УДК 303.064

СЕРГЕЙ ДЕМБИЦКИЙ,
кандидат социологических наук, научный сотрудник отдела методологии и методов социологии Института социологии НАН Украины

ТАТЬЯНА ЛЮБИВАЯ,
кандидат социологических наук, младший научный сотрудник отдела методологии и методов социологии Института социологии НАН Украины

Современные техники управления данными социологических исследований

Аннотация

Статья посвящена трем ключевым компонентам управления данными массовых социологических опросов: 1) логическому контролю ("чистке") массива; 2) модификации данных; 3) работе с пропущенными значениями. В первой части рассмотрены основы технической и логической "чистки" массива. Во второй — описаны способы преобразования данных в контексте определенных исследовательских ситуаций, а также техники перекодировки имеющихся и вычисления новых переменных. В третьей приведены методологические положения обработки пропущенных значений и некоторые методы работы с ними.

Особенностью статьи является ее практическая направленность — все рассматриваемые способы управления данными опираются на примеры, которые, в свою очередь, содержат команды на языке программирования R, получившем широкое распространение в научных кругах в качестве среды статистических вычислений и построения графиков.

Ключевые слова: *управление данными, "чистка" массива, пропущенные значения, R*

Введение

Общеизвестен тот факт, что во многих случаях подготовка к анализу уже собранных количественных данных может занимать значительно больше времени и забирать значительно больше сил, чем сам статистический анализ. Несмотря на это, в рамках отечественной эмпирической социологии проблема управления данными является в некотором смысле банальной. Соответствующие инструменты составляют само собой разумеющийся минимум, необходимый для успешной работы социолога количественного профиля. Вместе с тем спектр задач, решаемых в рамках управления количественными данными, значительно шире, чем может себе представить среднестатистический украинский социолог. Поэтому целью данной статьи является расширение концептуального поля отечественной социологии в части основ управления количественными данными, а также обогащение ее инструментария программными возможностями языка программирования *R*.

Статья предполагает некоторое (хотя бы начальное) знакомство с *R*[1], поскольку все примеры опираются на его использование. Большая часть команд, приведенных в работе, достаточно просты для понимания. Однако есть и синтаксис, предполагающий знакомство с *R* на уровне выше начального. Но, независимо от сложности, все приведенные здесь команды после незначительной модификации могут быть использованы в других исследовательских проектах.

Также стоит отметить, что примеры статьи основываются на использовании данных социологического мониторинга "Украинское общество", который проводится Институтом социологии НАН Украины с 1994 года.

Логический контроль ("чистка") массива

Очень часто специалисты, которые вводят данные и формируют массив, не являясь аналитиками, не обращают внимания на содержательное соотношение ответов в пределах каждой из анкет (например, относительно возраста и образования), поскольку процесс ввода данных часто доведен ими до автоматизма. Кроме того, когда оператор недостаточно опытен или внимателен, плохо ознакомился с анкетой или паспортом для ввода данных, в массив могут попасть значения, отличающиеся от соответствующих в анкете. Поэтому непосредственно после ввода данных количественных исследований возникает необходимость логического контроля массива. Под логическим контролем подразумевается проверка данных массива на предмет присутствия в них логических несоответствий. Например, если человек говорит, что его возраст 16 лет и при этом он имеет высшее образование, следует проверить, не является ли это ошибкой ввода.

В отечественной литературе этап логического контроля данных практически не описан. Его основы и правила написания соответствующего синтаксиса приведены в справке статистического пакета "OCA for Windows" (справка доступна из самой программы). В зарубежной литературе контро-

[1] Если читатель почувствует необходимость в дополнительном изучении основ *R*, мы рекомендуем отличную книгу Роберта Кабакова "*R* в действии" [Кабаков, 2014].

лю данных уделяется значительно больше внимания[1]. На популярном образовательном портале "Coursera" (https://www.coursera.org/) этой теме посвящен отдельный курс [Leek, s.a.].

Правильно проведенный контроль массива данных является одним из ключевых условий их качества. Например, в банк данных Европейского социального исследования (ESS, http://www.europeansocialsurvey.org/) от стран-участниц принимаются лишь данные, прошедшие тщательный контроль.

Контроль массива данных также называют "чисткой массива". Последнее определение чаще используется в кругу отечественных исследователей, поэтому далее мы будем использовать именно его. Это крайне важный этап управления данными, который позволяет:

— определить ошибки ввода операторами;

— найти анкеты, которые могли быть фальсифицированы;

— подготовить массив для дальнейшей работы.

Последний пункт не является основной сферой, для которой используется чистка данных. Хотя практика показывает, что некоторые анкеты, содержащие множество логических противоречий, желательно проверить дополнительно, связавшись с респондентом (при наличии его контактов), если ранее эта анкета не попала в выборочный контроль после полевых работ.

Чистку массива можно условно разделить на техническую и логическую. Техническая чистка используется для: 1) проверки вопросов с множественными альтернативами: контроль максимального / минимального количества выбранных вариантов ответа (на тот случай, если количество категорий, которые необходимо выбрать, установлено точно или ограничено определенным количеством); 2) проверка, не выбран ли в случае вопросов этого же типа вариант "Ничего из перечисленного" вместе с другими вариантами ответа; 3) проверка правильности переходов после вопросов-фильтров.

Логическая чистка — более творческая. Если в случае технической чистки исследователь может руководствоваться исключительно требованиями к заполнению анкеты, то при логической чистке ему необходимо подключить свое воображение и знания, чтобы определить те конфигурации ответов, которые в логическом смысле противоречивы. Наличие таких ситуаций обычно указывает на ошибки ввода данных или фальсификацию анкет.

