Опрос программистов

Penza Street Analitics

Table of Contents

# Вступление

Для того, чтобы создавать какое-либо системное или прикладное программное обеспечение, следует изучить целевую аудиторию потенциального продукта и интересы его возможных потребителей.

*Penza Street* заинтересована в создании инструментов, помогающих программистам, поэтому решила провести опрос среди программистов и попытаться понять, что это за люди, и чего они хотят.

Результаты исследования представлены ниже. Не все результаты были правильно и грамотно интерпретированы, так как сама структура опроса обладала массой изъянов. Компания *Penza Street* не рискнула проводить второй, более качественный опрос во избежание утери доверия среди целевой аудитории.

О серьёзности некоторых выводов говорить не приходится, поэтому, надеемся, вам понравится количество юмора в статье, ибо некоторые закономерности представляют из себя откровенный бред (*Но, статистически вероятный бред*).

## От автора

*Penza Street Analitics* - дочерняя организация от *Penza Street Company*, уникальной в своём роде компании. В чём её уникальность? Хотя бы в том, что все возможные должности в ней занимает один человек, а сама компания не закреплена ни одним правовым актом ни одной страны мира.

Так что, дорогие читатели. [*Павел Соломатин*](https://vk.com/crave_ozer_man), приятно познакомиться.

# Предупреждение!

В статье могут присутствовать циничные и даже оскорбительные для конкретных групп людей, но вполне цензурные, высказывания. Эти высказывания не отражают позицию автора, а являются лишь стереотипами, не соответствующими жизни.

Автор уточняет, что дискриминационные высказывания не касаются расового, национального и религиозного характера, что уже облегчает, если не отменяет наказание по законодательству РФ.

Наличие таких высказываний объясняется большим объёмом статьи, так как не каждый дочитает до конца, не заснув. Автор попытался будить читателей смехом и внутренним негодованием при прочтении.

# Список терминов

Текст исследования изобилует профессиональным и простонародным жаргоном (а также результатом работы личного сленгогенератора). Поэтому для неподготовленных читателей здесь приведён краткий список непонятных терминов.

* Биг Дата - (от англ. Big Data) отрасль программирования, занимающаяся обработкой больших объёмов данных
* Бородатый гуру - человек с высокой квалификацией в конкретной отрасли
* Буржуй - представитель буржуазного класса, он же зажиточный человек
* Веб - веб-разработка - разработка приложений, работающих в среде Интернет
* Великий Третий Змей - Python версии 3.0.0 и выше
* Геймер - игрок-любитель, либо профессионал
* Дата-саентист - (от англ. Data Science) работник в отрасли исследования и обработки данных
* Диструбутив - версия для распространения
* Информационный пролетариат - работники сферы IT
* Кодить - писать код
* Линух - траслитерационный перевод ОС Linux
* Малина - одноплатный компьютер Raspberry Pi
* Маргинал - индивид, которого трудно отнести к той или иной группе по конкретной систематизации, чаще всего, по социальной
* Олимп-прогер - программист, чаще всего школьник, спецализирующийся на решении олимпиадных заданий по программированию
* ОС - операционная система
* Питон - язык пограммирования Python
* Питонист - программист, пишущий код на Python
* Плюсы - язык программирования C++
* Полный профан - антоним для “бородатого гуру”, человек, совершенно неразбирающийся в конкретной отрасли
* Прогер - программист
* Прокаченный комп - компьютер с хорошими показателями производительности, благодаря хорошей сборке и программной оптимизации
* Пыха - язык программирования PHP
* Рандомный - случайный
* Ручное железо - программируемые микрокомпьютеры и микроконтроллеры
* Сахар - синтаксические конструкции, встроенные в языки программирования, упрощающие написание и понимание кода.
* Сферический единорог в вакууме - образ чего-либо идеального
* Топить за … - отстаивать ту или иную точку зрения
* Тянущий комп - компьютер, способный оперативно решать задачи, которые перед ним ставит программист, независимо от уровня сложности операций
* Цзиньпин - глава КНР, а также язык программирования Си
* Чалиться - продолжительно находиться где-либо без особых притеснений
* Шарпы - язык программирования С#
* Юзать - использовать, применять
* Яблоко-комп - компьютер марки Apple Macintosh
* Яблочная продукция - продукция компании Apple
* Яблоко-человек - пользователь продукции компании Apple

# История данных

Данные, использующиеся в опросе, были собраны за август 2019 года. В данных хранятся сведения о 111 программистах.

[Опрос](https://forms.yandex.ru/u/5d4336b07effdd03528c193c/) является до сих пор открытым, можете заполнить анкету. Это очень поможет ~~(смех из-под стола)~~.

Программисты по себе люди замкнутые и большинстве своём разрозненные личности. Собрать их в одном месте было нелегко. Если вам когда-то повезло пройти этот опрос, то скорее всего вы относились к этим категориям людей:

1. Выпускники Яндекс.Лицея 2019 года, которые были в офисе Яндекса на выпускном. (автор и сам таким является)
2. Участники летней смены образовательного центра “Сириус” (отдельная благодарность [*Алексею Медведеву*](https://vk.com/medal99) за помощь в сборе данных)
3. Программисты Пензы, которые каким-либо образом знакомы с автором статьи.

# Математический аппарат исследования

Кто считает себя бородатым гуру статистики или просто не хочет вникать в вот это вот всё, может смело пропустить этот раздел.

Кто совсем ничего не хочет понимать, пусть переходит к разделу “Распределение ответов”

Кто зашёл на пять минут, пусть сразу же переходит к разделу “Найденные взаимосвязи”

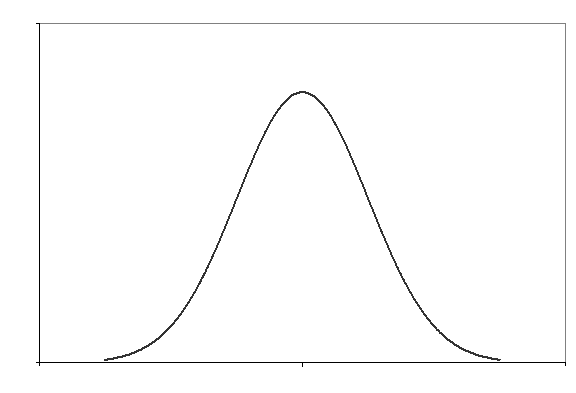
Математика - это всегда не просто, поэтому начнём издалека.

В основе исследования лежит статистика как наука в её чистом виде. Субъекты исследования - программисты. Объект исследования - предпочтения и навыки программистов и прочих посетителей опроса в мире информационных технологий. В данном случае, ответы - это свойства объектов, то есть опрошенных.

Любые критерии могут быть разделены на качественные и количественные. Большинство критериев в опросе было качественными, так как не было вопросов в стиле “оцените это по шкале от 1 до 10”. В это тоже есть минус опроса - нельзя ответить на вопрос “сколько?”. Качественные признаки ещё называют номинативными.

Однако, статистика на то и наука, что умеет много чего. Например:

*Для одной переменной*:

1. Если она качественная, проверить её на равномерность распределения (в каждой группе элементы встречаются одинаково часто)
2. Если она количественная, проверить её на нормальность распределения (особый вид распределения, выглядит как-то так)  За первый пункт отвечает [Критерий Хи-квадрат](https://en.wikipedia.org/wiki/Chi-squared_test), а за второй [Тест Шапиро-Уилка](https://en.wikipedia.org/wiki/Shapiro%E2%80%93Wilk_test).

Правда, как оказалось, данные вышли далеко не равномерные и не нормальные. Но статистика и это может обойти

*Для двух переменных*:

1. Если обе качественные, то проверить их на связанность, или *корреляцию*.
2. Если одна качественная, а другая количественная, то проверить различие количественного признака при разбиении на группы.
3. Если обе качественные, то проверить на пропорциональность групп по двум признакам.

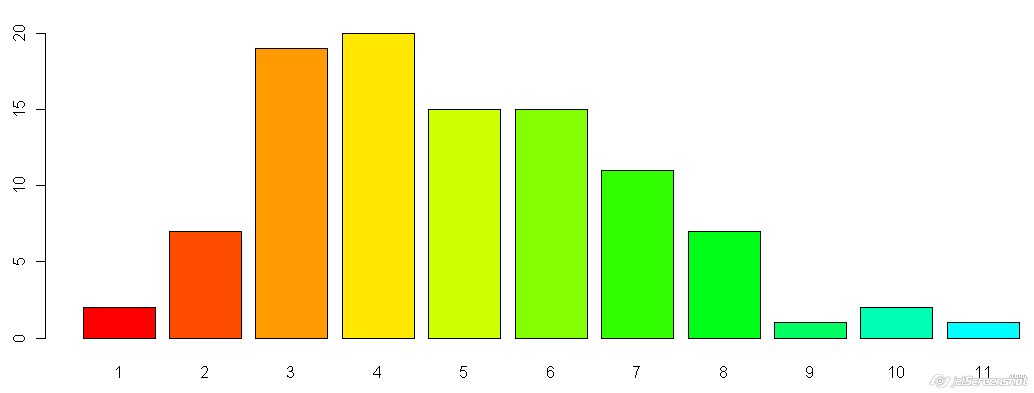
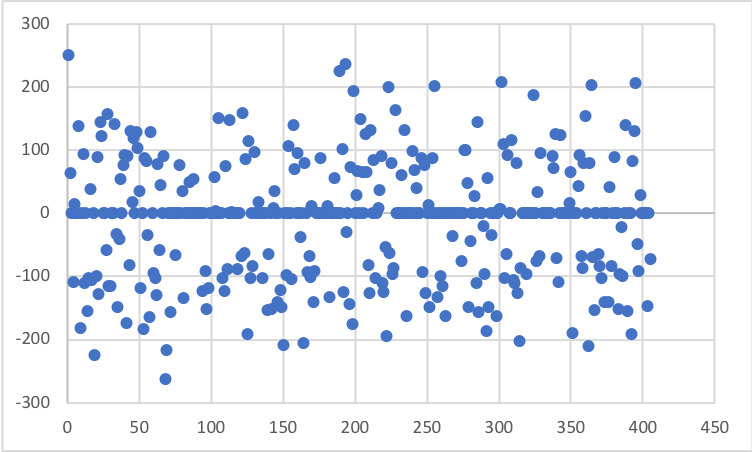
За первый пункт отвечает [Коэфиициент корреляции Пирсона](https://en.wikipedia.org/wiki/Pearson_correlation_coefficient), за второй - [Дисперсионный анализ](https://en.wikipedia.org/wiki/Analysis_of_variance), а за третий - [Точный тест Фишера](https://en.wikipedia.org/wiki/Fisher%27s_exact_test)

Правда, все первые два теста требуют “нормальности” данных, которой в этих данных нет.

Поэтому существуют их непараметрические аналоги:

1. [Коэффициент корреляции Кендала](https://en.wikipedia.org/wiki/Kendall_rank_correlation_coefficient)
2. [Тест Краскела-Уоллиса](https://en.wikipedia.org/wiki/Kruskal%E2%80%93Wallis_one-way_analysis_of_variance)

При визуализации данных будут использованы два вида графиков:

1. [Столбчатая гистограмма](https://en.wikipedia.org/wiki/Bar_chart) 
2. [Диаграмма рассеяния](https://en.wikipedia.org/wiki/Scatter_plot) 

# Программное обеспечение, использованное в исследовании

Кто не хочет вникать в то, что такое R, и как было реализовано это исследование, может смело пропускать этот раздел.

## Ссылки на программное обеспечение

В исследовании был использован язык программирования [**R**](https://www.r-project.org/) версии 3.6.1.

Готовый файл статьи был получен с помощью пакета [RMarkdown](https://rmarkdown.rstudio.com)

Для визуализации был использован пакет [ggplot2](https://ggplot2.tidyverse.org/), а также [ggthemes](https://CRAN.R-project.org/package=ggthemes). Для отображения красивых шрифтов на графиках был использован пакет [extrafont](https://CRAN.R-project.org/package=extrafont)

Для загрузки и выгрузки данных через формат .xlsx были использованы библиотеки [readxl](https://CRAN.R-project.org/package=readxl) и [writexl](https://CRAN.R-project.org/package=writexl)

## Код исследования

Дабы не грузить читателей кодовыми вставками, весь код исследования будет записан в одном месте - здесь.

Полную версию кода исследования можно увидеть на [GitHub](https://github.com/PenzaStreetGames/ProgOpros)

# rm(list = ls())  
options(stringsAsFactors = TRUE)  
library(readxl)  
library(dplyr)  
opros <- read\_excel("ProgOprosEdited.xlsx")  
opros <- select(opros, -number) # колонка номеров не нужна  
  
# Функция, преобразующая набор строк в фактор (словарь)  
factorise <- function(opros) {  
 factoring\_cols <-  
 c(  
 "gender",  
 "status",  
 "processor",  
 "microboard",  
 "desctop\_os",  
 "mobile\_os",  
 "editor\_theme",  
 "cycle\_recursion",  
 "cycle",  
 "java\_kotlin",  
 "zero\_division",  
 "indexing",  
 "typing",  
 "slow\_python",  
 "list\_mutable",  
 "sugar",  
 "list\_expressions",  
 "ternar\_module",  
 "patterns",  
 "mobile\_desctop",  
 "web",  
 "back\_front\_end",  
 "flask\_django",  
 "python",  
 "cpp",  
 "javascript",  
 "pascal",  
 "csharp",  
 "java",  
 "c",  
 "php",  
 "kotlin",  
 "lua",  
 "scratch",  
 "basic",  
 "go",  
 "ruby",  
 "fasm",  
 "bf",  
 "haskel",  
 "pycharm",  
 "vscode",  
 "idle",  
 "notepad",  
 "notepadpp",  
 "wing",  
 "sublime",  
 "jupiter",  
 "atom",  
 "console",  
 "machine\_learning",  
 "big\_data",  
 "metaprog",  
 "quantum",  
 "cryptography",  
 "math"  
 )  
 colnames(opros)  
 for (string in factoring\_cols) {  
 # print(string)  
 opros[[string]] <- factor(opros[[string]])  
 }  
   
 return(opros)  
}  
df\_struct <- list() # список, хранящий данные о структуре данных  
df\_struct$numeric\_vars <- # список количественных переменных  
 c(  
 "languages\_number",  
 "editors\_number",  
 "future\_number",  
 "humour",  
 "other\_opinion",  
 "sugar\_using",  
 "python\_discontent",  
 "middle\_answers",  
 "dont\_know",  
 "web\_using",  
 "apple"  
 )  
df\_struct$quality\_vars <- # список качественных переменных  
 c(  
 "gender",  
 "status",  
 "processor",  
 "microboard",  
 "desctop\_os",  
 "mobile\_os",  
 "editor\_theme",  
 "cycle\_recursion",  
 "cycle",  
 "java\_kotlin",  
 "zero\_division",  
 "indexing",  
 "typing",  
 "slow\_python",  
 "list\_mutable",  
 "sugar",  
 "list\_expressions",  
 "ternar\_module",  
 "patterns",  
 "mobile\_desctop",  
 "web",  
 "back\_front\_end",  
 "flask\_django",  
 "python",  
 "cpp",  
 "javascript",  
 "pascal",  
 "csharp",  
 "java",  
 "c",  
 "php",  
 "kotlin",  
 "lua",  
 "scratch",  
 "basic",  
 "go",  
 "ruby",  
 "fasm",  
 "bf",  
 "haskel",  
 "pycharm",  
 "vscode",  
 "idle",  
 "notepad",  
 "notepadpp",  
 "wing",  
 "sublime",  
 "jupiter",  
 "atom",  
 "console",  
 "machine\_learning",  
 "big\_data",  
 "metaprog",  
 "quantum",  
 "cryptography",  
 "math"  
 )  
df\_struct$vars <- colnames(opros) # список всех переменных  
df\_struct$simple\_vars <- # список логически полных переменных  
 c(  
 "gender",  
 "status",  
 "processor",  
 "microboard",  
 "desctop\_os",  
 "mobile\_os",  
 "editor\_theme",  
 "cycle\_recursion",  
 "cycle",  
 "java\_kotlin",  
 "zero\_division",  
 "indexing",  
 "typing",  
 "slow\_python",  
 "list\_mutable",  
 "sugar",  
 "list\_expressions",  
 "ternar\_module",  
 "patterns",  
 "mobile\_desctop",  
 "web",  
 "back\_front\_end",  
 "flask\_django",  
 "languages\_number",  
 "editors\_number",  
 "future\_number",  
 "humour",  
 "other\_opinion",  
 "sugar\_using",  
 "python\_discontent",  
 "middle\_answers",  
 "dont\_know",  
 "web\_using",  
 "apple"  
 )  
df\_struct$complex\_vars <- # список сгруппированных переменных  
 list(  
 languages = c(  
 "python",  
 "cpp",  
 "javascript",  
 "pascal",  
 "csharp",  
 "java",  
 "c",  
 "php",  
 "kotlin",  
 "lua",  
 "scratch",  
 "basic",  
 "go",  
 "ruby",  
 "fasm",  
 "bf",  
 "haskel"  
 ),  
 editors = c(  
 "pycharm",  
 "vscode",  
 "idle",  
 "notepad",  
 "notepadpp",  
 "wing",  
 "sublime",  
 "jupiter",  
 "atom",  
 "console"  
 ),  
 futures = c(  
 "machine\_learning",  
 "big\_data",  
 "metaprog",  
 "quantum",  
 "cryptography",  
 "math"  
 )  
 )  
result <- list() # список для хранения промежуточных и итоговых результатов  
  
opros <- factorise(opros) # факторизация всех сторковых столбцов  
# str(opros)  
  
## Стадия 1. Простая предобработка и гипотезы  
  
# Simple tables  
result$simple\_tables <- apply(opros, 2, function(x)   
 round(prop.table(table(x)), digits = 3))  
  
# Simple Fisher Test  
result$simple\_fisher\_test <- apply(opros, 2, function(x)  
 chisq.test(table(x)))  
  
# Shapiro Test  
result$shapiro\_test <- lapply(opros[df\_struct$numeric\_vars],   
 function(x) shapiro.test(x))  
  
## Стадия 2. Двойные таблицы и взаимосвязи  
  
# Double tables  
result$double\_tables <- lapply(opros, function(x)  
 lapply(opros, function(y) round(prop.table(table(x, y)), digits = 3)))  
  
# Check equals  
# sapply(opros, function(x)   
# sum(sapply(opros, function(y) all(as.vector(x) == as.vector(y)))))  
# all rigth  
  
# Double Fisher Test  
result$double\_fisher\_test <- lapply(opros[df\_struct$quality\_vars], function(x)   
 lapply(opros[df\_struct$quality\_vars][sapply(opros[df\_struct$quality\_vars], function(z) !all(as.vector(x) == as.vector(z)))],   
 function(y) fisher.test(table(x, y), simulate.p.value = T)))  
  
# Cor test Kendall  
result$cor\_test\_kendal <- lapply(opros[df\_struct$numeric\_vars], function(x)   
 lapply(opros[df\_struct$numeric\_vars]  
 [sapply(opros[df\_struct$numeric\_vars], function(z) !all(as.vector(x) == as.vector(z)))],  
 function(y) cor.test(x, y, method = "kendall")))  
  
# Kruskal test  
result$kruskal\_test <- lapply(opros[df\_struct$numeric\_vars], function(x)  
 lapply(opros[df\_struct$quality\_vars], function(y)  
 kruskal.test(x, y)))  
  
## Исследование взаимосвязей  
  
# Researching edges  
get\_edges <- function(data, level) {  
 df <- data.frame(x = c(), y = c(), p = c())  
 lapply(names(data), function(x)  
 lapply(names(data[[x]]), function(y)   
 {   
 if (data[[x]][[y]]$p.value < level)   
 if (!(paste(x, y) %in% c(apply(df, 1, function(x) paste(x["x"], x["y"])),  
 apply(df, 1, function(x) paste(x["y"], x["x"])))))  
 df <<- rbind(df, data.frame(x = x, y = y, p = data[[x]][[y]]$p.value))   
 }  
 ))  
 df <- df[df$p < level,]  
 df$p\_log <- abs(round(log(df$p, base = 10)))  
 df  
}  
  
# Removing obvious edges in Kruskal test  
remove\_obvious\_edges <- function(data) {  
 obvious\_edges <- list(  
 humour = c("gender", "status", "editor\_theme"),  
 other\_opinion = c("zero\_division", "indexing", "typing"),  
 python\_discontent = c("slow\_python", "list\_mutable", "sugar"),  
 sugar\_using = c("list\_expressions", "ternar\_module", "patterns"),  
 apple = c("mobile\_os", "desctop\_os"),  
 dont\_know = c("microboard", "list\_expressions", "ternar\_module", "patterns", "flask\_django"),  
 web\_using = c("web", "flask\_django"),  
 middle\_answers = c("processor", "desctop\_os", "mobile\_os", "cycle\_recursion",   
 "cycle", "java\_kotlin", "slow\_python", "list\_mutable", "sugar",  
 "mobile\_desctop", "flask\_django"),  
 languages\_number = c("python", "cpp",  
 "javascript", "pascal", "csharp", "java", "c",  
 "php", "kotlin", "lua", "scratch", "basic",  
 "go", "ruby", "fasm", "bf", "haskel"),  
 editors\_number = c("pycharm", "vscode", "idle", "notepad",  
 "notepadpp", "wing", "sublime", "jupiter",  
 "atom", "console"),  
 future\_number = c("machine\_learning", "big\_data", "metaprog",  
 "quantum", "cryptography", "math")  
 )  
 answer <- data.frame(x = c(), y = c(), p = c(), p\_log = c())  
 apply(result$important\_edges$kruskal\_test, 1, function(x) {  
 if (!(x[["x"]] %in% names(obvious\_edges) &   
 any(sapply(unlist(obvious\_edges[as.vector(x[["x"]])]), function(z) as.vector(x[["y"]]) == z)))) {  
 answer <<- rbind(answer, data.frame(x = c(x["x"]), y = c(x["y"]),   
 p = c(x["p"]), p\_log = c(x["p\_log"])))  
 }  
 })  
 row.names(answer) <- NULL  
 answer  
}  
  
get\_all\_edges <- function(level) {  
 result$important\_edges$cor\_test <<- get\_edges(result$cor\_test\_kendal, level)  
 result$important\_edges$fisher\_test <<- get\_edges(result$double\_fisher\_test, level)  
 result$important\_edges$kruskal\_test <<- get\_edges(result$kruskal\_test, level)  
 result$important\_edges$kruskal\_test <<- remove\_obvious\_edges(result$important\_edges$kruskal\_test)  
   
 # Uniting edges  
 result$important\_edges$all <<- rbind(result$important\_edges$fisher\_test, result$important\_edges$cor\_test,  
 result$important\_edges$kruskal\_test)  
 result$important\_edges$all$x <<- as.character(result$important\_edges$all$x)  
 result$important\_edges$all$y <<- as.character(result$important\_edges$all$y)  
 result$important\_edges$all$p <<- as.numeric(result$important\_edges$all$p)  
 result$important\_edges$all$p\_log <<- as.numeric(result$important\_edges$all$p\_log)  
 result$important\_edges$all  
}  
result$important\_edges$all <- get\_all\_edges(0.05)  
  
find\_opros\_col <- function(x) {  
 names(opros)[sapply(names(opros), function(y) all(as.vector(opros[[y]]) == as.vector(x)))]  
}  
  
# str(result$important\_edges$all)  
result$edged\_vars <- lapply(opros, function(x) {  
 x <- find\_opros\_col(x)  
 r <- apply(result$important\_edges$all[  
 apply(result$important\_edges$all, 1, function(y)  
 y["x"] == x | y["y"] == x),], 1, function(y) {  
 ifelse(y["x"] == x, y["y"], y["x"])  
 })  
 names(r) <- NULL  
 r})  
# result$edged\_vars  
  
result$edged\_vars\_list <- data.frame(var = df\_struct$vars,   
 edged\_vars = sapply(opros, function(x) {  
 x <- find\_opros\_col(x)  
 paste(result$edged\_vars[[x]], collapse = " ")  
 }))  
  
var\_labels <- c(gender = "Пол",  
 status = "Социальный\nстатус",  
 languages\_number = "Количество используемых языков",  
 editors\_number = "Количество используемых редакторов",   
 future\_number = "Количество выбранных перспективных отраслей",  
 humour = "Коэффциент юмора",  
 other\_opinion = "Коэффициент другого мнения",  
 python\_discontent = "Коэффициент недовольства питоном",  
 sugar\_using = "Коэффициент использования синтаксического сахара",  
 middle\_answers = "Коэффициент лояльности ответов",  
 dont\_know = "Коэффициент незнания",  
 web\_using = "Коэффициент использования web'а",   
 apple = "Коэффициент лояльности к Apple",  
 processor = "Выбор процессора",  
 microboard = "Выбор микроплаты",  
 desctop\_os = "Выбор\nнастольной ОС",  
 mobile\_os = "Выбор\nмобильной ОС",  
 editor\_theme = "Выбор\nтемы редактора",  
 cycle\_recursion = "Цикл\nили\nрекурсия",  
 cycle = "Выбор\nвида цикла",  
 java\_kotlin = "Java\nили\nKotlin",  
 zero\_division = "Мнение\nо делении\nна ноль",  
 indexing = "Выбор индексации",  
 typing = "Сколько будет\n1 + \"а\"",   
 slow\_python = "Мнение\nо скорости\nпитона",  
 list\_mutable = "Отношение\nк изменяемости\nсписков в питоне",  
 sugar = "Мнение\nо \"сахарности\"\nпитона",  
 list\_expressions = "Использование\nсписочных\nвыражений",  
 ternar\_module = "Использование\nтернарного\nмодуля",  
 patterns = "Использование\nшаблонов\nпроектирования",  
 mobile\_desctop = "Мобильная\nразработка\nили\nдесктоп",  
 web = "Необходимость\nвеба",   
 back\_front\_end = "Предпочитаемая\nотрасль\nвеба",  
 flask\_django = "Flask\nили\nDjango",  
 python = "Использование\nPython",  
 cpp = "Использование\nC++",   
 javascript = "Использование\nJavaScript",  
 pascal = "Использование\nPascal",  
 csharp = "Использование\nC#",  
 java = "Использование\nJava",  
 c = "Использование\nC",  
 php = "Использование\nPHP",  
 kotlin = "Использование\nKotlin",  
 lua = "Использование\nLua",   
 scratch = "Использование\nScratch",  
 basic = "Использование\nBasic",  
 go = "Использование\nGo",  
 ruby = "Использование\nRuby",  
 fasm = "Использование\nFASM",  
 bf = "Использование\nBrainfuck",  
 haskel = "Использование\nHaskel",  
 pycharm = "Использование\nPyCharm",  
 vscode = "Использование\nVisual Studio\nCode",  
 idle = "Использование\nIDLE",  
 notepad = "Использование\nБлокнота",  
 notepadpp = "Использование\nNotepad++",  
 wing = "Использование\nWing",  
 sublime = "Использование\nSublime Text",  
 jupiter = "Использование\nJupiter\nNotebook",  
 atom = "Использование\nAtom",   
 console = "Использование\nконсоли",  
 machine\_learning = "Ожидание\nразвития\nмашинного\nобучения",  
 big\_data = "Ожидание\nразвития\nбольших\nданных",  
 metaprog = "Ожидание\nразвития\nметапрограммирования",   
 quantum = "Ожидание\nразвития\nквантовой\nлогики",  
 cryptography = "Ожидание\nразвития\nкриптографии",  
 math = "Ожидание\nразвития\nприкладной\nматематики")  
spaced\_var\_labels <- sapply(var\_labels, function(x) gsub("\n", " ", x))  
full\_var\_labels <- c(  
 python = "01. Python",  
 cpp = "02. C++",   
 javascript = "03. JavaScript",  
 pascal = "04. Pascal",  
 csharp = "05. C#",  
 java = "06. Java",  
 c = "07. C",  
 php = "08. PHP",  
 kotlin = "09. Kotlin",  
 lua = "10. Lua",   
 scratch = "11. Scratch",  
 basic = "12. Basic",  
 go = "13. Go",  
 ruby = "14. Ruby",  
 fasm = "15. FASM",  
 bf = "16. Brainfuck",  
 haskel = "17. Haskel",  
 pycharm = "01. PyCharm",  
 vscode = "02. Visual Studio\nCode",  
 idle = "03. IDLE",  
 notepad = "04. Блокнот",  
 notepadpp = "05. Notepad++",  
 wing = "06. Wing",  
 sublime = "07. Sublime\nText",  
 jupiter = "08. Jupiter\nNotebook",  
 atom = "09. Atom",   
 console = "10. Консоль",  
 machine\_learning = "01. Машинное\nобучение",  
 big\_data = "02. Большие\nданные",  
 metaprog = "03. Метапрограммирование",   
 quantum = "04. Квантовая\nлогика",  
 cryptography = "05. Криптография",  
 math = "06. Прикладная\nматематика"  
)  
var\_positive\_flags <- c(  
 languages = "Использует",  
 editors = "Использует",  
 futures = "Ожидает"  
)  
complex\_var\_labels <- c(  
 languages = "Популярность\nязыков\nпрограммирования",  
 editors = "Популярность\nредакторов\nкода",  
 futures = "Популярность\nвозможных\nпередовых\nотраслей"  
)  
spaced\_complex\_var\_labels <- sapply(complex\_var\_labels, function(x) gsub("\n", " ", x))  
  
## Отрисовка графиков  
  
library(ggplot2)  
library(extrafont)  
loadfonts(device = "win")  
  
theme\_pablo <- theme(panel.background = element\_rect(fill = "grey70"),   
 plot.background = element\_rect(fill = "grey60"),  
 legend.background = element\_rect(fill = "grey80"),  
 text = element\_text(family = "Comic Sans MS"),  
 axis.text.x = element\_text(angle = 30, hjust = 1))  
  
draw\_plot <- function(x, y) {  
 if (x %in% df\_struct$numeric\_vars & y %in% df\_struct$numeric\_vars) {  
 random\_colors <- colors()[sample(1:length(colors()), size = 2, replace = F)]  
 plt <- ggplot(opros, aes(x = opros[[x]], y = opros[[y]],   
 size = I(5), color = I(random\_colors[1])))+  
 geom\_smooth(method = "lm", se = F, color = I(random\_colors[2]), size = I(3))+  
 geom\_point()+  
 scale\_x\_continuous(breaks = seq(1:10))+  
 geom\_jitter()+  
 xlab(spaced\_var\_labels[x])+  
 ylab(spaced\_var\_labels[y])+  
 ggtitle(paste("Взаимосвязь переменных\n", spaced\_var\_labels[x], "и",  
 spaced\_var\_labels[y]))+  
 theme\_pablo  
 }  
 else if (x %in% df\_struct$numeric\_vars & y %in% df\_struct$quality\_vars) {  
 plt <- ggplot(opros, aes(x = opros[[x]], fill = opros[[y]], color = I("black")))+  
 geom\_histogram(alpha = 0.8, binwidth = 1)+  
 facet\_wrap(~ opros[[y]], nrow = 3)+  
 xlab(spaced\_var\_labels[x])+  
 ylab("Частота")+  
 ggtitle(paste("Взаимосвязь переменных\n", spaced\_var\_labels[x], "и",  
 spaced\_var\_labels[y]))+  
 scale\_fill\_brewer(name = var\_labels[y],  
 type = "qual", palette = sample(1:8, size=1))+  
 scale\_x\_continuous(breaks = seq(1:10))+  
 theme\_pablo  
 }  
 else if (x %in% df\_struct$quality\_vars & y %in% df\_struct$numeric\_vars) {  
 plt <- ggplot(opros, aes(x = opros[[y]], fill = opros[[x]], color = I("black")))+  
 geom\_histogram(alpha = 0.8, binwidth = 1)+  
 facet\_wrap(~ opros[[x]], nrow = 3)+  
 xlab(spaced\_var\_labels[y])+  
 ylab("Количество")+  
 ggtitle(paste("Взаимосвязь переменных\n", spaced\_var\_labels[x], "и",  
 spaced\_var\_labels[y]))+  
 scale\_fill\_brewer(name = var\_labels[x],  
 type = "qual", palette = sample(1:8, size=1))+  
 scale\_x\_continuous(breaks = seq(1:10))+  
 theme\_pablo  
 }  
 else if (x %in% df\_struct$quality\_vars & y %in% df\_struct$quality\_vars){  
 plt <- ggplot(opros, aes(x = opros[[x]], fill = opros[[y]], color = I("black")))+  
 geom\_histogram(stat = "count")+  
 facet\_wrap(~ opros[[y]])+  
 xlab(spaced\_var\_labels[x])+  
 ylab("Количество")+  
 ggtitle(paste("Взаимосвязь переменных\n", spaced\_var\_labels[x], "и",  
 spaced\_var\_labels[y]))+  
 scale\_fill\_brewer(name = var\_labels[y],  
 type = "qual", palette = sample(1:8, size=1))+  
 theme\_pablo  
 }  
 plt  
}  
draw\_simple\_plot <- function(x) {  
 if (x %in% df\_struct$quality\_vars) {  
 plt <- ggplot(opros, aes(x = opros[[x]], fill = opros[[x]], color = I("black")))+  
 geom\_histogram(stat = "count")+  
 xlab(spaced\_var\_labels[x])+  
 ylab("Количество")+  
 ggtitle(paste("Распределение переменой\n", spaced\_var\_labels[x]))+  
 scale\_fill\_brewer(name = var\_labels[x],  
 type = "qual", palette = sample(1:8, size=1))+  
 theme\_pablo  
 }  
 else if (x %in% df\_struct$numeric\_vars) {  
 random\_color <- colors()[sample(1:length(colors()), size=1)]  
 plt <- ggplot(opros, aes(x = opros[[x]], fill = I(random\_color), color = I("black")))+  
 geom\_histogram(stat = "count")+  
 xlab(spaced\_var\_labels[x])+  
 ylab("Частота")+  
 ggtitle(paste("Распределение переменной\n", spaced\_var\_labels[x]))+  
 scale\_fill\_brewer(name = gsub(" ", "\n", var\_labels[x]),  
 type = "qual", palette = sample(1:8, size=1),   
 guide="colourbar")+  
 scale\_x\_continuous(breaks = seq(1:10))+  
 theme\_pablo  
 }  
 plt  
}  
result$simple\_plots <- lapply(opros, function(x) {  
 x <- find\_opros\_col(x)  
 plt <- draw\_simple\_plot(x)  
 plt  
})  
  
draw\_complex\_plot <- function(x) {  
 kit <- c()  
 for (i in 1:length(unlist(df\_struct$complex\_vars[x]))) {  
 y <- unlist(df\_struct$complex\_vars[x])[i]  
 kit <- c(kit,   
 ifelse(opros[y] == var\_positive\_flags[x], full\_var\_labels[y], "none"))  
 }  
 kit <- data.frame(var = kit[kit != "none"])  
 kit$var <- factor(kit$var)  
 plt <- ggplot(kit, aes(x = var, fill = var, col = I("black")))+  
 geom\_histogram(stat = "count")+  
 ggtitle(spaced\_complex\_var\_labels[x])+  
 xlab(spaced\_complex\_var\_labels[x])+  
 ylab("Частота")+  
 scale\_fill\_brewer(name = "Частота",  
 type = "qual", palette = 3)+  
 theme\_pablo  
 plt  
}  
  
result$complex\_plots <- list()  
result$complex\_plots$languages <- draw\_complex\_plot("languages")  
result$complex\_plots$editors <- draw\_complex\_plot("editors")  
result$complex\_plots$futures <- draw\_complex\_plot("futures")  
  
# gsub(" ", "\_", gsub("\n ", "-", draw\_plot("gender", "status")$labels$title))  
# result$edged\_vars  
df <- data.frame(x = c(), y = c())  
result$important\_plots <- lapply(opros, function(x) {  
 x <- find\_opros\_col(x)  
 lapply(opros[unlist(result$edged\_vars[x])], function(y) {  
 y <- find\_opros\_col(y)  
 plt <- draw\_plot(x, y)  
# if (!(paste(x, y) %in% c(apply(df, 1, function(x) paste(x["x"], x["y"])),  
# apply(df, 1, function(x) paste(x["y"], x["x"]))))) {  
# df <<- rbind(df, data.frame(x = x, y = y))  
# i <- gsub("\"", "", gsub(" ", "\_", gsub("\n ", "-", plt$labels$title)))  
# ggsave(file = paste0(getwd(), "/plots/important\_plots/", i, ".png"),   
# plot = plt, device = "png")}  
 plt})})  
  
# Graph creating  
library(igraph)  
net <- list()  
net$edges <- data.frame(from = result$important\_edges$all$x,  
 to = result$important\_edges$all$y,  
 weight = result$important\_edges$all$p\_log)  
net$vertices <- data.frame(  
 id = colnames(opros),  
 type = ifelse(colnames(opros) %in% df\_struct$numeric\_vars, "numeric", "quality"))  
graph <- graph.data.frame(net$edges, net$vertices, directed = F)  
V(graph)$color <- ifelse(net$vertices$id %in% df\_struct$numeric\_vars, "orange", "skyblue")  
E(graph)$width <- as.numeric(as.vector(net$edges$weight))  
l <- layout.kamada.kawai(graph)  
# result$egdes\_graph <- plot(graph, layout= l, vertex.size = 15)

# Данные

## Ссылка

Используемые в опросе данные в предобработанном виде можно скачать по [ссылке](https://yadi.sk/i/WmRRsDzbnQ9H-g)

Для полноценной работы скрипта исследования следует скопировать файл данных в ту же директорию, что и файл *Research.Rmd*.

## Структура данных

Всего в данных содержится 67 признаков о 111 программистах

Быстро понять структуру данных можно, ознакомившись с [опросом](https://forms.yandex.ru/u/5d4336b07effdd03528c193c/)

Рядом с переменной курсивом приведено её название в данных

### Качественные переменные

#### Общие признаки

* Пол *gender*
  + Мужской
  + Женский
  + Кафельный (*пол не уточнён*)
* Социальный статус *status*
  + Школьник
  + Студент
  + Работяга (трудоустроенный программист)
  + Фрилансер (программист, работающий на заказ)
  + Аутсорсер (внештатный программист)
  + Пенсионер (*наличие пенсионеров в данных весьма странно - IT рынок России не настолько стар*)

#### Железо

* Предпочитаемая марка процессора *processor*
  + Intel
  + AMD
  + Всё равно
* Выбор микроплаты *microboard*
  + Arduino
  + Raspberry Pi
  + Не, не слышал

#### Программное обеспечение

* Предпочитаемая настольная ОС *desctop\_os*
  + Windows
  + Linux
  + MacOS
  + Лишь бы был комп
* Предпочитаемая мобильная ОС *mobile\_os*
  + Android
  + iOS
  + Лишь бы был телефон
* Тема редактора кода *editor\_theme*
  + Светлая
  + Тёмная
  + Не фильтрованная (*тест на выявление алкопрогеров*)

#### Общепрограммистские вопросы

* Цикл или рекурсия? *cycle\_recursion* (*варианты ответа очевидны, имеется средний ответ*)
* For или While? *cycle*
* Java или Kotlin? *java\_kotlin*
* Мнение о делении на ноль *zero\_division*
  + Ошибка
  + Так нельзя делать
  + Бесконечность
* Выбор индексации *indexing*
  + С нуля
  + С единицы
  + С произвольного числа
* Сколько будет 1 + “а”? *typing*
  + Ошибка типа
  + “1а”
  + 98 (то есть 1 + utf8 код символа “a”)

#### Вопросы о питоне

* Мнение о скорости питона *slow\_python*
  + Нормально
  + Терпимо
  + Побыстрее бы
* Отношение к изменяемости списков *list\_mutable*
  + Позитивное
  + Нейтральное
  + Негативное
* Мнение о “сахарности” питона *sugar*
  + Слишком много сахара
  + Достаточно
  + Можно было послаще

#### Вопросы о использовании синтаскического сахара

* Использование тернарного модуля *ternar\_module*
  + Не знал
  + Не использую
  + Редко
  + Часто
* Использование списочных выражений *list\_expressions* (*ответы такие же*)
* Использование шаблонов проетирования *patterns*
  + Не знал о них
  + В питоне неприменимы
  + Редко
  + Часто

#### Вопросы о сферах программирования

* Мобильная или десктоп разработка? *mobile\_desctop*
* Использование веба *web*
  + Нет
  + Как прилагающееся
  + Конечно
* Бек или фронт энд? *back\_front\_end*
  + Бек-энд
  + Фронт-энд
  + Фул-стек
* Flask или Django? *flask\_django*
  + Flask
  + Django
  + Не знаю
  + Оба потянут
  + Оба не тянут

#### Используемые языки

В опросе было рассмотрено 17 наиболее известных ЯП (Dart не выбрал никто, в данных его нет). Каждому языку в данных соответствует переменная с его названием:

*python*, *cpp*, *javascript*, *pascal*, *csharp*, *java*, *c*, *php*, *kotlin*, *lua*, *scratch*, *basic*, *go*, *ruby*, *fasm*, *bf*, *haskel*

В переменных хранятся значения “Использует” или “Не использует” в зависимости от ответа

#### Используемые редакторы

В опросе было рассмотрено 10 редакторов кода, предпочтительно для питона, включая консоль. Каждому редактору в данных соответствует переменная с его названием:

*pycharm*, *vscode*, *idle*, *notepad*, *notepadpp*, *wing*, *sublime*, *jupiter*, *atom*, *jupyter*

В переменных хранятся значения “Использует” или “Не использует” в зависимости от ответа

#### Перспективные направления

В качестве разгрузочного вопроса участникам опроса предлагалось ответить, какие из отраслей IT являются, по их мнению, перспективными. Предлагалось 6 отраслей: большие данные, машинное обучение, квантовая логика, криптография, метапрограммирование и прикладная математика:

*big\_data*, *machine\_learning*, *quantum*, *cryptography*, *metaprog*, *math*

В переменных хранятся значения “Ожидает” или “Не ожидает” в зависимости от ответа

### Количественные переменные

Самым большим инструментарием по обработке данных статистика имеет для количественных переменных. Однако, структура опроса была такова, что очевидными количественными переменными были всего 3. Ещё 8 были получены путём объединения качественных переменных.

#### По имеющимся данным

* Количество языков программирования, которое знает программист *languages\_number*
* Количество редакторов, используемых программистом *editors\_number*
* Количество перспективных отраслей, которое видит программист *future\_number*

#### Синтезированные из качественных

Эти переменные обладают относительной шкалой от 0 до 10.

