

<sup>1</sup> Release 0.0001.0001

<sup>2</sup> Цифровая экономика

<sup>3</sup> Искусственный интеллект  
<sup>4</sup> в оценочной деятельности

<sup>5</sup> Практическое руководство по разработке систем поддержки  
<sup>6</sup> принятия решений оценщиками с использованием языков  
<sup>7</sup> программирования R и Python

<sup>8</sup> К. А. Мурашев

<sup>9</sup> 13 сентября 2021 г.

УДК 519(2+8+682)+004.891.2+330.4+338.5

ББК 16.6+22(16+17)+65.25

ГРНТИ 27.43.51+28.23.35+28.23.29+28.23.37+83.03.51

М91

Искусственный интеллект в оценочной деятельности: практическое руководство по разработке систем поддержки принятия решений оценщиками с использованием языков программирования R и Python / К. А. Мурашев — Inkeri, Санкт-Петербург, 12 августа 2021 г. – 13 сентября 2021 г., 94 с.

Данное произведение является результатом интеллектуальной деятельности и объектом авторского права. Распространяется на условиях лицензии [Creative Commons Attribution-Share Alike 4.0 International \(CC BY-SA 4.0\)](#), оригинальный текст которой доступен по [ссылке](#) [6], перевод которого на русский язык доступен по [ссылке](#) [7]. Разрешается копировать, распространять, воспроизводить, исполнять, перерабатывать, исправлять и развивать произведение либо любую его часть в том числе и в коммерческих целях при условии указания авторства и лицензирования производных работ на аналогичных условиях. Все новые произведения, основанные на произведении, распространяемом на условиях данной лицензии, должны распространяться на условиях аналогичной лицензии, следовательно все производные произведения также будет разрешено распространять, изменять, а также использовать любым образом, в т. ч. и в коммерческих целях.

Программный код, разработанный автором и использованный для решения задач, описанных в данном произведении, распространяется на условиях лицензии [Apache License Version 2.0](#) [3], оригинальный текст которой доступен по [ссылке](#) [19], перевод текста которой на русский язык доступен по [ссылке](#) [3]. Программный код на языке R [87], разработанный автором, а также иные рабочие материалы к нему доступны по [ссылке](#) на портале Github [62], а также по [запасной ссылке](#) [63]. Программный код на языке Python [20], разработанный автором, а также иные рабочие материалы к нему доступны по [ссылке](#) на портале Github [64], а также по [запасной ссылке](#) [65].

В процессе разработки данного материала равно как и программного кода автор использовал операционную систему [Kubuntu](#) [14]. Для подготовки данного материала использовался язык  $\text{\TeX}$  [84] с набором макрорасширений  $\text{\LaTeX} 2_{\epsilon}$  [85]. Конкретная техническая реализация заключается в использовании дистрибутива [TexLive](#) [86], редактора  $\text{\LaTeX}$  [53], компилятора  $\text{PdfLaTeX}$  и системы цитирования  $\text{BibLaTeX}$ /Biber. Исходный код и дополнительные файлы, необходимые для его компиляции, доступны по [ссылке](#) на портале Github [67], а также по [запасной ссылке](#) [68].

Материал подготовлен в форме гипертекста: ссылки на ресурсы, размещённые в информационно-телекоммуникационной сети «Интернет» [156], выделены синим (blue) цветом, внутренние перекрёстные ссылки выделены красным (red) цветом, библиографические ссылки выделены зелёным (green) цветом. При подготовке данного материала использовался шаблон [KOMAScript Book](#) [47]. В целях облегчения понимания согласования слов в сложноподчинённых предложениях либо их последовательности в тексте реализована графическая разметка, позволяющая понять

структуру предложения: слова, согласованные между собой внутри предложения, подчёркнуты одинаковыми линиями, данное решение применяется только в тех предложениях, в которых, по мнению автора, возможно неоднозначное толкование в части согласования слов внутри него.

Данный материал выпускается в соответствии с философией *Rolling Release* [108], что означает что он будет непрерывно дорабатываться по мере обнаружения ошибок и неточностей, а также в целях улучшения внешнего вида. Идентификатором, предназначенным для определения версии материала, служат её номер и дата релиза, указанные на титульном листе, а также в колонтитулах. История версий приводится в таблице 0.1 на следующей странице-4. Актуальная версия перевода в формате PDF доступна по [ссылке](#) [67], а также по [запасной ссылке](#) [68].

В целях соответствия принципам [устойчивого развития](#) [43, 113], установленным в частности Стратегией [The European Green Deal](#) [71] и являющимся приоритетными для [Единой Европы](#) [36, 17, 99], а также содействия достижению [углеродной нейтральности](#) [93] рекомендуется использовать материал исключительно в электронной форме без распечатывания на бумаге.

Для связи с автором данного перевода можно использовать

- любой клиент, совместимый с протоколом [Tox](#) [74, 114], Tox ID = 2E71 CA29 AF96 DEF6 ABC0 55BA 4314 BCB4 072A 60EC C2B1 0299 04F8 5B26 6673 C31D 8C90 7E19 3B35;
- адрес электронной почты: [kirill.murashev@tutanota.de](mailto:kirill.murashev@tutanota.de);
- <https://www.facebook.com/murashev.kirill/> [1];

Перед началом работы рекомендуется установить Git (см. подраздел [2.4.1 на с. 75–76](#)), выбрать каталог на локальном компьютере и выполнить следующие команды:

```
$ git clone git@github.com:Kirill-Murashev/  
AI_for_valuers_book.git  
$ git clone git@github.com:Kirill-Murashev/  
AI_for_valuers_R_source.git  
$ git clone git@github.com:Kirill-Murashev/  
AI_for_valuers_Python_source.git
```

## История версий

Таблица 0.0.1: История версий материала

№	Номер версии	Дата	Автор	Описание
0	1	2	3	4
1	0.0001.0001	2021-08-14	KAM	Initial

# Оглавление

88	<b>1. Предисловие</b>	<b>21</b>
89	<b>2. Технологическая основа</b>	<b>29</b>
90	2.1. Параметры использованного оборудования и программного обеспечения	29
91	2.2. Обоснование выбора языков R и Python в качестве средства анализа	
92	данных	29
93	2.2.1. Обоснование отказа от использования табличных процессоров	
94	в качестве средства анализа данных	29
95	2.2.2. R или Python	31
96	2.2.2.1. Общие моменты	31
97	2.2.2.2. Современное состояние	33
98	2.3. Система контроля версий Git	34
99	2.3.1. Общие сведения	34
100	2.3.2. Хеш-функции	37
101	2.3.3. Начало работы с Git и основные команды	40
102	2.3.4. Исключение файлов из списка отслеживания	54
103	2.3.5. Ветки проекта, указатели branch и Head	61
104	2.3.6. Работа с Github	65
105	2.3.6.1. Начало	65
106	2.3.6.2. Настройка соединения с удалённым репозиторием по-	
107	средством протокола SSH	66
108	2.3.6.2.1. Проверка наличия существующих SSH-ключей.	66
109	2.3.6.2.2. Генерация новой пары ключей.	66
110	2.3.6.2.3. Добавление публичного ключа на портал GitHub.	68
111	2.3.6.3. Создание и установка GPG ключа.	69
112	2.3.6.3.1. Основные сведения.	69
113	2.3.6.3.2. Проверка наличия существующих ключей.	70
114	2.3.6.3.3. Генерация пары ключей GPG	70
115	2.3.6.3.4. Добавление публичного ключа на портал GitHub.	71
116	2.3.6.4. Установление связи между локальным и удалённым	
117	репозиториями	72
118	2.3.6.4.1. Отправка содержимого локального репозито-	
119	рия в удалённый.	72

120	2.3.6.4.2. Получение содержимого удалённого репозитория. . . . .	73
121	2.3.6.4.3. Обновление репозитория. . . . .	73
122	2.3.7. Работа с Git в IDE . . . . .	74
123	2.3.7.1. Работа в RStudio . . . . .	74
124	2.3.7.2. Работа в Spyder . . . . .	74
125	2.3.7.3. Работа в PyCharm . . . . .	74
126	2.3.8. Заключение . . . . .	74
127	2.4. Установка и настройка . . . . .	75
128	2.4.1. Git . . . . .	75
129	2.4.1.1. Установка на операционных системах, основанных на Debian: Debian, Ubuntu, Mint и т. п. . . . .	75
130	2.4.1.2. Установка на операционной системе Windows . . . . .	75
131	2.4.1.3. Установка на macOS . . . . .	75
132	2.4.2. R . . . . .	76
133	2.4.2.1. Установка на операционных системах, основанных на Debian: Debian, Ubuntu, Mint и т. п. . . . .	76
134	2.4.2.1.1. Установка на операционных системах Ubuntu, Mint и производных от них. . . . .	76
135	2.4.2.1.2. Установка на операционной системе Debian. . . . .	77
136	2.4.2.2. Установка на операционных системах Windows и macOS . . . . .	78
137	2.4.3. RStudio . . . . .	78
138	2.4.4. Python . . . . .	79
139	2.4.5. Spyder . . . . .	79
140	2.4.6. PyCharm . . . . .	79
141	2.4.7. SQL . . . . .	79
142	<b>3. Математическая основа анализа данных</b>	<b>80</b>
143	<b>4. Основные понятия</b>	<b>81</b>
144	4.1. Что было раньше: курица или яйцо? Соотношение понятий statistics, machine learning, data mining, artificial intelligence . . . . .	81
145	<b>5. Каллиграфия анализа данных: краткое стилевое руководство</b>	<b>88</b>
146	5.1. Стилесое руководство по R . . . . .	88
147	5.1.1. Имена файлов и путей к ним . . . . .	88
148	5.1.2. Синтаксис . . . . .	89
149	5.1.2.1. Имена классов, функций и переменных . . . . .	89
150	5.1.2.2. Использование пробелов . . . . .	90
151	5.2. Стилесое руководство по Python . . . . .	93
152	<b>6. Автоматизированный сбор данных</b>	<b>94</b>

## 158 List of Algorithms

Рабочая версия

## Список иллюстраций

160	2.3.1.Локальная система контроля версий . . . . .	35
161	2.3.2.Схема работы централизованной системы контроля версий . . . . .	36
162	2.3.3.Схема работы распределённой системы контроля версий . . . . .	37
163	2.3.4.Общая схема работы Git . . . . .	38
164	2.3.5.Пример вычисления хеша . . . . .	39
165	2.3.6.Схема состояний файлов в системе Git . . . . .	43
166	2.3.7.Схема работы указателя Head . . . . .	61
167	2.3.8.Схема указателей Head и Branch . . . . .	62
168	2.3.9.Состояние репозитория после переноса указателя Head на ветку Develop	63
169	2.3.10Состояние репозитория при наличии нескольких веток . . . . .	64



## 170 Список таблиц

171	0.0.1 История версий материала . . . . .	4
172	0.0.2. . . . .	20
173	2.1.1.Параметры использованного оборудования . . . . .	29
174	2.1.2.Параметры использованного программного обеспечения . . . . .	30
175	4.1.1.Перечень понятий, описывающих группы методов анализа данных .	81
176	4.1.2.Обобщение сведений . . . . .	87

## Список литературы

- [1] URL: <https://www.facebook.com/murashev.kirill/> (дата обр. 28.07.2021).
- [2] Royal Institution Surveyors of Chartered (RICS). *RICS Valuation — Global Standards*. English. UK, London: RICS, 28 нояб. 2019. URL: <https://www.rics.org/eu/upholding-professional-standards/sector-standards/valuation/red-book/red-book-global/> (дата обр. 10.06.2020).
- [3] *Apache 2.0*. URL: [http://licenseit.ru/wiki/index.php/Apache\\_License\\_version\\_2.0#.D0.A2.D0.B5.D0.BA.D1.81.D1.82\\_.D0.BB.D0.B8.D1.86.D0.B5.D0.BD.D0.B7.D0.B8.D0.B8](http://licenseit.ru/wiki/index.php/Apache_License_version_2.0#.D0.A2.D0.B5.D0.BA.D1.81.D1.82_.D0.BB.D0.B8.D1.86.D0.B5.D0.BD.D0.B7.D0.B8.D0.B8) (дата обр. 17.08.2021).
- [4] Bioconductor. *Coding Style*. URL: <https://www.bioconductor.org/developers/how-to/coding-style/> (дата обр. 11.09.2021).
- [5] Scott Chacon. *Pro Git book*. Перевод на русский язык. URL: <https://git-scm.com/book/ru/v2> (дата обр. 25.08.2021).
- [6] Creative Commons. *Creative Commons Attribution-ShareAlike 4.0 International*. нояб. 2013. URL: <https://creativecommons.org/licenses/by-sa/4.0/legalcode>.
- [7] Creative Commons. *Creative Commons Attribution-ShareAlike 4.0 International RUS*. нояб. 2013. URL: <https://creativecommons.org/licenses/by-sa/4.0/legalcode.ru>.
- [8] *Constitution of the Federal Republic of Germany*. Английский, немецкий. URL: <https://www.bmi.bund.de/EN/topics/constitution/constitutional-issues/constitutional-issues.html> (дата обр. 06.09.2021).
- [9] Microsoft Corporation. *Microsoft Excel*. Английский. URL: <https://www.microsoft.com/en-us/microsoft-365/excel> (дата обр. 20.08.2021).
- [10] CorVVin. *Хеш-функция, что это такое?* URL: <https://habr.com/en/post/534596/> (дата обр. 25.08.2021).
- [11] *Debian official site*. URL: <https://www.debian.org> (дата обр. 08.09.2021).
- [12] *Debian Packages of R Software*. URL: <https://cran.r-project.org/bin/linux/debian/> (дата обр. 08.09.2021).
- [13] demche. *Занимательная археология: стилевое руководство R под лупой*. 1 июня 2019. URL: <https://habr.com/en/post/454432/> (дата обр. 11.09.2021).

- [14] Kubuntu devs. *Kubuntu official site*. Kubuntu devs. URL: <https://kubuntu.org/> (дата обр. 17.08.2021).
- [15] KDE e.V. *Plasma. KDE community*. Английский. KDE e.V. URL: <https://kde.org/plasma-desktop/> (дата обр. 19.08.2021).
- [16] Ed25519. URL: <https://ed25519.cr.yp.to/> (дата обр. 04.09.2021).
- [17] Institute Greater for a Europe. *Institute for a Greater Europe official site*. URL: <https://www.institutegreatereurope.com/> (дата обр. 15.04.2021).
- [18] StatSoft Europe. *Statistica: official site*. URL: <https://www.statistica.com/en/> (дата обр. 24.08.2021).
- [19] Apache Software Foundation. *Apache LicenseVersion 2.0*. Английский. URL: <https://www.apache.org/licenses/LICENSE-2.0> (дата обр. 17.08.2021).
- [20] Python Software Foundation. Английский. Python Software Foundation. URL: <https://www.python.org/> (дата обр. 17.08.2021).
- [21] The Apache Software Foundation. *OpenOffice Calc*. URL: <https://www.openoffice.org/product/calc.html> (дата обр. 20.08.2021).
- [22] The Document Foundation. *LibreOffice Calc*. Английский. URL: <https://www.libreoffice.org/discover/calc/> (дата обр. 20.08.2021).
- [23] The IFRS Foundation. *IFRS 13 Fair Value Measurement*. UK, London: The IFRS Foundation, 31 янв. 2016. URL: <http://eifrs.ifrs.org/eifrs/bnstandards/en/IFRS13.pdf> (дата обр. 10.06.2020).
- [24] Geeksforgeeks. *Difference between RSA algorithm and DSA*. URL: <https://www.geeksforgeeks.org/difference-between-rsa-algorithm-and-dsa/> (дата обр. 04.09.2021).
- [25] *GeoGebra official site*. URL: <https://www.geogebra.org/> (дата обр. 26.08.2021).
- [26] *Git Download for Windows*. URL: <https://git-scm.com/download/win> (дата обр. 29.08.2021).
- [27] *Git install on macOS*. URL: <https://git-scm.com/download/mac> (дата обр. 29.08.2021).
- [28] *Git official site*. URL: <https://git-scm.com/> (дата обр. 19.08.2021).
- [29] *Git на сервере — Протоколы*. URL: <https://git-scm.com/book/ru/v2/Git-%D0%BD%D0%B0-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%B5-%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D1%8B> (дата обр. 03.09.2021).
- [30] *GitHub Desktop*. URL: <https://desktop.github.com/> (дата обр. 19.08.2021).
- [31] *Github site*. URL: <https://github.com/> (дата обр. 03.09.2021).
- [32] *GNU Privacy Guard Howto*. URL: <https://help.ubuntu.com/community/GnuPrivacyGuardHowto> (дата обр. 06.09.2021).

- [33] *GnuPG — The Universal Crypto Engine*. URL: <https://gnupg.org/software/index.html> (дата обр. 06.09.2021).
- [34] Google. *Google Sheets*. URL: <https://www.google.com/sheets/about/> (дата обр. 20.08.2021).
- [35] Google. *Google's R Style Guide*. URL: <https://google.github.io/styleguide/Rguide.html> (дата обр. 11.09.2021).
- [36] Lisbon-Vladivostok Work group. *Initiative Lisbon-Vladivostok*. URL: <https://lisbon-vladivostok.pro/> (дата обр. 15.04.2021).
- [37] *Hadley Wickham*. URL: <http://hadley.nz/> (дата обр. 11.09.2021).
- [38] *Homebrew*. URL: <https://brew.sh/> (дата обр. 29.08.2021).
- [39] IBM. *SPSS: official page*. URL: <https://www.ibm.com/products/spss-statistics> (дата обр. 24.08.2021).
- [40] IHS Global Inc. *Eviews: official site*. URL: <https://www.eviews.com/home.html> (дата обр. 24.08.2021).
- [41] SAS Institute Inc. *SAS: official site*. URL: [https://www.sas.com/en\\_us/home.html](https://www.sas.com/en_us/home.html) (дата обр. 24.08.2021).
- [42] Intel. *Процессор Intel® Core™ i7-7500U*. Русский. тех. отч. URL: <https://ark.intel.com/content/www/ru/ru/ark/products/95451/intel-core-i7-7500u-processor-4m-cache-up-to-3-50-ghz.html> (дата обр. 19.08.2021).
- [43] Investopedia. *Sustainability*. URL: <https://www.investopedia.com/terms/s/sustainability.asp> (дата обр. 15.04.2021).
- [44] ISO. *Office Open XML*. URL: [https://standards.iso.org/ittf/PubliclyAvailableStandards/c071692\\_ISO\\_IEC\\_29500-4\\_2016.zip](https://standards.iso.org/ittf/PubliclyAvailableStandards/c071692_ISO_IEC_29500-4_2016.zip) (дата обр. 20.08.2021).
- [45] ISO/IEC. *ISO/IEC 10746-2:2009. Information technology — Open distributed processing — Reference model: Foundations — Part 2*. English. под ред. ISO/IEC. Standard. ISO/IEC, 15 дек. 2009. URL: <http://docs.cntd.ru/document/431871894> (дата обр. 01.03.2021).
- [46] ISO/IEC. *ISO/IEC 2382:2015. Information technology — Vocabulary*. English. под ред. ISO/IEC. ISO/IEC, 2015. URL: <https://www.iso.org/obp/ui/#iso:std:iso-iec:2382:ed-1:v1:en> (дата обр. 01.03.2021).
- [47] Markus Kohm. *koma-script — A bundle of versatile classes and packages*. 1994–2020. URL: <https://ctan.org/pkg/koma-script> (дата обр. 28.01.2021).
- [48] *LaTeXDraw official page*. URL: <http://latexdraw.sourceforge.net/> (дата обр. 26.08.2021).
- [49] Licenseit.ru. *GNU General Public License*. URL: [http://licenseit.ru/wiki/index.php/GNU\\_General\\_Public\\_License](http://licenseit.ru/wiki/index.php/GNU_General_Public_License) (дата обр. 23.08.2021).
- [50] Licenseit.ru. *GNU General Public License version 2*. URL: [http://licenseit.ru/wiki/index.php/GNU\\_General\\_Public\\_License\\_version\\_2](http://licenseit.ru/wiki/index.php/GNU_General_Public_License_version_2) (дата обр. 23.08.2021).

- [51] Licenseit.ru. *Python License version 2.1*. URL: [http://licenseit.ru/wiki/index.php/Python\\_License\\_version\\_2.1](http://licenseit.ru/wiki/index.php/Python_License_version_2.1) (дата обр. 23.08.2021).
- [52] StataCorp LLC. *Stata: official site*. URL: <https://www.stata.com/> (дата обр. 24.08.2021).
- [53] *LyX official site*. URL: <https://www.lyx.org/> (дата обр. 28.01.2021).
- [54] Machinelearning.ru. *Интеллектуальный анализ данных*. URL: [http://www.machinelearning.ru/wiki/index.php?title=%D0%98%D0%BD%D1%82%D0%B5%D0%BB%D0%BB%D0%B5%D0%BA%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9\\_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7\\_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85](http://www.machinelearning.ru/wiki/index.php?title=%D0%98%D0%BD%D1%82%D0%B5%D0%BB%D0%BB%D0%B5%D0%BA%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) (дата обр. 09.09.2021).
- [55] Machinelearning.ru. *Машинное обучение*. URL: [http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5\\_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5](http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) (дата обр. 09.09.2021).
- [56] Machinelearning.ru. *Нормальное распределение*. URL: [http://www.machinelearning.ru/wiki/index.php?title=%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5\\_%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5](http://www.machinelearning.ru/wiki/index.php?title=%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5) (дата обр. 02.03.2021).
- [57] Machinelearning.ru. *Параметрические статистические тесты*. URL: [http://www.machinelearning.ru/wiki/index.php?title=%D0%9A%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D0%B8%D1%8F:%D0%9F%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5\\_%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5\\_%D1%82%D0%B5%D1%81%D1%82%D1%8B](http://www.machinelearning.ru/wiki/index.php?title=%D0%9A%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D0%B8%D1%8F:%D0%9F%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D1%8B) (дата обр. 02.03.2021).
- [58] Machinelearning.ru. *Переобучение*. URL: <http://www.machinelearning.ru/wiki/index.php?title=%D0%9F%D0%B5%D1%80%D0%B5%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5> (дата обр. 09.09.2021).
- [59] Machinelearning.ru. *Теория сложности вычислений*. URL: [http://www.machinelearning.ru/wiki/index.php?title=%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F\\_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C](http://www.machinelearning.ru/wiki/index.php?title=%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C) (дата обр. 09.09.2021).
- [60] Machinelearning.ru. *Искусственный интеллект*. URL: [http://www.machinelearning.ru/wiki/index.php?title=%D0%98%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9\\_%D0%B8%D0%BD%D1%82%D0%B5%D0%BB%D0%BB%D0%B5%D0%BA%D1%82](http://www.machinelearning.ru/wiki/index.php?title=%D0%98%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D0%BB%D0%BB%D0%B5%D0%BA%D1%82) (дата обр. 09.09.2021).
- [61] LLC Minitab. *Minitab: official site*. URL: <https://www.minitab.com/en-us/> (дата обр. 24.08.2021).
- [62] Kirill A. Murashev. R. URL: [https://github.com/Kirill-Murashev/AI\\_for\\_valuers\\_R\\_source](https://github.com/Kirill-Murashev/AI_for_valuers_R_source).

