УДК 303.064

СЕРГІЙ ДЕМБІЦЬКИЙ,
кандидат соціологічних наук, науковий співробітник відділу методології і методів соціології Інституту соціології НАН України

ТЕТЯНА ЛЮБИВА,
кандидат соціологічних наук, молодший науковий співробітник відділу методології і методів соціології Інституту соціології НАН України

Сучасні техніки управління даними соціологічних досліджень

Анотація

Статтю присвячено трьом ключовим компонентам управління даними масових соціологічних опитувань: 1) логічному контролю ("чищенню") масиву; 2) модифікації даних; 3) роботі з пропущеними значеннями. У першій частині розглянуто основи технічного і логічного "чищення" масиву. У другій — описано способи перетворення даних у контексті певних дослідницьких ситуацій, а також техніки перекодування наявних і обчислення нових змінних. У третій наведено методологічні положення опрацювання пропущених значень і деякі методи роботи з ними.

Особливістю статті є її практична спрямованість — усі розглядувані способи управління даними спираються на приклади, котрі, своєю чергою, містять команди мовою програмування R, що набула неабиякого поширення в наукових колах як середовище статистичних обчислень і побудови графіків.

Ключові слова: *управління даними, "чищення" масиву, пропущені значення, R*

Вступ

Загальновідомо, що в багатьох випадках підготовка до аналізу вже зібраних кількісних даних може займати значно більше часу і забирати значно більше сил, ніж сам статистичний аналіз. Незважаючи на це, у рамках вітчизняної емпіричної соціології проблема управління даними є у певному сенсі банальною. Відповідні інструменти становлять самозрозумілий мінімум, необхідний для успішної роботи соціолога кількісного профілю. Разом із тим спектр задач, розв'язуваних у рамках управління кількісними даними, значно ширший, ніж може собі уявити середньостатистичний український соціолог. Тому метою нашої статті є розширення концептуального поля вітчизняної соціології у плані засад управління кількісними даними, а також збагачення її інструментарію програмними можливостями мови програмування *R*.

Стаття передбачає деяке (принаймні початкове) ознайомлення з *R*[1], оскільки всі приклади спираються на його використання. Переважна частина команд, наведених у праці, доволі прості для розуміння. Проте є й синтаксис, що передбачає ознайомлення з *R* на рівні, вищому за початковий. Але, незалежно від складності, всі наведені тут команди після незначної модифікації можуть бути використані в інших дослідницьких проектах.

Також слід зазначити, що приклади статті ґрунтуються на використанні даних соціологічного моніторингу "Українське суспільство", що його здійснює Інститут соціології НАН України від 1994 року.

Логічний контроль ("чищення") масиву

Дуже часто спеціалісти, які вводять дані й формують масив, оскільки не є аналітиками, не звертають уваги на змістове співвідношення відповідей у межах кожної з анкет (наприклад, щодо віку та освіти), оскільки процес уведення даних часто доведений ними до автоматизму. Крім того, коли оператор недостатньо досвідчений або неуважний, погано ознайомився з анкетою чи паспортом для введення даних, у масив можуть потрапити значення, що відрізняються від відповідних в анкеті. Тому безпосередньо після введення даних кількісних досліджень виникає необхідність логічного контролю масиву. Під логічним контролем ми розуміємо перевірку даних масиву на предмет присутності в них логічних невідповідностей. Наприклад, якщо людина говорить, що її вік 16 років і при цьому вона має вищу освіту, слід перевірити, чи не є це помилкою введення.

У вітчизняній літературі етап логічного контролю даних практично не описаний. Його засади та правила написання відповідного синтаксису наведено у довідці статистичного пакета "OCA for Windows" (довідка доступна із самої програми). У закордонній літературі контролю даних

[1] Якщо читач відчує необхідність у додатковому вивченні основ *R*, ми рекомендуємо чудову книгу Роберта Кабакова "*R* у дії" [Кабаков, 2014].

приділяють значно більше уваги[1]. На популярному освітньому порталі "Coursera" (https://www. coursera. org/) цій темі присвячено окремий курс [Leek, s.a.].

Правильно проведений контроль масиву даних є однією із ключових умов їхньої якості. Наприклад, у банк даних Європейського соціального дослідження (ESS, http://www. europeansocialsurvey. org/) від країн-учасниць приймаються лише дані, що пройшли ретельний контроль.

Контроль масиву даних також називають "чищенням масиву". Останнє визначення частіше використовують у колі вітчизняних дослідників, тому далі ми будемо використовувати саме його. Це дуже важливий етап управління даними, що дає змогу:

— визначити помилки введення операторами;

— знайти анкети, що могли бути фальсифіковані;

— підготувати масив для подальшої роботи.

Останній пункт не є основною сферою, для якої використовують чищення даних. Хоча практика показує, що деякі анкети, котрі містять чимало логічних суперечностей, бажано перевірити додатково, зв'язавшись із респондентом (за наявності його контактів), якщо раніше ця анкета не потрапила у вибірковий контроль після польових робіт.

Чищення масиву можна умовно поділити на технічне і логічне. Технічне чищення використовують для: 1) перевірки запитань із багатьма альтернативами: контроль максимальної / мінімальної кількості обраних варіантів відповіді (на той випадок, якщо кількість категорій, котру необхідно обрати, встановлено точно або обмежено певною кількістю); 2) перевірка, чи не обрано у випадку запитань цього самого типу варіант "Нічого з переліченого" разом з іншими варіантами відповіді; 3) перевірка правильності переходів після запитань-фільтрів.

Логічне чищення — більш творче. Якщо у випадку технічного чищення дослідник може керуватися суто вимогами до заповнення анкети, то при логічному чищенні йому необхідно підключити свою уяву і знання, щоб визначити ті конфіґурації відповідей, котрі в логічному сенсі суперечливі. Наявність таких ситуацій зазвичай вказує на помилки введення даних чи на фальсифікацію анкет.

Передусім аналітикові необхідно перевірити, чи нема дублів анкет у масиві, тобто введення однієї анкети два і більше разів. Найпростіший спосіб — це побудова одновимірного розподілу за ключовою ознакою[2]. Якщо за нею виявлено дублі, треба переконатися, що відповіді на запитання також ідентичні, і видалити дублювальну анкету. Якщо ж відповіді на запитання з однаковими номерами не ідентичні, тоді помилка введення була в номері анкети, і кожній анкеті треба присвоїти унікальний номер. Для пе-

---

1 Див., напр.: [Beaver, s.a.].

2 Ключова ознака (як правило, унікальний номер анкети) є важливим елементом опрацювання даних, який можна використовувати не тільки для чищення. Наприклад, при об'єднанні різних змінних одного масиву за ключовою ознакою можна порівняти спостереження (якщо в різних підмасивах вони розташовані у різному порядку).

