МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«Новосибирский национальный исследовательский государственный университет»

(Новосибирский государственный университет)

Структурное подразделение Новосибирского государственного университета –

Высший колледж информатики НГУ

КАФЕДРА ЕСТЕСТВЕННОНАУЧНЫХ ДИСЦИПЛИН

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

Дипломный проект

на квалификацию техник

Научный руководитель

к.ф.-м.н. ИВМиМГ СО РАН

Лукинов В.Л.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2017 г.

Студент IV курса,

гр. 14214

Кошкарева С. В.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2017 г.

Новосибирск

2017

СОДЕРЖАНИЕ

[ПЕРЕЧЕНЬ ТЕРМИНОВ И СОКРАЩЕНИЙ 3](#_Toc493531900)

[ВВЕДЕНИЕ 4](#_Toc493531901)

[1. ПОСТАНОВКА ЗАДАЧИ 7](#_Toc493531902)

[1.1. Описание предметной области 7](#_Toc493531903)

[1.2. Формулировка задачи 7](#_Toc493531904)

[1.3. Функциональные требования 8](#_Toc493531905)

[1.4. Нефункциональные требования 9](#_Toc493531906)

[1.5. Характеристики выбранных программных средств 9](#_Toc493531907)

[2. РЕАЛИЗАЦИЯ 10](#_Toc493531908)

[2.1. ООП модель 10](#_Toc493531909)

[2.1.1. Описание алгоритма поиска опечаток 15](#_Toc493531910)

[2.1.1.1. Реализация метода FindMisprints() для дискретных и категориальных значений 15](#_Toc493531911)

[2.1.1.2. Реализация метода FindMisprints() для непрерывных значений 16](#_Toc493531912)

[2.1.1.3. Реализация метода FindMisprints() для значений, содержащих даты 16](#_Toc493531913)

[2.1.2. Описание алгоритма поиска выбросов 17](#_Toc493531914)

[2.1.3. Описание алгоритма поиска непоследовательных дат 20](#_Toc493531915)

[2.1.4. Описание функционала 20](#_Toc493531916)

[3. РАЗРАБОТКА БИБЛИОТЕКИ 22](#_Toc493531917)

[4. ОТЛАДКА И ТЕСТИРОВАНИЕ 23](#_Toc493531918)

[РЕЗУЛЬТАТЫ 24](#_Toc493531919)

[ЗАКЛЮЧЕНИЕ 25](#_Toc493531920)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26](#_Toc493531921)

[ПРИЛОЖЕНИЯ 27](#_Toc493531922)

ПЕРЕЧЕНЬ ТЕРМИНОВ И СОКРАЩЕНИЙ

ВВЕДЕНИЕ

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

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 отрасль – доказательная медицина. Доказательная медицина подразумевает такой подход к медицинской практике, при котором каждое решение, относящееся к выбору метода лечения, должно 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 частой проблемой. Поэтому, прежде чем начать применять статистические методы, обрабатываемые данные следует привести к «каноническому» виду. Для этого необходимо идентифицировать возможные проблемы, а в дальнейшем, либо их удалить, либо заменить имеющиеся пропуски и опечатки разумными значениями. Еще одной важной проблемой является наличие выбросов. Под «выбросом» понимается значение, которое слишком велико или слишком мало по сравнению с большинством других имеющихся значений. Чувствительность разных статистических методов к наличию выбросов в данных различна. Так, наличие выбросов может сделать использование некоторых 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, таких как: пропущенные данные, выход величин за определенные пределы, непоследовательность дат, нарушение нормальности распределения. На данный момент валидизация данных зачастую осуществляется при помощи визуальной «ручной» проверки. Программная проверка существенно ускорит дальнейший анализ, а так же окажется более эффективной по сравнению с проверкой, выполняемой в ручном режиме.

Разработанная в рамках дипломной работы библиотека, должна решать такие проблемы, а также позволять статистикам в удобной форме анализировать данные исследования.

В ходе выполнения дипломной работы была изучена предметная область, разработана и описана архитектура библиотеки, идентифицирующей потенциальные проблемы исследования данных на начальном этапе анализа, и рассмотрен пример ее применения на реальных задачах.

1. ПОСТАНОВКА ЗАДАЧИ
   1. Оп**исание предметной области**

В большинстве случаев клинические исследования проводятся в параллельных группах. При проведении исследований в параллельных группах испытуемые двух или более групп получают различные курсы лечения или различные дозы лекарств. Для достижения статистической достоверности испытуемые распределяются по группам методом случайной выборки. Модели исследований в параллельных группах считаются оптимальными для определения эффектов лечения и формулирования выводов на основе полученных результатов.

Ввиду того, что объемы данных и размеры групп (выборок) могут сильно варьировать, а данные быть весьма разнообразными, возникает необходимость в более гибком подходе и выборе методов статистического анализа, подходящих конкретной задаче.