В первую очередь аналитику необходимо проверить, нет ли дублей анкет в массиве, то есть ввода одной анкеты два и более раз. Самый простой способ — это построение одномерного распределения по ключевому признаку[2]. Если по нему обнаружены дубли, следует удостовериться, что ответы на вопросы также идентичны, и удалить дублирующую анкету. Если же ответы на вопросы с одинаковыми номерами не идентичны, тогда ошибка ввода была в номере анкеты и каждой анкете надо присвоить уникальный

---

1 См., напр.: [Beaver, s.a.].

2 Ключевой признак (как правило, уникальный номер анкеты) является важным элементом обработки данных, который может использоваться не только для чистки. Например, при объединении разных переменных одного массива с помощью ключевого признака можно сопоставить наблюдения (если в различных подмассивах они идут в разном порядке).

номер. Для проверки идентичности ответов не по ключевому признаку, а по содержанию всех ответов можно использовать следующий 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("строка",takeData[i,"keyVar"],":",
                which(check == T)[which(check == T) != i], "\n")
    }
}
```

Сначала массив, требующий проверки, преобразуется в матрицу "takeData" и создается пустой вектор "shortlist". Далее посредством цикла for() для каждой строки матрицы с помощью функции apply() проверяется ее идентичность со всеми строками (в том числе и с самой собой) этой же матрицы, а результаты записываются в виде логического вектора "check". В данном случае "–70" указывает на то, что необходимо исключить переменную с семидесятым индексом, которая здесь является ключевым признаком (в противном случае каждое наблюдение будет уникальным). Если количество значений TRUE больше одного[1] (то есть содержание строки совпадает не только само с собой, но и содержанием, по крайней мере, еще одной строки), то в вектор "shortlist" записываются номера соответствующих строк, за исключением той, которая проверяется в данный момент. И если номера текущей строки нет в "shortlist", то с помощью функции cat() выводится результат, представляющий собой номер проверяемой в данный момент строки и тех строк, которые ее дублируют.

Проверив массив на наличие дублей, можно переходить непосредственно к чистке данных, в процессе которой аналитик формулирует ряд логических условий программными средствами. Результатом этой проверки является список с номерами анкет, в которых обнаружены логические противоречия. Когда в распоряжении исследователя находятся бумажные анкеты, он может обратиться к ним, чтобы понять, совершены ли соответствующие ошибки в процессе ввода данных оператором, и при необходимости исправить их. Также можно определить, не выделяются ли отдельные анкеты среди других по количеству ошибок. При наличии таковых их можно проконтролировать, связавшись с респондентом. Кроме того стоит проверить и другие анкеты соответствующего интервьюера.

Рассмотрим примеры технической (примеры 1 и 2) и логической (примеры 3 и 4) чистки.

---

[1] Обозначение логических операторов в *R* следующее: == (равно), < (меньше), <= (меньше или равно), > (больше), >= (больше или равно), != (не равно), | (или), & (и).

Пример 1. Вопрос с множественными альтернативами про членство в общественных и политических организациях не может одновременно содержать ответ про членство в одной или более одной организации из списка и вариант ответа "Не принадлежу ни к одной". Для проверки можно использовать следующий набор команд:

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

Сначала находится общее количество выборов общественных и политических организаций по каждому респонденту, которое записывается в переменную "b4_sum". Для этого используется функция rowSums(), относящаяся к категории функций (наряду с colSums(), rowMeans(), colMeans()), предназначенных для агрегирования информации в строках или столбцах массива. Далее с помощью двух логических условий, соответствующих описанным выше условиям, создается логический вектор "falserows_b4", в котором значения TRUE соответствуют тем строкам массива, в которых выявлено логическое противоречие, а FALSE — тем, в которых не выявлено. После этого с помощью функции which() создается вектор "wrong_b4", содержащий номера только тех строк (анкет), в которых выявлено логическое противоречие.

Пример 2. В мониторинге "Украинское общество" отсутствуют вопросы-фильтры. Поэтому представим, что респонденту, который заявляет о нежелании уезжать из населенного пункта, в котором он живет, не должен задаваться вопрос о возможном направлении переезда.

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

В первой команде кроме простого логического условия также используется отрицание результатов использования логической функции is.na(). Использование этой функции самой по себе возвращает TRUE для наблюдений с пропусками и FALSE для остальных наблюдений по соответствующей переменной (в нашем случае TRUE конвертируются в FALSE и наоборот).

Пример 3. Количество людей, с которыми респондент проживает в одной комнате, не должно быть больше количества людей, с которыми он проживает в одной квартире (доме).

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

Пример 4. Если в вопросе о виде занятости отмечен ответ "Не работаю и не имею никаких источников дохода", то в вопросе о размере заработной платы за последний месяц должен быть ноль.

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

Чем больше переменных используется в логических условиях, тем выше вероятность нахождения ошибки ввода или выявления фальсифицированной анкеты. В то же время стоит иметь в виду, что некоторые условия логической чистки не могут прямо говорить об ошибочном или неправильном ответе. В последнем примере респондент мог сказать о своём последнем личном доходе, а

не о доходе за последний месяц, или только в данном месяце он стал безработным без каких-либо источников дохода, а в прошлом месяце имел доход.

Кроме того, не стоит увлекаться такими "слабыми" условиями, как, например, "Люди с высшим образованием не могут быть полностью неудовлетворенными уровнем своего образования" или "Украинцы по национальности не могут исповедовать ислам". Хотя это и нестандартные ситуации, но вполне вероятные. Также следует придерживаться разумных пределов в количестве логических условий. Присутствие малой доли нелогичности в анкете вполне допустимо. Бывают ситуации, когда некоторые оценочные суждения в начале и в конце анкеты не совпадают, что может свидетельствовать о неопределенности мнения респондента по данному вопросу.

После реализации синтаксиса и получения перечня выявленных ошибок возможны два варианта дальнейших действий: 1) обращение к бумажным анкетам по их номеру и проверка правильности введенных данных (то есть сопоставление значений в массиве и анкете), исправление значений по необходимости; 2) работа исключительно с массивом данных.

Первый вариант является более предпочтительным, но тогда чистка занимает значительно больше времени. Второй вариант — менее затратный по времени, но в этом случае исследователю приходится принимать решение об изменении значений в переменных, содержащих несоответствия, или их исключении из анализа "на свой страх и риск".

В завершении этой части статьи мы приводим синтаксис, который структурирует работу, продемонстрированную в примерах, и приводит результаты к более удобному формату:

```
itogi <- list("ошибка в b4" = wrong_b4, "ошибка в i5" = wrong_i5,
              "ошибка в k7" = wrong_k7, "ошибка 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])
```

Сначала создается список "itogi" с информацией про пропуски, полученной в предыдущих четырех примерах. Далее создается вектор "inorder", в котором фиксируется порядок анкет в зависимости от количества найденных в них логических несоответствий (по убыванию). После этого создается еще один список "rawresult", в котором с помощью цикла for() и функции lapply() для каждой анкеты фиксируется наличие или отсутствие логического несоответствия в отношении каждой проверки. Наконец создается последний список "letsee", в котором для каждой анкеты остаются только те результаты, где зафиксировано наличие логических несоответствий. Этот финальный список может быть использован в качестве руководства для поиска и обработки анкет, попадающих под условия проверки.

Модификация данных

После того, как массив "почищен" и готов к работе, часто возникает необходимость модификации имеющихся переменных. В одних случаях это связано с общей логикой анализа (например, исключение из анализа так называ-

емых выбросов), в других — с требованиями используемого статистического метода (например, "подгонка" формы распределения), в третьих — с необходимостью получения новой информации на основании имеющейся (например, вычисление индексов). Рассмотрим эти ситуации более подробно.

В случае использования числовой переменной с большим диапазоном возможных значений следует проверить ее на наличие "выбросов" — нетипично больших или малых значений. Для визуализации последних можно использовать функцию `boxplot()`, позволяющую построить диаграмму размахов. В качестве данных создадим вектор значений "`salary`" на основании переменной "`l3`", исключив из нее респондентов без доходов и оставив только тех респондентов, которые были опрошены в рамках последнего этапа исследования, то есть в 2014 году.