ревірки ідентичності відповідей не за ключовою ознакою, а за змістом усіх відповідей, можна використовувати такий синтаксис:

```
shortlist <- list()
takeData <- finalUSind
takeData <- as.matrix(takeData)
for (i in 1:nrow(takeData)) {
    check <- apply(takeData[,-70], 1,
                   function(x) identical(x, takeData[i,-70]))
    if (table(check)["TRUE"] > 1) {
        shortlist <- c(shortlist,
                       which(check == T)[which(check == T) != i])
        if (!i%in%shortlist)
            cat("рядок",takeData[i,"keyVar"],":",
                which(check == T)[which(check == T) != i], "\n")
    }
}
```

Спершу масив, що потребує перевірки, перетворюють на матрицю "takeData" і створюють порожній вектор "shortlist". Далі, використовуючи цикл for(), для кожного рядка матриці через функцію apply() перевіряють його ідентичність з усіма рядками (в тому числі і з самим собою) цієї ж матриці, а результати записують у вигляді логічного вектора "check". У даному разі "–70" указує на те, що необхідно виключити змінну із сімдесятим індексом, котра тут є ключовою ознакою (інакше кожне спостереження буде унікальним). Якщо кількість значень TRUE більш як один[1] (тобто зміст рядка збігається не тільки сам із собою, а й зі змістом, принаймні, ще одного рядка), тоді у вектор "shortlist" записують номери відповідних рядків, крім того, що ми перевіряємо в даний момент. І якщо номера поточного рядка немає в "shortlist", тоді за допомогою функції cat() виводиться результат, що являє собою номер рядка, що перевіряється в даний момент, і тих рядків, що його дублюють.

Перевіривши масив на наявність дублів, можна переходити безпосередньо до чищення даних, у процесі якого аналітик формулює низку логічних умов програмними засобами. Результатом цієї перевірки є список з номерами анкет, в яких виявлено логічні суперечності. Коли в розпорядженні дослідника є паперові анкети, він може звернутися до них, аби зрозуміти, чи припустився оператор відповідних помилок у процесі введення даних, і за необхідності виправити їх. Також можна визначити, чи не вирізняються окремі анкети серед інших за кількістю помилок. За наявності таких помилок їх можна проконтролювати, зв'язавшись з респондентом. Крім того, слід перевірити й інші анкети відповідного інтерв'юєра.

Розгляньмо приклади технічного (приклади 1 і 2) і логічного (приклади 3 і 4) чищення.

---

[1] Позначення логічних операторів у *R* таке: == (дорівнює), < (менше), <= (менше або дорівнює), > (більше), >= (більше або дорівнює), ! = (не дорівнює), | (або), & (і).

Приклад 1. Запитання із багатьма альтернативами щодо членства у громадських і політичних організаціях не може одночасно містити відповідь про членство в одній чи більш як одній організації зі списку і варіант відповіді "Не належу до жодної". Для перевірки можна використати такий набір команд:

```
finalUSind$b4_sum <- rowSums(finalUSind[15:27])
falserows_b4 <- finalUSind$b4_sum > 0 & finalUSind$b4_14 == 1
wrong_b4 <- which(falserows_b4 == T)
```

Спочатку знаходять загальну кількість виборів громадських і політичних організацій для кожного респондента, яку записують у змінну "b4_sum". Для цього використовують функцію rowSums(), що належить до категорії функцій (поряд із colSums(), rowMeans(), colMeans()), призначених для аґреґування інформації в рядках або стовпчиках масиву. Далі на підставі двох логічних умов, що відповідають описаним вище умовам, створюють логічний вектор "falserows_b4", в якому значення TRUE відповідають тим рядкам масиву, в яких виявлено логічну суперечність, а FALSE — тим, в яких не виявлено. Після цього через функцію which() створюють вектор "wrong_b4", що містить номери лише тих рядків (анкет), в яких виявлено логічну суперечність.

Приклад 2. У моніторинґу "Українське суспільство" відсутні запитання-фільтри. Тож уявімо, що респондентові, котрий заявляє про небажання їхати з населеного пункту, в якому він живе, не мають ставити запитання про можливий напрямок переїзду.

```
falserows_i5 <- finalUSind$i4 == "Hi" & !is.na(finalUSind$i5)
wrong_i5 <- which(falserows_i5 == T)
```

У першій команді крім простої логічної умови також використовують заперечення результатів застосування логічної функції is.na(). Використання цієї функції самої по собі повертає TRUE для спостережень із пропусками і FALSE для інших спостережень за відповідною змінною (у нашому випадку TRUE конвертують у FALSE і навпаки).

Приклад 3. Кількість людей, з якими респондент мешкає в одній кімнаті, не має перевищувати кількості людей, з якими він мешкає в одній квартирі (будинку).

```
falserows_k7 <- finalUSind$k7 > finalUSind$k6
wrong_k7 <- which(falserows_k7 == T)
```

Приклад 4. Якщо в запитанні про вид зайнятості відзначено відповідь "Не працюю і не маю жодних джерел доходу", тоді в запитанні про розмір заробітної плати за останній місяць має бути нуль.

```
false_l3 <- finalUSind$n1 == "Не працюю і…" & finalUSind$l3 > 0
wrong_l3 <- which(false_l3 == T)
```

Чим більше змінних використовують в логічних умовах, тим вище можливість знаходження помилки введення чи виявлення фальсифікованої анкети. Разом із тим слід мати на увазі, що деякі умови логічного чищення не можуть прямо свідчити про помилкову чи неправильну відповідь. В останньому прикладі респондент міг сказати про свій останній особистий дохід, а

не про дохід за останній місяць, або тільки в даному місяці він став безробітним без жодних джерел доходу, а в минулому місяці мав дохід.

Крім того, не варто захоплюватися такими "слабкими" умовами, як, наприклад, "Люди з вищою освітою не можуть бути цілком незадоволеними рівнем своєї освіти" або "Українці за національністю не можуть сповідувати іслам". Хоча це й нестандартні ситуації, але цілком ймовірні. Також слід дотримуватися розумних меж у кількості логічних умов. Присутність малої частки нелогічності в анкеті припустима. Трапляються ситуації, коли деякі оцінні судження на початку і наприкінці анкети не збігаються, що може свідчити про невизначеність думки респондента з даного запитання.

Після реалізації синтаксису й отримання переліку виявлених помилок можливі два варіанти подальших дій: 1) звернення до паперових анкет за їхнім номером і перевірка правильності введених даних (тобто зіставлення значень у масиві й анкеті), виправлення значень за потреби; 2) робота тільки з масивом даних.

Перший варіант кращий, але тоді чищення займає значно більше часу. Другий варіант — менш витратний за часом, але в цьому разі дослідникові доводиться приймати рішення щодо зміни значень у змінних, що містять невідповідності, або щодо вилучення їх з аналізу "на власний страх і ризик".

У завершенні цієї частини статті ми наводимо синтаксис, що структурує роботу, продемонстровану у прикладах, і приводить результати до зручнішого формату:

```
itogi <- list("помилка в b4" = wrong_b4, "помилка в i5" = wrong_i5,
              "помилка в k7" = wrong_k7, "помилка в l3" = wrong_l3)
inorder <- names(sort(table(unlist(itogi)),decreasing = T))
rawresult <- list()
for (i in inorder) {
    rawresult[[i]] <- lapply(itogi, function(x) as.numeric(i) %in% x)
}
letsee <- lapply(rawresult, function(x) x[x == T])
```

Спочатку створюють список "itogi" з інформацією про пропуски, отриманою в попередніх чотирьох прикладах. Далі створюють вектор "inorder", в якому фіксується порядок анкет залежно від кількості знайдених у них логічних невідповідностей (за убуванням). Після цього створюють ще один список "rawresult", в якому, застосовуючи цикл for() і функцію lapply(), для кожної анкети фіксують наявність чи відсутність логічної невідповідності стосовно кожної перевірки. Нарешті створюють останній список "letsee", в якому для кожної анкети залишають лише ті результати, де зафіксовано наявність логічних невідповідностей. Цей фінальний список можна використовувати як рекомендацію для пошуку й опрацювання анкет, що підлягають умовам перевірки.