Методы статистического анализа делятся на параметрические и непараметрические. Параметрические методы имеют более высокую точность и эффективность по сравнению, но имеют ограничения. Эти ограничения накладываются на входные данные - они должны быть нормально распределены. Поэтому, перед применением статистических моделей обязательно нужно удостовериться, что данные имеют нормальное распределение.

При статистическом анализе данных клинических исследований необходимо выполнять рутинные процедуры, поддающиеся автоматизации. Создание единого промышленного программного комплекса алгоритмов статистического анализа данных клинических исследований в параллельных группах, реализованных в виде R-библиотеки с сопутствующей документацией, позволит существенно сократить время проведения статистического анализа и поможет проводить более качественные исследования в сжатые сроки.

* 1. Формулировка задачи

Статистический анализ является неотъемлемой частью практически любого исследования, и только с его помощью можно объективно судить о результатах исследования пополнить доказательную базу.

Целью работы являлась реализация …

Были поставлены и решены следующие задачи:

Программирование осуществлялось на языке R, с использованием следующих пакетов:

* base
* methods
* utils
* grDevices
* plyr
* xlsx
* devtools
* roxygen2

Использовалась система контроля версий git и репозиторий на сервере GitHub.

* 1. Функциональные требования

В рамках дипломной работы были поставлены следующие задачи:

* Поиск пропущенных значений (незаполненных полей).
* Поиск опечаток.
* Поиск «выдающихся значений» или «выбросов».
* Исследование нормальности распределений методами Шапиро-Уилка, Андерсона-Дарлинга, Крамера-фон Мизеса, Лиллиефорса, Шапиро-Франчиа, вывод результатов в виде таблицы в .doc-файле или .csv-файле.
* Сравнение числовых показателей в 2 группах U-критерием Манна-Уитни или t-критерием Стьюдента в зависимости от типа распределений показателя или по выбору пользователя, вывод результатов в виде таблицы в .doc файле или .csv файле, построение графиков boxplot.
* Сравнение категориальных показателей в двух группах точным двусторонним критерием Фишера или критерием хи-квадрат в зависимости от типа распределений показателя, или по выбору пользователя, вывод результатов в виде таблицы в .doc-файле или .csv-файле, построение графиков в виде столбчатых диаграмм.
  1. Нефункциональные требования
* Ввод данных должен осуществляется 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 или Bitbucket.
* Тестирование созданной библиотеки.
* Использование встроенной графической системы ggplot2.
  1. Характеристики выбранных программных средств

Для написания проекта 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 рефлексивное программирование, т.е. является мультипарадигменным языком.

Язык R был разработан на основе языка программирования S. С 2011 года он поддерживается и развивается организацией R Foundation. 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.

R имеет обширное сообщество и развитую систему поддержки, включающую обновление компонентов среды, интерактивную помощь и различные образовательные ресурсы.

RStudio представляет собой бесплатную интегрированную среду разработки (IDE) для R. Благодаря ряду своих особенностей этот активно развивающийся программный продукт делает работу с R очень удобной.

1. РЕАЛИЗАЦИЯ
   1. щ
   2. ООП модель

Поставленные задачи были реализованы с использованием ООП модели S4 языка R. Созданная логическая структура модели имеет три составляющих:

* Файлы
* Колонки таблицы
* Типы ошибок

Ниже, на рисунке представлена диаграмма классов ...



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 directory, которое используется для назначения каталога файловой системы, в которой будет создан файл-отчет. Полный путь к файлу генерируется методом setFileReportDirectory(), путем добавления к указанной 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…

Структура данных таблицы, с которой работает библиотека, может состоять из четырех типов значений: дата, непрерывные, дискретные и категориальные (номинальные). Непрерывные переменные (continuous variables) могут принимать любые численные значения, которые естественным образом упорядочены на числовой оси (например, рост, вес). Дискретные переменные (discrete variables) 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 (categorial variables) 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. Структура колонок таблицы представлена на рисунке…



Родительский класс Column имеет поле column\_index, в котором хранится номер колонки в исследуемой таблице, и два метода geColumnIndex() 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 вызовов метода FindMisprints() для поиска опечаток и метода FindOutliers() для поиска выбросов.

Объекты класса Discrete описывают такие колонки таблицы, значения которых представляют собой дискретные переменные. Для реализации не только поиска, но также исправления опечаток была разработана структура словаря, состоящая из множества ключей и соответствующего этим ключам одного значения. 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 п. 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 Binary(), который 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 вызов метода FindMisprints(), но для каждого из классов существует своя реализация данного метода, более подробно это описано в пункте…

Как было сказано ранее, процесс валидации «сырых» данных должен выявлять следующие типы ошибок:

* Опечатки.
* Непоследовательные даты.
* Выбросы.
* Пропущенные значения.

Для каждого из перечисленных типов ошибок был создан класс, все вместе они объединены в единую иерархическую структуру На рисунке изображена логическая структура опечаток.



