

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Новосибирский национальный исследовательский государственный университет»
(Новосибирский государственный университет, НГУ)
Структурное подразделение Новосибирского государственного университета –
Высший колледж информатики Университета (ВКИ НГУ)
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Направление подготовки: 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

Образовательная программа: 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

**СОЗДАНИЕ БИБЛИОТЕКИ АЛГОРИТМОВ ДЛЯ
СТАТИСТИЧЕСКОГО АНАЛИЗА ДАННЫХ КЛИНИЧЕСКИХ
ИССЛЕДОВАНИЙ В ПАРАЛЛЕЛЬНЫХ ГРУППАХ**

утверждена приказом по ВКИ НГУ № 02/3-204 от «27» апреля 2017 г.

Кошкаревой Софьи Владимировны, группа 14214 _____
(фамилия, имя, отчество студента) (подпись студента)

«К защите допущена»

Заведующий кафедрой, к.ф.-м.н
(ученая степень, звание)

Попов Л.К., / _____
(ФИО) / (подпись)

«__» _____ 20__ г.

Руководитель ВКР

к.ф.-м.н,
(ученая степень, звание)

Лукинов В.Л., / _____
(ФИО) / (подпись)

«__» _____ 20__ г.

Дата защиты: «__» _____ 20__ г.

Новосибирск
2017

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ ТЕРМИНОВ И СОКРАЩЕНИЙ	3
ВВЕДЕНИЕ	4
1. ПОСТАНОВКА ЗАДАЧИ	5
1.1. Описание предметной области	5
1.2. Формулировка задачи	7
1.3. Функциональные требования	7
1.4. Нефункциональные требования	7
1.5. Характеристики выбранных программных средств	8
2. РЕАЛИЗАЦИЯ	9
2.1. Алгоритмы решения задач	9
2.1.1. Опечатки	9
2.1.2. Выбросы	10
2.1.3. Генерация отчетов	11
2.2. Объектно-ориентированная модель	11
2.2.1. Файлы	12
2.2.2. Типы значений колонок таблицы	14
2.2.3. Типы ошибок	16
2.2.4. Описание реализации поиска опечаток	18
2.2.4.1. Метод поиска опечаток для дискретных значений	18
2.2.4.2. Метода поиска опечаток для непрерывных значений	20
2.2.4.3. Метод поиска опечаток для дат	23
2.2.5. Реализация поиска выбросов	25
2.2.6. Реализация поиска неупорядоченных дат	27
2.2.7. Создание сводной таблицы значений	27
3. РАЗРАБОТКА БИБЛИОТЕКИ	29
4. ОТЛАДКА И ТЕСТИРОВАНИЕ	31
РЕЗУЛЬТАТЫ	32
ЗАКЛЮЧЕНИЕ	35
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	36

ПЕРЕЧЕНЬ ТЕРМИНОВ И СОКРАЩЕНИЙ

Выбросы	Значение, которое слишком велико или слишком мало по сравнению с большинством других имеющихся значений;
Параллельные группы	это две или более группы испытуемых, одна или более из которых получают исследуемый препарат, а другая группа является контрольной;
Нормальное распределение	(распределение Гаусса) – это распределение вероятностей, совпадающее с функцией Гаусса;
ИКР	интерквартильный размах;

ВВЕДЕНИЕ

Целью данной работы является разработка программного обеспечения, предназначенного для проведения статистического анализа данных, полученных в результате клинических исследований.

На протяжении своего развития медицинское сообщество всегда старалось найти более эффективные способы лечения и диагностики болезней. Первоначальные методы были неэффективны из-за применения метода проб и ошибок и интуитивных обобщений. Для решения этой проблемы в медицине сформировалась новая область – доказательная медицина.

Доказательная медицина подразумевает такой подход к медицинской практике, при котором каждое решение, относящееся к выбору метода лечения, обязано иметь научное обоснование. Оно основывается на данных, полученных в ходе четко спланированного и документированного исследования, использующего методы статистического анализа.

Опечатки и пропуски отдельных значений в данных исследований являются постоянной проблемой. Поэтому, прежде чем начать применять статистические методы, обрабатываемые данные следует привести к приемлемому для обработки виду. Для этого необходимо идентифицировать возможные проблемы, а в дальнейшем, либо их удалить некорректные данные, либо заменить имеющиеся пропуски и опечатки разумными значениями, что и было реализовано в данной работе. Еще одной важной проблемой является наличие выбросов. Под «выбросом» понимается значение, которое слишком велико или слишком мало по сравнению с большинством других имеющихся значений.

Разработанная библиотека решает проблему подготовки входных данных для статистического анализа в автоматическом режиме. Библиотека сводит к минимуму затраченное на это время, а также позволяет биостатистикам анализировать данные исследования в удобной форме.

1. ПОСТАНОВКА ЗАДАЧИ

1.1. Описание предметной области

В большинстве случаев клинические исследования проводятся в параллельных группах. Параллельные группы – это две или более группы испытуемых, одна или более из которых получают исследуемый препарат, а другая группа является контрольной. Для достижения статистической достоверности испытуемые распределяются по группам методом случайной выборки.

При статистическом анализе данных клинических исследований необходимо выполнять следующие рутинные процедуры проверки входных данных, поддающиеся автоматизации:

- Поиск пропущенных значений (незаполненных полей);
- Исправление опечаток;
- Обнаружение выбросов;
- Выявление неупорядоченных дат;
- Оценка нормальности распределения данных.

На данный момент выполнение этих процедур происходит в ручном режиме. Такой подход является неэффективным из-за человеческой невнимательности. При таком подходе невозможно обработать большой объем данных за разумное время.

Поэтому программная проверка существенно ускорит дальнейший анализ, а так же окажется более эффективной по сравнению с проверкой, выполняемой в ручном режиме.

Методы статистического анализа делятся на:

- Параметрические;
- Непараметрические.

Параметрические методы имеют более высокую точность и эффективность по сравнению с непараметрическими, но имеют ограничения на входные данные – данные должны быть нормально распределены.

Нормальное распределение (или распределение Гаусса) – это распределение вероятностей, совпадающее с функцией Гаусса, вычисляемой по формуле рис 1:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

Рис 1. – функция Гаусса

Где

μ — математическое ожидание (среднее значение),

σ — среднеквадратическое отклонение.

Стандартным нормальным распределением называется нормальное распределение с математическим ожиданием $\mu = 0$ и стандартным отклонением $\sigma = 1$. Стандартное распределение отображено на рис 2 зеленой линией.

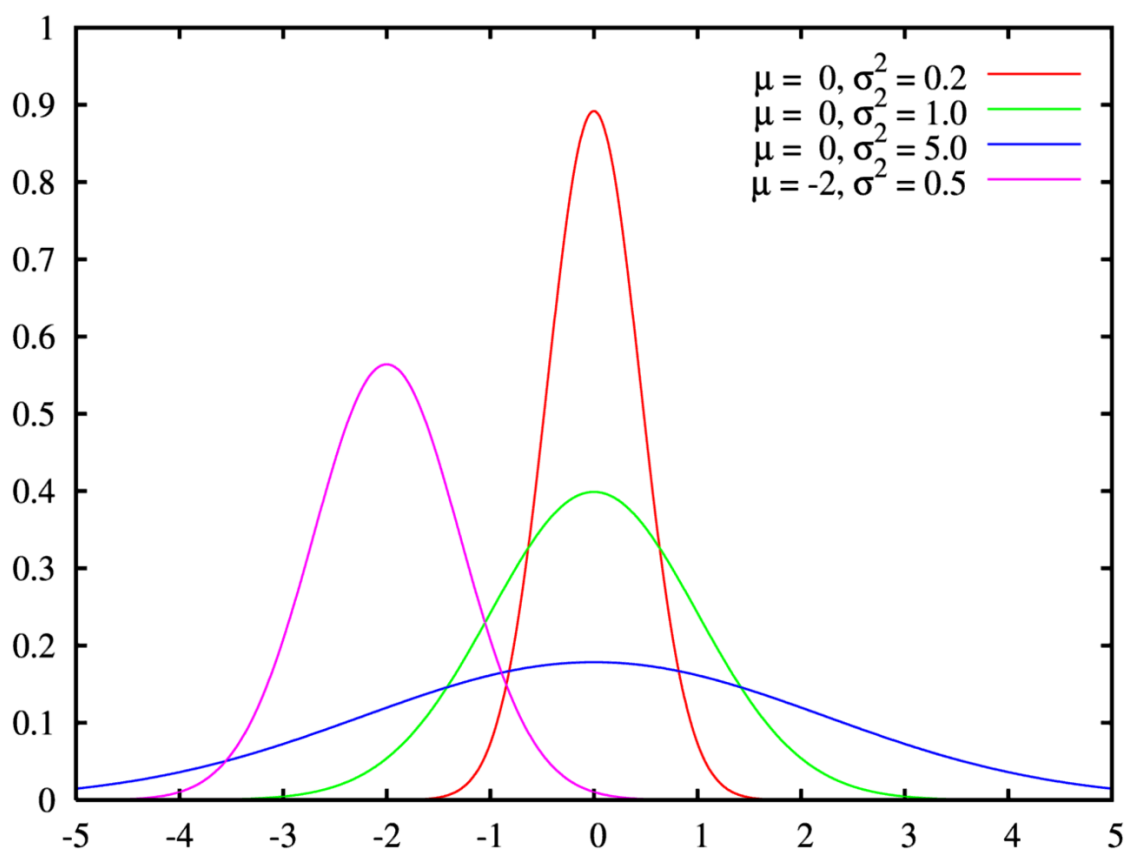


Рис 2. – стандартное распределение

Перед применением статистических моделей нужно удостовериться, что данные имеют нормальное распределение.