Модифікація даних

Після того, як масив "почищений" і готовий до роботи, часто виникає необхідність модифікації наявних змінних. В одних випадках це пов'язане із загальною логікою аналізу (наприклад, виключення з аналізу так званих викидів), в інших — із вимогами використовуваного статистичного методу

(наприклад, "підгонка" форми розподілу), у третіх — із необхідністю отримання нової інформації на підставі наявної (наприклад, обчислення індексів). Розгляньмо ці ситуації докладніше.

У разі використання числової змінної з великим діапазоном можливих значень слід перевірити її на наявність "викидів" — нетипово великих або малих значень. Для візуалізації останніх можна застосувати функцію boxplot(), що дає змогу побудувати діаграму розмахів. В якості даних створимо вектор значень "salary" на підставі змінної "l3", виключивши з неї респондентів без доходів і замінивши лише тих респондентів, яких було опитано в рамках останнього етапу дослідження, тобто у 2014 році.

```
finalUSind$salary <- NA
indexes <- which(finalUSind$l3 > 0 & finalUSind$year == 2014)
finalUSind$salary[indexes] <- finalUSind$l3[indexes]
boxplot(salary, horizontal = T, las = 1, col = "slategray3",
        outpch = 16, outcol = "slategray3")
```

Отримана діаграма (див. рис. 1) демонструє нижню і верхню межі "викидів", квартилі, медіану і самі "викиди" (вони позначені точками).

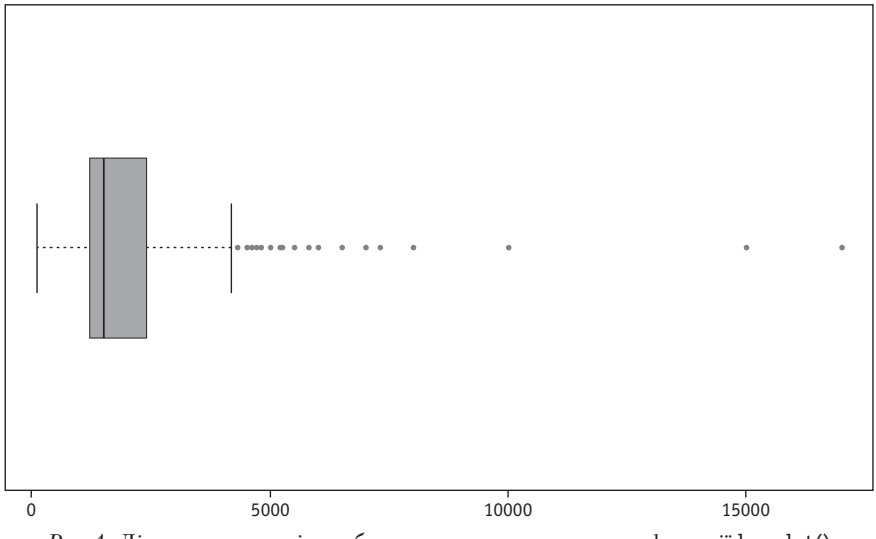

Рис. 1. Діаграма розмахів, побудована з використанням функції boxplot()

Для змістової деталізації числових параметрів цієї діаграми використовують функції boxplot. stats() і barplot(). Перша показує нижню межу "викидів" (замість неї може бути показано мінімальне значення, якщо воно більше за нижню межу), перший квартиль, медіану, другий квартиль, верхню межу "викидів" (може бути показано максимальне значення, якщо воно менше за верхню межу)[1]. Друга дає змогу проаналізувати частотний розподіл "викидів" через стовпчикову діаграму (див. рис. 2).

[1] Нижня межа дорівнює $Q1 - 1{,}5 \times \text{IQR}$, а верхня — $Q3 + 1{,}5 \times IQR$, де $Q1$ — перший квартиль, $Q3$ — третій квартиль, IQR — міжквартильна широта.

```
boxplot.stats(salary)$stats
[1]    110   1200    1500    2400    4200

barplot(table(boxplot.stats(salary)$out),horiz = T,
        las = 1, col = "slategray3")
```

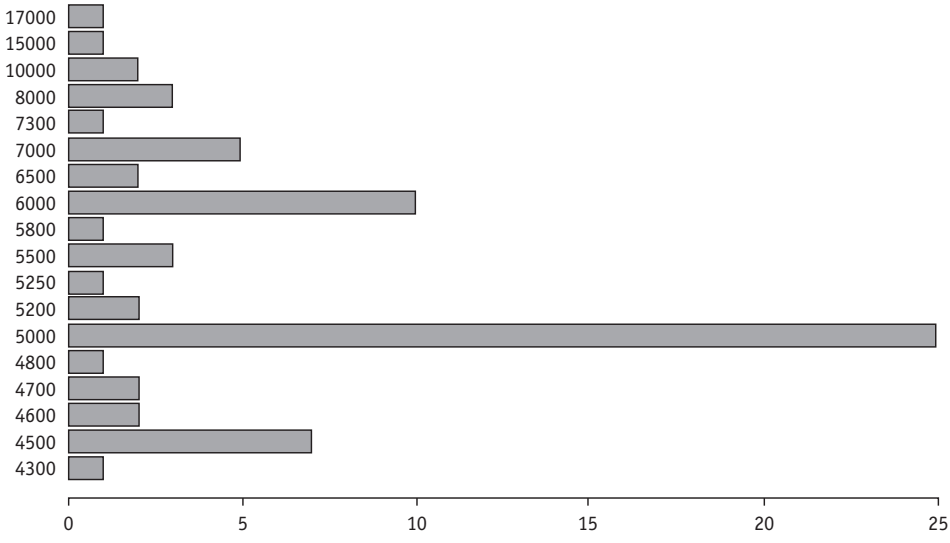

Рис. 2. Стовпчикова діаграма, побудована через функцію barplot()

Рішення про те, що саме зрештою вважати викидом, завжди є дещо суб'єктивним і ґрунтується на інтерпретації отриманих результатів. У цьому прикладі ми вирішили, що викидами будуть всі значення, які дорівнюють або перевищують 10 000. Щоб створити нову змінну, що не містить викидів, можна використовувати таку команду:

```
finalUSind$l3new <- finalUSind$l3[finalUSind$l3 < 10000]
```

Іноді виникає необхідність перетворення тієї чи іншої змінної за вимогами щодо аналізованих даних з боку застосовуваного методу. Також може знадобитися приведення різних шкал до однієї розмірності. Прикладами таких ситуацій є, відповідно, використання кластерного аналізу і побудова лінійних графіків. Так, у рамках кластерного аналізу змінні із ширшим діапазоном значень справлятимуть більший вплив на утворення кластерів, що далеко не завжди виправдано. Що стосується побудови лінійних графіків, то аналіз змінних із вужчим діапазоном значень буде ускладненим. У таких випадках слушною є нормалізація чи обчислення *Z*-значень.