```
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")
```

Полученная диаграмма (см. рис. 1) показывает нижнюю и верхнюю границы "выбросов", квартили, медиану и сами "выбросы" (они отмечены точками).

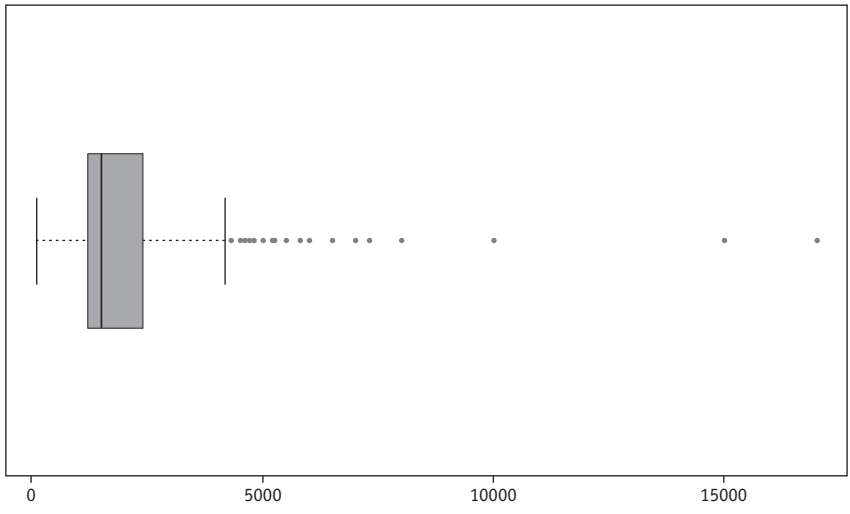

Рис. 1. Диаграмма размахов, построенная с помощью функции `boxplot()`

Для содержательной детализации числовых параметров этой диаграммы используются функции `boxplot.stats()` и `barplot()`. Первая показывает нижнюю границу "выбросов" (вместо нее может быть показано минимальное значение, если оно больше нижней границы), первый квартиль, медиану, второй квартиль, верхнюю границу "выбросов" (может быть показано максимальное значение, если оно меньше верхней границы)[1]. Вторая позво-

[1] Нижняя граница равна $Q1 - 1{,}5 \times \text{IQR}$, а верхняя — $Q3 + 1{,}5 \times IQR$, где $Q1$ — первый квартиль, $Q3$ — третий квартиль, IQR — межквартильная широта.

ляет проанализировать частотное распределение “выбросов” с помощью столбчатой диаграммы (см. рис. 2).

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

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

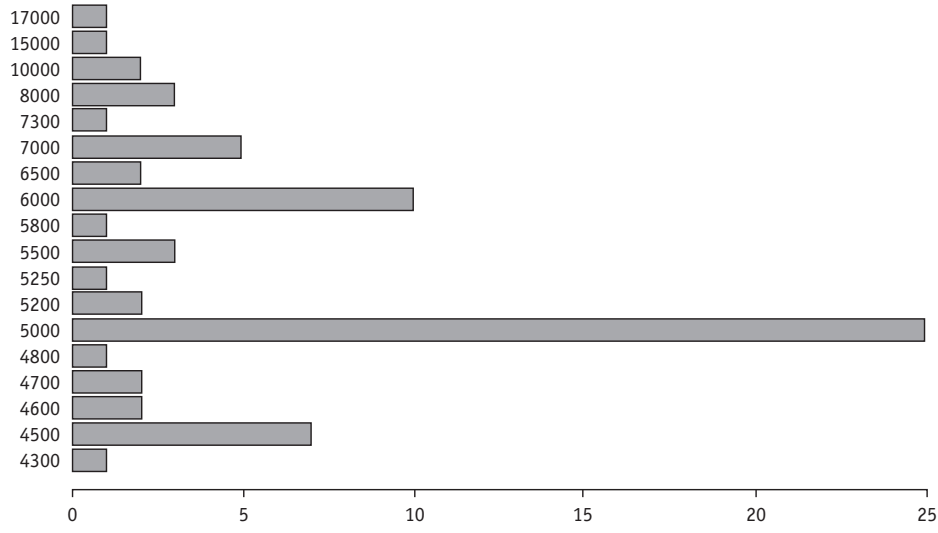

Рис. 2. Столбчатая диаграмма, построенная с помощью функции barplot()

Решение о том, что же в итоге считать выбросом, всегда является несколько субъективным и основывается на интерпретации полученных результатов. В этом примере мы решили, что выбросами будут все значения равные или превышающие 10 000. Чтобы создать новую переменную, не содержащую выбросов, можно использовать следующую команду:

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

Порой возникает необходимость преобразования той или иной переменной исходя из требований, предъявляемых к анализируемым данным со стороны используемого метода. Также может потребоваться приведение различных шкал к одной размерности. Примерами таких ситуаций являются, соответственно, использование кластерного анализа и построение линейных графиков. Так, в рамках кластерного анализа переменные с более широким диапазоном значений будут иметь большее влияние на образование кластеров, что далеко не всегда оправданно. Что касается построения линейных графиков, то анализ переменных с более узким диапазоном значений будет затруднен. В таких случаях подойдет нормализация или расчет *Z*-значений.

Допустим, мы хотим разделить респондентов на группы в зависимости от их роста и веса. Для нормализации в *R* используется функция scale():

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

Переменная g7 включает информацию про рост респондента (интерквартильная широта равна 96 см), а g8 — про вес (интерквартильная широта равна 17 кг). После стандартизации влияние размерности шкалы на результаты кластерного анализа будет нивелировано (в данном случае мы исходим из того, что обе переменные одинаково важны в процессе разделения респондентов на группы).

Теперь рассмотрим пример, касающийся изменения уровней интернальности/экстернальности (интерналы объясняют изменения в своей жизни внешними и не зависящими от них обстоятельствами, экстерналы, наоборот, берут ответственность на себя) и материальной обеспеченности респондентов в целом по стране. Для анализа использованы линейные графики. Но прежде чем их построить, необходимо рассчитать соответствующие индексы. В случае оценки материального положения семьи[1] достаточно рассчитать среднее значение для каждого года. Для этого можно использовать функцию aggregate():

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

Что касается индекса экстернальности/интернальности, то расчет будет более сложным. Сначала создадим и запишем в объект lineGraph таблицу сопряженности (см. табл.) с помощью следующего синтаксиса[2]:

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

Таблица

Данные для расчета индекса экстернальности/интернальности

Год	Варианты ответа				
	В основном от внешних обстоятельств	В некоторой мере от меня, но больше от внешних обстоятельств	В одинаковой мере от меня и от внешних обстоятельств	В большей мере от меня, чем от внешних обстоятельств	В основном от меня
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 Используется шкала от 0 до 10, где 0 — самая низкая оценка, 10 — самая высокая.

2 Внешний вид таблицы отредактирован в соответствии с форматом статьи.

Теперь для каждого года необходимо из суммы наблюдений, попадающих в первые две категории, отнять сумму для двух последних, а полученное значение разделить на общее количество наблюдений, после чего можно стандартизировать индекс:

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

Далее приведены два графика (см. рис. 3): первый построен на основании "сырых" индексов, второй — нормализированных. В первом случае связь переменных практически невозможно проследить, во втором она выражена очень отчетливо.

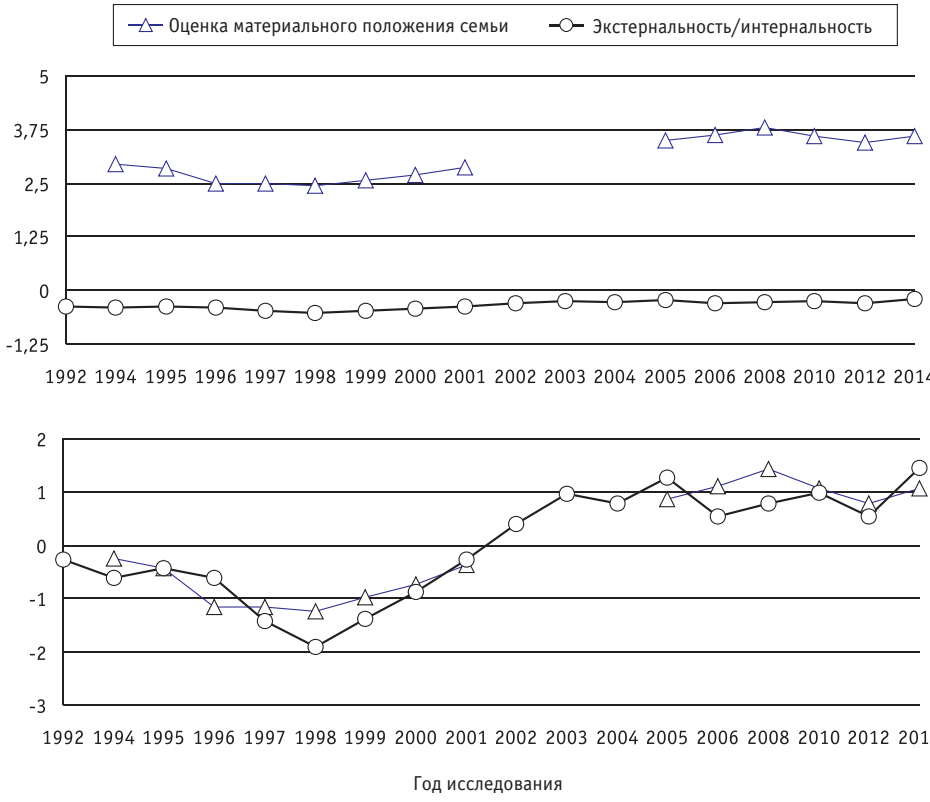

Рис. 3. Линейный график до и после нормализации значений

Другими распространенными способами преобразования являются логарифмическое, квадратного корня, обратное и квадратное [Шипунов, 2014: с. 62–64]. Логарифмическое преобразование может дать нормальное распределение, когда оно скошено вправо. Кроме того, оно делает зависимости между переменными более линейными и уравнивает дисперсии. Поскольку логарифмическое преобразование непригодно для работы с нулями, рекомендуется прибавлять единицу:

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

Преобразование квадратного корня похоже по действию на логарифмическое. При этом оно может работать с нулями, но не может — с отрицательными значениями:

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

Обратное преобразование используется в случае необходимости стабилизации дисперсии. Так же, как и логарифмическое, оно предполагает добавление единицы:

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

Квадратное преобразование подходит для случаев, когда распределение скошено влево (может привести к нормальному). Оно линеаризует зависимости и выравнивает дисперсии:

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

Кроме описанных способов модификации, часто возникает необходимость расчета новых переменных (как на основании условий специальных методик, которые применялись в рамках исследования, так и, с учетом своего опыта, на основании отдельных переменных), а также перекодировки имеющихся (например, распределение респондентов по возрастным категориям или регионам проживания).

При необходимости дихотомизации числовой переменной можно использовать функцию ifelse(). В качестве входящих данных опять использованы ответы респондентов в 2014 году про доход за последний месяц. При этом не исключены те, у кого доход отсутствует:

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

Функция ifelse() принимает три аргумента: 1) логическое условие (в нашем случае — доход больше 1500); 2) значение, возвращаемое функцией в случае истинности условия для респондента (в нашем случае — ноль); 3) значение, возвращаемое в случае ложности условия для респондента (в нашем случае — 1).

Вторым простым способом разбивки числовых переменных является автоматизированная процедура категоризации с помощью функции bin.var(), позволяющая применить один из трех методов: 1) построение интервалов равных размеров (величина интервала равна $Xmax/n$, где $Xmax$ — максимальное значение в распределении, n — количество интервалов); 2) построение интервалов одинаковой наполненности (по N/n респондентов в каждой группе, где N — размер выборки, n — количество интервалов); 3) построение интервалов на основании кластеризации методом k-средних. Сначала рассмотрим общую форму соответствующей команды:

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

В качестве первого аргумента указывается числовая переменная массива или вектор (как в нашем случае), в качестве второго (bins) — количество интервалов, которые необходимо построить, в качестве третьего (method) — метод разбивки, с помощью последнего (labels) задается формат наименова-

ния групп. Теперь проанализируем, к каким результатам приводит каждый из трех методов (для наглядности результаты приводятся в отредактированном виде):