- [63] Kirill A. Murashev. R. URL: <https://web.tresorit.com/l/1Zgvt#kBA5FiY0Qtverp8Rjz6gyg>.
- [64] Kirill A. Murashev. R. URL: [https://github.com/Kirill-Murashev/AI\\_for\\_valuers\\_Python\\_source](https://github.com/Kirill-Murashev/AI_for_valuers_Python_source).
- [65] Kirill A. Murashev. R. URL: <https://web.tresorit.com/l/VGZE5#XqySAkmjYODAIcOp1ZWpmg>.
- [66] Kirill A. Murashev. *RICS Valuation — Global Standards 2020. Russian translation.* TeX. 28 июля 2021. URL: <https://web.tresorit.com/l/oFpJF#xr3UGoxLvszsn4vAaHtjqw>.
- [67] Kirill A. Murashev. *Искусственный интеллект в оценочной деятельности: практическое руководство по разработке систем поддержки принятия решений оценщиками с использованием языков программирования R и Python.* Inkeri. URL: [https://github.com/Kirill-Murashev/AI\\_for\\_valuers\\_book](https://github.com/Kirill-Murashev/AI_for_valuers_book).
- [68] Kirill A. Murashev. *Искусственный интеллект в оценочной деятельности: практическое руководство по разработке систем поддержки принятия решений оценщиками с использованием языков программирования R и Python.* Inkeri. URL: [https://web.tresorit.com/l/3xiTP#1p8pFnG\\_9No9izLFd09xaA](https://web.tresorit.com/l/3xiTP#1p8pFnG_9No9izLFd09xaA).
- [69] *Notepad++ site.* URL: <https://notepad-plus-plus.org/> (дата обр. 29.08.2021).
- [70] Linux Kernel Organization. *The Linux Kernel Archives.* Linux Kernel Organization. URL: <https://www.kernel.org/> (дата обр. 26.08.2021).
- [71] European Parliament. *The European Green Deal.* 15 янв. 2020. URL: [https://www.europarl.europa.eu/doceo/document/TA-9-2020-0005\\_EN.html](https://www.europarl.europa.eu/doceo/document/TA-9-2020-0005_EN.html) (дата обр. 15.04.2021).
- [72] Philip Zimmermann — creator of PGP. URL: <https://philzimmermann.com/EN/background/index.html> (дата обр. 06.09.2021).
- [73] Risan Bagja Pradana. *Upgrade Your SSH Key to Ed25519.* URL: <https://medium.com/risa/upgrade-your-ssh-key-to-ed25519-c6e8d60d3c54> (дата обр. 04.09.2021).
- [74] Tox Project. *Tox project official site.* URL: <https://tox.chat/> (дата обр. 09.03.2021).
- [75] Qt. Английский. URL: <https://www.qt.io/> (дата обр. 19.08.2021).
- [76] R Foundation. *The Comprehensive R Archive Network.* URL: <https://cran.r-project.org/> (дата обр. 24.08.2021).
- [77] *R operators.* URL: <https://rdr.io/r/base/Syntax.html> (дата обр. 13.09.2021).
- [78] PBC RStudio. *R Markdown official page.* URL: <https://rmarkdown.rstudio.com/> (дата обр. 24.08.2021).
- [79] *SHA3-512 online hash function.* URL: [https://emn178.github.io/online-tools/sha3\\_512.html](https://emn178.github.io/online-tools/sha3_512.html) (дата обр. 25.08.2021).
- [80] Stackexchange. *RSA vs. DSA for SSH authentication keys.* URL: <https://security.stackexchange.com/questions/5096/rsa-vs-dsa-for-ssh-authentication-keys> (дата обр. 04.09.2021).



- [81] Stackexchange. *What is the difference between data mining, statistics, machine learning and AI?* URL: <https://stats.stackexchange.com/questions/5026/what-is-the-difference-between-data-mining-statistics-machine-learning-and-ai/21669> (дата обр. 09.09.2021).
- [82] Statsoft. *Solving trees*. URL: <http://statsoft.ru/home/textbook/modules/stclatre.html> (дата обр. 20.08.2021).
- [83] PBC Studio. *RStudio official site*. Английский. URL: <https://www.rstudio.com/> (дата обр. 19.08.2021).
- [84] CTAN team. *TeX official site*. English. CTAN Team. URL: <https://www.ctan.org/> (дата обр. 15.11.2020).
- [85] LaTeX team. *LaTeX official site*. English. URL: <https://www.latex-project.org/> (дата обр. 15.11.2020).
- [86] *TeXLive official site*. URL: <https://www.tug.org/texlive/> (дата обр. 15.11.2020).
- [87] The R Foundation. *The R Project for Statistical Computing*. Английский. The R Foundation. URL: <https://www.r-project.org/> (дата обр. 17.08.2021).
- [88] Jeffrey A. Walker. *Applied Statistics for Experimental Biology. Chapter 2 Organization – R Projects and R Notebooks*. Английский. вер. 2021-08-20. 20 авг. 2021. URL: [https://www.middleprofessor.com/files/applied-biostatistics\\_bookdown/\\_book/organization-r-projects-and-r-notebooks.html](https://www.middleprofessor.com/files/applied-biostatistics_bookdown/_book/organization-r-projects-and-r-notebooks.html) (дата обр. 13.09.2021).
- [89] Hadley Alexander Wickham. *Advanced R Style guide*. английский. URL: <http://adv-r.had.co.nz/Style.html> (дата обр. 11.09.2021).
- [90] Hadley Alexander Wickham. *Advanced R: The S3 object system*. URL: <http://adv-r.had.co.nz/S3.html> (дата обр. 13.09.2021).
- [91] Hadley Alexander Wickham. *The tidyverse style guide*. английский. URL: <https://style.tidyverse.org/index.html> (дата обр. 11.09.2021).
- [92] Wikipedia. *Bash (Unix shell)*. URL: [https://en.wikipedia.org/wiki/Bash\\_\(Unix\\_shell\)](https://en.wikipedia.org/wiki/Bash_(Unix_shell)) (дата обр. 02.09.2021).
- [93] Wikipedia. *Carbon neutrality*. URL: [https://en.wikipedia.org/wiki/Carbon\\_neutrality](https://en.wikipedia.org/wiki/Carbon_neutrality) (дата обр. 15.04.2021).
- [94] Wikipedia. *COVID-19 pandemic*. Английский. URL: [https://en.wikipedia.org/wiki/COVID-19\\_pandemic](https://en.wikipedia.org/wiki/COVID-19_pandemic) (дата обр. 18.08.2021).
- [95] Wikipedia. *DSA*. URL: <https://ru.wikipedia.org/wiki/DSA> (дата обр. 04.09.2021).
- [96] Wikipedia. *ECDSA*. URL: <https://ru.wikipedia.org/wiki/ECDSA> (дата обр. 04.09.2021).
- [97] Wikipedia. *Efficient-market hypothesis*. URL: [https://en.wikipedia.org/wiki/Efficient-market\\_hypothesis](https://en.wikipedia.org/wiki/Efficient-market_hypothesis) (дата обр. 29.10.2020).

- 399 [98] Wikipedia. *Euclidean distance*. URL: [https://en.wikipedia.org/wiki/Euclidean\\_distance](https://en.wikipedia.org/wiki/Euclidean_distance) (дата обр. 18.08.2021).  
400
- 401 [99] Wikipedia. *Greater Europe*. URL: [https://en.wikipedia.org/wiki/Greater\\_Europe](https://en.wikipedia.org/wiki/Greater_Europe) (дата обр. 15.04.2021).  
402
- 403 [100] Wikipedia. *HTTPS*. URL: <https://en.wikipedia.org/wiki/HTTPS> (дата обр. 03.09.2021).  
404
- 405 [101] Wikipedia. *Infix notation*. URL: [https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%B8%D0%BA%D1%81%D0%BD%D0%B0%D1%8F\\_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%B8%D0%BA%D1%81%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C) (дата обр. 13.09.2021).  
406  
407
- 408 [102] Wikipedia. *Kelly Johnson (engineer)*. URL: [https://en.wikipedia.org/wiki/Kelly%5C\\_Johnson\\_\(engineer\)](https://en.wikipedia.org/wiki/Kelly%5C_Johnson_(engineer)) (дата обр. 06.11.2020).  
409
- 410 [103] Wikipedia. *KISS principle*. URL: [https://en.wikipedia.org/wiki/KISS\\_principle](https://en.wikipedia.org/wiki/KISS_principle) (дата обр. 06.11.2020).  
411
- 412 [104] Wikipedia. *List of Linux distributions : Debian – based*. URL: [https://en.wikipedia.org/wiki/Category:Debian-based\\_distributions](https://en.wikipedia.org/wiki/Category:Debian-based_distributions) (дата обр. 26.08.2021).  
413  
414
- 415 [105] Wikipedia. *Office Open XML*. URL: [https://ru.wikipedia.org/wiki/Office\\_Open\\_XML](https://ru.wikipedia.org/wiki/Office_Open_XML) (дата обр. 20.08.2021).  
416
- 417 [106] Wikipedia. *PGP*. URL: <https://ru.wikipedia.org/wiki/PGP> (дата обр. 06.09.2021).  
418
- 419 [107] Wikipedia. *Robert Gentleman*. URL: [https://en.wikipedia.org/wiki/Robert\\_Gentleman\\_\(statistician\)](https://en.wikipedia.org/wiki/Robert_Gentleman_(statistician)) (дата обр. 25.08.2021).  
420
- 421 [108] Wikipedia. *Rolling Release*. URL: [https://ru.wikipedia.org/wiki/Rolling\\_release](https://ru.wikipedia.org/wiki/Rolling_release) (дата обр. 28.01.2021).  
422
- 423 [109] Wikipedia. *Ross Ihaka*. URL: [https://en.wikipedia.org/wiki/Ross\\_Ihaka](https://en.wikipedia.org/wiki/Ross_Ihaka) (дата обр. 25.08.2021).  
424
- 425 [110] Wikipedia. *RSA*. URL: <https://ru.wikipedia.org/wiki/RSA> (дата обр. 04.09.2021).  
426
- 427 [111] Wikipedia. *SHA-3*. URL: <https://ru.wikipedia.org/wiki/SHA-3> (дата обр. 26.08.2021).  
428
- 429 [112] Wikipedia. *SSH*. URL: <https://ru.wikipedia.org/wiki/SSH> (дата обр. 03.09.2021).  
430
- 431 [113] Wikipedia. *Sustainability*. English. URL: <https://en.wikipedia.org/wiki/Sustainability> (дата обр. 15.04.2021).  
432
- 433 [114] Wikipedia. *Wikipedia: Tox protocol*. URL: [https://en.wikipedia.org/wiki/Tox\\_\(protocol\)](https://en.wikipedia.org/wiki/Tox_(protocol)) (дата обр. 09.03.2021).  
434



- 435 [115] Wikipedia. *Архитектура компьютера*. Russian. URL: [https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0\\_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%B0](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%B0) (дата обр. 06.08.2021).
- 439 [116] Wikipedia. *Байесовская вероятность*. URL: [https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D0%B5%D1%81%D0%BE%D0%B2%D1%81%D0%BA%D0%B0%D1%8F\\_%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D0%B5%D1%81%D0%BE%D0%B2%D1%81%D0%BA%D0%B0%D1%8F_%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C) (дата обр. 09.09.2021).
- 443 [117] Wikipedia. *Высокоуровневый язык программирования*. URL: [https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9\\_%D1%8F%D0%B7%D1%8B%D0%BA\\_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) (дата обр. 23.08.2021).
- 448 [118] Wikipedia. *Детерминированный алгоритм*. URL: [https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9\\_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) (дата обр. 25.08.2021).
- 452 [119] Wikipedia. *Дискретное логарифмирование*. URL: [https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B5\\_%D0%BB%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B5_%D0%BB%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) (дата обр. 04.09.2021).
- 456 [120] Wikipedia. *Интегрированная среда разработки*. URL: [https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F\\_%D1%81%D1%80%D0%B5%D0%B4%D0%B0\\_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) (дата обр. 29.08.2021).
- 461 [121] Wikipedia. *Класс (программирование)*. URL: [https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81\\_\(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) (дата обр. 13.09.2021).
- 465 [122] Wikipedia. *Коллизия хеш-функции*. URL: [https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%BB%D0%B8%D0%B7%D0%B8%D1%8F\\_%D1%85%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%BB%D0%B8%D0%B7%D0%B8%D1%8F_%D1%85%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8) (дата обр. 25.08.2021).
- 468 [123] Wikipedia. *Конечное поле (поле Галуа)*. URL: [https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D0%BE%D0%B5\\_%D0%BF%D0%BE%D0%BB%D0%B5](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D0%BE%D0%B5_%D0%BF%D0%BE%D0%BB%D0%B5) (дата обр. 04.09.2021).
- 471 [124] Wikipedia. *Кох, Вернер*. URL: [https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%85,\\_%D0%92%D0%B5%D1%80%D0%BD%D0%B5%D1%80](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%85,_%D0%92%D0%B5%D1%80%D0%BD%D0%B5%D1%80) (дата обр. 06.09.2021).
- 473 [125] Wikipedia. *Минус*. URL: <https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BD%D1%83%D1%81> (дата обр. 13.09.2021).

- 475 [126] Wikipedia. *Непараметрическая статистика*. URL: [https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F\\_%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0) (дата обр. 20.08.2021).
- 479 [127] Wikipedia. *Переменная (математика)*. URL: [https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F\\_%D0%B2%D0%B5%D0%BB%D0%B8%D1%87%D0%B8%D0%BD%D0%B0](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B2%D0%B5%D0%BB%D0%B8%D1%87%D0%B8%D0%BD%D0%B0) (дата обр. 20.08.2021).
- 483 [128] Wikipedia. *Переменная (программирование)*. URL: [https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F\\_\(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5\)](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) (дата обр. 20.08.2021).
- 487 [129] Wikipedia. *Полнота по Тьюрингу*. URL: [https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D0%BE%D1%82%D0%B0\\_%D0%BF%D0%BE\\_%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3%D1%83](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D0%BE%D1%82%D0%B0_%D0%BF%D0%BE_%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3%D1%83) (дата обр. 23.08.2021).
- 490 [130] Wikipedia. *Принцип Дирихле*. URL: [https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF\\_%D0%94%D0%B8%D1%80%D0%B8%D1%85%D0%BB%D0%B5\\_\(%D0%BA%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%82%D0%BE%D1%80%D0%B8%D0%BA%D0%B0\)](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF_%D0%94%D0%B8%D1%80%D0%B8%D1%85%D0%BB%D0%B5_(%D0%BA%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%82%D0%BE%D1%80%D0%B8%D0%BA%D0%B0)) (дата обр. 25.08.2021).
- 494 [131] Wikipedia. *Пятецкий-Шapiro, Илья Иосифович*. URL: [https://ru.wikipedia.org/wiki/%D0%9F%D1%8F%D1%82%D0%B5%D1%86%D0%BA%D0%B8%D0%B9-%D0%A8%D0%B0%D0%BF%D0%B8%D1%80%D0%BE,%D0%98%D0%BB%D1%8C%D1%8F\\_%D0%98%D0%BE%D1%81%D0%B8%D1%84%D0%BE%D0%B2%D0%B8%D1%87](https://ru.wikipedia.org/wiki/%D0%9F%D1%8F%D1%82%D0%B5%D1%86%D0%BA%D0%B8%D0%B9-%D0%A8%D0%B0%D0%BF%D0%B8%D1%80%D0%BE,%D0%98%D0%BB%D1%8C%D1%8F_%D0%98%D0%BE%D1%81%D0%B8%D1%84%D0%BE%D0%B2%D0%B8%D1%87) (дата обр. 09.09.2021).
- 499 [132] Wikipedia. *Расстояние городских кварталов*. URL: [https://en.wikipedia.org/wiki/Taxicab\\_geometry](https://en.wikipedia.org/wiki/Taxicab_geometry) (дата обр. 18.08.2021).
- 501 [133] Wikipedia. *Сверхвысокоуровневый язык программирования*. URL: [https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%B5%D1%80%D1%85%D0%B2%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9\\_%D1%8F%D0%B7%D1%8B%D0%BA\\_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%B5%D1%80%D1%85%D0%B2%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) (дата обр. 23.08.2021).
- 507 [134] Wikipedia. *Свободная лицензия*. URL: [https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F\\_%D0%BB%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BB%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F) (дата обр. 23.08.2021).
- 510 [135] Wikipedia. *Свободное программное обеспечение*. Русский. URL: [https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5\\_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5\\_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) (дата обр. 18.08.2021).

- [136] Wikipedia. *Сильная форма Гипотезы эффективного рынка*. URL: [https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%BE%D1%82%D0%B5%D0%B7%D0%B0\\_%D1%8D%D1%84%D1%84%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE\\_%D1%80%D1%8B%D0%BD%D0%BA%D0%B0%D0%A2%D1%80%D0%B8\\_%D1%84%D0%BE%D1%80%D0%BC%D1%8B\\_%D1%80%D1%8B%D0%BD%D0%BE%D1%87%D0%BD%D0%BE%D0%B9\\_%D1%8D%D1%84%D1%84%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D0%B8](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%BE%D1%82%D0%B5%D0%B7%D0%B0_%D1%8D%D1%84%D1%84%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE_%D1%80%D1%8B%D0%BD%D0%BA%D0%B0%D0%A2%D1%80%D0%B8_%D1%84%D0%BE%D1%80%D0%BC%D1%8B_%D1%80%D1%8B%D0%BD%D0%BE%D1%87%D0%BD%D0%BE%D0%B9_%D1%8D%D1%84%D1%84%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D0%B8) (дата обр. 18.08.2021).
- [137] Wikipedia. *Сценарный язык*. URL: [https://ru.wikipedia.org/wiki/%D0%A1%D1%86%D0%B5%D0%BD%D0%B0%D1%80%D0%BD%D1%8B%D0%B9\\_%D1%8F%D0%B7%D1%8B%D0%BA](https://ru.wikipedia.org/wiki/%D0%A1%D1%86%D0%B5%D0%BD%D0%B0%D1%80%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) (дата обр. 23.08.2021).
- [138] Wikipedia. *Форк*. URL: <https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BA> (дата обр. 11.09.2021).
- [139] Wikipedia. *Хеш-функция*. URL: <https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F> (дата обр. 25.08.2021).
- [140] Wikipedia. *Циммерман, Филипп*. URL: <https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D0%BC%D0%BC%D0%B5%D1%80%D0%BC%D0%B0%D0%BD,%D0%A4%D0%B8%D0%BB%D0%B8%D0%BF%D0%BF> (дата обр. 06.09.2021).
- [141] Wikipedia. *Частотная вероятность*. URL: [https://ru.wikipedia.org/wiki/%D0%A7%D0%B0%D1%81%D1%82%D0%BE%D1%82%D0%BD%D0%B0%D1%8F\\_%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C](https://ru.wikipedia.org/wiki/%D0%A7%D0%B0%D1%81%D1%82%D0%BE%D1%82%D0%BD%D0%B0%D1%8F_%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C) (дата обр. 09.09.2021).
- [142] Wikipedia. *Эллиптическая кривая*. URL: [https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%BB%D0%B8%D0%BF%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F\\_%D0%BA%D1%80%D0%B8%D0%B2%D0%B0%D1%8F](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%BB%D0%B8%D0%BF%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BA%D1%80%D0%B8%D0%B2%D0%B0%D1%8F) (дата обр. 04.09.2021).
- [143] *Xcode page*. URL: <https://developer.apple.com/xcode/> (дата обр. 29.08.2021).
- [144] et al Yihui Xie. *formatR: Format R Code Automatically*. 1 июня 2021. URL: <https://cran.r-project.org/web/packages/formatR/index.html> (дата обр. 11.09.2021).
- [145] *Как запустить Bash скрипт в Linux*. URL: <https://wiki.merionet.ru/serveinye-resheniya/63/kak-zapustit-bash-skript-v-linux/> (дата обр. 02.09.2021).
- [146] *Калькулятор недвижимости*. URL: <https://spb.cian.ru/kalkulator-nedvizhimosti/> (дата обр. 10.09.2021).
- [147] Кирилл Кринкин. *Введение в архитектуру ЭВМ и элементы ОС. Курс лекций*. Русский. Computer Science Center. URL: <https://www.youtube.com/watch?v=FzN8zzMRTlw&list=PLlb7e2G7aSpRZ9wDzXI-VYpk59acLF0Ir> (дата обр. 23.08.2021).

- 554 [148] Артём Матяшов. *Git. Большой практический выпуск*. Русский. URL: <https://www.youtube.com/watch?v=SEvR780hGtw> (дата обр. 03.09.2021).
- 555
- 556 [149] связи и массовых коммуникаций Российской Федерации Министерство цифрового развития. *Свободное программное обеспечение в госорганах*. Русский.
- 557 URL: <https://www.gnu.org/philosophy/free-sw.ru.html> (дата обр. 18.08.2021).
- 558
- 559
- 560 [150] Фонд свободного программного обеспечения. *Что такое свободная программа?* Русский. Фонд свободного программного обеспечения. URL: <https://www.gnu.org/philosophy/free-sw.ru.html> (дата обр. 18.08.2021).
- 561
- 562
- 563 [151] Программирование на С и С++. Онлайн справочник программиста на С и С++. *Оператор*. URL: <http://www.c-cpp.ru/books/operatory> (дата обр. 20.08.2021).
- 564
- 565
- 566 [152] Виталий Радченко. *Открытый курс машинного обучения. Тема 5. Композиции: бэггинг, случайный лес*. URL: <https://habr.com/en/company/ods/blog/324402/> (дата обр. 20.08.2021).
- 567
- 568
- 569 [153] Министерство финансов России. *Международный стандарт финансовой отчётности (IFRS) 13 «Оценка справедливой стоимости»*. с изменениями на 11 июля 2016 г. Russian. Russia, Moscow: Минфин России, 28 дек. 2015.
- 570
- 571 URL: <https://normativ.kontur.ru/document?moduleId=1&documentId=326168#10> (дата обр. 10.06.2020).
- 572
- 573
- 574 [154] Министерство цифрового развития Российской Федерации. *Национальная программа «Цифровая экономика Российской Федерации»*. 29 окт. 2020. URL: <https://digital.gov.ru/ru/activity/directions/858/> (дата обр. 29.10.2020).
- 575
- 576
- 577 [155] Министерство экономического развития РФ. *Федеральные стандарты оценки*. URL: [https://www.consultant.ru/document/cons\\_doc\\_LAW\\_126896/](https://www.consultant.ru/document/cons_doc_LAW_126896/).
- 578
- 579 [156] Российская Федерация. *Федеральный Закон «Об информации, информационных технологиях и о защите информации»*. 149-ФЗ. Russian. Russia, Moscow, 14 июля 2006. URL: <https://normativ.kontur.ru/document?moduleId=1&documentId=376603&cwi=22898> (дата обр. 07.07.2020).
- 580
- 581
- 582
- 583 [157] Российская Федерация. *Федеральный закон «Об оценочной деятельности в Российской Федерации»*. 29 июля 1998. URL: <https://normativ.kontur.ru/document?moduleId=1&documentId=396506&cwi=7508> (дата обр. 18.08.2021).
- 584
- 585

Таблица 0.0.2.

# Глава 1.

## Предисловие

«Лучший способ в чём-то  
разобраться — это попробовать  
научить этому другого».  
Народная мудрость

---

«Лучший способ в чём-то  
разобраться до конца — это  
попробовать научить этому  
компьютер».  
Дональд Э. Кнут

---

Целью данной работы является попытка объединения наработок в областях оценочной деятельности и искусственного интеллекта. Автор предпринимает попытку доказать возможность применения современных технологий искусственного интеллекта в сфере оценки имущества, его эффективность и наличие ряда преимуществ относительно иных методов определения стоимости и анализа данных открытых рынков. В условиях заданного руководством России курса на цифровизацию экономики и, в особенности, на развитие технологий искусственного интеллекта [154] внедрение методов машинного обучения в повседневную практику оценщиков представляется логичным и необходимым.

Данная работа писалась в условиях распространения новой коронавирусной инфекции [94], внесшей дополнительный вклад в процессы цифровизации во всём мире. Можно по-разному относиться к проблематике данного явления, однако нельзя отрицать его влияние на общество и технологический уклад ближайшего будущего. Повсеместный переход на технологии искусственного интеллекта, замена человеческого труда машинным, беспрецедентный рост капитализации компаний, сделавших ставку на развитие интеллектуальной собственности, делают невозможным

игнорирование необходимости цифровой трансформации оценочной деятельности в России.