Создание библиотеки алгоритмов статистического анализа данных для клинических исследований в параллельных группах, позволит существенно сократить время проведения статистического анализа и поможет проводить более качественные исследования в короткие сроки.

1.2. Формулировка задачи

Целью работы являлась разработка библиотеки языка R с инструкцией по эксплуатации, предназначенной для проведения статистического анализа данных клинических исследований в параллельных группах.

Программирование осуществлялось на языке R, с использованием следующих библиотек:

- base;
- methods;
- utils;
- grDevices;
- plyr;
- xlsx;
- devtools;
- roxygen2.

Для обеспечения возможности возврата к определённым старым версиям разработки, использовалась система контроля версий git и репозиторий на сервере GitHub.

1.3. Функциональные требования

В рамках дипломной работы были поставлены следующие задачи:

- Поиск пропущенных значений (незаполненных полей);
- Поиск опечаток;
- Поиск «выдающихся значений» или «выбросов»;
- Проверка на упорядочение дат;
- Исследование нормальности распределения различными статистическими методами.

1.4. Нефункциональные требования

- Ввод данных должен осуществляться в виде Excel-таблиц или внутренней структуры языка R – таблиц данных (data.frame);
- Реализация методами ООП, используя объектную модель S4 в языке R;
- Использование системы контроля версий Git в связке с сервером GitHub;
- Тестирование созданной библиотеки.

1.5. Характеристики выбранных программных средств

Для решения задачи был выбран язык R и среда разработки RStudio.

R – язык программирования высокого уровня, предназначенный для статистической обработки данных с упором на визуализацию и воспроизводимость. Он так же является свободной кроссплатформенной программной средой вычислений с открытым исходным кодом в рамках проекта GNU. R – интерпретируемый язык с интерфейсом командной строки. Также он сочетает в себе процедурное, функциональное, объектно-ориентированное и программирование.

Сегодня R является безусловным лидером среди свободно распространяемых систем статистического анализа. R обладает хорошей расширяемостью с помощью пакетов. Каждый такой пакет представляет собой библиотеку, содержащую набор специфических функций. В среде R реализованы многие статистические методы: линейные и нелинейные модели, проверка статистических гипотез, анализ временных рядов, классификация, кластеризация, графическая визуализация. Так же, одной из особенностей языка является поддержка графических возможностей, которая позволяет визуализировать данные в виде различных графиков и диаграмм.

2. РЕАЛИЗАЦИЯ

2.1. Алгоритмы решения задач

Различные статистические методы по-разному относятся к наличию выбросов во входных данных. Наличие выбросов может сделать использование определенных статистических моделей *невозможным*, и в то же время, никак не сказаться на результатах других.

Именно поэтому перед проведением статистического анализа необходимо выполнять проверку начальных данных на валидность.

Валидизация данных – это процесс обнаружения и исправления ошибок.

Для каждого рода ошибок был разработан свой алгоритм обнаружения, о которых будет рассказано ниже.

2.1.1. Опечатки

Опечатки – это некорректно введенные пользователем данные. Примером опечаток может служить:

- Данные, не входящие в допустимый набор значений.
- Наличие букв в числовых данных;
- Неправильные разделители между числами в записи дат;
- Лишние пробелы в записи десятичных дробей.

На рис 3 изображена блок-схема алгоритма поиска опечаток для данных, имеющих набор определенных допустимых значений. Такие значения могут содержать, к примеру, записи и тяжести состояния пациента (только значения «удовлетворительное», «средней тяжести», «тяжёлое», «крайне тяжёлое»).

Остальные виды опечаток определяются с помощью шаблонов, заданных регулярными выражениями.

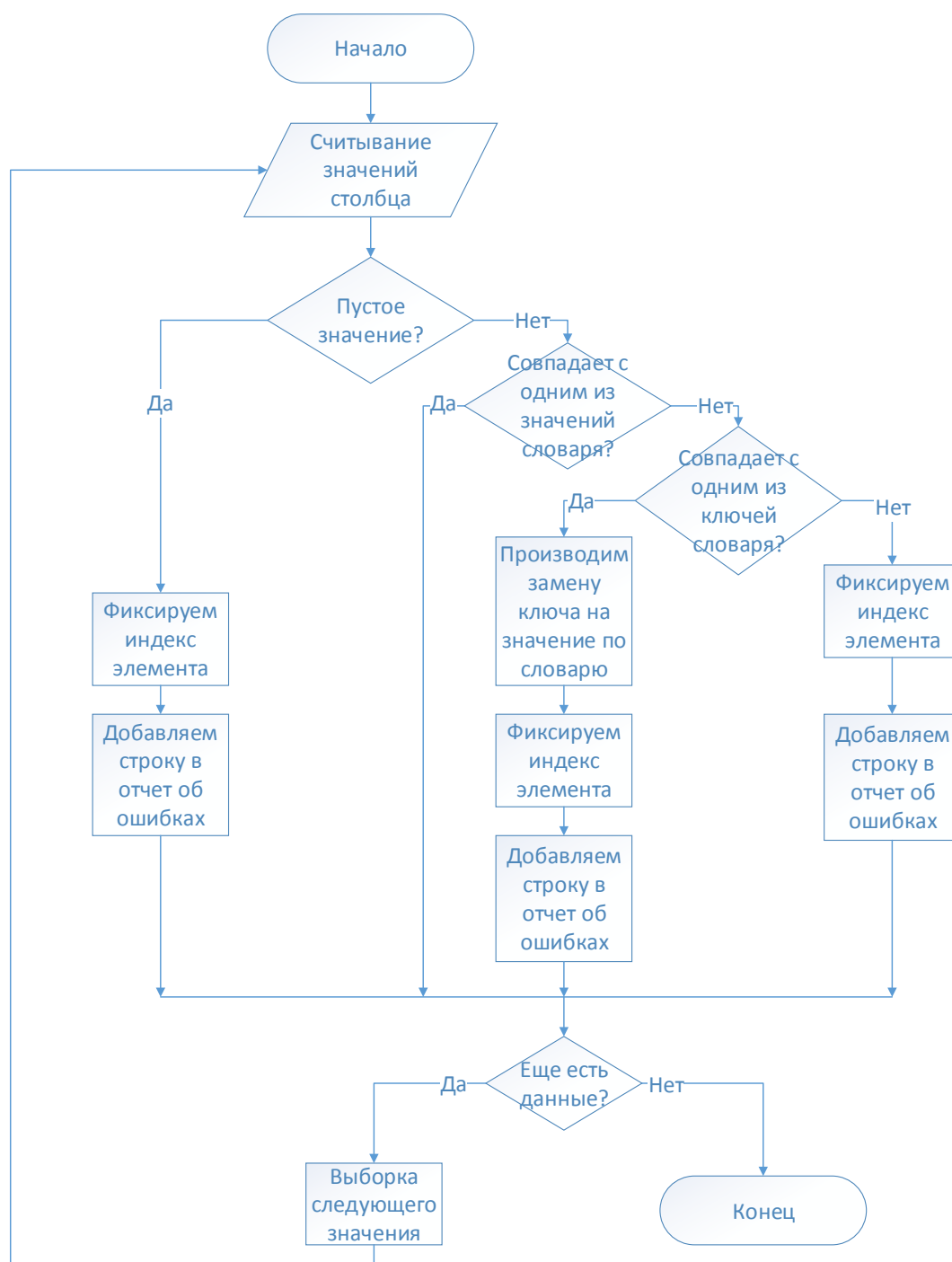


Рис 3. – блок-схема алгоритма поиска опечаток для данных, имеющих набор определенных допустимых значений

2.1.2. Выбросы

Под «выбросом» понимается значение, которое слишком велико или слишком мало по сравнению с большинством других имеющихся значений. К примеру, средний вес большинства участников исследования – 75 кг. Но также в исследовании приняли участие два человека весом 55 кг и 110 кг, именно эти значения будут выбросами.

Для обнаружения выбросов была использована функция `boxplot.stats()`, производная графической функции высокого уровня `boxplot()`, которая служит для построения диаграмм размахов. А `boxplot.stats()`, в свою очередь, используется для сбора статистики, необходимой для создания диаграмм размахов. Подробный механизм реализации описан в пункте 2.2.5.

2.1.3. Генерация отчетов

Библиотека позволяет создавать пользовательские отчеты об ошибках нескольких типов:

- Текстовый файл, в который построчно записываются сообщения о найденных некорректных данных. Текст сообщения содержит тип ошибки, позицию некорректного значения в исходной таблице данных, само значение.
- Excel-таблица с несколькими стилями, которые применяются для наглядного отображения найденных проблем.
- Сводная таблица данных в виде Excel-файла, которая для каждого столбца исходных данных содержит его различные значения и частоту их встречаемости. Она дает верное представление о содержимом входных данных перед началом их обработки, а также помогает в определении допустимого набора значений для исправления опечаток.

2.2. Объектно-ориентированная модель

Поставленные задачи были реализованы с использованием объектно-ориентированной модели (ООП) модели S4 языка R. Созданная логическая структура модели имеет три составляющих:

- Файлы
- Колонки таблицы
- Типы ошибок

Для каждого рода ошибок был реализован специальный алгоритм поиска, которые описаны ниже.