* Чувство юмора (или маргинальность) *humour*
  + Ответы про пенсионера, кафельный пол и нефильтрованную тему повышают условный показатель “юмора”
* Другое мнение *other\_opinion*
  + Нестандартные ответы на вопросы об индексации, типизации и делении на ноль повышают условный показатель “другого мнения”
* Недовольство питоном *python\_discontent*
  + Негативные ответы о возможностях питона повышают условный показатель “недовольства питоном”
* Использование синтаксического сахара *suger\_using*
  + Использование тернарного модуля, списочных выражений и шаблонов проектирования повышают условный показатель “использования сахара”
* Лояльность ответов *middle\_answers*
  + Количество компромиссных ответов
* Коэффициент незнания *dont\_know*
  + Количество ответов в духе “Не знаю”
* Использование веба *web\_using*
  + Количество наиболее объемлющих ответов по тематике веба
* Лояльность к Apple *apple*
  + Предпочтение macOS и iOS прочим продуктам

Ответы на все вопросы анкеты были обязательны и в некоторых местах не имели подходящего ответа, в этом есть минус опроса. Зато, в данных нет ни одного пропущенного значения.

Кому нечего делать, может пересчитать количество упомянутых переменных :)

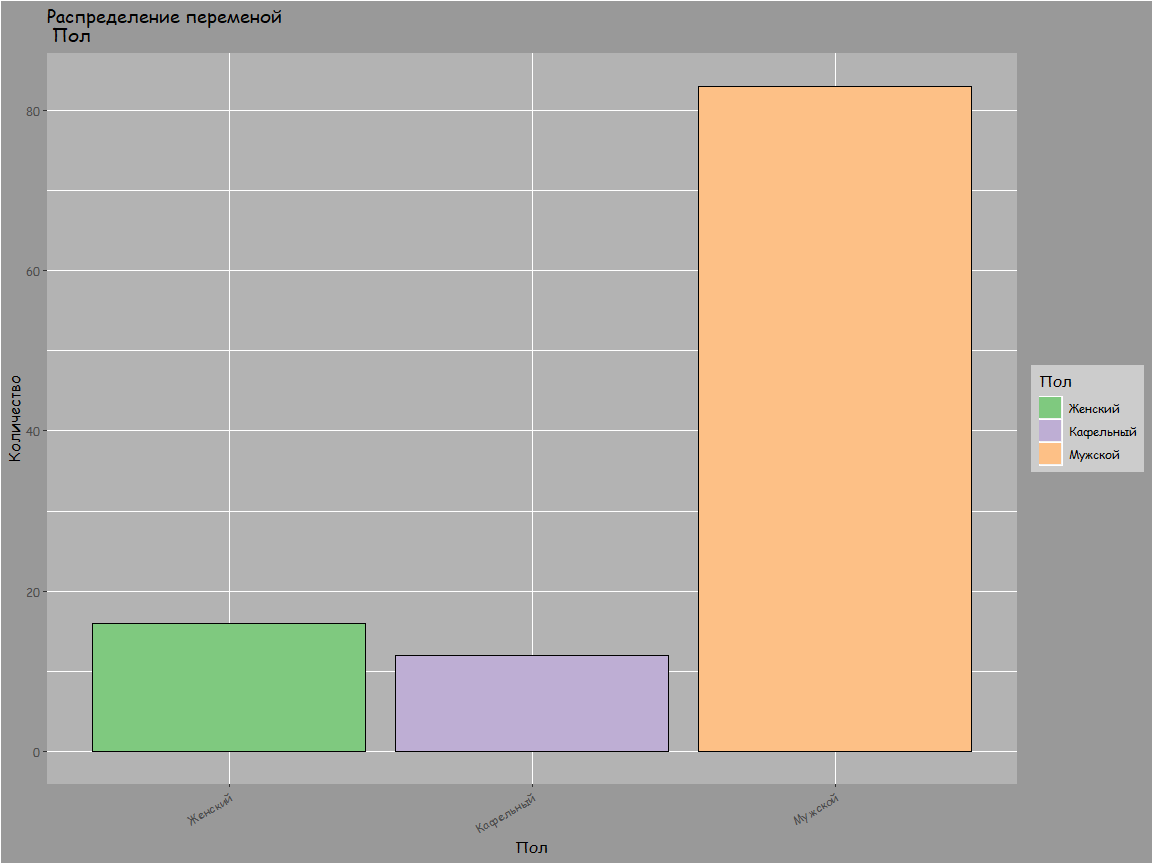
# Распределения ответов

В этом разделе представлены распределения всех переменных в графическом и табличном виде, к каждому из которых приложено возможное объяснение.

Не у всех переменных есть таблица распределения. Это сделано для того, чтобы уменьшить кровотечение из ваших глаз при лицезрении дробных величин в качестве групп разделения.

## Физическо-психологические характеристики опрошенных

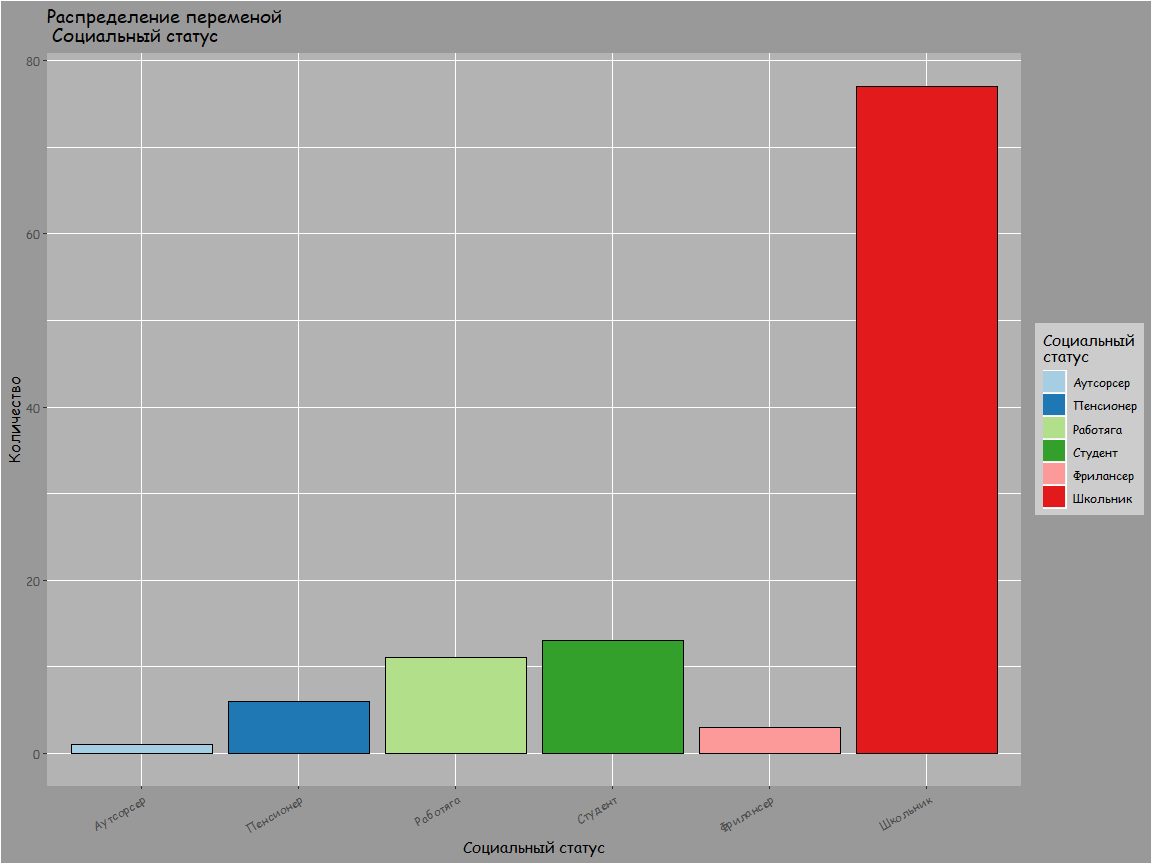
### Пол



|  |  |
| --- | --- |
| x | Freq |
| Женский | 0.144 |
| Кафельный | 0.108 |
| Мужской | 0.748 |
| *Выводы*: |  |

1. Статистика явно не на стороне феминисток. Доля женщин в исследованном сегменте ИТ статистически значимо меньше, чем доля мужчин
2. Зато представители меньшинств могут ликовать. Прогеры очень толерантны к ним.

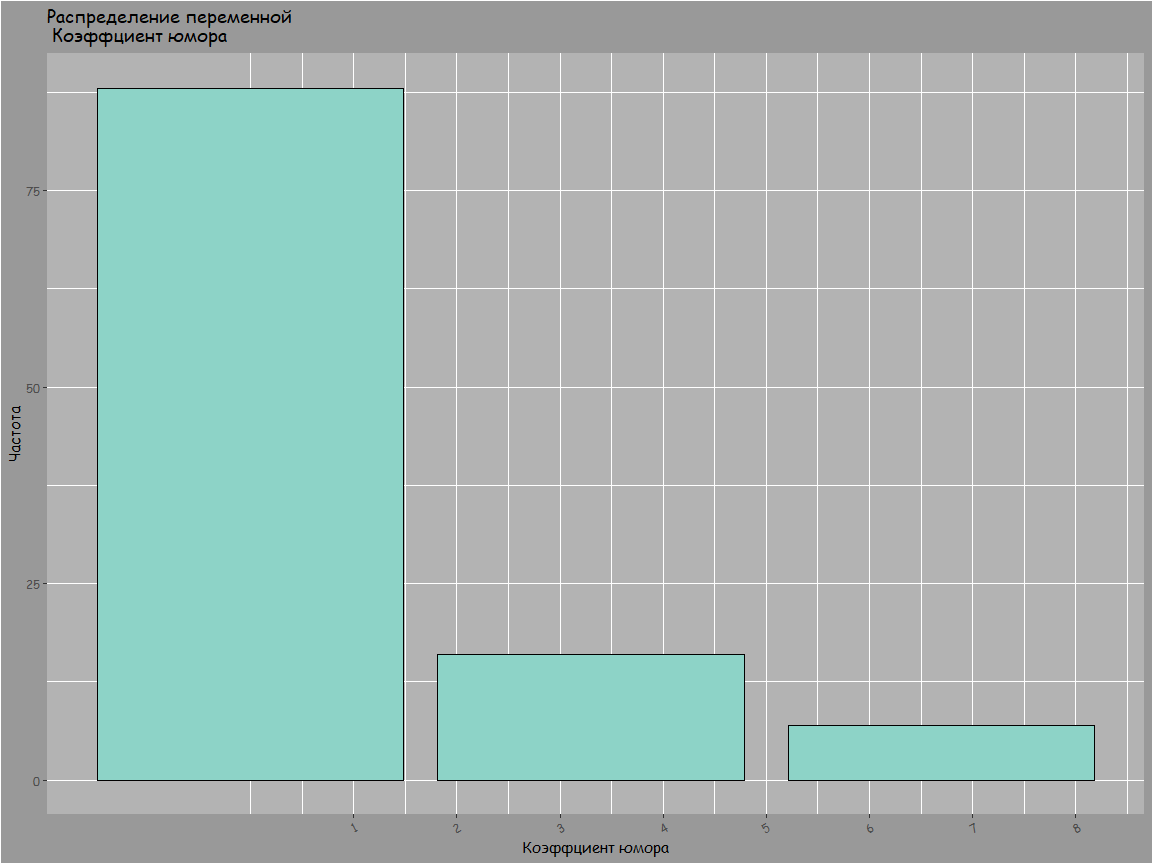
### Социальный статус



|  |  |
| --- | --- |
| x | Freq |
| Аутсорсер | 0.009 |
| Пенсионер | 0.054 |
| Работяга | 0.099 |
| Студент | 0.117 |
| Фрилансер | 0.027 |
| Школьник | 0.694 |
| *Выводы*: |  |

1. В данных прослеживается явная возрастная перекошенность в сторону молодых. Это свойство именно этой выборки, а не программистов в целом.
2. 1 человек знал, что такое “Аутсорсер”, либо это рандомный клик.
3. Несмотря на “молодость” выборки, суммарная доля студентов, работяг и пенсионеров близка к четверти. Автор не считает слишком мнительным ставить под сомнение статус пенсионера у некоторых опрошенных.

### Чувство юмора (или маргинальность)

 *Примечание*:

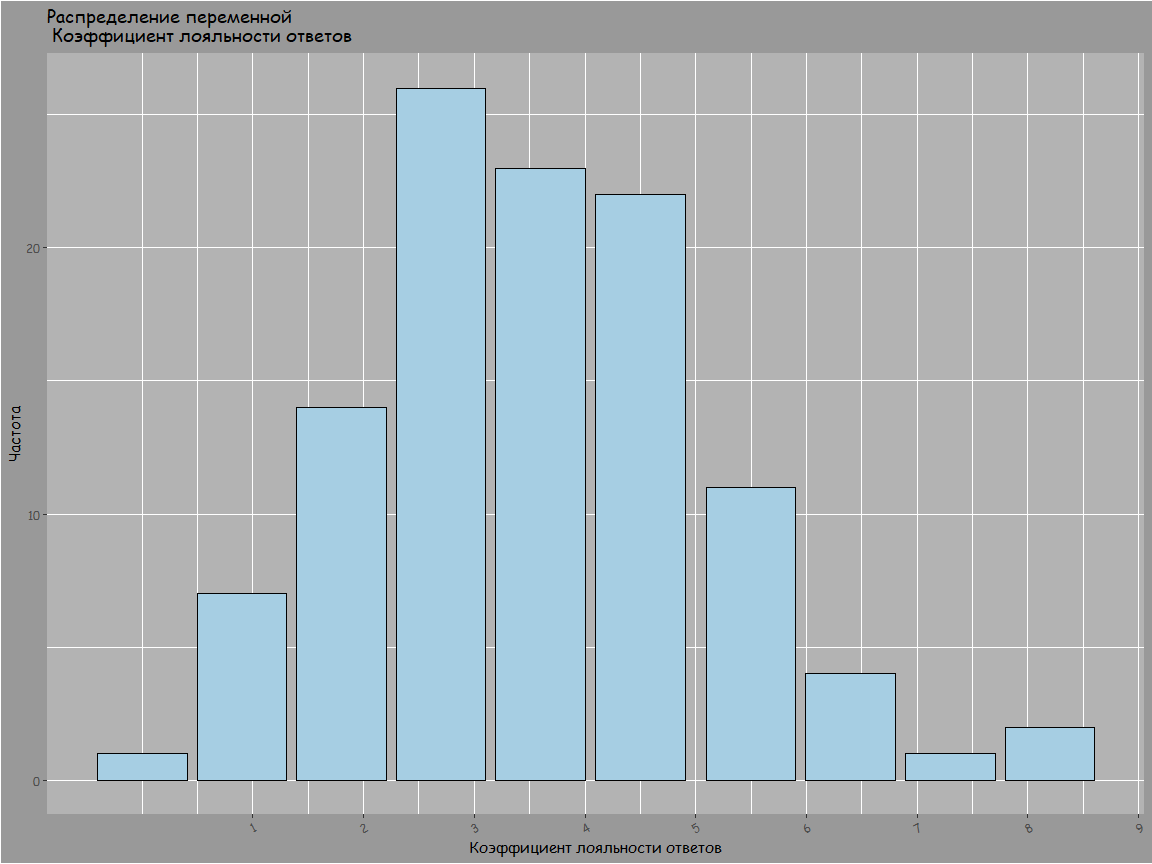
Неоднозначность названия критерия объясняется тем, что высший балл достался бы тому, чьи ответы были бы “кафельный пол”, “пенсионер” и “нефильтрованная тема”. Психология подсказывает три возможные причины таких ответов:

1. Человек обладает чувством юмора
2. Человек не хочет распростараняться о себе, т. е. он гик
3. Человек обладает больной психикой, т. е. он конченный маргинал

*Выводы*:

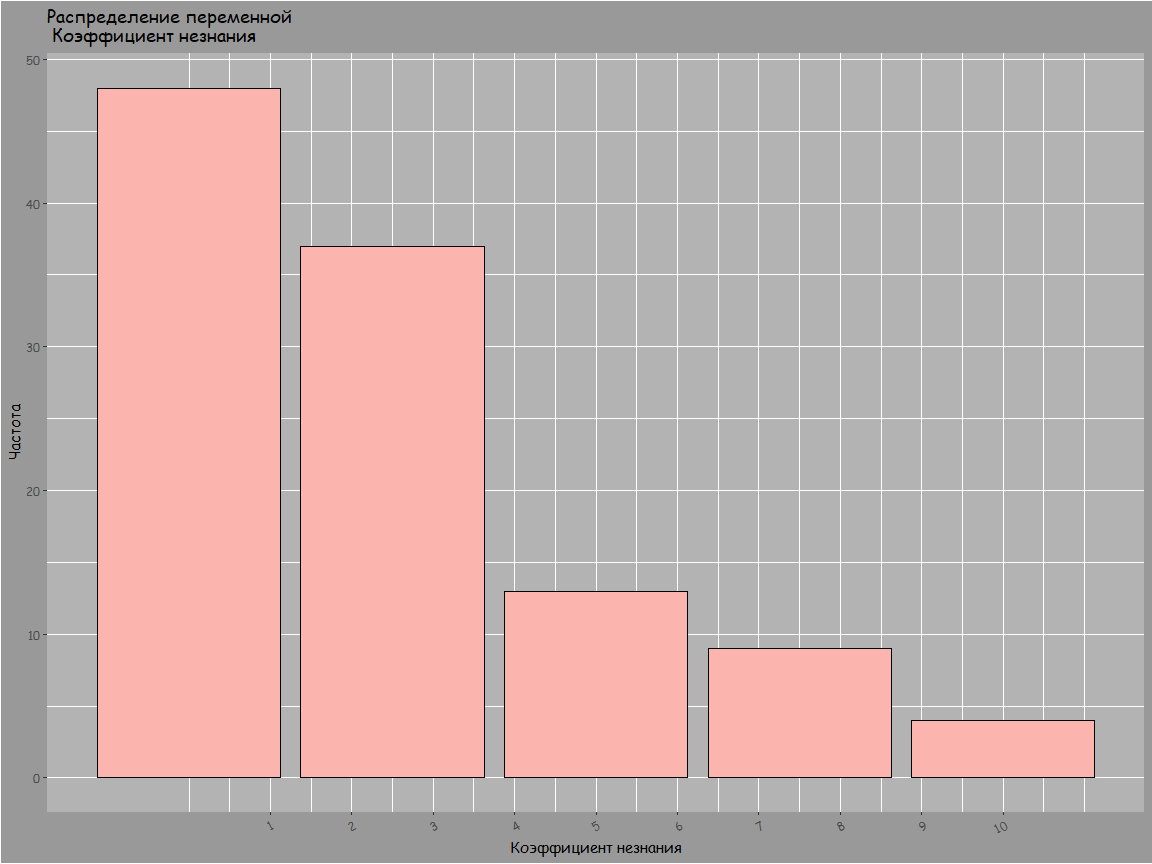
1. ~~Хорошо, что~~ В данных не попалось ни одного “пенсионера кафельного пола, предпочитающего нефильтрованную тему”. Это говорит о неокончательной маргинализации программистов.
2. Программисты очень ответственно относятся к заполнению самых абсурдных форм и не находят юмор всегда уместным.

### Лояльность ответов

 *Выводы*:

1. Большинство опрошенных дали среднее количество компромиссных ответов
2. Особо яростных и непримиримых идеалистов, оказалось немного
3. Около правой границы графика можно увидеть небольшое возмущение. Это мудрецы, проникшиеся вселенской гармонией, ищущие истину не в пределах, а в середине. Там также могли оказаться мямли и потенциальные политики.

### Коэффициент незнания

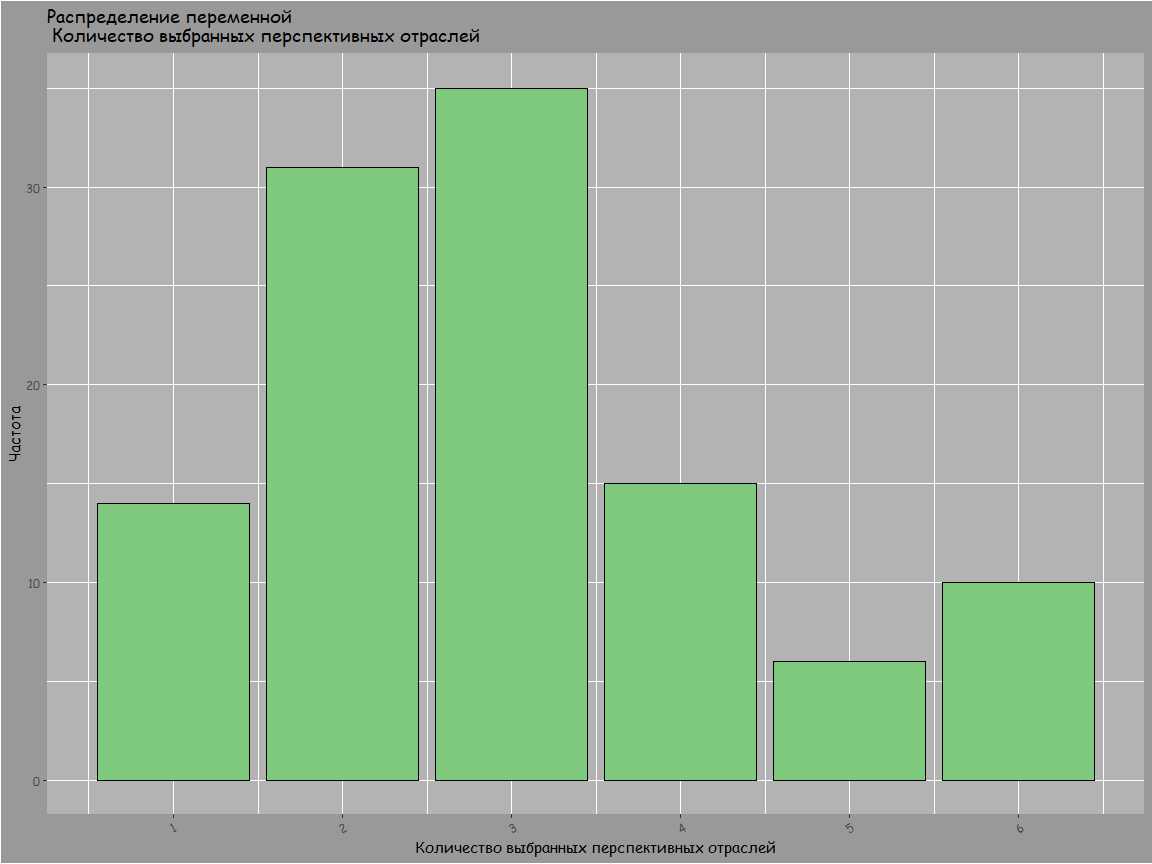
 *Примечание*:

Опрос не имел перед собой задачи оценить умственные способности программистов. В качестве понятия знания здесь взято количество ответов, которые могли бы дать самые рассудительные сотрудники ИТ.

*Выводы*:

1. Количество “полных профанов” в программировании не так уж велико
2. Хотя, возможна ситуцаия, что правильные ответы просто навязаны взглядом большинства

### Количество перспективных отраслей, которое видит программист

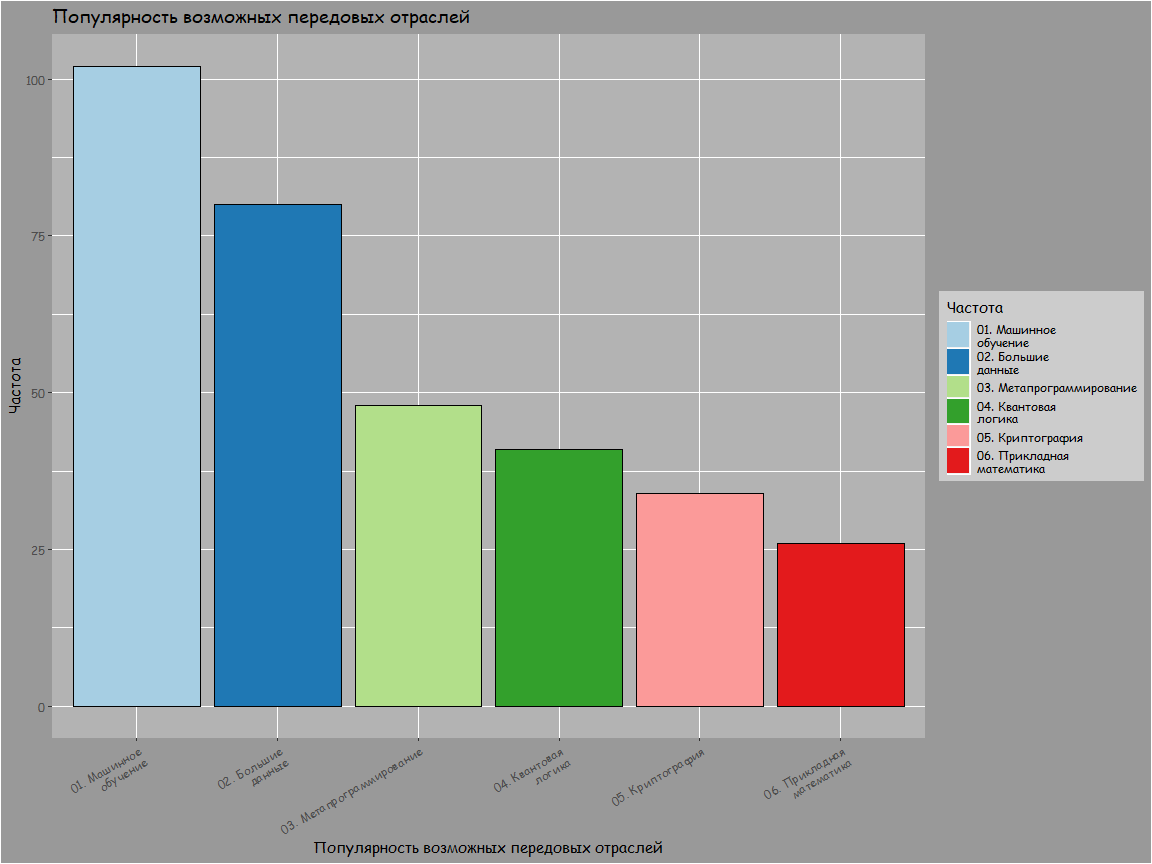


|  |  |
| --- | --- |
| x | Freq |
| 1 | 0.126 |
| 2 | 0.279 |
| 3 | 0.315 |
| 4 | 0.135 |
| 5 | 0.054 |
| 6 | 0.090 |

*Выводы*:

1. Часть испытуемых тупо выбрала всё, что не есть хорошо
2. Оптимальное число возможных перспективных отраслей - 3

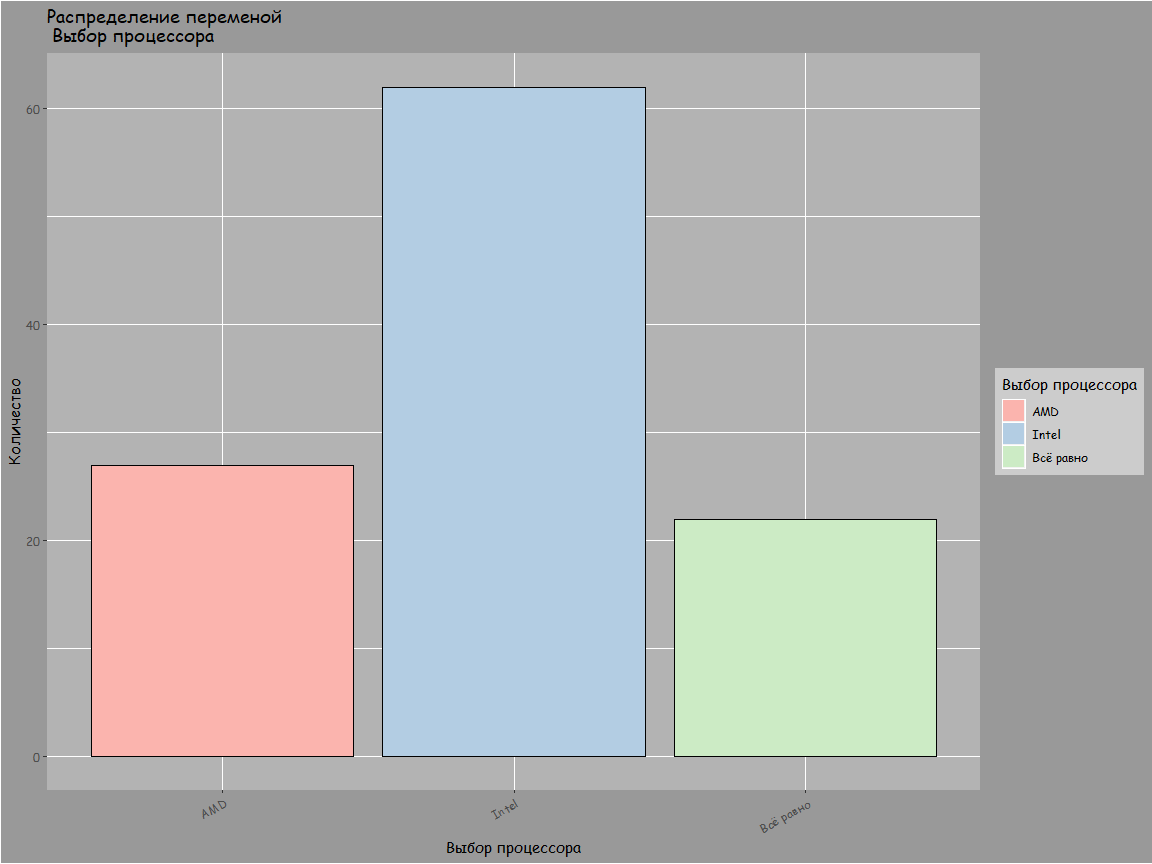
### Самые популярные возможные перспективные отрасли

 *Выводы*:

1. Абсолютное большинство горой стоит за то, что машинное обучение станет передовой отраслью. Страшным сном программистов мог бы стать искусственный интеллект, пишуший программы.
2. Немного меньший процент опрошенных считает Биг Дату перспективной отраслью
3. Метапрограммирвование, квантовая логика и криптография обладают мнением о перспективности меньше среднего.
4. Программисты не любят или не видят применение прикладной математики

## Железо

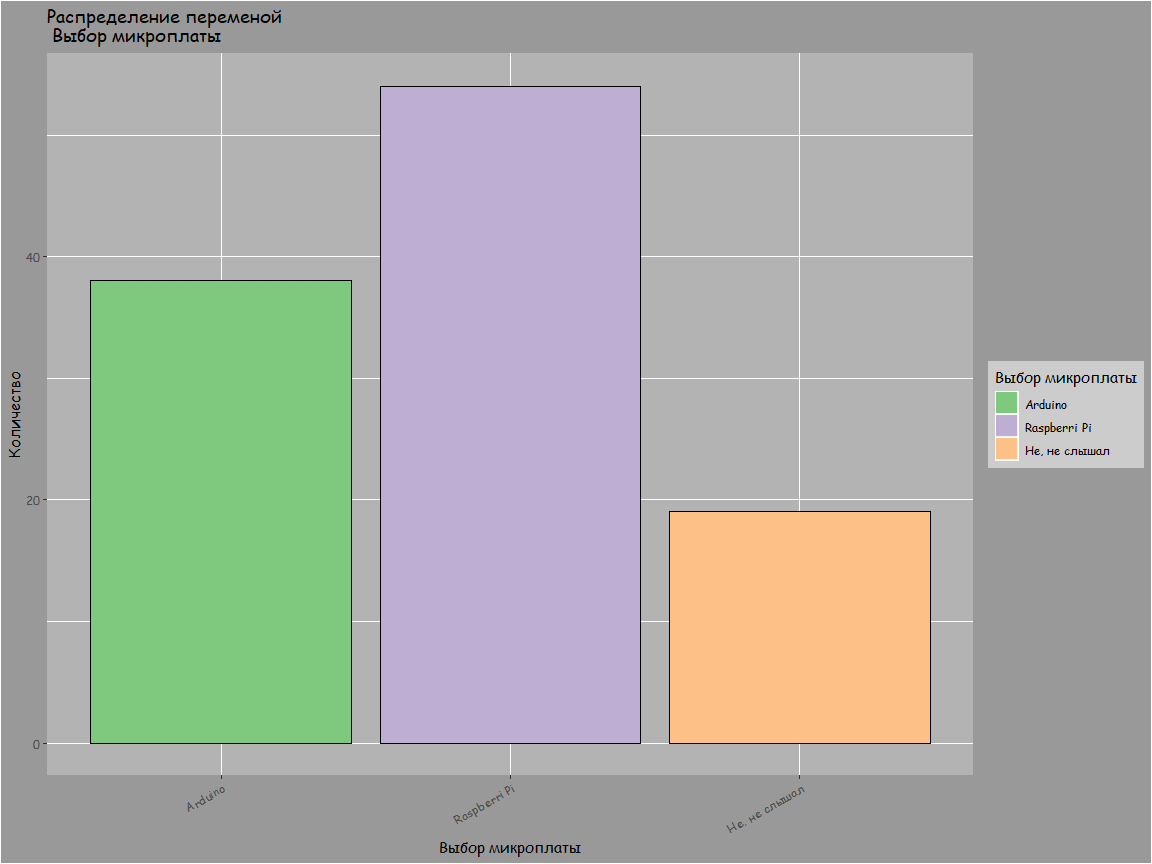
### Выбор процессора



|  |  |
| --- | --- |
| x | Freq |
| AMD | 0.243 |
| Intel | 0.559 |
| Всё равно | 0.198 |
| *Выводы*: |  |

1. Intel пользуется большей популярностью в массах, нежели AMD
2. Одна пятая, которой “всё равно”, либо является представителями буржуазии, у которых комп настолько прокачен, что тянет всё, либо это пожилые геймеры, которым хватает игры [Super Mario Bros](https://supermariobros.io), которая, как известно, не так требовательна к ресурсам ПК.

### Выбор микроплаты

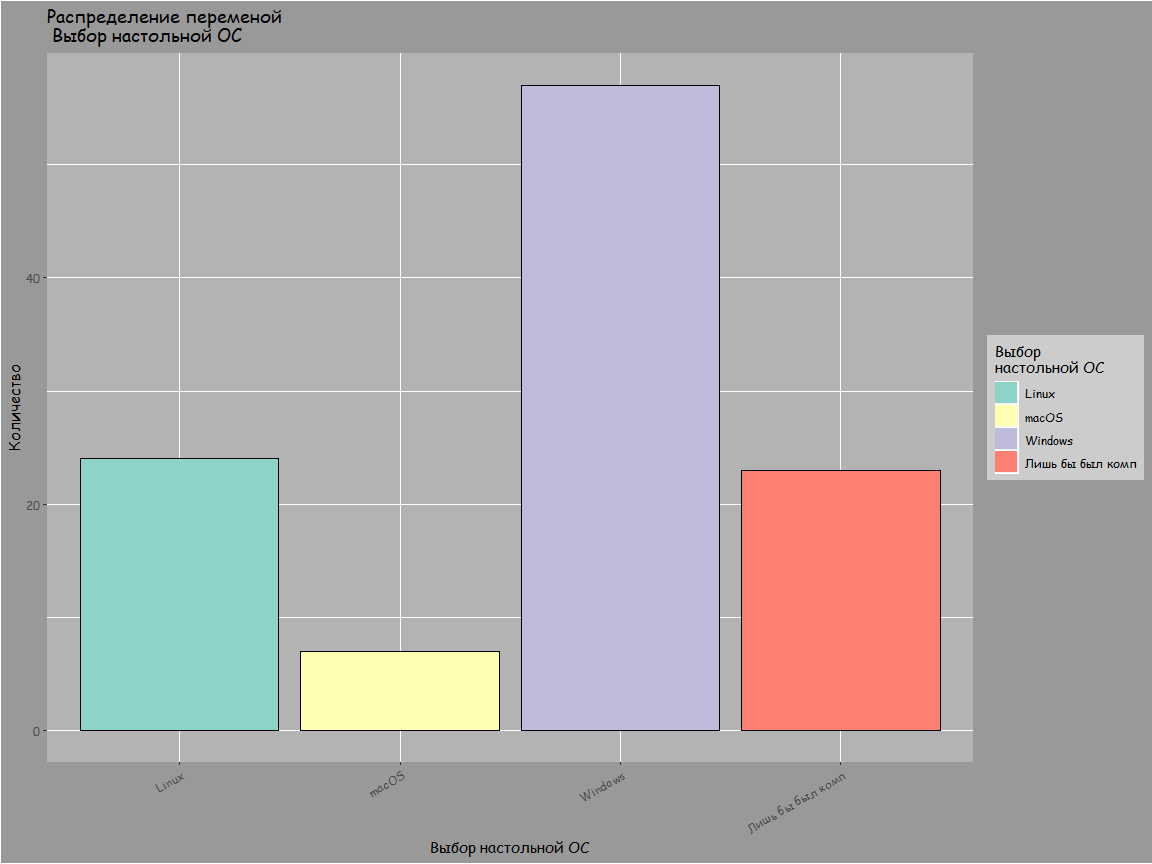


|  |  |
| --- | --- |
| x | Freq |
| Arduino | 0.342 |
| Raspberri Pi | 0.486 |
| Не, не слышал | 0.171 |
| *Выводы*: |  |

1. Отмечена высокая осведомлённость программистов в области “ручного железа”
2. Хотя сравнение микроконтроллера и одноплатного компьютера было не совсем логичным, оно стало интересным индикатором знаний программиста (*см. ниже*)

## Программное обеспечение

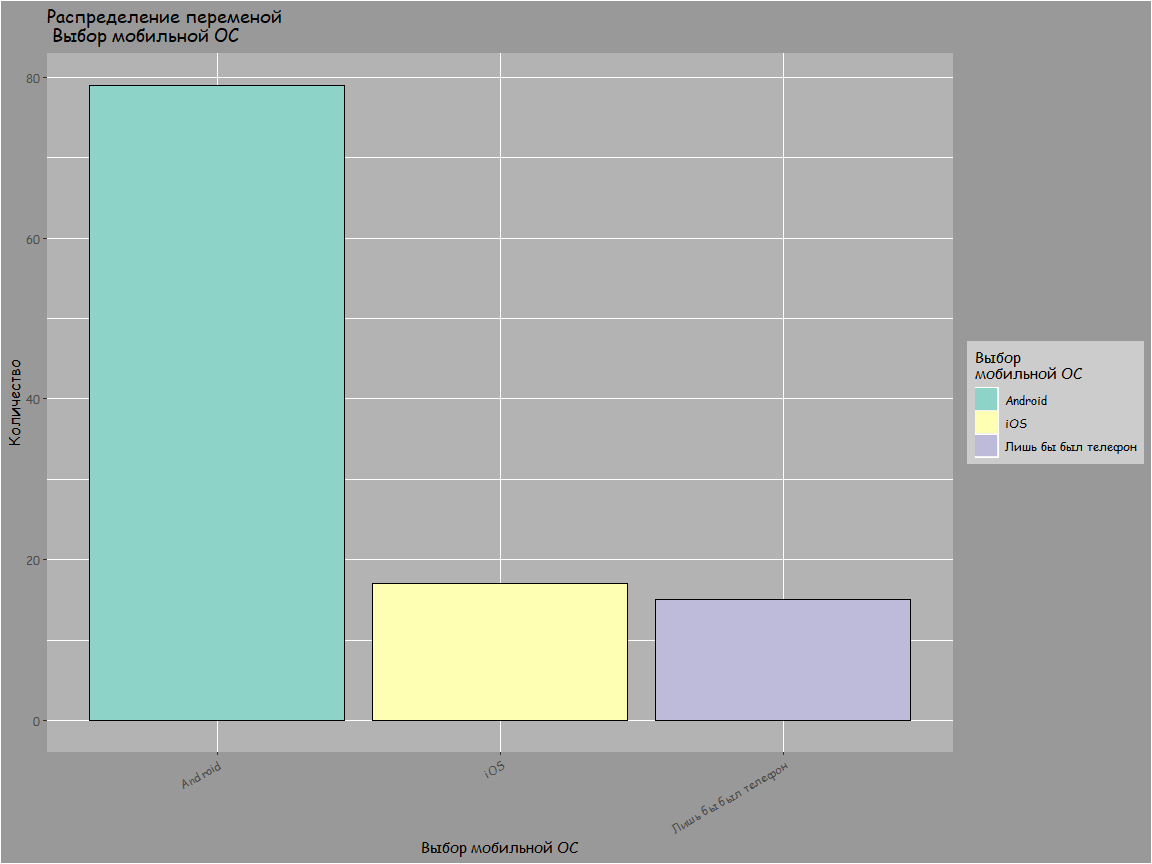
### Предпочитаемая настольная ОС



|  |  |
| --- | --- |
| x | Freq |
| Linux | 0.216 |
| macOS | 0.063 |
| Windows | 0.514 |
| Лишь бы был комп | 0.207 |
| *Выводы*: |  |

1. Окна являются наиболее популяной ОС среди программистов, хотя Linux с его армией дистрибутивов являются неплохим подспорьем майкрософтовской гегемонии.
2. Количество пользователей яблоко-компов допустимо невелико, однако миру приходится считаться с их существованием.