Скажімо, ми хочемо об'єднати респондентів у групи залежно від їхнього зросту й ваги. Для нормалізації в *R* використовується функція scale():

```
data$rost <- scale(finalUSind$g7)
data$ves <- scale(finalUSind$g8)
```

Змінна g7 включає інформацію про зріст респондента (інтерквартильна широта дорівнює 96 см), а g8 — про вагу (інтерквартильна широта дорівнює 17 кг). Після стандартизації вплив розмірності шкали на результати кластерного аналізу буде нівельовано (тут ми виходимо з того, що обидві змінні однаково важливі у процесі розподілу респондентів за групами).

Тепер розгляньмо приклад, що стосується зміни рівнів інтернальності/екстернальності (інтернали пояснюють зміни у своєму житті зовнішніми та незалежними від них обставинами, екстернали, навпаки, беруть відповідальність на себе) і матеріальної забезпеченості респондентів у країні загалом. Для аналізу використано лінійні графіки. Але перш ніж їх побудувати, необхідно обчислити відповідні індекси. При оцінюванні матеріального становища сім'ї[1] достатньо обчислити середнє значення для кожного року. Для цього можна використати функцію aggregate():

```
finalUSind$l2num <- as.numeric(finalUSind$l2)
indfirst <- aggregate(finalUSind$l2num, by=list(finalUSind$year),
                      FUN=mean, na.rm = T)
indfirstfin <- scale(indfirst$x)
```

Що стосується індексу екстернальності/інтернальності, то розрахунок буде складнішим. Спершу створимо й запишемо в об'єкт lineGraph таблицю спряженості (див. табл.) з використанням такого синтаксису[2]:

```
lineGraph <- table(finalUSind$year,finalUSind$f4)
```

Таблиця

Дані для обчислення індексу екстернальності/інтернальності

Рік	Варіанти відповіді				
	Переважно від зовнішніх обставин	Деякою мірою від мене, але більше від зовнішніх обставин	Однаковою мірою від мене і від зовнішніх обставин	Більшою мірою від мене, ніж від зовнішніх обставин	Переважно від мене
1992	358	611	450	215	115
1994	509	553	385	180	166
1995	528	513	409	186	166
...	...	...	...	...	...
2010	289	550	560	211	189
2012	303	581	549	184	183
2014	296	508	548	255	191

1 Використовується шкала від 0 до 10, де 0 — найнижча оцінка, 10 — найвища.

2 Зовнішній вигляд таблиці відредаґовано відповідно до формату статті.

Тепер для кожного року необхідно із суми спостережень, що потрапляють у перші дві категорії, відняти суму для двох останніх, а отримане значення розділити на загальну кількість спостережень, після чого можна стандартизувати індекс:

```
indsec <- (rowSums(lineGraph[4:5,])
           – rowSums(lineGraph[1:2,])) / rowSums(lineGraph)
indsecfin <- scale(indsec)
```

Далі наведено два графіки (див. рис. 3): перший побудовано на підставі "сирих" індексів, другий — нормалізованих. У першому випадку зв'язок змінних практично неможливо простежити, у другому він виражений дуже чітко.

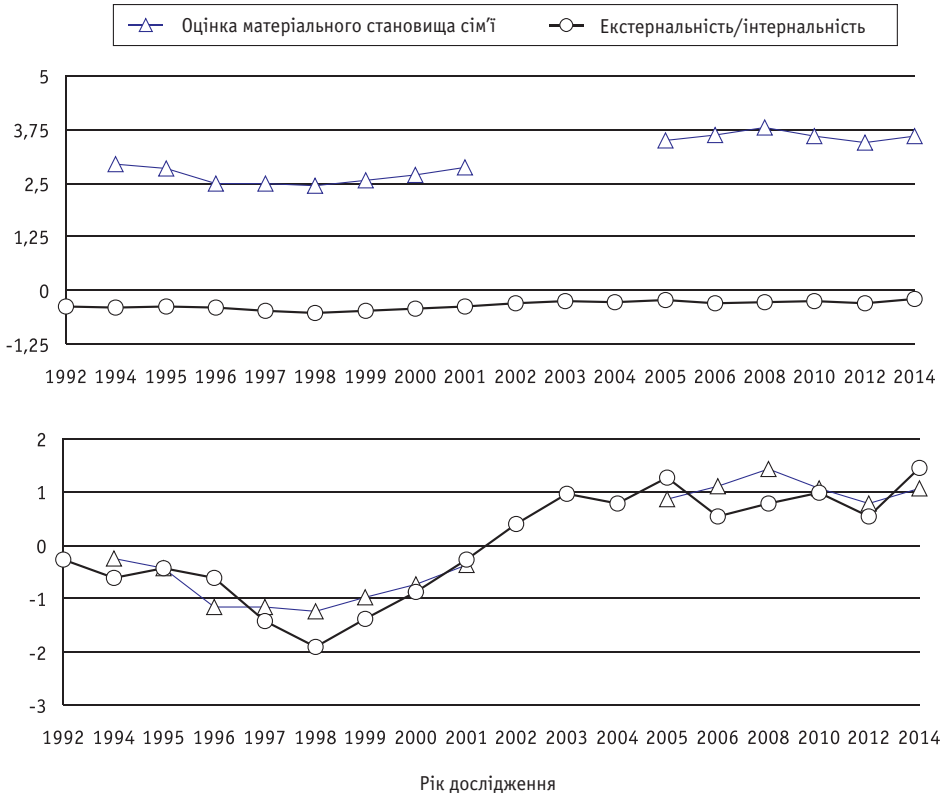

Рис. 3. Лінійний графік до і після нормалізації значень

Іншими поширеними способами перетворення є логарифмічне, квадратного кореня, зворотне і квадратне [Шипунов, 2014: с. 62—64]. Логарифмічне перетворення може дати нормальний розподіл, коли воно скошене вправо. Крім того, воно робить залежності між змінними більш лінійними і зрівнює дисперсії. Оскільки логарифмічне перетворення непридатне для роботи з нулями, рекомендується додавати одиницю:

```
log(data$var1 + 1)
```

Перетворення квадратного кореня схоже за дією з логарифмічним. При цьому воно може працювати з нулями, але не може — з неґативними значеннями:

```
sqrt(data$var1)
```

Зворотне перетворення використовують за потреби стабілізації дисперсії. Так само, як і логарифмічне, воно передбачає додавання одиниці:

```
1/(data$var1 +1)
```

Квадратне перетворення підходить для випадків, коли розподіл скошений уліво (може привести до нормального). Воно лінеаризує залежності й вирівнює дисперсії:

```
data$var1^2.
```

Крім описаних способів модифікації часто виникає необхідність в обчисленні нових змінних (як на підставі умов спеціальних методик, що застосовувалися в рамках дослідження, так і, з урахуванням свого досвіду, на підставі окремих змінних), а також у перекодуванні наявних (наприклад, розподіл респондентів за віковими категоріями чи регіонами проживання).