2.2.1. Файлы

В процессе выявления различных ошибок требуется обеспечить ввод и вывод различных данных. Для этого была разработана составляющая «файлы». Ее логическая структура представлена на рис 4.

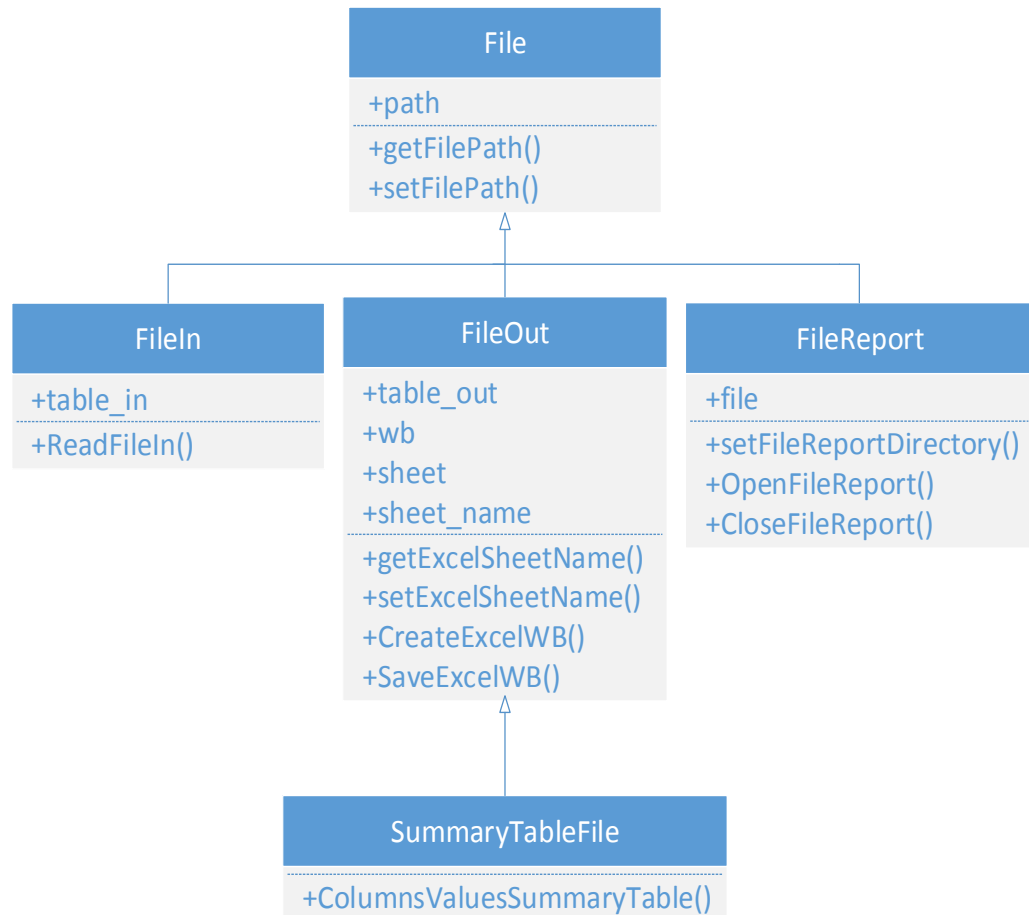


Рис 4. – логическая структура файлов

Класс-родитель File имеет одно поле path и два метода для взаимодействия с ним. Метод setFilePath() устанавливает полный путь к файлу, а метод getFilePath() служит для получения текущего. У класса File существует три дочерних класса: FileIn, FileOut и FileReport. Каждый из них наследует родительские методы и поле path, значение которого для каждого класса будет разным.

Класс FileIn имеет метод ReadFileIn(), реализующий считывание .csv-файла с начальными данными и поле table_in, куда помещается результат считывания файла в виде таблицы данных.

Результат исправления опечаток в исходных данных присваивается полю `table_out` класса `FileOut`.

Также, в классе `FileOut` реализованы методы для работы с Excel и созданы необходимые для этих методов поля. Поля `wb` и `sheet` являются объектами класса `jobjRef` из библиотеки `rJava`, которая, в свою очередь, используется библиотекой `xlsx` для связи Java и R. Поле `sheet_name` содержит строку с названием листа для новой рабочей книги Excel. Назначить новое название можно используя `setExcelSheetName()`, а чтобы узнать текущее — метод `getExcelSheetName()`.

Метод `CreateExcelWB()` нужен для создания новой рабочей книги с именованным листом, и добавления на него итоговой таблицы данных, которая будет раскрашена. Также в этом методе создается новая пустая строка в шапке таблицы для того, чтобы разместить там легенду обозначений различных типов ошибок.

`SaveExcelWB()` сохраняет рабочую книгу, используя в качестве полного пути содержимое поля `path` объекта `FileOut`. Перед сохранением файла для всех столбцов таблицы устанавливается автоподбор ширины столбца и закрепляется первая строка легенды таблицы. Методы `CreateExcelWB()` и `SaveExcelWB()` являются, по сути, «оберткой» для работы с библиотекой `xlsx`.

Для работы с текстовым файлом и записью в него сообщений для пользователя был создан класс `FileReport`. У него есть поле `file`, которое используется для создания файла-отчета. Полный путь к файлу генерируется методом `setFileReportDirectory()`, путем добавления к указанной при вызове метода директории строки «`Report_`» и текущий даты и времени. Метод `OpenFileReport()` создает по указанному полному пути файл и открывает соединение для записи в текстовом режиме, а метод `CloseFileReport()` закрывает это соединение.

Для создания сводной таблицы, которая должна содержать названия столбцов исходной таблицы, их значения и частоту встречаемости каждого из значений, был создан класс `SummaryTableFile`, являющийся потомком `FileOut`, метод `ColumnsValuesSummaryTable()`. Механизм работы данного метода описан в пункте 2.2.7.

2.2.2. Типы значений колонок таблицы

Структура данных таблицы, с которой работает библиотека, может состоять из четырех типов значений: дата, непрерывные, дискретные и категориальные (номинальные).

В виде дат в результатах исследования может указываться время измерения различных показателей пациента, время его поступления в клинику и время выписки. Важно следить за тем, чтобы даты повторных измерений сохраняли последовательность.

Непрерывные переменные могут принимать любые численные значения, которые естественным образом упорядочены на числовой оси (например, рост, вес).

Дискретные переменные могут принимать счётное множество упорядоченных значений, которые могут просто обозначать целочисленные данные или ранжировать данные по степени проявления на упорядоченной ранговой шкале (клиническая стадия опухоли, тяжесть состояния пациента).

Категориальные переменные являются неупорядоченными и используются для качественной классификации (пол, цвет глаз, место жительства); в частности, они могут быть бинарными (дихотомическими) и иметь категорические значения: 1/0, да/нет, имеется/отсутствует. Поэтому каждый из четырех типов значений описывает свой класс, и для каждого из них реализован свой метод поиска ошибок. Логическая структура колонок таблицы представлена на рис 5.

