jäi 100-realiste andmestike korral kuuel juhul 1. liiki vea tegemise tõenäosus 0,04 ja 0,06 vahele ehk kaks korda rohkematel juhtudel. Samuti olid $n = 100$ korral Little'i testil kõrvalekalded ideaalsest nivoost 0,05 märksa väiksemad kui t-testide testil.

Liikudes edasi 500 ja 2000 rea pikkuste andmestike juurde, on aga raske leida ühtegi põhjust, miks mitte kasutada just t-testide testi. Neist $36/2 = 18$ parameetrite kombinatsioonist leiduvad 3, mille korral ei jää Little'i testi 1. liiki vea tegemise punktihinnang 0,045 ja 0,055 vahele. Samas on see näitaja t-testide testi puhul vaid 1. Nende 18 variandi puhul tuli Little'i testi keskmine kõrvalekalle nivoost 0,05 umbes 0,0088, t-testide testil aga üksnes 0,0015, mis on pea 6 korda väiksem. 500 ja 2000 rea pikkuste andmestike korral tuli ka keskmine tööaeg t-testide testil märgatavalt lühem kui Little'i testil.

Sisuka hüpoteesi kontrollimine

Sisuka hüpoteesi simulatsioonid olid t-testide testi kõige tugevam külg. Jättes kõrvale parameetrite kombinatsiooni $(n = 30, m = 10, v = 1)$, mille puhul Little'i test üldse ei töötanud, tuleb Little'i testi keskmiseks võimsuseks umbes 63%, t-testide testil aga koguni 80%. Neist $16 - 1 = 15$ variandist kahel on Little'i testi võimsus veidi suurem, üheksal variandil on võimasus suurem t-testide testil. Ülejäänud neljal juhul suutsid mõlemad testid eksimatult igas simulatsioonis sisuka hüpoteesi kasuks otsustada.

Kõige suuremaks erinevuseks t-testide testi ning Little'i testi vahel sisuka hüpoteesi simulatsioonides kujunes programmide tööks kulunud aeg. Little'i testi kasutades pole parata: tuleb alati leida kõik liidetavad, millest statistik d^2 koosneb. Neist igaüks vajab muuhulgas ka kovariatsioonimaatriksi hindamist ning selle pöördmaatriksi arvutamist.

Isegi kui pärast mingit arvu liitmisi summa vajalikust hii-ruut jaotuse kvantiilist suuremaks saab, ei saa arvutusi lõpetada ja alternatiivset hüpoteesi vastu võtta, sest liidetavad võivad ka negatiivsed olla ehk summa võib ka väheneda. Sama ei kehti t-testide testi korral: niipea kui üks t-test statistilise erinevuse tuvastab, saab terve testi ära lõpetada ning sisuka hüpoteesi vastu võtta. Eelmainitud 15 parameetrite kombinatsiooni korral tuli Little'i testi keskmine tööaeg 220 sekundit 1000 simulatsiooni kohta, t-testide testil aga ainult 15.

Kui eeldada, et andmestikke, kus puudumise struktuur on MAR, on ligikaudu sama palju kui andmestikke, kus selleks struktuuriks on MCAR, ja kui eeldada, et ülejäänud antud töös vaadeldavad parameetrid esinevad andmestikes võrdsete tõenäosustega, saab ka t-testide testi puhul leida tõenäosused *P_T*, et juhul, kui võeti vastu sisukas hüpotees, oli andmestikus päriselt ka puudumise struktuur MAR.

Tabel 3. Little'i testi ja t-testide testi ülal mainitud tingimustel sisuka hüpoteesi õigesti vastu võtmise tõenäosused.

n	*m*	*P_L*	*P_T*
30	3	0,885	0,785
100	3	0,947	0,905
500	3	0,953	0,953
2000	3	0,953	0,954

Ka siit tabelist on näha, et objektide arvu $n = 100$ korral võib veel tasuda Little'i testi eelistada t-testide testile. Need tõenäosused sai leida vaid veergude arvu $m = 3$ korral, sest suuremate tunnuste arvude puhul ei ühti antud töös sisuka hüpoteesi ja nullhüpoteesi korral kasutatud tunnuste arvud. Lisaks on need tõenäosused veidi eksitavad, sest just väikeste tunnuste arvude korral töötasid mõlemad testid kõige õigemini. Seetõttu võib selle tabeli tulpade *P_L* ja *P_T* väärtustest mõelda kui vastava ridade arvu *n* korral maksimaalsetest või ideaalsetest õigesti sisuka hüpoteesi vastu võtmise tõenäosustest.

4.3 MM-test

Täpsed R-i koodid, millega antud töös kõik MM-testi simulatsioonid sooritati, leiab lisast number 7. Kõigi testide simulatsioonide tulemused leiab tabelite kujul lisadest 8-9.

Kolmest uuritud testist töötas MM-test kindlasti kõige halvemini. Esimene probleem, mis MM-testi puhul silma paistab, on tema madal töökindlus. Kui Little'i test ei töötanud kahe parameetrite kombinatsiooni korral 36-st, siis MM-test ei töötanud neist koguni kuuel juhul. Iseenesest tunduvad need 6 parameetrite kombinatsiooni olevat just sellised, kus erinevate puudumismustrite arv tuleb suur ja seega igat üksikut mustrit ennast esineb vähe. Samade andmestikega oli ka Little'i testil probleeme, kuid MM-testil tulid need probleemid veelgi teravamalt välja.