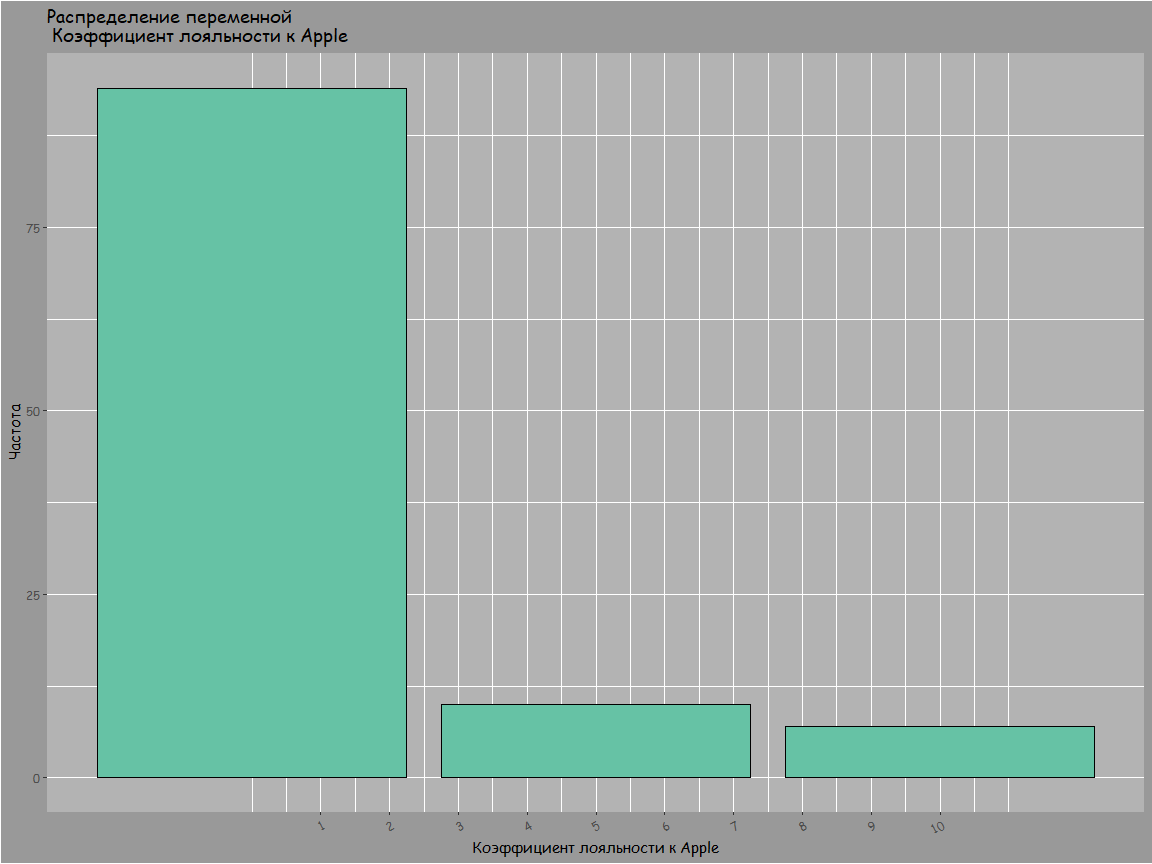
### Предпочитаемая мобильная ОС



|  |  |
| --- | --- |
| x | Freq |
| Android | 0.712 |
| iOS | 0.153 |
| Лишь бы был телефон | 0.135 |
| *Выводы*: |  |

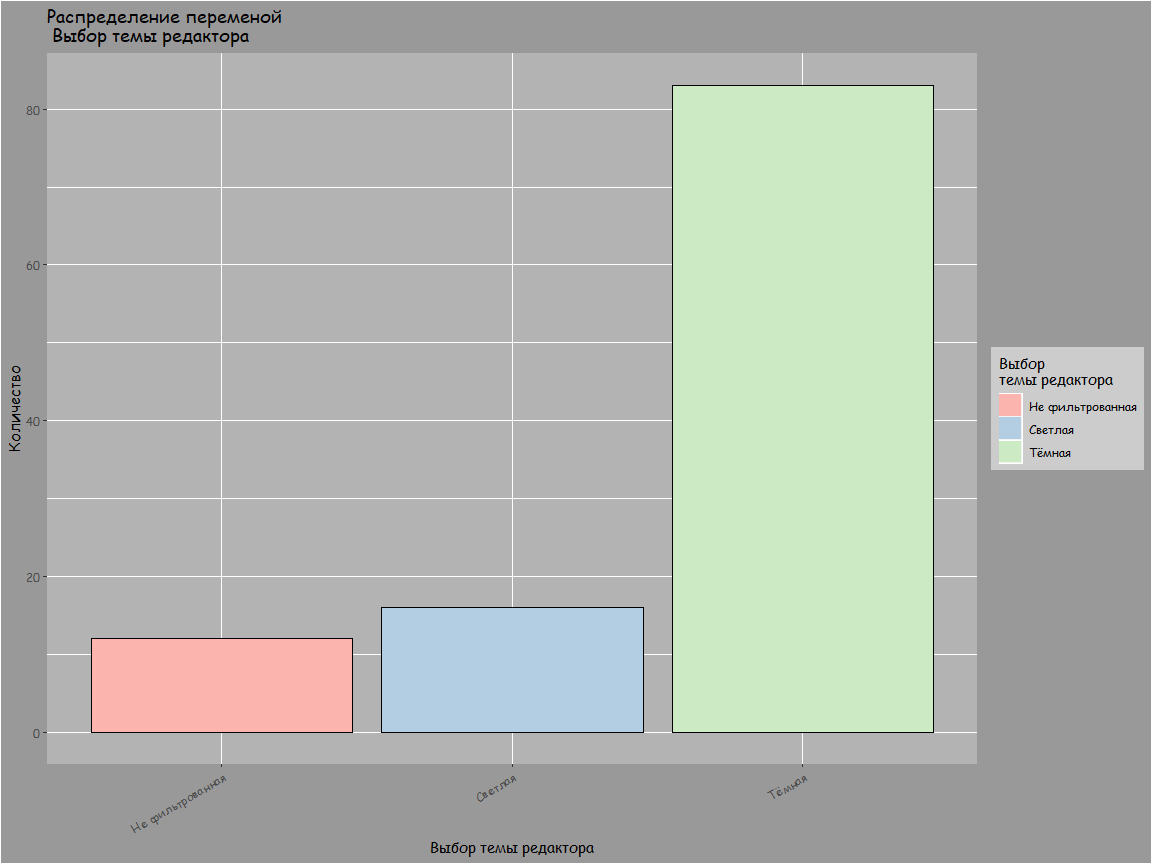
1. Доля предпочтения андроида указывает на успешность социалистической революции и процветание пролетариата. Доля ~~буржуев~~ пользователей Apple в мобильном секторе значительно меньше.

### Лояльность к Apple

 *Выводы*:

1. Не так уж и много людей, полностью одобряющих яблочную продукцию
2. ~~Процентное отношение нелояльных к эпплу программистов в приближении равно доле бедных людей в России.~~ На самом деле их (бедных) всего 12% :)
3. На основе этого распределния *далее* будет сделана интересная закономерность

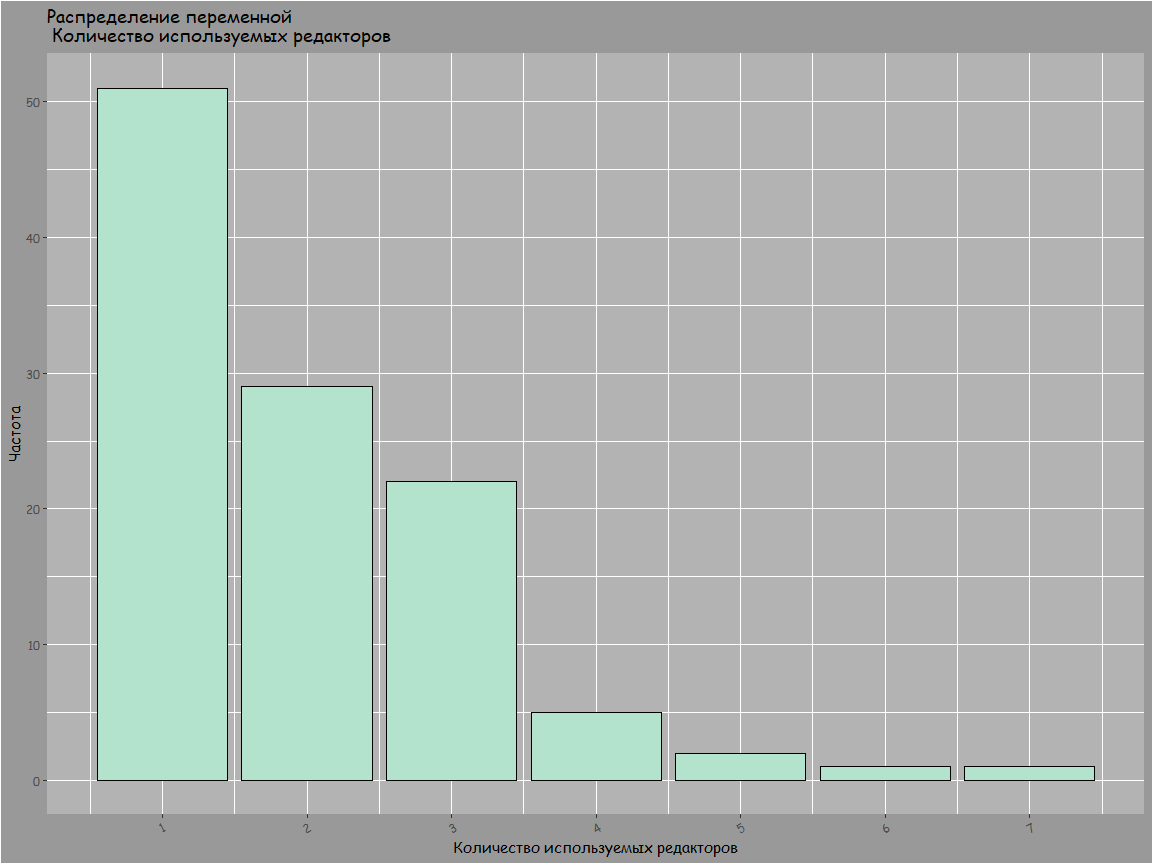
### Тема редактора кода



|  |  |
| --- | --- |
| x | Freq |
| Не фильтрованная | 0.108 |
| Светлая | 0.144 |
| Тёмная | 0.748 |
| *Выводы*: |  |

1. Программисты давно выбрали свою [сторону силы](https://ru.wikipedia.org/wiki/%D0%97%D0%B2%D1%91%D0%B7%D0%B4%D0%BD%D1%8B%D0%B5_%D0%B2%D0%BE%D0%B9%D0%BD%D1%8B).
2. Обнаружена массивная группа алконавтов - около 10% опрошенных.
3. Совет для тех, кто выбрал светлую тему: оглядывайтесь почаще.

### Количество редакторов, используемых программистом



|  |  |
| --- | --- |
| x | Freq |
| 1 | 0.459 |
| 2 | 0.261 |
| 3 | 0.198 |
| 4 | 0.045 |
| 5 | 0.018 |
| 6 | 0.009 |
| 7 | 0.009 |

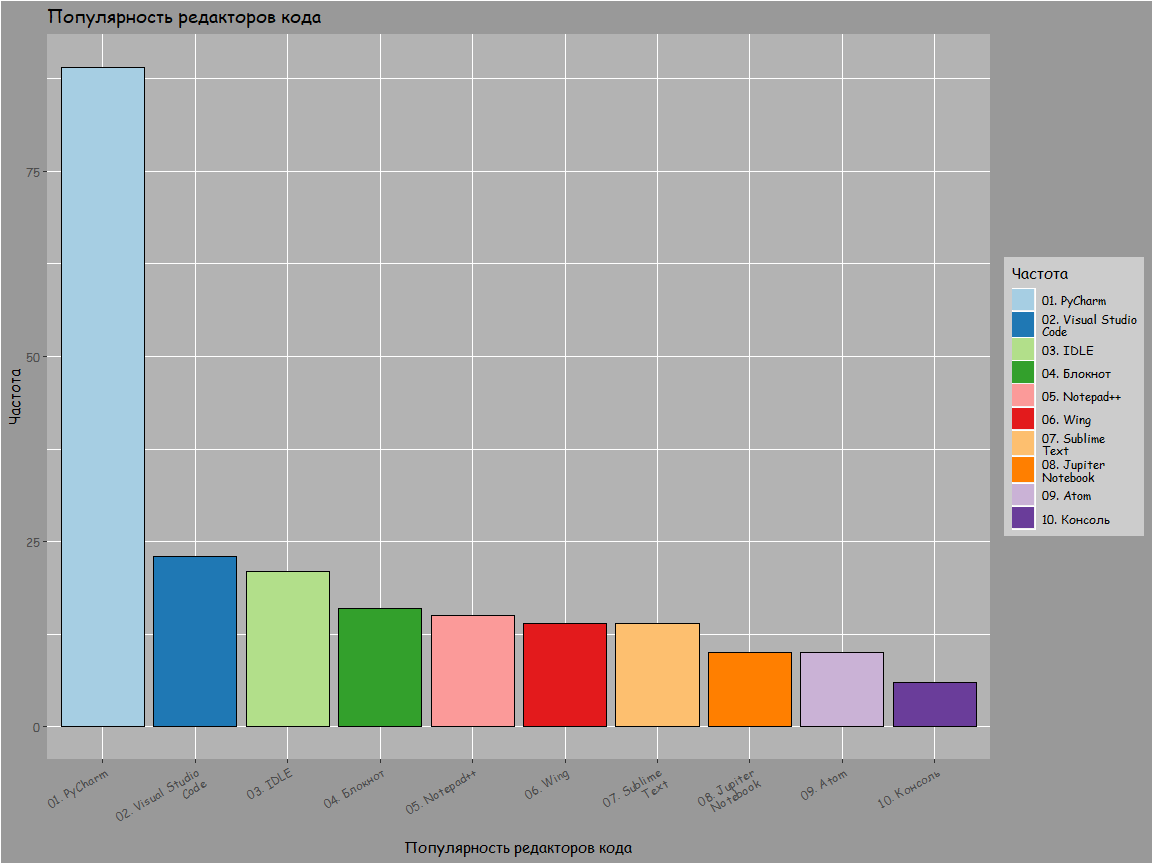
*Примечания*:

1. Следует учитывать, что опрос проводился среди питонистов, и редакторы предлагались для питона. Возможно, на самом деле, используемых редакторов могло быть больше.

*Выводы*:

1. Большинству для счастья много не надо
2. Все десять предложенных редакторов использовать никто не додумался

### Самые популярные редакторы



*Примечания*:

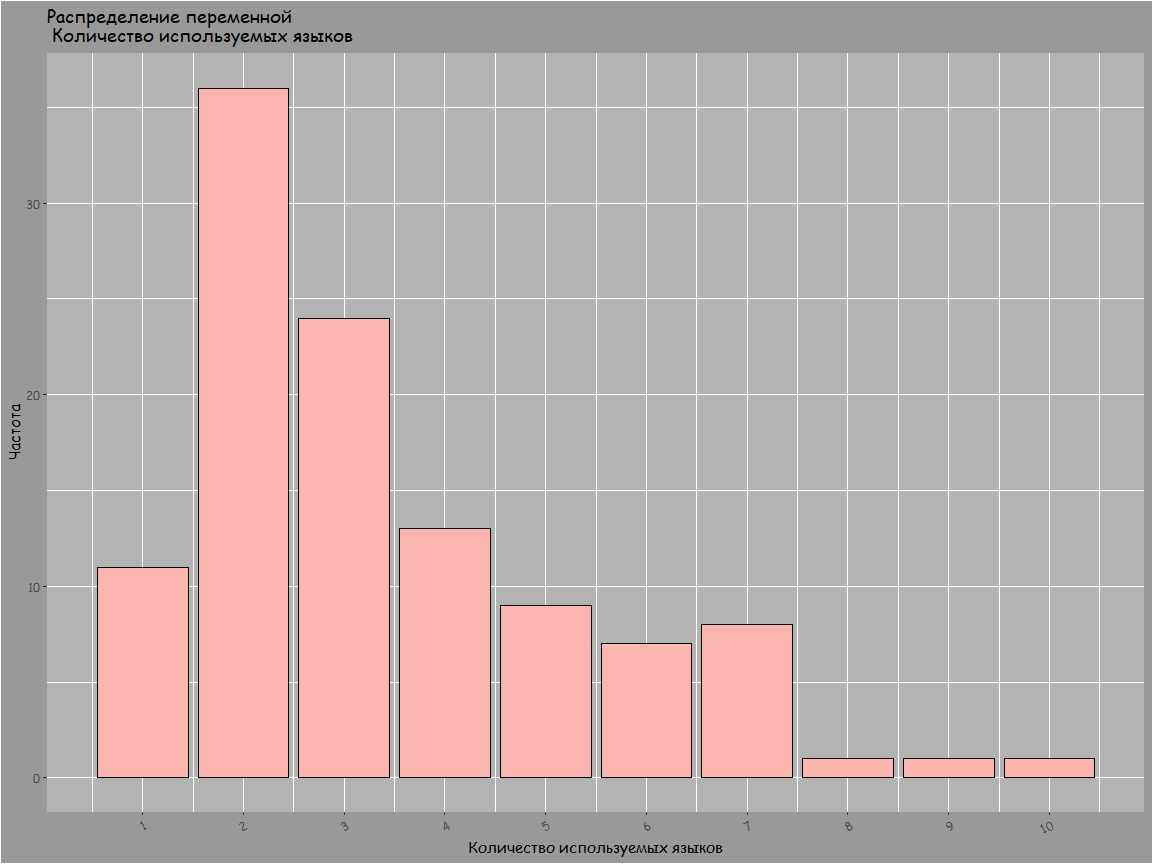
1. Следует учитывать, что опрос проводился среди питонистов, и редакторы предлагались для питона. Возможно, есть и другие популярные редакторы.

*Выводы*:

1. С кратным отрывом среди питонистов более популярным является PyCharm. Хвала компании JetBrains за качественное ПО, доступное не всегда по лицензионным ключам.
2. Wing, активно предлагаемый студентам Яндекс.Лицея на первом году обучения в общей массе не получил должного распространения.
3. IDLE, Блокнот и Notepad++, несмотря на небольшой фунционал (особенно у Блокнота) являются достаточно популярными.
4. Хоть VS Code является лишь редактором кода, а не средой разработки, он занял второе место.
5. Sublime Text, Jupyter Notebook и Atom являются достаточно мощными инструментами, но не находящими достаточного одобрения информационного пролетриата
6. Нашлись извращенцы, которые кодят в консоли. Хотелось бы посмотреть на то, как они пилят проекты.

## Общепрограммистские вопросы

### Количество языков программирования, которое знает программист

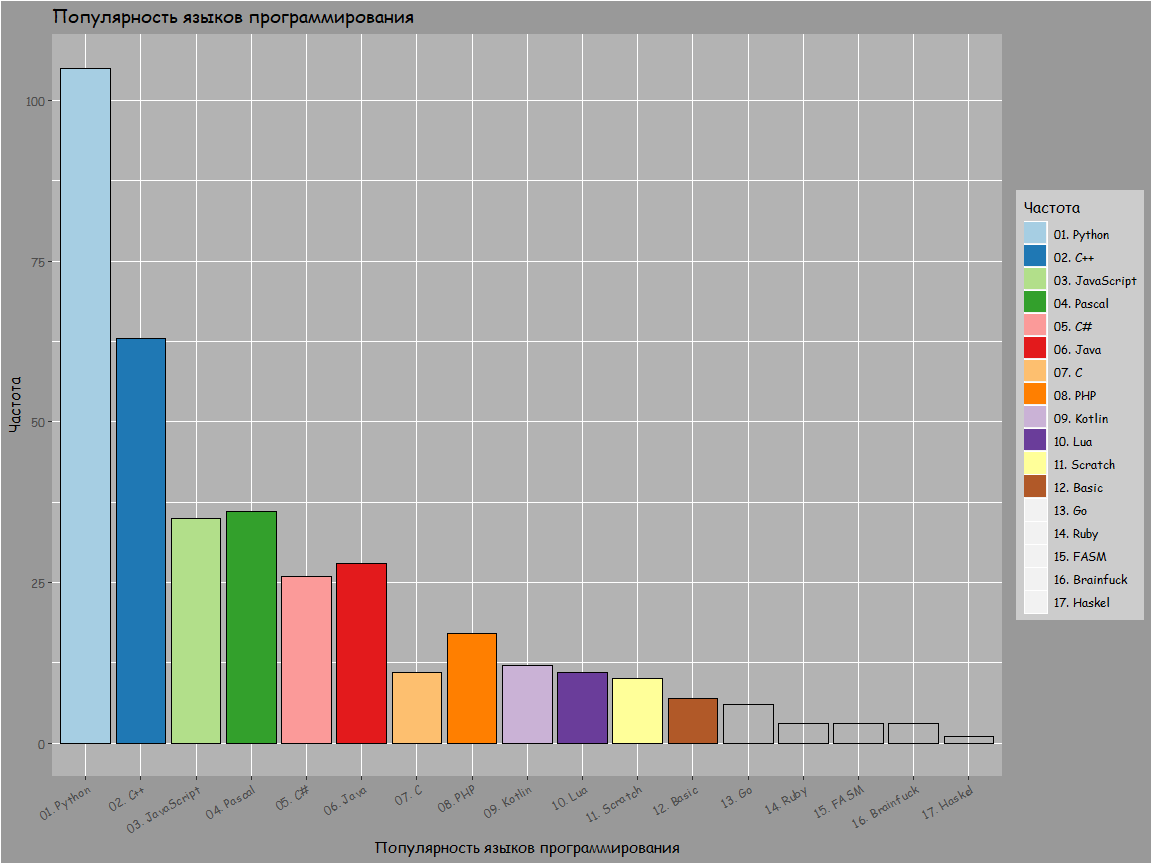


|  |  |
| --- | --- |
| x | Freq |
| 1 | 0.099 |
| 2 | 0.324 |
| 3 | 0.216 |
| 4 | 0.117 |
| 5 | 0.081 |
| 6 | 0.063 |
| 7 | 0.072 |
| 8 | 0.009 |
| 9 | 0.009 |
| 10 | 0.009 |

*Выводы*:

1. Оптимальное количество используемых языков - 2
2. Достаточно большие группы программистов встречаются вплоть до 7 языков
3. Лучший результат - 10 из 17 предложенных языков. Этот человек, либо машина, либо кремлебот, ибо пропаганду приходится распространять на разных сайтах и платформах.

### Самые популярные языки программирования

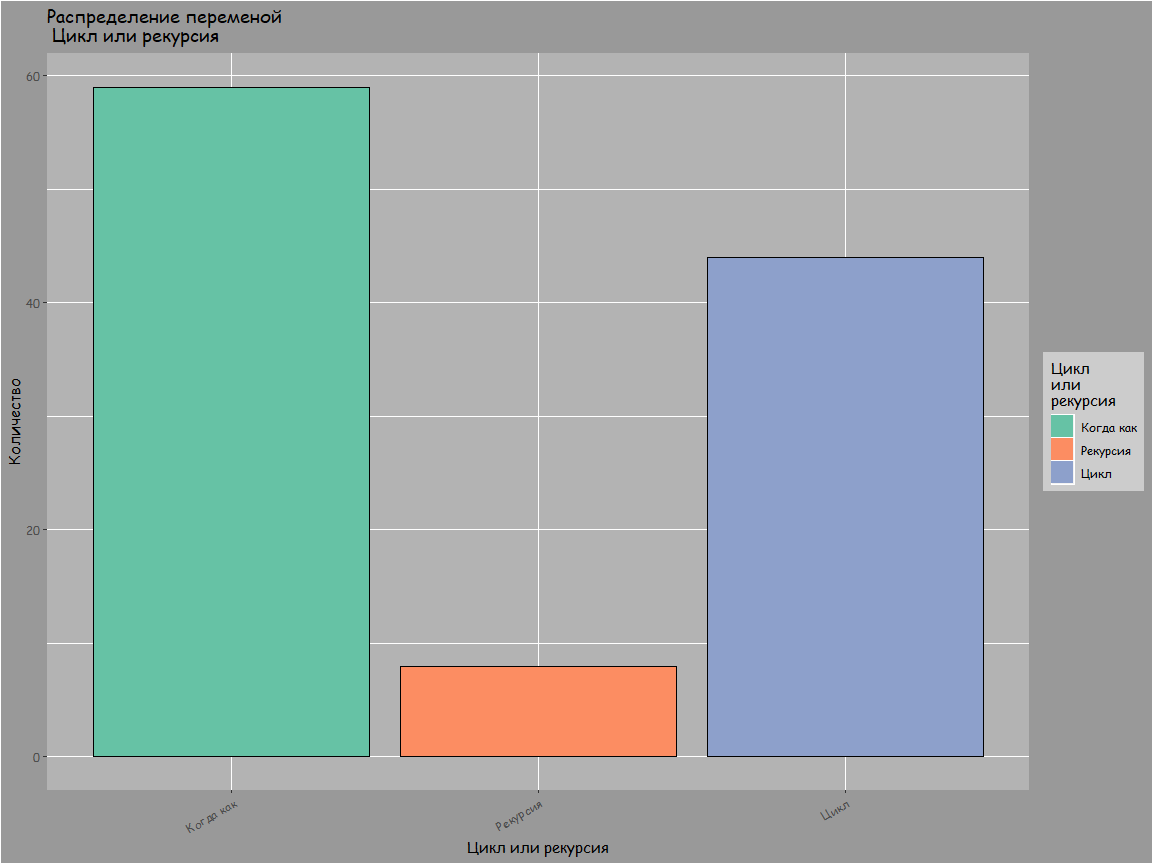
 *Примечания*:

1. Вариант ответа “Pascal” появился не с самого начала опроса, поэтому, его реальная доля используемости может быть больше.
2. На все языки не хватило цветов палитры диаграммы. Это лишний раз подчёркивает маловажность непопулярных вариатов.
3. Исследование написано на языке R, который даже не предлагался в опросе. Автор исследования и сам не ожидал, что писать придётся на нём.

*Выводы*:

1. А что вы ещё ожидали, если опрос проводился среди питонистов. Однако, не все 111 опрошенных используют питон.
2. На втором месте по популярности идут плюсы, и делают это совершенно заслуженно. Язык мощный, быстрый и всеобъемлющий.
3. Далее идёт JavaScript и старина Pascal. Первый знаменит веб-разработкой на нём, а второй тем, что школьники всей нашей многонациональной, стремящейся быть цифровой сверхдержавой, страны начинают знакомство с программированием именно с этого [пятидесятилетнего деда](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%81%D0%BA%D0%B0%D0%BB%D1%8C_(%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))
4. Далее следуют C# (музыканты его ещё называют до-бемоль) и Java. Как Микрософт и Оракл не соревновались в создании языка на виртуальной машине, выиграть расположение этого сектора прогеров у них не получилось.
5. Далее идут Си и PHP (в простонародье - пыха). Первый никогда не умрёт (если только из-за плюсов), а второй до сих пор терроризирует всю интернет-безопасность своими вставками в формы.
6. Kotlin оказался слишком молодым и незрелым, но у него есть все шансы занять место джавы под солнцем.
7. Далее по популярности следуют Lua, Scratch и Basic. Автор не находит, ни слов ни стереотипов о языке Lua, поэтому воздержится от комментария. Scracth - вот с чего надо начинать знакомство с программирвоанием - наглядно и просто. Что же касается Бейсика, то непонятно, почему этот современник Фортрана до сих пор коптит небо и пачкает своими манускриптами листы ГИА.
8. Go чалится на дне топ-листа рядом с Ruby (он же оксид алюминия с примесью хрома), FASM (разновидность ассемблера) и Brainfuck (воздержусь от нецензурных переводов и назову его “Вынос мозга”, хотя это не передаёт всей его сущности)
9. Подпирает дно топ-листа функциональный язык Haskel. (no comments)

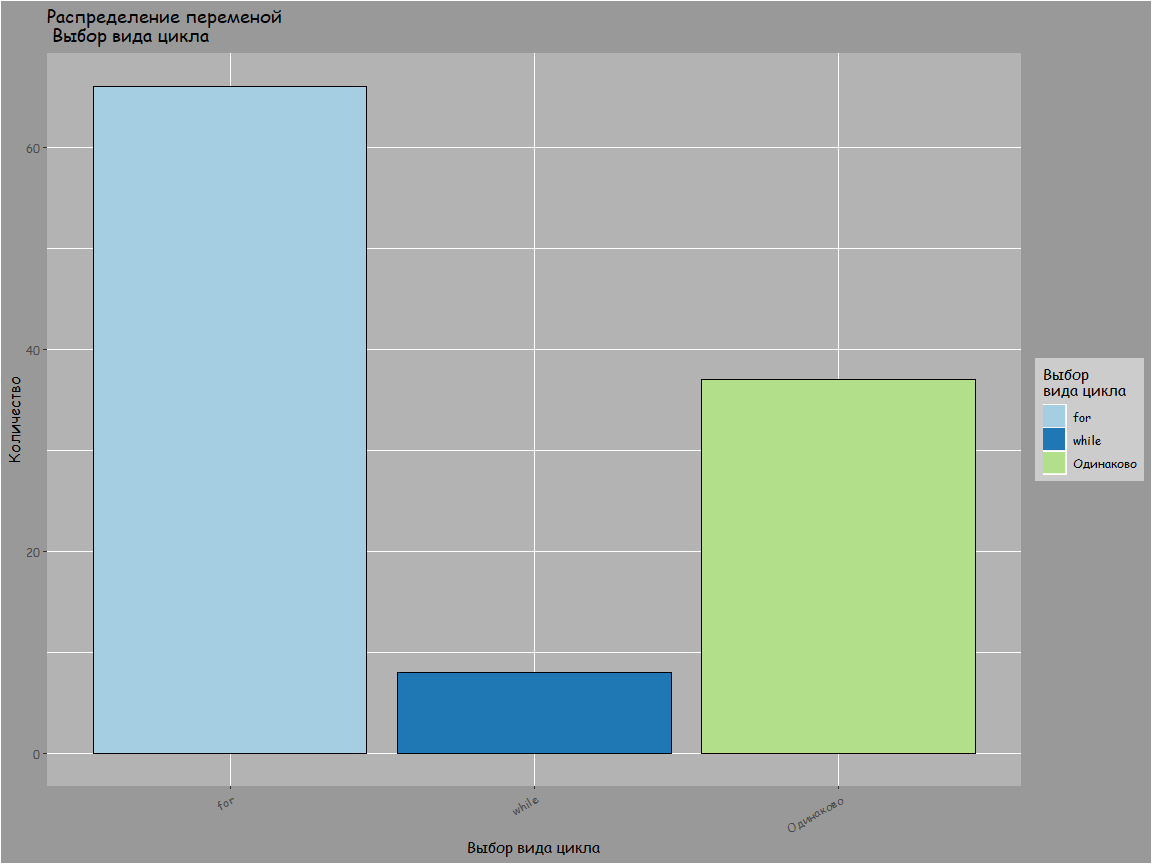
### Цикл или рекурсия?



|  |  |
| --- | --- |
| x | Freq |
| Когда как | 0.532 |
| Рекурсия | 0.072 |
| Цикл | 0.396 |
| *Выводы*: |  |

1. Половина опрошенных прониклись вселенской мудростью и нашли оба подхода: объектный и функциональный - применимыми в конкретных случаях.
2. Малый процент выбравших рекурсию можно объяснить тем, что многие так и не смогли из неё выбраться.

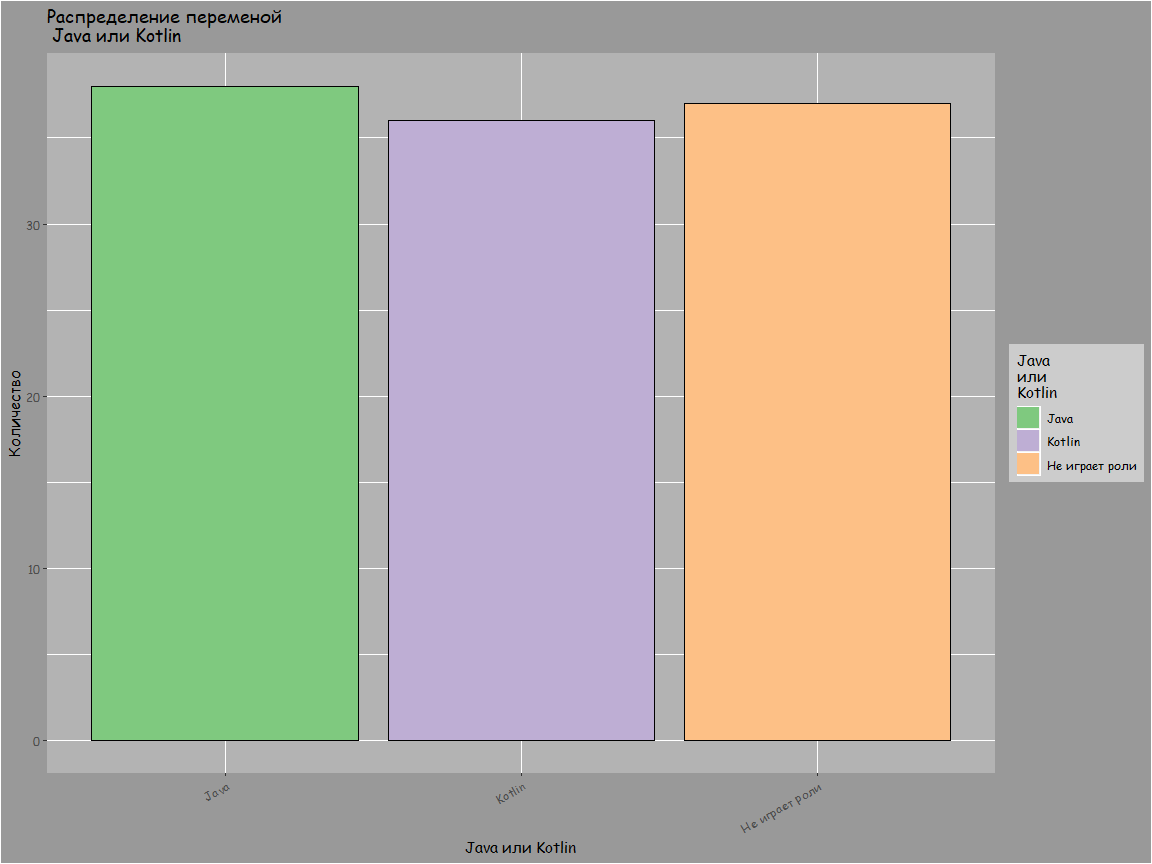
### For или While?



|  |  |
| --- | --- |
| x | Freq |
| for | 0.595 |
| while | 0.072 |
| Одинаково | 0.333 |
| *Выводы*: |  |

1. Большинство опрошенных не застали времена ассемблера, поэтому сочли встроенную переменную в цикле for более удобной.
2. Количество проникшихся вселенской мудростью по этому вопросу оказалось меньше, по сравнению с прошлым вопросом.

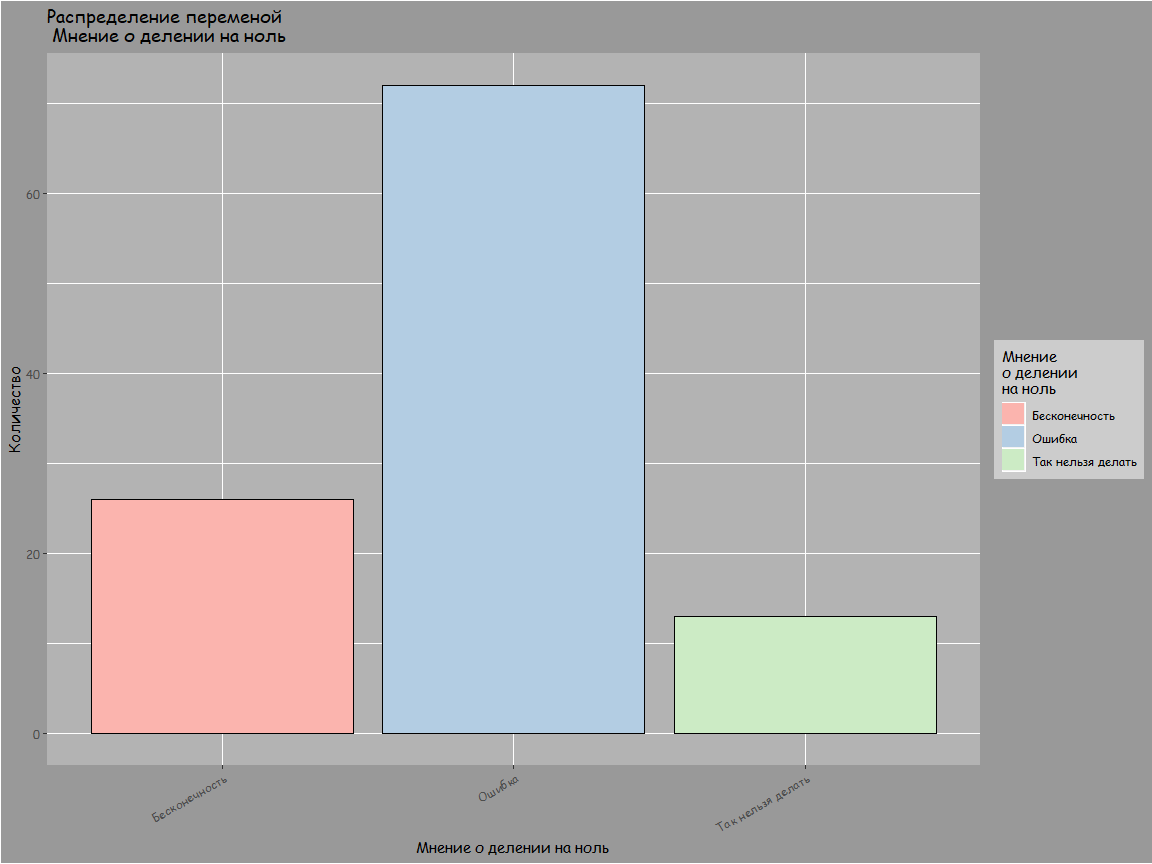
### Java или Kotlin?



|  |  |
| --- | --- |
| x | Freq |
| Java | 0.342 |
| Kotlin | 0.324 |
| Не играет роли | 0.333 |
| *Выводы*: |  |

1. Это одна из немногих переменных, которая распределилась практически равномерно
2. Ровным счётом непонятно, что же лучше и как быть
3. Холиварная война по предпочтению java и kotlin могла бы привести к полной [аннигиляции](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BD%D0%B8%D0%B3%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F) программистов.

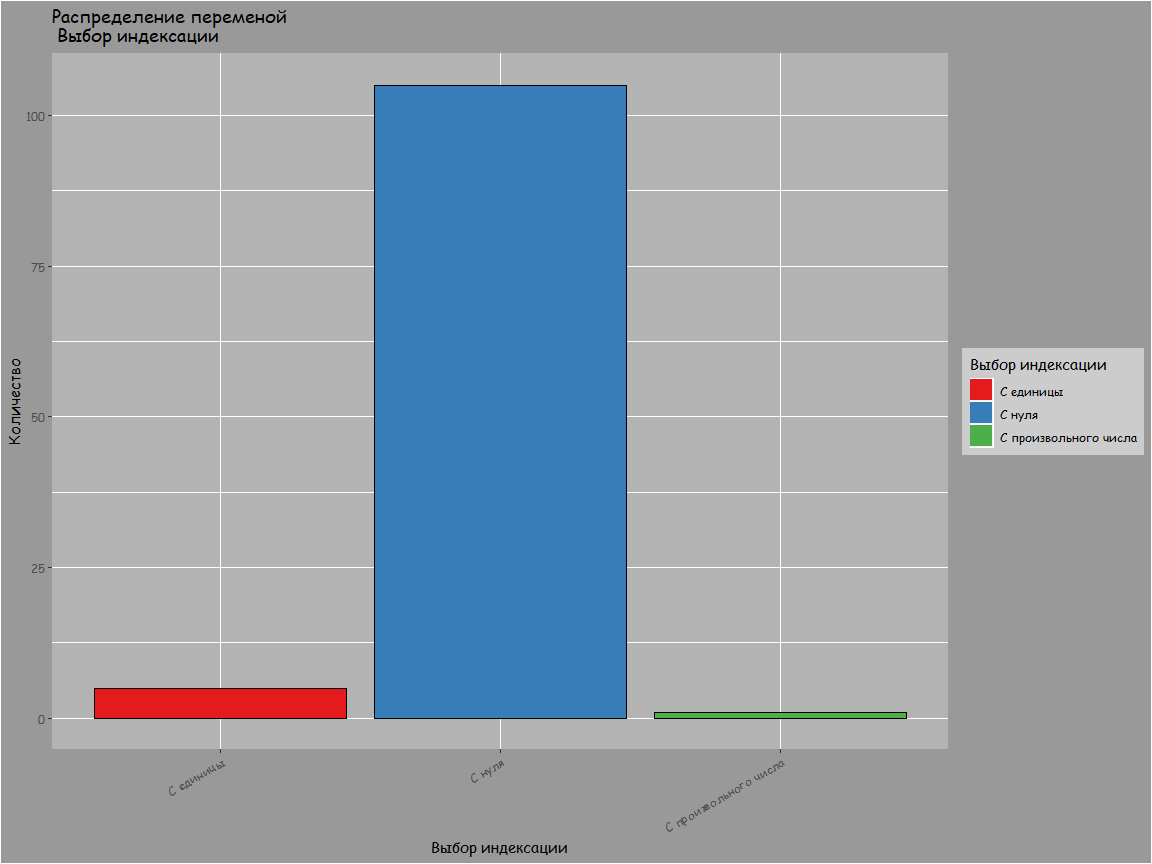
### Мнение о делении на ноль



|  |  |
| --- | --- |
| x | Freq |
| Бесконечность | 0.234 |
| Ошибка | 0.649 |
| Так нельзя делать | 0.117 |
| *Выводы*: |  |

1. Занудное большинство считает всё необычное и неоднозначное поломкой системы и стремится во что бы то ни стало избегать таких ситуаций.
2. Сердце радуется за наличие среди прогеров этих 23% от общей массы, которые, возможно, разбираются в пределах.
3. Юные [Пьеры Ланцели](https://ru.wikipedia.org/wiki/%D0%92%D0%B0%D0%BD%D1%86%D0%B5%D0%BB%D1%8C,_%D0%9F%D1%8C%D0%B5%D1%80_%D0%9B%D0%BE%D1%80%D0%B0%D0%BD) и ответвления большинства составляют около 12%.