```
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
```

Как видно, в случае анализа доходов наиболее сбалансированную структуру (как в смысле величины интервалов, так и в смысле наполненности категорий) обеспечивает использование третьего метода.

В случае перекодировки номинальных переменных, а также в случае необходимости полного контроля при создании нескольких категорий на основании числовой переменной наилучшим способом является использование функции within(). Для примера рассмотрим перекодировку переменной о семейном статусе респондента с шестью категориями: 1 — "Никогда не был(-а) в браке", 2 — "Состою в зарегистрированном браке", 3 — "Состою в фактическом, незарегистрированном браке", 4 — "Разведен(-а) официально", 5 — "Разошелся(-ась), хотя официально не разведен(-а)", 6 — "Вдовец (вдова)". Новая переменная будет иметь четыре категории: "Никогда не состоял(-а) в браке" (первая категория), "Состою в браке (гражданском или фактическом)" (вторая и третья категории), "Разошелся или разведен" (четвертая и пятая категории), "Вдовец (вдова)" (шестая категория). Синтаксис *R* следующий:

```
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
})
```

В первой строке мы указываем название массива, в котором осуществляются преобразования (finalUSind), во второй — присваиваем новой переменной пропуски (этот шаг обязательный), в дальнейших с помощью логических выражений определяем условия для сопоставления старых и новых категорий. Используя другие логические операторы и "перевернув" порядок присвоения значений, можно сократить приведенный 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
})
```

Еще одним альтернативным вариантом в данном случае будет такой (здесь порядок присвоения не играет роли):

```
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
})
```

Когда необходимо вычислить новую переменную на основании имеющихся, используются привычные математические операции[1]. Также могут использоваться и встроенные в *R* функции. Для начала рассмотрим построение индекса пропорциональности веса. Он рассчитывается как отношение веса тела человека в килограммах к квадрату его роста в метрах. Исходя из того, что в массиве рост зафиксирован в сантиметрах, а вес — в килограммах, синтаксис может принять один из двух вариантов:

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

В случае, когда необходимо вычислить большее количество новых переменных, удобно использовать функцию transform(). Например, при переводе переменных, содержащих информацию в гривнях (доход респондента, минимально приемлемый уровень дохода по его оценке, уровень дохода, при котором, по мнению респондента, можно жить хорошо и т.д.), использование этой функции будет выглядеть следующим образом:

```
finalUSind <- transform(finalUSind,
                        l3_dollar = l3 / currency,
                        l5_dollar = l5 / currency,
                        l6_dollar = l6 / currency,
```

[1] Обозначение основных математических операторов в *R* следующее: + (сложение), – (вычитание), × (умножение), / (деление), ^ или ** (возведение в степень).