Teine suur MM-testi probleem seisneb testi pikkades tööaegades. Suurimad parameetrid, mille kombinatsiooni korral MM-testi testiti, olid nullhüpoteesi olukorras (500, 6, 0,3). Iga simulatsioon võttis selle parameetrite kombinatsiooni korral umbes 7,2 sekundit aega. Võrdluseks: samades tingimustes võttis ühe Little'i testi üks simulatsioon vaid 0,21 sekundit ning t-testide testi üks simulatsioon vaid 0,047 sekundit aega. Teisisõnu ei sooritatud antud töös MM-teste parajasti neil andmestikel, mille ridade arv korda veergude arv oli 6000 või rohkem, sest need oleksid liigselt aega nõudnud.

Kolmas ja võimalik, et kõige tõsisem MM-testi probleem on tema olematu täpsus. Nullhüpoteesi tingimustes testiti MM-testi 18 parameetrite kombinatsiooni korral. Neist vaid ühel jäi testi 1. liiki vea tegemise tõenäosuse punktihinnang 0,045 ja 0,055 vahele. Keskmine

erinevus ideaalnivoost 0,05 oli 0,22. Erinevalt kahest eelmisest vaadeldud testist ei paranenud MM-testi täpsus ka suuremate andmestike korral: 500-realiste andmestike korral tehti 1. liiki viga erinevate parameetrite *m* ja *p* väärtuste kombinatsioonide korral 10%, 11%, 5%, 21%, 42% ja 58% simulatsioonidest.

Kuna niisugune ebatäpsus on täiesti vastuvõetamatu, polnud vajadust MM-testi sisuka hüpoteesi tingimustes kuigi põhjalikult testida. Vähem simulatsioone sooritades sai paljude parameetrite väärtuste kombinatsioonide korral MM-testi sisuka hüpoteesi tingimustes siiski testitud. Kolmest testist jäi MM-testi võimsus enamus juhtudel kõige madalamaks. Ka ühel õnnelikul juhul, kus MM-testi võimsus kõige suurem oli, ei tähendaks testi vastu võetud alternatiivne hüpotees suurt midagi, kuna tõenäosus 1. liiki viga teha oleks ikkagi liiga suur.

Järgmisena sai proovitud minimaalse mustrigrupi suurust 6 varasema kahe asemel: sellel oli testile nii häid kui halbu tagajärgi. Kuna sedasi jäeti kõrvale palju rohkem andmeridasid, pidi andmestik veelgi suurem olema, et test üleüldse töötaks. Ühegi 30 ega 100 rea pikkuse andmestiku korral selline MM-test ei töötanud. Kuue parameetrite väärtuste kombinatsiooni korral, mille puhul simulatsioonid MM-testi selle variandiga läbi viidi, jäi testi 1. liiki vea tegemise tõenäosus 4% ning 9% vahele, mis on märksa parem kui 4% ja 59% vahele jäämine nagu algse MM-testi ning samade parameetrite väärtuste kombinatsioonide puhul.

Isegi kui mitte hoolida väiksemate andmestike puhul mittetöötamisest, ei ole minimaalse mustrigrupi suurusega 6 MM-test esialgsest eriti kasulikum, sest sellise testi võimsus on väga madal. Kõigi erinevate sisuka hüpoteesi olukordade 500-realiste andmestike pealt suutis see MM-test võtta sisuka hüpoteesi vastu keskmiselt umbes 24 protsendil simulatsioonidest. Võrdluseks: samade parameetrite väärtuste kombinatsioonide korral võttis t-testide test sisukat hüpoteesi vastu keskmiselt 99,7 protsendil simulatsioonidest.

Kokkuvõte

Antud töös võrreldi kolme testi. Neist MM-test töötas ülejäänud kahe testiga võrreldes äärmiselt kehvasti: tegi liiga tihti 1. ja 2. liiki vigu, töötas kõige aeglasemalt ja omas kõige suuremat tõenäosust veateadet anda ehk üldse mitte otsustada suuta.

Kahest ülejäänud testist töötas Little'i test paremini väiksemate, ülimalt 100 reaga ja t-testide test suuremate, 500 ja enama reaga andmestike korral. Andmestike ridade arvu kasvades lähenesid mõlema testi võimsused ühele. Samuti töötasid mõlemad testid õigemini väiksemate tunnuste arvude korral.

Väiksemate andmestike korral tekkis t-testide testil probleeme. Kuna see test tegi väikeste andmestike korral 1. liiki viga liiga tihti, oli t-testide testil siis Little'i testist väiksem tõenäosus õigesti sisukat hüpoteesi vastu võtta. Paraku polnud ka Little'i test väiksemate andmestike korral veatu: kui tunnuste arv objektidel oli liiga suur, ei pruukinud Little'i test üldse töötada, vaid võis anda veateate.

Samuti jäi Little'i testi võimsus t-testide testi omale alla ning ka Little'i testi tööajad olid pikemad. Tööaeg pole küll väikeste andmestike korral probleemne ning siiski soovitatakse antud töös väiksemate andmestike peal struktuuri MCAR testimiseks Little'i testi ning suuremate andmestike peal t-testide testi kasutada.