За необхідності дихотомізації числової змінної можна використовувати функцію ifelse(). Як вхідні дані знову використано відповіді респондентів у 2014 році про дохід за останній місяць. При цьому не виключено й тих, у кого дохід відсутній:

```
num.data <- finalUSind$l3[finalUSind$year == 2014]
poor.and.rich <- ifelse(num.data <= 1500, 0, 1)
```

Функція ifelse() приймає три арґументи: 1) логічну умову (у нашому випадку — дохід понад 1500); 2) значення, що повертається функцією у разі істинності умови для респондента (в нашому випадку — нуль); 3) значення, що повертається у разі хибності умови для респондента (в нашому випадку — 1).

Другим простим способом розбивки числових змінних є автоматизована процедура категоризації через функцію bin.var(), що дає змогу застосувати один із трьох методів: 1) побудову інтервалів рівних розмірів (розмір інтервалу дорівнює $Xmax/n$, де $Xmax$ — максимальне значення в розподілі, n — кількість інтервалів); 2) побудову інтервалів однакової наповненості (по N/n респондентів у кожній групі, де N — розмір вибірки, n — кількість інтервалів); 3) побудову інтервалів на підставі кластеризації методом k-середніх. Спершу розгляньмо загальну форму відповідної команди:

```
bin.var(salary[salary < 10000], bins=4, method="intervals", labels=NULL))
```

Як перший арґумент вказується числова змінна масиву або вектор (як у нашому випадку), як другий (bins) — кількість інтервалів, яку необхідно побудувати, як третій (method) — метод розбивки, завдяки останньому (labels) задається формат найменування груп. Тепер проаналізуймо, до яких ре-

зультатів веде кожен із трьох методів (для наочності результати наведено у відредаґованому вигляді):

```
table(bin.var(salary[salary < 10000], bins=4,
        method="intervals", labels=NULL))
 (102,2080]      1115
(2080,4060]       383
(4060,6030]        56
(6030,8010]        11

table(bin.var(salary[salary< 10000], bins=4,
        method="proportions", labels=NULL))
 (110,1190]       392
(1190,1500]       397
(1500,2400]       388
(2400,8000]       388

table(bin.var(salary[salary< 10000], bins=4,
        method="natural", labels=NULL))
 (-Inf,1450]      678
(1450,2400]       499
(2400,4200]       322
(4200,8000]        66
```

Як можна бачити, у випадку аналізу доходів найбільш збалансовану структуру (як у плані величини інтервалів, так і щодо наповненості категорій) забезпечує використання третього методу.

У разі перекодування номінальних змінних, а також за потреби у повному контролі при створенні кількох категорій на підставі числової змінної найліпшим способом є використання функції within(). Для прикладу розгляньмо перекодування змінної про сімейний статус респондента із шістьма категоріями: 1 — "Ніколи не перебував(-ла) у шлюбі", 2 — "Перебуваю в зареєстрованому шлюбі", 3 — "Перебуваю у фактичному, незареєстрованому шлюбі", 4 — "Розлучений(-а) офіційно", 5 — "Розійшовся(-ася), хоча офіційно не розлучений(-на)", 6 — "Удівець (удова)". Нова змінна матиме чотири категорії: "Ніколи не перебував(-ла) у шлюбі" (перша категорія), "Перебуваю у шлюбі (цивільному чи фактичному)" (друга і третя категорії), "Розійшовся або розлучений" (четверта і п'ята категорії), "Вдівець (удова)" (шоста категорія). Синтаксис *R* такий:

```
finalUSind <- within(finalUSind,{
    s3_4 <- NA
    s3_4[as.numeric(s3) == 1] <- 1
    s3_4[as.numeric(s3) == 2] <- 2
    s3_4[as.numeric(s3) == 3] <- 2
    s3_4[as.numeric(s3) == 4] <- 3
    s3_4[as.numeric(s3) == 5] <- 3
    s3_4[as.numeric(s3) == 6] <- 4
})
```

У першому рядку ми вказуємо назву масиву, в якому здійснюють перетворення (finalUSind), у другому присвоюємо новій змінній пропуски (цей крок обов'язковий), у подальших за допомогою логічних виразів визначаємо умови для зіставлення старих і нових категорій. Використовуючи інші логічні оператори і "перевернувши" порядок присвоєння значень, можна скоротити наведений синтаксис:

```
finalUSind <- within(finalUSind,{
    s3_4 <- NA
    s3_4[as.numeric(s3) == 6] <- 4
    s3_4[as.numeric(s3) < 6] <- 3
    s3_4[as.numeric(s3) < 4] <- 2
    s3_4[as.numeric(s3) == 1] <- 1
})
```

Ще одним альтернативним варіантом у даному випадку буде такий (тут порядок присвоєння не має значення):

```
finalUSind <- within(finalUSind,{
    s3_4 <- NA
    s3_4[as.numeric(s3) == 6] <- 4
    s3_4[as.numeric(s3) == 2 | as.numeric(s3) == 3] <- 2
    s3_4[as.numeric(s3) == 4 | as.numeric(s3) == 5] <- 3
    s3_4[as.numeric(s3) == 1] <- 1
})
```

Коли необхідно обчислити нову змінну на підставі наявних, використовують звичні математичні операції[1]. Також можна використовувати й умонтовані в *R* функції. Спершу розгляньмо побудову індексу пропорційності ваги. Він обчислюється як відношення ваги тіла людини в кілограмах до квадрата її зросту в метрах. Виходячи з того, що в масиві зріст зафіксований у сантиметрах, а вага — в кілограмах, синтаксис може прийняти один із двох варіантів:

```
finalUSind$g8and7 <- finalUSind$g8/(finalUSind$g7*finalUSind$g7/10000)
finalUSind$g8and7 <- finalUSind$g8/(finalUSind$g7**2/10000)
```

У випадку, коли необхідно обчислити більшу кількість нових змінних, зручно використовувати функцію transform(). Наприклад, при переведенні змінних, що містять інформацію у гривнях (дохід респондента, мінімально прийнятний рівень доходу за його оцінкою, рівень доходу, за якого, на думку респондента, можна жити добре, тощо), застосування цієї функції виглядатиме так:

```
finalUSind <- transform(finalUSind,
                        l3_dollar = l3 / currency,
                        l5_dollar = l5 / currency,
```

[1] Позначення основних математичних операторів у *R* таке: + (додавання), – (віднімання), × (множення), / (ділення), ^ або ** (зведення до степеня).

```
                    l6_dollar = l6 / currency,
                    l7_dollar = l7 / currency,
                    l8_dollar = l8 / currency,
                    l9_dollar = l9 / currency,
                    l10_dollar = l10 / currency,
                    l11_dollar = l11 / currency,
                    l12_dollar = l12 / currency,
)
```

У першому рядку вказано масив (до знака присвоювання), в який записують нові змінні і з якого беруть змінні для розрахунку (як перший аргумент функції). Далі наведено вираз для обчислення всіх необхідних змінних (змінна "currency" містить інформацію про курс долара).