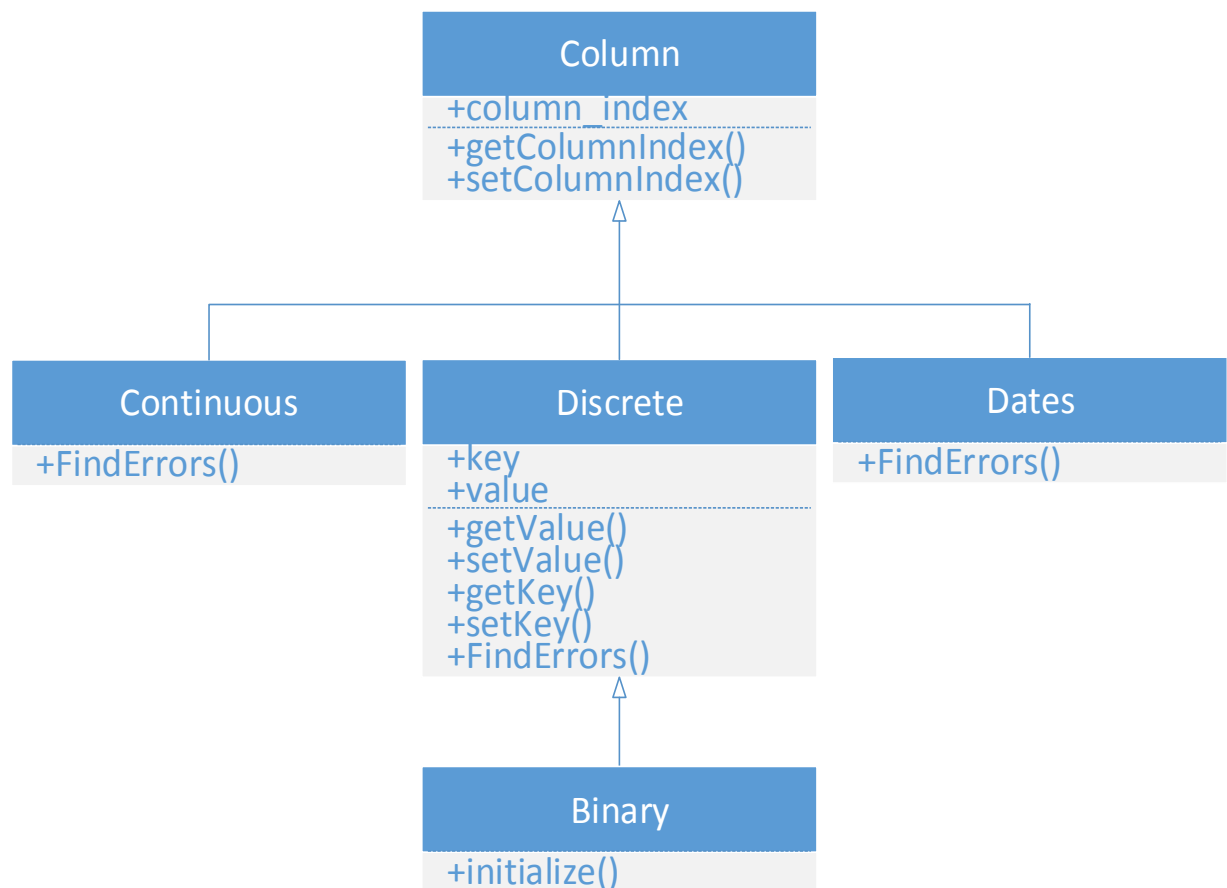


Рис 5. – логическая структура колонок таблицы

Родительский класс **Column** имеет поле `column_index`, в котором хранится номер колонки в исследуемой таблице, и два метода `getColumnIndex()`, `setColumnIndex()` для задания нового значения этого поля и получения текущего. Данные методы доступны во всех четырех дочерних классах.

Объекты класса **Continuous** описывают колонки таблицы, содержащие непрерывные переменные. Метод `FindErrors()` для объектов класса **Continuous**, состоит из поиска опечаток и поиска выбросов.

Объекты класса **Discrete** описывают дискретные переменные. Для поиска и исправления опечаток была разработана структура словаря.

Словарь состоит из ключей и значений. Одному значению может соответствовать множество ключей.

Поля объекта **Discrete**, `key` (ключ) и `value` (значение), являются переменными R-типа `list`, т.е. они имеют вложенную структуру, представляющую из себя массив массивов, и могут содержать в себе сочетания любых типов данных. Это позволяет эффективно, т.е. в одном объекте, хранить разнородную информацию. Более подробное описание работы со структурой словаря описано ниже, в пункте 2.2.4.1.

Задать новые значения для полей класса `Discrete`, `key` и `value`, можно при помощи созданных методов `setValue()` и `setKey()`, а получить их текущие значения при помощи `getValue()` и `getKey()` соответственно. Эти методы также доступны в дочернем классе.

Дочерний класс `Binary()` имеет более узкую специализацию и описывает только бинарные категориальные переменные (например, пол). Во время инициализации класса `Binary()` значения поля `value` устанавливаются по умолчанию: 0 и 1. При необходимости, они могут быть изменены при помощи родительского метода `setValue()`.

Объекты класса `Dates` необходимы для описания элементов таблицы, содержащих даты.

Метод `FindErrors()` для классов `Discrete` и `Dates` содержит метод поиска опечаток, но для каждого из этих двух классов существует своя реализация данного метода, более подробно это описано в пункте 2.2.4.

2.2.3. Типы ошибок

Процесс валидации «сырых» данных должен выявлять следующие типы ошибок:

- Опечатки.
- Неупорядоченные даты.
- Выбросы.
- Пропущенные значения (незаполненные поля).

Для каждого типа ошибок был создан свой класс, который наследуется от класса `Error`. Это сделано для их объединения по общим признакам, таким как:

- Индексы ошибки в таблице;
- Стилль типа ошибки для раскраски итоговой таблицы;
- Название стилия в легенде таблицы;
- Позиция легенды ошибки внутри итоговой таблицы.

На рис 6. изображена логическая структура типов ошибок.

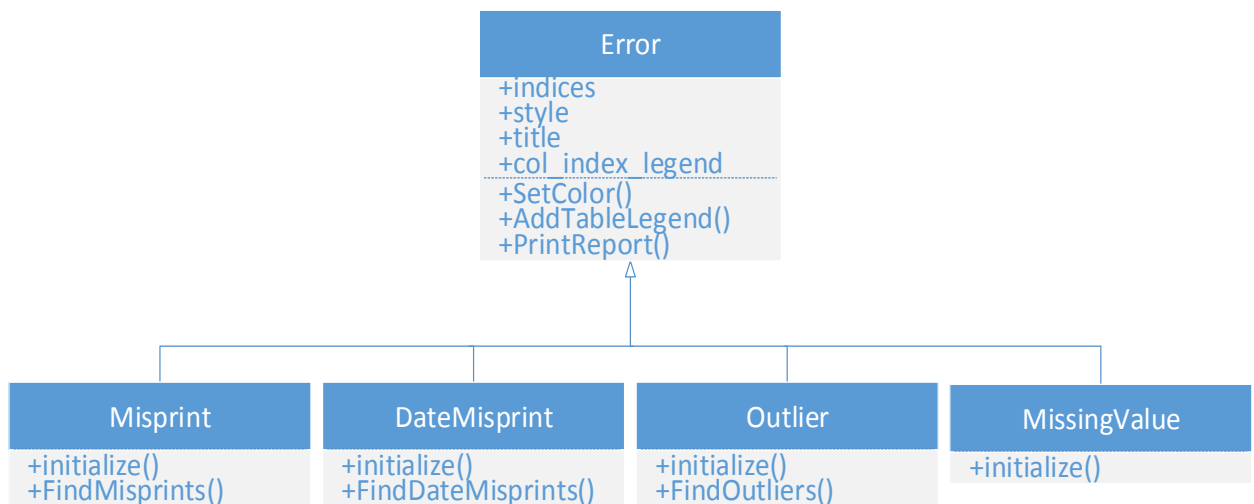


Рис 6. – логическая структура типов ошибок

Родительский класс `Error` имеет следующие поля: `indices`, которое хранит индексы ячеек таблицы, содержащих ошибки, поле `style` используемое для хранения названия стиля, который должен быть применен для раскраски неправильно заполненных ячеек, поле `title` содержащее строку с названием типа ошибки (эта строка является легендой результирующей таблицы), и `col_index_legend`, в котором хранится позиция ячейки таблицы с легендой.

Значения для полей `style`, `title` и `col_index_legend` устанавливаются во время инициализации каждого из дочерних классов `Error`. Для этого используется созданный метод `initialize()`, а не стандартный конструктор. Каждый вызов метода `FindErrors()` изменяет поле `indices`, добавляя новые значения индексов ячеек таблицы, содержащих ошибки. После того, как был выполнен анализ всех столбцов таблицы, итоговая таблица может быть раскрашена. Для этого был создан метод `SetColor()`, который взаимодействует с объектом новой рабочей книги `Excel`, созданной при вызове метода `CreateExcelWB()`. В методе `SetColor()` осуществляется доступ ко всем ячейкам рабочей книги и устанавливаются новые стили для тех ячеек, индексы которых содержит поле `indices`. Выбор стиля для типа ошибки осуществляется исходя из значения, содержащегося в поле `style`. Внутри метода `SetColor()` также осуществляется вызов метода `AddTableLegend()`, при помощи которого происходит добавление легенды в шапку таблицы.

Метод `PrintReport()` отвечает за то, чтобы производить запись сообщений о найденных ошибках в пользовательский текстовый файл. Он принимает в качестве одного из аргументов объект класса `FileReport` и

использует его поле `file` с открытым соединением. Сообщения отчета об ошибках представлены на рис 7.

```
MissingValue in row 3 column возраст (3)
UnsolvedMisprint in row 4 column возраст (3)
Outlier in row column возраст (3)
It is impossible to determine outliers, there are unsolved misprints in the column column возраст (3)
MissingValue in row 7 column вес (4)
Outlier in row 5 column вес (4)
Outlier in row 6 column вес (4)
Outlier in row 107 column вес (4)
Misprint in row 38 column диабет (5)
Misprint in row 45 column диабет (5)
Outlier in row 22 column euroscore<5 5-10 >10 (12)
Outlier in row 27 column euroscore<5 5-10 >10 (12)
```

Рис 7. – сообщения отчета об ошибках

Тип ошибки определяется автоматически, путем определения того, к какому из потомков класса `Error` относится переданный методу объект.

2.2.4. Описание реализации поиска опечаток

Поиск опечаток, а также, по возможности, их исправление, осуществляется при помощи метода `FindMisprints()`. У него существует три реализации, для каждого из четырех типов значений, описанных выше, которые может содержать исследуемая таблица.

2.2.4.1. Метод поиска опечаток для дискретных значений

Доступ к элементам определенной колонки в исходной таблице данных осуществляется по ее индексу, который был передан при вызове метода. Полученные значения элементов присваиваются новой переменной, представляющей собой одномерный массив. В цикле, последовательно проверяется каждый элемент этого массива на предмет различных ошибок.

В первую очередь осуществляется поиск пропущенных значений. Если проверка выявила существование пропущенного значения в колонке, то его индекс передается полю объекта класса `missingValue` для дальнейшей раскраски, и вызывается метод `PrintReport()`, который производит запись сообщения о найденной ошибке и ее координатах в пользовательский текстовый файл-отчет.

Если результат проверки на заполнение оказался успешным и значение ячейки отлично от пустой строки, то начинается проверка на совпадение этого элемента с одним из значений словаря. Для корректности сравнения, значение словаря и значение элемента колонки приводятся к верхнему регистру, во избежание ошибок.