Актуальность предложенного автором исследования заключается во-первых в том, что оно даёт практический инструментарий, позволяющий делать обоснованные, поддающиеся верификации выводы на основе использования исключительно объективных информации и данных,<sup>1</sup> непосредственно наблюдаемых на открытых рынках, без использования каких-либо иных их источников, подверженных субъективному влиянию со стороны их авторов. Во-вторых, предложенные и рассмотренные в данной работе методы обладают весьма широким функционалом, позволяющим использовать их при решении широкого круга задач, выходящих за рамки работы над конкретной оценкой. Важность обеих причин автор видит в том, что на 2021 год в России в сфере оценочной деятельности сложилась ситуация, которую можно охарактеризовать тремя состояниями:

- состояние неопределённости будущего отрасли;
- состояние интеллектуального тупика;
- состояние технологической отсталости.

---

<sup>1</sup>По мнению автора, отличие между информацией и данными заключается в том, что под информацией понимаются:

- знания о предметах, фактах, идеях и т. д., которыми могут обмениваться люди в рамках конкретного контекста [45];
- знания относительно фактов, событий, вещей, идей и понятий, которые в определённом контексте имеют конкретный смысл [46],

таким образом, в контексте данного материала под информацией следует понимать совокупность сведений, образующих логическую схему: теоремы, научные законы, формулы, эмпирические принципы, алгоритмы, методы, законодательные и подзаконные акты и т. п.

Данные же представляют собой:

- формы представления информации, с которыми имеют дело информационные системы и их пользователи [45];
- поддающееся многократной интерпретации представление информации в формализованном виде, пригодном для передачи, связи или обработки [46],

таким образом, в контексте данного материала под данными следует понимать собой совокупность результатов наблюдений о свойствах тех или иных объектов и явлений, выраженных в объективной форме, предполагающей их многократные передачу и обработку.

Например: информацией является знание о том, что для обработки переменных выборки аналогов, имеющих распределение отличное от [нормального](#) [56], в общем случае, некорректно использовать [параметрические методы](#) [57] статистического анализа; данные в этом случае — это непосредственно сама выборка.

Иными словами, оперируя терминологией [архитектуры ЭВМ](#) [115], данные — набор значений переменных, информация — набор инструкций.

Во избежание двусмысленности в тексте данного материала эти термины приводятся именно в тех смыслах, которые описаны выше. В случае необходимости также используется более общий термин «сведения», обобщающий оба вышеуказанных понятия. В ряде случаев, термины используются в соответствии с принятым значением в контексте устоявшихся словосочетаний.



Первая проблема заключается в неопределённости как правового регулирования отрасли, так и её экономики. Введённая около четырёх лет назад система квалификационных аттестатов оценщиков, на которую регулятор, заказчики и, возможно, часть самих оценщиков возлагали надежду как на фильтр, позволяющий оставить в отрасли только квалифицированных специалистов, сократить предложение оценочных услуг и, следовательно, способствовать росту вознаграждений за проведение оценки, не оправдала ожиданий. Несмотря на существенное сокращение числа оценщиков, имеющих право подписывать отчёты об оценке, не произошло никаких значимых изменений ни в части объёма предложения услуг, ни в части уровня цен на них. Фактически произошло лишь дальнейшее развитие уже существовавшего ранее института подписантов отчётов — оценщиков, имеющих необходимые квалификационные документы и выпускающих от своего имени отчёты, в т. ч. и те, в подготовке которых они не принимали участия. В ряде случаев подписант мог и вовсе не читать отчёт либо даже не видеть его в силу своего присутствия в другом регионе, отличном от региона деятельности компании, выпустившей отчёт. При этом, как ни странно, доход таких «специалистов» не вырос существенным образом. Всё это очевидным образом приводит к недовольству регуляторов в адрес оценочного сообщества. В таких условиях следует ожидать неизбежного дальнейшего ужесточения регулирования и усугубления положения добросовестных оценщиков и оценочных компаний. Вместе с тем было бы ошибочным считать, что виной всему являются исключительно сами оценщики и их работодатели. В существенной степени проблемы квалификации и качества работы оценщиков вызваны не их нежеланием добросовестно выполнять свою работу, а отсутствием у заказчиков интереса к серьёзной качественной оценке. Не секрет, что в большинстве случаев оценка является услугой, навязанной требованиями закона либо кредитора, не нужной самому заказчику, которого очевидно волнует не качество отчёта об оценке, а соответствие определённой в нём стоимости ожиданиям и потребностям заказчика, его договорённостям с контрагентами. В таких условиях, с одной стороны, экономика не создаёт спрос на качественную оценку, с другой — сами оценщики не предлагают экономике интересные решения и новые ценности, которые могли бы принести в отрасль дополнительные финансовые потоки.

Вторая проблема тесно связана с первой и выражается в том числе в наблюдаемом на протяжении последних примерно 10 лет падении качества отчётов об оценке и общей примитивизации работы оценщика. Суть данной проблемы можно кратко сформулировать в одной фразе: «раньше молодые оценщики спрашивали „как проанализировать данные рынка и построить модель для оценки“, сейчас они задают вопрос „где взять корректировку на ”Х“”». Установление метода корректировок в качестве доминирующего во всех случаях даже без анализа применимости других методов стало логичным итогом процесса деградации качества отчётов об оценке. При этом источником подобных корректировок чаще всего являются отнюдь не данные открытого рынка. Как и в первом случае винить в этом только самих оценщиков было бы неправильным. В условиях работы в зачастую весьма жёстких временных рамках и за небольшое вознаграждение, оценщик часто лишён возможности провести самостоятельный анализ тех или иных свойств открытого рынка, вследствие

и по причине чего вынужден использовать внешние нерыночные данные в том числе и непроверенного качества. Со временем это становится привычкой, убивающей творчество и стремление к поиску истины.

Третья проблема также неразрывно связана с двумя первыми. Отсутствие конкуренции, основанной на стремлении оказывать как можно более качественные услуги, недостаточная капитализация отрасли, выражающаяся в том числе в относительно невысоких зарплатах оценщиков, не вполне последовательное регулирование отрасли со стороны государства — всё это создаёт условия, при которых у оценщиков отсутствует стимул, а зачастую и возможность внедрять инновации.

Данная работа служит следующей основной цели: дать в руки оценщика инструменты, позволяющие ему просто и быстро извлекать полезные сведения из сырых данных открытых рынков, интерпретировать их, выдвигать гипотезы, выбирать среди них наиболее перспективные и в итоге получать готовые модели предсказания различных свойств объекта оценки, в том числе его стоимости. Есть некоторая надежда, что применение технологий искусственного интеллекта позволит, не увеличивая трудоёмкость, а скорее напротив, снижая её, повысить качество работы оценщика, усилить доказательную силу отчётов об оценке и в итоге позволит создать новые ценности, предлагаемые оценщиками экономике, государству, потребителям, а главное всему обществу.

Особенностью данной работы является её практическая направленность: в тексте содержатся все необходимые инструкции, формулы, описания и фрагменты программного кода либо ссылки на них, необходимые и достаточные для воспроизведения всех рассмотренных методов и их описания в отчётах об оценке.

Данная работа состоит из двух частей. Первая посвящена в большей степени теории, описанию методов, а также применению языка [R](#) [87]. Вторая имеет большую практическую направленность и содержит руководства по применению языка [Python](#) [20]. Объяснение данного факта содержится далее в разделе ССЫЛКА. В работе будут рассмотрены следующие вопросы:

- a) автоматизированный сбор данных с веб-ресурсов;
- b) семантический анализ текстов объявлений;
- c) работа с геоданными;
- d) первичная интерпретация и визуализация данных открытых рынков;
- e) проверка статистических гипотез;
- f) задачи классификации;
- g) корреляционный анализ;
- h) регрессионный анализ;
- i) анализ временных рядов;



- j) задачи многомерного шкалирования;
- k) байесовская статистика;
- l) деревья классификации;
- m) случайные леса;
- n) нейронные сети;
- o) глубокое обучение;
- p) обучение с подкреплением;
- q) нечёткая логика.

Вышеприведённый перечень не является исчерпывающим и будет дорабатываться по мере развития проекта.

Данная работа основана на четырёх основополагающих принципах и предпосылках.

- a) *Принцип «вся информация об активе учтена в его цене».* Данный принцип говорит о том, что существует функциональная зависимость между ценой актива (обязательства) и его свойствами. Он тесно связан с [Гипотезой эффективного рынка \[97\]](#), лежащей в основе технического биржевого анализа. При этом для целей настоящей работы данная гипотеза принимается в её [сильной форме эффективности \[136\]](#). С точки зрения оценщика это означает, что нет необходимости искать какие-либо данные кроме тех, которые непосредственно и объективно наблюдаются на рынке.
- b) *Принцип «максимального использования релевантных наблюдаемых исходных данных и минимального использования ненаблюдаемых исходных данных».* Данный принцип согласуется с требованиями п. 3 [Международного стандарта финансовой отчётности 13 «Оценка справедливой стоимости» \[153\]](#) (IFRS 13 [23]), а также, например, принципами [Всемирных стандартов оценки RICS \[66\]](#) (RICS Valuation — Global Standards [2]) и основывается на них. С точки зрения оценщика данный принцип означает, что лучшая практика оценки заключается в работе непосредственно с данными открытых рынков, а не чьей-либо их интерпретацией, существующей, например, в виде готовых наборов корректировок, порой весьма далёких от реальности.
- c) *Принцип KISS [103]* (keep it simple stupid, вариации: keep it short and simple, keep it simple and straightforward и т. п.), предложенный американским авиационным инженером [Келли Джонсоном \[102\]](#), ставший официальным принципом проектирования и конструирования ВМС США с 1960 г. Данный принцип заключается в том, что при разработке той или иной системы следует использовать самое простое решение из возможных. Применительно к тематике данной работы это означает, что в тех случаях, когда автор сталкивался с проблемой

выбора способа решения задачи в условиях неопределённости преимуществ и недостатков возможных вариантов, он всегда выбирал самый простой способ. Например в задаче кластеризации, выбирая между видами расстояний, автор делает выбор в пользу [евклидова](#) либо [манхэттенского](#) расстояний [98, 132].

- d) *Принцип «не дай алгоритму уничтожить здравый смысл».* Данный принцип означает необходимость самостоятельного осмысления всех результатов выполнения процедур, в т. ч. и промежуточных. Возможны ситуации, когда полученные результаты могут противоречить здравому смыслу и априорным знаниям о предметной области, которыми обладает оценщик либо пользователи его работы. Следует избегать безоговорочного доверия к результатам, выдаваемым алгоритмами. Если построенная модель противоречит априорным знаниям об окружающей реальности, то следует помнить, что другой реальности у нас нет, тогда как модель может быть скорректирована либо заменена на другую.

Все описанные этапы действий описаны таким образом, что позволяют сразу же без каких-либо дополнительных исследований воспроизвести всё, что было реализовано в данной работе. От пользователей потребуется только установить необходимые программные средства, создать свой набор данных для анализа и загрузить его в пакет. Все действия по установке и настройке описаны внутри данного руководства. Важным аспектом является то обстоятельство, что при подготовке данного исследования использовалось исключительно [свободное программное обеспечение](#) [150, 135, 149]. Таким образом, любой читатель сможет воспроизвести все описанные действия без каких-либо затрат на приобретение тех или иных программных продуктов.

От пользователей данного руководства не требуется наличие специальных познаний в области разработки программного обеспечения, software engineering и иных аспектов computer science. Некоторые понятия вроде «класс», «метод», «функция», «оператор», «регулярные выражения» и т. п. термины из сферы программирования могут встречаться в тексте руководства, однако их понимание либо непонимание пользователем не оказывает существенного влияния на восприятие материала в целом. В отдельных случаях, когда понимание термина является существенным, как например в случае с термином «переменная», в тексте руководства приводится подробное объяснение смысла такого термина, доступное для понимания неспециалиста.

Также от пользователей руководства не требуется (хотя и является желательным) глубокое понимание математической статистики, дифференциальных вычислений, линейной алгебры, комбинаторики, методов исследования операций, методов оптимизации и иных разделов математики и математической статистики, хотя и предполагается наличие таких познаний на уровне материала, включённого в школьную программу и программу технических и экономических специальностей вузов России. В тексте руководства приводится описание смысла и техники всех применённых статистических методов, математических операций и вычислений в объёме,

достаточном, по мнению автора, для обеспечения доказательности при использовании методов, рассмотренных в данной работе. Автор всегда приводит ссылки на материалы, подтверждающие приведённые им описания за исключением случаев общеизвестных либо очевидных сведений. Особое внимание автор уделяет соблюдению требований к информации и данным, имеющим существенное значение для определения стоимости объекта оценки, установленных Федеральным законом «Об оценочной деятельности в Российской Федерации» [157], а также Федеральными стандартами оценки [155].

Сведения, приведённые в настоящем руководстве, являются, по мнению автора, достаточными для обеспечения выполнения вышеуказанных требований к информации, содержащейся в отчёте об оценке. Таким образом, использование описаний процедур, приведённых в настоящем руководстве, скорее всего должно быть достаточным при использовании изложенных в нём методик в целях осуществления оценочной деятельности и составлении отчёта об оценке. Однако, автор рекомендует уточнять требования, предъявляемые к отчёту об оценке со стороны саморегулируемой организации, в которой состоит оценщик, а также со стороны заказчиков и регуляторов.

В силу свободного характера лицензии, на условиях которой распространяется данная работа, она, равно как и любая её часть, может быть скопирована, воспроизведена, переработана либо использована любым другим способом любым лицом в т. ч. и в коммерческих целях при условии распространения производных материалов на условиях такой же лицензии. Таким образом, автор рекомендует использовать тексты, приведённые в настоящем руководстве для описания выполненных оценщиком процедур.

По мнению автора, данное руководство и описанные в нём методы могут быть особенно полезны в следующих предметных областях:

- оценка и переоценка залогов и их портфелей;
- контроль за портфелями залогов со стороны регулятора банковской сферы;
- оценка объектов, подлежащих страхованию, и их портфелей со стороны страховщиков;
- оценка объектов со стороны лизинговых компаний;
- оценка больших групп активов внутри холдинговых компаний и предприятий крупного бизнеса;
- мониторинг стоимости государственного и муниципального имущества;
- оценка в целях автоматизированного налогового контроля;
- государственная кадастровая оценка;
- экспертиза отчётов об оценке, контроль за деятельностью оценщиков со стороны СРО.

Иными словами, особенная ценность применения методов искусственного интеллекта в оценке возникает там, где имеет место необходимость максимальной беспристрастности и незаинтересованности в конкретном значении стоимости.

В данном руководстве не содержатся общие выводы касательно параметров открытых рынков как таковых, не выводятся общие формулы, применимые всегда и для всех объектов оценки. Вместо этого в распоряжение пользователей предоставляется набор мощных инструментов, достаточный для моделирования ценообразования на любом открытом рынке, определения стоимости любого объекта оценки на основе его актуальных данных. В случае необходимости пользователь, применяя рассмотренные методы, может самостоятельно разработать предсказательную модель для любых рынков и объектов. Забегая вперёд, можно сказать, что при решении конкретной практической задачи применение всех описанных методов не является обязательным, а если быть точным — явно избыточным. В тексте руководства содержатся рекомендации по выбору методов на основе имеющихся свойств данных, рассматриваются сильные и слабые стороны каждого из них.

Несмотря на изначально кажущуюся сложность и громоздкость методов, при более детальном знакомстве и погружении в проблематику становится ясно, что применение предложенных реализаций методов существенно сокращает время, необходимое для выполнения расчёта относительно других методов сопоставимого качества, а сама процедура сводится к написанию и сохранению нескольких строк кода при первом применении и их вторичному многократному использованию для новых наборов данных при будущих исследованиях.

Автор выражает надежду, что данное руководство станет для кого-то первым шагом на пути изучения языков [R](#) [87] и [Python](#) [20], а также погружения в мир анализа данных, искусственного интеллекта и машинного обучения.

## Глава 2.

## Технологическая основа

### 2.1. Параметры использованного оборудования и программного обеспечения

При выполнении всех описанных в данной работе процедур, равно как и написании её текста использовалась следующая конфигурация оборудования.

Таблица 2.1.1. Параметры использованного оборудования

№	Категория	Модель (характеристика)	Источник
0	1	2	3
1	Процессор	4 × {} Intel ® Core ™ i7-7500U CPU @ 2.70GHz	[42]
2	Память	11741076B	

При выполнении всех описанных в данной работе процедур, равно как и написании её текста использовалась следующая конфигурация программного обеспечения.

Как видно из таблиц 2.1, 2.1 для анализа данных и разработки систем поддержки принятия решений на основе искусственного интеллекта вполне достаточно оборудования, обладающего средними характеристиками, а также свободных или, по крайней мере, бесплатных программных средств.

### 2.2. Обоснование выбора языков R и Python в качестве средства анализа данных

#### 2.2.1. Обоснование отказа от использования табличных процессоров в качестве средства анализа данных

На сегодняшний день очевиден факт того, что доминирующим программным продуктом, используемым в качестве средства выполнения расчётов, в среде русских оценщиков является приложение MS Excel [9]. Следом за ним идут его бесплатные аналоги LibreOffice Calc и OpenOffice Calc [22, 21], первый из которых

Таблица 2.1.2. Параметры использованного программного обеспечения

№	Категория/наименование	Значение/версия	Источник
0	1	2	3
1	Операционная система	Kubuntu 20.04	[14]
2	KDE Plasma	5.18.5	[15]
3	KDE Frameworks	5.68.0	[15]
4	Qt	5.12.8	[75]
5	R	4.1.1 (2021-08-10) "— "Kick Things"	[87]
6	RStudio	1.4.1717	[83]
7	Git	2.25.1	[28]
8	Github Desktop	2.6.3-linux1	[30]
9	Geogebra Classic	6.0.660.0-offline	[25]
10	LaTeXDraw	4.0.3-1	[48]
11	Python	3.8.10	
12	Spyder	3.3.6	
13	PyCharm Community	2021.2.1	
14	Kate	19.12.3	

является также не только бесплатным, но и [свободным программным обеспечением](#) [150, 135, 149]. В ряде случаев используется [Google Sheets](#) [34]. Не оспаривая достоинства этих продуктов, нельзя не сказать о том, что они являются универсальными средствами обработки данных общего назначения и, как любые универсальные средства, сильны своей многофункциональностью и удобством, но не шириной и глубиной проработки всех функций. Во всех вышеуказанных программных продуктах в виде готовых функций реализованы некоторые основные математические и статистические процедуры. Также само собой присутствует возможность выполнения расчётов в виде формул, собираемых вручную из простейших [операторов](#) [151]. Однако возможности этих продуктов для профессионального анализа данных абсолютно недостаточны. Во-первых, в них имеются ограничения на размер и размерность исследуемых данных. Во-вторых, в них отсутствуют средства реализации многих современных методов анализа данных. Если первое ограничение не столь важно для оценщиков, редко имеющих дела с по-настоящему большими наборами данных и существенным числом [переменных](#) [127, 128] в них, второе всё же накладывает непреодолимые ограничения на пределы применимости таких программных продуктов. Например, ни одно из вышеперечисленных приложений не позволяет использовать методы [непараметрической статистики](#) [126] либо, например, решить задачи построения [деревьев классификации](#) [82] и их [случайных лесов](#) [152]. Таким образом, следует признать, что, оставаясь высококачественными универсальными средствами для базовых расчётов, вышеперечисленные приложения не могут быть использованы для профессионального анализа данных на современном уровне.

При этом их использование порой бывает необходимым на первоначальном исследовании



дования. Некоторые исходные данные, предоставляемые оценщику для обработки, содержатся в электронных таблицах. Такие таблицы помимо полезных сведений могут содержать посторонние данные, тексты, графики и изображения. В практике автора был случай предоставления ему для анализа данных в форме электронной таблицы формата `xlsx` [105, 44], имеющей размер около 143 МБ, содержащей помимо подлежащей анализу числовой информации о товарах их рекламные описания в текстовом виде и фотографии, составляющие свыше 90 % размера файла. Тем не менее просмотр исходных данных средствами табличных процессоров и создание нового файла, содержащего только необходимые для анализа данные, нередко является подготовительным этапом процесса анализа. В последующих разделах будут даны практические рекомендации касательно его реализации. По мнению автора, по состоянию на 2021 год лучшим табличным процессором является `LibreOffice Calc` [22], превосходящий `MS Excel` [9] по ряду характеристик.

## 2.2.2. R или Python

### 2.2.2.1. Общие моменты

Можно с уверенностью сказать, что по состоянию на второе полугодие 2021 года доминирующими и самыми массовыми техническими средствами анализа данных, машинного обучения и разработки искусственного интеллекта<sup>1</sup> являются языки программирования `R` [87] и `Python` [20]. Оба они являются **сверхвысокоуровневыми** [133] **сценарными** (скриптовыми) [137] языками программирования. Высокоуровневым называется такой язык программирования, в основу которого заложена сильная абстракция, т. е. свойство описывать данные и операции над ними таким образом, при котором разработчику не требуется глубокое понимание того, как именно машина их обрабатывает и исполняет [117]. **Сверхвысокоуровневым** [133] языком является такой язык программирования, в котором реализована очень сильная абстракция. Иными словами, в отличие от **языков программирования высокого уровня** [117], в коде, разработанном на которых, описывается принцип «как нужно сделать», код, выполненный на **сверхвысокоуровневых языках** [133] описывает лишь принцип «что нужно сделать». **Сценарным** (скриптовым) [137] языком называется такой язык программирования, работа которого основана на исполнении сценариев, т. е. программ, использующих уже готовые компоненты. Таким образом, можно сделать вывод, что **сверхвысокоуровневые языки** лучше всего подходят для тех, кто только начинает погружаться в программирование и не обладает экспертными знаниями в вопросах **архитектуры ЭВМ** [115].<sup>2</sup>

Оба языка распространяются на условиях **свободных лицензий** [134] с незначительными отличиями. `R` распространяется на условиях лицензии `GNU GPL 2` [50], `Python` — на условиях лицензии `Python Software Foundation License` [51], являющейся совместимой с `GNU GPL` [49]. Отличия между ними не имеют никакого

<sup>1</sup>Разница между этими понятиями будет описана далее в ССЫЛКА

<sup>2</sup>Для первичного ознакомления с вопросами архитектуры ЭВМ автор рекомендует просмотреть [данный курс лекций](#) [147].

практического значения для целей настоящего руководства и применения любого из этих языков в оценочной деятельности в целом. Следует лишь знать основной факт: использование этих языков является легальным и бесплатным в том числе и для коммерческих целей. Основное отличие между этими языками заключается в частности в том, что Python — язык общего назначения, широко применяемый в различных областях, тогда как R — специализированный язык статистического анализа и машинного обучения. В целом можно сказать, что задачи анализа данных могут одинаково успешно решаться средствами обоих языков. Также они оба являются [Тьюринг-полными](#) [129] языками.

Преимущества R основаны на том факте, что он изначально был разработан двумя профессиональными статистиками: [Ross Ihaka](#) [109], [Robert Gentleman](#) [107], по первым буквам имён которых он и был назван. Дальнейшее развитие языка также осуществляется прежде всего силами профессиональных математиков и статистиков, вследствие чего для R реализовано значительное количество библиотек, выполняющих практически все доступные на сегодняшнем уровне развитии науки статистические процедуры. Кроме того, можно быть уверенным в абсолютной корректности всех алгоритмов, реализованных в этих библиотеках. К тому же этот язык особенно популярен в академической среде, что означает факт того, что в случае, например, выхода какой-то статьи, описывающей новый статистический метод, можно быть уверенным, что соответствующая библиотека, реализующая этот метод выйдет в ближайшее время либо уже вышла. Кроме того, важным преимуществом R являются очень хорошо проработанные средства вывода графической интерпретации результатов анализа.