У багатьох випадках при обчисленні нових змінних дуже зручно використовувати вбудовані, тобто доступні в *R* за умовчанням функції. Слушним прикладом є обчислення Інтегрального індексу соціального самопочуття, розробленого Євгеном Головахою і Наталією Паніною [Головаха, 1997]. На першому етапі обчислення цього індексу необхідно знайти суму значень за 20-ма індикаторами. Спершу розглянемо синтаксис без використання вбудованої функції:

```
finalUSindTest <- transform(finalUSindTest, f6_sum = as.numeric(f6_1)
                    + as.numeric(f6_2) + as.numeric(f6_3)
                    + as.numeric(f6_4) + as.numeric(f6_5)
                    + as.numeric(f6_6) + as.numeric(f6_7)
                    + as.numeric(f6_8) + as.numeric(f6_9)
                    + as.numeric(f6_10) + as.numeric(f6_11)
                    + as.numeric(f6_12) + as.numeric(f6_13)
                    + as.numeric(f6_14) + as.numeric(f6_15)
                    + as.numeric(f6_16) + as.numeric(f6_17)
                    + as.numeric(f6_18) + as.numeric(f6_19)
                    + as.numeric(f6_20))
```

З метою компактності тут можна використовувати функцію rowSums(), але перед цим усі індикатори необхідно привести до числового вигляду (функція приймає лише числові змінні, а зараз вони є змінними-факторами). Для цього ми вдаємося до функції sapply(), використання якої полягає в застосуванні заданого методу до кожного компонента списку (масив є його окремим випадком), результатом чого є вектор значень[1]:

```
finalUSind$f6_sum <- rowSums(sapply(finalUSind[184:203], as.numeric))
```

Тут звернення до змінних здійснюють не за іменами змінних, а за їхніми індексами. Змінним із f6_1 до f6_20 відповідають індекси від 184-го до 203-го. Як можна бачити, у даному разі для одержання суми індикаторів знадобилося значно менше рядків коду.

---

[1] Знайти правильне рішення тут допомогли Кирило Захаров і Олександр Виноградов, за що автори статті висловлюють їм особливу подяку.

Далі необхідно скориґувати отриману змінну (f6_sum) з урахуванням того, що четверту категорію ("Не цікавить") при обчисленні індексу треба прийняти за двійку. Отже, сума двадцяти індикаторів має давати максимум 60, а не 80 балів. Для розв'язання цієї задачі необхідно підрахувати для кожного респондента кількість четвірок і відняти відповідне значення, помножене на два із сумарного індексу. Тут ми використовуємо цикл for():

```
finalUSind$f6_min <- 0
for (i in 184:203) {
    min <- finalUSind[i] == "Не цікавить"
    finalUSind$f6_min <- finalUSind$f6_min + min
}
finalUSind$f6_sum <- finalUSind$f6_sum -- finalUSind$f6_min*2
finalUSind$f6_min <- NULL
```

Перша команда в тілі циклу створює набір значень, що складаються з TRUE (істина), якщо значення змінної дорівнює чотирьом, і FALSE (хиба), якщо ні. Оскільки істина в математичних операціях приймається за одиницю, а неправда — за нуль, легко підрахувати кількість четвірок щодо кожного респондента (для цього використовують змінну f6_min). Останній рядок синтаксису видаляє лічильник четвірок, оскільки для подальшої роботи він не потрібен.

Робота із пропущеними значеннями

Багато змінних (а в соціологічних дослідженнях, ґрунтованих на масових опитуваннях, вони нерідко становлять більшість) мають пропущені значення. Останні призводять до зниження статистичної потужності, а також можуть бути причиною систематичних помилок [Бослаф, 2015: с. 450].

Опрацювання пропущених значень є доволі розвиненою дослідницькою цариною з узвичаєною термінологією і множиною рішень для різних дисциплін і конкретних досліджень. Зі спробою широкого узагальнення засад опрацювання пропущених даних у соціальних науках можна ознайомитися, наприклад, у праці Даніеля Ньюмана [Newman, 2014]. У рамках нашої статті ми звернемося до головних понять цієї теорії, а також до основних методів розв'язання проблеми пропущених значень.

Заведено вирізняти три види пропусків — цілком випадкові, випадкові та невипадкові. Ця термінологія веде початок від відомої праці Дональда Рубіна [Rubin, 1976]. Цілком випадкові пропуски (ЦВП-припущення) мають місце тоді, коли підвибірка наявних значень за змінною(-ми), що підлягає(-ють) вивченню, і далі є моделлю генеральної сукупності. За приклад може правити випадок, коли пропуски за деякою змінною (наприклад, політичні преференції) не залежать від значень змінних-предикторів (наприклад, стать, вік, регіон проживання тощо), а також від значень самих пропусків (наприклад, не виникає ситуації, коли респонденти з певною політичною позицією частіше за інших не дають відповіді на відповідне запитання). Вибір моделі цілком випадкових пропусків — єдине припущення, котре можна перевірити емпірично. Що стосується випадкових і невипад-

кових пропусків, то відповідні припущення неможливо перевірити на підставі наявного масиву.

При випадкових пропусках (ВП-припущення) їхні значення залежать від значень змінних-предикторів і не залежать від власних значень пропусків. Так, якщо пропуски у відповідях на запитання про політичні преференції частіше трапляються серед людей старшого віку (але всередині цієї групи вони розподілені випадково), то йдеться про випадкові пропуски. У цьому разі виникає можливість зсуву результатів оцінювання параметрів за вибіркою загалом (якщо значення за відповідною підгрупою відрізняється від загального середнього).

Якщо ж можливість пропусків за певними змінними залежить від величини самих пропущених значень за цими змінними, тоді йдеться про невипадкові пропуски (НП-припущення). Наприклад, люди з лівими політичними поглядами з меншою ймовірністю (суто гіпотетичне припущення) схильні повідомляти відповідну інформацію. Такі пропуски вносять систематичні помилки в результати аналізу.

Методи опрацювання пропущених значень поділяють на традиційні та сучасні. До перших відносять порядкове і попарне видалення спостережень, заміщення середнім і заміщення із використанням регресії. До других — заміщення шляхом оцінювання максимальної правдоподібності, множинне заміщення, селективну модель і модель змішаних патернів.

За порядкового видалення з масиву виключаються будь-які спостереження, у значеннях змінних яких присутній бодай один пропуск. Такий підхід можна використовувати тільки при ЦВП-припущенні. Але навіть у такій ситуації його використання призводить до зниження статистичної потужності.

Як і в разі порядкового видалення, попарне видалення підходить тільки за ЦВП-припущення. Попарне видалення полягає в тому, що в кожному конкретному випадку аналізу з нього видаляють лишень ті спостереження, в яких є принаймні один пропуск за релевантними для даного аналізу змінними. Як і в разі порядкового видалення, воно є слушним суто за ЦВП-припущення. Тим самим створюється додаткова проблема, що полягає у використанні відмінних вибірок у рамках перевірки різних гіпотез одного дослідницького проекту.

Використання заміщення пропущених значень вибірковим середнім не рекомендується у жодному разі. Це пов'язане з тим, що такий спосіб опрацювання пропусків призводить до зсуву оцінок (оскільки зменшує дисперсію змінних і коваріацію між ними), незалежно від використовуваного припущення. Винятком є оцінювання самого середнього значення.

Заміщення з використанням регресії (замість пропущених використовують передбачені значення) зумовлює схожі із попереднім способом проблеми. Так, отримувані значення потрапляють в один патерн, що відповідає регресійній прямій, що вносить зсув як у плані дисперсії, так і в плані коваріації. Частково цю проблему можна розв'язати додаванням випадкових залишкових значень до передбачених величин. Але назагал цей спосіб заміщення доречний за ВП-припущення [Peugh, 2004; Cheema, 2014].