Если значение колонки не совпало ни с одним значение словаря, проверяется совпадение элемента с одним из ключей. Оба значения так же приводятся к верхнему регистру.

Если совпадение найдено, то по нужному индексу в итоговой таблице производится замена ключа на соответствующее ему значение в словаре, а индексы элемента передаются полю объекта класса `Misprint` для раскраски итоговой таблицы. Далее происходит вызов метода `PrintReport()`. Ячейка, содержащая данный элемент в итоговой таблице будет выделена как исправленная опечатка.

В случае, когда элемент колонки не совпал ни с одним из ключей, индексы этого элемента в таблице (номер строки и номер колонки) передаются полю объекта класса `UnsolvedMisprint`, и производится запись сообщения об ошибке в файл при помощи метода `PrintReport()`.

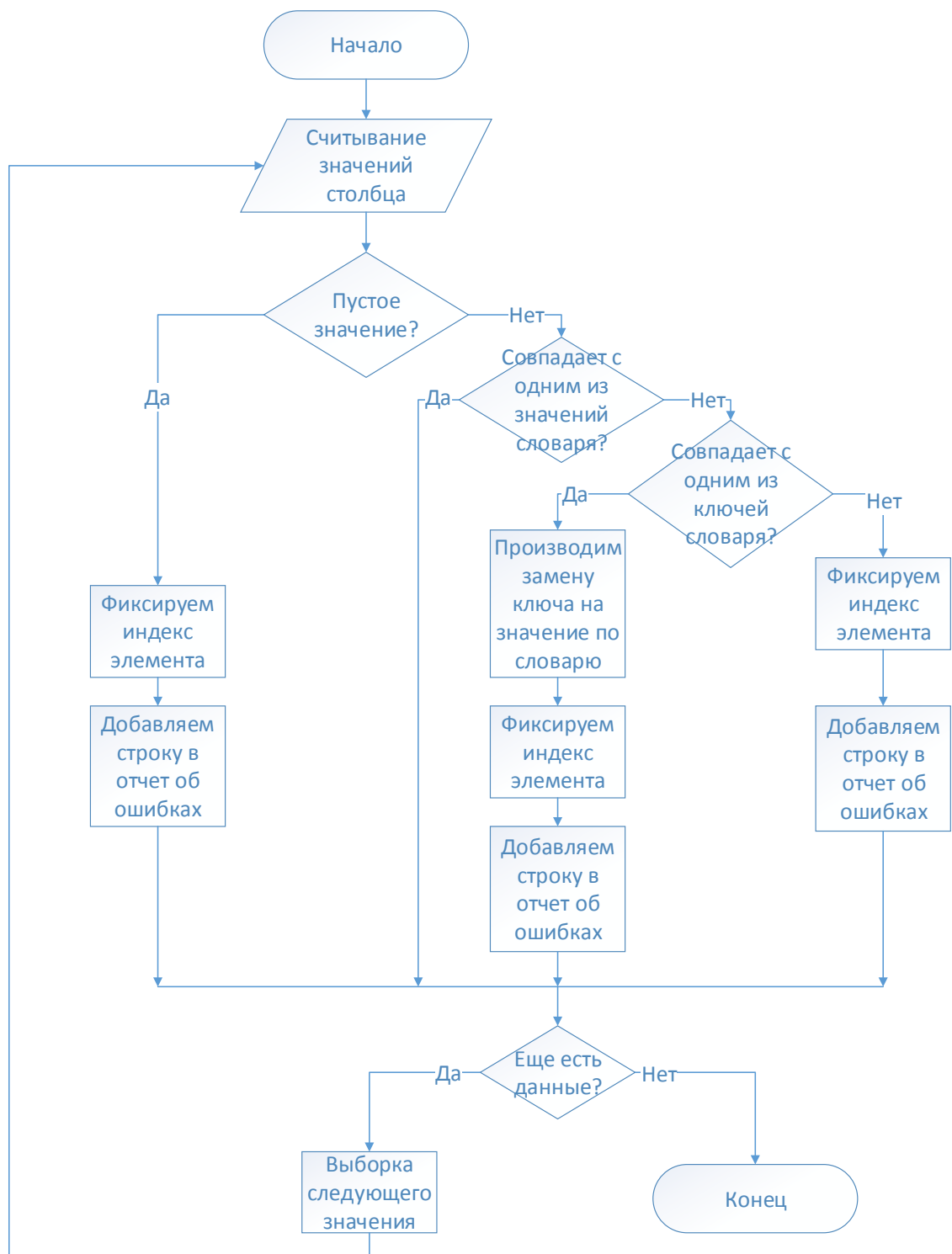


Рис 8. – блок-схема алгоритма поиска опечаток для дискретных значений

2.2.4.2. Метода поиска опечаток для непрерывных значений

Так же, как и в методе, реализованном для категориальных переменных, в первую очередь внутри метода производится доступ к определенной колонке и проверка на заполнение всех ее элементов.

Далее значение элемента колонки сравнивается с шаблоном, заданным регулярным выражением, показанным на рис 9.,:

"^((\\d)+)[,.]|([[:space:]])?(\\d)+)?\$"

Рис 9. – шаблон регулярного выражения для непрерывных значений

где

^ – символ начала строки,

(\\d)+ – одно число и более,

[,.] – один из двух указанных символов (точка или запятая),

| – логическое ИЛИ,

[[:space:]]? – пустые символы ноль или один раз,

\$ – символ конца строки

Данный шаблон был разработан для работы с целыми и дробными числами.

Если значение элемента не соответствует шаблону, это может означать, что оно содержит буквы, или любые другие символы. Следовательно, индексы элемента передаются полю объекта класса `UnsolvedMisprint`, который, как было сказано ранее, хранит в себе индексы элементов таблицы, которые будут раскрашены как неисправленные опечатки. Далее производится запись сообщения о найденной ошибке в файл при помощи `PrintReport()`.

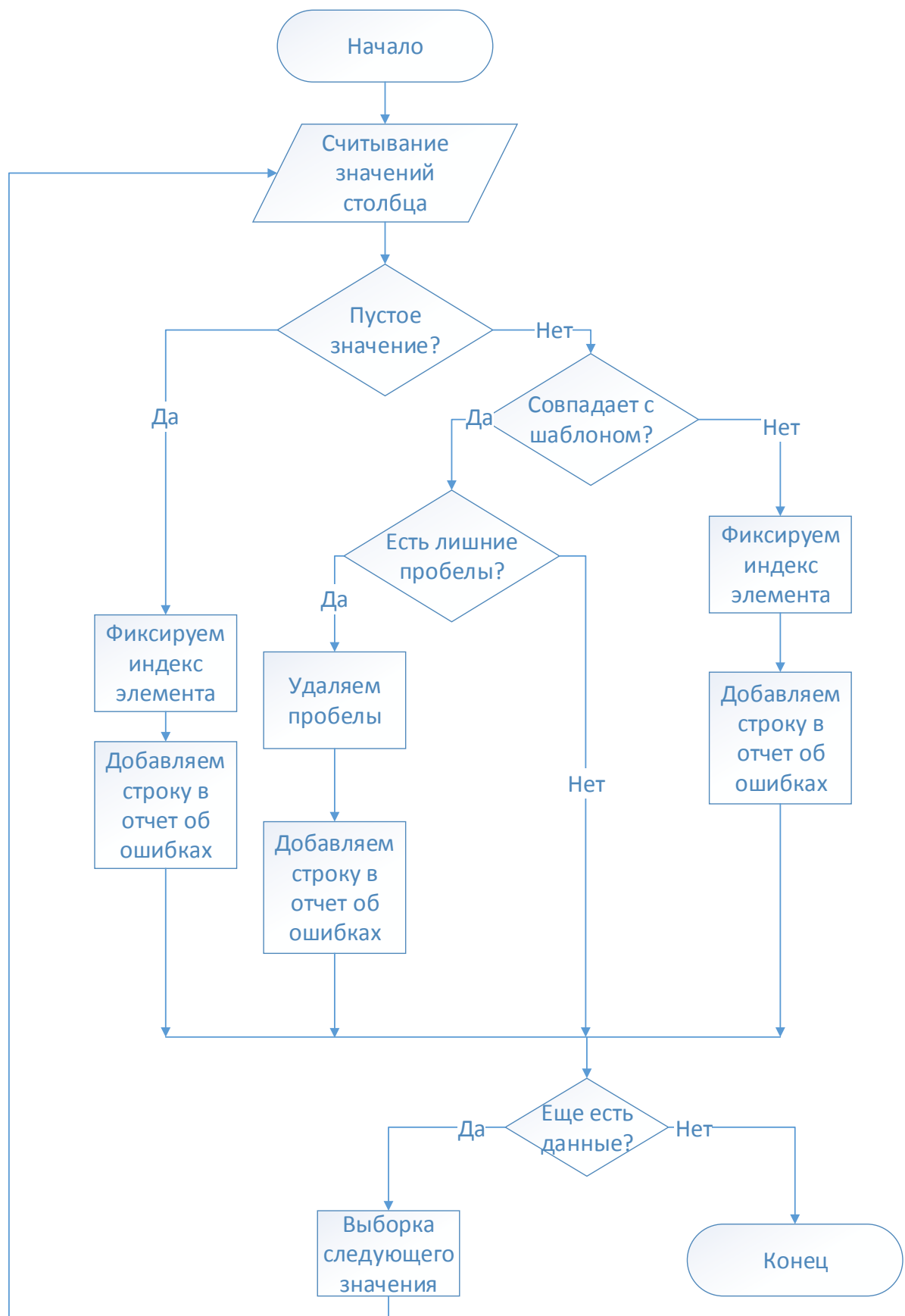


Рис 10. блок-схема алгоритма поиска опечаток для непрерывных значений

2.2.4.3. Метод поиска опечаток для дат

Данная реализация схожа с предыдущей, но отличается шаблоном применяемого регулярного выражения, который используется для сравнения со значением элемента. Регулярное выражение изображено на **Ошибка! Источник ссылки не найден.**