Недостатки R, как это часто бывает, следуют из его достоинств. Язык и его библиотеки поддерживаются в первую очередь силами математиков-статистиков, а не программистов, что приводит к тому, что язык относительно плохо оптимизирован с точки зрения software engineering, многие решения выглядят неочевидными и неоптимальными с точки зрения способов обращения к памяти, интерпретации в машинные команды, исполнения на процессоре. Это приводит к высокому потреблению ресурсов машины, в первую очередь памяти, медленному исполнению процедур. При этом, говоря о медленном исполнении, следует понимать относительность этой медлительности. Выполнение команды за 35 мс вместо 7 мс не замечается человеком и обычно не имеет сколько-нибудь определяющего значения. Проблемы с производительностью становятся заметны только при работе с данными большой размерности: миллионы наблюдений, тысячи переменных. В практических задачах, с которыми сталкиваются оценщики, подобная размерность данных выглядит неправдоподобной, вследствие чего можно говорить об отсутствии существенных недостатков языка R для целей применения в оценочной деятельности в целом и в целях задач, решаемых в данном руководстве, в частности. Следующей условной проблемой R является огромное количество библиотек<sup>3</sup> и ещё более огромное количество возможных вариантов решения задач и предлагаемых для этого методов. Даже

<sup>3</sup>По состоянию на 24 августа 2021 существует 18089 официальных библиотек, содержащихся на [официальной странице](#) [76] проекта.



опытный аналитик может растеряться, узнав о том, что его задача может быть решена десятками способов, выбор лучшего из которых сам по себе является нетривиальной задачей. Данную особенность конечно же нельзя считать недостатком самого языка R.

Преимуществом Python является его универсальность и существенно большая распространённость. Освоение основ данного языка для целей одной предметной области может быть полезным в дальнейшем, если по каким-то причинам оценщик захочет решать с его помощью задачи иного класса. Данный язык разработан и поддерживается профессиональными программистами, что означает его относительно приемлемую оптимизацию, превосходящую R, но уступающую, например C++.

К недостаткам Python можно отнести меньшее число библиотек, содержащих статистические процедуры. Кроме того, нет такой же уверенности в безупречности их алгоритмов. При этом следует отметить, что подобные риски присутствуют лишь в новых библиотеках, реализующих экспериментальные либо экзотические статистические процедуры. Для целей оценки как правило вполне достаточно уже относительно отработанных и проверенных библиотек.

Подводя итог, можно сказать, что нет однозначного ответа, какой из вышеупомянутых языков является предпочтительным для целей анализа данных в оценке. R развивается, оптимизируется и всё больше избавляется от «детских болезней» неоптимизированности, для Python создаются новые мощные библиотеки статистического анализа. Поэтому вопрос остаётся открытым.

Следует кратко упомянуть о том, что помимо R и Python в целях анализа данных также используются вендорские программные продукты такие как SAS [41], SPSS [39], Statistica [18], Minitab [61], Stata [52], EvIEWS [40] и ряд других. Однако все они являются платными, при этом стоимость лицензии на самый мощный из них — SAS начинается, как правило, от нескольких десятков тысяч долларов. В остальном, кроме привычного для большинства пользователей графического интерфейса они не имеют явных преимуществ перед R и Python, предоставляя при этом даже меньше возможностей.

#### 2.2.2.2. Современное состояние

Вышеприведённый текст, содержащийся в предыдущей секции (2.2.2.1) был написан автором в 2019 году. За прошедший период произошли некоторые изменения, требующие внимания. В настоящее время Python серьёзно опережает R по распространённости в среде аналитиков данных. Можно говорить о некотором консенсусе, согласно которому R является средством разработки и анализа данных для научных целей, тогда как Python применяется в бизнес среде. Несмотря на это, автор считает, что в целях анализа данных данные языки вполне взаимозаменяемы. Некоторые библиотеки портированы из одного из них в другой. При этом нельзя не признать, что за последние годы R существенно сдал позиции в пользу Python. В особенности это справедливо именно для российского рынка разработки систем анализа данных. Определённый пик интереса к R в России имел место в 2015–2017 годах, после чего его популярность пошла на спад. В мире пик интереса к R пришёлся на 2016–2018

годы после чего его популярность стабилизировалась. Язык продолжает активно развиваться.

В российской практике коммерческого анализа данных его заказчики, как правило, требуют реализации на Python, применение вместо него R чаще всего приходится обосновывать отдельно. Таким образом, можно говорить о том, что применение Python де факто является стандартом. Кроме того, продвижению Python во всём мире способствует позиция компаний интернет-гигантов, использующих его в своих системах машинного обучения. Следующим фактором успеха Python является его широкое распространение в теме разработки нейронных сетей, также являющееся следствием практик крупных IT-компаний. Также Python широко распространён и за пределами области анализа данных, что означает существенно большее число специалистов, владеющих им. При этом для R разработан ряд уникальных отраслевых библиотек, содержащих специфические функции. R безоговорочно лидирует в области биоинформатики, моделирования химических процессов, социологии.

При этом, R по-прежнему предоставляет существенно более широкие возможности визуализации, а также позволяет легко разрабатывать веб-интерфейсы посредством [Shiny](#). R имеет отличный инструмент написания документации к коду в процессе разработки самого кода — [R Markdown](#).

Подводя итоги, можно сказать о том, что современным оценщикам следует иметь навыки разработки и анализа данных с использованием обоих этих языков: R поможет применять самые свежие методы и создавать качественные понятные пользователям описания и визуализации, Python пригодится там, где требуется разработка серьёзной промышленной системы, предназначенной для многократного выполнения одинаковых задач. В целом же можно повторить основной тезис: данные языки в существенной степени взаимозаменяемы.

## 2.3. Система контроля версий Git

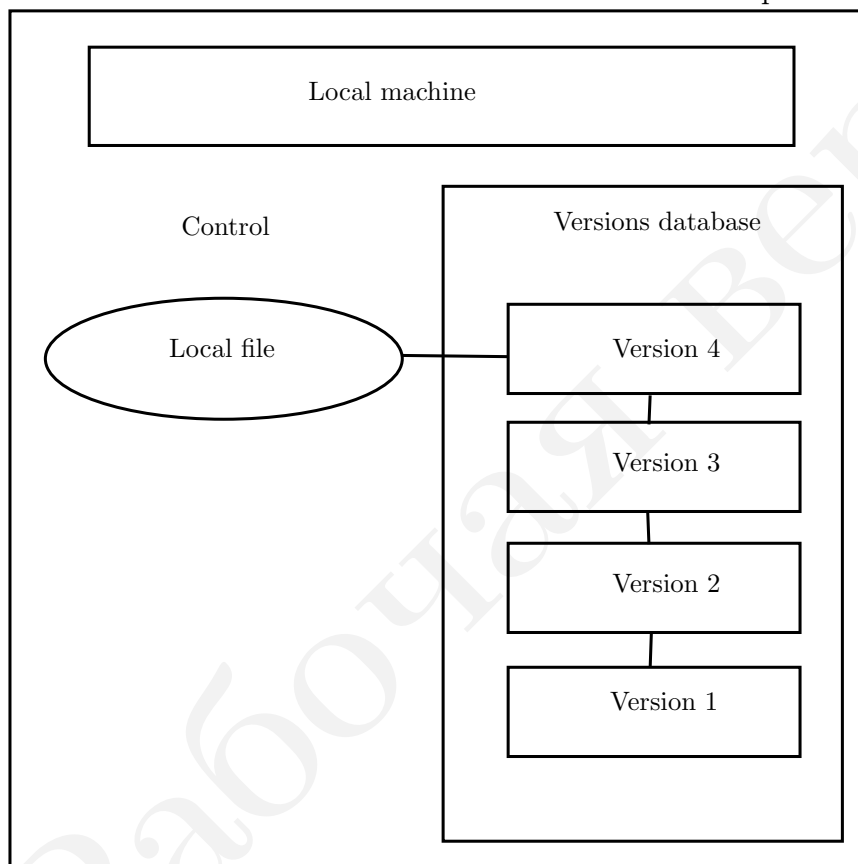
### 2.3.1. Общие сведения

Данный раздел не имеет отношения непосредственно к анализу данных, однако содержит сведения, полезные для комфортной работы при его осуществлении. Кроме того, использование систем контроля версий де факто является стандартом при любой серьёзной разработке, особенно в случае совместной работы над одним проектом нескольких аналитиков. Основная часть материала является пересказом [видеоурока](#) по работе с Git [148].

Система [Git](#) [28] — это одна из систем контроля версий. Система контроля версий — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии. Как правило подразумевается контроль версий файлов, содержащих исходный код программного обеспечения, хотя возможен контроль версий практически любых типов файлов [5]. Такие системы позволяют не только хранить версии файлов, но и содержат всю историю их изменения, позволяя отслеживать пошаговое изменение каждого бита файла.

1049 Это бывает особенно полезно в тех случаях, когда необходимо иметь возможность  
1050 «откатить» изменения в случае наличия в них ошибок либо тогда, когда над одним  
1051 и тем же проектом работает несколько разработчиков либо их команд. Конечно же  
1052 можно просто создавать полные копии всех файлов проекта. Однако данный спо-  
1053 соб полезен лишь для создания бэкапов на случай каких-то аварийных ситуаций.  
1054 В обычной работе он, как минимум, неудобен, а, как максимум, просто не спосо-  
1055 бен обеспечить пошаговое отслеживание изменений файлов и тем более слияние  
1056 результатов нескольких команд, параллельно работающих над одними и теми же  
1057 файлами. Для решения данной проблемы были разработаны локальные системы  
1058 контроля версий, содержащие базу данных всех изменений в файлах, примерная  
1059 схема организации которых показана на рисунке 2.3.1.

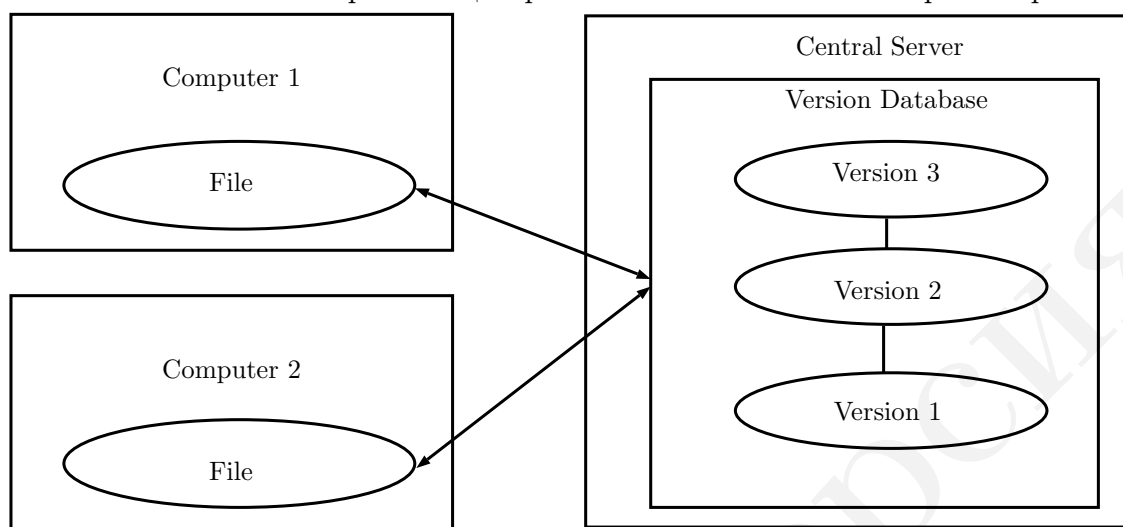
Рис. 2.3.1. Локальная система контроля версий



1060 Современные системы контроля версия бывают централизованными и распреде-  
1061 лёнными. Первые устроены таким образом, что вся история изменений файлов хра-  
1062 нится на центральном сервере, на который пользователи отправляют свои измене-  
1063 ния, и с которого они их получают. Общая схема работы централизованной системы  
1064 контроля версий приведена на рисунке 2.3.2 на следующей странице. Недостатком  
1065 такой системы являет её зависимость от работы центрального сервера. В случае  
1066 его остановки пользователи не смогут обрабатывать изменения, принимать и от-

1067 правлять их. Также существует риск полной потери всей истории в случае оконча-  
1068 тельного отказа сервера.

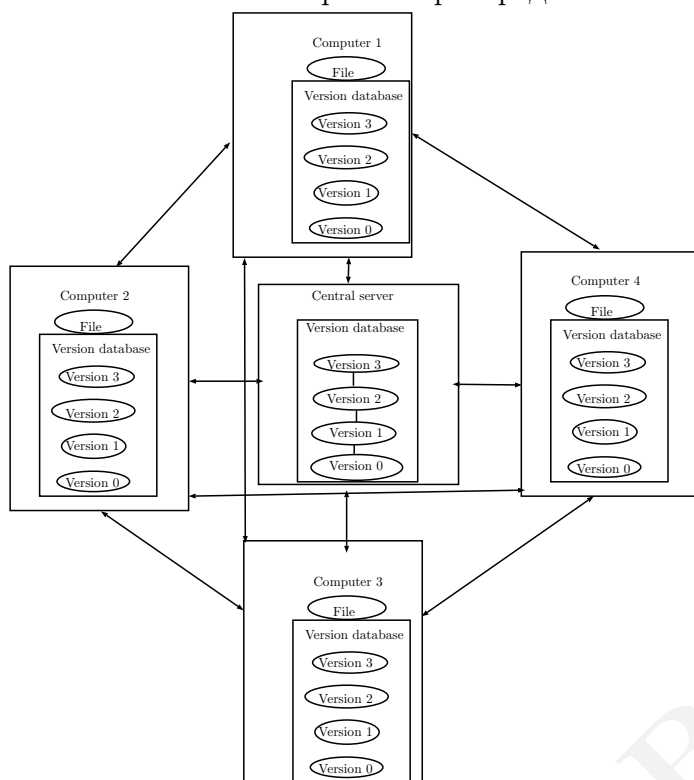
Рис. 2.3.2. Схема работы централизованной системы контроля версий



1069 Распределённые системы контроля версий лишены данного недостатка, поскольку у каждого пользователя хранится полная история изменений. В связи с этим  
1070 каждый пользователь может продолжать работать с системой контроля при от-  
1071 сутствии связи с сервером. После восстановления работоспособности последнего,  
1072 пользователь сможет синхронизировать свою историю изменений с другими разра-  
1073 ботчиками. Даже в случае полного отказа сервера команда сможет просто перевести  
1074 хранение на другой и продолжить работу в прежнем режиме. Общая схема работы  
1075 распределённой системы приведена на рисунке 2.3.3.

1077 Особенностью работы системы Git является заложенный в ней принцип работы.  
1078 В отличие от некоторых других систем контроля версий, принцип которых основан  
1079 на хранении исходного файла и списка изменений к нему, Git хранит состояние  
1080 каждого файла после его сохранения, создавая его «снимок». В терминологии Git  
1081 каждый такой снимок называется commit. При этом создаются ссылки на каждый  
1082 из файлов. В случае, если при создании нового commit Git обнаруживает, что какие-  
1083 то файлы не были изменены, система не включает сами файлы в новый commit,  
1084 а лишь указывает ссылку на последнее актуальное состояние файла из предыду-  
1085 щего commit, обеспечивая таким образом эффективность дискового пространства.  
1086 При этом каждый commit в целом ссылается на предыдущий, являющийся для него  
1087 родительским. На рисунке 2.3.4 на с. 38 показана общая схема работы системы Git.  
1088 Линиями со сплошным заполнением показана передача нового состояния файла, воз-  
1089 никшего в результате внесения в него изменений, прерывистым — передача ссылки  
1090 на состояние файла, не подвергавшегося изменениям, из прежнего commit. На мо-  
1091 мент времени 0 (initial commit) все файлы находились в состоянии 0. Затем в файлы  
1092 В и С были внесены изменения, тогда как файл А остался в прежнем состоянии.  
1093 В процессе создания commit № 1 Git сделал снимок состояния файлов В1 и С1, а так-

Рис. 2.3.3. Схема работы распределённой системы контроля версий



1094 же создал ссылку на состояние файла A0. Далее изменения были внесены в файл  
 1095 В. В процессе создания commit № 2 Git сохранил состояние файла B2, а также со-  
 1096 здал ссылки на состояния файлов A0 и C1 в предыдущем commit № 1. Затем были  
 1097 внесены изменения во все три файла, в результате чего на этапе создания commit  
 1098 № 3 Git сделал снимок состояний всех трёх файлов.

1099 Внимательный читатель скорее всего обратит внимание на третий тип линий  
 1100 — пунктир, которому соответствует подпись «hash». Чтобы понять, каким обра-  
 1101 зом в Git реализуется целостность версий, необходимо обратиться к понятию [хеш-](#)  
 1102 [функции](#) [10, 139].

### 1103 2.3.2. Хеш-функции

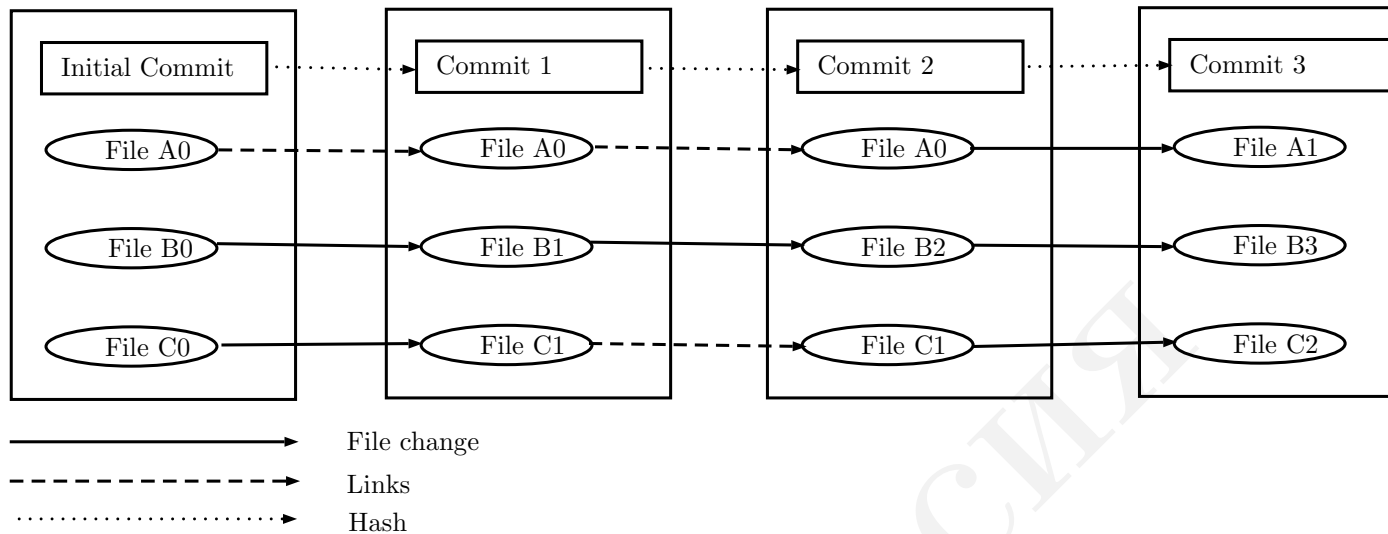
1104 Приведём основные определения.

1105 **Хеш функция (функция свёртки)** — функция, представляющая собой [детерми-](#)  
 1106 [нированный математический алгоритм](#) [118], осуществляющая преобразова-  
 1107 ние данных произвольной длины в результирующую битовую строку фикси-  
 1108 рованной длины.

1109 **Хеширование** — преобразование, осуществляемое хеш-функцией.

1110 **Сообщение (ключ, входной массив)** — исходные данные.

Рис. 2.3.4. Общая схема работы Git



1111 **Хеш (хеш-сумма, хеш-код, сводка сообщения)** — результат хеширования.

1112 Согласно [Принципу Дирихле](#) [130], между хешем и сообщением в общем отсутству-  
 1113 ет однозначное соответствие. При этом, число возможных значений хеша меньше  
 1114 числа возможных значений сообщения. Ситуация, при которой применение одной  
 1115 и той же хеш-функции к двум различным сообщениям приводит к одинаковому  
 1116 значению хеша, называется «[коллизией хеш функции](#)» [122]. Т.е. коллизия имеет  
 1117 место тогда, когда  $H(x) = H(y)$ .

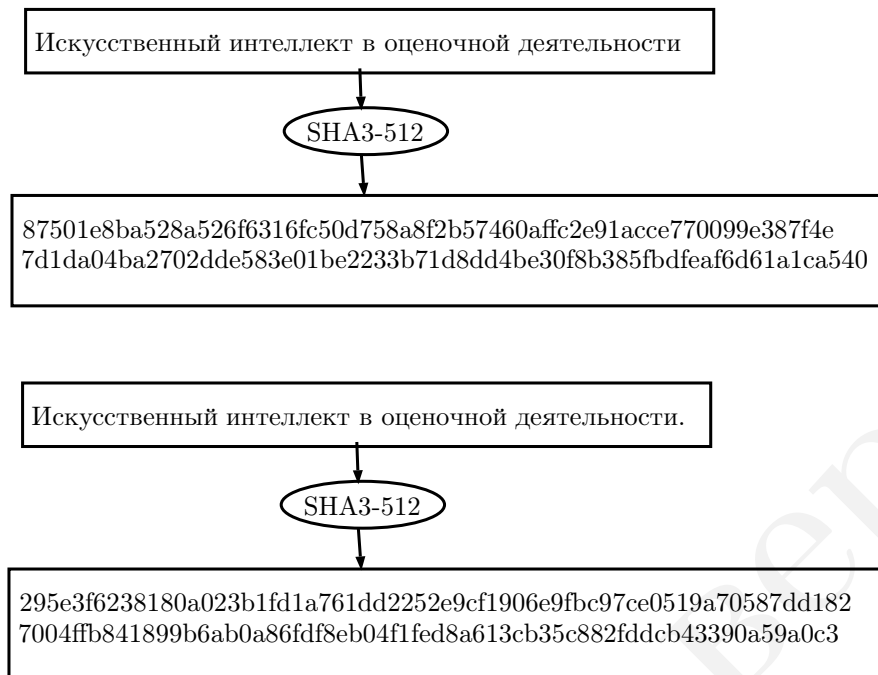
1118 Теоретическая «идеальная» хеш-функция отвечает следующим требованиям:

- 1119 а) является детерминированной, то есть её применение к одному и тому же со-  
 1120 общению приводит к одному и тому же значению хеша любое число раз;
- 1121 б) значение хеша быстро вычисляется для любого сообщения;
- 1122 в) зная значение хеша, невозможно определить значение сообщения;
- 1123 д) невозможно найти такие два разных сообщения, применение хеширования  
 1124 к которым приводило бы к одинаковому значению хеша (т.е. идеальная хеш-  
 1125 функция исключает возможность возникновения коллизии);
- 1126 е) любое изменение сообщения (вплоть до изменения значения одного бита) из-  
 1127 меняет хеш настолько сильно, что новое и старое значения выглядят никак  
 1128 не связанными друг с другом.

1129 Как правило, название хеш-функции содержит значение длины результирующей  
 1130 битовой строки. Например хеш-функция [SHA3-512](#) [111] возвращает строку длиной  
 1131 в 512 бит. Воспользуемся [одним](#) [79] из онлайн-сервисов вычисления хеша и посчи-  
 1132 таем его значение для названия данной книги. Как видно на рисунке [2.3.5 на сле-](#)  
 1133 [дующей странице](#), результатом вычисления хеш-функции является строка длиной

1134 в 512 бит, содержащая 128 шестнадцатеричных чисел. При этом, можно наблюдать,  
1135 что добавление точки в конце предложения полностью меняет значение хеша.

Рис. 2.3.5. Пример вычисления хеша



1136 Длина хеша в битах определяет максимальное количество сообщений, для кото-  
1137 рых может быть вычислен уникальный хеш. Расчёт осуществляется по формуле.

$$2^n \quad (2.3.1)$$

1138 , где  $n$  — длина строки в битах.

1139 Так, для функции SHA3-512 число сообщений, имеющих уникальный хеш состав-  
1140 ляет:  $2^{512} \sim 1.340781 \times 10^{154}$ . Таким образом, можно говорить о том, что современные  
1141 хеш-функции способны генерировать уникальный хеш для сообщений любой дли-  
1142 ны.