### Выбор индексации

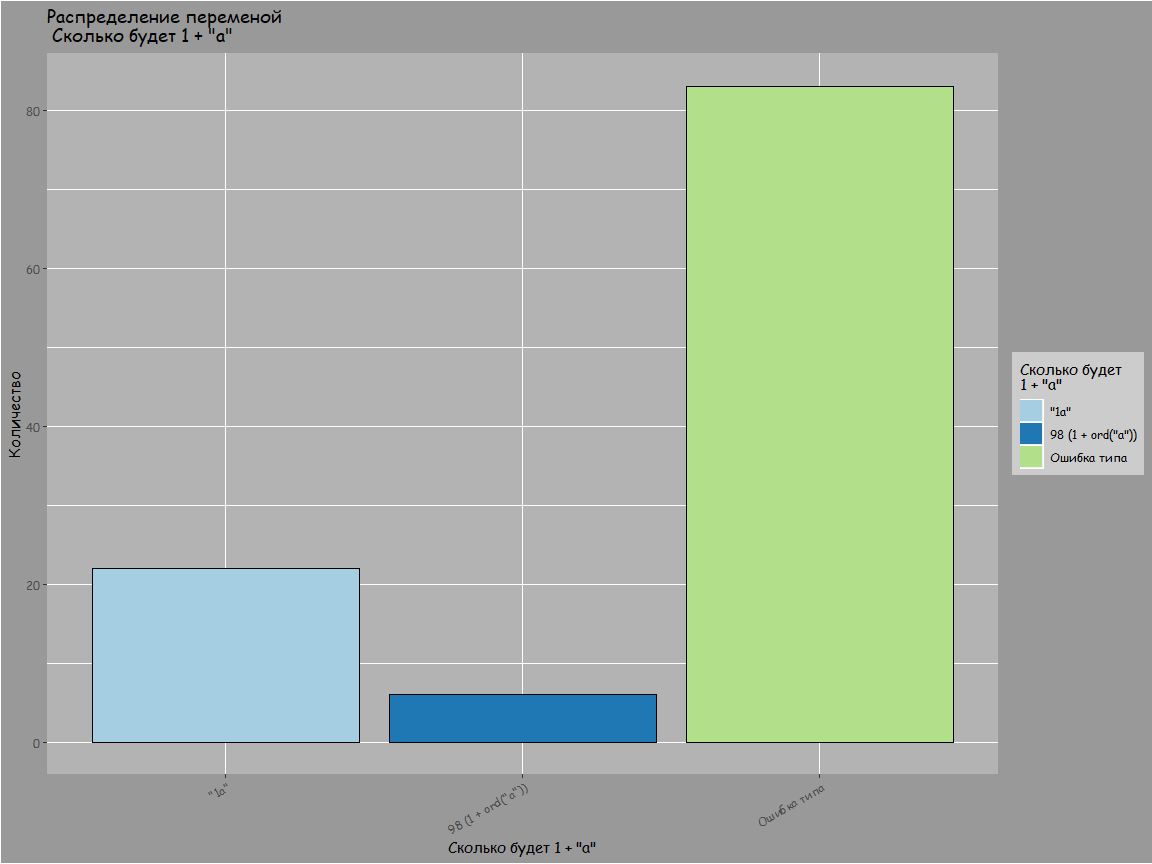


|  |  |
| --- | --- |
| x | Freq |
| С единицы | 0.045 |
| С нуля | 0.946 |
| С произвольного числа | 0.009 |
| *Выводы*: |  |

1. Программисты почти превратились в компьютеры, ибо разделяют их нумерацию с нуля.
2. Истинные мечтатели программирования и ярые революционеры знают, где трава зеленее

Не во всех современных языках программирования нумерация начинается с нуля. Например в R нумерация начинается с единицы, а в Паскале ~~(кто-то назвал его современным :) )~~ - с произвольного числа.

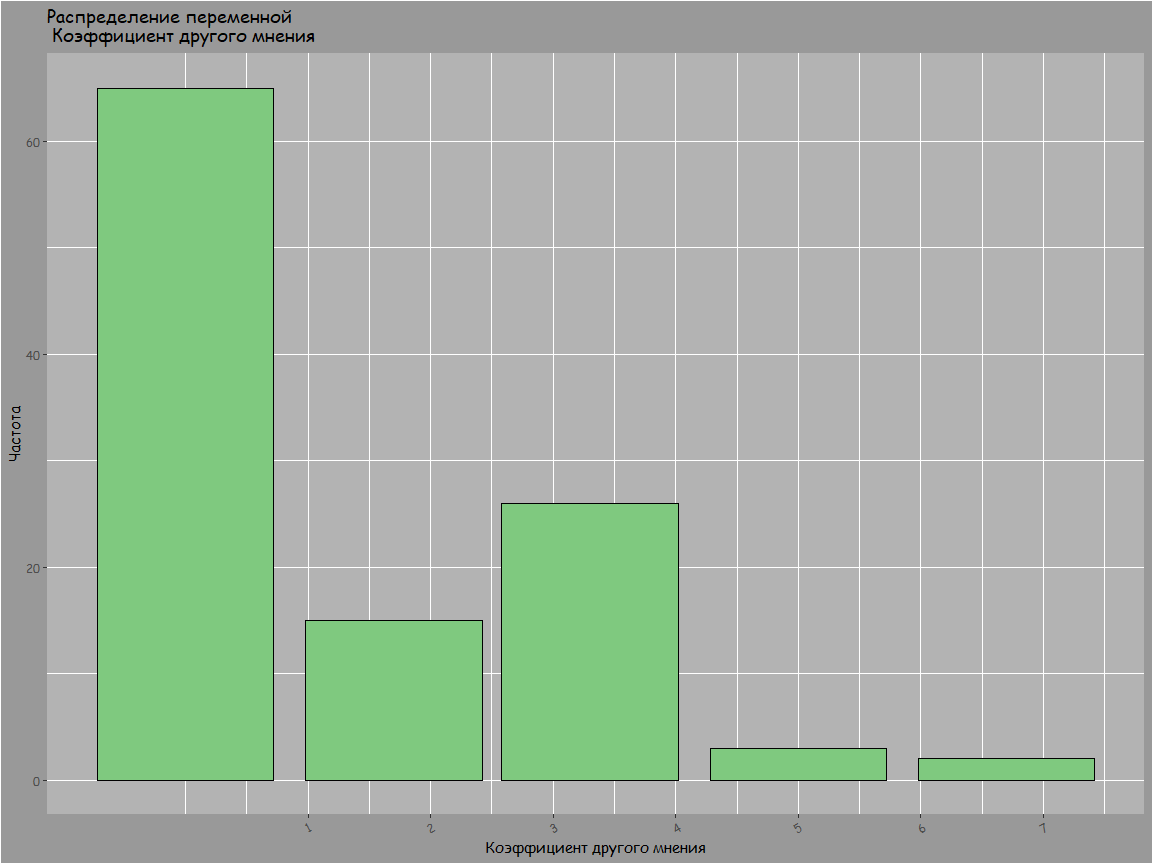
### Сколько будет 1 + “a”?



|  |  |
| --- | --- |
| x | Freq |
| “1а” | 0.198 |
| 98 (1 + ord(“a”)) | 0.054 |
| Ошибка типа | 0.748 |
| *Выводы*: |  |

1. Большиство, вторя всяким питонам и джавам, говорит, что смешение типов невозможно.
2. Доля любителей слабой типизации достаточно велика.

### Другое мнение

 *Примечание*:

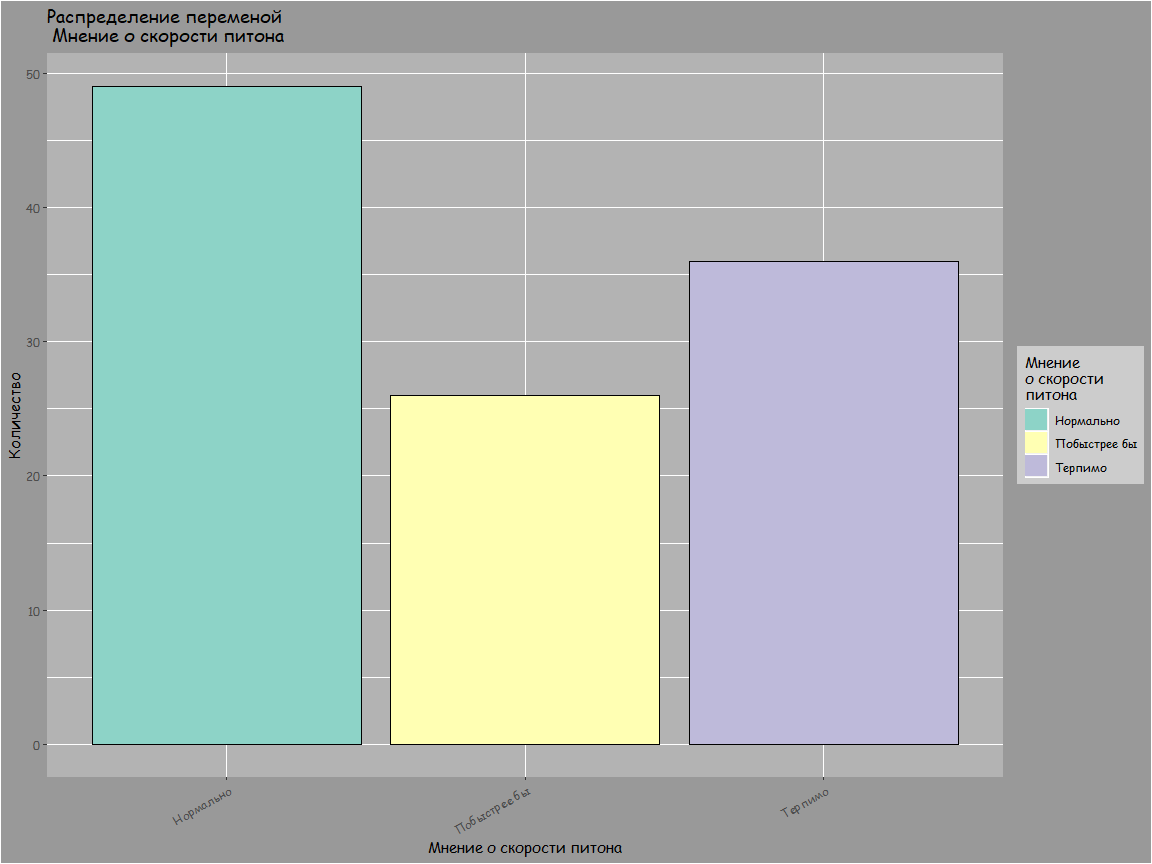
1. Данный критерий оценивал количество ответов, несовпадающих с ответами большинства по спорным вопросам программирования (деление на ноль, индексирование и типизация)
2. Коэффииент и не носил бы другого названия, если бы большинство по этой шкале получило ноль.

*Выводы*:

1. Не нашлось ни одного, кто-бы получал бы ∞ при делении на ноль, индексировал массивы с произвольного числа и получал 98 при сложении 1 и “a”. А хотелось бы взглянуть на этого “сферического единорога в вакууме”
2. Достаточно много программистов дали ответы, несильно отличающиеся от ответов большинства.

## Вопросы о питоне

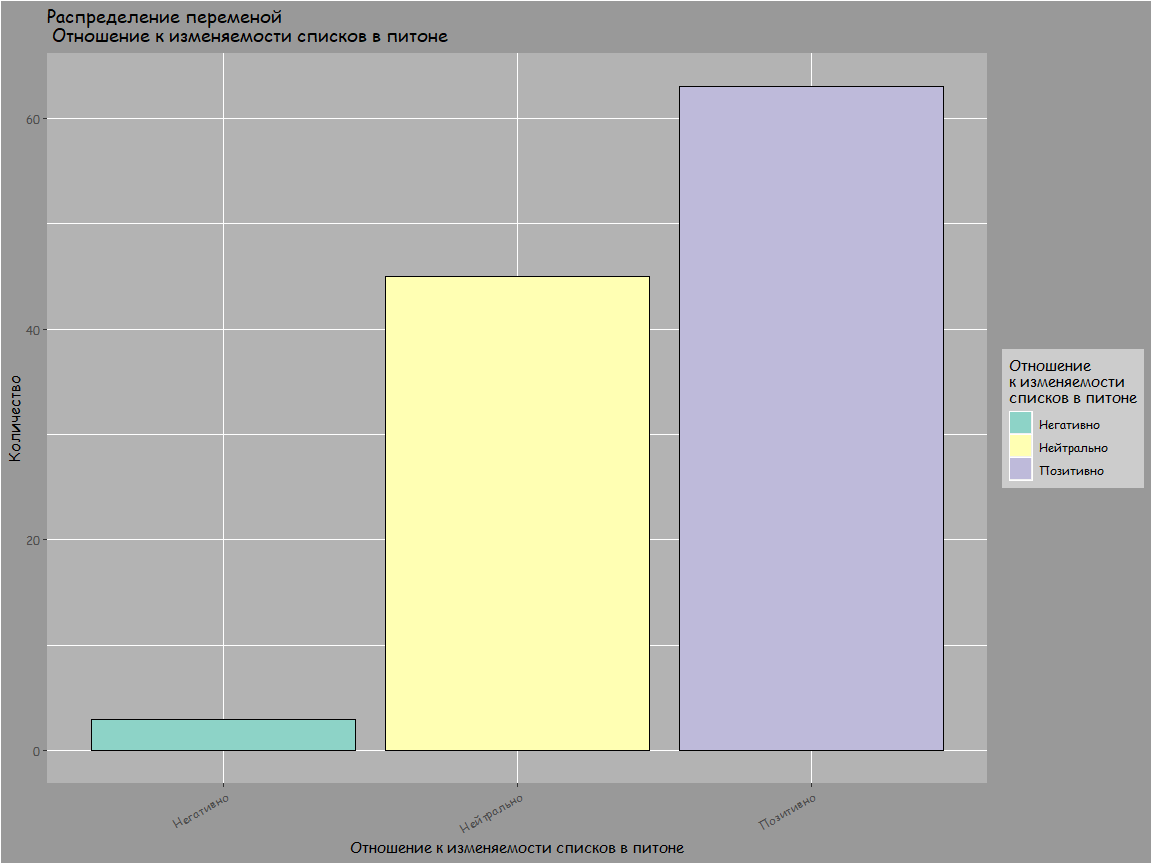
### Мнение о скорости питона



|  |  |
| --- | --- |
| x | Freq |
| Нормально | 0.441 |
| Побыстрее бы | 0.234 |
| Терпимо | 0.324 |
| *Выводы*: |  |

1. Где-то внутри столбика “побыстрее бы” прячутся ленивые олимп-прогеры и дата-саентисты.
2. Около половины программистов вообще не парится
3. Доля терпил довольно высока.

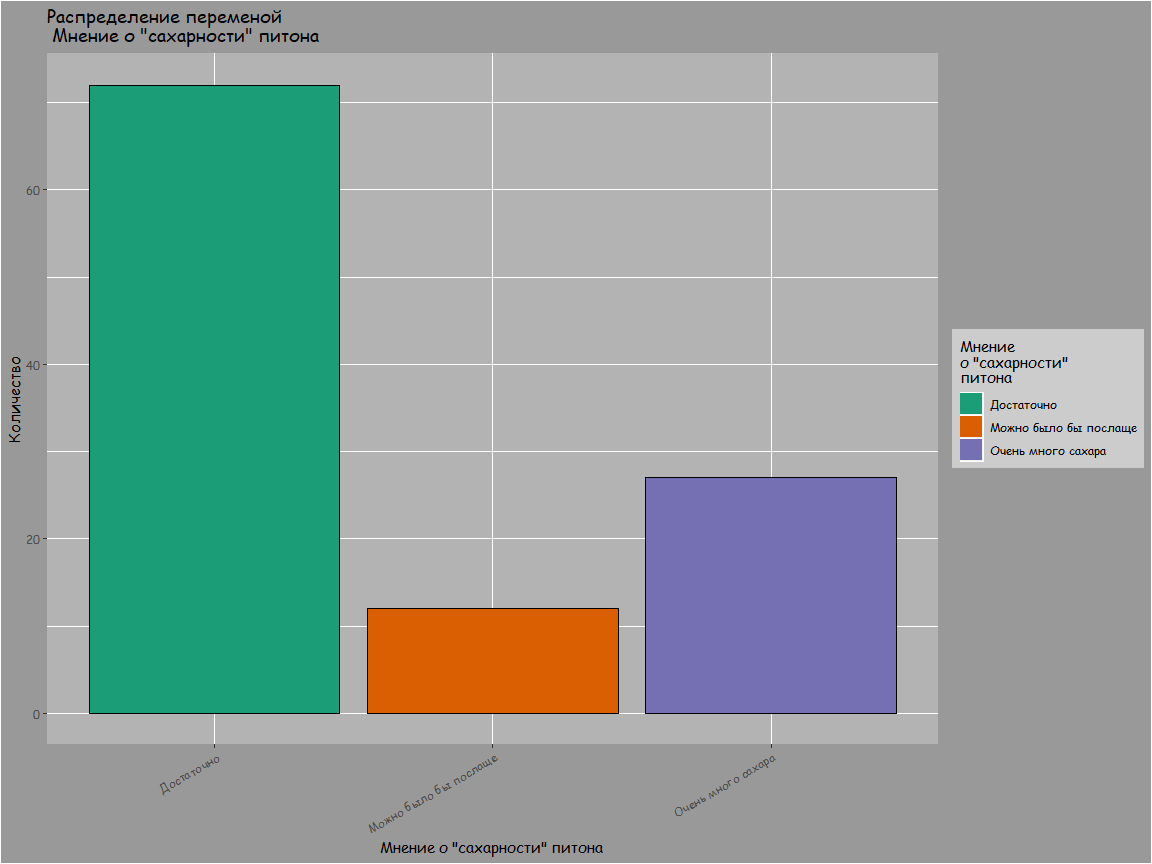
### Отношение к изменяемости списков



|  |  |
| --- | --- |
| x | Freq |
| Негативно | 0.027 |
| Нейтрально | 0.405 |
| Позитивно | 0.568 |
| *Выводы*: |  |

1. С одной стороны, изменяемость списков (и прочих изменяемых коллекций) позволяет менять элементы без необходимости копирования всего списка.
2. Но с другой стороны, она вызывает геморроидальные боли при записи элементов якобы “разных” списков. Поэтому в данном случае, их всё равно приходится копировать.
3. Как бы там ни было, большинство ответов составил вариант “позитивно”

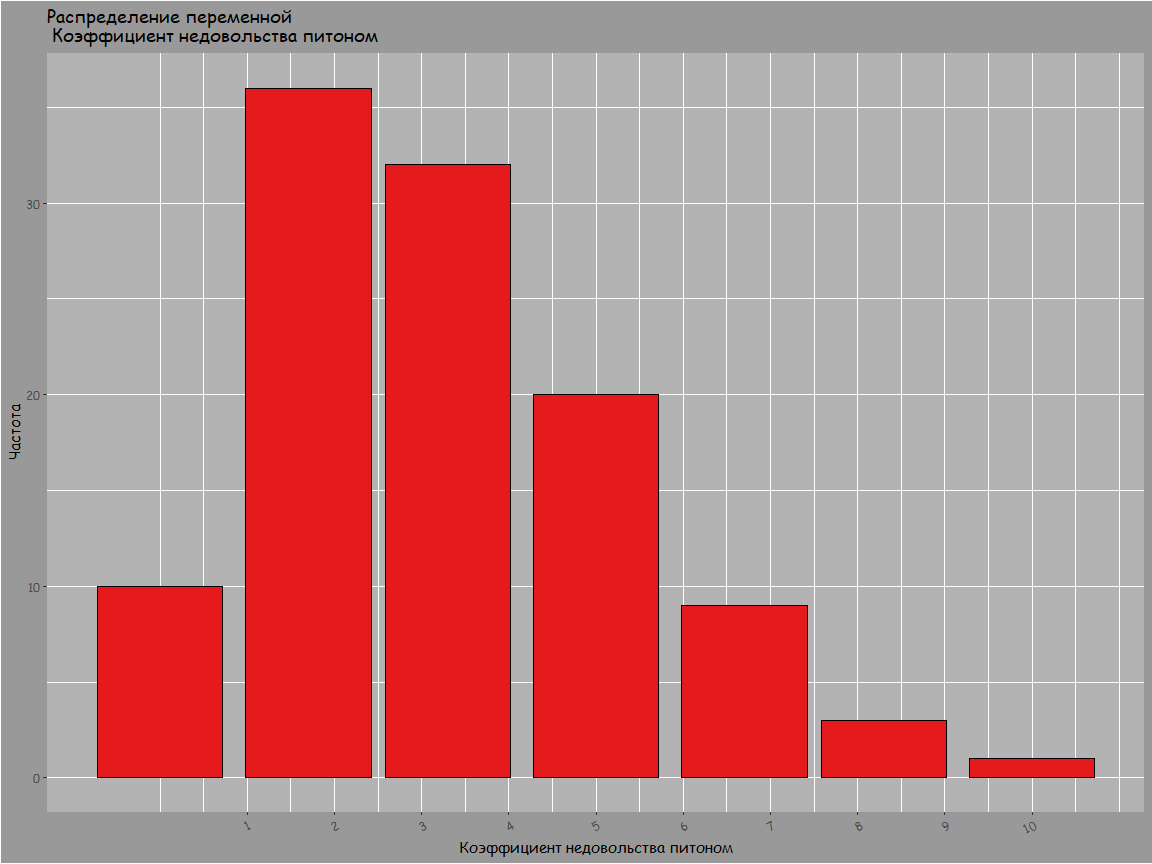
### Мнение о “сахарности” питона



|  |  |
| --- | --- |
| x | Freq |
| Достаточно | 0.649 |
| Можно было бы послаще | 0.108 |
| Очень много сахара | 0.243 |
| *Выводы*: |  |

1. Основная масса вполне довольна удобством методов питона. Конечно, ведь по сравнению с другими языками, это просто спаснение от соблюдения типизации и расстановки скобок и точек с запятой.
2. Четверть даже не потянула всей мощи Великого Третьего Змея.
3. Количество диабетиков и сладкоежек составило ~ 11% от опрошенных

### Недовольство питоном



*Примечания*:

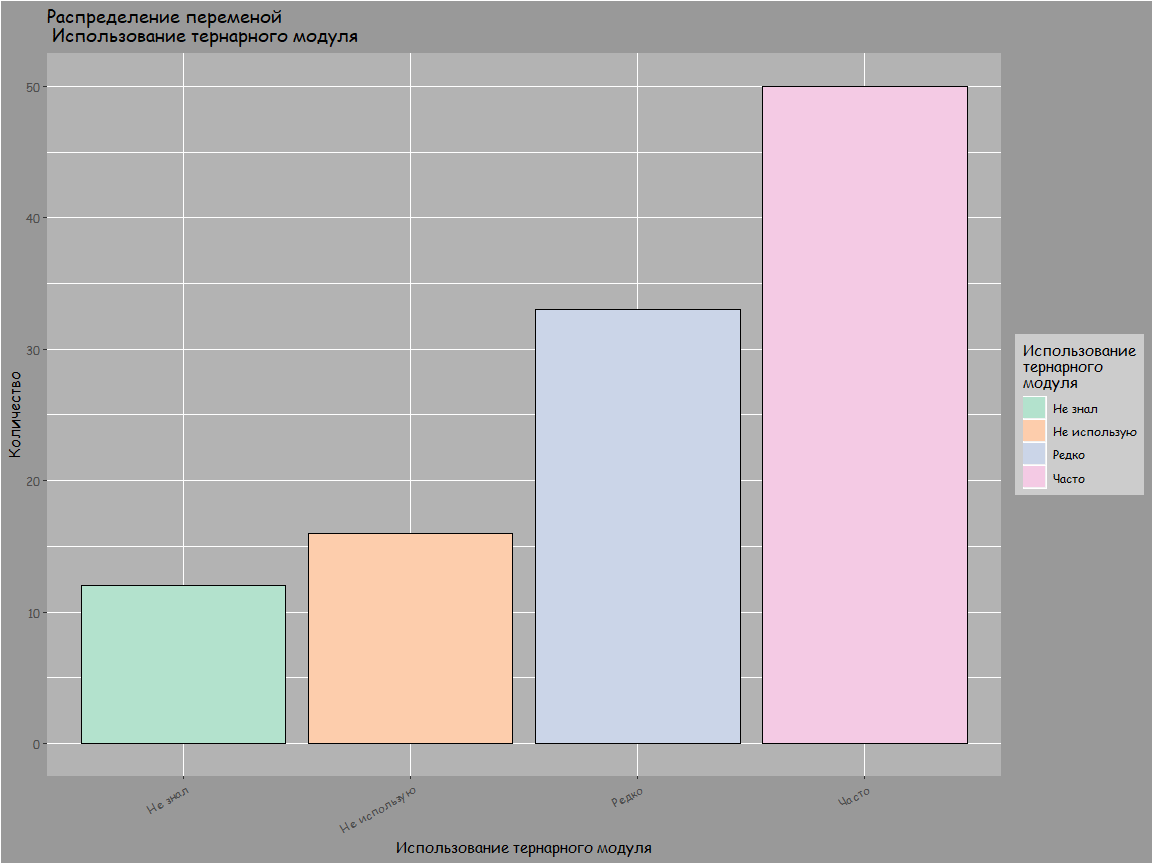
1. Мнение людей, не использующих питон, не учитывались, так как нельзя критиковать то, о чём даже не знаешь.

*Выводы*:

1. Было бы странно увидеть группу людей, полностью недовольным питоном, но активно использующим его.
2. Возможно, люди ждут чего-то болшего, чем питон

## Вопросы о использовании синтаскического сахара

### Использование тернарного модуля

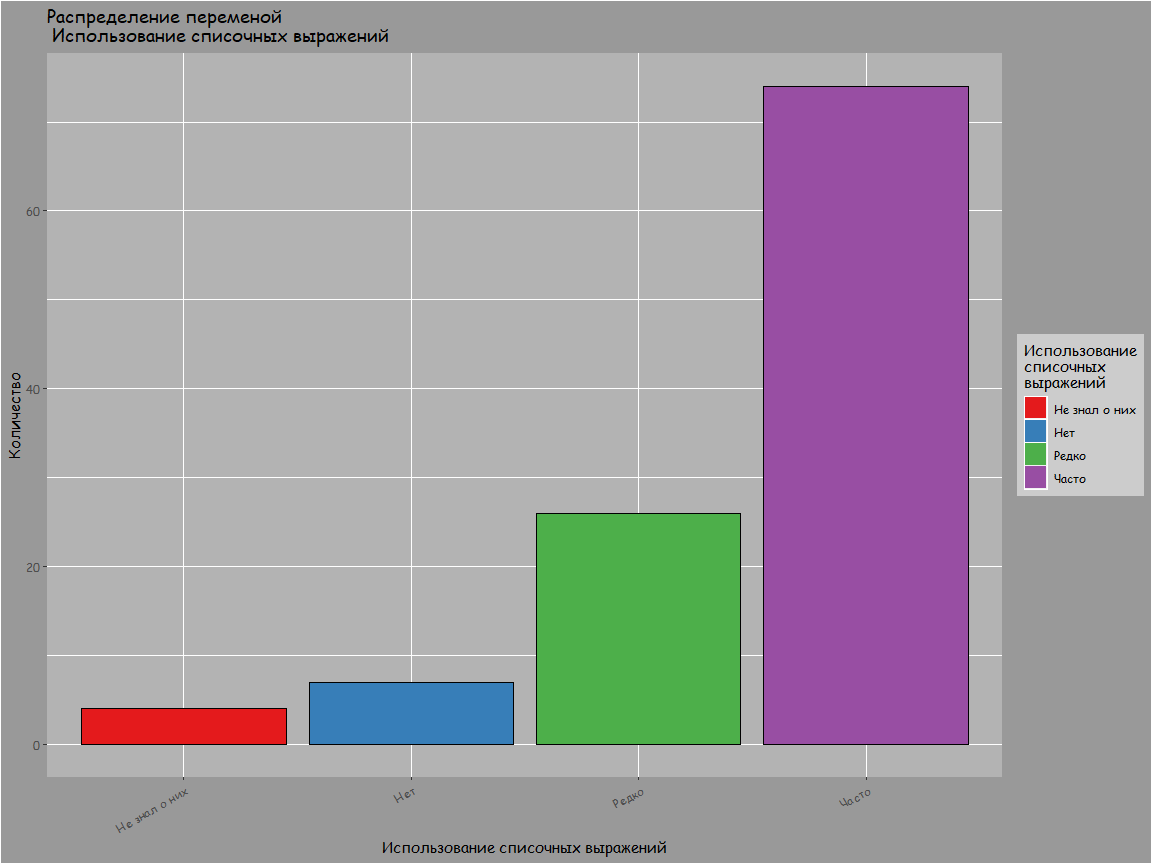


|  |  |
| --- | --- |
| x | Freq |
| Не знал | 0.108 |
| Не использую | 0.144 |
| Редко | 0.297 |
| Часто | 0.450 |

*Выводы*:

1. Тернарный модуль является довольно популярной конструкцией, к тому же, относительно удобной.
2. 14% от опрошеных составляют те, для кого тернарный модуль является относительно неудобным. Это лишний раз доказывает относительность полезности языковых плюшек

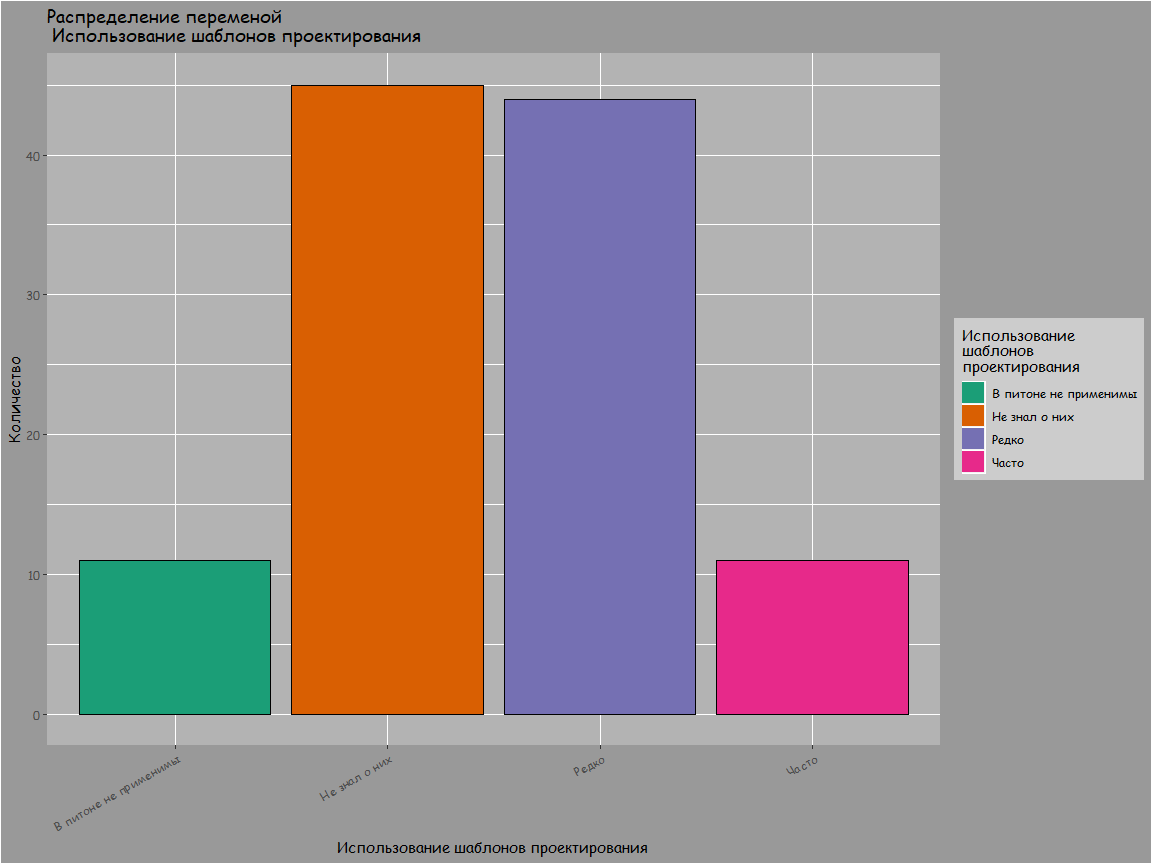
### Использование списочных выражений



|  |  |
| --- | --- |
| x | Freq |
| Не знал о них | 0.036 |
| Нет | 0.063 |
| Редко | 0.234 |
| Часто | 0.667 |
| *Выводы*: |  |

1. Девять десятых опрошенных хоть раз пробовали и часть из них осознали всю мощь циклов внутри квадратных скобок
2. Отмечена большая просвященность программистов в этом вопросе

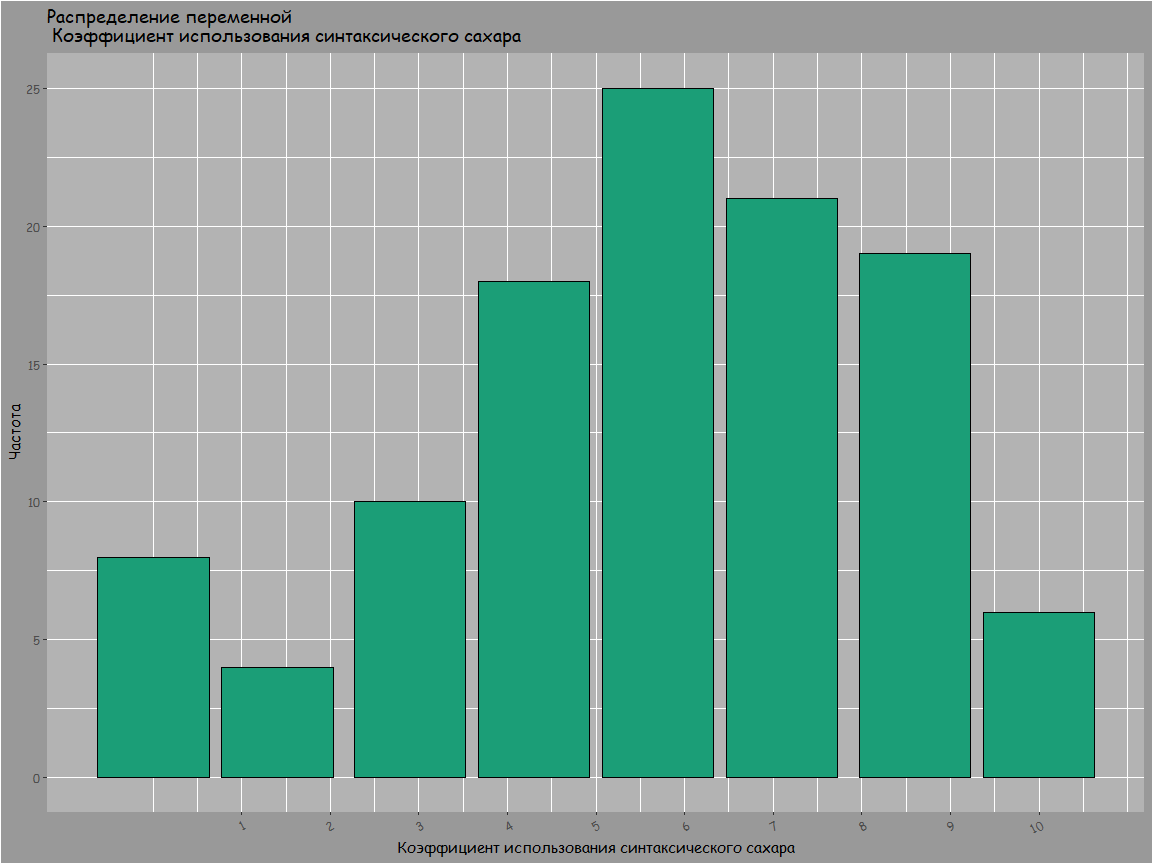
### Использование шаблонов проетирования



|  |  |
| --- | --- |
| x | Freq |
| В питоне не применимы | 0.099 |
| Не знал о них | 0.405 |
| Редко | 0.396 |
| Часто | 0.099 |
| *Выводы*: |  |

1. Вопрос проектирования ООП не затронул 40% от опрошенных
2. Автор солидарен с мнением о неприменимости шаблонов проектирования в питоне. По крайней мере, автор не застал ни одной реализации [абстрактной фабрики](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D0%B0%D1%8F_%D1%84%D0%B0%D0%B1%D1%80%D0%B8%D0%BA%D0%B0_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) или [моста](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D1%81%D1%82_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) в деле.
3. Половина опрошенных в той или иной мере использует паттерны проектирования

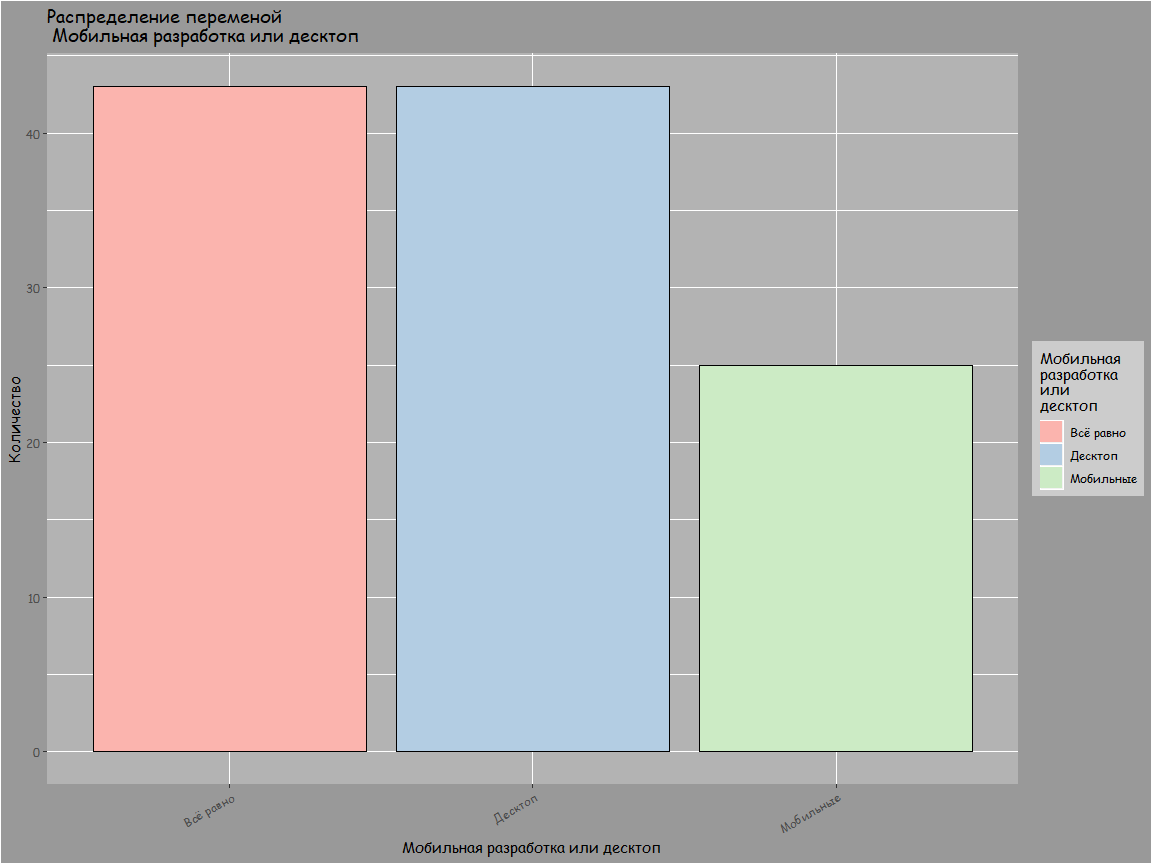
### Использование синтаксического сахара

 *Выводы*:

1. Особо выделяются из распределения две группы: довольно большая группа, не использующая ничего и примерон такая же по габаритам группа, участники которой используют всё и часто.
2. Больше всего опрошенных используют “сахар” 50 на 50.