Nii t-testide test kui ka Little'i test toimisid kokkuvõtteks siiski piisavalt hästi ja täpselt. Kui nende vahelt valida tuleks, oleks t-testide test vahest paremgi valik, sest see test töötas suuremate andmestike korral igati paremini: kiiremini ja õigemini nii nullhüpoteesi kui ka sisuka hüpoteesi tingimustes.

Kasutatud kirjandus

1. Allwood, M 2008, *The Satterthwaite Formula for Degrees of Freedom in the Two-Sample t-Test*. Kättesaadav: <http://apcentral.collegeboard.com/apc/public/repository/ap05_stats_allwood_fin4prod.pdf>. [22.04.2015].
2. Enders, CK 2010, *Applied Missing Data Analysis*. Spring Street, New York.
3. Jamshidian, M, Jalal, S & Jansen, C 2014, „MissMech: An R Package for Testing Homoscedasticity, Multivariate Normality, and Missing Completely at Random (MCAR)“, *Journal of Statistical Software*, vol. 56.
4. Little, RJA 1988, „A Test of Missing Completely at Random for Multivariate Data With Missing Values“, *Journal of the American Statistical Assosication*, vol. 83, 1198-1202.
5. Manly, BFJ 1994, *Multivariate Statistical Methods: A Primer, Second Edition*. Boca Raton, Florida.
6. Rubin, DB 1976, „Inference and Missing Data“, *Biometrika*, vol. 63, 581-592.

Lisad

Lisa 1. Kasutatud t-testide testi kood nullhüpoteesi olukorra simuleerimiseks, kirjutatud R-is.

```
aeg = proc.time()
mitu = 100000 #simulatsioonide arv
n = 30 #objektide arv
m = 3 #tunnuste arv igal objektil
puuduvad = 0.1 #mis tn-sega vaatlused puuduvad
loendur = 0
eitoimu = 0
jagatis = 1 / 20 / m / (m - 1) #et seda iga kord arvutama ei peaks

for (tsm in 1:mitu) {
	a = matrix(rnorm(n * m, 100, 15), nrow = n, ncol = m)
	missing = matrix(runif(n * m), nrow = n, ncol = m) < puuduvad
	a[missing] = NA

	eija = 0 #kas andmetes leiduvad 2 seotud tunnust?
	for (i in 1:m) { #siit vaatan, kas vaatlus puudub või ei
		for (j in 1:m) { #neid võrdlen vastavalt sellele omavahel
			if (i != j) {
				on = c()
				ei = c()
				for (k in 1:n) {
					if (is.na(a[k, i])) {
						ei = c(ei, a[k, j])
					}
					else {
						on = c(on, a[k, j])
					}
				}
				on = na.omit(on)
				ei = na.omit(ei)
				if ((length(on) > 1) & (length(ei) > 1)) {
					if (t.test(on, ei)$p.value < jagatis) {
						eija = 1 #leiduvad
						loendur = loendur + 1
						break
					}
				}
				else { #mitu võrdlemist ära jäid? halb, kui jäi
					eitoimu = eitoimu + 1
				}
			}
		}
		if (eija == 1) {
			break
```

```
                }
        }
}

print(paste(100 * loendur / mitu, "protsenti"))
p = loendur / mitu
arv = qt(0.975, mitu - 1) #t-jaotuse kvantiil, vabadusastmed
print(p - arv * sqrt(p * (1 - p) / mitu))
print(p + arv * sqrt(p * (1 - p) / mitu))
proc.time() - aeg
print(eitoimu)
```

Lisa 2. Kasutatud t-testide testi kood lihtsama sisuka hüpoteesi olukorra simuleerimiseks, kirjutatud R-is.

```
aeg = proc.time()
mitu = 100000 #simulatsioonide arv
n = 30 #objektide arv
m = 3 #tunnuste arv igal objektil
puuduvad = 0.15 #mis tn-sega vaatlused puuduvad
loendur = 0
eitoimu = 0 #mitu simulatsiooni ära jäid
jagatis = 1 / 20 / (m - 1) / (m - 2) #et seda iga kord arvutama ei peaks

for (tsm in 1:mitu) {
        a = matrix(rnorm(n * m, 100, 15), nrow = n, ncol = m)
        missing = matrix(runif(n * m), nrow = n, ncol = m) < puuduvad
        for (i in 1:n) {
                arv = runif(1)
                if (a[i, 1] < 100) {
                        if (arv < 0.05) {
                                a[i, 3] = NA
                        }
                }
                else {
                        if (arv < 0.45) {
                                a[i, 3] = NA
                        }
                }
        }
        if (m > 3) {
                a[1:n, 4:m][missing[1:n, 4:m]] = NA
                #esimese kahe tunnuse väärtused ei puudu eal
        }

        eija = 0 #kas andmetes leiduvad 2 seotud tunnust?
        for (i in 3:m) { #siit vaatan, kas vaatlus puudub või ei
                for (j in 1:m) { #neid võrdlen vastavalt sellele omavahel
                        if (i != j) {
```

```
                        on = c()
                        ei = c()
                        for (k in 1:n) {
                                if (is.na(a[k, i])) {
                                        ei = c(ei, a[k, j])
                                }
                                else {
                                        on = c(on, a[k, j])
                                }
                        }
                        on = na.omit(on)
                        ei = na.omit(ei)
                        if ((length(on) > 1) & (length(ei) > 1)) {
                                if (t.test(on, ei)$p.value < jagatis) {
                                        eija = 1 #jah, andmetes leiduvad 2 seotud tunnust
                                        loendur = loendur + 1
                                        break
                                }
                        }
                        else { #mitu võrdlemist ära jäid? halb, kui jäi
                                eitoimu = eitoimu + 1
                        }
                }
        }
        if (eija == 1) {
                break
        }
    }
}

print(paste(100 * loendur / mitu, "protsenti"))
p = loendur / mitu
arv = qt(0.975, mitu - 1) #t-jaotuse kvantiil, vabadusastmed
print(p - arv * sqrt(p * (1 - p) / mitu))
print(p + arv * sqrt(p * (1 - p) / mitu))
proc.time() - aeg
print(eitoimu)
```