Родительский класс Error имеет следующие поля: indices, которое хранит индексы ячеек таблицы, содержащих ошибки, поле style использующееся для хранения названия стиля, который должен быть применен для раскраски неправильно заполненных ячеек, поле title содержащее строку с названием типа ошибки (эта строка, по сути, 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 рабой 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. Строка сообщения выглядит следующим образом:



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

* + 1. Описание алгоритма 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.

* + - 1. Реализация метода FindMisprints() для дискретных и категориальных 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 вектор, или в более привычном понимании, 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 пустого элемента 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 поверки 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 PrintReport().

Блок-схема

* + - 1. Реализация метода FindMisprints() 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 ее элементов на пропуски.

Далее значение элемента колонки сравнивается с шаблоном, заданным регулярным выражением, которое имеет тип: начало строки одно или более чисел, далее может быть (разделитель точка или запятая (если значение – это десятичная дробь) одно число и более), конец строки. 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().

Блок-схема

* + - 1. Реализация метода FindMisprints() для значений, содержащих даты

Данная реализация схожа с предыдущей, но отличается шаблоном регулярного выражения, применяемым для сравнения со значением элемента. Он имеет вид:.

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 и добавляет индексы таких элементов полю объекта класса Misprint.

Блок-схема

* + 1. Описание алгоритма 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. Графики этого типа очень популярны, поскольку позволяют дать очень полную статистическую характеристику анализируемой совокупности. Кроме того, диаграммы размаха можно использовать для визуальной экспресс-оценки разницы между двумя и более группами (например, между датами отбора проб, экспериментальными группами, участками пространства, и т.п.). Строение получаемых при помощи этой функции "ящиков с усами" представлено ниже на рисунке 1:

[](http://3.bp.blogspot.com/-sqSGopnp0lo/Uvu_wl_dPQI/AAAAAAAAAgs/F2DBOSdfiU4/s1600/boxplot.PNG)

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 х ИКР от этой границы. Аналогично, нижний "ус" простирается от нижней границы "ящика" до наименьшего выборочного значения, находящегося в пределах расстояния 1.5 х ИКР от этой границы. Наблюдения, находящиеся за пределами "усов", потенциально могут быть выбросами.

Блок-схема

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 численные, вместо этого они представляются как значения строкового типа. Именно поэтому, для того, чтобы включить эти значения в анализ выбросов, был использован шаблон, заданный при помощи регулярного выражения: начало строки одно или более чисел разделитель точка или запятая одно и более чисел конец 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 численному, и выполняется поиск выбросов с помощью boxplot.stats(), которая анализирует сразу весь массив. Результат работы функции хранится в виде сложной структуры массива массивов. Значения (не их индексы) из переданного массива, потенциально являющимися выбросами, содержатся в элементе out данной структуры. Если он не пустой, то его значения 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).

Однако всегда следует внимательно относиться подобным нестандартным наблюдениям - они вполне могут оказаться "нормальными" для исследуемой совокупности, и поэтому не должны удаляться из анализа без дополнительного расследования причин их появления.

* + 1. Описание алгоритма поиска непоследовательных дат

Метод FindDateMisprints() реализует поиск непоследовательных дат внутри таблицы. Одними 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.

* + 1. Описание функционала

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 данного функционала является метод TableColumnForExcel(), который работает с объектом класса…. 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 преобразуется 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, о чем сообщается в шапке результирующей 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.

1. РАЗРАБОТКА БИБЛИОТЕКИ

После написания классов и методов, все файлы были объединены в библиотеку. Для этого использовался вспомогательные библиотеки devtools и roxygen2.

1. ОТЛАДКА И ТЕСТИРОВАНИЕ

* Проблема с перезаписью файлов отчета и раскрашенной таблицы, добавить время в название файла.
* Проблема интерпретации значений в колонках средой R, из-за опечаток числовые значения становятся строками.
* Проблема с созданием единой таблицы, в которой содержатся значения каждой колонки и частота их встречаемости, т.к. длина столбцов получается разной, а для компоновки таблицы данных или матрицы они должны быть одной длины.

РЕЗУЛЬТАТЫ

Вставить скриншоты таблиц (до и после), картинки с графиками

ЗАКЛЮЧЕНИЕ

В ходе работы были проведен обзор средств работы с большими данными и на их основе разработана система в рамках лямбда архитектуры. Система удовлетворяет всем поставленным требованиям:

• Легко горизонтально масштабируема

• Обладает высокой отказоустойчивостью

• Обладает высоким быстродействием

• Легко расширяема на разные источники данных

• Реализует требуемый функционал

В рамках работы была доказана целесообразность применения технологий работы с большими данными, применительно к области улучшения безопасности дорожного движения. На основании результатов, полученных с помощью этой системы, могут быть предприняты мероприятия по снижению аварийности дорог. Также система эффективна как мониторинговая.

Библиотека обладает большими возможностями и потенциалом к дальнейшему развитию и улучшению. Таким образом, все поставленные задачи в ходе работы были выполнены, а цель работы можно считать достигнутой.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

ПРИЛОЖЕНИЯ

Здесь будет вставлен код примера работы библиотеки с реальной таблицей