`"^((\\d){2})([.,]|[-/])(\\d{2})([.,]|[-/])(\\d{2}|(\\d){4})$"`

Рис 11. – шаблон регулярного выражения для непрерывных значений

где

^ – символ начала строки,

(\\d){2}) – два числа,

([.,]|[-/]) – один из четырех указанных символов («,» или «.» , или «/», или «-» или «-»),

((\\d){2}|(\\d){4}) – два или четыре числа,

\$ – символ конца строки

Если значение элемента не соответствует шаблону, то производится запись сообщения о найденной опечатке в пользовательский файл, а индексы этого элемента передаются полю объекта класса `UnsolvedMisprint`.

Все значения элементов колонки, которые подошли под указанный выше шаблон, являются датами, но существует возможность того, что они записаны неверно. Поэтому производится дополнительная проверка разделителей между числами в записи даты и замена их на точку, если существует ошибка. Далее, вызванный метод `PrintReport()` сообщает о найденной и исправленной опечатке, если таковая выявлена, и добавляет индексы таких элементов полю объекта класса `Misprint`. Наглядное представление описанного алгоритма приведено на рис 12.

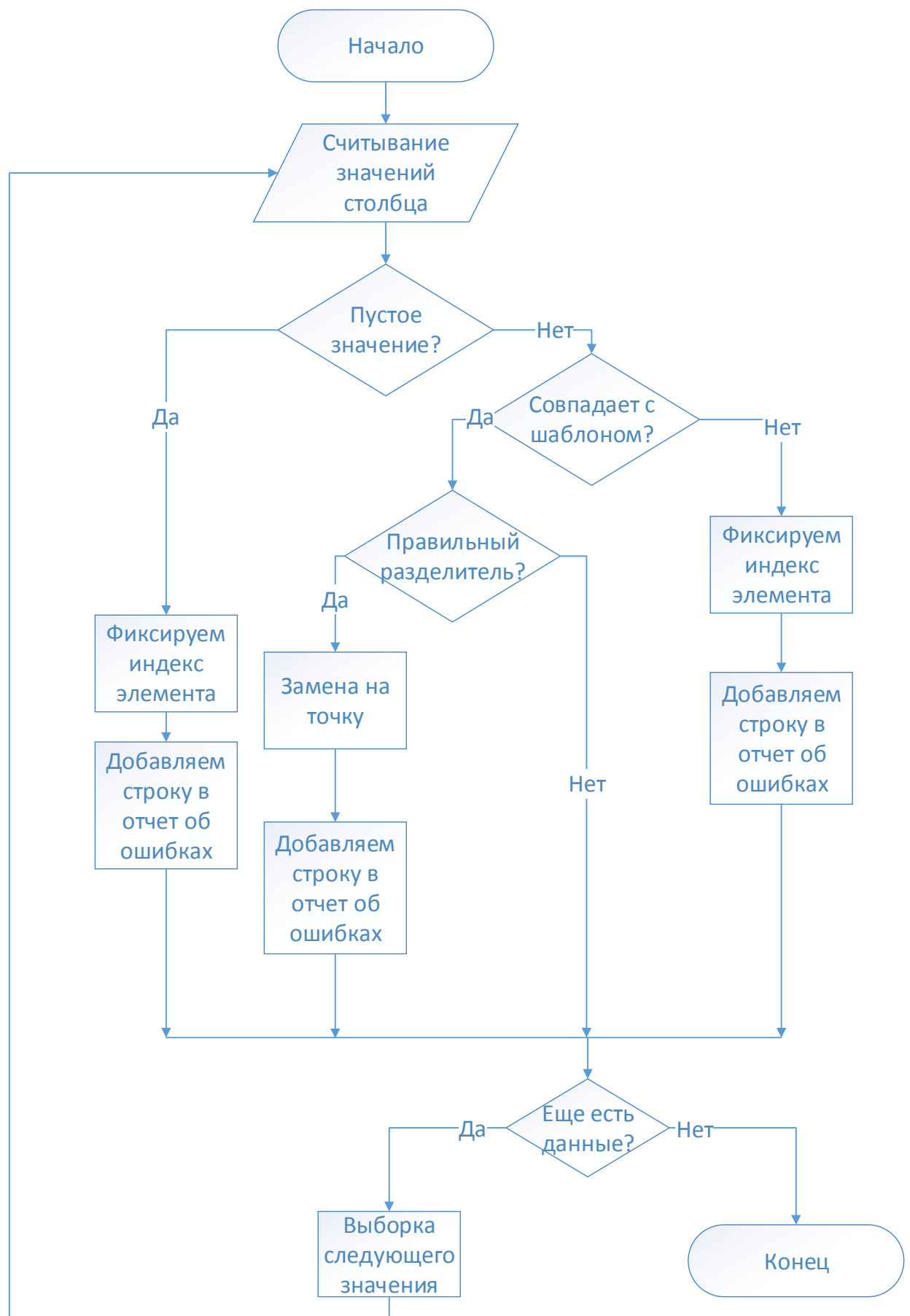


Рис 12. – блок-схема алгоритма поиска опечаток для дат

2.2.5. Реализация поиска выбросов

Поиск выбросов осуществляется только среди значений элементов колонок, которые описывает класс Continuous, т.е. непрерывных переменных. Анализ возможен лишь в том случае, когда среди них нет неисправленных опечаток. В противном случае подобная оценка будет ошибочной и, следовательно, бесполезной.

Выбросы в столбце определяются при помощи функции `boxplot.stats()`, производной графической функции высокого уровня `boxplot()`, которая служит для построения диаграмм размахов. `boxplot.stats()`, в свою очередь, используется для сбора статистики, необходимой для создания диаграмм размахов.

Диаграммы размахов, или "ящики с усами" (англ. box-whisker plots), получили свое название за характерный вид: точку или линию, соответствующую медиане или средней арифметической, окружает прямоугольник ("ящик"), длина которого соответствует одному из показателей разброса или точности оценки генерального параметра. Дополнительно от этого прямоугольника отходят "усы", также соответствующие по длине одному из показателей разброса или точности. Строение получаемых при помощи этой функции "ящиков с усами" представлено ниже на **Ошибка! Источник ссылки не найден..**

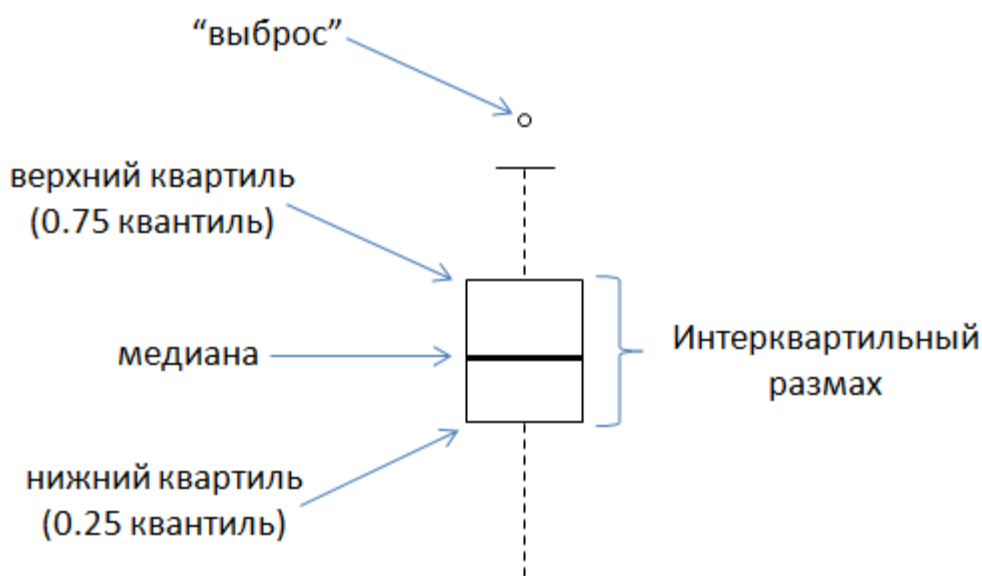


Рис 13. – диаграмма размахов

В R при построении диаграмм размахов используются устойчивые (робастные) оценки центральной тенденции (медиана) и разброса

(интерквартильный размах, далее ИКР). Верхний "ус" простирается от верхней границы "ящика" до наибольшего выборочного значения, находящегося в пределах заданного расстояния, чаще всего используется коэффициент $1.5 \times \text{ИКР}$ от этой границы. Аналогично, нижний "ус" простирается от нижней границы "ящика" до наименьшего выборочного значения, находящегося в пределах расстояния $1.5 \times \text{ИКР}$ от этой границы. Наблюдения, находящиеся за пределами "усов", потенциально могут быть выбросами.