Lisa 3. Kasutatud t-testide testi kood keerukama sisuka hüpoteesi olukorra simuleerimiseks, kirjutatud R-is.

```
aeg = proc.time()
mitu = 100000 #simulatsioonide arv
n = 30 #objektide arv
m = 3 #tunnuste arv igal objektil
puuduvad = 0.15 #mis tn-sega vaatlused puuduvad
loendur = 0
eitoimu = 0 #mitu simulatsiooni ära jäid
jagatis = 1 / 20 / (m - 1) / (m - 2) #et seda iga kord arvutama ei peaks
```

```
for (tsm in 1:mitu) {
	a = matrix(rnorm(n * m, 100, 15), nrow = n, ncol = m)
	missing = matrix(runif(n * m), nrow = n, ncol = m) < puuduvad
	for (i in 1:n) {
		arv = runif(1)
		tehtud = 0
		if ((a[i, 1] < 100) & (a[i, 2] > 100)) {
			tehtud = 1
			if (arv < 0.05) {
				a[i, 3] = NA
			}
		}
		if ((a[i, 1] > 100) & (a[i, 2] < 100)) {
			tehtud = 1
			if (arv < 0.45) {
				a[i, 3] = NA
			}
		}
		if (tehtud == 0) {
			if (arv < puuduvad) {
				a[i, 3] = NA
			}
		}
	}
	if (m > 3) {
		a[1:n, 4:m][missing[1:n, 4:m]] = NA
		#esimese kahe tunnuse väärtused ei puudu eal
	}

	eija = 0 #kas andmetes leiduvad 2 seotud tunnust?
	for (i in 3:m) { #siit vaatan, kas vaatlus puudub või ei
		for (j in 1:m) { #neid võrdlen vastavalt sellele omavahel
			if (i != j) {
				on = c()
				ei = c()
				for (k in 1:n) {
					if (is.na(a[k, i])) {
						ei = c(ei, a[k, j])
					}
					else {
						on = c(on, a[k, j])
					}
				}
				on = na.omit(on)
				ei = na.omit(ei)
				if ((length(on) > 1) & (length(ei) > 1)) {
					if (t.test(on, ei)$p.value < jagatis) {
						eija = 1 #jah, andmetes leiduvad 2 seotud tunnust
						loendur = loendur + 1
```

```
                                        break
                                }
                        }
                        else { #mitu võrdlemist ära jäid? halb, kui jäi
                                eitoimu = eitoimu + 1
                        }
                }
        }
        if (eija == 1) {
                break
        }
    }
}

print(paste(100 * loendur / mitu, "protsenti"))
p = loendur / mitu
arv = qt(0.975, mitu - 1) #t-jaotuse kvantiil, vabadusastmed
print(p - arv * sqrt(p * (1 - p) / mitu))
print(p + arv * sqrt(p * (1 - p) / mitu))
proc.time() - aeg
print(eitoimu)
```