## Вопросы о сферах программирования

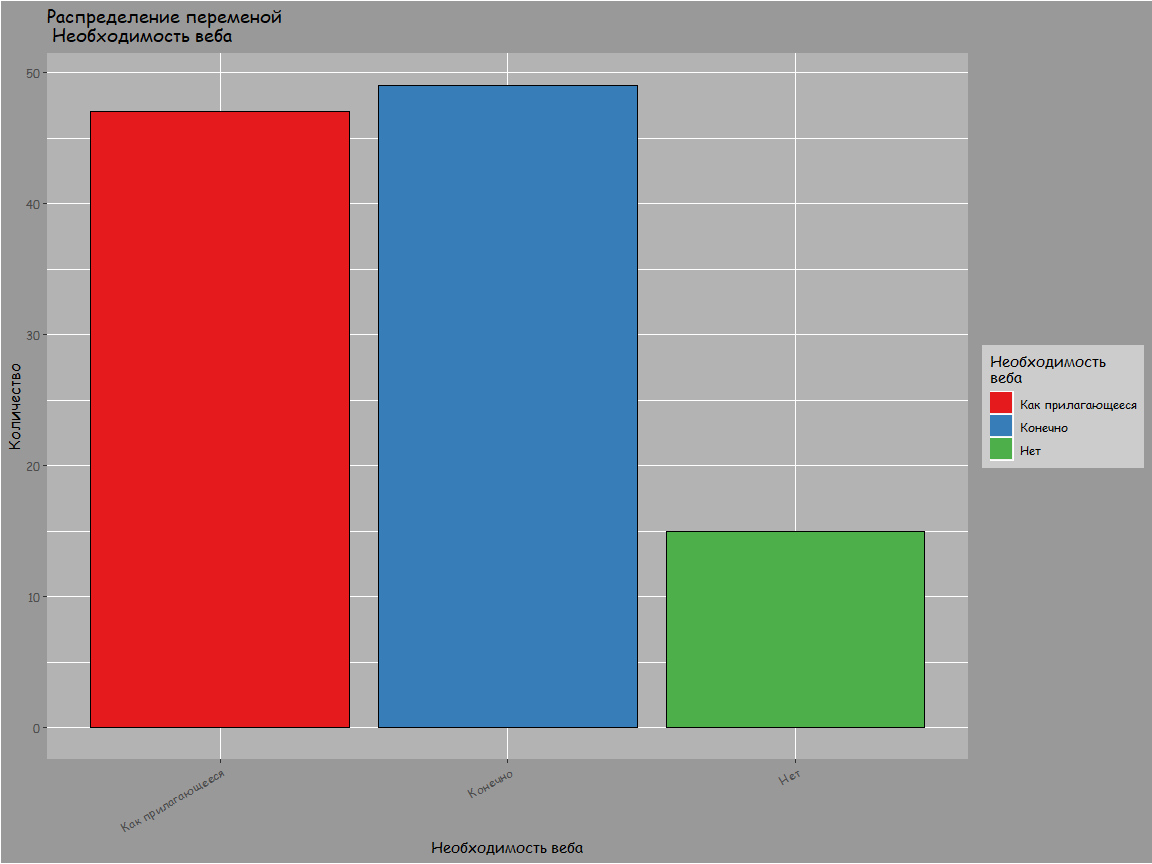
### Мобильная или десктоп разработка?



|  |  |
| --- | --- |
| x | Freq |
| Всё равно | 0.387 |
| Десктоп | 0.387 |
| Мобильные | 0.225 |
| *Выводы*: |  |

1. Более трети имеют достаточно инертное мнение по предпочтению мобильной или настольной разработки
2. Доля “настольников” соразмерна доле безразличных и философов
3. Доля любителей покодить под смартфоны приблизительно равна четверти от всего объёма

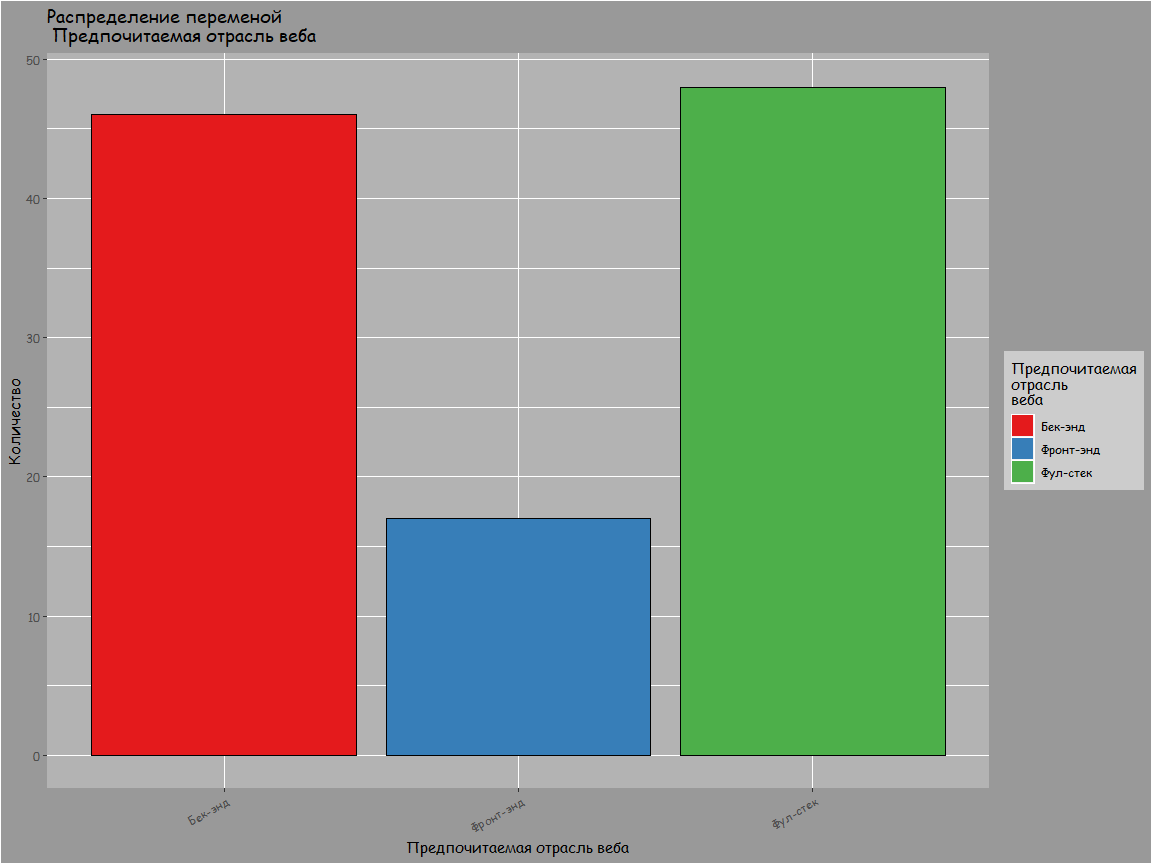
### Необходимость веба



|  |  |
| --- | --- |
| x | Freq |
| Как прилагающееся | 0.423 |
| Конечно | 0.441 |
| Нет | 0.135 |
| *Выводы*: |  |

1. Более 85% опрошенных признают важность веба, правда, половина из них видит в нём лишь интерфейс для пользовательских проектов
2. Остальные 15%, вероятно, ожидают прихода суперприложений и исчезновения браузеров в статусе архаизмов.

### Бек или фронт энд?

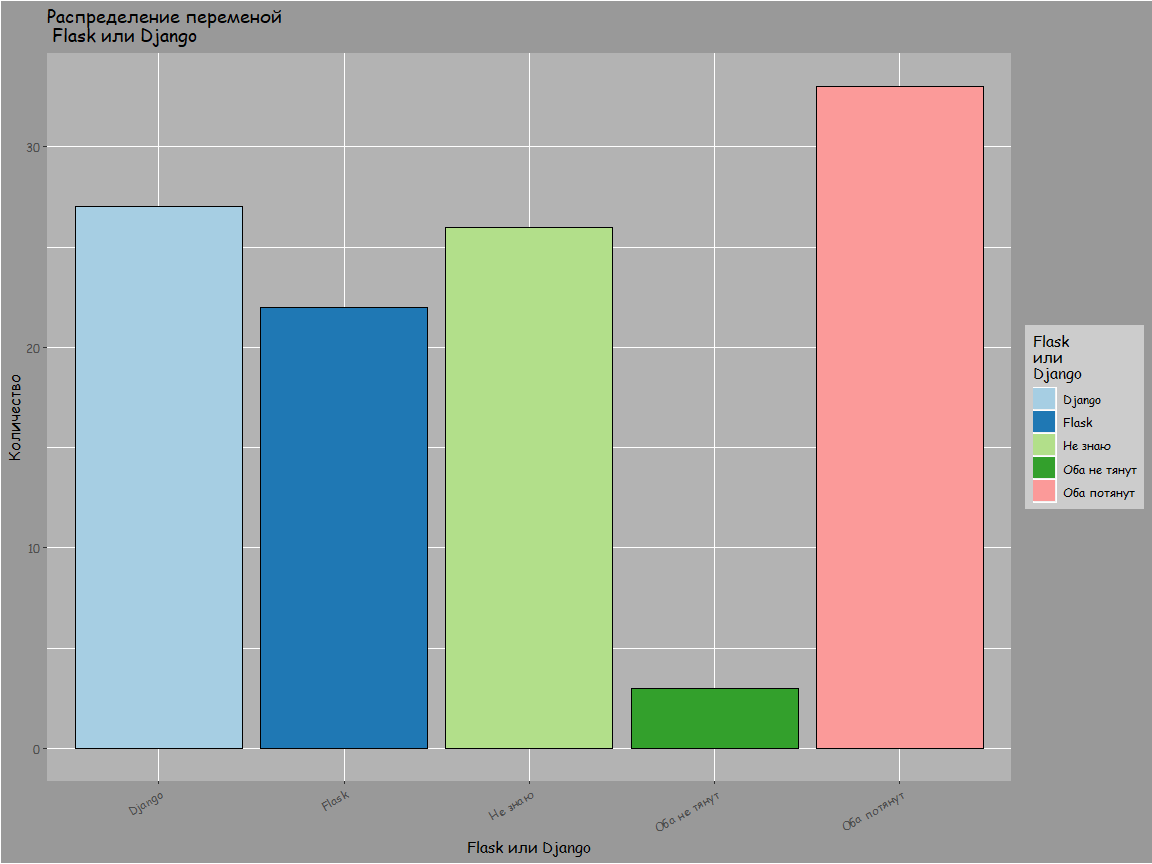


|  |  |
| --- | --- |
| x | Freq |
| Бек-энд | 0.414 |
| Фронт-энд | 0.153 |
| Фул-стек | 0.432 |
| *Выводы*: |  |

1. Куда больше прогеры любят побэкендить, нежели пофронтендить. Что, впрочем, логично, ведь опрашивались программисты, а не дизайнеры.
2. Также программисты предпочитают запереться в танке и топить и за фронт, и за тыл. Пофулстечить, так сказать.

(здесь должна была быть реклама игры про танки, но мне не заплатили)

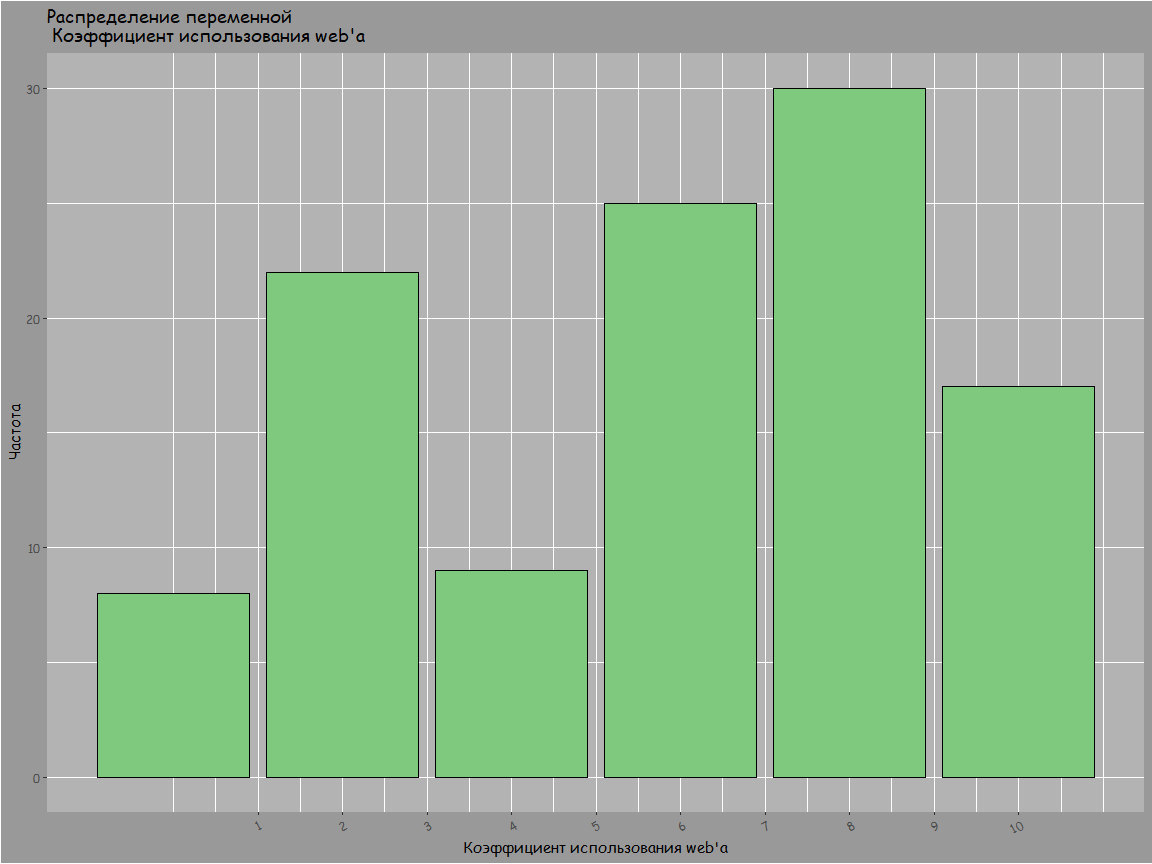
### Flask или Django?



|  |  |
| --- | --- |
| x | Freq |
| Django | 0.243 |
| Flask | 0.198 |
| Не знаю | 0.234 |
| Оба не тянут | 0.027 |
| Оба потянут | 0.297 |
| *Выводы*: |  |

1. Пессимистов в вопросе возможностей этих двух фреймворков, оказалось довольно мало
2. Вселенская мудрость вновь подсказала большинству, что важен не функционал, а применимость его в конкретных ситуациях. “Оба потянут” - таков самый популярный ответ.
3. Частота ответа “Не знаю” показывает, что вопрос веб-разработки в питоне не интересовал около четверти опрошенных
4. С небольшим отрывом Джанго опережает Фласк

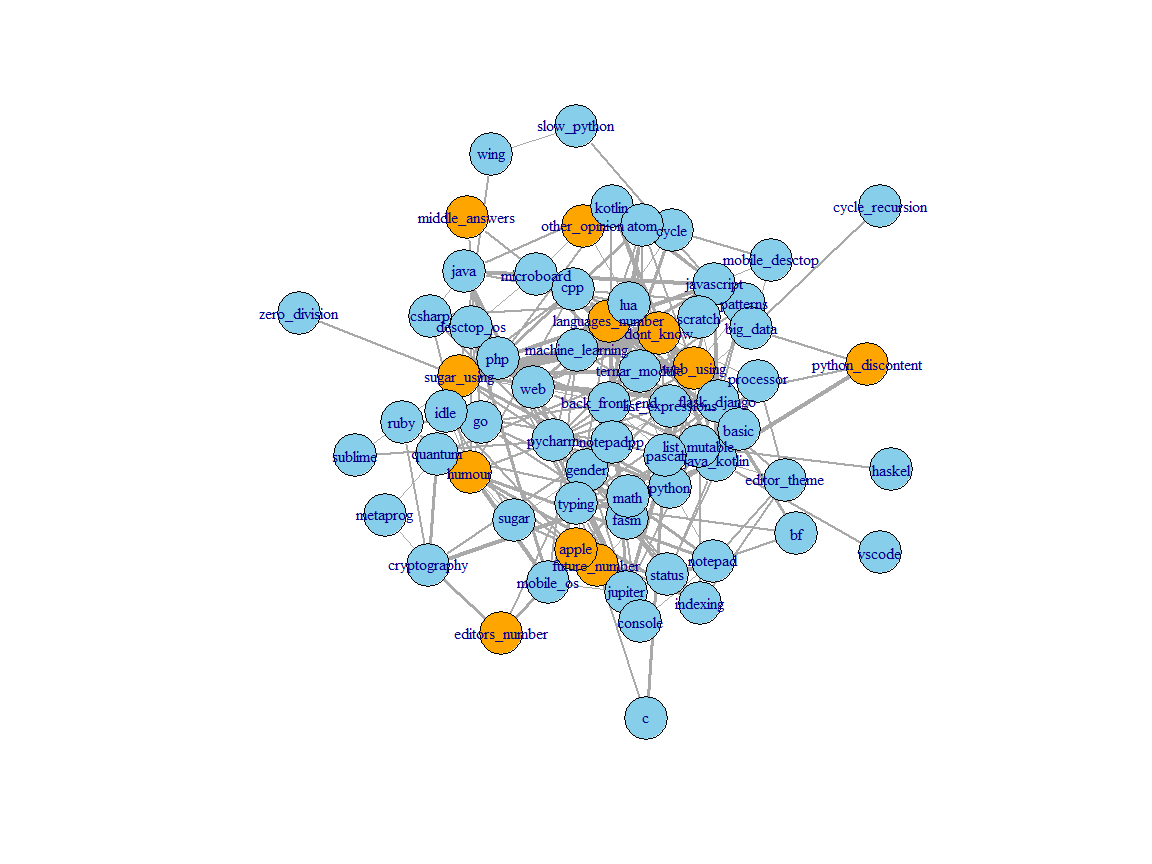
### Использование веба

 *Выводы*:

1. Веб используется программистами довольно хорошо
2. По форме график напоминает распределение переменной “использование синтаксического сахара” (об этом *ниже*)

# Найденные взаимосвязи переменных

Сначала, была принята попытка найти взаимосвязь визуально между всеми возможными комбинациями переменных. Но комбинаторика была не на стороне исследователя, ведь комбинаций было около 3000. Выбрать что-то не только статистически вероятное, но и интересное оказалось слишком сложно.

Далее была попытка визуализироввать взаимосвязь переменных в виде графа с учётом статистической значимости связей. Получилось что-то такое: 

Если вам кажется, что это не очень-то понятно и наглядно, то вам не кажется. На одном графе указаны 67 вершин и 225 связей между ними. О читаемости такой информации говорить не приходится.

Последняя и итоговая попытка была реализована следующим образом. Сначала с помощью функциональной мощи языка R и статистических методов были найдены связи, которые *статистически вероятно* существуют.

Далее графики этих связей исследовались на предмет понятности имеющихся в них зависимостей. В итоговый документ вошли только те графики зависимостей, которые автор смог бы объяснить не только себе, но и другим. Автор оставляет за любым заинтересовавшимся право на исследование прочих, неопубликованных взаимосвязей и публикацию их в отдельном исследовании.

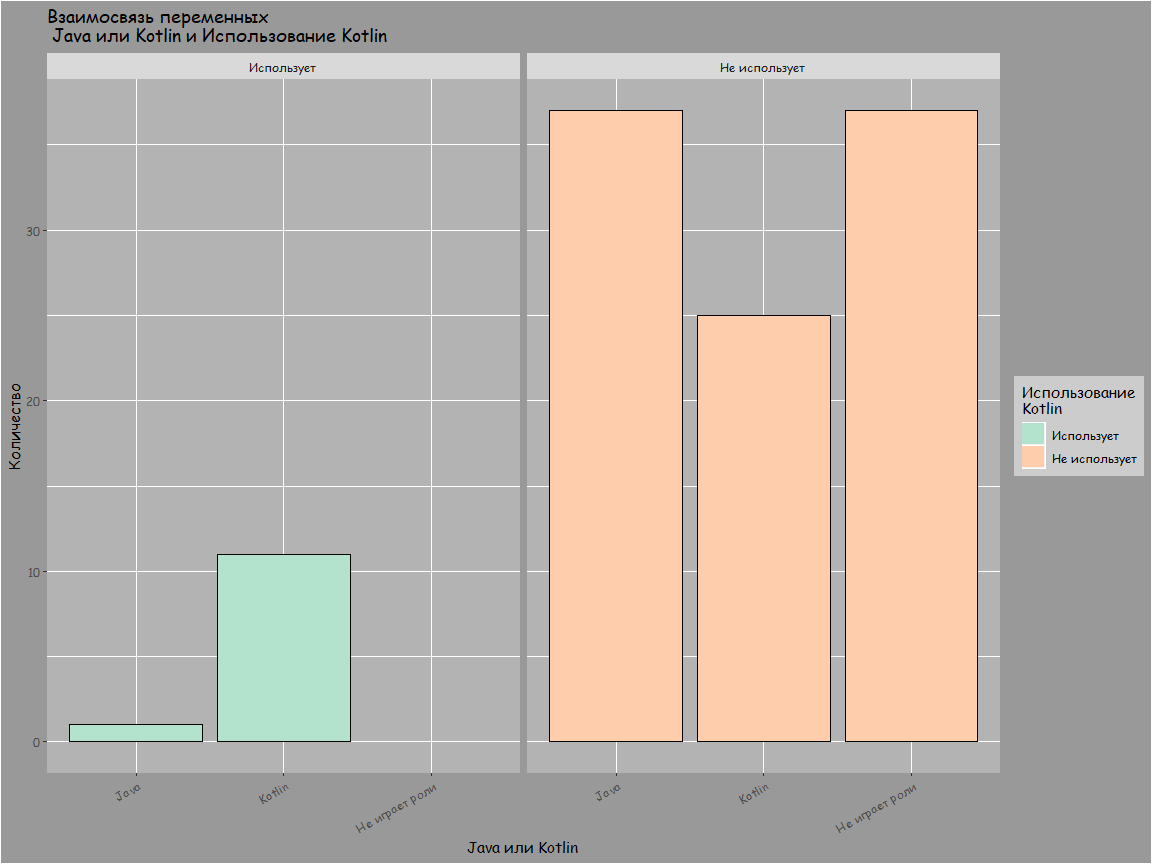
Поздравляю тех, кто дочитал до этого момента! Сейчас начнётся самое интересное…

И ещё. Если названии графика написано: “Взаимосвязь переменных A и B”, то это не значит, что A зависит от B, или B зависит от A. Направление зависимости может быть любым, а также этой взаимосвязи может не существовать вообще :D Но автор старался подобрать *статистически вероятные* зависимости.

Статистическая вероятность существования взаимосвязей проверялась с помощью точного критерия Фишера, корреляции Кендала и критерия Краскела-Уолесса. Статистически вероятной считалась связь, p-уровень значимости которой меньше 0,05.

## Так всё-таки Java или Kotlin?!

### Java или Kotlin и использование Kotlin

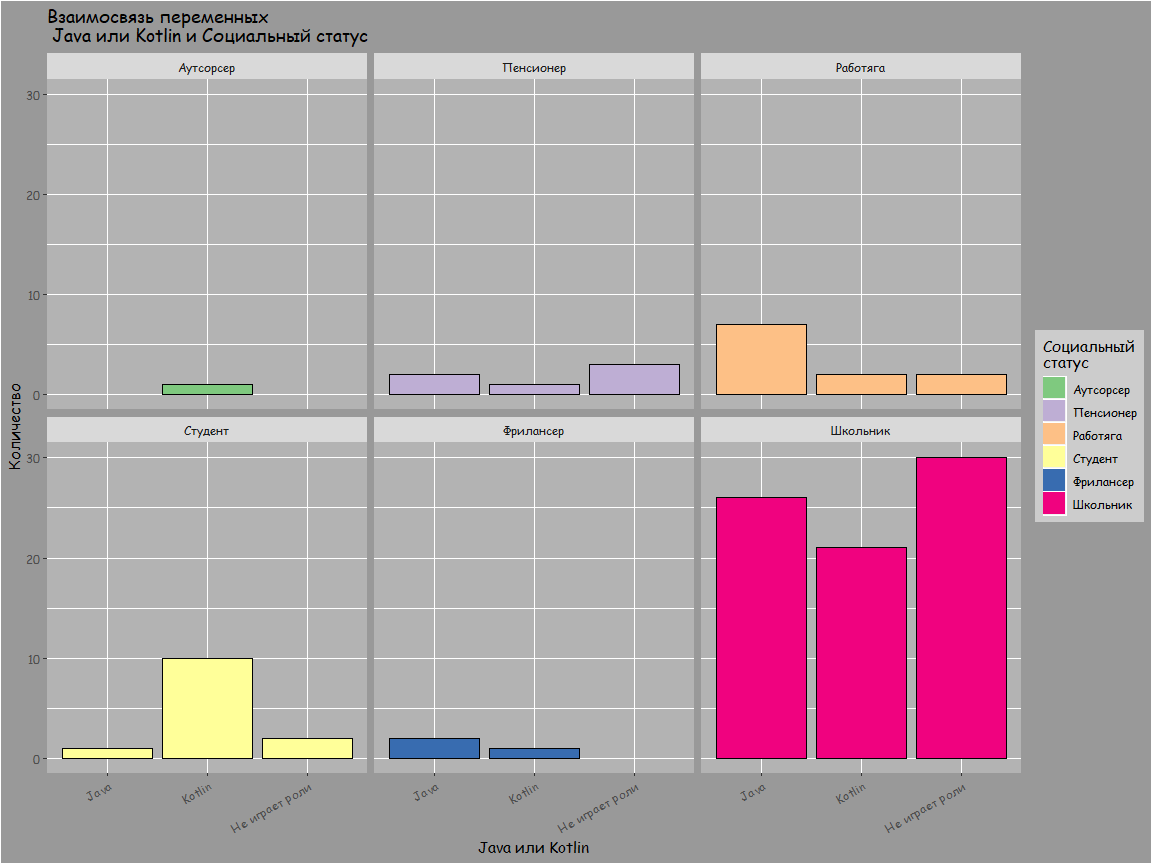


|  |  |  |  |
| --- | --- | --- | --- |
|  | Java | Kotlin | Не играет роли |
| Использует | 0.009 | 0.099 | 0.000 |
| Не использует | 0.333 | 0.225 | 0.333 |

*Выводы*:

1. Ну вот теперь всё ясно! Почти все, кто использовал kotlin, выбрали именно его.
2. Остальная серая масса была вольна говорить всё, что заблагорассудится

### Java или Kotlin и Социальный статус

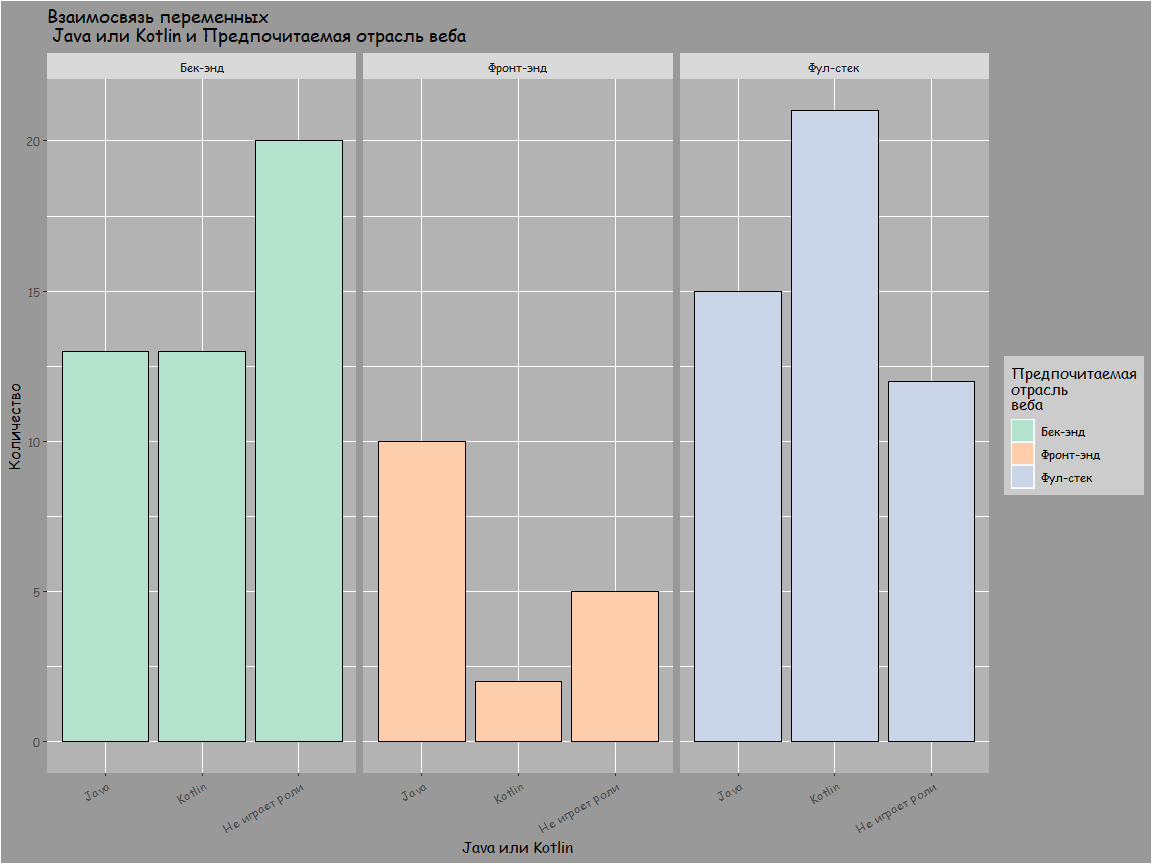


|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Аутсорсер | Пенсионер | Работяга | Студент | Фрилансер | Школьник |
| Java | 0.000 | 0.018 | 0.063 | 0.009 | 0.018 | 0.234 |
| Kotlin | 0.009 | 0.009 | 0.018 | 0.090 | 0.009 | 0.189 |
| Не играет роли | 0.000 | 0.027 | 0.018 | 0.018 | 0.000 | 0.270 |

*Выводы*:

1. Из всей информации на графике надо заметить две вещи. Во-первых, работяги чаще выбирают Java.
2. Во-вторых студенты в большинстве своём выбирают Kotlin.

### Java или Kotlin и Предпочиатемая отрасль веба

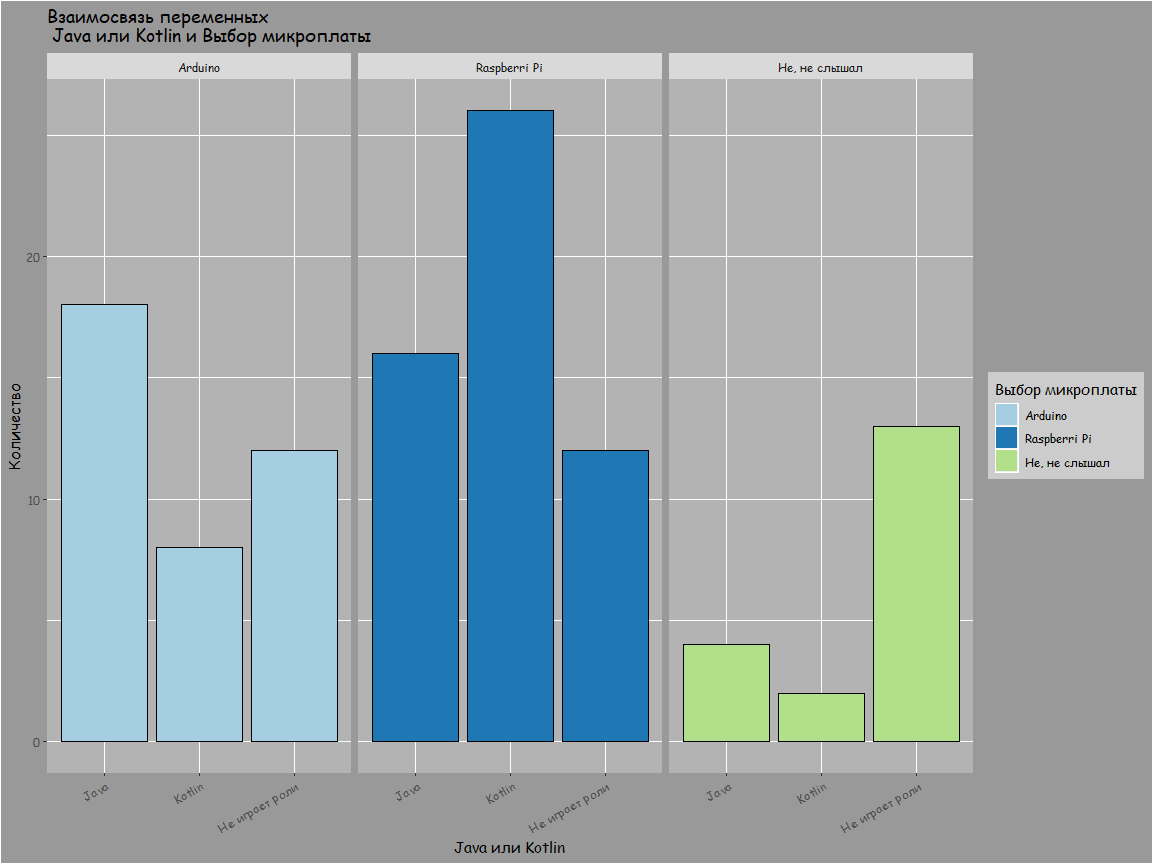


|  |  |  |  |
| --- | --- | --- | --- |
|  | Бек-энд | Фронт-энд | Фул-стек |
| Java | 0.117 | 0.090 | 0.135 |
| Kotlin | 0.117 | 0.018 | 0.189 |
| Не играет роли | 0.180 | 0.045 | 0.108 |

*Выводы*:

1. Нельзя сказать, что бек-эндерам фиолетово, но их мнение весьма лояльно к выбору между собратьями по JVM.
2. Фронт-эндеры предпочитают джаву, непонятно почему
3. Фул-стекеры на стороне Kotlin, но это не точно

### Java или Kotlin и Выбор микроплаты

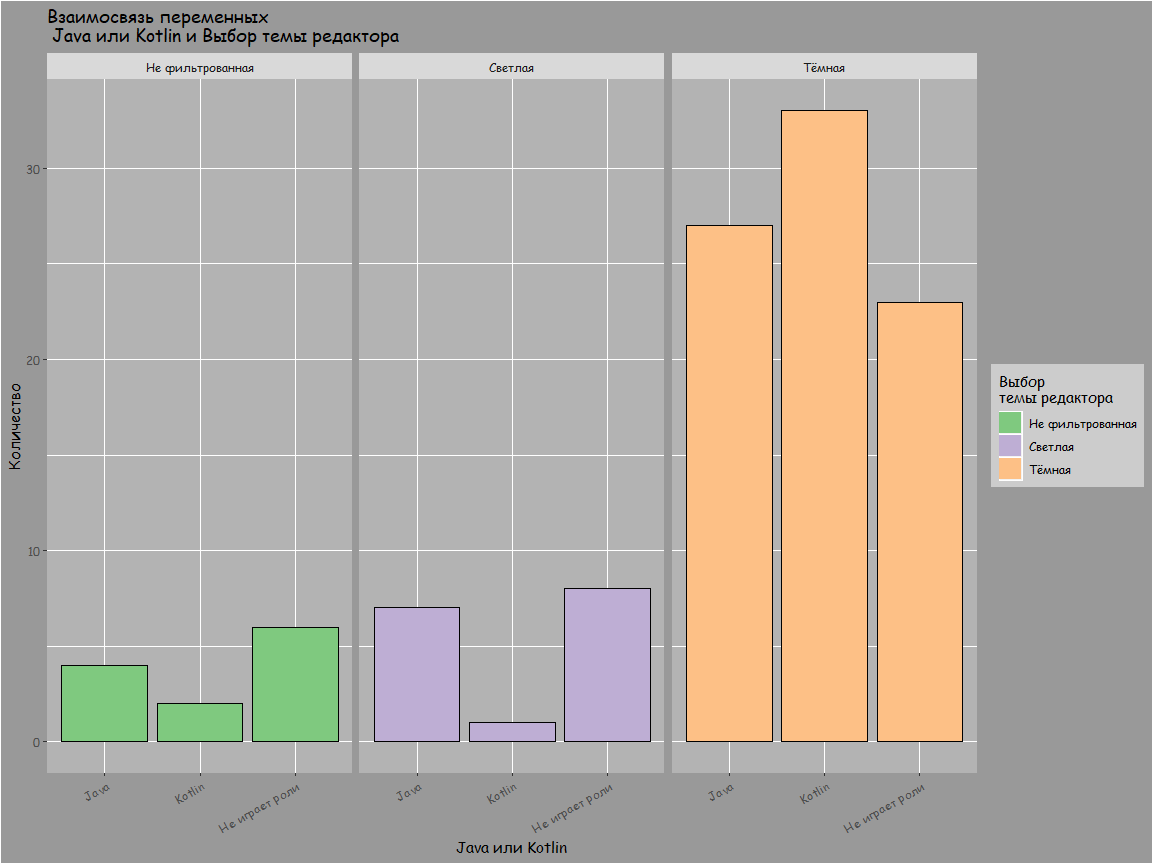


|  |  |  |  |
| --- | --- | --- | --- |
|  | Arduino | Raspberri Pi | Не, не слышал |
| Java | 0.162 | 0.144 | 0.036 |
| Kotlin | 0.072 | 0.234 | 0.018 |
| Не играет роли | 0.108 | 0.108 | 0.117 |

*Выводы*:

1. Выбравшие Arduino более благосколнны к джаве
2. Выбравшие Малину предпочитают котлин
3. Остальным однофигственно

### Java или Kotlin и Тема редактора

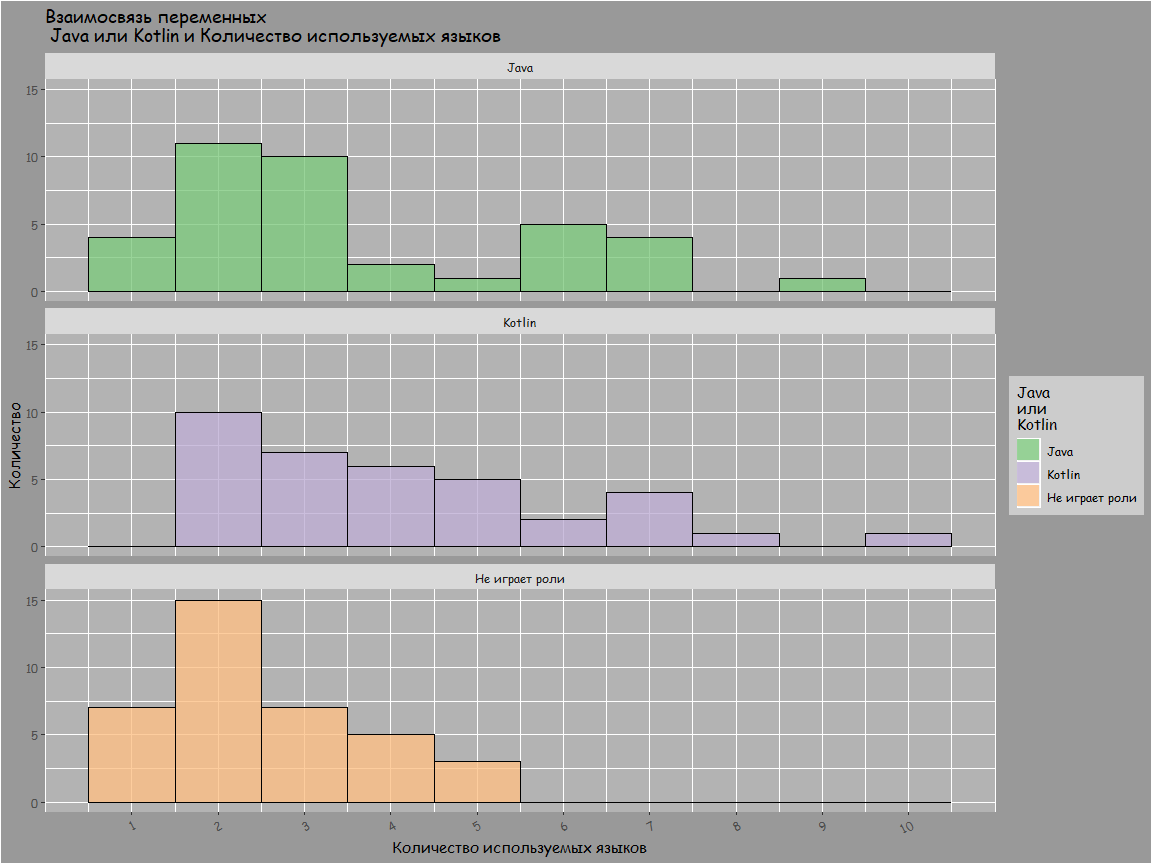


|  |  |  |  |
| --- | --- | --- | --- |
|  | Не фильтрованная | Светлая | Тёмная |
| Java | 0.036 | 0.063 | 0.243 |
| Kotlin | 0.018 | 0.009 | 0.297 |
| Не играет роли | 0.054 | 0.072 | 0.207 |

*Выводы*:

1. Любители светлой темы недолюбливают котлин
2. Несильно намного, но среди “тёмных” лидирует котлин
3. Нефильтрованным не до этого противостояния, видимо под нефильтрованное не заходит

### Java или Kotlin и Количество языков

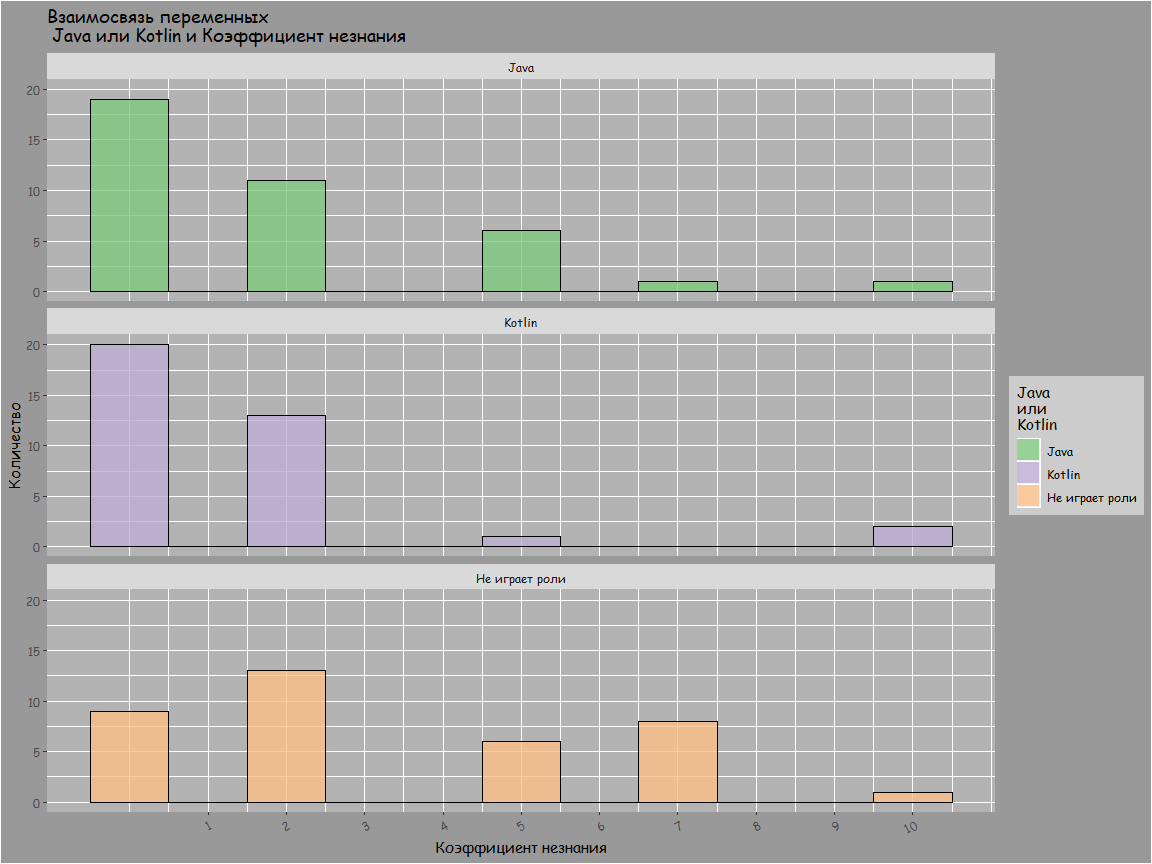


|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Java | 0.036 | 0.099 | 0.090 | 0.018 | 0.009 | 0.045 | 0.036 | 0.000 | 0.009 | 0.000 |
| Kotlin | 0.000 | 0.090 | 0.063 | 0.054 | 0.045 | 0.018 | 0.036 | 0.009 | 0.000 | 0.009 |
| Не играет роли | 0.063 | 0.135 | 0.063 | 0.045 | 0.027 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |

*Выводы*:

1. Выбравшие котлин испоьзуют больше языков
2. Те, кому роли не играет, ~~умом не вышли~~ знают максимум 5 языков
3. Джава идёт среднячком