1143 Таким образом, Git в процессе создания нового commit сначала вычисляет его хеш-  
1144 сумму, а затем фиксирует состояние. При этом в каждом commit присутствует ссыл-  
1145 ка на предыдущий, также имеющий свою хеш-сумму. Таким образом, обеспечивается  
1146 целостность истории изменений, поскольку значение хеш-суммы каждого после-  
1147 дующего commit вычисляется на основе сообщения, содержащего в т. ч. свою хеш-  
1148 сумму. В этом случае любая модификация содержимого данных, образующих лю-  
1149 бой commit, неизбежно приведёт к изменению всех последующих хешей, что не оста-  
1150 нется незамеченным.



### 2.3.3. Начало работы с Git и основные команды

Для того, чтобы начать работать с Git прежде всего его конечно же следует установить. Как правило, с этим не возникает никаких сложностей. Однако всё же вопросы установки Git кратко рассмотрены в подразделе [2.4.1 Git 75–76](#).

В данном подразделе преимущественно рассматриваются аспекты работы с ним через командную строку. Данный выбор обусловлен тем обстоятельством, что существует множество графических интерфейсов для работы с Git, которые активно развиваются, меняют дизайн и расширяют функционал. Кроме того, появляются новые продукты. Среди такого разнообразия всегда можно выбрать какой-то наиболее близкий для себя вариант. Таким образом, автор не видит смысла останавливаться на разборе какого-то конкретного графического интерфейса. Более важной задачей является изложение сути и основных принципов работы, понимание которых обеспечит успешную работу с Git безотносительно конкретных программных средств. Кроме того, следует отметить, что практически все современные IDE [120] имеют свои средства и интерфейс для работы с Git. В дальнейшем в главах, посвящённых непосредственно применению R и Python, будут рассмотрены вопросы использования Git средствами RStudio, Spyder и PyCharm.

В данном подразделе описывается работа с Git через командную строку в операционной системе Kubuntu. Большая часть изложенного применима для любой операционной системы. Для начала работы с Git откроем терминал и выполним три основные настройки, а именно укажем:

- имя пользователя;
- адрес электронной почты;
- текстовый редактор по умолчанию.

Для конфигурации Git существует специальная утилита *git config*, имеющая три уровня глобальности настроек:

```
$ git config --system
```

— системный уровень: затрагивает все репозитории всех пользователей системы;

```
$ git config --global
```

— глобальный уровень: затрагивает все репозитории конкретного пользователя системы;

```
$ git config --local
```

— локальный уровень: затрагивает конкретный репозиторий;



Представим, что необходимо задать общие настройки конкретного пользователя, т.е. использовать уровень `global`, что, может быть актуально, например, при использовании рабочего компьютера. Сделаем следующие настройки:

```
$ git config --global user.name "First.Second"
$ git config --global user.email user-adress@host.com
$ git config --global core.editor "kate"
```

— мы задали имя пользователя, адрес его электронной почты, отображаемые при выполнении `commit`, а также указали текстовый редактор по умолчанию. В данном случае был указан редактор `Kate`. Естественно можно указать любой другой удобный редактор. В случае использования операционной системы `Windows` необходимо указывать полный путь до исполняемого файла (имеет расширение `.exe`) текстового редактора, а также `a`. Например, в случае использования 64-х разрядной `Windows` и редактора `Notepad++` [69] команда может выглядеть так:

```
$ git config --global core.editor "'C:\Program Files\Notepad
\notepad.exe' -multiInst -notabbar -nosession -noPlugin"
```

— перечень команд для различных операционных систем и текстовых редакторов содержится на [соответствующей странице](#) сайта `Git` [28].

Для начала создадим тестовый каталог, с которым и будем работать в дальнейшем при обучении работе с `Git`. Зайдём в папку, в которой хотим создать каталог и запустим терминал в ней. После чего введём команду:

```
$ mkdir git-lesson
```

— мы только что создали новый каталог средствами командной строки. Затем введём команду:

```
$ cd git-lesson
```

— переходим в только что созданный каталог.

Для просмотра содержимого каталога используем следующую команду:

```
$ ls -la
```

— собственно самой командой является `ls`, а «`-la`» представляет собой её аргументы: «`-l`» — отвечает за отображение файлов и подкаталогов списком, а «`-a`» — за отображение скрытых файлов и подкаталогов.

Для создания репозитория введём команду:

```
$ git init
```

— `Git` ассоциирует текущую папку с новым репозиторием.

В случае, если всё прошло хорошо, терминал возвратит следующее сообщение:

```
> Initialized empty Git repository in /home/.../git-lesson/.
git/
```

Теперь ещё раз введём:

```
$ ls -la
```

— следует обратить внимание на то, что появилась папка `.git`, в которой и будет храниться вся история версий проекта, содержащегося в папке `git-lesson`.

Создадим первый файл внутри папки:

```
$ touch file1.py
```

— расширение указывает на то, что это файл языка Python.

Система Git уже должна была отследить наличие изменения состояния проекта, произошедшее вследствие создания нового файла. Для проверки изменений состояния используем команду:

```
$ git log
```

— и получим сообщение следующего содержания:

```
> fatal: your current branch 'master' does not have any
commits yet
```

— дело в том, что в истории изменений по-прежнему нет никаких записей. Для получения дополнительных сведений используем команду:

```
$ git status
```

— терминал возвратит следующее сообщение:

```
> On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be
   committed)
   file1.py

nothing added to commit but untracked files present (use "
git add" to track)
```

— как видно, Git сообщает о том, что файл `file1.py` не отслеживается, кроме того, как следует из последней части сообщения терминала, в настоящее время вообще не фиксируются никакие изменения, поскольку ничего не было добавлено в лист отслеживания. При этом сам Git предлагает использовать команду `git add` для добавления файлов в него. Прежде чем сделать это, необходимо разобраться в том, в каких состояниях, с точки зрения Git, могут в принципе находиться файлы.

Все файлы, находящиеся в рабочем каталоге, могут иметь один из следующих статусов:

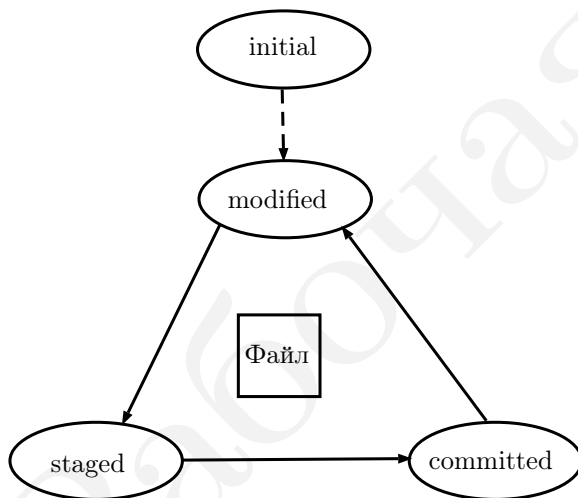
- tracked — отслеживаемые, т. е. находящиеся под версионным контролем;
- untracked — не отслеживаемые, т. е. не находящиеся под версионным контролем.

Ко второй категории, как правило, относятся временные файлы, например логи, хранение которых в репозитории нецелесообразно. Файлы первой категории могут находиться в одной из следующих состояний:

- initial — начальное состояние файла, в котором он находился в момент включения его в лист отслеживания, т. е. сообщения ему статуса tracked.
- modified — состояние файла после внесения в него изменений и его сохранения;
- staged — промежуточное состояние файла, в котором он находится после передачи его состояния Git, но до формирования последним его снимка.
- committed — состояние файла, зафиксированное Git, и представляющее его версию, к которой впоследствии будет возможно вернуться.

Соответственно после внесения новых изменений файл, находящийся в состоянии committed, переходит в состояние modified, после чего возможен новый цикл преобразований его статуса. Схема изменений состояния файлов приведена на рисунке 2.3.6.

Рис. 2.3.6. Схема состояний файлов в системе Git



Для перевода файла из состояния modified в состояние staged следует использовать команду

```
$ git add <file.name1> <file.name2>
```

— данная процедура также называется добавлением файла в индекс. Индекс — область памяти, в которой находятся файлы, подготовленные для включения в commit.

Далее для выполнения процедуры commit даётся команда

```
1315  
1316 $ git commit -m "message"  
1317
```

1318 — аргумент `-m` и следующее за ним сообщение служат для задания краткого опи-  
1319 сания того, какие изменения были внесены. Рекомендуется давать содержательные  
1320 комментарии, позволяющие понять смысл изменений.

1321 Как видно, не обязательно совершать процедуру `commit` сразу в отношении всех  
1322 файлов, находящихся в состоянии `modified`. Существует возможность группировать  
1323 их и, посредством перевода конкретных файлов в состояние `staged`, формировать  
1324 группы файлов, чьё состояние подлежит фиксации.

1325 Добавим файл `file.py` в индекс.

```
1326  
1327 $ git add file1.py  
1328
```

1329 Далее снова проверим статус:

```
1330  
1331 $ git status  
1332
```

1333 — на этот раз терминал возвратит новое сообщение:

```
1334  
1335 > On branch master  
1336  
1337 No commits yet  
1338  
1339 Changes to be committed:  
1340   (use "git rm --cached <file>..." to unstage)  
1341       new file:   file1.py  
1342
```

1343 Как можно видеть, теперь Git «видит» файл `file1.py` и готов сделать «снимок» но-  
1344 вого состояния репозитория. Для выполнения процедуры `commit` введём команду:

```
1345  
1346 $ git commit -m "First commit"  
1347
```

1348 — мы только что сделали первый `commit`, т.е. зафиксировали состояние репозито-  
1349 рия. Терминал возвратит следующее сообщение:

```
1350  
1351 > [master (root-commit) 1306b16] First commit  
1352   1 file changed, 0 insertions(+), 0 deletions(-)  
1353   create mode 100644 file1.py  
1354
```

1355 Теперь повторим ранее уже использованную команду:

```
1356  
1357 $ git log  
1358
```

1359 — терминал в отличие от первого раза, когда мы наблюдали сообщение о невоз-  
1360 можности вывода сведений о событиях в репозитории, на этот раз возвращает  
1361 осмысленное сообщение:

```
1362  
1363 > commit 1306b16f5fe40ccf8b141d716d9313df8e1983a1 (HEAD ->  
1364   master) Author: Kirill Murashev <kirill.murashev@gmail.  
1365   com>
```

```
Date:    Tue Aug 31 19:03:49 2021 +0200
First commit
```

— можно увидеть хеш-сумму данного commit, его автора, а также время создания commit и сопроводительное сообщение к нему. Для получения более детальных сведений можно использовать команду `git show`, сообщив ей в качестве аргумента хеш-сумму интересующего commit. Сделаем это, скопировав и вставив значение хеш-суммы:<sup>4</sup>

```
$ git show 1306b16f5fe40ccf8b141d716d9313df8e1983a1
```

— в качестве аргумента команды в данном случае была использована хеш-сумма. Терминал возвратит сообщение с данными об интересующем commit:

```
> commit 1306b16f5fe40ccf8b141d716d9313df8e1983a1 (HEAD ->
    master)
Author: Kirill Murashev <kirill.murashev@gmail.com>
Date:    Tue Aug 31 19:03:49 2021 +0300

    First commit

diff --git a/file1.py b/file1.py
new file mode 100644
index 0000000..e69de29
```

В дополнение к уже имеющимся данным приводятся сведения о том, какие именно изменения имели место. В данном случае видно, что имело место добавление в репозиторий нового файла.

Примерно такие же сведения можно получить в случае использования команды `git log` с аргументом `-p`.

```
$ git log -p

> commit 1306b16f5fe40ccf8b141d716d9313df8e1983a1 (HEAD ->
    master) Author: Kirill Murashev <kirill.murashev@gmail.
    com> Date:    Tue Aug 31 19:03:49 2021 +0300

    First commit

diff --git a/file1.py b/file1.py
new file mode 100644
index 0000000..e69de29
```

— в данном случае сообщения вообще идентичны.

<sup>4</sup>Для копирования и вставки в окне терминала следует использовать сочетания клавиш `ctrl+shift+c`, `ctrl+shift+v` соответственно.

Рассмотрим ещё одну полезную команду `git restore`. Данная команда возвращает состояние файла к тому состоянию, которое было зафиксировано при создании последнего commit. Рассмотрим пример. Откроем файл `file1.py` в редакторе Kate<sup>5</sup> непосредственно из терминала:

```
$ kate file.py
```

— далее напишем в нём любой текст и сохраним файл. После чего проверим его статус с помощью уже известной команды `git status`:

```
$ git status

> On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed
   )
  (use "git restore <file>..." to discard changes in working
   directory)
       modified:   file1.py

no changes added to commit (use "git add" and/or "git commit
-a")
```

— как видим, Git обнаружил изменение файла. Теперь введём команду:

```
$ git restore file.py
```

— файл, возвращён в состояние, в котором он находился на момент создания последнего commit, т. е. снова является пустым, в чём легко убедиться, открыв его.

Следующей рассматриваемой командой будет `git diff`. Данная команда позволяет понять, какие именно изменения были внесены в файл. Вновь откроем файл `file1.py` в текстовом редакторе. Введём в него текст, например «Liberte, egalite, fraternite». После чего сохраним файл. Выполним команду `git diff` и посмотрим на результат.

```
$git diff

> diff --git a/file1.py b/file1.py
index e69de29..72d6a2a 100644
--- a/file1.py
+++ b/file1.py
@@ -0,0 +1 @@
+Liberte, egalite, fraternite
```

— в нижней части сообщения терминала после символа «+» мы видим добавленный в файл текст. Git всегда отображает добавленный текст после знака «+», а удалённый после знака «-». Проверим статус файла:

<sup>5</sup>Естественно редактор может быть любой

```

1457 $ git status
1458
1459
1460 > On branch master
1461 Changes not staged for commit:
1462   (use "git add <file>..." to update what will be committed)
1463   (use "git restore <file>..." to discard changes in working
1464     directory)
1465         modified:   file1.py
1466
1467 no changes added to commit (use "git add" and/or "git commit
1468   -a")
1469

```

1470 — Git зафиксировал изменения файла. Теперь добавим файл в индекс, т. е. изменим его состояние на staged:

```

1471
1472 $ git add file1.py
1473
1474

```

1475 — далее ещё раз проверим статус файла:

```

1476 $ git status
1477
1478
1479 > On branch master
1480 Changes to be committed:
1481   (use "git restore --staged <file>..." to unstage)
1482         modified:   file1.py
1483

```

1484 — Git перевёл файл в состояние staged. Для того, чтобы ещё раз посмотреть изменения в файле, находящемся в состоянии staged можно использовать ту же команду `git diff`, при условии сообщения ей аргумента `--staged`, без которого она не сможет отобразить изменения, поскольку они уже были включены в индекс.

```

1485
1486
1487 $ git diff --staged
1488
1489
1490
1491 > diff --git a/file1.py b/file1.py
1492 index e69de29..d77d790 100644
1493 --- a/file1.py
1494 +++ b/file1.py
1495 @@ -0,0 +1 @@
1496 +Liberte, egalite, fraternite
1497

```

1498 Выполним commit:

```

1499
1500 $ git commit -m "Second commit"
1501

```

1502 — терминал возвратит сообщение:

```

1503 > [master 700a993] Second commit
1504 1 file changed, 1 insertion(+)
1505
1506

```



1507 — посмотрим на историю изменений:

```
1508 $ git log
1509
1510
1511 > commit 700a993db7c5f682c33a087cb882728adc485198 (HEAD ->
1512     master)
1513 Author: Kirill Murashev <kirill.murashev@gmail.com>
1514 Date:    Tue Aug 31 20:51:06 2021 +0200
1515
1516     Second commit
1517
1518 commit 1306b16f5fe40ccf8b141d716d9313df8e1983a1
1519 Author: Kirill Murashev <kirill.murashev@gmail.com>
1520 Date:    Tue Aug 31 19:03:49 2021 +0200
1521
1522     First commit
1523
```

1524 — можно наблюдать сведения о двух выполненных commit.

1525 В случае использования той же команды с аргументом `-p` можно увидеть всю  
1526 историю конкретных изменений.

```
1527 $ git log -p
1528
1529 > commit 700a993db7c5f682c33a087cb882728adc485198 (HEAD ->
1530     master)
1531 Author: Kirill Murashev <kirill.murashev@gmail.com>
1532 Date:    Tue Aug 31 20:51:06 2021 +0300
1533
1534     Second commit
1535
1536 diff --git a/file1.py b/file1.py
1537 index e69de29..d77d790 100644
1538 --- a/file1.py
1539 +++ b/file1.py
1540 @@ -0,0 +1 @@
1541 +Liberte, egalite, fraternite
1542
1543 commit 1306b16f5fe40ccf8b141d716d9313df8e1983a1
1544 Author: Kirill Murashev <kirill.murashev@gmail.com>
1545 Date:    Tue Aug 31 19:03:49 2021 +0300
1546     First commit
1547 diff --git a/file1.py b/file1.py
1548 new file mode 100644
1549 index 0000000..e69de29
1550
```

1551 Существует упрощённый способ передачи Git сведений для совершения commit.  
1552 Вместо последовательного ввода команд `git add` с указанием перечня файлов и `git`

commit можно использовать единую команду `git commit` с аргументами `-am`. Вторым аргументом, как уже было сказано ранее, необходимо для формирования сообщения, сопровождающего commit. Первый же заменяет собой предварительное использование команды `git add`, указывая Git на необходимость включения в индекс всех отслеживаемых файлов, т. е. имеющих статус `tracked`. Внесём любые изменения в файл `file1.py`. Проверим наличие изменений:

```
$ git status

> On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  use "git restore <file>..." to discard changes in working
  directory
        modified:   file1.py

no changes added to commit (use "git add" and/or "git commit
-a")
```

— после чего выполним добавление в индекс и commit одной командой.

```
$ git commit -am "Third commit"
> [master fbff919] Third commit
1 file changed, 1 insertion(+)
```

— проверим историю:

```
$ git log -p

> commit fbff919fab14ab6d41c993d3b86253c41037e075 (HEAD ->
  master)
Author: Kirill Murashev <kirill.murashev@gmail.com>
Date:   Tue Aug 31 21:25:45 2021 +0300

    Third commit

diff --git a/file1.py b/file1.py
index d77d790..bf6409f 100644
--- a/file1.py
+++ b/file1.py @@ -1,2 @@
  Liberte, egalite, fraternite
+Жизнь, свобода, собственность

commit 700a993db7c5f682c33a087cb882728adc485198
Author: Kirill Murashev <kirill.murashev@gmail.com>
Date:   Tue Aug 31 20:51:06 2021 +0300
```

```

1599
1600     Second commit
1601
1602 diff --git a/file1.py b/file1.py
1603 index e69de29..d77d790 100644
1604 --- a/file1.py
1605 +++ b/file1.py
1606 @@ -0,0 +1 @@
1607 +Liberte, egalite, fraternite
1608
1609 commit 1306b16f5fe40ccf8b141d716d9313df8e1983a1
1610 Author: Kirill Murashev <kirill.murashev@gmail.com>
1611 Date: Tue Aug 31 19:03:49 2021 +0300
1612
1613     First commit
1614
1615 diff --git a/file1.py b/file1.py
1616 new file mode 100644
1617 index 0000000..e69de29
1618

```

1619 — можно наблюдать уже три commit.

1620 Следующей полезной командой является `git mv`. Данная команда позволяет, в част-  
 1621 ности, переименовывать либо перемещать файлы. При этом её выполнение автома-  
 1622 тически переводит файл в состояние `staged`, минуя состояние `modified`. Выполним  
 1623 переименование:

```

1624 $ git mv file1.py file-1.py
1625
1626

```

1627 — затем проверим состояние:

```

1628 $ git status
1629
1630
1631 > On branch master
1632 Changes to be committed:
1633   (use "git restore --staged <file>..." to unstage)
1634   renamed:    file1.py -> file-1.py
1635

```

1636 — как можно увидеть, файл с новым именем готов к `commit`. Выполним `commit`.

```

1637 $ git commit -m "Fourth commit"
1638
1639
1640 > [master 284073c] Fourth commit
1641 1 file changed, 0 insertions(+), 0 deletions(-)
1642 rename file1.py => file-1.py (100%)
1643

```

1644 — изменения файла зафиксированы.

Следующей заслуживающей внимания командой является `git rm`. Данная команда удаляет файл.

```
$ git rm file-1.py
```

— проверим выполнение операции:

```
$ git status

> On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    file-1.py
```

— как видно из сообщения Git в терминале, существует возможность восстановить удалённый файл в том состоянии, которое было зафиксировано при выполнении последнего commit. Выполним команду для восстановления файла:

```
$ git restore --staged file-1.py
```

— затем проверим его состояние:

```
$ git status

> On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be
   committed)
  (use "git restore <file>..." to discard changes in working
   directory)
        deleted:    file-1.py

no changes added to commit (use "git add" and/or "git commit
-a")
```

— как следует из сообщения Git, файл `file-1.py` больше не находится в индексе, для его возвращения туда необходимо выполнить команду `git restore` без указания каких-либо аргументов.

```
$ git restore file-1.py
```

— ещё раз проверим состояние:

```
$ git status

> On branch master nothing to commit, working tree clean
```

— файл снова включён в индекс, его состояние соответствует состоянию, зафиксированному при выполнении последнего commit. Сам файл при этом вновь присутствует в каталоге.

1695 Команда `git rm` также может быть использована для передачи файлу статуса  
 1696 `untracked` без его удаления из каталога. Для этого ей необходимо сообщить аргумент  
 1697 `--cached`.

```
1698 $ git rm --cached file-1.py
1699
1700
1701 >rm 'file-1.py'
```

1703 — файл был исключён из индекса, а также из списка отслеживания, но при этом  
 1704 остался в каталоге, в чём можно легко убедиться:

```
1705 $ git status
1706 > On branch master
1707 Changes to be committed:
1708   (use "git_restore --staged <file>..." to unstage)
1709     deleted:    file-1.py
1710
1711 Untracked files:
1712   (use "git_add <file>..." to include in what will be
1713     committed)
1714     file-1.py
```

1717 — есть изменения, доступные для `commit`, а также в каталоге присутствует неот-  
 1718 слеживаемый файл (статус `untracked`).

```
1719 $ ls -la
1720 > total 0\
1721 drwx----- 1 user.name root  0 jaan  1 1970 .
1722 drwx----- 1 user.name root  0 jaan  1 1970 ..
1723 -rwx----- 1 user.name root 84 sept  1 19:08 file-1.py
1724 drwx----- 1 user.name root  0 jaan  1 1970 .git
```

1727 — файл присутствует в каталоге.  
 1728 Выполним `commit`:

```
1729 $ git commit -m "Fifth commit"
1730 > [master 7abee55] Fifth commit
1731 1 file changed, 2 deletions(-)
1732 delete mode 100644 file-1.py
```

1735 — далее посмотрим историю изменений:

```
1736 $ git log
1737
1738
1739 > commit 7abee55d2631cf7cf2e94e58f30f36b2be807948 (HEAD ->
1740   master)
1741 Author: Kirill Murashev <kirill.murashev@gmail.com>
1742 Date:    Wed Sep 1 19:52:04 2021 +0300
```

```

1743     Fifth commit
1744
1745 commit 284073c521af8b73e16f324698f24040e4b9ee7e
1746 Author: Kirill Murashev <kirill.murashev@gmail.com>
1747 Date:   Wed Sep 1 18:16:46 2021 +0300
1748
1749     Fourth commit
1750
1751 commit fbff919fab14ab6d41c993d3b86253c41037e075
1752 Author: Kirill Murashev <kirill.murashev@gmail.com>
1753 Date:   Tue Aug 31 21:25:45 2021 +0300
1754
1755     Third commit
1756
1757 commit 700a993db7c5f682c33a087cb882728adc485198
1758 Author: Kirill Murashev <kirill.murashev@gmail.com>
1759 Date:   Tue Aug 31 20:51:06 2021 +0300
1760
1761     Second commit
1762
1763 commit 1306b16f5fe40ccf8b141d716d9313df8e1983a1
1764 Author: Kirill Murashev <kirill.murashev@gmail.com>
1765 Date:   Tue Aug 31 19:03:49 2021 +0300
1766
1767     First commit
1768

```

1769 — проверим наличие файла в каталоге:

```

1770 $ ls -la
1771 > total 0
1772
1773 drwx----- 1 user.name root 0 jaan 1 1970 .
1774 drwx----- 1 user.name root 0 jaan 1 1970 ..
1775 -rwx----- 1 user.name root 84 sept 1 19:08 file-1.py
1776 drwx----- 1 user.name root 0 jaan 1 1970 .git
1777

```

1778 — а также его статус:

```

1779 $ git status
1780 > On branch master
1781 Untracked files:
1782   (use "git add <file>..." to include in what will be
1783     committed)
1784       file-1.py
1785
1786
1787 nothing added to commit but untracked files present (use "
1788   git add" to track)

```

— файл присутствует в каталоге и имеет статус `untracked`.  
Вернём файл в индекс.

```
$ git add file-1.py
```

— файл вновь имеет статус `tracked`.