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

В первой строке указан массив (до знака присваивания), в который записываются новые переменные и из которого берутся переменные для расчета (в качестве первого аргумента функции). Далее приведены выражения для расчета всех необходимых переменных (переменная "currency" несет в себе информацию про курс доллара).

Во многих случаях при расчете новых переменных очень удобно использовать встроенные, то есть доступные в *R* по умолчанию функции. Хорошим примером является расчет Интегрального индекса социального самочувствия, разработанного Евгением Головахой и Наталией Паниной [Головаха, 1997]. На первом этапе расчета этого индекса необходимо найти сумму значений по 20-ти индикаторам. Сначала рассмотрим синтаксис без использования встроенной функции:

```
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))
```

В целях компактности здесь можно использовать функцию rowSums(), но перед этим все индикаторы необходимо привести к числовому виду (функция принимает только числовые переменные, а сейчас они являются переменными-факторами). Для этого мы прибегаем к функции sapply(), использование которой заключается в применении заданного метода к каждому компоненту списка (массив является его частным случаем), результатом чего является вектор значений[1]:

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

Здесь обращение к переменным осуществляется не с помощью имен переменных, а с помощью их индексов. Переменным с f6_1 по f6_20 соответствуют индексы с 184-го по 203-й. Как видно, в данном случае для суммирования индикаторов понадобилось значительно меньше строк кода.

---

[1] Найти правильное решение здесь помогли Кирилл Захаров и Александр Виноградов, за что авторы статьи выражают им особую благодарность.

Далее необходимо скорректировать полученную переменную (f6_sum) с учетом того, что четвертая категория ("Не интересует") при расчете индекса должна быть принята за двойку. Следовательно, сумма двадцати индикаторов должна давать максимум 60, а не 80 баллов. Для решения этой задачи необходимо подсчитать для каждого респондента количество четверок и вычесть соответствующее значение, умноженное на два из суммарного индекса. Здесь мы используем цикл for():

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

Первая команда в теле цикла создает набор значений, состоящих из TRUE (истина), если значение переменной равно четырем, и FALSE (ложь), если нет. Поскольку истина в математических операциях принимается за единицу, а ложь — за ноль, легко посчитать количество четверок по каждому респонденту (для этого используется переменная f6_min). Последняя строка синтаксиса удаляет счетчик четверок, поскольку для дальнейшей работы он не нужен.

Работа с пропущенными значениями

Многие переменные (а в социологических исследованиях, основывающихся на массовых опросах, они нередко составляют большинство) имеют пропущенные значения. Последние ведут к снижению статистической мощности, а также могут быть причиной систематических ошибок [Бослаф, 2015: с. 450].

Обработка пропущенных значений является достаточно развитой исследовательской областью с общепринятой терминологией и множеством решений для различных дисциплин и конкретных исследований. С попыткой широкого обобщения основ обработки пропущенных данных в социальных науках можно ознакомиться, например, в работе Даниэля Ньюмана [Newman, 2014]. В рамках же данной статьи мы обратимся к главным понятиям этой теории, а также основным методам решения проблемы пропущенных значений.

Принято выделять три вида пропусков — полностью случайные, случайные и неслучайные. Данная терминология ведет свое начало от известной работы Дональда Рубина [Rubin, 1976]. Полностью случайные пропуски (ПСП-допущение) имеют место в тех случаях, когда подвыборка имеющихся значений по переменной(-ым), подлежащей(-им) изучению, попрежнему является моделью генеральной совокупности. Примером может служить случай, когда пропуски по некоторой переменной (например, политические предпочтения) не зависят от значений переменных-предикторов (например, пол, возраст, регион проживания и т.д.), а также от значений самих пропусков (например, не возникает ситуации, когда респонденты с определенной политической позицией чаще других не дают ответа на соот-

ветствующий вопрос). Выбор модели полностью случайных пропусков — единственное допущение, которое можно проверить эмпирически. Что касается случайных и неслучайных пропусков, соответствующие допущения невозможно проверить на основании имеющегося массива.

При случайных пропусках (СП-допущение) их значения зависят от значений переменных-предикторов и не зависят от собственных значений пропусков. Так, если пропуски в ответах на вопрос о политических предпочтениях чаще встречаются среди людей старшего возраста (но внутри этой группы они распределены случайно), то речь идет о случайных пропусках. В этом случае возникает вероятность смещения результатов оценивания параметров по выборке в целом (если значение по соответствующей подгруппе отличается от общего среднего).

Если же вероятность пропусков по определенным переменным зависит от величины самих пропущенных значений по этим переменным, то говорят о неслучайных пропусках (НП-допущение). Например, люди с левыми политическими взглядами с меньшей вероятностью (чисто гипотетическое допущение) склонны сообщать соответствующую информацию. Такие пропуски вносят систематические ошибки в результаты анализа.

Методы обработки пропущенных значений разделяют на традиционные и современные. К первым относят построчное и попарное удаление наблюдений, замещение средним и замещение с использованием регрессии. Ко вторым — замещение посредством оценки максимального правдоподобия, множественное замещение, селективную модель и модель смешанных паттернов.

При построчном удалении из массива исключаются любые наблюдения, в значениях переменных которых присутствует хотя бы один пропуск. Такой подход может использоваться только при ПСП-допущении. Но даже в такой ситуации его использование ведет к снижению статистической мощности.

Как и в случае построчного удаления, попарное удаление подходит исключительно при ПСП-допущении. Попарное удаление заключается в том, что в каждом конкретном случае анализа из него удаляются только те наблюдения, в которых присутствует хотя бы один пропуск по релевантным для данного анализа переменным. Как и в случае построчного удаления, оно подходит исключительно при ПСП-допущении. Тем самым создается дополнительная проблема, заключающаяся в использовании отличающихся выборок в рамках проверки различных гипотез одного исследовательского проекта.