Lisa 4. Kasutatud Little'i testi kood nullhüpoteesi olukorra simuleerimiseks, kirjutatud R-is.

```
library(MissMech)
library(norm)

aeg = proc.time()
mitu = 100000 #simulatsioonide arv
n = 30 #objektide arv
m = 3 #tunnuste arv igal objektil
puuduvad = 0.1 #mis tn-sega vaatlused puuduvad
loendur = 0
eitoimu = 0 #mitu simulatsiooni ära jäid

for (k in 1:mitu) {
        a = matrix(rnorm(n * m, 100, 15), nrow = n, ncol = m)
        missing = matrix(runif(n * m), nrow = n, ncol = m) < puuduvad
        a[missing] = NA
        order = OrderMissing(a, del.lesscases = 1) #del.lesscases on vaikimisi 0
        J = order$g #erinevate puudumismustrite arv
        if (J > 1) { #kui puuduvaid andmeid andmestikus pole, jääb simulatsioon ära
                summa = 0
                vana = 1
                pj = 0
                for (j in 1:J) {
                        eija = is.na(order$patused[j,]) == F #NA -> F ning 1 -> T selle reaga
                        if (sum(eija == T) > 0) { #objektid, kellel on ainult NA-d, jätan välja
```

```
                        pj = pj + sum(eija == T) #et vabadusastmete arv leida
                        y = c(rep(0, sum(eija == T))) #keskväärtused j. mustris
                        muu = c(rep(0, sum(eija == T))) #kõigi vaatluste keskväärtused
                        tsm = 1
                        for (i in 1:m) {
                                if (eija[i] == T) {
                                        y[tsm] =
                                        mean(order$data[vana:order$spatcnt[j], i])
                                        muu[tsm] = mean(a[, i], na.rm = T)
                                        tsm = tsm + 1
                                }
                        }
                        abi = prelim.norm(a[, eija == T])
                        vastus = em.norm(abi, showits = F, criterion = 0.0001)
                        nii = getparam.norm(abi, vastus, corr = T)
                        b = nii$r #korrelatsioonid
                        for (i in 1:dim(b)[1]) { #lisan dispersioonid
                                b[i, i] = nii$sdv[i]^2
                        }
                        b = n / (n - 1) * b #nüüd on nihketa hinnang
                        summa = summa + order$patcnt[j] *
                        t(y - muu) %*% solve(b) %*% (y - muu)
                }
                vana = order$spatcnt[j] + 1
        }
        if (summa > qchisq(0.95, df = (pj - m))) {
                loendur = loendur + 1
        }
    }
    else {
        eitoimu = eitoimu + 1
    }
}

print(paste(100 * loendur / (mitu - eitoimu), "protsenti"))
p = loendur / (mitu - eitoimu)
arv = qt(0.975, mitu - eitoimu - 1) #t-jaotuse kvantiil, vabadusastmed
print(p - arv * sqrt(p * (1 - p) / (mitu - eitoimu)))
print(p + arv * sqrt(p * (1 - p) / (mitu - eitoimu)))
proc.time() - aeg
```

Lisa 5. Kasutatud Little'i testi kood lihtsama sisuka hüpoteesi olukorra simuleerimiseks, kirjutatud R-is.

```
library(MissMech)
library(norm)

aeg = proc.time()
mitu = 100000 #simulatsioonide arv
n = 30 #objektide arv
```

```
m = 3 #tunnuste arv igal objektil
puuduvad = 0.15 #mis tn-sega vaatlused puuduvad
loendur = 0
eitoimu = 0 #mitu simulatsiooni ära jäid

for (k in 1:mitu) {
	a = matrix(rnorm(n * m, 100, 15), nrow = n, ncol = m)
	missing = matrix(runif(n * m), nrow = n, ncol = m) < puuduvad
	for (i in 1:n) {
		arv = runif(1)
		if (a[i, 1] < 100) {
			if (arv < 0.05) {
				a[i, 3] = NA
			}
		}
		else {
			if (arv < 0.45) {
				a[i, 3] = NA
			}
		}
	}
	if (m > 3) {
		a[1:n, 4:m][missing[1:n, 4:m]] = NA
		#esimese kahe tunnuse väärtused ei puudu eal
	}
	order = OrderMissing(a, del.lesscases = 1) #del.lesscases on vaikimisi 0
	J = order$g #erinevate puudumismustrite arv
	if (J > 1) { #kui puuduvaid andmeid andmestikus pole, jääb simulatsioon ära
		summa = 0
		vana = 1
		pj = 0
		for (j in 1:J) {
			eija = is.na(order$patused[j,]) == F #NA -> F ning 1 -> T selle reaga
			if (sum(eija == T) > 0) { #objektid, kellel on ainult NA-d, jätan välja
				pj = pj + sum(eija == T) #et vabadusastmete arv leida
				y = c(rep(0, sum(eija == T))) #keskväärtused j. mustris
				muu = c(rep(0, sum(eija == T))) #kõigi vaatluste keskväärtused
				tsm = 1
				for (i in 1:m) {
					if (eija[i] == T) {
						y[tsm] =
						mean(order$data[vana:order$spatcnt[j], i])
						muu[tsm] = mean(a[, i], na.rm = T)
						tsm = tsm + 1
					}
				}
				abi = prelim.norm(a[, eija == T])
				vastus = em.norm(abi, showits = F, criterion = 0.0001)
				nii = getparam.norm(abi, vastus, corr = T)
				b = nii$r #korrelatsioonid
```

```
                        for (i in 1:dim(b)[1]) { #lisan dispersioonid
                                b[i, i] = nii$sdv[i]^2
                        }
                        b = n / (n - 1) * b #nüüd on nihketa hinnang
                        summa = summa + order$patcnt[j] *
                        t(y - muu) %*% solve(b) %*% (y - muu)
                }
                vana = order$spatcnt[j] + 1
        }
        if (summa > qchisq(0.95, df = (pj - m))) {
                loendur = loendur + 1
        }
    }
    else {
        eitoimu = eitoimu + 1
    }
}

print(paste(100 * loendur / (mitu - eitoimu), "protsenti"))
p = loendur / (mitu - eitoimu)
arv = qt(0.975, mitu - eitoimu - 1) #t-jaotuse kvantiil, vabadusastmed
print(p - arv * sqrt(p * (1 - p) / (mitu - eitoimu)))
print(p + arv * sqrt(p * (1 - p) / (mitu - eitoimu)))
proc.time() - aeg
```