#### 2.3.4. Исключение файлов из списка отслеживания

В процессе разработки нередко возникают файлы, отслеживание которых скорее всего является нецелесообразным, например файлы, содержащие логи. При этом их постоянное присутствие в списке файлов, имеющих статус `untracked`, осложняет работы и также является нежелательным. В связи с этим существует механизм исключения ряда файлов или подкаталогов из под всей системы версионирования, называемый *gitignore*.

Выполним ряд процедур. До этого все действия выполнялись путём последовательного ввода команд. В данном случае будет показано, как можно использовать заготовленные скрипты. Использование скриптов является очень удобным тогда, когда существует необходимость многократного ввода длинной последовательности команд. В рассматриваемом примере будет рассмотрена последовательность всего из пяти команд. Для создания скрипта необходимо написать его текст в текстовом редакторе, сохранить файл с расширением `txt` (например `script1.txt`), после чего запустить терминал в каталоге с файлом и указать системе на то, что данный файл является исполняемым, т. е. передать ему права `execute`. Напишем скрипт:

```
# создаём подкаталог
mkdir log
# переходим в новый подкаталог
cd log/
# создаём файл
touch log.txt
# возвращаемся в каталог верхнего уровня
cd ..
# проверяем статус
git status
```

— смысл того, что выполняет команда раскрыт в комментарии, предшествующем ей. Следует обратить внимание на то, что команды, передаваемые терминалу пишутся на языке [Bash \[92\]](#), в котором игнорируется всё, что написано в строке после символа «`#`». Передадим файлу права `execute` путём ввода команд в терминала, запущенном из каталога, содержащего файл. Можно использовать любую (двоичную либо символическую) запись:

```
$ chmod u+x script
```



1833 — либо:

```
1834 $ chmod 744 script
```

1837 — для проверки наличия прав в системе Kubuntu и многих других можно использовать команду:

```
1839 $ ls -l script1
```

1842 — в случае наличия прав execute терминал возвратит ответ, содержащий имя файла, выделенное **зелёным** цветом.

1844 Теперь следует вернуться в окно терминала, запущенное в каталоге изучаемого репозитория после чего просто ввести нём полный путь до созданного скрипта:

```
1846 $ ~/.../Scripts/script1
```

1849 — в случае правильных действий терминал возвратит сообщение:

```
1850 > On branch maste
1851 r Changes to be committed:
1852   (use "git restore --staged <file>..." to unstage)
1853       new file:   file-1.py
1854
1855 Untracked files:
1856   (use "git add <file>..." to include in what will be
1857       committed)
1858       log/
1859
1860
```

1861 В данном случае автор использовал подготовленный bash скрипт. Аналогичного результата можно добиться путём простого последовательного ввода команд. Подробнее о запуске скриптов в операционных системах, основанных на ядре Linux, можно прочитать, например [здесь](#) [145]. Возвращаясь к теме Git, отметим, что в каталоге появилась неотслеживаемая папка log. Создадим файл с именем .gitignore:

```
1866 $ kate .gitignore
```

1869 — при этом сразу же откроется окно текстового редактора. Следует сделать небольшое отступление и сказать о том, что состав файлов и папок, подлежащих исключению из списка, подлежащего версионированию, в существенной степени зависит от используемого языка программирования. В дальнейшем будут рассмотрены вопросы автоматизации создания файла .gitignore. Сейчас же кратко рассмотрим подготовленные файлы для языков Python и R. Ниже приводится примерное содержание файла .gitignore, предназначенного для репозитория, содержащего код на языке Python:

```
1877 # Byte-compiled / optimized / DLL files
1878 __pycache__ /
1879 *.py[cod]
1880 *$py.class
```

```
1882
1883 # C extensions
1884 *.so
1885
1886 # Distribution / packaging
1887 .Python
1888 build/
1889 develop-eggs/
1890 dist/
1891 downloads/
1892 eggs/
1893 .eggs/
1894 lib/
1895 lib64/
1896 parts/
1897 sdist/
1898 var/
1899 wheels/
1900 share/
1901 python-wheels/
1902 *.egg-info/
1903 .installed.cfg
1904 *.egg MANIFEST
1905
1906 # PyInstaller
1907 # Usually these files are written by a python script from a
1908     template
1909 # before PyInstaller builds the exe, so as to inject date/
1910     other infos into it.
1911 *.manifest
1912 *.spec
1913
1914 # Installer logs
1915 pip-log.txt
1916 pip-delete-this-directory.txt
1917
1918 # Unit test / coverage reports
1919 htmlcov/
1920 .tox/
1921 .nox/
1922 .coverage
1923 .coverage.*
1924 .cache nosetests.xml
1925 coverage.xml
```

```
147 *.cover
148 *.py,cover
149 .hypothesis/
150 .pytest_cache/
151 cover/
152
153 # Translations
154 *.mo
155 *.pot
156
157 # Django stuff:
158 *.log
159 local_settings.py
160 db.sqlite3
161 db.sqlite3-journal
162
163 # Flask stuff:
164 instance/
165 .webassets-cache
166
167 # Scrappy stuff:
168 .scrappy
169
170 # Sphinx documentation
171 docs/_build/
172
173 # PyBuilder
174 .pybuilder/
175 target/
176
177 # Jupyter Notebook
178 .ipynb_checkpoints
179
180 # IPython
181 profile_default/
182 ipython_config.py
183
184 # pyenv
185 # For a library or package, you might want to ignore these
186 # files since the code is
187 # intended to run in multiple environments; otherwise,
188 # check them in:
189 # .python-version
190 # pipenv
```

```
1989 # According to pypa/pipenv#598, it is recommended to
1971 include Pipfile.lock in version control.
1972 # However, in case of collaboration, if having platform-
1973 specific dependencies or dependencies
1974 # having no cross-platform support, pipenv may install
1975 dependencies that don't work, or not
1976 # install all needed dependencies.
1977 #Pipfile.lock
1978 # PEP 582; used by e.g. github.com/David-OConnor/pyflow
1979 __pypackages__/_
1980
1981 # Celery stuff
1982 celerybeat-schedule
1983 celerybeat.pid
1984
1985 # SageMath parsed files
1986 *.sage.py
1987
1988 # Environments
1989 .env
1990 .venv
1991 env/
1992 venv/
1993 ENV/
1994 env.bak/
1995 venv.bak/
1996
1997 # Spyder project settings
1998 .spyderproject
1999 .spyproject
2000
2001 # Rope project settings
2002 .ropeproject
2003
2004 # mkdocs documentation
2005 /site
2006
2007 # mypy
2008 .mypy_cache/
2009 .dmypy.json dmypy.json
2010
2011 # Pyre type checker
2012 .pyre/
2013
```

```

1310 # pytype static type analyzer
1311 .pytype/
1312
1313 # Cython debug symbols
1314 cython_debug/
1315
1316 # учебная строка, добавлена автором
1317 log/

```

— можно сказать, что файл содержит в себе в т. ч. набор относительно простых регулярных выражений. В частности символ «\*» означает возможность наличия любых символов. Заключение последовательности символов в квадратные скобки означает возможность присутствия на данном месте любого из них. В частности в строке 3 содержится указание на необходимость игнорирования файлов, имеющих любое имя и одно из следующих расширений: .рус, .руо, .руд.

Примерное содержание файла .gitignore, предназначенного для репозитория, содержащего код на языке R:

```

2032 # History files
2033 .Rhistory
2034 .Rapp.history
2035
2036 # Session Data files
2037 .RData
2038
2039 # User-specific files
2040 .Ruserdata
2041
2042 # Example code in package build process
2043 *-Ex.R
2044
2045 # Output files from R CMD build
2046 /*.tar.gz
2047
2048 # Output files from R CMD check
2049 /*.Rcheck/
2050
2051 # RStudio files
2052 .Rproj.user/
2053
2054 # produced vignettes
2055 vignettes/*.html
2056 vignettes/*.pdf
2057
2058 # OAuth2 token, see https://github.com/hadley/httr/releases/

```

2059	<code>tag/v0.3</code>	
2060	<code>.httr-oauth</code>	28
2061		29
2062	<code># knitr and R markdown default cache directories</code>	30
2063	<code>*_cache/</code>	31
2064	<code>/cache/</code>	32
2065		33
2066	<code># Temporary files created by R markdown</code>	34
2067	<code>*.utf8.md</code>	35
2068	<code>*.knit.md</code>	36
2069		37
2070	<code># R Environment Variables</code>	38
2071	<code>.Renviron</code>	39
2072		40
2073	<code># pkgdown</code>	41
2074	<code>site docs/</code>	42
2075		43
2076	<code># translation temp files</code>	44
2077	<code>po/*~</code>	45
2078		46
2079	<code># учебная строка, добавлена автором</code>	47
2080	<code>log/</code>	48
2081		

2082 — используем любой из указанных файлов, сохраним его и проверим статус:

```

2083 $ git status
2084 > On branch master
2085 Changes to be committed:
2086   (use "git restore --staged <file>..." to unstage)
2087     new file:   file-1.py
2088
2089 Untracked files:
2090   (use "git add <file>..." to include in what will be
2091     committed)
2092     .gitignore
2093
2094 
```

2095 — как видим папка log пропала и появился файл .gitignore. Добавим его в индекс:

```

2096 $ git add .gitignore
2097
2098 
```

2099 — а затем выполним commit:

```

2100 $ git commit -m "Sixth commit"
2101 > [master e4adf82] Sixth commit
2102 2 files changed, 142 insertions(+)
2103 create mode 100644 .gitignore
2104 create mode 100644 file-1.py
2105
2106 
```

— теперь в случае создания в каталоге любого файла, чьё имя подпадает под правила, описанные в файле `.gitignore`, он сразу же исключается из списка наблюдения со стороны системы версионирования. Забегая вперёд, можно сказать, что, чаще всего отсутствует необходимость создавать такой файл вручную. Данная функция реализована во многих IDE и будет рассмотрена далее.

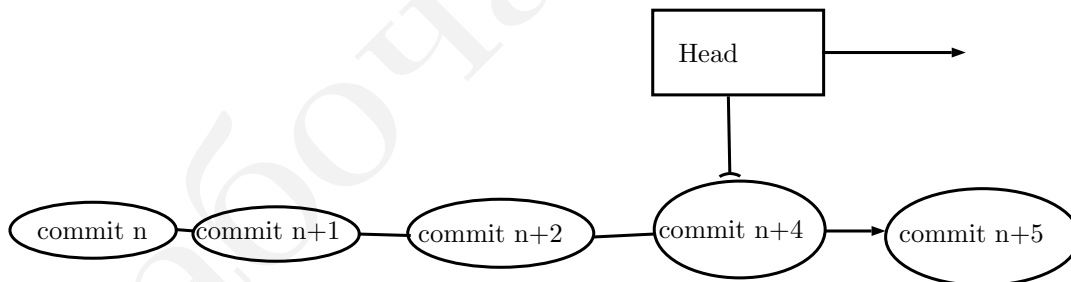
### 2.3.5. Ветки проекта, указатели `branch` и `Head`

В предыдущих подразделах рассматривалась линейная модель созданий версий, которые последовательно формировались одна за другой путём проведения процедуры `commit`. Git позволяет осуществлять ветвление версий. Посмотрим на текущий статус репозитория:

```
$ git status
> On branch master nothing to commit, working tree clean
```

— обратим внимание на сообщение, возвращённое терминалом, содержащее ссылку на некую `branch master`. Для того, чтобы разобраться в данном вопросе, следует вспомнить основные принципы работы Git, описанные в подразделах 2.3.1–2.3.2 на с. 34–39. Каждый `commit` имеет хеш-сумму, содержащую в т.ч. ссылку на предыдущий `commit`. Таким образом формируется неразрывная цепочка версий. Помимо этого в Git реализована работа указателя `Head`, представляющего собой метку, указывающую на один из `commit`. Местонахождение этой метки указывает Git, в каком именно состоянии репозиторий находится в данный момент. При каждом выполнении `commit` указатель `Head` смещается на новый `commit`. Схема работы указателя `Head` показана на рисунке 2.3.7.

Рис. 2.3.7. Схема работы указателя `Head`



Установить текущее местонахождение указателя `Head` можно с помощью команды `git log`.

```
$ git log
> commit e4adf8280c5d95a6f5796dba8e028012565de958
(HEAD -> master)
Author: Kirill Murashev <kirill.murashev@gmail.com>
```



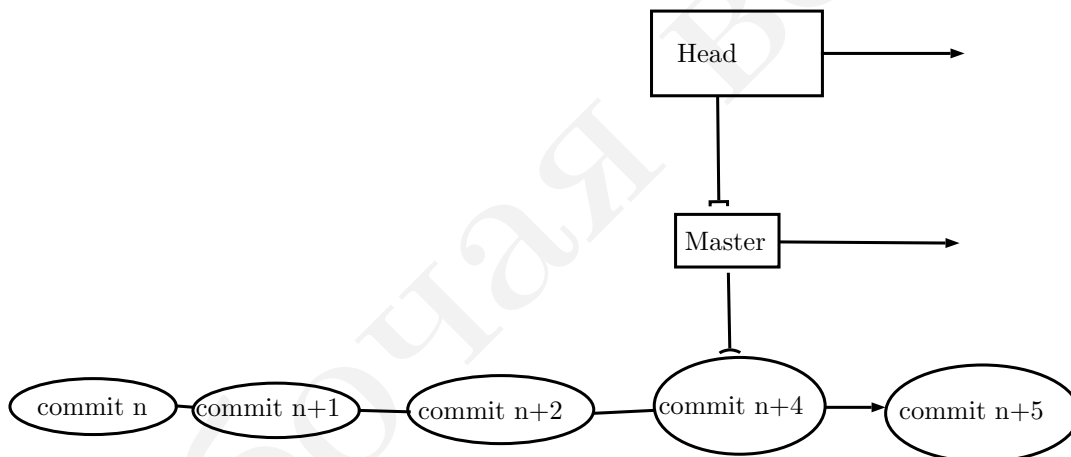
```

2139 Date:    Thu Sep 2 20:42:49 2021 +0300
2140
2141     Sixth commit
2142
2143 commit 7abee55d2631cf7cf2e94e58f30f36b2be807948 Author:
2144     Kirill Murashev <kirill.murashev@gmail.com>
2145 Date:    Wed Sep 1 19:52:04 2021 +0300
2146
2147     Fifth commit
2148
2149 commit 284073c521af8b73e16f324698f24040e4b9ee7e
2150
2151 :...skipping...
2152

```

2153 — как следует из ответа терминала, указатель Head находится на последнем шестом commit ветки Master. При этом ветка также представляет собой некий указатель. Таким образом, схема организации указателей выглядит так, как это показано на рисунке 2.3.8.

Рис. 2.3.8. Схема указателей Head и Branch



2157 При наличии достаточной степени развития проекта хорошей практикой считается хранение стабильной версии в ветке Master (в современных системах часто используется наименование Main).<sup>6</sup>

2160 При этом, для новых изменений, находящихся в стадии разработки и тестирования, рекомендуется использовать отдельную ветку. Для создания новой ветки следует использовать команду `git branch <name>`:

```

2163 $ git branch Develop
2164
2165

```

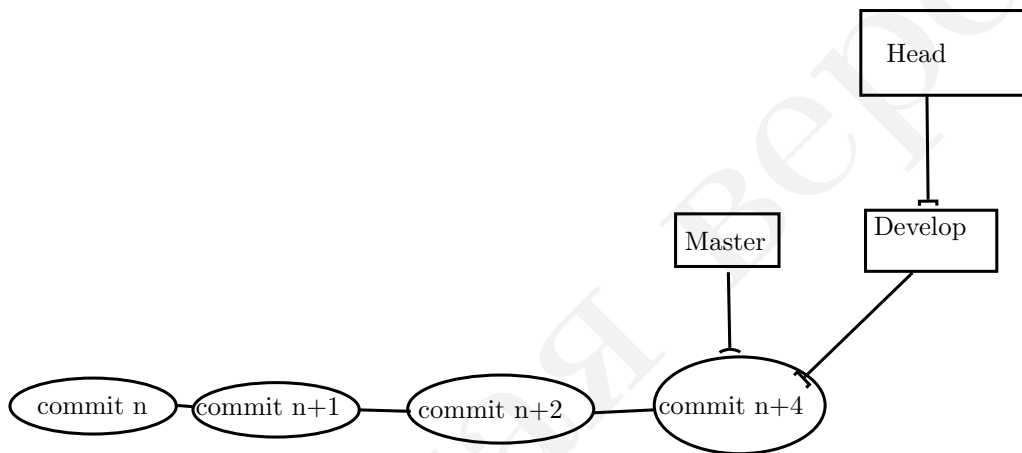
<sup>6</sup> Данное решение обусловлено политическими причинами, поскольку слово Master может ассоциироваться с рабовладением.

— была создана новая ветка Develop. Для перемещения указателя Head на неё используем команду:

```
$ git checkout Develop
```

— состояние репозитория выглядит следующим образом: см. рисунок 2.3.9. Теперь все последующие commit будут сопровождаться указателем ветки Develop, тогда как указатель Master останется на прежнем месте. В случае обратного перемещения указателя Head на ветку Master состояние файлов проекта вернётся к тому, каким оно было в момент создания commit, на который теперь указывает Head. При этом все изменения, сделанные в ветке Develop будут сохранены в ней и доступны в случае перемещения Head на них. После определённого количества перемещений и доработок проект может выглядеть, например так, как показано на рисунке 2.3.10 на следующей странице.

Рис. 2.3.9. Состояние репозитория после переноса указателя Head на ветку Develop



Предположим, что, достигнув состояния, показанного на рисунке на рисунке 2.3.10 на следующей странице, оценщик приходит к выводу о необходимости слияния всех веток в ветку master. Сначала можно посмотреть, какие ветки в принципе существуют.

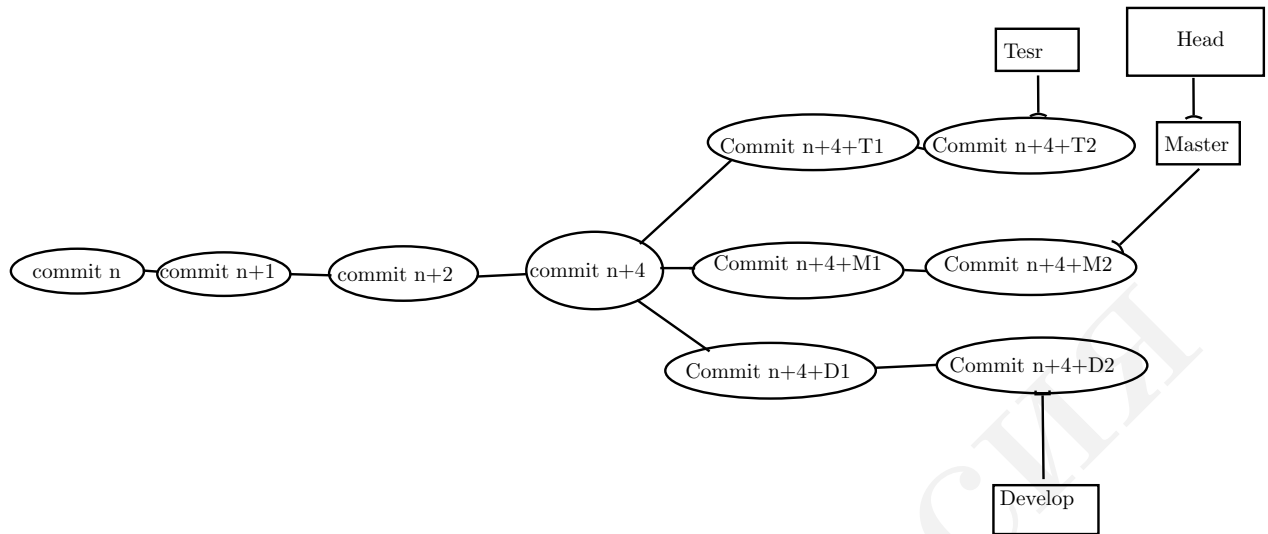
```
$ git branch -a
>
  develop
* master
  test
```

Как видно, указатель Head уже находится на целевой ветке. Если это не так, его следует туда перенести:

```
$ git checkout master
```

После этого выполняем команду `git merge`, в качестве аргумента которой используется имя ветки, которую предполагается объединить с master:

Рис. 2.3.10. Состояние репозитория при наличии нескольких веток



```
2198 $ git merge develop
2199
2200
```

2201 — в случае, когда последний commit из ветки master является прямым родителем  
 2202 для ветки develop объединение происходит путём простого перемещения указателя  
 2203 Head на master, а затем их совместного перемещения на последний commit ветки  
 2204 develop. Данный способ объединения называется fast forward. После такого объеди-  
 2205 нения может быть целесообразным удалить ветку develop, поскольку дальнейшая  
 2206 работа будет вестись в master. Для этого следует выполнить команду:

```
2207 $ git branch -d develop
2208
2209
```

2210 — теперь все изменения, ранее сделанные в ветке develop по-прежнему доступны  
 2211 уже в ветке master. В случае, когда объединяемые ветки не являются родитель-  
 2212 ской и дочерней относительно друг друга процедура объединения происходит более  
 2213 сложным образом. При этом используется та же команда. В результате её выпол-  
 2214 нения формируется новый commit, называемый merge commit. При этом проис-  
 2215 ходит перемещение указателей master и Head на данный commit. В случае, если  
 2216 commits, являющиеся родительскими по отношению к merge commit имеют толь-  
 2217 ко непересекающиеся дополнения относительно последнего общего родительского  
 2218 commit и не имеют взаимоисключающих правок, объединение происходит автома-  
 2219 тически и не требует внимания пользователя. Ситуация, при которой имеет место  
 2220 т. н. merge conflict, будет рассмотрена в подразделе 2.3.7 на с. 74–74. Последова-  
 2221 тельное выполнение команд git branch и git checkout можно заменить одним  
 2222 командой:

```
2223 $ git checkout -b <branch.name>
2224
2225
```

2226 — в случае необходимости отведения новой ветки не от того commit, на который

указывает Head вторым аргументом этой команды должна быть хеш-сумма того commit, от которого необходимо отвести ветку.

## 2.3.6. Работа с Github

### 2.3.6.1. Начало

В материале, изложенном выше в подразделах 2.3.3–2.3.5 2.3.3 на с. 40–65, речь шла о работе с локальным репозиторием, хранящимся на компьютере пользователя. При этом при командной работе часто требуется наличие общего доступа к рабочему каталогу. Также наличие удалённой версии репозитория позволяет распространять разработки на широкую аудиторию. Кроме того, наличие удалённого репозитория позволяет иметь дополнительный бэкап, не зависящий от физического устройства пользователя. Следуя **принципу KISS**, положенному в основу данной работы, в настоящем разделе будет рассмотрена работа с наиболее популярным сервисом удалённых репозиториях — **GitHub** [31]. Следует отметить, что существует значительное количество альтернатив, кроме того существует возможность хранения удалённого репозитория на собственном удалённом сервере.