Использование замещения пропущенных значений выборочным средним не рекомендуется в любом случае. Это связано с тем, что такой способ обработки пропусков ведет к смещенным оценкам (поскольку уменьшает дисперсию переменных и ковариацию между ними), независимо от используемого допущения. Исключением является оценка самого среднего значения.

Замещение с использованием регрессии (вместо пропущенных используются предсказанные значения) обусловливает схожие с предыдущим способом проблемы. Так, получаемые значения попадают в один паттерн, отвечающий регрессионной прямой, что вносит смещение как в смысле дисперсии, так и в смысле ковариации. Частично эта проблема может быть решена посредством добавления случайных остаточных значений к предсказанным величинам. В целом же данный способ замещения подходит при СП-допущении [Peugh, 2004; Cheema, 2014].

Замещение посредством оценки максимального правдоподобия представляет собой итерационный процесс, каждая итерация которого включает два этапа: 1) средние значения переменных и ковариационная матрица используются для расчета регрессионных уравнений, предсказывающих пропущенные значения; 2) полученный массив используется для расчета обновленных средних значений и ковариационной матрицы. Эти этапы повторяются до тех пор, пока средние значения и ковариационная матрица не перестанут изменяться. Можно сказать, что в данном случае используется последовательный подход.

В случае множественного замещения создается набор альтернативных массивов данных (от пяти и более), в которых пропущенные значения замещаются предсказанными посредством регрессии с добавлением случайного элемента. После этого средние значения переменных и ковариационные матрицы обобщаются с целью получения итоговой оценки. Этот подход можно назвать конкурентным.

Как замещение посредством максимального правдоподобия, так и множественные замещения подходят для ситуаций, в которых исследователь исходит из ПСП- или СП-допущений. Для случаев, когда делается НП-допущение, предназначены селективная модель и модель смешанных паттернов.

Селективная модель соединяет основное регрессионное уравнение с дополнительным регрессионным уравнением, предсказывающим вероятность ответов. Две части модели связываются посредством скоррелированных остатков (correlated residuals), и эта связь является механизмом, с помощью которого корректируются смещения модели, связанные с пропусками.

Модель смешанных паттернов предполагает формирование подгрупп наблюдений, в которых обнаруживаются одинаковые паттерны пропусков данных, с дальнейшей оценкой интересующих параметров в каждой подгруппе. После этого находится средневзвешенная оценка данных параметров.

В отличие от современных методов обработки пропущенных значений при ПСП- или СП-допущениях, методы обработки при НП-допущении не получили достаточно широкого признания [Enders, 2010: p. 56–328].

Обзор техник работы с пропущенными данными в *R* сделан в работе Роберта Кабакова [Кабаков, 2014: с. 472–498]. Здесь мы коснемся лишь некоторых из них. В качестве примера проанализируем пропуски в переменной про доход респондента.

Прежде всего необходимо изучить структуру пропусков в соотношении с другими переменными массива (пол, возраст, семейный статус, образование, общая оценка материального состояния семьи). Для этого используем функцию md.pattern() библиотеки mice:

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

Результат будет иметь следующий вид:

	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

Первый столбец показывает общее количество наблюдений, соответствующих паттернам, представленным в строках под названиями переменных массива, то есть начиная со второй строки и ниже. Сами паттерны читаются следующим образом: нули говорят, что по этой переменной наблюдение имеет пропуск, единицы — пропуска нет. Завершающий столбец показывает общее количество пропусков у каждого такого наблюдения. Следовательно, 14 600 наблюдений массива не имеют пропусков (под всеми переменными стоит единица, завершающий столбец равен нулю); 1360 наблюдений имеют пропуск по переменной "доход" (income) и т.д. В последней строке показано общее количество пропусков по каждой переменной и общее количество пропусков по всему массиву (суммарное значение завершающего столбца).

Альтернативным способом представления тех же самых данных является построение специальной диаграммы с помощью функции aggr() библиотеки *VIM* (см. рис. 4):

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

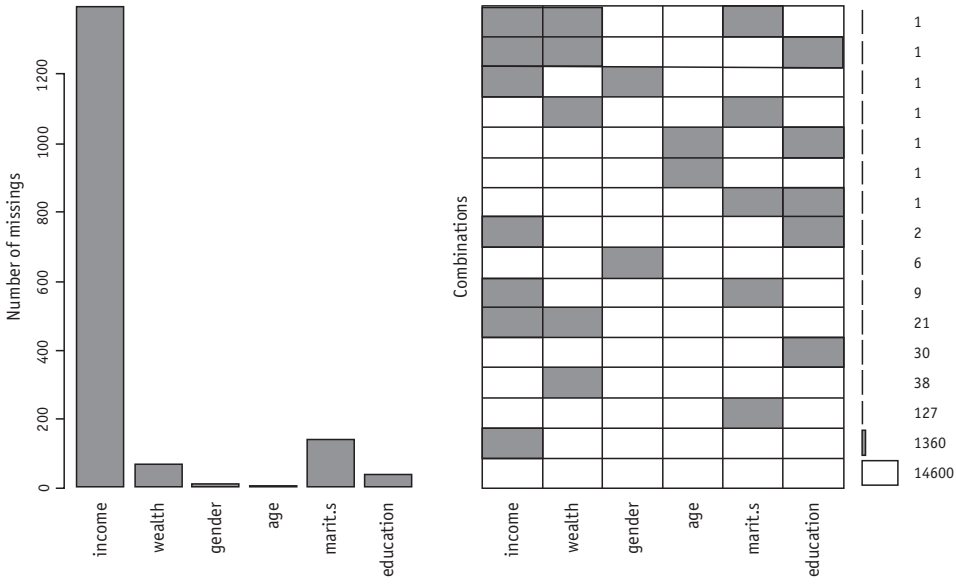

Рис. 4. Диаграмма пропусков, построенная с помощью функции aggr()

Как видно, основная часть пропусков содержится именно в переменной про доходы респондента. Вместе с тем пропуски по этой переменной достаточно редко встречаются совместно с пропусками по другим переменным (если бы такие ситуации присутствовали, их следовало бы изучить дополнительно).