### Java или Kotlin и Коэффициент незнания



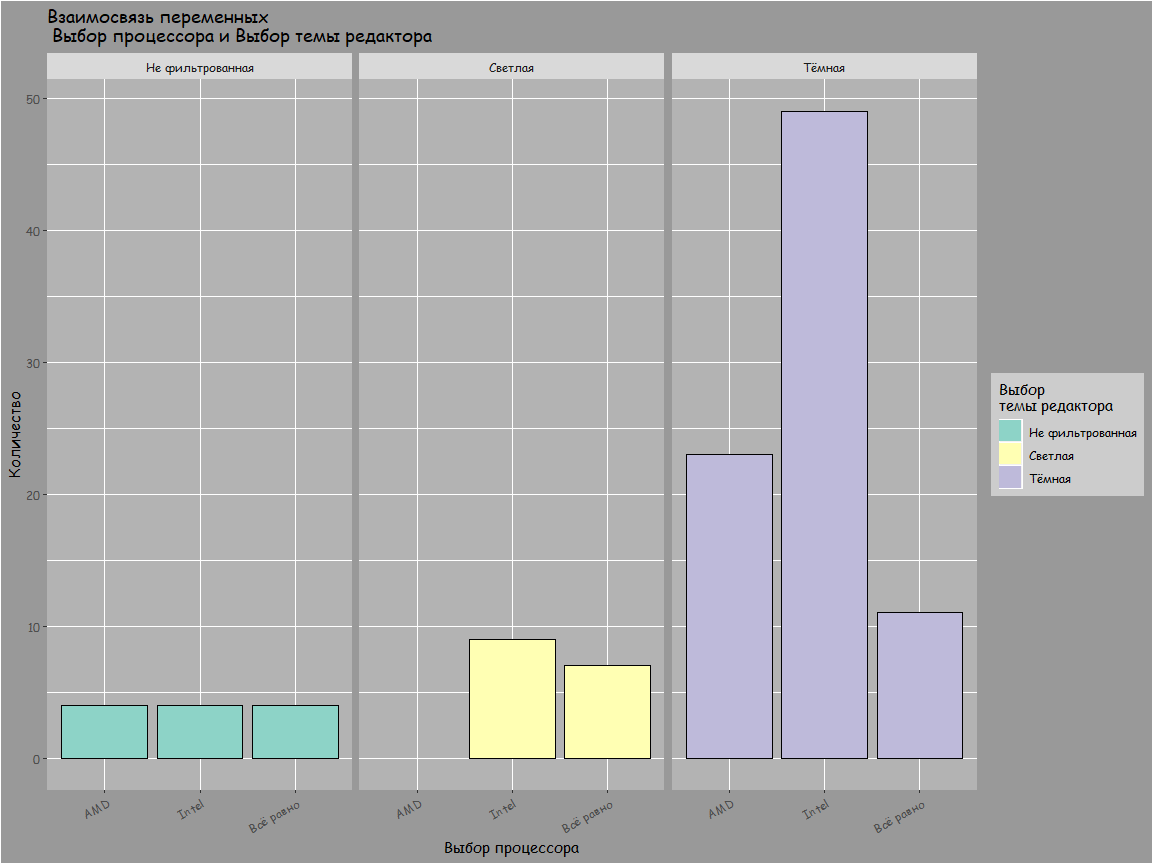
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 2.5 | 5 | 7.5 | 10 |
| Java | 0.171 | 0.099 | 0.054 | 0.009 | 0.009 |
| Kotlin | 0.180 | 0.117 | 0.009 | 0.000 | 0.018 |
| Не играет роли | 0.081 | 0.117 | 0.054 | 0.072 | 0.009 |

*Выводы*:

1. Лишнее доказательство большего уровня “кодовой культурности” среди котлинистов (или котлинёров, или котлинщиков, в общем, не суть)

## Скажи какой процессор, и я скажу, кто ты

### Выбор процессора и тема редактора



|  |  |  |  |
| --- | --- | --- | --- |
|  | Не фильтрованная | Светлая | Тёмная |
| AMD | 0.036 | 0.000 | 0.207 |
| Intel | 0.036 | 0.081 | 0.441 |
| Всё равно | 0.036 | 0.063 | 0.099 |

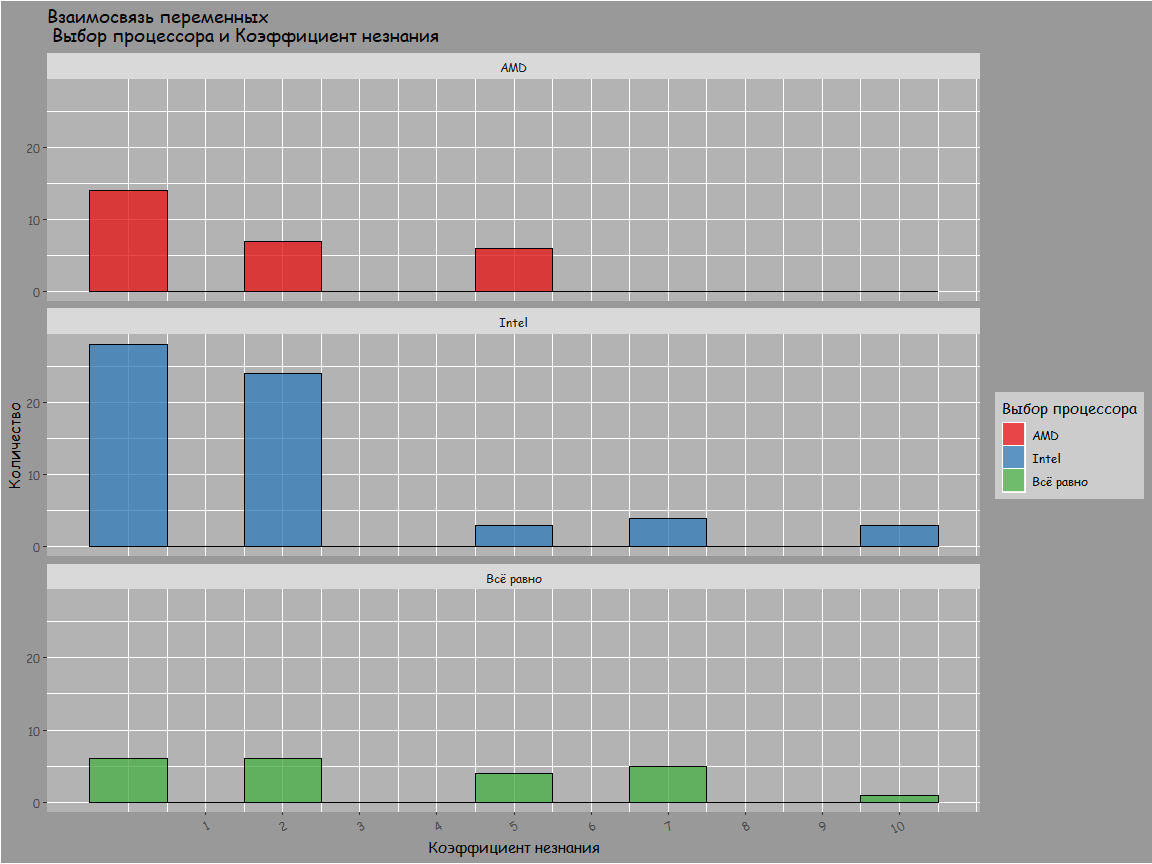
*Примечания*:

1. По состоянию на 2020 год о главенстве Интела говорить не приходится, ведь АМД давно выпустила 16-поточный [Ryzen](https://ru.wikipedia.org/wiki/Ryzen), который снится Интелу. Предпочтение большинством именно Intel можно объяснить заработанным имиджем среди российского айти-пролетариата.

*Выводы*:

1. Любители светлой темы не переваривают AMD. Это уже второе доказательство биологической изолированности этого вида
2. Приспешники Дарта Вейдера топят за Intel
3. В очередной раз автор поражается аморфности этих ~~алкашей~~ любителей нефильтрованной темы. Это даже не мнение большинства, а какой-то рандом.

### Выбор процессора и коэффициент незнания



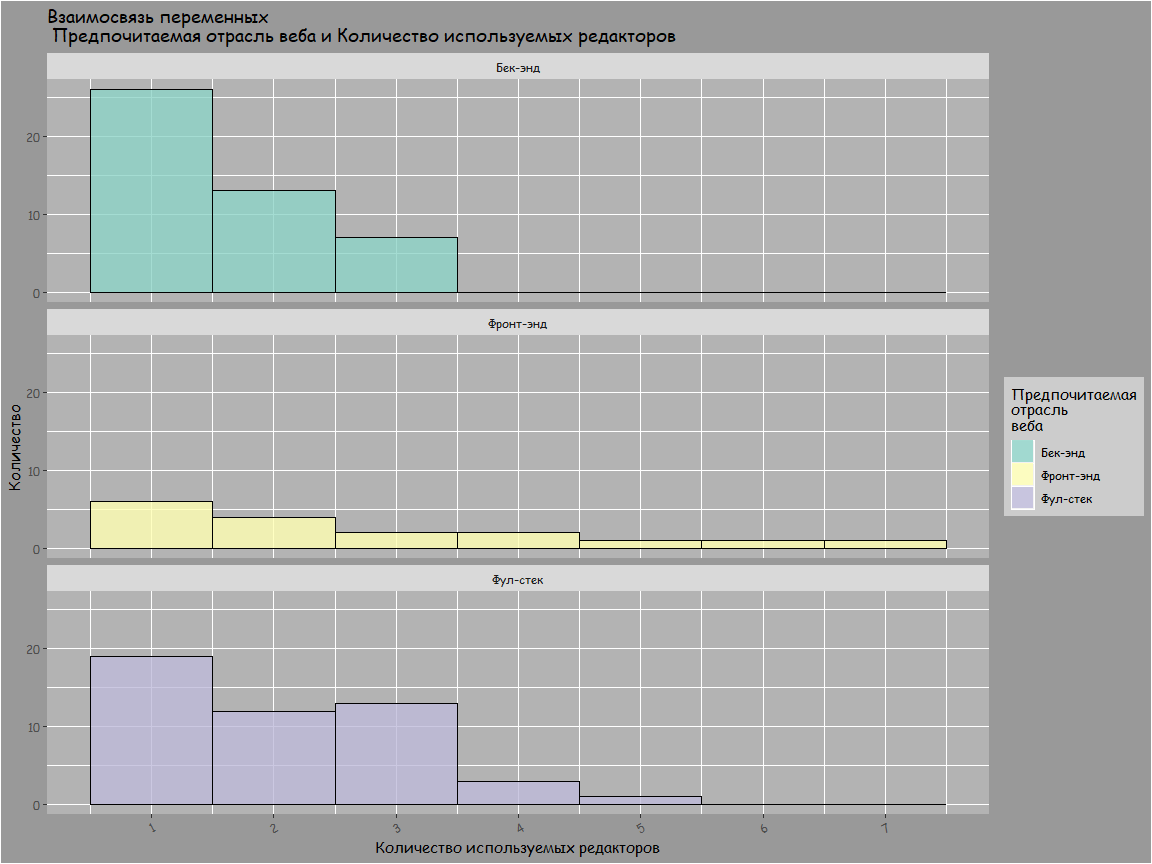
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 2.5 | 5 | 7.5 | 10 |
| AMD | 0.126 | 0.063 | 0.054 | 0.000 | 0.000 |
| Intel | 0.252 | 0.216 | 0.027 | 0.036 | 0.027 |
| Всё равно | 0.054 | 0.054 | 0.036 | 0.045 | 0.009 |

*Выводы*:

1. График говорит сам за себя. Предпочитающие AMD отличаются высокой программной грамотностью.
2. У интелистов тоже неплохой результат, но замечены выбросы в сторону полного незнания.

## Статистика и Тайны веба

### Отрасль веба и количество редакторов

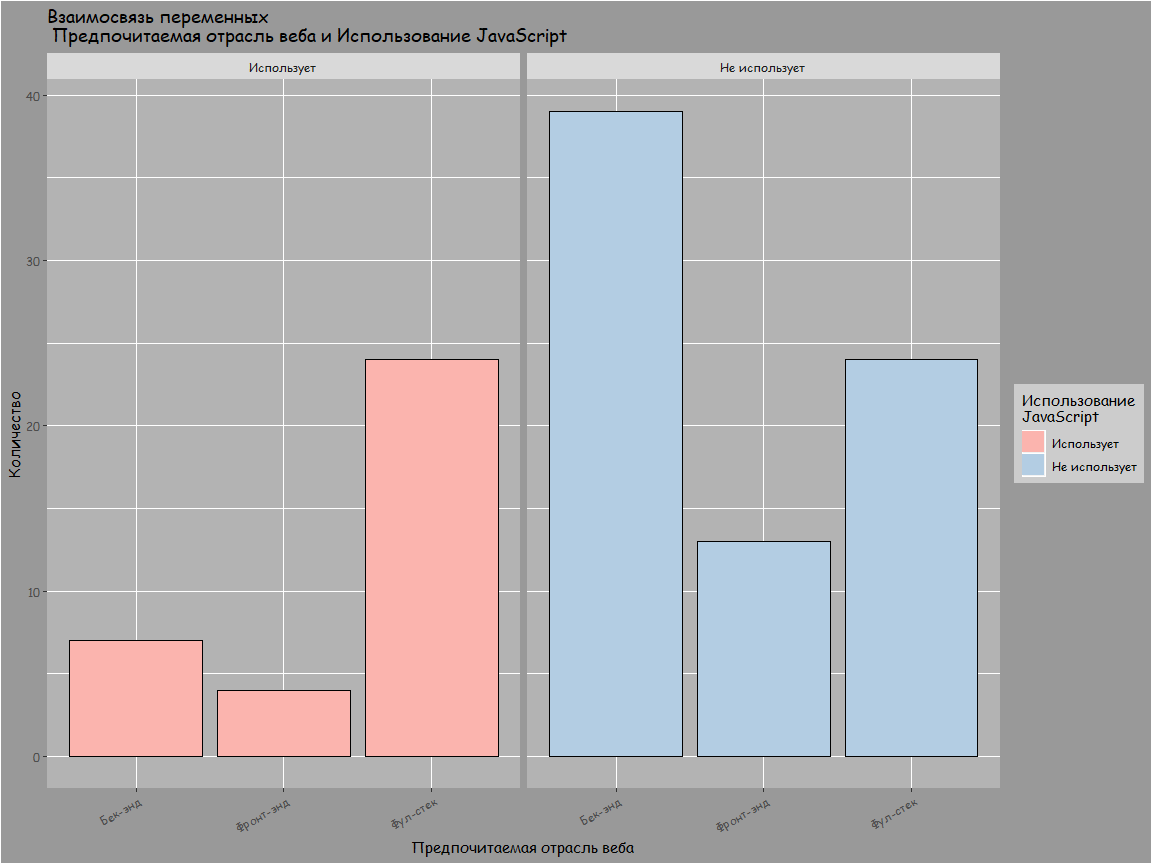


|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Бек-энд | 0.234 | 0.117 | 0.063 | 0.000 | 0.000 | 0.000 | 0.000 |
| Фронт-энд | 0.054 | 0.036 | 0.018 | 0.018 | 0.009 | 0.009 | 0.009 |
| Фул-стек | 0.171 | 0.108 | 0.117 | 0.027 | 0.009 | 0.000 | 0.000 |

*Выводы*:

1. Бек-эндерам (заднеконечным, в дословном переводе) трёх редакторов кода для полного счастья хватает
2. Запросы фул-стекеров немного выше - до 5 редакторов
3. Передне-конечные (они же фронт-эндеры) буквально “размазаны” вдоль оси абсцисс. Что не удивительно, код редактировать приходится на совершенно разных наречиях: от html и css до php и javascript. Хотя WebStorm и PhpStorm от JetBrains неплохо включают в себя перечисленный функционал, о них речь не шла.

### Отрасль веба и использование JavaScript

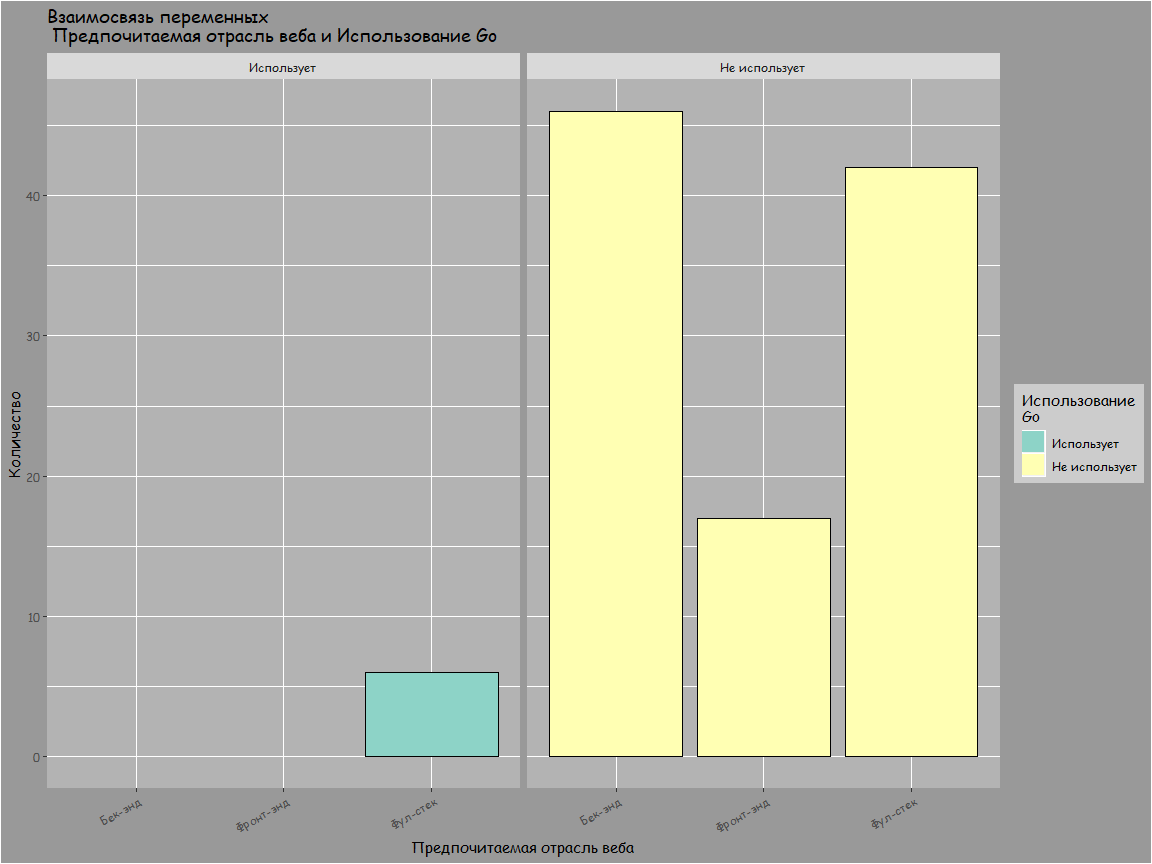


|  |  |  |  |
| --- | --- | --- | --- |
|  | Бек-энд | Фронт-энд | Фул-стек |
| Использует | 0.063 | 0.036 | 0.216 |
| Не использует | 0.351 | 0.117 | 0.216 |

*Выводы*:

1. Использование JavaScript весьма характерно для фул-стекеров. Он зарекомендовал себя как язык по обе стороны веба
2. Однако, как ни странно, достаточно малая доля разработчиков клиентской части использует недоджаву-перескрипт (он же javascript). Возможно, стандартных бустраповских форм и скриптов им вполне хватает.

### Отрасль веба и использование Go

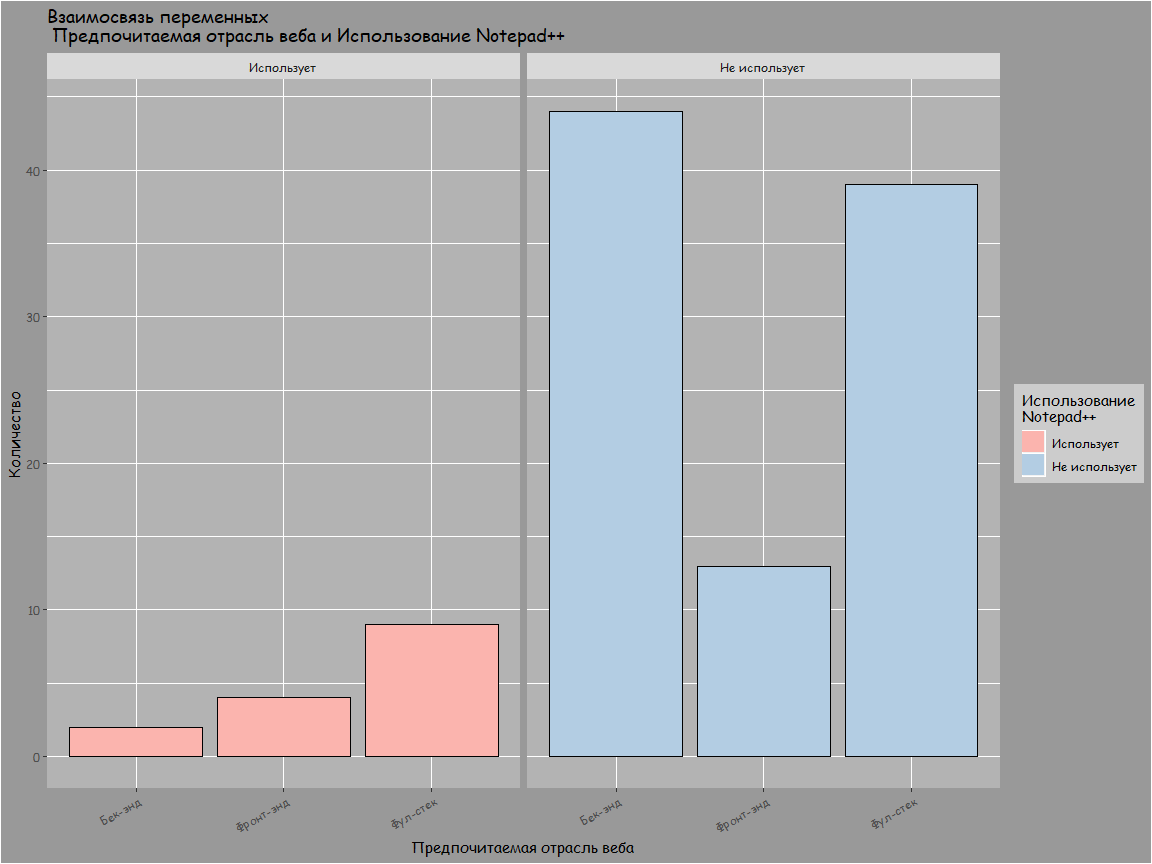


|  |  |  |  |
| --- | --- | --- | --- |
|  | Бек-энд | Фронт-энд | Фул-стек |
| Использует | 0.000 | 0.000 | 0.054 |
| Не использует | 0.414 | 0.153 | 0.378 |

*Выводы*:

1. Непонятно с чем это связано, но использующие Go программисты предпочитают только фул-стек. Наверное, это почётные кавалеры ордена программного кода, знающие что и как кодить.

### Отрасль веба и использование Notepad++

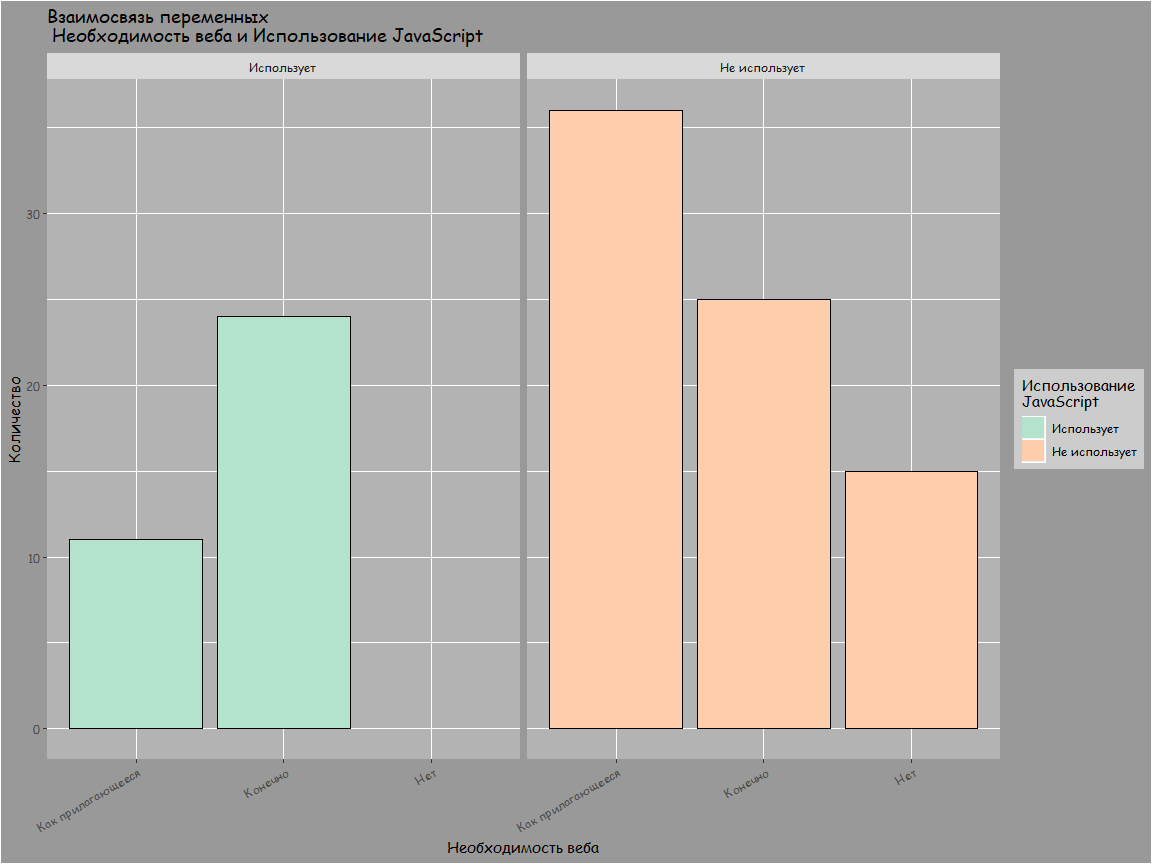


|  |  |  |  |
| --- | --- | --- | --- |
|  | Бек-энд | Фронт-энд | Фул-стек |
| Использует | 0.018 | 0.036 | 0.081 |
| Не использует | 0.396 | 0.117 | 0.351 |

*Выводы*:

1. Унтер-блокнот (он же notepad++) имеет большее влияние на разработчиков фронт-энда и фул-стека, нежели бек-энда

### Необходимость веба и использование JavaScript

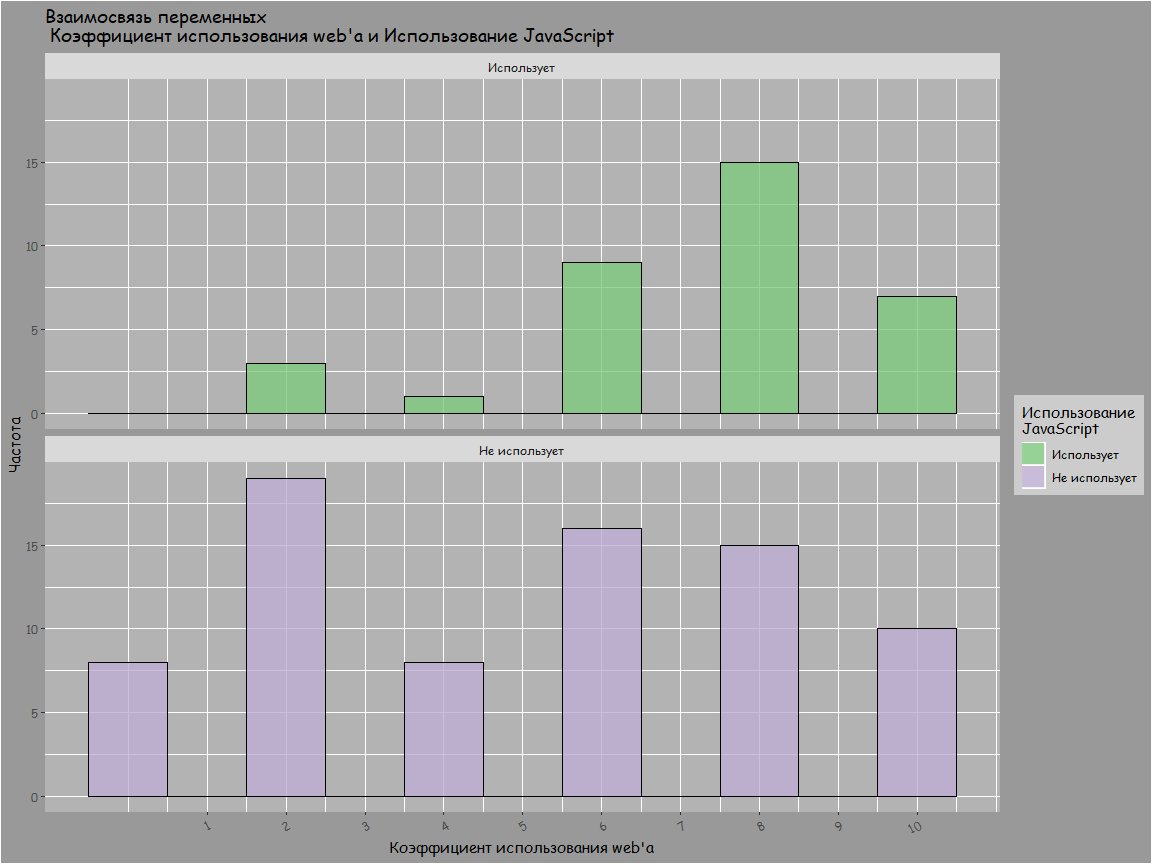


|  |  |  |  |
| --- | --- | --- | --- |
|  | Как прилагающееся | Конечно | Нет |
| Использует | 0.099 | 0.216 | 0.000 |
| Не использует | 0.324 | 0.225 | 0.135 |

*Выводы*:

1. Самое странное в графиках - когда на месте столбца гистограммы ничего нет. Не знаю, какая секта проповедует JS, но её участники свято верят в необходимость веба.

### Использование веба и использование JavaScript

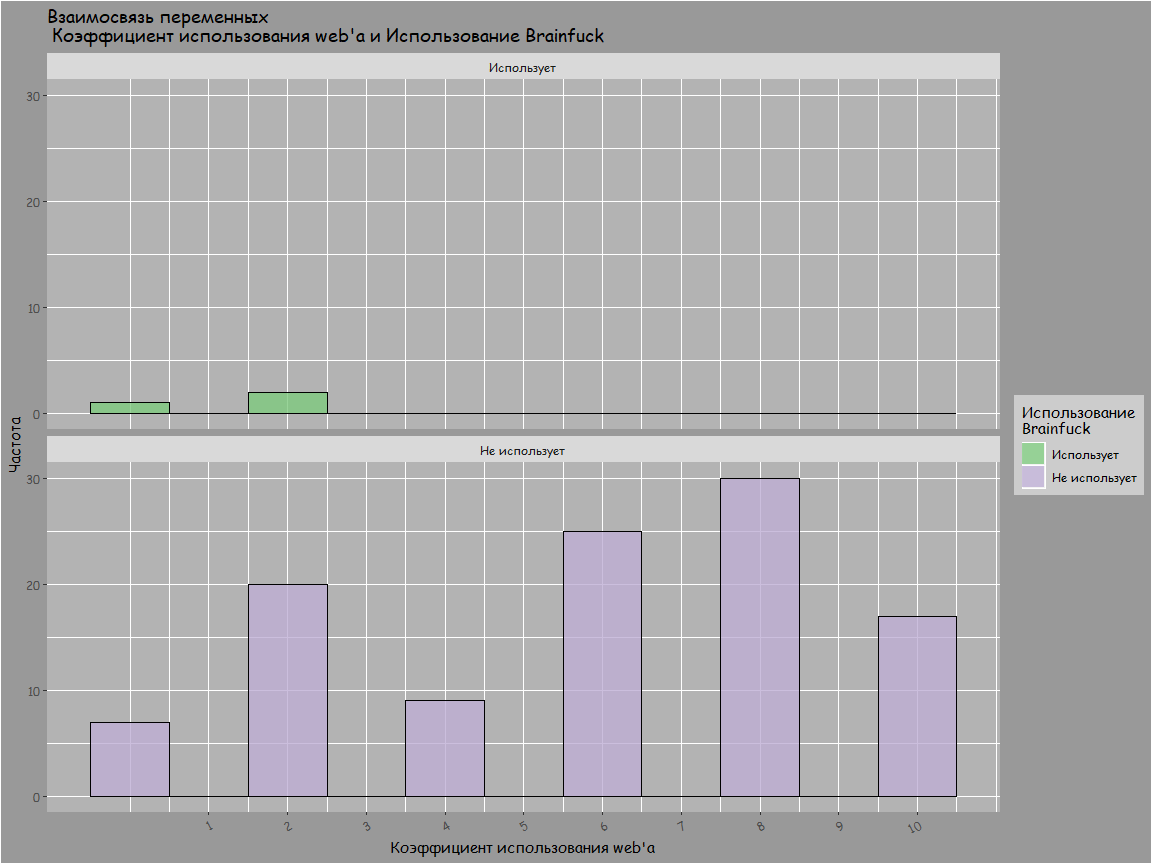


|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 2 | 4 | 6 | 8 | 10 |
| Использует | 0.000 | 0.027 | 0.009 | 0.081 | 0.135 | 0.063 |
| Не использует | 0.072 | 0.171 | 0.072 | 0.144 | 0.135 | 0.090 |

*Выводы*:

1. Знание JavaScript однозначно повышает интернет-грамотность программиста
2. Однако, автор бы не сказал, что оно прямо-таки обязательно для ориентации в пространстве WWW.

### Использование веба и использование Brainfuck

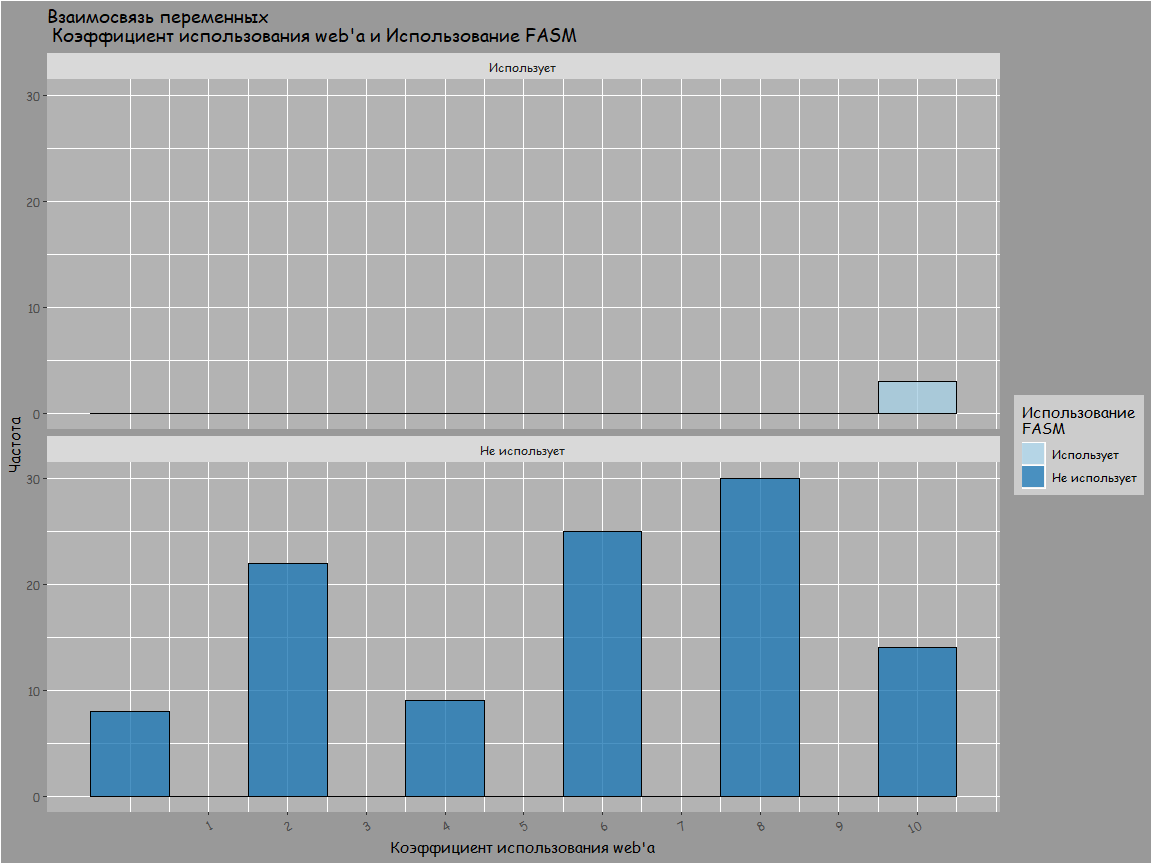


|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 2 | 4 | 6 | 8 | 10 |
| Использует | 0.009 | 0.018 | 0.000 | 0.000 | 0.00 | 0.000 |
| Не использует | 0.063 | 0.180 | 0.081 | 0.225 | 0.27 | 0.153 |

*Выводы*:

1. Если веб-разработчик использует brainfuck, то это говорит о нём не с лучшей стороны.

### Использование веба и использование FASM

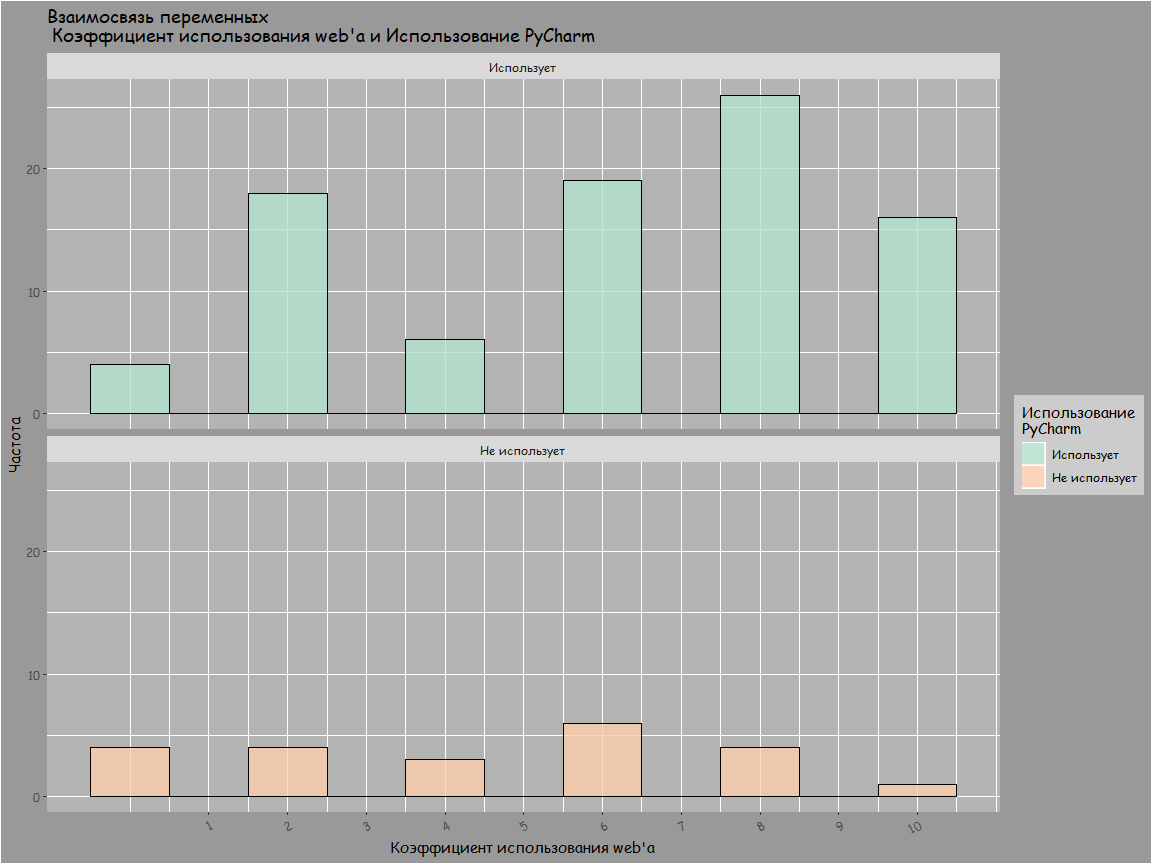


|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 2 | 4 | 6 | 8 | 10 |
| Использует | 0.000 | 0.000 | 0.000 | 0.000 | 0.00 | 0.027 |
| Не использует | 0.072 | 0.198 | 0.081 | 0.225 | 0.27 | 0.126 |

*Выводы*:

1. Совершенно необяснимое явление. Но, судя по данным, использование ассемблера превращает программиста в знатока веб-разработки.

### Использование веба и нужность PyCharm



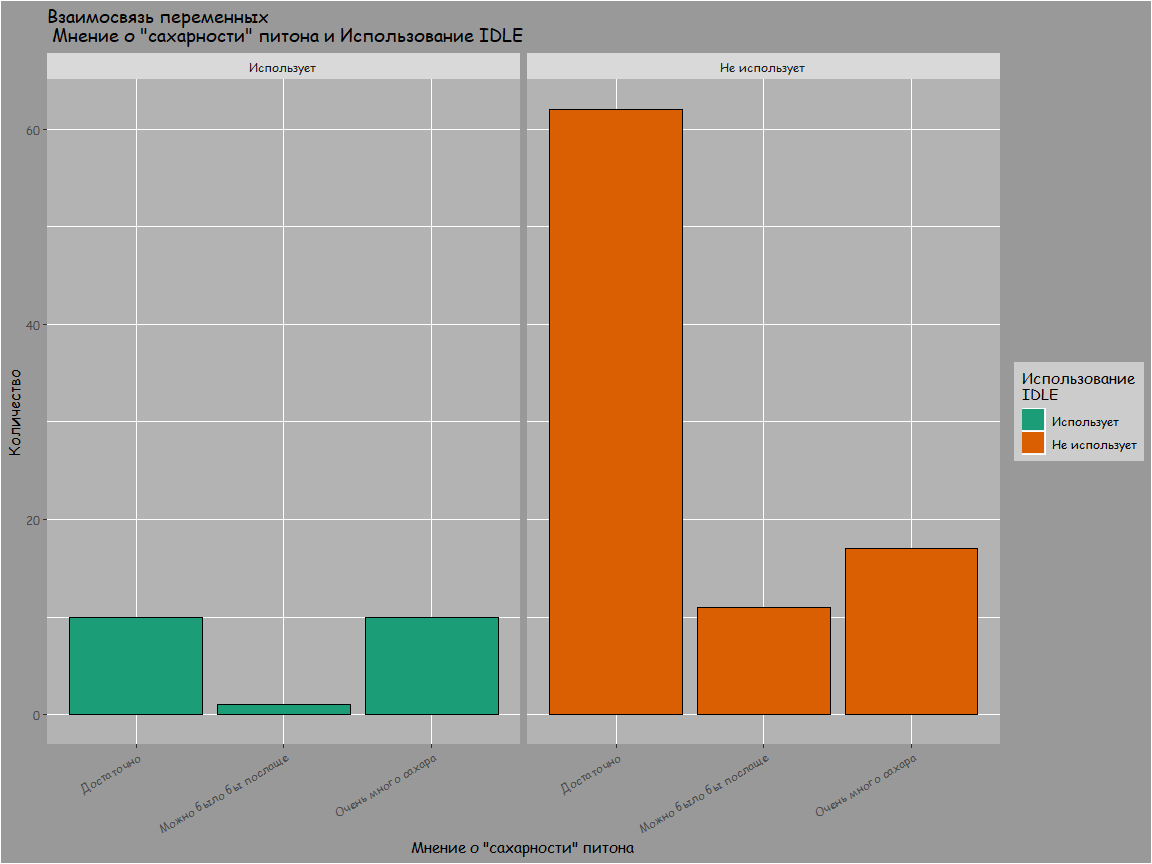
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 2 | 4 | 6 | 8 | 10 |
| Использует | 0.036 | 0.162 | 0.054 | 0.171 | 0.234 | 0.144 |
| Не использует | 0.036 | 0.036 | 0.027 | 0.054 | 0.036 | 0.009 |

*Выводы*:

1. Не совсем очевидная зависимость, но использование пайчарма увеличивает грамотность программиста в вебе.