Для начала работы с GitHub необходимо осуществить регистрацию, которая вряд ли может вызвать у кого затруднение в 2021 году. Для создания своего первого репозитория необходимо в меню профиля выбрать пункт Your Repositories и далее создать свой, что также вряд ли может вызвать затруднения. В появившемся меню следует ввести имя репозитория латинскими символами, затем выбрать тип репозитория: публичный либо приватный. В первом случае доступ к репозиторию (но его изменению его содержимого) будет о неограниченного круга пользователей. Для доступа к материалам достаточно иметь ссылку на репозиторий. Во втором случае доступ даже к просмотру будут иметь только те, кому будет предоставлены соответствующие права. Следующие пункты меню позволяют добавить файл README, содержащий основные сведения о проекте, файл .gitignore, сформированный по шаблону, разработанному для конкретного языка, а также выбрать лицензию, на условиях которой возможно легальное использование продукта.

Для обеспечения связи между локальным репозиторием и его удалённой версией необходимо зайти в него и выбрать меню **Code**. В данном меню можно выбрать одно из трёх средств передачи данных:

- протокол **HTTPS** [100];
- протокол **SSH** [112];
- средства командной строки GitHub CLI, в свою очередь также реализующие передачу данных посредством протоколов:
  - **HTTPS**;
  - **SSH**.

В общем случае рекомендуется использовать протокол SSH. С точки зрения начинающего пользователя различие заключается в том, что при использовании протокола HTTPS каждый раз для соединения с удалённым репозиторием потребуется ввод логина и пароля, тогда как в случае с SSH — нет. При этом для того, чтобы использовать SSH необходимо провести первоначальные настройки. На самом деле, протокол SSH является предпочтительным по ряду технических причин среди которых можно выделить более высокий уровень безопасности, а также эффективное сжатие данных. Для подробного ознакомления с преимуществами и недостатками различных протоколов рекомендуется ознакомиться со [следующим официальным материалом \[29\]](#).

### 2.3.6.2. Настройка соединения с удалённым репозиторием посредством протокола SSH

Для установления связи с удалённым репозиторием GitHub посредством протокола SSH необходимо осуществить ряд действий, а именно генерировать пару SSH-ключей, а затем добавить их в профиль аккаунта на портале GitHub.

**2.3.6.2.1. Проверка наличия существующих SSH-ключей.** Для начала необходимо проверить наличие существующих ключей. Для этого следует запустить Терминал и ввести команду:

```
$ ls -al ~/.ssh
```

— в случае наличия существующих ключей Терминал возвратит примерно следующее сообщение:

```
> total 20
drwx----- 2 user.name user.name 4096 aug 14 14:42 .
drwxr-xr-x 36 user.name user.name 4096 sept 1 09:14 ..
-rw----- 1 user.name user.name 464 aug 11 11:05
    id_ed25519
-rw-r--r-- 1 user.name user.name 107 aug 11 10:04
    id_ed25519.pub
-rw-r--r-- 1 user.name user.name 1326 aug 11 19:08
    known_hosts
```

— в этом случае можно пропустить второй этап, описанный в подсекции [2.3.6.2.2–68](#), и перейти к третьему этапу, описанному в подсекции [2.3.6.2.3 на с. 68–69](#). В случае отсутствия существующей пары необходимо осуществить её генерацию.

**2.3.6.2.2. Генерация новой пары ключей.** Для создания пары ключей на основе алгоритма RSA, необходимо запустить Терминал и выполнить команду:

```
$ ssh-keygen -t rsa -b 4096 -C "user.name@host.com"
```

— указав при этом тот адрес электронной почты, который указан в профиле на GitHub. Помимо адреса электронной почты аргументами команды являются: алгоритм генерации ключа и его длина в битах. Следует сказать, что алгоритм RSA не является единственным. О различиях между алгоритмами RSA [110], DSA [95], ECDSA [96] и Ed25519 [16] можно почитать в следующих статьях и комментариях к ним: [24, 80, 73]. В целом, можно сказать, что для целей обучения анализу данных, равно как и для большинства практических целей оценщиков нет существенной разницы в том, какой алгоритм будет использован при создании пары ключей. Однако, с точки зрения соответствия лучшим практикам и современным тенденциям можно сказать следующее:

- a) Алгоритм DSA несколько устарел и подвержен уязвимости, поскольку решение проблемы вычислительной сложности взятия логарифмов в конечных полях [119], на которой он и был основан, было найдено в 2013 году.
- b) Схожий с DSA алгоритм ECDSA лишён указанного недостатка, поскольку основан не на конечном числовом поле [123], а на группе точек эллиптической кривой [142]. При этом криптографическая стойкость алгоритма в существенной степени зависит от возможности компьютера генерировать случайные числа.
- c) Алгоритм RSA обеспечивает достаточную надёжность при условии достаточной длины ключа. Длина ключа в 3072 либо 4096 бит является достаточной. Данный алгоритм является рекомендуемым в том случае, если нет возможности использовать алгоритм Ed25519.
- d) Алгоритм Ed25519 является предпочтительным во всех случаях, когда система технически способна работать с ним. Данный алгоритм обеспечивает хорошую криптостойкость, при этом работа с ключами происходит существенно быстрее, чем при использовании алгоритма RSA. Длина публичного ключа составляет всего 68 символов, тогда как RSA генерирует публичный ключ длиной в 544 символа (при 3072 бит).

Таким образом, вместо вышеуказанной команды рекомендуется использовать команду:

```
$ ssh-keygen -t ed25519 -C "user.name@host.com"
```

— адрес электронной почты также должен совпадать с тем, который указан в профиле на портале GitHub. Терминал возвратит сообщение:

```
> Enter a file in which to save the key (/home/user.name/.ssh/id_ed25519): [Press enter]
```

— предложив нажать Enter для сохранения ключей в каталоге по умолчанию. Следует согласиться с предложением и перейти к этапу создания пароля:

```
2347 > Enter passphrase (empty for no passphrase): [Type a
2348 passphrase]
2349 > Enter same passphrase again: [Type passphrase again]
```

2352 — ключи SSH готовы. Для возможности работы с ними необходимо добавить их в ssh-agent. Для этого сначала необходимо запустить ssh-agent в фоновом режиме, выполнив последовательно две команды:

```
2355 $ sudo -s -H
2356 $ eval "$(ssh-agent -s)"
```

2359 — далее осуществляется добавление самого ключа:

```
2360 $ ssh-add ~/.ssh/id_ed25519
```

2363 — ключи зарегистрированы в ssh-agent и могут быть использованы для взаимодействия с порталом GitHub.

2365 **2.3.6.2.3. Добавление публичного ключа на портал GitHub.** Для того, чтобы добавить в профиль на портале GitHub публичный ssh ключ необходимо получить его значение. Для начала следует установить xclip:

```
2368 $ sudo apt-get update
2369 $ sudo apt-get install xclip
```

2372 — теперь существует возможность автоматически копировать возвращаемые терминалом сообщения в буфер обмена. Сделаем это:

```
2374 $ xclip -selection clipboard < ~/.ssh/id_ed25519.pub
```

2377 — в данный момент буфер обмена содержит значение публичного ssh ключа. После этого необходимо зайти в свой профиль на портале GitHub и найти пункт меню **Settings**, а затем **SSH and GPG keys**. После этого следует нажать на кнопку **New SSH key**. Откроется меню, состоящее из двух полей: заголовка и значение ключа. В поле заголовка можно ввести любые символы, например имя, фамилию и должность. В поле значения ключа необходимо вставить содержимое буфера обмена. Существует семь возможных начальных символов ключа:

- 2384 ● «ssh-rsa»;
- 2385 ● «ecdsa-sha2-nistp256»;
- 2386 ● «ecdsa-sha2-nistp384»;
- 2387 ● «ecdsa-sha2-nistp521»;
- 2388 ● «ssh-ed25519»;
- 2389 ● «sk-ecdsa-sha2-nistp256@openssh.com»;



- «sk-«ssh-ed25519@openssh.com»

— в зависимости от применённого алгоритма. В случае совпадения практического значения с одним из возможных, можно сделать вывод о том, что все подготовительные операции были выполнены корректно. Нет необходимости вглядываться в имеющееся на практике значение: система в любом случае не зарегистрирует ключ, не отвечающий требованиям по маске. В том случае, если ключ прошёл валидацию, кнопка Add SHH key, расположенная ниже поля, станет активной. После её нажатия произойдёт добавление ключа.

Перед началом использования связи по SSH протоколу рекомендуется провести проверку. Для этого в терминале следует ввести команду:

```
$ ssh -T git@github.com
```

— терминал запросит ввести пароль, установленный при генерации ключей. В случае установления успешной связи терминал возвратит сообщение:

```
> Hi Kirill-Murashev! You've successfully authenticated, but
  GitHub does not provide shell access.
```

— связь установлена, возможна работа с удалённым репозиторием.

### 2.3.6.3. Создание и установка GPG ключа.

**2.3.6.3.1. Основные сведения.** Использование GPG ключей необходимо для подтверждения подлинности авторства commit. Использование подписи ключом GPG не является обязательным условием при работе с GitHub. Более того, в повседневной рутинной практике оценки чаще всего не возникает необходимость создания публичного репозитория и верификации commit. Однако с учётом возрастающих рисков киберугроз, усиления важности вопросов информационной безопасности, а также порой возникающей необходимости юридического доказывания авторства отчёта об оценке и подлинности его содержания, краткое изучение вопросов использования цифровой подписи представляется целесообразным. Весьма интересной выглядит история проекта. Его первоначальное название G10 является символической отсылкой к 10-й статье Конституции Федеративной Республики Германии [8], гарантирующей тайну переписки и связи. Наиболее известной программой, осуществляющей шифрование и подпись сообщений и файлов, стала PGP (Pretty Good Privacy) [106], разработанная в 1991 году Филиппом Циммерманом [72, 140]. В 1997 году был выпущен открытый стандарт OpenPGP. Его open-source реализацией стал GNU Privacy Guard (GnuPG или GPG) [33], разработанный в 1999 году Вернером Кохом [124].

Для начала, как и в случае с ключами SSH, ключи GPG (приватный и публичный) необходимо генерировать. GitHub поддерживает несколько алгоритмов генерации ключей:

- RSA

- 2432 • ElGamal
- 2433 • DSA
- 2434 • ECDH
- 2435 • ECDSA
- 2436 • EdDSA

2437 — рекомендуемым по умолчанию является алгоритм RSA&RSA 4096.

2438 **2.3.6.3.2. Проверка наличия существующих ключей.** Необходимо запустить  
 2439 Терминал и использовать команду:

```
2440 $ gpg --list-secret-keys --keyid-format=long
2442
```

2443 — либо, в зависимости от системы:

```
2444 $ gpg2 --list-keys --keyid-format LONG
2445
2446
```

2447 — во втором случае может потребоваться предварительная настройка, выполняемая  
 2448 путём выполнения команды:

```
2449 $ gpg2 --list-keys --keyid-format LONG
2450
2451
```

2452 — в случае отсутствия пары ключей, следует перейти к шагу, описанному в под-  
 2453 секции [2.3.6.3.3–71](#), в случае наличия данный шаг можно пропустить и перейти  
 2454 к описанному в подсекции [2.3.6.3.4 на следующей странице–72](#).

2455 **2.3.6.3.3. Генерация пары ключей GPG** В Терминале следует ввести команду:

```
2456 $ gpg --gen-key
2457
2458
```

2459 — Терминал возвратит сообщение, предложив выбрать алгоритм:

```
2460 > Please select what kind of key you want:
2461 (1) RSA and RSA (default)
2462 (2) DSA and Elgamal
2463 (3) DSA (sign only)
2464 (4) RSA (sign only)
2465
2466
```

2467 — следует выбрать 1 либо 2. Далее терминал предложит выбрать длину ключа.  
 2468 Рекомендуется использовать длину в 4096 бит в случае выбора пункта RSA&RSA  
 2469 и 2048 в случае DSA&Elgamal. Далее следует указать срок действия пары ключей  
 2470 либо поставить «0» для генерации бессрочных ключей. Данный выбор не является  
 2471 необратимым: срок действия пары ключей возможно изменить впоследствии. Далее  
 2472 необходимо указать данные пользователя и придумать пароль.

2473 После генерации пары следует проверить её существование путём использования  
 2474 команды:

```
2475 $ gpg --list-secret-keys --keyid-format=long
2476
2477
```

2478 — либо, в зависимости от системы:

```
2479 $ gpg2 --list-keys --keyid-format LONG
2480
2481
```

2482 — Терминал возвратит примерно следующее сообщение:

```
2483 pub      dsa2048/169D4D0EC86C0000 2021-08-14 [SC]
2484
2485     ...
2486 uid                               [ultimate] kirill.murashev (my-key) <
2487     kirill.murashev@gmail.com>
2488
2489     ...
```

2490 — в данном случае идентификатором публичного ключа является значение «169D4D0EC86C0000».  
2491 Введём команду:

```
2492 $ gpg --armor --export 169D4D0EC86C0000
2493
2494
```

2495 — Терминал возвратит полное значение публичного ключа, начинающееся с:

```
2496 -----BEGIN PGP PUBLIC KEY BLOCK-----
2497
2498
```

2499 — и заканчивающееся:

```
2500 -----BEGIN PGP PUBLIC KEY BLOCK-----
2501
2502
```

2503 — полученное значение необходимо скопировать, после чего можно перейти к следу-  
2504 ющему шагу. Дополнительные сведения о работе с GPG можно получить по [ссыл-  
2505 ке \[32\]](#).

2506 **2.3.6.3.4. Добавление публичного ключа на портал GitHub.** Необходимо зай-  
2507 ти на портал GitHub. В меню **Settings** выбрать пункт **SSH and GPG keys**, далее  
2508 нажать **New GPG key**, вставить значение публичного ключа из буфера обмена  
2509 и нажать **Add GPG key**. При выполнении последнего действия система предло-  
2510 жит ввести пароль от аккаунта.

2511 Теперь существует возможность создавать подписанные commit. Для подписи  
2512 конкретного commit следует использовать дополнительные аргумент **-S** команды  
2513 **git commit**. Пример такой команды:

```
2514 $ git commit -S -m "commit_message"
2515
2516
```

2517 — при этом система потребует ввести пароль, придуманный при генерации пары  
2518 ключей. Для включения глобальной опции подписания всех commit по умолчанию  
2519 следует ввести команду:

```
2520 $ git config --global --edit
2521
2522
```

2523 — в открывшемся окне текстового редактора установить следующие значения:

```
[user]
    name = <User.Name>
    email = <user.name@host.com>
    signingkey = <169D4D0EC86C0000> [gpg]
    program = gpg2 [commit]
    gpgsign = true
```

— значения, заключённые в <>, естественно должны быть своими.

#### 2.3.6.4. Установление связи между локальным и удалённым репозиториями

##### 2.3.6.4.1. Отправка содержимого локального репозитория в удалённый.

В первую очередь необходимо скопировать ссылку на репозиторий из меню **Code**. После этого следует зайти в каталог локального репозитория, запустить из него Терминал и ввести команду:

```
$ git remote add origin <hyperref>
```

— указав вместо «hyperref» конкретную ранее скопированную ссылку на удалённый репозиторий.

Для просмотра настроек удалённого репозитория следует ввести команду:

```
git remote -v
```

— терминал возвратит, например такое сообщение:

```
origin https://github.com/Kirill-Murashev/
      AI_for_valuers_book.git (fetch)
origin https://github.com/Kirill-Murashev/
      AI_for_valuers_book.git (push)
```

— как видим имеет место существование двух репозитория:

- **fetch** служит для чтения содержимого удалённого репозитория;
- **push** — для отправки содержимого локального репозитория в удалённый.

Для отправки данных на удалённый сервер следует применить команду:

```
git push origin master
```

— где:

- **push** — указание на действие, которое необходимо выполнить;
- **origin** — наименование сервера, на который следует отправить данные;
- **master** — название ветки, в которую необходимо отправить данные.

Содержимое локального репозитория в том состоянии, в котором оно было зафиксировано в последнем commit, отправлено в удалённый репозиторий на портале GitHub. При этом создаётся дополнительный указатель `remotes/origin/master`, называемый веткой слежения. Данный указатель следует для хранения данных о том, на каком commit находится указатель `Head` в ветке `master` на удалённом сервере `origin`.

**2.3.6.4.2. Получение содержимого удалённого репозитория.** Для получение содержимого удалённого репозитория на локальный компьютер необходимо выбрать каталог, в который планируется загрузка и запустить из него Терминал. При этом следует иметь ввиду, что в данном каталоге будет сформирована новая папка, имя которой будет повторять имя удалённого репозитория — источника. Далее следует использовать команду `git clone`, аргументом которой будет являться ссылка на удалённый репозиторий. Следующие команды предназначены для создания локальной копии репозитория полезных, а зачастую и необходимых для изучения данного материала:

```
$ git clone git@github.com:Kirill-Murashev/  
AI_for_valuers_book.git
```

— создание локальной копии исходного кода данного руководства, его версии в формате PDF, а также дополнительных материалов, использованных при создании.

```
$ git clone git@github.com:Kirill-Murashev/  
AI_for_valuers_R_source.git
```

— создание локальной копии репозитория, содержащего код на языке R, предназначенного для выполнения процедур, описанных в данном руководстве.

```
$ git@github.com:Kirill-Murashev/  
AI_for_valuers_Python_source.git
```

— создание локальной копии репозитория, содержащего код на языке Python, предназначенного для выполнения процедур, описанных в данном руководстве.

**2.3.6.4.3. Обновление репозитория.** В процессе работы особенно в случае совместной работы нескольких специалистов над одним проектом возникает необходимость частой синхронизации локальных репозитория разработчиков. Для выполнения обновления содержимого локального репозитория следует зайти в его локальный каталог, запустить Терминал и использовать команду `git pull`, в качестве аргументов которой указываются имена сервера и ветки:

```
git pull origin master
```

— Git загрузит изменения в случае их наличия.

Поскольку данный проект активно развивается, автор рекомендует выполнять обновление репозитория, содержащих текст данного руководства и программный код, не реже одного раза в месяц.

В процессе совместной работы нескольких специалистов может возникнуть ситуация, при которой они оба захотят отправить свои изменения на сервер. В Git предусмотрена защита: разработчик, отправивший свои изменения позже, получит сообщение об ошибке и предложение выполнить `git pull` в том случае, если его изменения конфликтуют с изменениями первого разработчика, т. е. сервер не может выполнить процедуру `fast forward`. Во избежание такой ситуации рекомендуется всегда сначала использовать команду `git pull`, обновляющую данные о том, на каком `commit` находится указатель `Head` на сервере, и загружающую изменения. Для обновления данных и перемещения ветки слежения без загрузки новых `commit` с сервера можно использовать команду:

```
git fetch origin
```

— а затем использовать команду:

```
git merge origin/master
```

— использование последовательности этих команд равнозначно использованию одной команды `git pull`.

## 2.3.7. Работа с Git в IDE

### 2.3.7.1. Работа в RStudio

End

### 2.3.7.2. Работа в Spyder

End

### 2.3.7.3. Работа в PyCharm

End

End

## 2.3.8. Заключение

Данный раздел содержал лишь основные сведения и инструкции по работе с Git и Github, достаточные для первичной настройки и начала работы. Для более подробного ознакомления с Git и Github можно порекомендовать просмотр данного [видеоурока \[148\]](#), а также изучение [официального руководства \[5\]](#).

## 2.4. Установка и настройка

### 2.4.1. Git

#### 2.4.1.1. Установка на операционных системах, основанных на Debian: Debian, Ubuntu, Mint и т. п.

В операционных системах, основанных на ядре Linux [70], относящихся к ветке Debian [104], Git зачастую бывает уже установлен вместе с системой. Чтобы проверить наличие Git в командную строку терминала следует ввести:

```
git
```

В случае наличия Git в системе, терминал возвратит длинное сообщение, начинающееся примерно следующим образом:

```
usage: git [--version] [--help] [-C <path>] [-c <name>=<
value>]
[--exec-path[=<path>]] [--html-path] [--man-path] [--info-
path] [-p | --paginate | -P | --no-pager] [--
no-replace-objects] [--bare] [--git-dir=<path
>] [--work-tree=<path>] [--namespace=<name>]
```

В случае его отсутствия:

```
Command 'git' not found, did you mean:
```

Во втором случае следует использовать следующие команды:

```
sudo apt update -y
sudo apt install git -y
```

Процесс проходит автоматически и не требует внимания со стороны пользователя.

#### 2.4.1.2. Установка на операционной системе Windows

Установка Git на Windows осуществляется обычным для данной операционной системы образом. Необходимо загрузить установочный файл с соответствующей страницы [26] и запустить процесс установки, желательно приняв при этом все настройки по умолчанию.

#### 2.4.1.3. Установка на macOS

Существует несколько способов установки Git на macOS. Их перечень приведён на соответствующей странице [27] сайта Git. Следует отметить, что в случае наличия в системе Xcode [143] Git также уже присутствует, и его установка не требуется. В данном материала приводится один из возможных способов. Для начала необходимо установить менеджер пакетов Homebrew [38]. Для этого в командной строке терминала необходимо ввести следующую команду:



```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com
/Homebrew/install/HEAD/install.sh)"
```

После этого можно перейти к установке самого Git. Для этого в командной строке терминала необходимо ввести следующую команду:

```
brew install git
```

Как и в случае, описанном выше в секции [2.4.1.1 на предыдущей странице](#)–75, процесс проходит автоматически и не требует внимания со стороны пользователя.

## 2.4.2. R

### 2.4.2.1. Установка на операционных системах, основанных на Debian: Debian, Ubuntu, Mint и т. п.

**2.4.2.1.1. Установка на операционных системах Ubuntu, Mint и производных от них.** Как правило для установки достаточно зайти в Центр приложений, ввести в строку поиска «CRAN» и установить R посредством графического интерфейса. Однако, в зависимости от дистрибутива есть вероятность получения относительно устаревшей версии. Для получения сведений о текущей версии R следует зайти на [официальный сайт](#) [87] и узнать там номер и дату последнего релиза. На момент написания данных строк таковой является версия 4.1.1 (Kick Things) от 2021-08-10. Для проверки версии, установленной в системе в Терминале следует ввести команду:

— в случае автора терминал возвратил сообщение:

```
R version 4.1.1 (2021-08-10) -- "Kick Things"
Copyright (C) 2021 The R Foundation for Statistical
Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and 'citation()'
on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
```

— версия соответствует последнему релизу. В большинстве случаев установка и использование не самой последней версии не вызывает никаких проблем. Однако, в случае, если есть стремление использовать самых свежий стабильный релиз следует отказаться от установки через Центр приложений и выполнить следующую последовательность команд. Сначала необходимо добавить доверенный ключ:

```
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
E298A3A825C0D65DFD57CBB651716619E084DAB9
```

— система возвратит следующее сообщение:

```
Executing: /tmp/apt-key-gpghome.cul0ddtmN1/gpg.1.sh --
keyserver keyserver.ubuntu.com --recv-keys
E298A3A825C0D65DFD57CBB651716619E084DAB9
gpg: key 51716619E084DAB9: public key "Michael Rutter <
marutter@gmail.com>" imported
gpg: Total number processed:
1 gpg: imported: 1
```

— ключ добавлен, можно добавить репозиторий:

```
$ sudo add-apt-repository 'deb https://cloud.r-project.org/
bin/linux/ubuntu focal-cran40/'
```

— далее обновляем зависимости и устанавливаем R:

```
$ sudo apt update -y
$ sudo apt install r-base
```

— R установлен и готов к использованию.