В первую очередь выполняется проверка значения счетчика неисправленных опечаток. Если он равен нулю, то это означает, что можно приступить к анализу значений элементов колонки. Иначе, происходит вызов метода печати в файл `PrintReport()` и запись предупреждения о невозможности проведения анализа, т.к. среди значений определенной колонки присутствуют опечатки, а так же сообщается номер колонки.

Далее осуществляется доступ к элементам колонки по переданному индексу и поиск среди них пропущенных значений.

Функция `boxplot.stats()` работает только с численными значениями. Опечатки в колонке таблицы, содержащей непрерывные переменные (например, пробел после запятой в десятичной дроби или разные разделители между числами), препятствуют правильному распознаванию их значений системой, как числовых, вместо этого они представляются как значения строкового типа. Именно поэтому, для того, чтобы включить данные значения в анализ выбросов, был использован шаблон, изображенный на рис 14., заданный при помощи регулярного выражения:

`"^(\d)+([.,](\d)+)?$"`

Рис 14. – шаблон регулярного выражения для непрерывных значений

где

`^` – символ начала строки,

`(\d)+` – одно число и более,

`[.,]` – один из двух указанных символов (точка или запятая),

`([.,](\d)+)?` – разделитель , одно число и более ноль или один раз,

`$` – символ конца строки

Все значения элементов колонки сравниваются с данным шаблоном и при совпадении сохраняются в новую переменную, используемую для анализа. Далее, среди значений этой переменной производится замена

разделителей в десятичных дробях на точку. После замены используется приведение типов каждого элемента переменной к числовому, и выполняется поиск выбросов с помощью `boxplot.stats()`, которая анализирует сразу весь массив. Результат работы функции хранится в виде вложенной структуры массива массивов. Значения (не их индексы) из переданного массива, потенциально являющимися выбросами, содержатся в элементе `out` данной структуры. Если он содержит какие-либо значения, то они сопоставляются с исходными значениями колонки, для того, чтобы определить позицию уже найденных выбросов в таблице. Для сравнения разделители в исходных элементах, все еще являющихся строковыми, тоже преобразуются. Во время вызова метода `PrintReport()` происходит печать сообщения о найденных выбросах и их позиции в таблице (номера строки и столбца).

К подобным нестандартным наблюдениям всегда следует относиться внимательно. Они вполне могут оказаться «нормальными» для исследуемой совокупности, и поэтому не должны удаляться из анализа без дополнительного изучения причин их появления.

2.2.6. Реализация поиска неупорядоченных дат

Метод `FindDateMisprints()` реализует поиск непоследовательных дат внутри таблицы. Двумя из его аргументов являются два объекта класса `Dates`, которые описывают колонки с разными датами, которые предстоит проверить. Например, это могут быть даты первичных и повторных замеров или даты поступления и выписки пациента. После сортировки колонок по индексам производится приведение каждого из их элементов к типу данных `R date`, с форматом представления `месяц.день.год`, и присваивание полученных значений двум новым переменным. Далее эти переменные поэлементно сравниваются между собой. Если какое-либо из значений первой колонки с датами оказалось больше значения второй колонки, находящегося в этой же строке, то его индексы добавляются в поле `indices` объекта `DateMisprint`. В результирующей таблице будут выделены ячейки обеих колонок в этой строке.

2.2.7. Создание сводной таблицы значений

Данный функционал необходим для того, чтобы перед началом проверки «сырых» данных таблицы на валидность, пользователь получил

представление о значениях, которые она содержит. Исходя из этого, он сможет верно сопоставить столбцы таблицы (вернее, их индексы) и классы, которые должны их описывать, а так же задать ключи и значения словарей, которые будут использоваться для поиска и исправления опечаток.

Реализацией является метод `ColumnsValuesSummaryTable()`, который работает с объектом класса `SummaryTableFile`. В нем названия колонок исходной таблицы, полученные при помощи функции `colnames()`, присваиваются новой переменной. Далее, в цикле к элементам каждого столбца исходной таблицы с «сырыми» данными применяется функция `table()`. Ее результат содержит все различные значения столбца и частоту встречаемости каждого из этих значений. Он приводится к таблице данных. Для того чтобы получить нужный формат используется функция транспонирования таблицы данных как матрицы, а после еще раз преобразование в таблицу данных. Полученная таблица добавляется в созданную ранее пустую результирующую переменную в виде новых строк. Т.к. количество различных значений в разных столбцах таблицы неодинаковое, то это создает проблемы для создания результирующей таблицы обычным путем, т.к. ее строки должны быть одной длины. Поэтому для решения этой проблемы использовалась функция `rbind.fill()` из библиотеки `plyr`, которая заменяет значения недостающих столбцов на NA. Если количество различных значений в одной колонке больше десяти, то для отображения будут использованы только первые десять значений колонки таблицы.

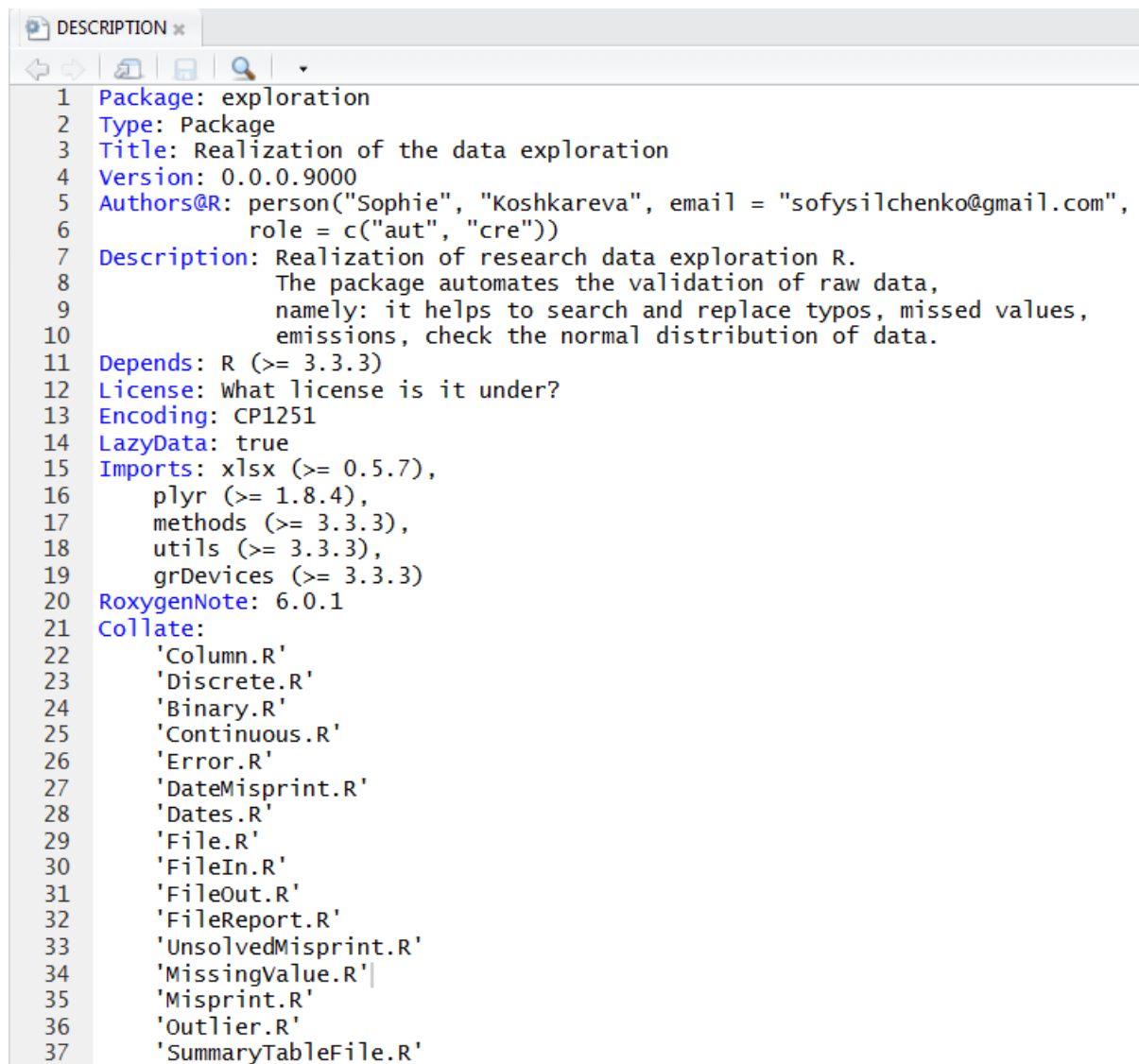
Названия столбцов исходной таблицы вставляются перед каждой новой строкой, описывающей различные значения и их частоту. Далее происходит запись полученной результирующей таблицы в новую рабочую книгу Excel, путем вызова родительского метода `CreateExcelWB()`. Устанавливаются необходимые стили оформления и при помощи метода `SaveExcelWB()` производится сохранение нового .xlsx-файла.

3. РАЗРАБОТКА БИБЛИОТЕКИ

После написания классов и методов, все содержащие их файлы были объединены в новую библиотеку. Для сборки использовались вспомогательные библиотеки devtools и roxygen2.