Lisa 6. Kasutatud Little'i testi kood keerukama sisuka hüpoteesi olukorra simuleerimiseks, kirjutatud R-is.

```
library(MissMech)
library(norm)

aeg = proc.time()
mitu = 100000 #simulatsioonide arv
n = 30 #objektide arv
m = 3 #tunnuste arv igal objektil
puuduvad = 0.15 #mis tn-sega vaatlused puuduvad
loendur = 0
eitoimu = 0 #mitu simulatsiooni ära jäid

for (k in 1:mitu) {
    a = matrix(rnorm(n * m, 100, 15), nrow = n, ncol = m)
    missing = matrix(runif(n * m), nrow = n, ncol = m) < puuduvad
    for (i in 1:n) {
        arv = runif(1)
        tehtud = 0
        if ((a[i, 1] < 100) & (a[i, 2] > 100)) {
            tehtud = 1
            if (arv < 0.05) {
                a[i, 3] = NA
            }
```

```
		}
		if ((a[i, 1] > 100) & (a[i, 2] < 100)) {
			tehtud = 1
			if (arv < 0.45) {
				a[i, 3] = NA
			}
		}
		if (tehtud == 0) {
			if (arv < puuduvad) {
				a[i, 3] = NA
			}
		}
}
if (m > 3) {
		a[1:n, 4:m][missing[1:n, 4:m]] = NA
		#esimese kahe tunnuse väärtused ei puudu eal
}
order = OrderMissing(a, del.lesscases = 1) #del.lesscases on vaikimisi 0
J = order$g #erinevate puudumismustrite arv
if (J > 1) { #kui puuduvaid andmeid andmestikus pole, jääb simulatsioon ära
		summa = 0
		vana = 1
		pj = 0
		for (j in 1:J) {
			eija = is.na(order$patused[j,]) == F #NA -> F ning 1 -> T selle reaga
			if (sum(eija == T) > 0) { #objektid, kellel on ainult NA-d, jätan välja
				pj = pj + sum(eija == T) #et vabadusastmete arv leida
				y = c(rep(0, sum(eija == T))) #keskväärtused j. mustris
				muu = c(rep(0, sum(eija == T))) #kõigi vaatluste keskväärtused
				tsm = 1
				for (i in 1:m) {
					if (eija[i] == T) {
						y[tsm] =
						mean(order$data[vana:order$spatcnt[j], i])
						muu[tsm] = mean(a[, i], na.rm = T)
						tsm = tsm + 1
					}
				}
				abi = prelim.norm(a[, eija == T])
				vastus = em.norm(abi, showits = F, criterion = 0.0001)
				nii = getparam.norm(abi, vastus, corr = T)
				b = nii$r #korrelatsioonid
				for (i in 1:dim(b)[1]) { #lisan dispersioonid
					b[i, i] = nii$sdv[i]^2
				}
				b = n / (n - 1) * b #nüüd on nihketa hinnang
				summa = summa + order$patcnt[j] *
				t(y - muu) %*% solve(b) %*% (y - muu)
			}
			vana = order$spatcnt[j] + 1
```

```
                }
                if (summa > qchisq(0.95, df = (pj - m))) {
                        loendur = loendur + 1
                }
        }
        else {
                eitoimu = eitoimu + 1
        }
}

print(paste(100 * loendur / (mitu - eitoimu), "protsenti"))
p = loendur / (mitu - eitoimu)
arv = qt(0.975, mitu - eitoimu - 1) #t-jaotuse kvantiil, vabadusastmed
print(p - arv * sqrt(p * (1 - p) / (mitu - eitoimu)))
print(p + arv * sqrt(p * (1 - p) / (mitu - eitoimu)))
proc.time() - aeg
```

Lisa 7. Kasutatud MM-testi kood nullhüpoteesi olukorra simuleerimiseks, kirjutatud R-is.

```
library(MissMech)
aeg = proc.time()
mitu = 100000 #simulatsioonide arv
n = 30 #objektide arv
m = 3 #tunnuste arv igal objektil
puuduvad = 0.1 #mis tn-sega vaatlused puuduvad
loendur = 0
eitoimu = 0 #mitu simulatsiooni ära jäid

for (k in 1:mitu) {
        a = matrix(rnorm(n * m, 100, 15), nrow = n, ncol = m)
        missing = matrix(runif(n * m), nrow = n, ncol = m) < puuduvad
        a[missing] = NA
        order = OrderMissing(a, del.lesscases = 1)
        if (order$g > 1) { #F-n nõuab, et oleks vahemalt 2 erinevat puudumismustrit.
                if (TestMCARNormality(a, del.lesscases = 1, imputation.method =
                "Normal", seed = NA)$pnormality < 0.05) {
                        loendur = loendur + 1
                }
        }
        else {
                eitoimu = eitoimu + 1
        }
}

print(paste(100 * loendur / (mitu - eitoimu), "protsenti"))
p = loendur / (mitu - eitoimu)
arv = qt(0.975, mitu - eitoimu - 1) #t-jaotuse kvantiil, vabadusastmed
print(p - arv * sqrt(p * (1 - p) / (mitu - eitoimu)))
print(p + arv * sqrt(p * (1 - p) / (mitu - eitoimu)))
proc.time() - aeg
```

Lisa 8. Kolme testi nullhüpoteesi simulatsioonide tulemused.