**2.4.2.1.2. Установка на операционной системе Debian.** Одной из основных особенности операционной системы Debian [11] является её стабильность и надёжность. В особенности это касается её ветки stable. Однако достоинства и недостатки часто являются продолжением друг друга. Многие приложения, доступные из стандартных репозиториях Debian могут быть представлены в версиях, отступающих от актуальных на 0.5–2 года. Таким образом, в случае использования операционной системы Debian ветки stable рекомендуется провести самостоятельную установку актуальной версии R. Следует выполнить последовательность команд:

```
$ sudo apt install dirmngr --install-recommends
```

— либо, в случае её недоступности:

```
$ sudo apt install software-properties-common
```

— обе эти команды добавляют необходимый в дальнейшем инструмент `add-apt-repository`. Далее устанавливаем инструмент, необходимый для обеспечения работы протокола `https` при передаче данных из репозитория:

```
2778 $ sudo apt install apt-transport-https
```

2781 Далее добавляем доверенный ключ:

```
2782 $ sudo apt-key adv --keyserver keys.gnupg.net --recv-key '
2783 E19F5F87128899B192B1A2C2AD5F960A256A04AF '
2784
2785
```

2786 — система возвратит сообщение:

```
2787 Executing: /tmp/apt-key-gpghome.y6W4E0Gtfp/gpg.1.sh --
2788 keyserver keys.gnupg.net --recv-key
2789 E19F5F87128899B192B1A2C2AD5F960A256A04AF
2790
2791 gpg: key AD5F960A256A04AF: 4 signatures not checked due to
2792 missing keys gpg: key AD5F960A256A04AF: public key "
2793 Johannes_Ranke_(Wissenschaftlicher_Berater)<johannes.
2794 ranke@jrwb.de>" imported
2795 gpg: Total number processed: 1
2796 gpg: imported: 1
2797
```

2798 — теперь можно перейти к установке самого R. Следует обратить внимание на тот факт,  
2799 что содержание аргумента приведённой ниже команды зависит от используемой  
2800 версии OS Debian. На момент написания этих строк текущей стабильной версией  
2801 является Debian 11 «bullseye». В этом случае команда будет выглядеть следующим  
2802 образом:

```
2803 $ deb http://cloud.r-project.org/bin/linux/debian bullseye -
2804 cran40/
2805
2806
```

2807 — для получения дополнительных сведений следует обращаться к [соответствующей](#)  
2808 [странице](#) [12] сайта R. Далее следует выполнить последовательность команд:

```
2809 $ sudo apt update -y
2810 $ sudo apt install r-base
2811
2812
```

2813 — R установлен и готов к использованию.

#### 2814 2.4.2.2. Установка на операционных системах Windows и macOS

2815 В данном случае установка не требует никаких специфических действий и осу-  
2816 ществляется путём загрузки установочного файла с [соответствующей страницы](#)  
2817 сайта R [87] и запуска установщика.

#### 2818 2.4.3. RStudio

2819 Независимо от используемой операционной системы самым простым способом  
2820 установки RStudio является загрузка установочного образа, соответствующего опе-  
2821 рационной системе, со [страницы](#) сайта RStudio [83].

#### 2822 2.4.4. Python

2823 End

#### 2824 2.4.5. Spyder

2825 End

#### 2826 2.4.6. PyCharm

2827 End

#### 2828 2.4.7. SQL

2829 End

2830 End

2831 Теоретическая часть

2832 Глава 3.

2833 Математическая основа анализа  
2834 данных

2835 End

## Глава 4.

### Основные понятия

#### 4.1. Что было раньше: курица или яйцо? Соотношение понятий **statistics, machine learning, data mining, artificial intelligence**

На сегодняшний день можно говорить о существовании множества понятий, описывающих применение математических и статистических методов при решении практических задач. В целом, можно без преувеличения сказать, что в настоящее время нет такой области деятельности человека, в которой бы не применялись математические методы и модели. Невозможно охватить все аспекты применения математических методов — в данном разделе будут рассмотрены лишь интересующие нас вопросы анализа данных применительно к оценке. Существует несколько общепринятых понятий, описывающих группы методов и подходов, применяемых при анализе данных и укоренившихся в сознании общества. В таблице 4.1.1 приводится перечень наиболее распространённых терминов.

Таблица 4.1.1. Перечень понятий, описывающих группы методов анализа данных

№	Англоязычный термин	Русскоязычный термин
0	1	2
1	Statistics	Математическая статистика
2	Machine Learning	Машинное обучение
3	Data mining	Интеллектуальный анализ данных
4	Artificial intelligence	Искусственный интеллект

Из таблицы 4.1.1 следует, существуют как минимум, четыре разных понятия, описывающих широкую, но всё же единую с точки зрения конечной цели область. Целью данного раздела является попытка разобраться в следующих вопросах:

- что представляет из себя каждое направление;
- что есть общего у них, и в чём они различаются;

- какие именно методы и средства используются в каждом из этих направлений;
- чем мы будем заниматься в процессе изучения данной работы.

Следует отметить, что на сегодняшний день в вопросе того, как именно следует разделять эти понятия, отсутствует консенсус. На эту тему продолжают вестись дискуссии. Забегая вперёд, можно сказать, что скорее всего нет смысла говорить о жёстком разделении этих понятий. Едва большинство конкретных методов могут применяться в рамках каждого из этих направлений. И всё же, по мнению автора, вопрос соотношения вышеуказанных понятий заслуживает должного внимания. Следует отметить, что большая часть рассуждений и выводов, сделанных в данном разделе, является не более чем мнением автора и не должна рассматриваться как-то иначе.

Вряд ли требуется много слов для того, чтобы объяснить, что представляет собой **математическая статистика**. В целом, можно сказать, что данный раздел математики тесно связан с Теорией вероятности и использует единый с ней понятийный аппарат. При этом, математическая статистика допускает как **частотный** [141], так и **байесовский** [116] подход к понятию вероятности. В целом, можно сказать, что методы математической статистики, основанные на частотной вероятности, основываются на свойствах данных (например распределениях), исходят из базовой предпосылки о случайности распределения значений переменных, случайности каких-либо различий между выборками и отсутствии значимых зависимостей между переменными и предназначены для построения интерпретируемых моделей, описывающих взаимосвязь между данными. Как правило, применение методов статистики начинается со спецификации модели, методы которой, в общем случае, позволяют вывести конкретную модель на основе минимизации функции потерь, неизбежных вследствие наличия «шума» [81]. Сильной стороной математической статистики является хорошая интерпретируемость результатов, а также возможность применения в условиях ограниченного числа наблюдений. Слабой стороной является зависимость от распределения значений переменных, возможность работы только с *данными, характеризующими отношения*. Классическим примером применения методов математической статистики является построение линейной регрессионной модели.

**Машинное обучение** представляет построение алгоритмов, способных обучаться. Обучением алгоритма (программы) на основе опыта считается такой процесс, при котором по мере обучения производительность этого алгоритма (программы) возрастает в соответствии с некой производительности. Основным направлением машинного обучения является обучение по прецедентам. Автоматический сбор и распознавание исходных данных также относятся к задачам машинного обучения. При этом многие методы машинного обучения напрямую взяты из математической статистики [81]. Машинное обучение находится на стыке математической статистики, методов оптимизации и классических математических дисциплин, но также имеет собственную специфику, связанную с проблемами **вычислительной эффективности** [59] и **переобучения** [58] [55]. Примером применения методов машинного обучения является построение случайного леса решающих деревьев.

**Интеллектуальный анализ данных** в узком смысле, предложенном в 1992 году [Г. И. Пятецким-Шапиро](#) [Г. И. Пятецким-Шапиро](#) [131], представляет собой процесс обнаружения в сырых данных ранее неизвестных, нетривиальных, практически полезных, доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности [54]. В широком смысле интеллектуальный анализ данных представляет собой применение человеком методов математической статистики и машинного обучения применительно к конкретным данным для решения конкретных задач. Как правило, речь идёт о большом и сложном наборе данных. Целью применения методов интеллектуального анализа данных чаще всего является получение знаний в тех областях, в которых их не хватает либо предсказание точных значений будущих наблюдений. При этом, как правило, нет цели формирования выводов о свойствах данных либо эта цель второстепенна [81]. Концепция интеллектуального анализа данных базируется на следующих предпосылках:

- данные могут быть неточными, неполными (содержать пропуски), противоречивыми, разнородными, косвенными, и при этом иметь гигантские объёмы, вследствие чего их интерпретация в конкретном случае требует значительных интеллектуальных усилий;
- сами алгоритмы анализа данных могут обладать элементами машинного «интеллекта», в частности, способностью обучаться по прецедентам, т. е. общие выводы на основе частных наблюдений, при этом разработка подобных алгоритмов также требует значительных интеллектуальных усилий;
- процессы переработки сырых данных в структурированные, структурированные в информацию, а информации в знания уже не могут быть выполнены вручную, и требуют нетривиальной автоматизации [54].

Понятие **искусственный интеллект** представляет собой область деятельности, включающую в себя науку и технологии, направленную на создание искусственных интеллектуальных агентов, т. е. алгоритмов и компьютерных программ, способных действовать так, как мог бы действовать настоящий интеллектуальный агент, например человек. Искусственный интеллект не обязательно должен основываться на индуктивных методах. Например, он может быть просто настроен на увеличение значений некой переменной  $X$  всеми доступными способами безотносительно конкретных применяемых при этом алгоритмов. В определённом смысле любые успешные действия машины, осуществляемые ей одновременно не детерминировано и не случайным образом, можно назвать искусственным интеллектом [81, 60].

Следует отметить, что данное понятие очень популярно в массовой культуре, искусстве и общественном сознании, вследствие чего подвержено неверному толкованию. Современное понятие искусственного интеллекта, а также современное состояние науки о нём не предполагают создание искусственного интеллекта равного человеческому либо хоть сколько-нибудь сопоставимого с ним. Более того,



в случае необходимости проведения параллелей с биологическими интеллектуальными агентами, правильнее говорить не об искусственном интеллекте, а об искусственных инстинктах и рефлексам. Стремление минимизировать функцию ошибки при построении модели линейной регрессии можно сравнить с инстинктом, а попытку перебора её коэффициентов — с рефлексом.

В определённом смысле можно сказать, что правильнее говорить не о создании интеллектуальных агентов (вопрос того, что такое интеллект, чем он отличается от сознания, разума, воли — является слишком сложным и выходит за рамки науки о искусственном интеллекте), а о создании рациональных агентов, т. е. таких, которые обеспечивают принятие рациональных, а по возможности оптимальных, решений на основе фактически доступных данных [81].

Подытоживая всё вышесказанное можно повторить, что чёткое разграничение этих четырёх понятий нецелесообразно, а скорее всего невозможно. Ниже приводятся краткие выводы, сделанные автором в ходе изучения вопроса соотношения данных понятий:

**Математическая статистика** — наука и совокупность практических методов, нацеленные на количественный анализ данных, описание их свойств и построение моделей. Основывается на вероятностных моделях. При этом использование компьютера не является обязательным.

**Машинное обучение** — область деятельности, нацеленная на создание алгоритмов способных обучаться, а также оптимизацию вычислений. При этом конечной целью данных алгоритмов является предсказание значений новых наблюдений. Как правило, не фокусируется на свойствах данных. Чаще основывается на байесовском подходе к вероятности. Использование компьютера является обязательным, при этом важную роль играет вычислительная эффективность.

**Интеллектуальный анализ данных** — область деятельности, направленная на получение новых знаний и объяснение закономерностей в конкретной области путём применения методов статистики и машинного обучения. Фокусируется на практическом применении, оптимизации наборов данных. Использование компьютера является обязательным.

**Искусственный интеллект** — область деятельности, направленная на создание рациональных агентов, способных применять любые доступные методы для формирования рациональных решений на основе всех доступных данных. Компьютер сам является агентом.

В таблице 4.1.2 на с. 87 приведены обобщающие сведения. Возникает вопрос: чем именно мы будем заниматься в процессе работы с данным руководством. С одной стороны, как уже было сказано выше, нельзя провести жёсткое разграничение между всеми этими понятиями. С другой, каждое из этих направлений деятельности имеет свою специфику. Кроме того, нельзя забывать о том, что целью данной работы

является применение современных технологий анализа данных в конкретной области деятельности — оценке стоимости. Для начала можно сделать краткий обзор уже существующих в российской практике разработок в каждом направлении.

**Математическая статистика** в российской оценочной практике применяется сравнительно давно. Существует некоторый набор литературы и методических материалов, посвящённых применению её методов в оценке. Можно вспомнить ряд работ С. В. Грибовского, Н. П. Барина, В. Г. Мисовца, а также иных авторов, посвящённых вопросам применения математических методов в оценочной деятельности. Основным предметом интереса данных работ чаще всего выступает применение корреляционного и регрессионного анализа в их параметрических вариантах. Как правило в них рассматривается применение программного продукта Microsoft Excel в качестве основного и единственного инструмента анализа. Не погружаясь в рассуждения о методах, предлагаемых в данных работах, их достоинствах и недостатках, в целом, можно высказать некоторое сожаление о том, что данное направление развития оценки в принципе не получает широкого распространения, почти полностью уступая место практикам оценки, основанным на применении минимального числа наблюдений, называемых в терминологии оценки объектами-аналогами, и последующей корректировки значений на основе неких «справочников». Таким образом, можно сказать, что применение методов математической статистики в оценке не является чем-то принципиально новым. Данная работа также будет включать существенный объём материала, посвящённого их применению. Как уже было сказано выше, одним из достоинств математической статистики является хорошая интерпретируемость результатов, что является важным в условиях необходимости их доказывания и защиты. Ещё одной сильной стороной является развитый аппарат описательных методов, позволяющий делать общие выводы о свойствах данных, в нашем случае — свойствах открытых рынков. Таким образом, рассмотрение и применение методов математической статистики представляется необходимым.

**Машинное обучение** пока что не получило широкого распространения в среде профессиональных оценщиков. При этом уже существует ряд сервисов, позволяющих предсказывать стоимость объектов на основе методов машинного обучения. Примером является [Калькулятор недвижимости \[146\]](#), созданный Циан.Групп. По мнению автора, необходимость применения методов машинного обучения обусловлена в частности тем, что существенная часть данных открытых рынков не относится не только к *данным, характеризующим отношения*, но в принципе не представляет собой количественные данные. Адрес, конструктивные особенности, техническое состояние — всё это качественные данные, с трудом поддающиеся какой-либо трансформации в количественные. Существуют методы квалиметрии, позволяющие провести преобразование качественных данных в количественные, однако в любом случае на выходе могут быть получены лишь порядковые данные, применение к которым методов математической статистики возможно лишь в ограниченном объёме. При этом многие методы машинного обучения свободны от зависимости от типа исходных данных, а также вида их распределения. Кроме того, существует необходимость автоматизации сбора исходных данных. Ситуация, при которой оценщики либо их помощники вручную копируют данные с сайтов объявлений, а затем вруч-

ную вставляют в свои рабочие материалы, выгляди нелепой для 2021 года.

**Интеллектуальный анализ данных** представляет собой междисциплинарное направление деятельности. Очевидно, что планирование исследования, выбор источников данных, поиск и объяснение закономерностей, соотнесение промежуточных и итоговых результатов с априорными знаниями, их интерпретация и описание, — всё это является важной частью деятельности оценщика.

**Искусственный интеллект** не имеет однозначного и точного определения. Любой недетерминированный алгоритм, позволяющий решать практические задачи неслучайным образом может считаться реализацией искусственного интеллекта.

Таким образом, все четыре направления представляют интерес и будут рассматриваться в данной работе. Можно лишь в очередной раз повторить мысль о том, что любое их разделение носит приблизительный и отчасти условный характер.

Таблица 4.1.2. Обобщение сведений

	Математическая статистика	Машинное обучение	Интеллектуальный анализ данных	Искусственный интеллект
Приоритетный подход к вероятности	Частотный	Байесовский	Оба	Байесовский
Опирается на свойства данных	Да	Нет	Да	Нет
Требовательность к количеству наблюдений	Умеренная	Высокая	Высокая	Средняя
Применение компьютера	Желательно	Обязательно	Обязательно	Обязательно
Основной инструмент	R	Python	R и Python	Python
Включает в себя методы подготовки эксперимента и отбора данных	Нет	Нет	Да	Нет
Предполагает наличие знаний в области, в которой проводится эксперимент	Нет	Нет	Да	Нет
Стремится к вычислительной эффективности	Нет	Да	Да	Да
Позволяет делать общие выводы о свойствах данных	Да	Нет	Да	Нет
Позволяет давать точные прогнозы значений новых наблюдений	Частично	Да	Да	Да
Основная область знаний, к которой относится направление	Математика	Информатика	Междисциплинарная	Информатика
Обеспечивает хорошую интерпретацию результатов	В большинстве случаев	Не всегда	Да	Нет

## Глава 5.

# Каллиграфия анализа данных: краткое стилевое руководство

Как известно, код чаще читают, чем пишут. Чтобы код мог быть прочитан и понят не только со стороны его автора, последнему следует придерживаться рекомендаций т. н. стиливых руководств.

### 5.1. Стилиевое руководство по R

В отличие от Python, в R нет единого стилиевого стандарта. Существует как минимум четыре руководства:

- руководство, разработанное Hadley Alexander Wickham [37] — *Advanced R. Style Guide* [89];
- расширенная версия предыдущего — *The tidyverse style guide* [91];
- руководство от компании Google, представляющее собой форк [138] The tidyverse style guide — *Google's R Style Guide* [35];
- руководство от проекта Bioconductor — *Coding Style* [4].

Также существует [пост](#) [13] на портале Habr.com, обобщающий опыт применения стиливых руководств. Далее приводятся некоторые наиболее важные, по мнению автора, стиливые принципы написания кода на R, применяемые им самим. Кроме того, рекомендуется установка пакета `formatR` [144], автоматизирующего часть задач по стилистическому оформлению кода. Некоторые из приведённых ниже принципов не являются единственно верными и допускают вариации. Все рекомендуемые варианты изложены, как уже было сказано выше, в [89, 91, 35, 4].

#### 5.1.1. Имена файлов и путей к ним

Имена файлов скриптов и путь к ним должны содержать только латинские буквы, цифры и специальные символы. Использование пробелов и кириллических букв

не допускается. Следует назначать информативные имена. Основными расширениями имён файлов, содержащих код на R являются:

- .R — расширение скриптов;
- .Rmd — расширение файлов [R Markdown](#) [78] и [R Notebook](#) [88].

Таким образом:

- \home\username\my-projects\depreciation\auto\r-code\gazelle\_dep.R — *правильно*;
- \home\username\my-projects\износ\транспорт\R-code\gazelle\_dep.R — *неправильно*;
- \home\username\my projects\depreciation\auto\r code\gazelle\_dep.R — *неправильно*;
- \home\username\my projects\AAA\BBB\CCC\589509380943.R — *неправильно*.

В случае необходимости последовательного запуска файлов, их имена следует предварять цифрами, соответствующими порядку запуска, например:

```
000-load.R
001-parse.R
002-analyze.R
...
009-visualize.R
010-results.R
```

— в случае возникновения впоследствии необходимости добавления файла между уже существующими, можно использовать добавление букв к цифровому префиксу:

```
...
002aa-analyze.R
002ab-check.R
```

В именах файлов следует использовать только строчные буквы. [89, 91]

## 5.1.2. Синтаксис

### 5.1.2.1. Имена классов, функций и переменных

В данном вопросе до сих пор нет единства. В целом можно говорить о пяти возможных вариантах:

```

3095 lowerCamel
3096 period.separation
3097 lower_case_with_underscores
3098 alllowercase
3099 UpperCamel
3100
3101

```

3102 — [89, 91] рекомендуют использовать стиль *lower\_case\_with\_underscores* во всех  
 3103 случаях, [35] — *lower\_case\_with\_underscores* для переменных и *UpperCamel* для функ-  
 3104 ций, [4] — *lowerCamel* для переменных и функций и *UpperCamel* для классов [121].  
 3105 При этом в самом R изначально реализован стиль *period.separator* как для классов,  
 3106 так и для функций, при этом часть функций имеет названия в стиле *alllowercase*.  
 3107 Автор использует следующий вариант:

- 3108 • имена классов — UpperCamel;
- 3109 • имена функций — lowerCamel;
- 3110 • имена переменных — lower\_case\_with\_underscores.

3111 Данный выбор обусловлен желанием избежать использовать точек в именах, по-  
 3112 скольку использование точек лучше зарезервировать за объектами системы S3 [90].  
 3113 Использование стиля lowerCamel в именах переменных приводит к плохой чита-  
 3114 емости в случае их значительной длины. Приведённый выше подход к названиям  
 3115 является лишь обобщением опыта автора и не может считаться единственно верным  
 3116 или даже просто предпочтительным. [13, 89, 91, 35, 4]

### 3117 5.1.2.2. Использование пробелов

3118 Во всех случаях пробел ставится после запятой и никогда не ставится перед ней:

```

3119 #Правильно:
3120 x[, 1]
3121
3122
3123 #Неправильно:
3124 x[, 1]
3125 x[, 1]
3126 x[, 1]
3127

```

3128 В общем случае пробел не ставится ни до, ни после как открывающих, так и за-  
 3129 крывающих круглых скобок:

```

3130 #Правильно:
3131 mean(x, na.rm=TRUE)
3132
3133
3134 #Неправильно:
3135 mean_(x, na.rm=TRUE)
3136 mean( x, na.rm=TRUE )
3137

```

Из этого правила существует исключение: при использовании операторов `if`, `for` и `while` после них перед открывающей скобкой ставится пробел:

```
#Правильно
if (debug) {
  show(x)
}

#Неправильно
if(debug) {
  show(x)
}
```

Пробел также ставится после закрывающей круглой скобки для сообщения функции её аргументов, заключаемых в фигурные скобки:

```
#Правильно :
function(x) {}

#Неправильно
function(x){}
function(x){}
```

Оператор охвата `{{}}` должен содержать внутренние пробелы:

```
#Правильно
max_by <- function(data, var, by) {
  data %>%
    group_by({{by}}) %>%
    summarise(maximum = max({{var}}, na.rm = TRUE))
}

#Неправильно
max_by <- function(data, var, by) {
  data %>%
    group_by({{by}}) %>%
    summarise(maximum = max({{var}}, na.rm = TRUE))
}
```

Большинство операторов в *инфиксной нотации* [101] т. е. таких операторов, которые расположены между операндами, например `=`, `+`, `-`, `/`, `<-`, отделяются пробелами:

```
#Правильно :
height <- (feet * 12) + inches
mean(x, na.rm = TRUE)

#Неправильно :
```



```

3185 height<-feet*12+inches
3186 mean(x, na.rm=TRUE)
3187

```

3188 Существует ряд исключений из этого правила:

- 3189 • операторы с **наиболее высоким приоритетом** [77]: ::, :::, \$, @, [, [[, ^,  
3190 **унарный** - [125], унарный +, : используются без пробелов:

```

3191 #Правильно
3192 sqrt(x^2 + y^2)
3193 df$z x <- 1:10
3194
3195 #Неправильно
3196 sqrt(x ^ 2 + y ^ 2)
3197 df $ z x <- 1 : 10
3198
3199

```

- 3200 • односторонние формулы, в которых правая часть является единственным иден-  
3201 тификатором:

```

3202 #Правильно:
3203 ~foo
3204 tribble(
3205   ~col1, ~col2, "a", "b"
3206 )
3207
3208 #Неправильно
3209 ~ foo
3210 tribble(
3211   ~ col1, ~ col2, "a", "b"
3212 )
3213
3214

```

3215 — при этом односторонние формулы, имеющие сложный идентификатор в правой  
3216 части, требуют пробелов:

```

3217 #Правильно
3218 ~ .x + .y
3219
3220 #Неправильно
3221 ~.x + .y
3222
3223

```

- 3224 • специфические выражения tidy: !!, !!!:

```

3225
3226 #Правильно :
3227 call (!!xyz)
3228
3229 #Неправильно :
3230 call(!! xyz)
3231 call( !! xyz)
3232 call(! !xyz)
3233

```

3234 • оператор вызова помощи:

```

3235
3236 #Правильно :
3237 package?stats
3238 ?mean
3239
3240 #Неправильно
3241 package ? stats
3242 ? mean
3243

```

3244 Использование дополнительных пробелов целесообразно в случае необходимости  
 3245 выравнивания строк по операторам = и <-.  
 3246

```

3247 list(
3248   total=a+b+c,
3249   mean=(a+b+c)/n
3250 )
3251

```

3252 — при этом в любом случае нельзя ставить пробелы там, где это не рекомендуется  
 3253 либо запрещено.

```

3254 End
3255 End
3256 End

```

## 3257 5.2. Стилизовое руководство по Python

```

3258 End
3259 End

```

3260 Глава 6.

3261 Автоматизированный сбор данных

3262 The End