Порядок действий:

- Установка библиотек devtools и roxygen2;
- Создание каркаса новой библиотеки;
- Перенос файлов, в которых определены классы и методы в нужную директорию внутри библиотеки;
- Изменение файла DESCRIPTION (рис 15);
- Изменение файла NAMESPACE;
- Добавление документации (рис 16).



```
DESCRIPTION
1 Package: exploration
2 Type: Package
3 Title: Realization of the data exploration
4 Version: 0.0.0.9000
5 Authors@R: person("Sophie", "Koshkareva", email = "sofysilchenko@gmail.com",
6               role = c("aut", "cre"))
7 Description: Realization of research data exploration R.
8               The package automates the validation of raw data,
9               namely: it helps to search and replace typos, missed values,
10              emissions, check the normal distribution of data.
11 Depends: R (>= 3.3.3)
12 License: what license is it under?
13 Encoding: CP1251
14 LazyData: true
15 Imports: xlsx (>= 0.5.7),
16         plyr (>= 1.8.4),
17         methods (>= 3.3.3),
18         utils (>= 3.3.3),
19         grDevices (>= 3.3.3)
20 RoxygenNote: 6.0.1
21 Collate:
22   'Column.R'
23   'Discrete.R'
24   'Binary.R'
25   'Continuous.R'
26   'Error.R'
27   'DateMisprint.R'
28   'Dates.R'
29   'File.R'
30   'FileIn.R'
31   'FileOut.R'
32   'FileReport.R'
33   'UnsolvedMisprint.R'
34   'MissingValue.R'|
35   'Misprint.R'
36   'Outlier.R'
37   'SummaryTableFile.R'
```

Рис 15. – настройки файла DESCRIPTION

Read raw value from .csv-file and create new object class FileIn.

Description

Read raw value from .csv-file and create new object class FileIn.

Value

an object class FileIn theObject.

Slots

`table_in`

a data.frame to loaded data.

Examples

```
myfile <- new("FileIn")
myfile <- setFilePath(myfile, "D:/data.csv")
myfile <- ReadFileIn(myfile)
```

Рис 16. – страница из документации библиотеки

4. ОТЛАДКА И ТЕСТИРОВАНИЕ

На этапе отладки и тестирования были выявлены и решены следующие ошибки:

- Проблема перезаписи файлов отчета и итоговой раскрашенной таблицы. Проблема была решена путем добавления текущего времени и даты в название файла;
- Проблема интерпретации значений в колонках таблицы средой R, из-за опечаток (например, лишний пробел после запятой в десятичной дроби) все числовые значения колонки становятся строковыми и не включаются в анализ наряду с числовыми. Данная проблема была исправлена, при помощи шаблонов регулярных выражений и приведения типов;
- Проблема с созданием сводной таблицы, в которой содержатся значения каждой колонки и частота их встречаемости. Длина столбцов получается разной, а для компоновки таблицы данных или матрицы они должны быть одной длины. Проблема устранена с помощью функции `rbind.fill` из библиотеки `plyr`, которая

РЕЗУЛЬТАТЫ

Разработанные алгоритмы были проверены на реальных входных данных. На рис 17. и рис 18. представлены входные данные и результат обработки.

R24C1	f_x	CABG		
	1	2	3	4
1	Пациент	пол	возраст	вес
2	CABG	M		75
3	CABG	M	бальзаков	56
4	CABG	M	56	300
5	CABG	M	37	200
6	CABG	M	57	
7	CABG	M	44	61
8	CABG	Ж	56	81
9	CABG	M	59	66
10	CABG	Ж	59	84
11	CABG	M	59	86
12	CABG	M	46	84
13	CABG	M	66	88
14	CABG	M	60	59
15	CABG	M	55	74
16	CABG	Ж	63	89
17	CABG	M	58	71
18	CABG	M	53	74

Рис 17. – фрагмент исходного файла с опечатками и пропущенным значением

R32C1	f_x	CABG		
	1	2	3	4
1	Пропущенные значения	Исправления	Опечатки	Выбросы
2	Пациент	пол	возраст	вес
3	CABG	1		75
4	CABG	1	бальзаковский	56
5	CABG	1	56	300
6	CABG	1	37	200
7	CABG	1	57	
8	CABG	1	44	61
9	CABG	0	56	81
10	CABG	1	59	66
11	CABG	0	59	84
12	CABG	1	59	86
13	CABG	1	46	84
14	CABG	1	66	88
15	CABG	1	60	59
16	CABG	1	55	74
17	CABG	0	63	89
18	CABG	1	58	71

Рис 18. – результат обработки

На рис 20 изображен фрагмент входных данных, который содержит нарушение упорядоченности дат, опечатку (в качестве разделителя запятая вместо точки) и пустые ячейки. А на рис 20 представлена итоговая раскрашенная таблица с исправлением.

24	42
Дата вмешательства	
19.02.2009	20.02.2008
08.05.2009	09.05.2009
13,05,2009	17.05.2009
12.05.2009	12.05.2010
18.08.2009	18.08.2010
14.10.2009	14.10.2010
26.03.2010	26.03.2011
30.08.2010	30.08.2010
01.02.2010	

Рис 19. – фрагмент исходного файла с ошибками в датах

24	42
Дата вмешательства	
19.02.2009	20.02.2008
08.05.2009	09.05.2009
13.05.2009	17.05.2009
12.05.2009	12.05.2010
18.08.2009	18.08.2010
14.10.2009	14.10.2010
26.03.2010	26.03.2011
30.08.2010	30.08.2010
01.02.2010	

Рис 20. – исправленная таблица

Фрагмент результирующей сводной таблице показан на рис 21.

R31C14	f_x									
	1	2	3	4	5	6	7	8	9	10
1	Пациент									
2	CABG	PCI								
3	51	72								
4	пол									
5	0	1	Ж	М						
6	12	60	11	40						
7	возраст									
8	22	24	25	32	37	38	40	43	44	45
9	1	1	1	2	2	2	1	1	2	1
10	вес									
11	45	53	55	56	57	58	59	60	60.5	61
12	2	1	2	2	3	1	2	2	1	3
13	диабет									
14	0	1	1 тип	2 тип						
15	113	8	1	1						
16	Класс по NYHA									
17	2	3								
18	28	95								
19	ТИА\ОНМК									
20	0	1								
21	118	5								
22	АГ ст									
23	0	1	2	3						
24	17	17	28	61						

Рис 21. – фрагмент сводной таблицы

На рис 22 приведен график плотности распределения одной из колонок исходных данных.

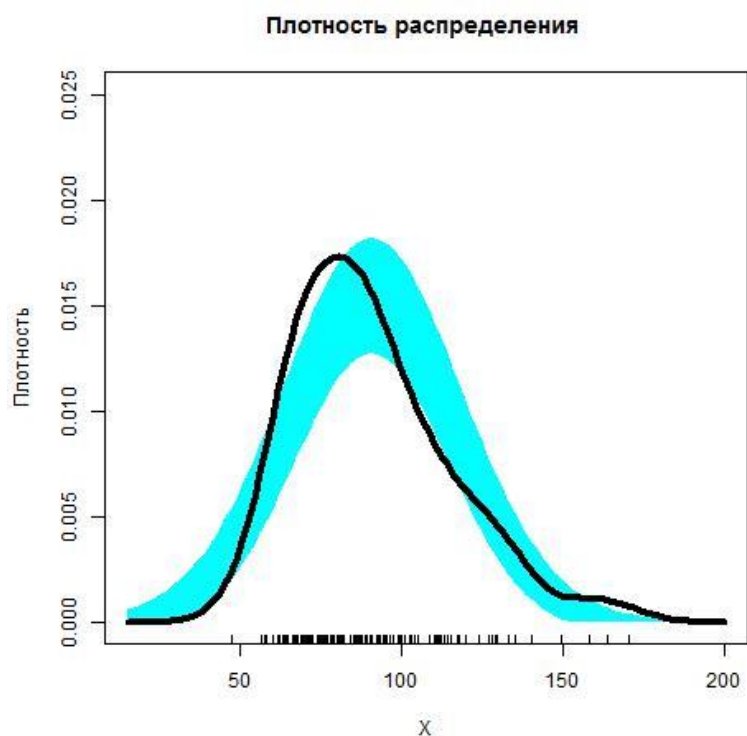


Рис 22. – график плотности распределения

ЗАКЛЮЧЕНИЕ

Разработанная в рамках дипломной работы библиотека, решает поставленные задачи проверки входных данных, а также позволяет биостатистикам анализировать данные исследования в удобной форме за короткий промежуток времени.

Созданная библиотека алгоритмов для статистического анализа данных клинических исследований удовлетворяет всем поставленным требованиям:

- Выявление пропущенных значений (незаполненных полей).
- Поиск опечаток.
- Поиск выделяющихся значений или «выбросов».
- Проверка на упорядочение дат;
- Исследование нормальности распределения различными статистическими методами.

В ходе выполнения дипломной работы была изучена предметная область, разработана и описана архитектура библиотеки, идентифицирующей потенциальные проблемы исследования данных, используя ООП модели S4 на языке R. Был рассмотрен пример применения библиотеки на реальных задачах.

Таким образом, удалось создать библиотеку проверки входных данных, которая в дальнейшем позволит переложить рутинные действия на компьютер и дает возможность биостатистикам анализировать данные исследования удобной форме.

Выражаю искреннюю благодарность моему научному руководителю, Лукинову Виталию Леонидовичу, за помощь в подготовке данной работы и поддержку.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