## Питон. Быстрее медленного

### Мнение о сахарности питона и использование IDLE

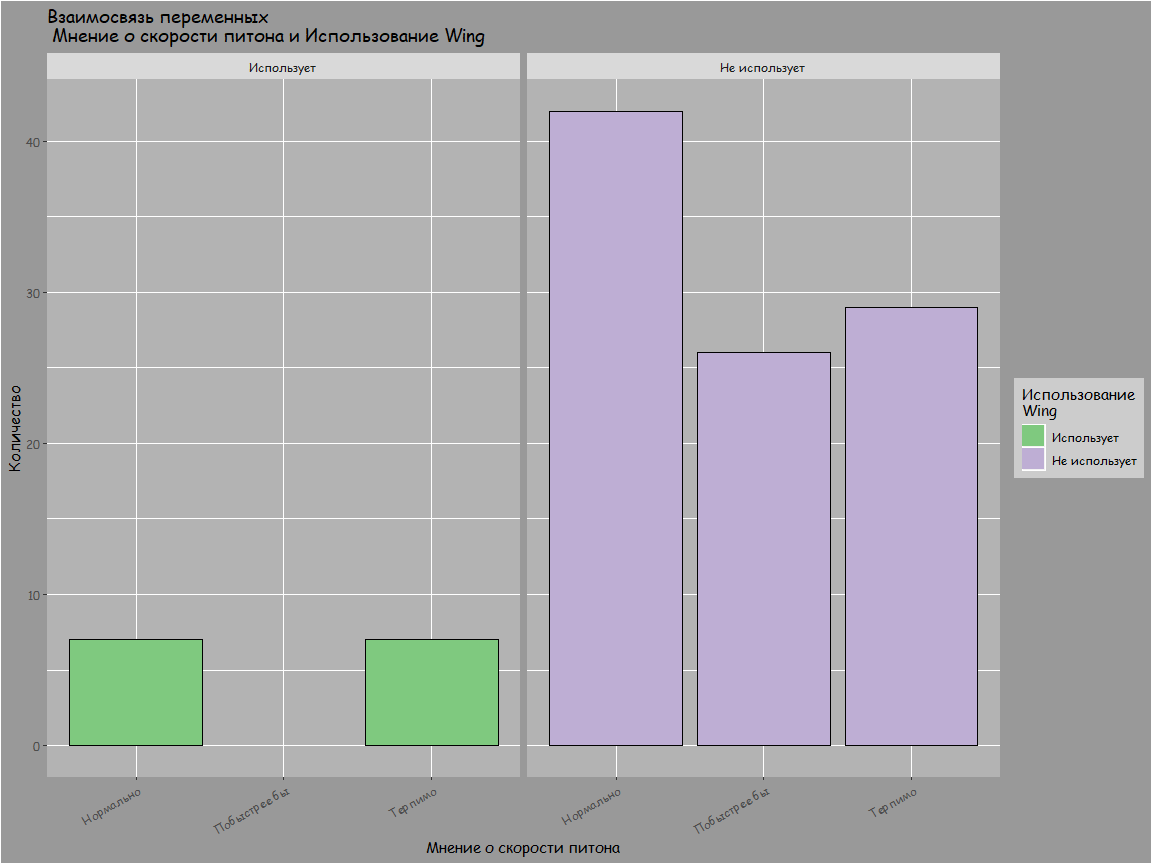


|  |  |  |  |
| --- | --- | --- | --- |
|  | Достаточно | Можно было бы послаще | Очень много сахара |
| Использует | 0.090 | 0.009 | 0.090 |
| Не использует | 0.559 | 0.099 | 0.153 |

*Выводы*:

1. Ясно прослеживается, что пользователи IDLE более довольны питоном и его возможностями.
2. Вывод прост: если программист использует идлу, то это либо новичок, которому всего не только хватает, но ещё и много, либо это бородатый кодер, обожающий минимализм и варианты из коробки.

### Мнение о скорости питона и использование Wing



|  |  |  |  |
| --- | --- | --- | --- |
|  | Нормально | Побыстрее бы | Терпимо |
| Использует | 0.063 | 0.000 | 0.063 |
| Не использует | 0.378 | 0.234 | 0.261 |

*Выводы*:

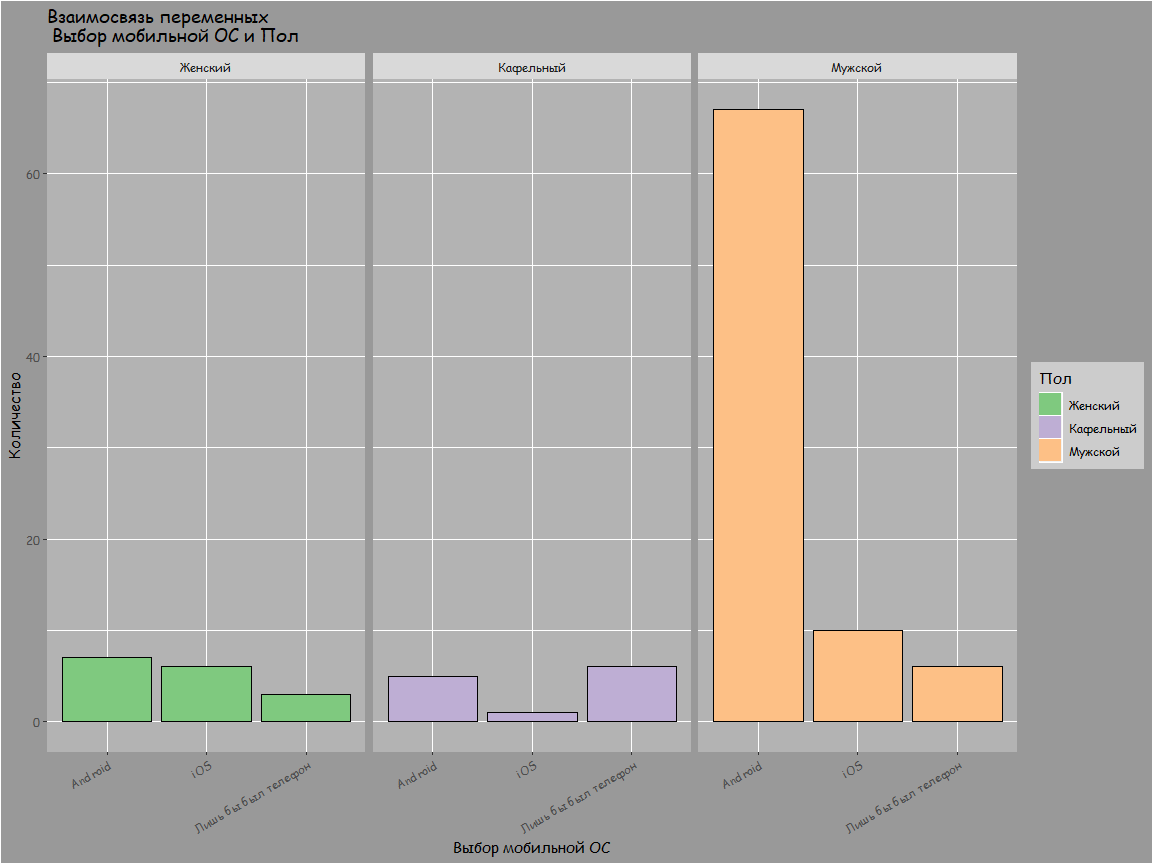
1. Отмечено полное удовлетворение скоростью питона пользователей wing.
2. Здесь два варианта: либо wing разгоняет питон до невиданных скоростей, либо программисты, пишущие в винге не ставят перед ним сверхсложных задач (например, трилионный перебор). Автор склоняется ко второму выводу.

## О классовом разделении по признаку операционки

Оказывается, о человеке (программисте) можно многое сказать, зная его мобильную и настольную операционную систему.

### *Пол и мобильная ОС*

Ради этой зависимости стоило делать исследование!

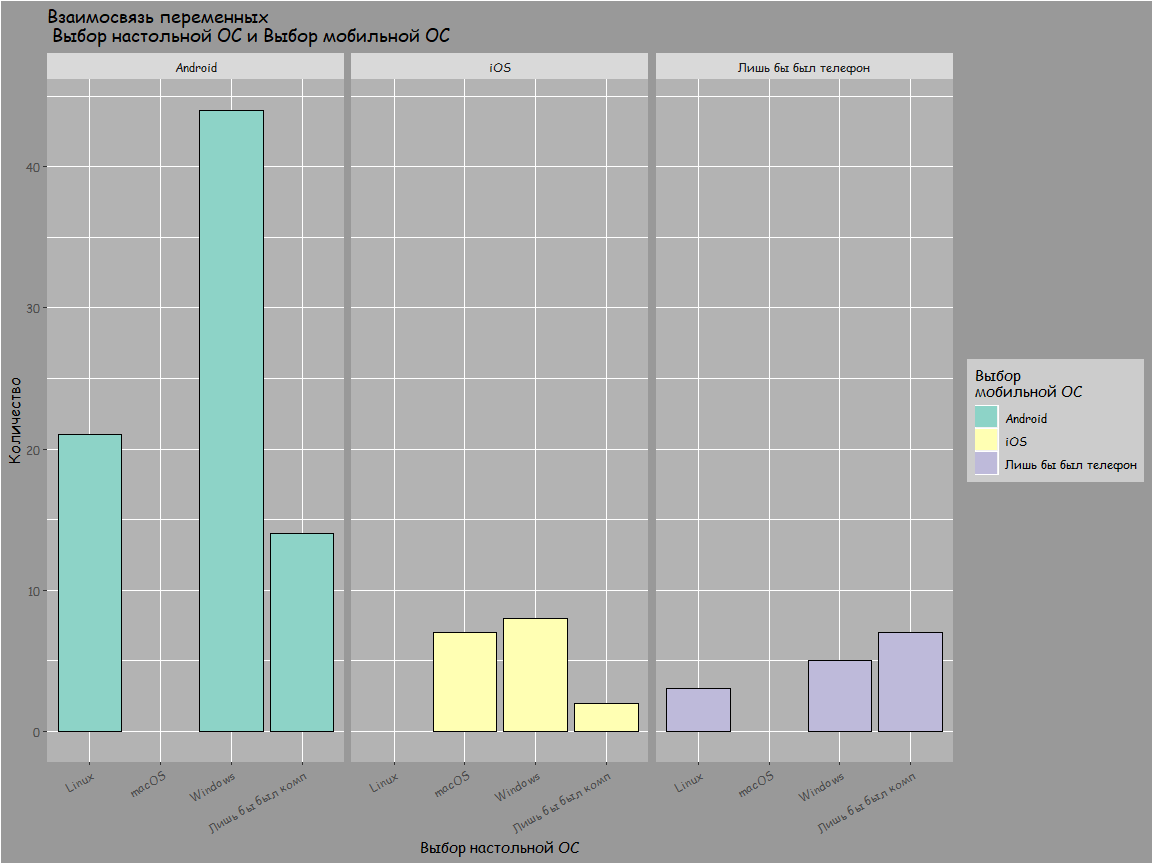


|  |  |  |  |
| --- | --- | --- | --- |
|  | Женский | Кафельный | Мужской |
| Android | 0.063 | 0.045 | 0.604 |
| iOS | 0.054 | 0.009 | 0.090 |
| Лишь бы был телефон | 0.027 | 0.054 | 0.054 |

*Выводы*:

1. Вот вы и попались, девочки! Замечена почти половинная симпатия среди женского пола к iOS, чем не могут похвастаться ни мужской, ни тем более кафельный пол.
2. Так как единственной маркой, обладающей яблочной операционкой является Apple, то весьма прослеживаются элитарно-буржуйские замашки слабого пола. После этого вывода, следует считать российский феминизм лишённым всякой цели, кроме окончательного порабощения андроидовских работяг.

### Настольная ОС и мобильная ОС

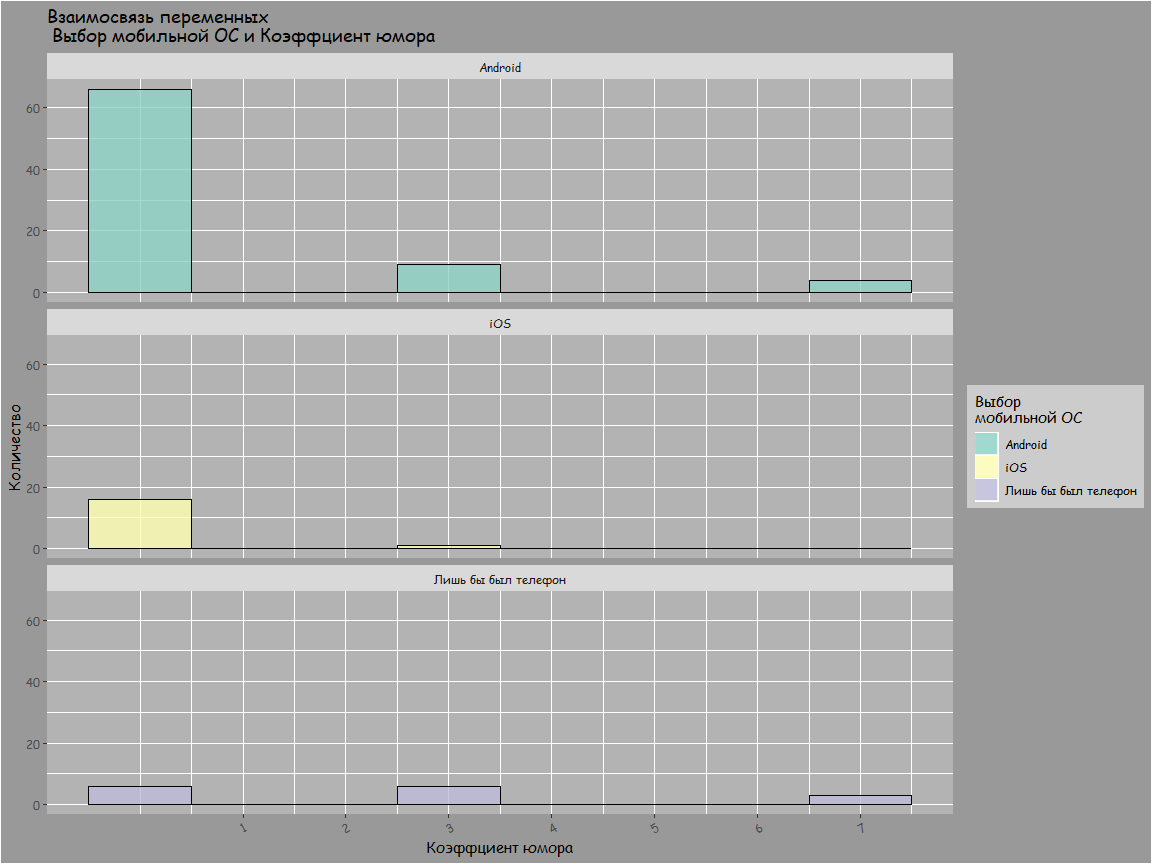


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Linux | macOS | Windows | Лишь бы был комп |
| Android | 0.189 | 0.000 | 0.396 | 0.126 |
| iOS | 0.000 | 0.063 | 0.072 | 0.018 |
| Лишь бы был телефон | 0.027 | 0.000 | 0.045 | 0.063 |

*Выводы*:

1. Стоит отметить, что все предпочитатели macOS, к тому же, выбрали iOS. Нормальные люди, избежали такого выбора.
2. Любители айфонов не переваривают линух ни в каких его проявлениях. Это лишний раз доказывает эндемичность и биологическую изоляцию яблоко-людей.

### Мобильная ОС и коэффициент юмора

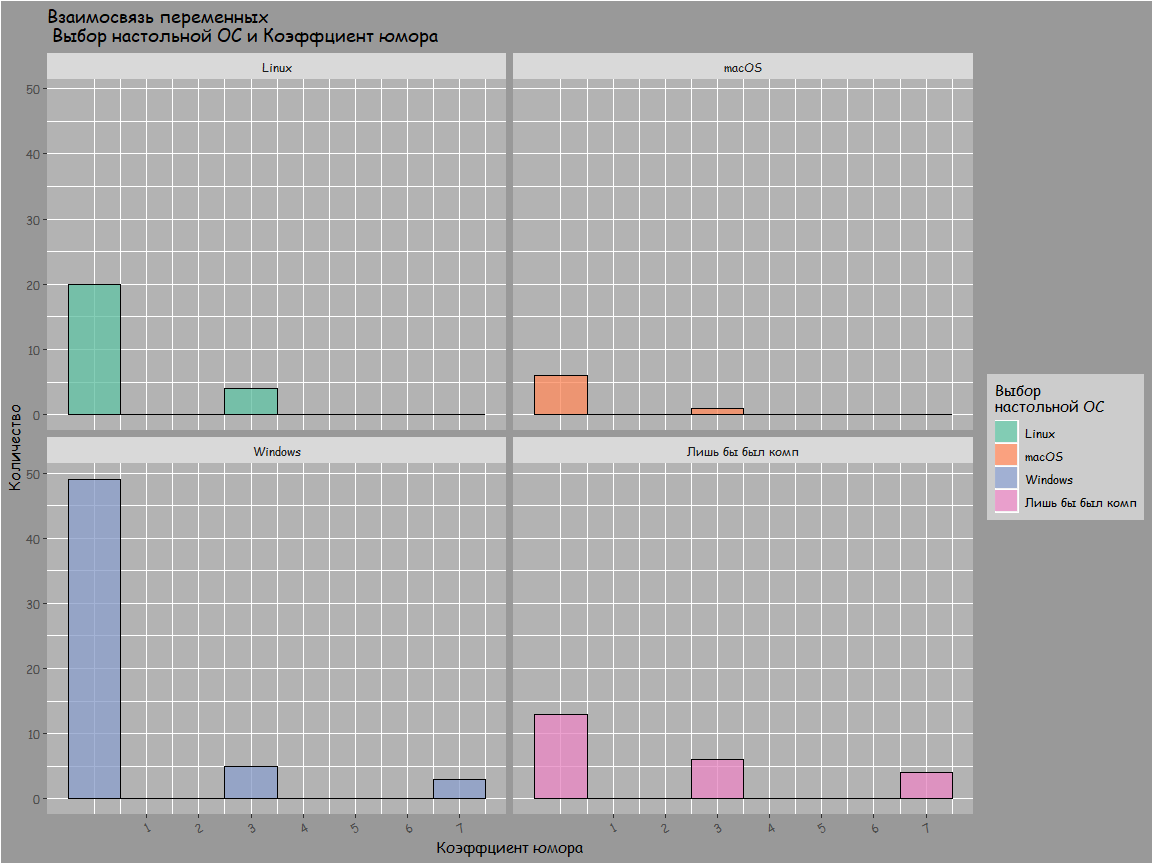


|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 3.3 | 6.7 |
| Android | 0.595 | 0.081 | 0.036 |
| iOS | 0.144 | 0.009 | 0.000 |
| Лишь бы был телефон | 0.054 | 0.054 | 0.027 |

*Выводы*:

1. Пофигисты заняли первое место по условному коэффициенту юмора
2. Пользователи iOS оказались самыми не смешливыми. Скорее всего, жизнь покалечила.

### Настольная ОС и коэффициент юмора



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Linux | macOS | Windows | Лишь бы был комп |
| 0 | 0.180 | 0.054 | 0.441 | 0.117 |
| 3.3 | 0.036 | 0.009 | 0.045 | 0.054 |
| 6.7 | 0.000 | 0.000 | 0.027 | 0.036 |

*Примечания*:

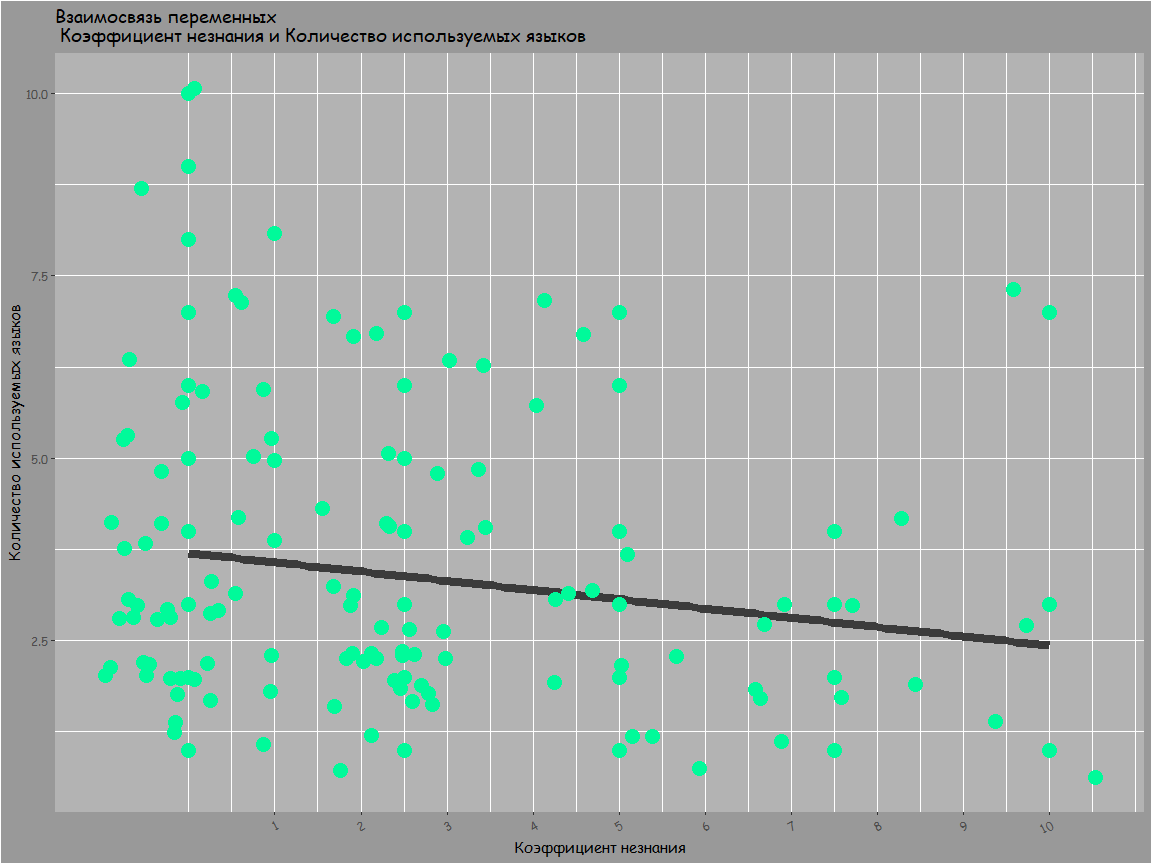
1. Показатель юмора оказался очень субъективным и неочевидным признаком. Возможно, это всего лишь маргинальность и безответственность опрошенных при заполнении анкеты.

*Выводы*:

1. Недопетросяны собрались среди пользователей Окон, а также среди лояльной серой массы.
2. Пользователи Линукса и Мака показали не очень высокий показатель юмора.

## Определение знаний программиста без вскрытия черепа

### Коэффициент незнания и количество языков

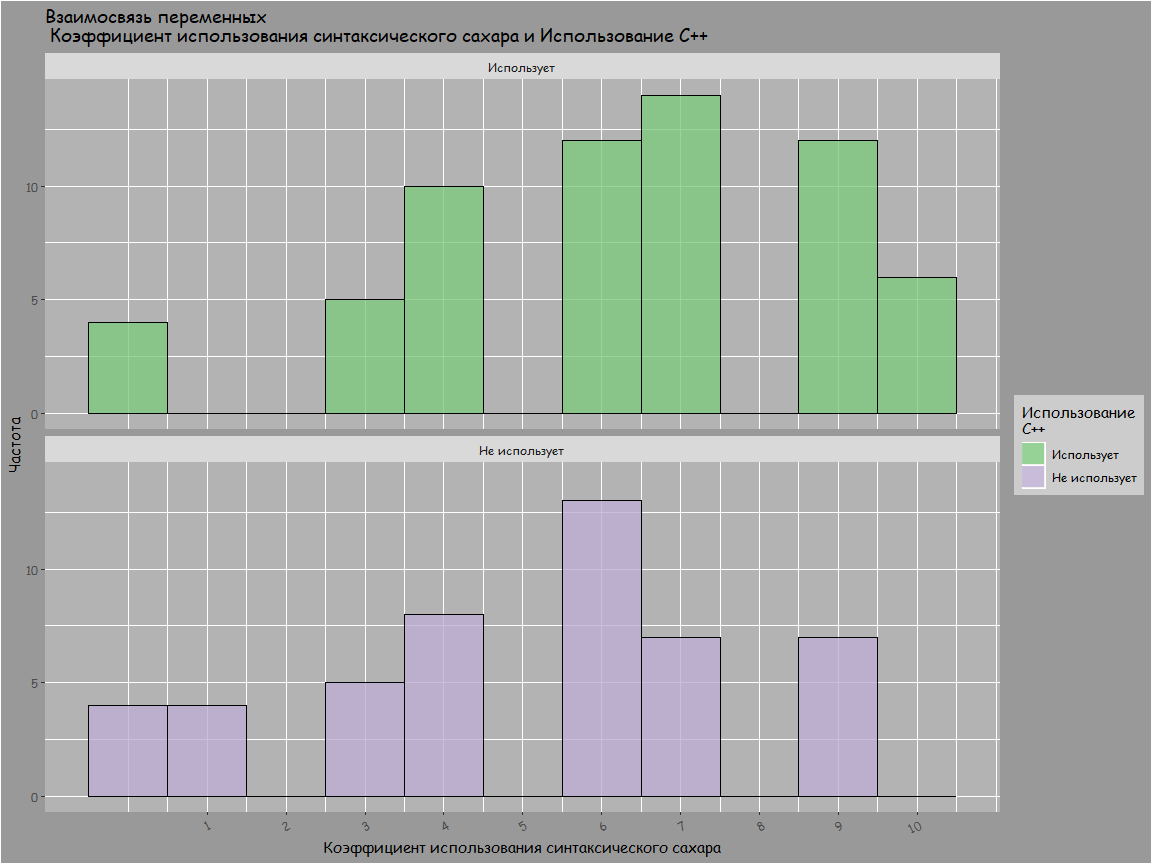


|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 | 0.027 | 0.117 | 0.099 | 0.054 | 0.054 | 0.036 | 0.018 | 0.009 | 0.009 | 0.009 |
| 2.5 | 0.018 | 0.144 | 0.054 | 0.045 | 0.027 | 0.018 | 0.027 | 0.000 | 0.000 | 0.000 |
| 5 | 0.027 | 0.027 | 0.027 | 0.009 | 0.000 | 0.009 | 0.018 | 0.000 | 0.000 | 0.000 |
| 7.5 | 0.009 | 0.036 | 0.027 | 0.009 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 10 | 0.018 | 0.000 | 0.009 | 0.000 | 0.000 | 0.000 | 0.009 | 0.000 | 0.000 | 0.000 |

*Выводы*:

1. Чем больше программист знает языков, тем больше он дал взыешенных ответов. И наоборот - чем больше программист дал взвешенных ответов, тем больше языков он знает.

### Использование сахара и использование С++

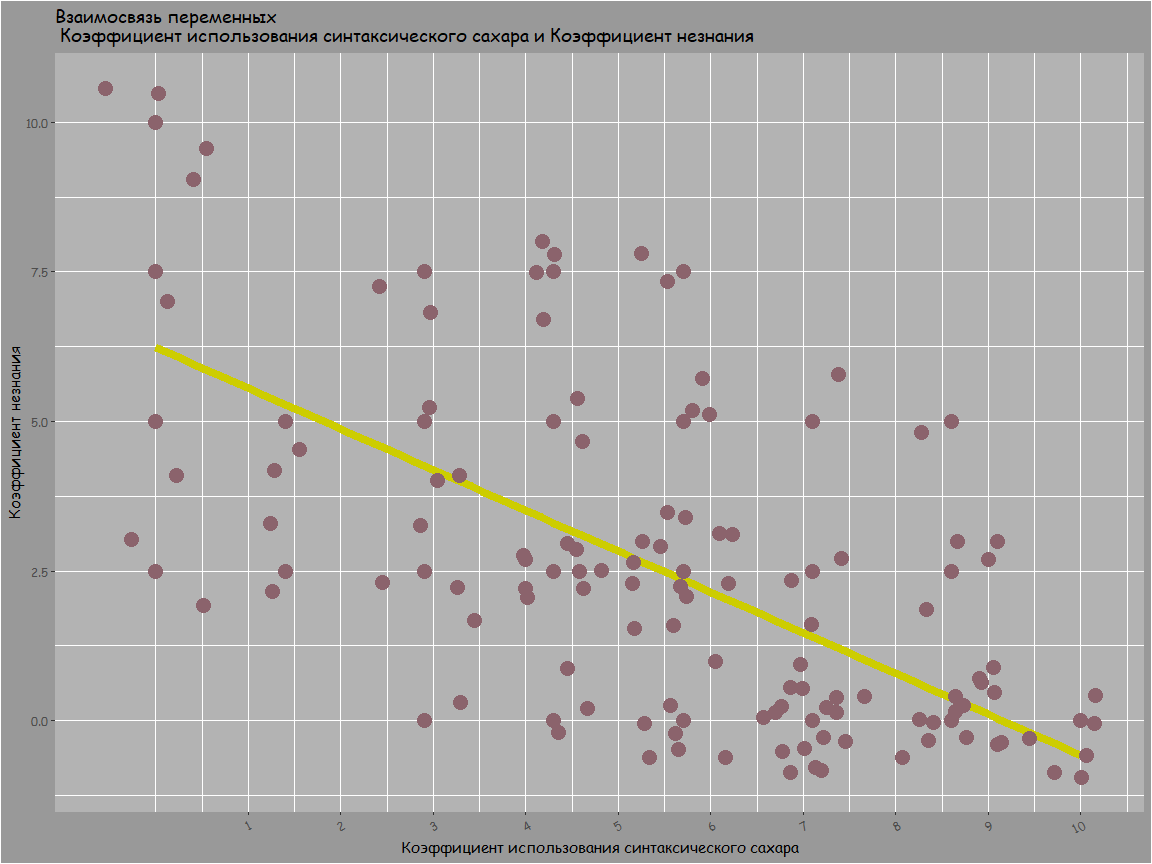


|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1.4 | 2.9 | 4.3 | 5.7 | 7.1 | 8.6 | 10 |
| Использует | 0.036 | 0.000 | 0.045 | 0.090 | 0.108 | 0.126 | 0.108 | 0.054 |
| Не использует | 0.036 | 0.036 | 0.045 | 0.072 | 0.117 | 0.063 | 0.063 | 0.000 |

*Выводы*:

1. Использование С++ повышает использование синтаксического сахара

### Использование сахара и коэффициент незнания

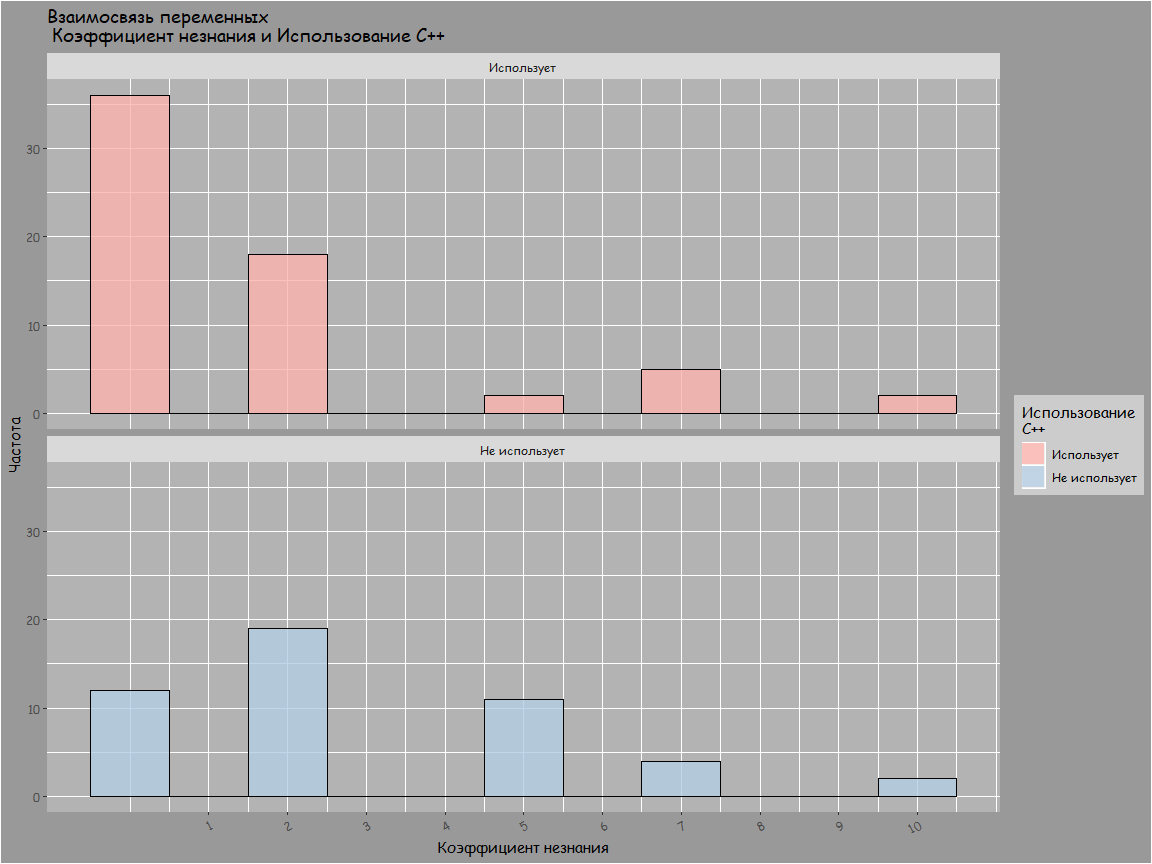


|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 2.5 | 5 | 7.5 | 10 |
| 0 | 0.000 | 0.018 | 0.009 | 0.009 | 0.036 |
| 1.4 | 0.000 | 0.018 | 0.018 | 0.000 | 0.000 |
| 2.9 | 0.009 | 0.036 | 0.027 | 0.018 | 0.000 |
| 4.3 | 0.027 | 0.081 | 0.018 | 0.036 | 0.000 |
| 5.7 | 0.063 | 0.117 | 0.027 | 0.018 | 0.000 |
| 7.1 | 0.153 | 0.027 | 0.009 | 0.000 | 0.000 |
| 8.6 | 0.126 | 0.036 | 0.009 | 0.000 | 0.000 |
| 10 | 0.054 | 0.000 | 0.000 | 0.000 | 0.000 |

*Выводы*:

1. Чем больше прогер юзает ~~снюс~~ синтаксический сахар, тем меньше он не знает
2. Созданный коэффициент хорошо описывает знания программиста.
3. Если вам сказали, что тот прогер “слишком много знал”, скорее всего, он кодил на плюсах.

### Коэффициент незнания и использование С++

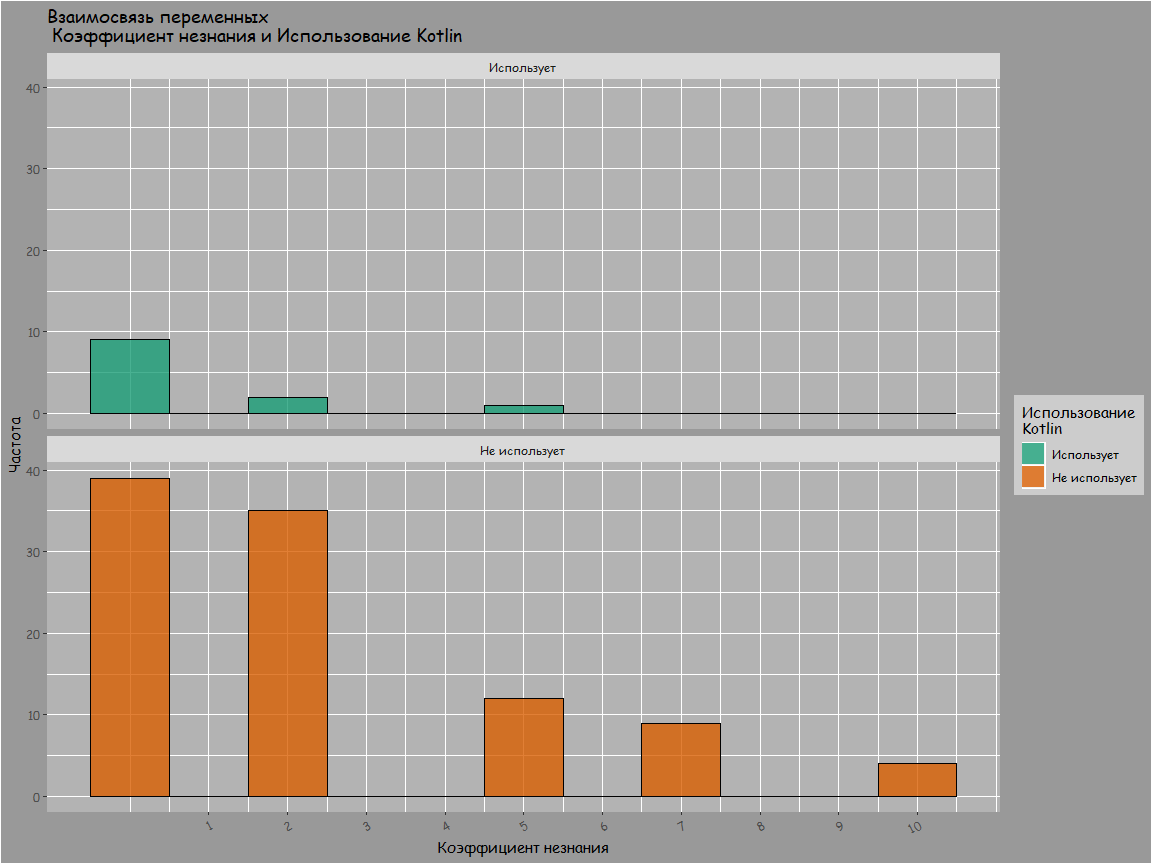


|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 2.5 | 5 | 7.5 | 10 |
| Использует | 0.324 | 0.162 | 0.018 | 0.045 | 0.018 |
| Не использует | 0.108 | 0.171 | 0.099 | 0.036 | 0.018 |

*Выводы*:

1. Знание C++ повышает IT-грамотность программиста. Что вполне разумно, ведь до изучения плюсов надо много чего изведать.

### Коэффициент незнания и использование Kotlin



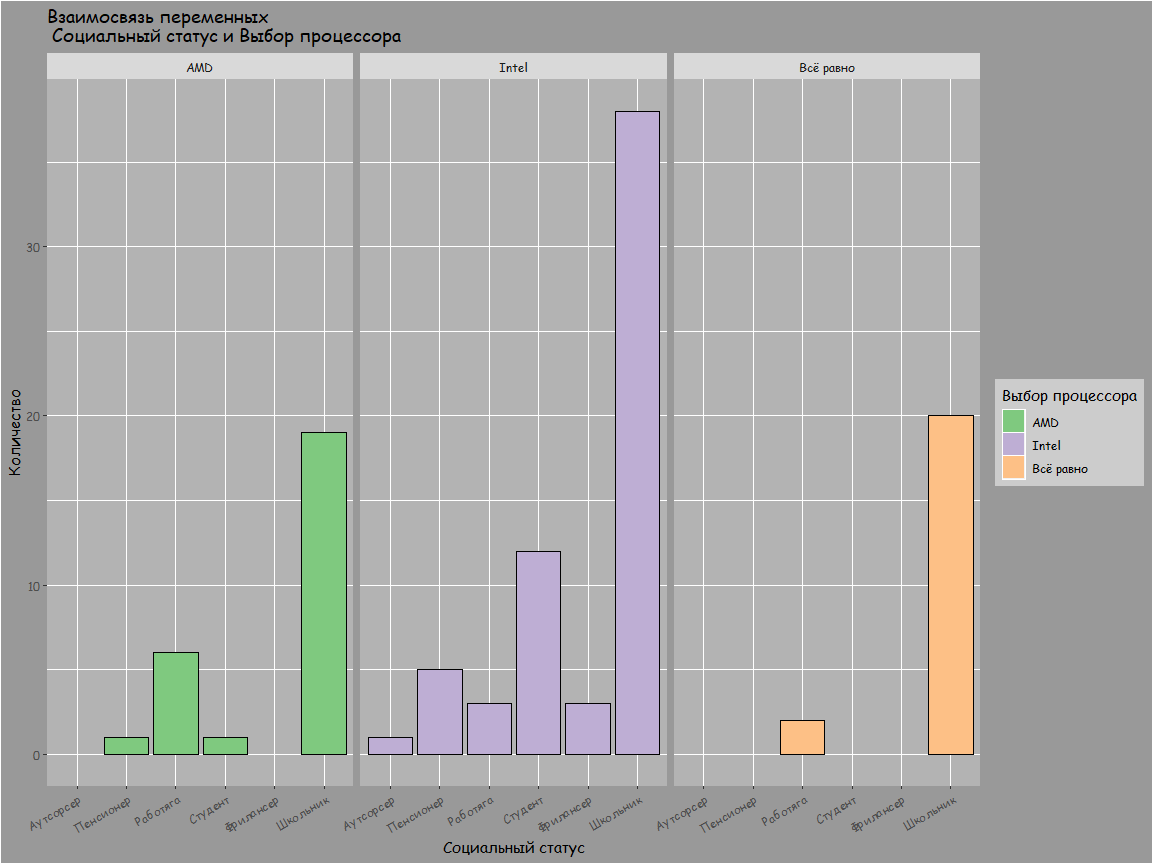
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 2.5 | 5 | 7.5 | 10 |
| Использует | 0.081 | 0.018 | 0.009 | 0.000 | 0.000 |
| Не использует | 0.351 | 0.315 | 0.108 | 0.081 | 0.036 |

*Выводы*:

1. Котлинисты, несмотря на свою немногочисленность, показали высокие результаты айти-грамотости.