n	*m*	*p*	*n_L*	*aeg_L*	*min_L*	*mean_L*	*max_L*	*n_T*	*aeg_T*	*min_T*	*mean_T*	*max_T*
30	3	0,04	1000000	2	0,0453	**0,0458**	0,0463	1000000	2	0,1014	**0,102**	0,1026
30	3	0,1	500000	4	0,0428	**0,0434**	0,044	1000000	3	0,1581	**0,1589**	0,1596
30	3	0,3	500000	8	0,0349	**0,0354**	0,0359	1000000	2	0,0714	**0,0719**	0,0724
30	6	0,04	500000	4	0,0473	**0,0479**	0,0485	500000	5	0,2812	**0,2825**	0,2837
30	6	0,1	500000	8	0,0433	**0,0439**	0,0444	500000	6	0,384	**0,3853**	0,3867
30	6	0,3	100000	20	0,0467	**0,0481**	0,0494	500000	8	0,113	**0,1139**	0,1147
30	16	0,04	100000	24	0,0852	**0,087**	0,0888	100000	22	0,7713	**0,7739**	0,7765
30	16	0,1						100000	25	0,8669	**0,869**	0,8711
30	16	0,3						100000	61	0,2459	**0,2486**	0,2512
100	3	0,04	500000	6	0,047	**0,0475**	0,0481	1000000	3	0,1429	**0,1436**	0,1443
100	3	0,1	250000	9	0,0465	**0,0474**	0,0482	1000000	3	0,0641	**0,0646**	0,0651
100	3	0,3	250000	13	0,0445	**0,0453**	0,0461	1000000	3	0,0499	**0,0503**	0,0507
100	6	0,04	250000	14	0,0445	**0,0453**	0,0461	250000	11	0,3691	**0,371**	0,3729
100	6	0,1	250000	22	0,043	**0,0438**	0,0446	250000	14	0,0958	**0,0969**	0,0981
100	6	0,3	100000	64	0,0353	**0,0364**	0,0376	250000	14	0,0505	**0,0513**	0,0522
100	16	0,04	100000	42	0,0478	**0,0492**	0,0505	100000	44	0,8941	**0,896**	0,8978
100	16	0,1	100000	84	0,0723	**0,074**	0,0756	25000	102	0,2224	**0,2276**	0,2328
100	16	0,3	100000	59	0,7259	**0,7323**	0,7388	25000	112	0,0565	**0,0594**	0,0623
500	3	0,04	250000	14	0,0481	**0,049**	0,0498	250000	9	0,0494	**0,0503**	0,0511
500	3	0,1	250000	19	0,0484	**0,0492**	0,0501	100000	11	0,0475	**0,0488**	0,0502
500	3	0,3	100000	23	0,0486	**0,05**	0,0513	250000	10	0,0491	**0,05**	0,0508
500	6	0,04	100000	36	0,0485	**0,0498**	0,0512	100000	52	0,0508	**0,0522**	0,0535
500	6	0,1	100000	70	0,046	**0,0473**	0,0486	100000	51	0,0485	**0,0498**	0,0512
500	6	0,3	100000	208	0,0416	**0,0429**	0,0441	100000	47	0,047	**0,0483**	0,0497
500	16	0,04	100000	321	0,0497	**0,051**	0,0524	10000	412	0,0539	**0,0585**	0,0631
500	16	0,1	10000	797	0,0453	**0,0496**	0,0539	10000	405	0,0435	**0,0477**	0,0519
500	16	0,3	10000	1238	0,1682	**0,1757**	0,1832	10000	370	0,0462	**0,0505**	0,0548
2000	3	0,04	100000	47	0,0486	**0,05**	0,0513	100000	72	0,0471	**0,0485**	0,0498
2000	3	0,1	100000	54	0,0486	**0,0499**	0,0513	100000	69	0,0474	**0,0487**	0,0501
2000	3	0,3	100000	88	0,0476	**0,0489**	0,0502	100000	57	0,0471	**0,0484**	0,0497
2000	6	0,04	25000	203	0,0471	**0,0498**	0,0525	25000	398	0,0462	**0,0488**	0,0515
2000	6	0,1	10000	382	0,0475	**0,0518**	0,0561	25000	366	0,0485	**0,0513**	0,054
2000	6	0,3	10000	994	0,0437	**0,0479**	0,0521	25000	305	0,0462	**0,0488**	0,0515
2000	16	0,04	10000	2151	0,0445	**0,0487**	0,0529	10000	3029	0,0457	**0,05**	0,0543
2000	16	0,1	10000	6610	0,042	**0,0461**	0,0502	10000	2929	0,0447	**0,0489**	0,0531
2000	16	0,3	10000	24952	0,0545	**0,0591**	0,0637	10000	2448	0,0466	**0,0509**	0,0552