Далее следует проверить, связано ли распределение пропусков для переменной о доходах с распределениями значений других переменных. Для этого создается новая переменная, содержащая единицы для отсутствующих данных по переменной "income" и нули для наблюдений с ответами. Далее найдем силу связи между полученной переменной и другими переменными массива:

```
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
Нищенское	0.94	0.06
Бедное	0.93	0.07
Среднее	0.90	0.10
Зажиточное	0.83	0.17[1]

x	0	1
Мужской	0.91	0.09
Женский	0.92	0.08

x	0	1
Никогда не был (-а) в браке	0.86	0.14
Состою в браке	0.91	0.09
Разведен, разошелся	0.92	0.08
Вдовец (вдова)	0.98	0.02

x	0	1
Начальное, неполное среднее	0.94	0.06
Среднее общее	0.89	0.11
Среднее специальное	0.91	0.09
Первая ступень высшего образования	0.92	0.08
Полное высшее образование	0.92	0.08

[1] Из представленных результатов исключена категория "Богатое", поскольку соответствующая группа респондентов предельно мала, а их ответы про доходы свидетельствуют о крайне скромном финансовом положении.

Кратко остановимся на синтаксисе. В первой строке с помощью функций is.na() и abs() возвращаются единицы для пропусков и нули для имеющихся значений, а результаты записываются в переменную "income.na". Во второй строке с помощью функции cor() подсчитывается коэффициент корреляции Пирсона для полученной переменной и возраста ($r_s = 0{,}17$). Соответствующие результаты округляются с помощью функции round().

Далее создается специальная функция special.table(), предназначенная для построения таблиц сопряженности между переменной "income.na" и категориальными переменными массива. После этого с помощью функции lapply() создаются и записываются в список na.tables соответствующие таблицы. В завершение с помощью цикла for() таблицы выводятся на консоль.

Мы не случайно выбрали таблицы сопряженности для категориальных переменных — они отчетливее показывают имеющуюся взаимосвязь по сравнению с коэффициентами, рассчитываемыми для табличных данных. Как видно, материальное состояние и семейный статус влияют на вероятность неответа на вопрос о доходах (можно было бы использовать критерий независимости χ^2 для проверки значимости, но на таких больших выборках он теряет свою информативность). Также существует слабая взаимосвязь между неответами и возрастом. Эти результаты предоставляют больше свидетельств для выбора СП-, а не ПСП-допущения. Конечно же, в данном случае высока вероятность НП-допущения, поскольку речь идет о такой чувствительной теме, как заработок (об этом говорит и соответствующая таблица сопряженности).

Далее представлено только использование множественного замещения (на случай принятия СП-допущения), поскольку методы обработки пропусков при НП-допущении в *R*, как и в других статистических пакетах, не реализованы, а соответствующая задача в рамках данной статьи перед нами не стоит.

Для демонстрации множественного замещения мы применили подход, реализованный в библиотеке mice. Он состоит из трех этапов: 1) с помощью функции mice() моделируется набор массивов для тех значений, которые пропущены (по умолчанию таких массивов пять); 2) с помощью функции with() к каждому из полученных на предыдущем этапе массивов, совмещенному с основным массивом, применяется необходимый статистический метод (например, линейная регрессия); 3) с помощью функции pool() объединяются результаты, полученные для каждого из массивов на предыдущем этапе. Далее приведен только соответствующий синтаксис без самих результатов (результатом в данном случае будет привычная сводка на основании линейной регрессии):

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

Наконец, необходимо сказать о реализации в *R* построчного и попарного удаления наблюдений из анализа, пригодных при принятии ПСП-допущения и небольшой доле самих пропусков (менее 10%). Построчное удаление можно выполнить с помощью функции complete.cases() или na.omit():

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

Что касается попарного удаления, то возможность его использования обычно заложена в инструментарий самих функций. Например, чуть ранее в этой статье была использована следующая команда:

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

В данном случае аргумент "use" отвечает за специфику обработки пропущенных значений, а его вариант "pairwise.complete.obs" обеспечивает попарное удаление. Соответственно, при необходимости осуществления попарного удаления пропусков в рамках использования конкретного статистического метода в *R* следует правильно сконфигурировать отвечающий за это аргумент. Чтобы узнать, как это сделать, следует обратиться к соответствующей документации.

Заключение

На современном этапе развития того, что принято называть "наукой о данных" (*Data Science*)[1], многие инструменты управления (данными) переходят из категории малоизвестных и малоиспользуемых в категорию обязательных к применению, независимо от сферы деятельности ученого. Это прежде всего связано с развитием программных средств реализации таких инструментов. И если использование в общем виде различных подходов управления данными в контексте применения возможностей современных языков программирования не выдвигает на данном этапе неразрешимых проблем (частично это подтверждается и содержанием данной статьи), то изучение специфики различных типов социологического исследования с целью разработки программных решений является одной из приоритетных задач современной эмпирической социологии.

Источники

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.

[1] Хотя, по мнению одного из авторов данной статьи (Сергея Дембицкого), следует говорить не о *Data Science*, а о *Data Toolkit*, то есть об инструментарии по работе с данными (подр. см.: [Дембицкий, s.a.]).

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.

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.