Заміщення через оцінювання максимальної правдоподібності являє собою ітераційний процес, кожна ітерація якого має два етапи: 1) середні значення змінних і коваріаційна матриця використовуються для обчислення регресійних рівнянь, що передбачають пропущені значення; 2) отриманий масив використовують для розрахунку оновлених середніх значень і коваріаційної матриці. Ці етапи повторюють доти, доки середні значення і коваріаційна матриця перестануть змінюватися. Можна сказати, що в цьому разі використовують послідовний підхід.

У разі множинного заміщення створюється набір альтернативних масивів даних (від п'яти і більше), в яких пропущені значення заміщуються передбаченими через регресію із додаванням випадкового елементу. Після цього середні значення змінних і коваріаційні матриці узагальнюють для одержання підсумкової оцінки. Цей підхід можна назвати конкурентним.

Як заміщення через максимальну правдоподібність, так і множинне заміщення підходять для ситуацій, в яких дослідник виходить із ЦВП- чи ВП-припущень. Для випадків, коли маємо НП-припущення, призначені селективна модель і модель змішаних патернів.

Селективна модель пов'язує основне регресійне рівняння з додатковим регресійним рівнянням, що передбачає імовірність відповідей. Дві частини моделі зв'язуються через скорельовані залишки (*correlated residuals*), і цей зв'язок є механізмом, за допомогою якого коригуються зсуви моделі, пов'язані із пропусками.

Модель змішаних патернів передбачає формування підгруп спостережень, в яких виявляються однакові патерни пропусків даних, із подальшим оцінюванням потрібних параметрів у кожній підгрупі. Після цього знаходять середньозважену оцінку даних параметрів.

На відміну від сучасних методів опрацювання пропущених значень при ЦВП- чи ВП-припущеннях методи опрацювання при НП-припущенні не дістали широкого визнання [Enders, 2010: p. 56–328].

Огляд технік роботи з пропущеними даними в *R* зроблено у праці Роберта Кабакова [Кабаков, 2014: с. 472–498]. Тут ми заторкнемо лише деякі з них. Як приклад проаналізуймо пропуски у змінній про дохід респондента.

Передусім слід вивчити структуру пропусків у співвідношенні з іншими змінними масиву (стать, вік, сімейний статус, освіта, загальна оцінка матеріального становища сім'ї). Для цього використовуємо функцію md. pattern() бібліотеки mice:

```
md.pattern(na.data)
```

Результат матиме такий вигляд:

	age	gender	education	wealth	marit.s	income	
14600	1	1	1	1	1	1	0
1360	1	1	1	1	1	0	1
38	1	1	1	0	1	1	1
6	1	0	1	1	1	1	1
1	0	1	1	1	1	1	1
127	1	1	1	1	0	1	1

30	1	1	0	1	1	1	1
21	1	1	1	0	1	0	2
1	1	0	1	1	1	0	2
9	1	1	1	1	0	0	2
1	1	1	1	0	0	1	2
2	1	1	0	1	1	0	2
1	0	1	0	1	1	1	2
1	1	1	0	1	0	1	2
1	1	1	1	0	0	0	3
1	1	1	0	0	1	0	3
	2	7	35	62	139	1395	1640

Перший стовпчик показує загальну кількість спостережень, що відповідають патернам, поданим у рядках під назвами змінних масиву, тобто від другого рядка і нижче. Самі патерни читаємо так: нулі означають, що за цією змінною спостереження має пропуск, одиниці — пропуску нема. Завершальний стовпчик показує загальну кількість пропусків у кожного такого спостереження. Отже, 14 600 спостережень масиву не мають пропусків (під усіма змінними стоїть одиниця, завершальний стовпчик дорівнює нулю); 1360 спостережень мають пропуск за змінною "дохід" (income) і т. д. В останньому рядку показано загальну кількість пропусків щодо кожної змінної і загальну кількість пропусків для всього масиву (сумарне значення завершального стовпчика).

Альтернативним способом подання тих самих даних є побудова спеціальної діаграми через функцію aggr() бібліотеки *VIM* (див. рис. 4):

```
aggr(na.data, prop=FALSE, numbers=TRUE)
```

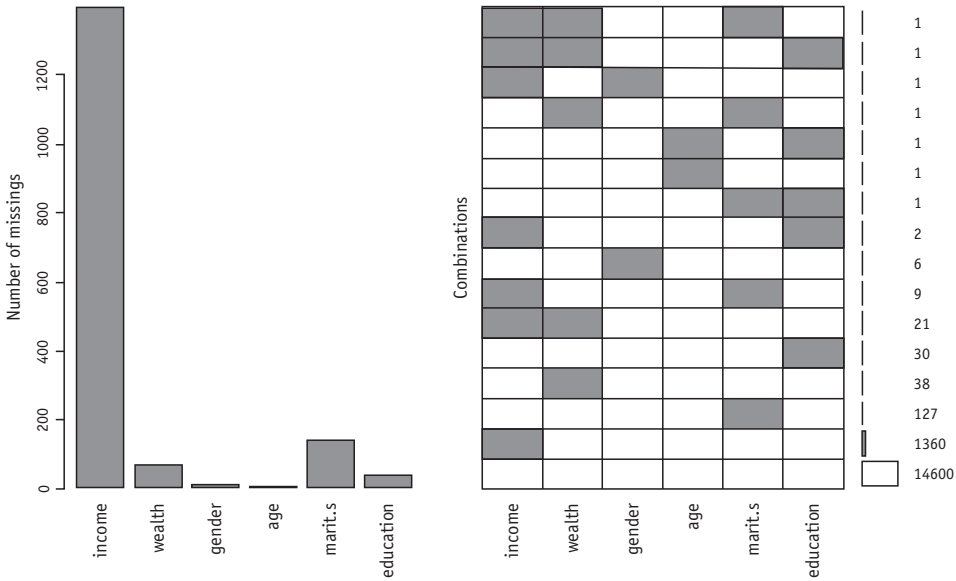

Рис. 4. Діаграма пропусків, побудована через функцію aggr()

Як можна бачити, основна частина пропусків міститься саме у змінній про доходи респондента. Водночас пропуски за цією змінною доволі рідко зустрічаються разом із пропусками за іншими змінними (якби такі ситуації були присутні, їх варто було б вивчити додатково).

Далі слід перевірити, чи пов'язаний розподіл пропусків для змінної про доходи із розподілами значень інших змінних. Для цього створюють нову змінну, що містить одиниці для відсутніх даних для змінної "income" і нулі для спостережень із відповідями. Далі знайдемо силу зв'язку між отриманою змінною й іншими змінними масиву:

```
na.data$income.na <- abs(is.na(na.data$income))
round(cor(na.data[c(4,7)], use="pairwise.complete.obs"),2)
              age    income.na
       age    1.00       −0.17
income.na   −0.17         1.00

special.table <- function(x) prop.table(table(x, na.data$income.na),1)
na.tables <- lapply(na.data[-c(1,4,7)], special.table)
for (i in c("wealth","gender","marit.s","education")) {
    print(round(na.tables[[i]],2))
}
```

x	0	1
Злиденне	0.94	0.06
Бідне	0.93	0.07
Середнє	0.90	0.10
Заможне	0.83	0.17[1]

x	0	1
Чоловіча	0.91	0.09
Жіноча	0.92	0.08

x	0	1
Ніколи не був (-ла) у шлюбі	0.86	0.14
Перебуваю у шлюбі	0.91	0.09
Розлучений, розійшовся	0.92	0.08
Вдівець (вдова)	0.98	0.02

x	0	1
Початкова, неповна середня	0.94	0.06
Середня загальна	0.89	0.11
Середня спеціальна	0.91	0.09
Перший ступінь вищої освіти	0.92	0.08
Повна вища освіта	0.92	0.08

[1] З поданих результатів виключено категорію "Багате", оскільки відповідна група респондентів гранично мала, а їхні відповіді про доходи свідчать про вкрай скромне фінансове становище.