n	*m*	*p*	*n_M*	*aeg_M*	*min_M*	*mean_M*	*max_M*
30	3	0,04	25000	866	0,1516	**0,1561**	0,1606
30	3	0,1	10000	446	0,1816	**0,1893**	0,197
30	3	0,3	10000	690	0,1934	**0,2013**	0,2092
30	6	0,04	10000	415	0,2396	**0,2488**	0,258
30	6	0,1	10000	599	0,3161	**0,3253**	0,3344
30	6	0,3					
30	16	0,04					
30	16	0,1					
30	16	0,3					
100	3	0,04	25000	314	0,1532	**0,1578**	0,1623
100	3	0,1	10000	479	0,1153	**0,1217**	0,1281
100	3	0,3	10000	932	0,0713	**0,0765**	0,0817
100	6	0,04	10000	644	0,243	**0,2515**	0,26
100	6	0,1	10000	959	0,2609	**0,2696**	0,2783
100	6	0,3	10000	3663	0,8415	**0,8485**	0,8555
100	16	0,04	10000	1531	0,5787	**0,5883**	0,5979
100	16	0,1					
100	16	0,3					
500	3	0,04	10000	627	0,0986	**0,1046**	0,1106
500	3	0,1	10000	432	0,0998	**0,1058**	0,1118
500	3	0,3	25000	462	0,0425	**0,045**	0,0476
500	6	0,04	10000	1466	0,2017	**0,2097**	0,2177
500	6	0,1	10000	3097	0,4104	**0,4201**	0,4298
500	6	0,3	10000	7157	0,5731	**0,5828**	0,5925

n – objektide arv ühes simulatsioonis
m – tunnuste arv igal objektil
p – iga objekti iga tunnuse konstantne puudumise tõenäosus
n_L, *n_T*, *n_M* – kolme testi vastava parameetrite kombinatsiooni korral kasutatud simulatsioonide arvud
aeg_L, *aeg_T*, *aeg_M* – kolme testi vastava parameetrite kombinatsiooni korral keskmiselt tuhandeks simulatsiooniks kulunud ajad sekundites
min_L, *min_T*, *min_M* – kolme testi vastava parameetrite kombinatsiooni korral sisuka hüpoteesi vastu võtmise tõenäosuste 95-protsendiste usaldusvahemike alampiirid
max_L, *max_T*, *max_M* – kolme testi vastava parameetrite kombinatsiooni korral sisuka hüpoteesi vastu võtmise tõenäosuste 95-protsendiste usaldusvahemike ülempiirid
mean_L, *mean_T*, *mean_M* – kolme testi vastava parameetrite kombinatsiooni korral sisuka hüpoteesi vastu võtmise tõenäosuste punktihinnangud

Lisa 9. Little'i testi ning t-testide testi sisuka hüpoteesi simulatsioonide tulemused.

n	*m*	*v*	*n_L*	*aeg_L*	*min_L*	*mean_L*	*max_L*	*n_T*	*aeg_T*	*min_T*	*mean_T*	*max_T*
30	3	1	1000000	3	0,4059	**0,4069**	0,4078	5000000	1	0,528	**0,5285**	0,5289
30	3	2	1000000	3	0,2318	**0,2327**	0,2335	5000000	1	0,2814	**0,2818**	0,2822
30	10	1						200000	21	0,4958	**0,498**	0,5002
30	10	2	100000	26	0,0632	**0,0647**	0,0663	200000	18	0,4583	**0,4605**	0,4627
100	3	1	1000000	4	0,9499	**0,9503**	0,9508	1000000	1	0,9683	**0,9687**	0,969
100	3	2	1000000	5	0,7186	**0,7195**	0,7204	1000000	2	0,6768	**0,6777**	0,6786
100	10	1	50000	48	0,1096	**0,1124**	0,1151	500000	15	0,7803	**0,7814**	0,7825
100	10	2	50000	57	0,074	**0,0763**	0,0786	500000	31	0,2488	**0,25**	0,2512
500	3	1	250000	10	1	**1**	1	500000	6	1	**1**	1
500	3	2	250000	14	1	**1**	1	500000	8	0,9999	**0,9999**	0,9999
500	10	1	10000	274	0,5944	**0,604**	0,6136	500000	9	1	**1**	1
500	10	2	10000	288	0,282	**0,2909**	0,2998	500000	9	0,9871	**0,9874**	0,9877
2000	3	1	100000	40	1	**1**	1	100000	28	1	**1**	1
2000	3	2	100000	55	1	**1**	1	100000	34	1	**1**	1
2000	10	1	10000	1309	1	**1**	1	100000	29	1	**1**	1
2000	10	2	10000	1159	0,9728	**0,9758**	0,9788	100000	40	1	**1**	1

n – objektide arv ühes simulatsioonis

m – tunnuste arv igal objektil

v – mitme esimese tunnuse väärtuseid kasutati kolmanda tunnuse väärtuse puudumise tõenäosuse määramiseks igal objektil

n_L, *n_T* – kahe testi vastava parameetrite kombinatsiooni korral kasutatud simulatsioonide arvud

aeg_L, *aeg_T* – kahe testi vastava parameetrite kombinatsiooni korral keskmiselt tuhandeks simulatsiooniks kulunud ajad sekundites

min_L, *min_T* – kahe testi vastava parameetrite kombinatsiooni korral sisuka hüpoteesi vastu võtmise tõenäosuste 95-protsendiste usaldusvahemike alampiirid

max_L, *max_T* – kahe testi vastava parameetrite kombinatsiooni korral sisuka hüpoteesi vastu võtmise tõenäosuste 95-protsendiste usaldusvahemike ülempiirid

mean_L, *mean_T* – kahe testi vastava parameetrite kombinatsiooni korral sisuka hüpoteesi vastu võtmise tõenäosuste punktihinnangud

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Ivo Adermann,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
   "Andmete puudumise struktuuri määramise testid", mille juhendaja on Ene Käärik,

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi Dspace'is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi Dspace'i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **28.04.2015**