Коротко спинімося на синтаксисі. У першому рядку через функції is.na() і abs() повертаємо одиниці для пропусків і нулі для наявних значень, а результати записуємо у змінну "income.na". У другому рядку через функцію cor() підраховуємо коефіцієнт кореляції Пірсона для отриманої змінної та віку (r_s = 0,17). Відповідні результати округляємо через функцію round().

Далі створюємо спеціальну функцію special.table(), призначену для побудови таблиць спряженості між змінною "income. na" і категоріальними змінними масиву. Після цього через функцію lapply() створюємо і записуємо у список na.tables відповідні таблиці. Наостанок за допомоги циклу for() таблиці виводимо на консоль.

Ми не випадково обрали таблиці спряженості для категоріальних змінних — вони відчутніше демонструють наявний взаємозв'язок порівняно з коефіцієнтами, обчислюваними для табличних даних. Як можна бачити, матеріальне становище і сімейний статус впливають на ймовірність невідповіді на запитання про доходи (можна було б використовувати критерій незалежності χ^2 для перевірки значимості, але на таких великих вибірках він утрачає свою інформативність). Також існує слабкий взаємозв'язок між невідповідями та віком. Ці результати надають більше свідчень для вибору ВП-, а не ЦВП-припущення. Звісно, у даному разі висока ймовірність НП-припущення, оскільки йдеться про таку чутливу тему, як заробіток (про це свідчить і відповідна таблиця спряженості).

Далі подано тільки використання множинного заміщення (на випадок прийняття ВП-припущення), оскільки методи опрацювання пропусків за НП-припущення в *R*, як і в інших статистичних пакетах, не реалізовані, а відповідне завдання в рамках цієї статті перед нами не стоїть.

Для демонстрації множинного заміщення ми застосували підхід, реалізований у бібліотеці mice. Він складається із трьох етапів: 1) через функцію mice() моделюється набір масивів для тих значень, що пропущені (за умовчанням таких масивів п'ять); 2) через функцію with() до кожного з отриманих на попередньому етапі масивів, сполученого з основним масивом, застосовується необхідний статистичний метод (наприклад, лінійна регресія); 3) через функцію pool() об'єднуються результати, отримані для кожного з масивів на попередньому етапі. Далі наведено лише відповідний синтаксис без самих результатів (результатом у цьому разі буде звичне зведення на підставі лінійної регресії):

```
mice.data <- mice(na.data)
fit <- with(mice.data, lm(income ~ wealth))
pooled <- pool(fit)
summary(pooled)
```

Зрештою, необхідно сказати про реалізацію в *R* порядкового і попарного видалення спостережень з аналізу, придатних при прийнятті ЦВП-припущення і невеликій частці самих пропусків (менш як 10%). Порядкове видалення можна виконати через функцію complete.cases() або na.omit():

```
final.na.data <- na.data[complete.cases(na.data),]
final.na.data <- na.omit(na.data)
```

Що стосується попарного видалення, то можливість його використання зазвичай закладена в інструментарій самих функцій. Наприклад, трохи раніше в цій статті було використано таку команду:

```
cor(na.data[c(4,7)], use="pairwise.complete.obs")
```

У цьому разі арґумент "use" відповідає за специфіку опрацювання пропущених значень, а його варіант "pairwise.complete.obs" забезпечує попарне видалення. Відповідно, за необхідності здійснення попарного видалення пропусків у рамках використання конкретного статистичного методу в *R* слід правильно сконфіґурувати відповідальний за це арґумент. Щоб дізнатися, як це зробити, треба звернутися до відповідної документації.

Висновки

На сучасному етапі розвитку того, що заведено називати "наукою про дані" (*Data Science*)[1] , багато інструментів управління (даними) переходять із категорії маловідомих і малозастосовуваних до категорії обов'язкових до застосування, незалежно від сфери діяльності вченого. Це передусім пов'язане з розвитком програмних засобів реалізації таких інструментів. І якщо застосування в загальному вигляді різних підходів управління даними в контексті застосування можливостей сучасних мов програмування не висуває на даному етапі нерозв'язуваних проблем (частково це підтверджується і змістом даної статті), то вивчення специфіки різноманітних типів соціологічного дослідження з метою розроблення програмних рішень є одним із пріоритетних завдань сучасної емпіричної соціології.

Джерела

Бослаф С. Статистика для всех / Сара Бослаф. — М. : ДМК Пресс, 2015. — 586 с.

Головаха Е. Интегральный индекс социального самочувствия (ИИСС): конструирование и применение социологического теста в массовых опросах / Евгений Головаха, Наталия Панина. — К. : Ин-т социологии НАН Украины, 1997. — 64 с.

Дембицкий С. Существует ли "опасная зона" в Data Science? [Электронный ресурс] / Сергей Дембицкий. — Режим доступа : http://soc-research.info/blog/index_files/danger_zone.html.

Кабаков Р. R в действии / Роберт Кабаков. — М. : ДМК Пресс, 2014. — 580 с.

Шипунов А. Наглядная статистика. Используем R! / Алексей Шипунов, Евгений Балдин, Полина Волкова, Антон Коробейников, София Назарова, Сергей Петров, Вадим Суфиянов. — М. : ДМК Пресс, 2012. — 298 с.

Beaver M. Survey Data Cleaning Guidelines: SPSS and Stata (Working Paper) [Electronic resource] / Margaret Beaver. — Access mode : http://ageconsearch.umn.edu/bitstream/123553/2/idwp123.pdf.

[1] Хоча, на думку одного з авторів цієї статті (Сергія Дембіцького), слід говорити не про *Data Science*, а про *Data Toolkit*, тобто про інструментарій для роботи з даними (докл. див.: [Дембицкий, s.a.]).

Cheema J. A Review of Missing Data Handling Methods in Education Research / Jehanzeb Cheema // Review of Educational Research. — 2014. — № 4. — P. 487–508.

Enders C. Applied Missing Data Analysis / Craig Enders. — New York ; London : The Guilford Press, 2010. — 377 p.

Leek J. Getting and Cleaning Data [Electronic resource] / Jeff Leek, Roger Peng, Brian Caffo. — Access mode: https://www.coursera.org/course/getdata.

Newman D. Missing Data: Five Practical Guidelines / Daniel Newman // Organizational Research Methods. — 2014. — № 4. — P. 372–411.

Peugh J. Missing Data in Educational Research: A Review of Reporting Practices and Suggestions for Improvement / James Peugh, Craig Enders // Review of Educational Research. — 2004. — № 4. — P. 525–556.