## Эволюция программиста в течение жизни

### Социальный статус и выбор процессора

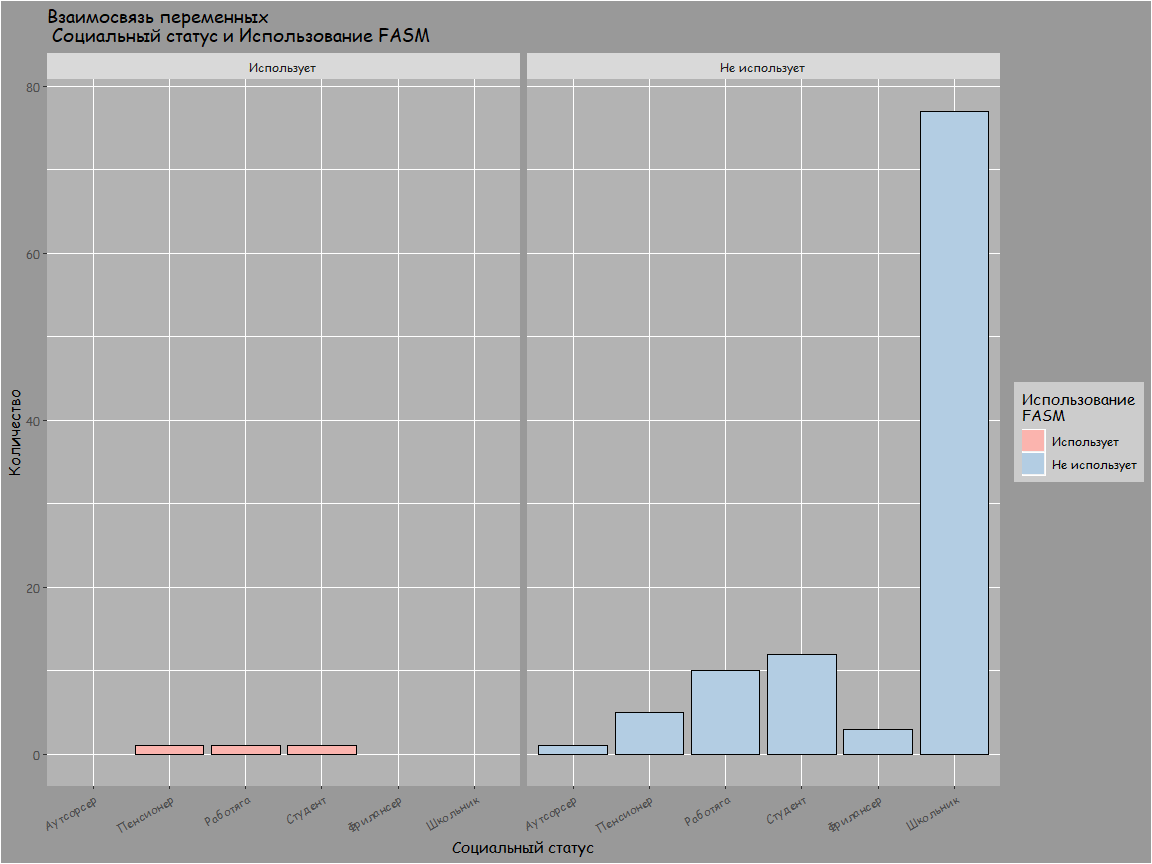


|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Аутсорсер | Пенсионер | Работяга | Студент | Фрилансер | Школьник |
| AMD | 0.000 | 0.009 | 0.054 | 0.009 | 0.000 | 0.171 |
| Intel | 0.009 | 0.045 | 0.027 | 0.108 | 0.027 | 0.342 |
| Всё равно | 0.000 | 0.000 | 0.018 | 0.000 | 0.000 | 0.180 |

*Выводы*:

1. Все социальные группы представлены только для варианта ответа Intel. Это может говорить либо о популярности интела среди мальчишек и девчонок, а также их родителей, а также и их родителей, либо о неразборчивости людей в тех или иных группах.
2. Работяги более склоняются к выбору AMD. Автор считает, что их мнению можно доверять.
3. Всё равно на марку процессора в основном шкiльникам. Видимо, у них ещё впереди штурм мощностей железа.

### Социальный статус и использование FASM

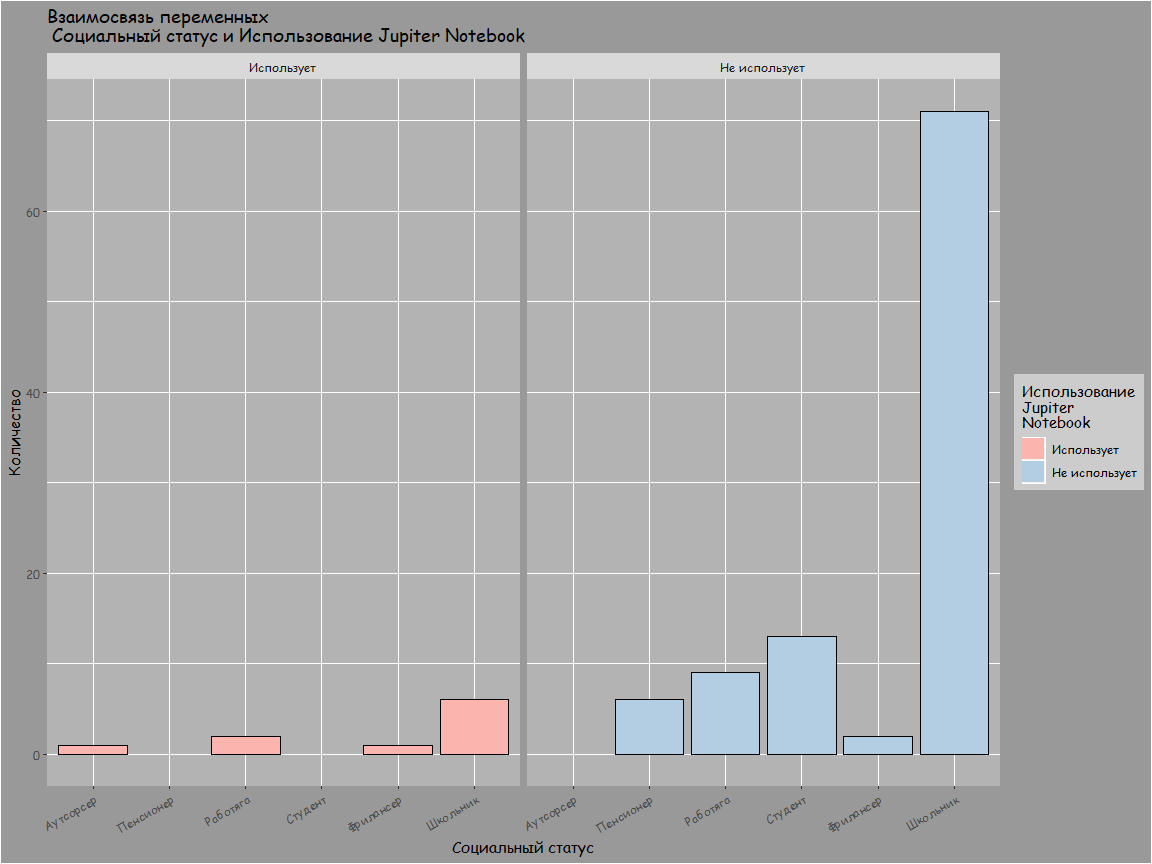


|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Аутсорсер | Пенсионер | Работяга | Студент | Фрилансер | Школьник |
| Использует | 0.000 | 0.009 | 0.009 | 0.009 | 0.000 | 0.000 |
| Не использует | 0.009 | 0.045 | 0.090 | 0.108 | 0.027 | 0.694 |

*Выводы*:

1. В исследовании нашлись три динозавра: пенсионер, работяга и студент. Их объединило знание ассемблера. Автор призывает беречь этот вид от полного исчезновения.

### Социальный статус и использование Jupyter Notebook

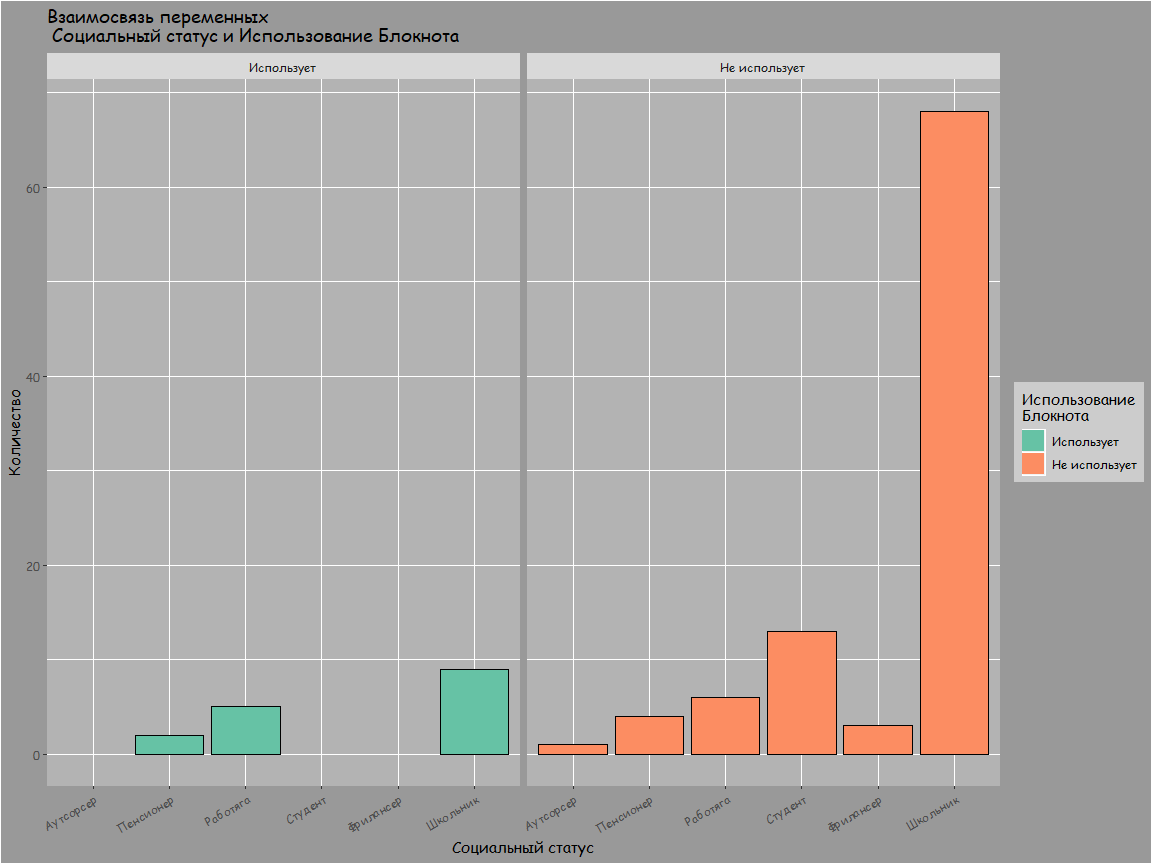


|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Аутсорсер | Пенсионер | Работяга | Студент | Фрилансер | Школьник |
| Использует | 0.009 | 0.000 | 0.018 | 0.000 | 0.009 | 0.054 |
| Не использует | 0.000 | 0.054 | 0.081 | 0.117 | 0.018 | 0.640 |

*Выводы*:

1. Jupyter Notebook использует небольшая часть школьников и работяг. Хотя его интерактивность бывает иногда очень полезной.

### Социальный статус и использование Блокнота

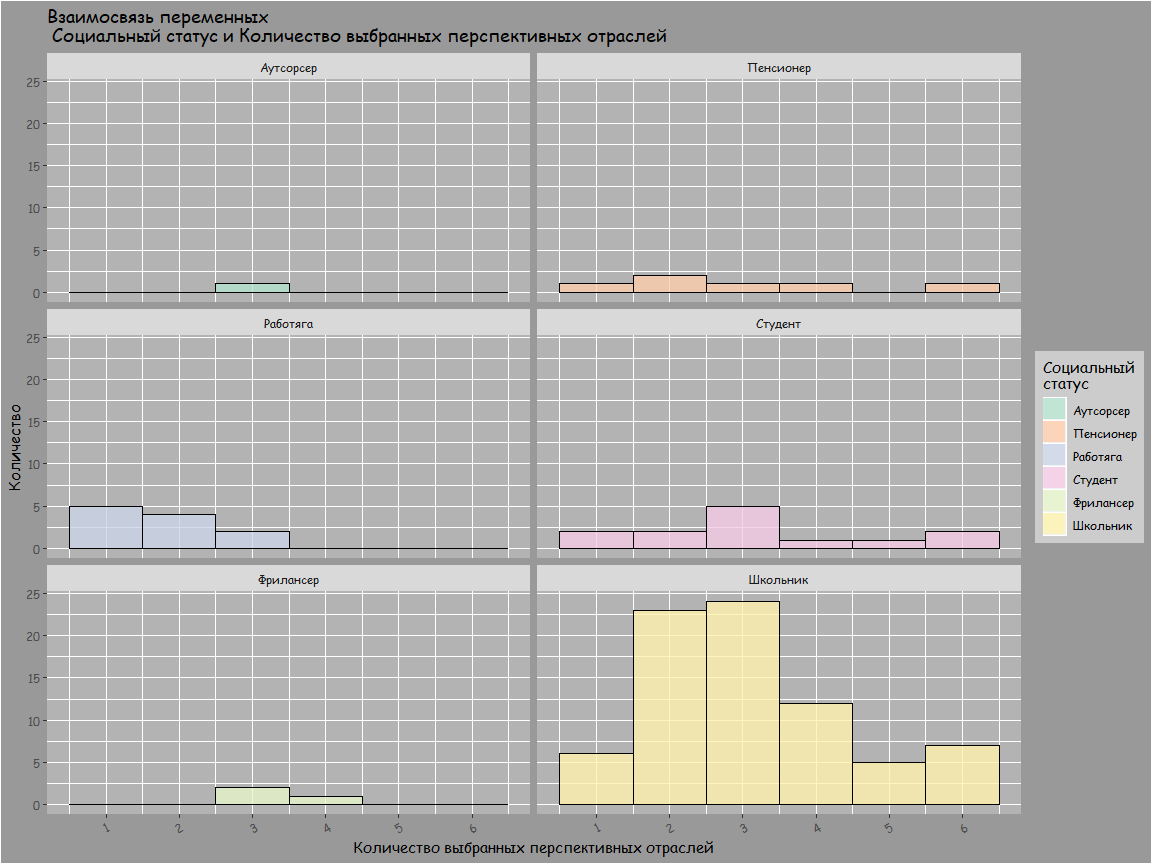


|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Аутсорсер | Пенсионер | Работяга | Студент | Фрилансер | Школьник |
| Использует | 0.000 | 0.018 | 0.045 | 0.000 | 0.000 | 0.081 |
| Не использует | 0.009 | 0.036 | 0.054 | 0.117 | 0.027 | 0.613 |

*Выводы*:

1. Пенсионеры и работяги до сих пор используют Блоктнот в качестве редактора кода. Наверное, по причине давности своей закалки.
2. Школьники же используют его по другим причинам: скорее всего, это зелёность и молодость, либо извращённый вкус к коду.

### Социальный статус и количество перспективных отраслей



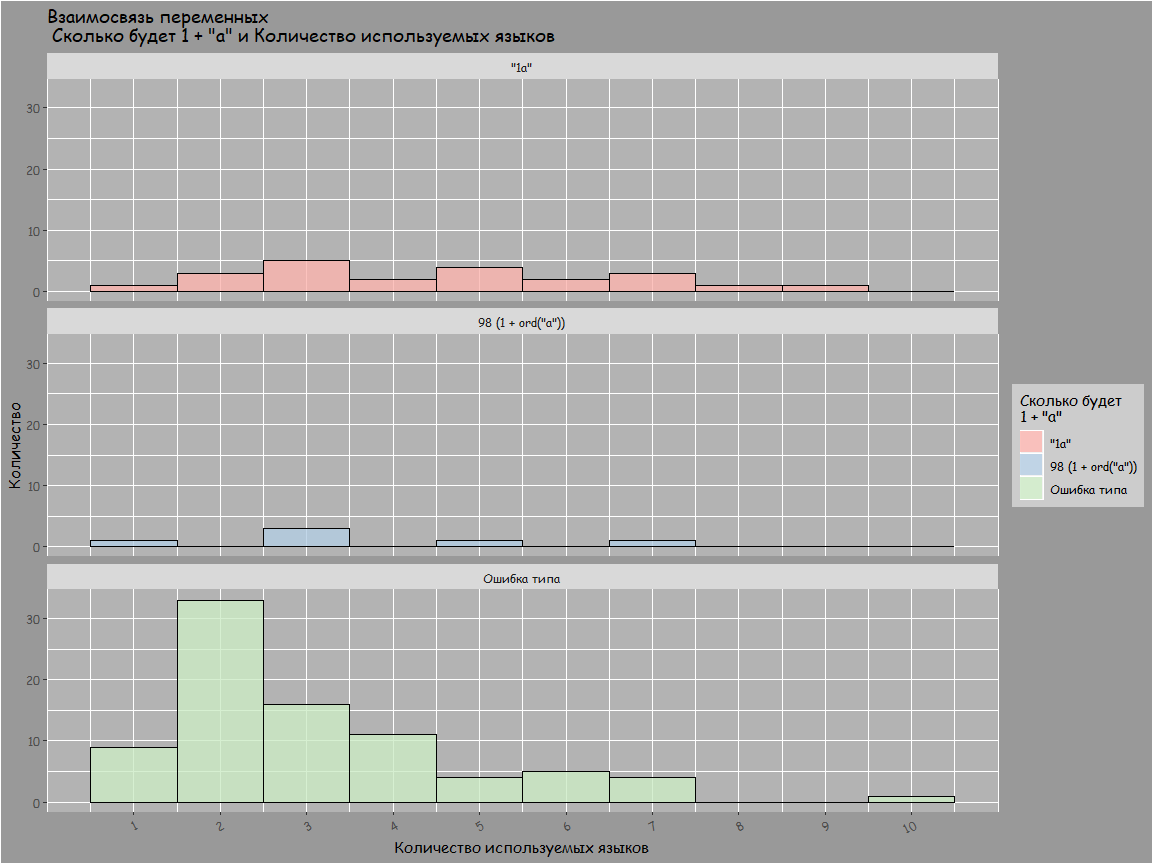
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 |
| Аутсорсер | 0.000 | 0.000 | 0.009 | 0.000 | 0.000 | 0.000 |
| Пенсионер | 0.009 | 0.018 | 0.009 | 0.009 | 0.000 | 0.009 |
| Работяга | 0.045 | 0.036 | 0.018 | 0.000 | 0.000 | 0.000 |
| Студент | 0.018 | 0.018 | 0.045 | 0.009 | 0.009 | 0.018 |
| Фрилансер | 0.000 | 0.000 | 0.018 | 0.009 | 0.000 | 0.000 |
| Школьник | 0.054 | 0.207 | 0.216 | 0.108 | 0.045 | 0.063 |

*Выводы*:

1. Работяги весьма пессимистичны и из шести предложенных не выбирали больше трёх. Им виднее, у них есть опыт за плечами.
2. У пенсионеров и студентов нет единого мнения о количестве перспективных отраслей.

## А зачем было прибавлять к единице “а”?

### 1 + “a” и количество языков

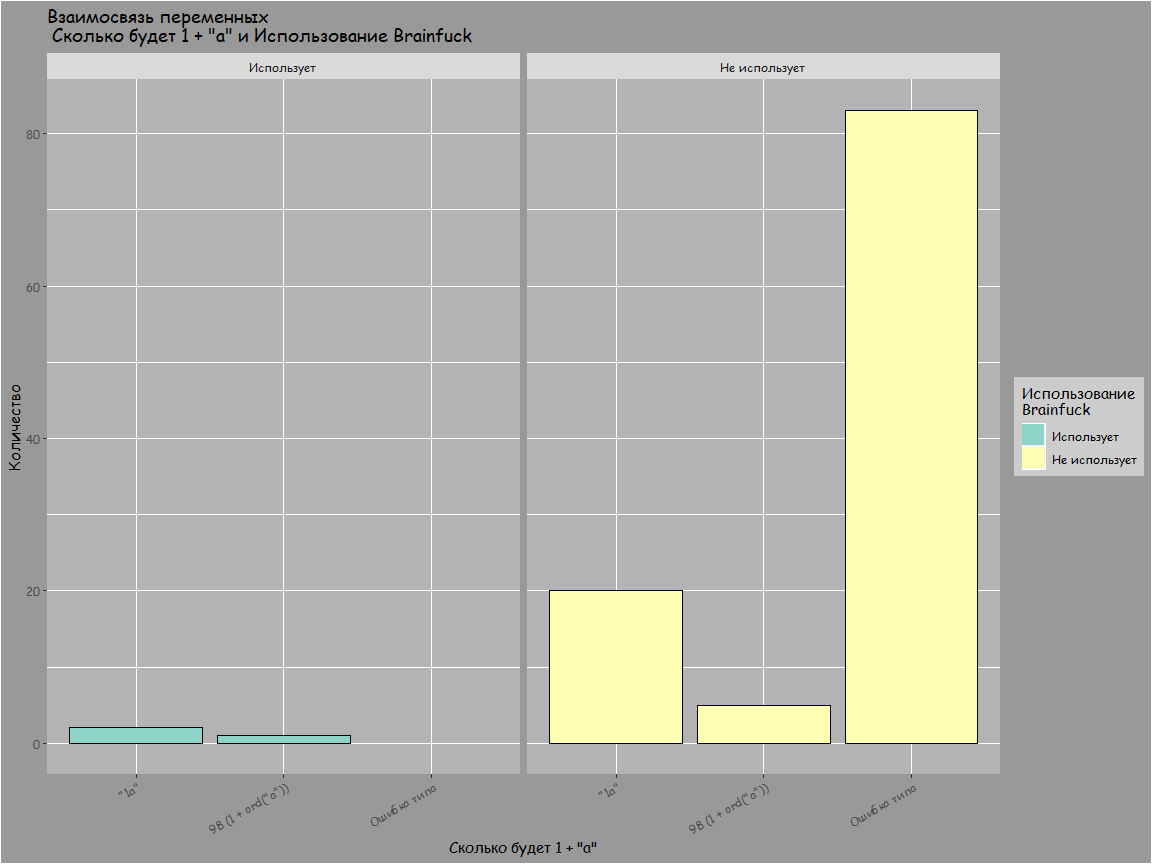


|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| “1а” | 0.009 | 0.027 | 0.045 | 0.018 | 0.036 | 0.018 | 0.027 | 0.009 | 0.009 | 0.000 |
| 98 (1 + ord(“a”)) | 0.009 | 0.000 | 0.027 | 0.000 | 0.009 | 0.000 | 0.009 | 0.000 | 0.000 | 0.000 |
| Ошибка типа | 0.081 | 0.297 | 0.144 | 0.099 | 0.036 | 0.045 | 0.036 | 0.000 | 0.000 | 0.009 |

*Выводы*:

1. Среди ответов “1а” и “98” много опытных программистов, перенюхавших все виды типизации.
2. Большинство всё равно признало такую операцию ошибкой типа.

### 1 + “a” и использование Brainfuck

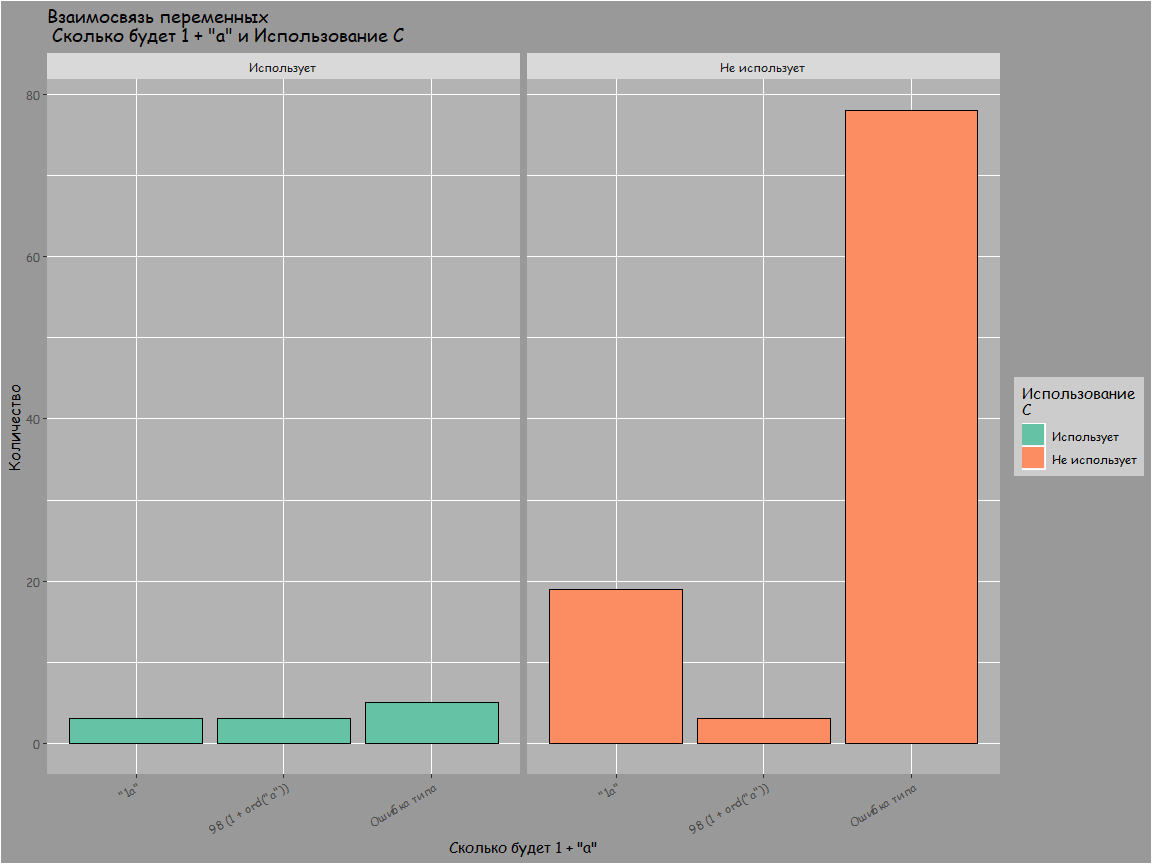


|  |  |  |  |
| --- | --- | --- | --- |
|  | “1а” | 98 (1 + ord(“a”)) | Ошибка типа |
| Использует | 0.018 | 0.009 | 0.000 |
| Не использует | 0.180 | 0.045 | 0.748 |

*Выводы*:

1. Не уверен, но возможно, brainfuck калечит людей, что следует из названия.
2. А, возможно, они просто начинают мыслить как настоящие нигилисты, не признающие прежних устоев.

### 1 + “a” и Использование C

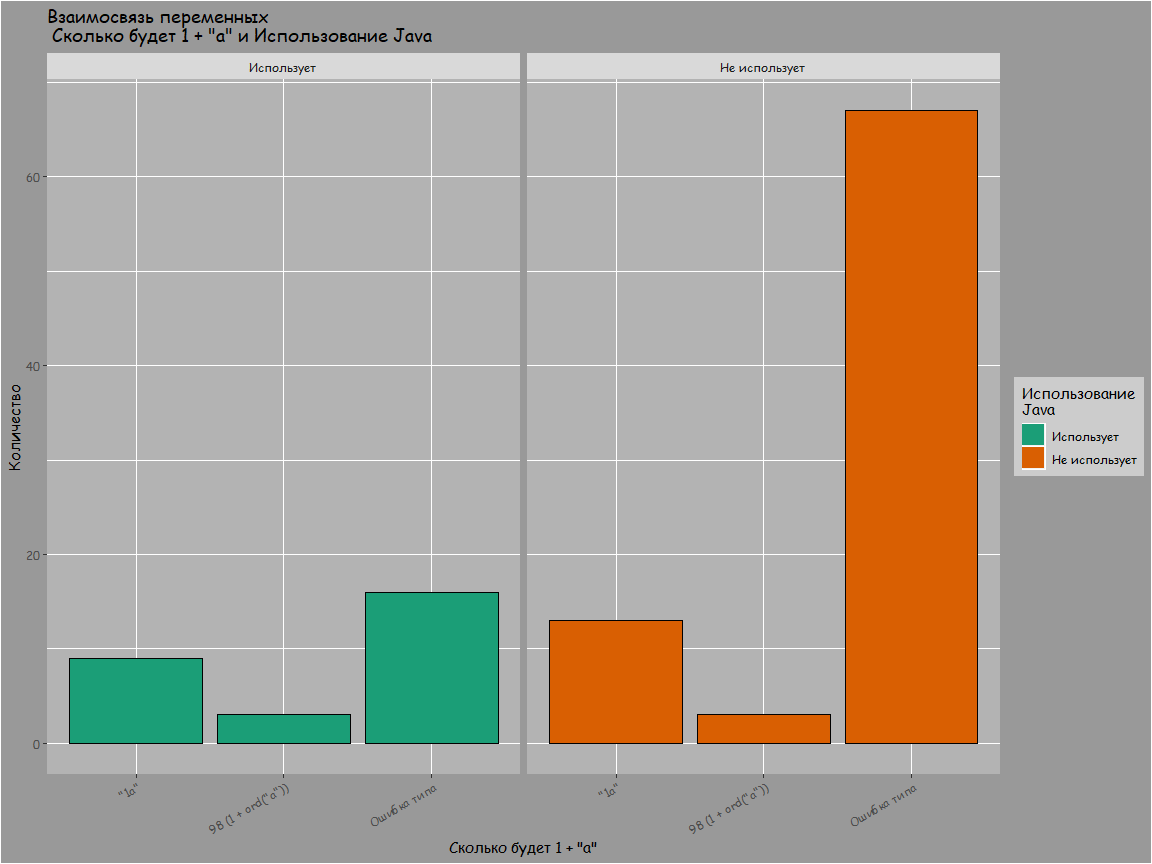


|  |  |  |  |
| --- | --- | --- | --- |
|  | “1а” | 98 (1 + ord(“a”)) | Ошибка типа |
| Использует | 0.027 | 0.027 | 0.045 |
| Не использует | 0.171 | 0.027 | 0.703 |

*Выводы*:

1. Изучение неглубоких недр программирования дарует людям другую точку зрения.

### 1 + “a” и использование Java

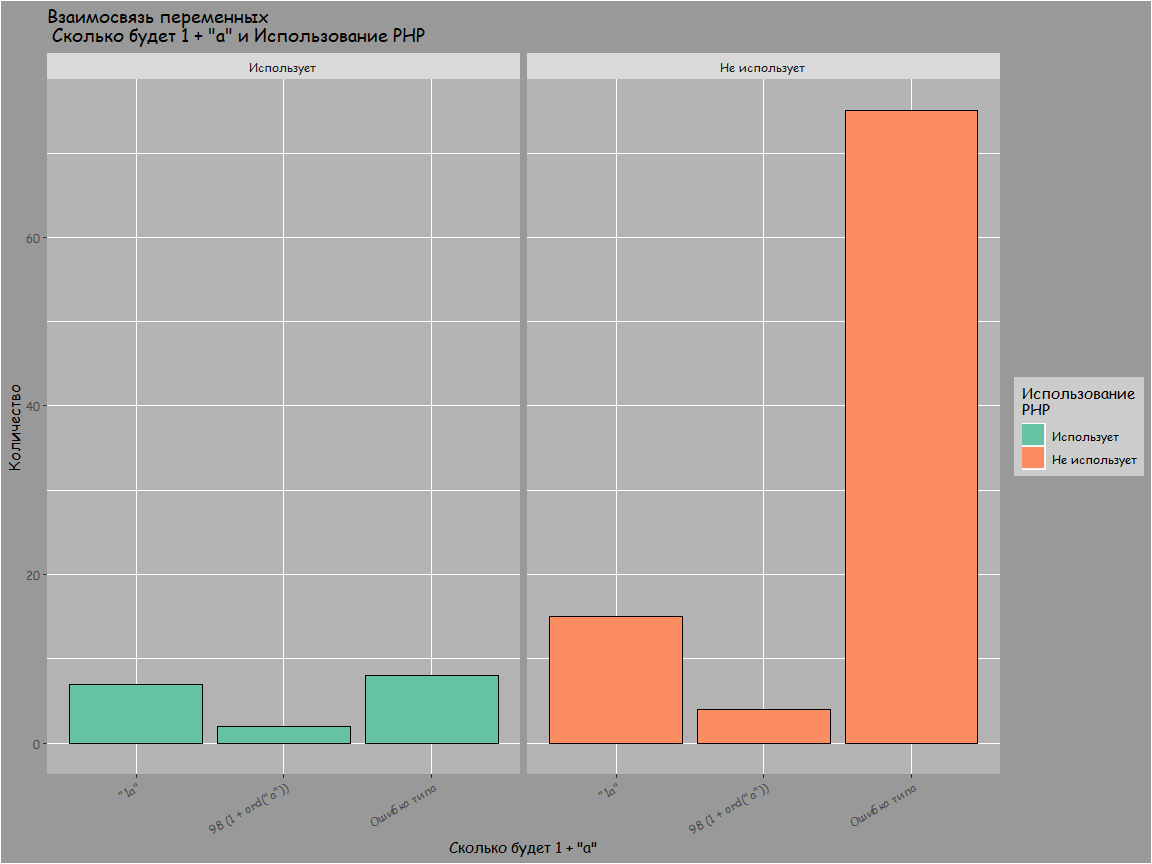


|  |  |  |  |
| --- | --- | --- | --- |
|  | “1а” | 98 (1 + ord(“a”)) | Ошибка типа |
| Использует | 0.081 | 0.027 | 0.144 |
| Не использует | 0.117 | 0.027 | 0.604 |

*Выводы*:

1. Изучение джавы действует на мозги прогеров в плане типизации в большей степени, чем это делает Цзиньпин (Cи).

### 1 + “a” и использование PHP



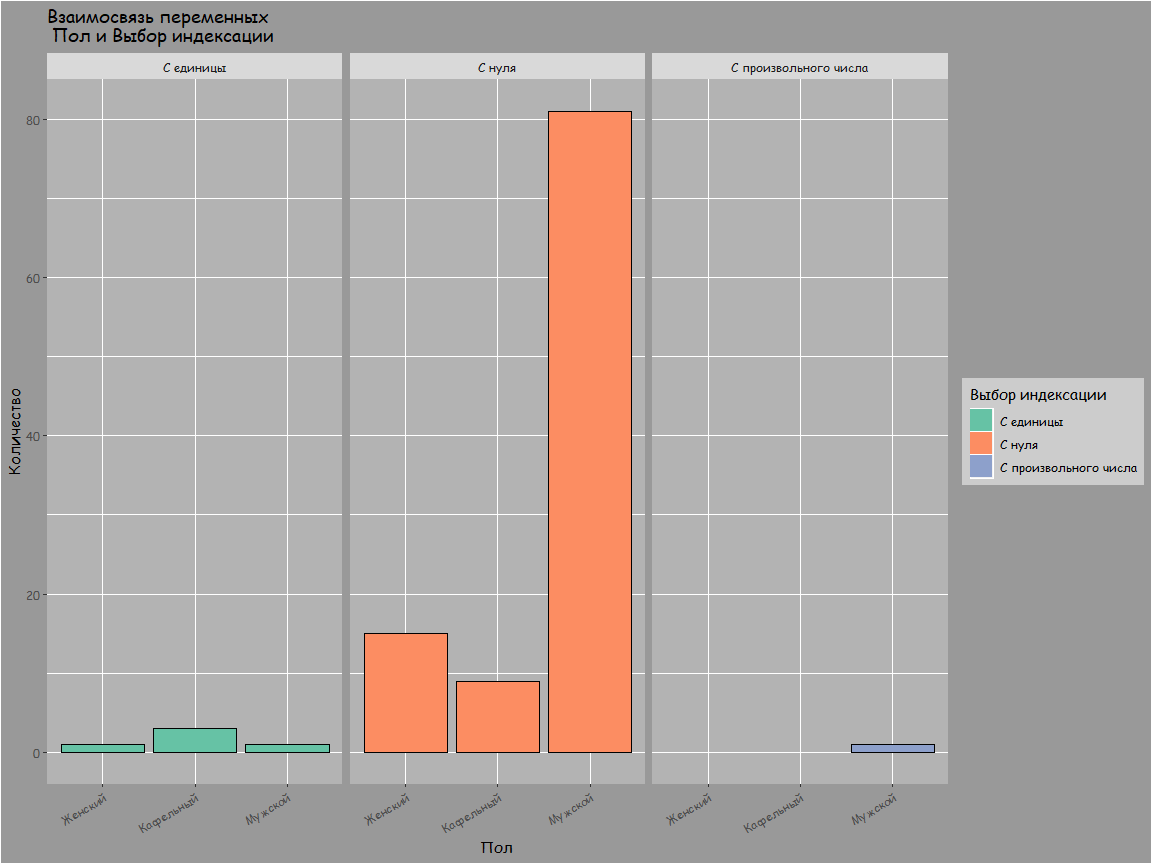
|  |  |  |  |
| --- | --- | --- | --- |
|  | “1а” | 98 (1 + ord(“a”)) | Ошибка типа |
| Использует | 0.063 | 0.018 | 0.072 |
| Не использует | 0.135 | 0.036 | 0.676 |

*Выводы*:

1. Использование Пыхи также заставляет людей задуматься над действительностью ошибки типа.

## Как выбрать пол, если ребёнок - программист?

### Пол и выбор индексации

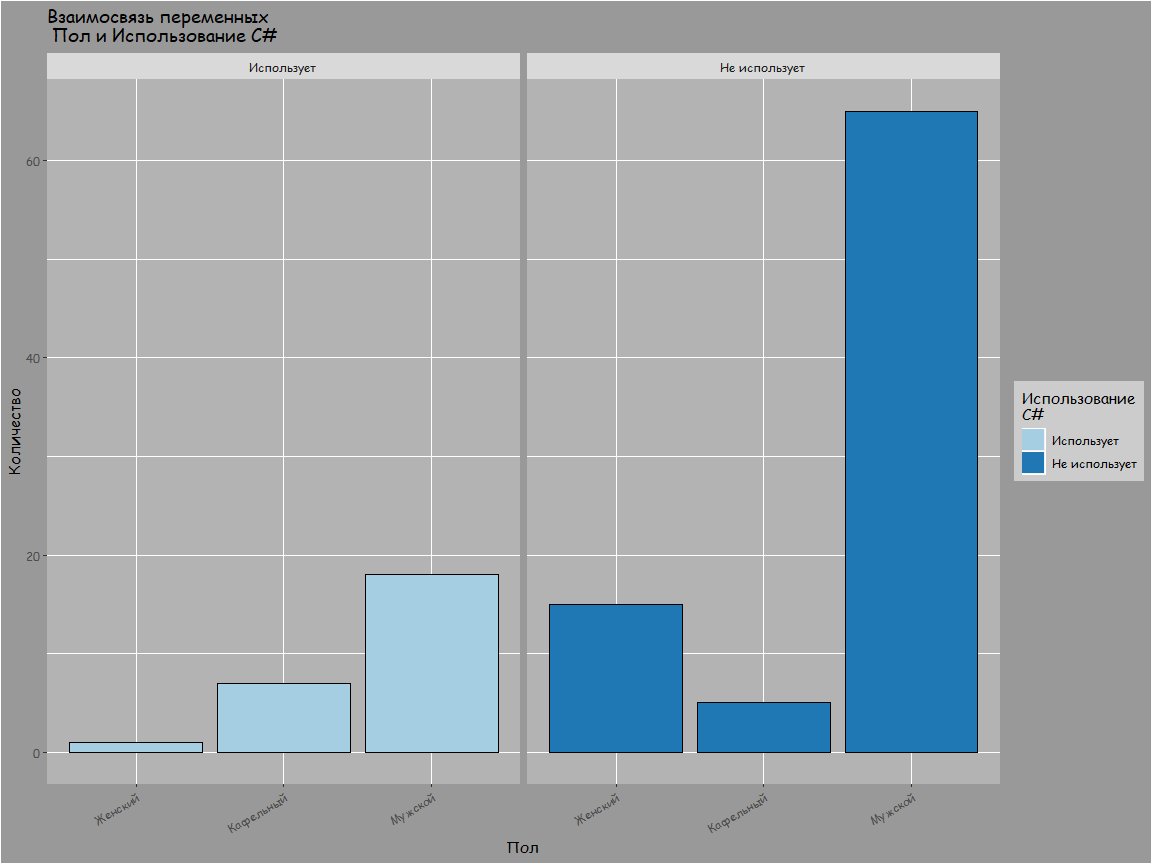


|  |  |  |  |
| --- | --- | --- | --- |
|  | С единицы | С нуля | С произвольного числа |
| Женский | 0.009 | 0.135 | 0.000 |
| Кафельный | 0.027 | 0.081 | 0.000 |
| Мужской | 0.009 | 0.730 | 0.009 |

*Выводы*:

1. О необходимости индексировать с единицы больше всего задумываются граждане кафельного пола.
2. Только настоящий мужик (а он в этой категории один), мог так смело высказаться за нумерацию с произвольного числа.

### Пол и использование C#

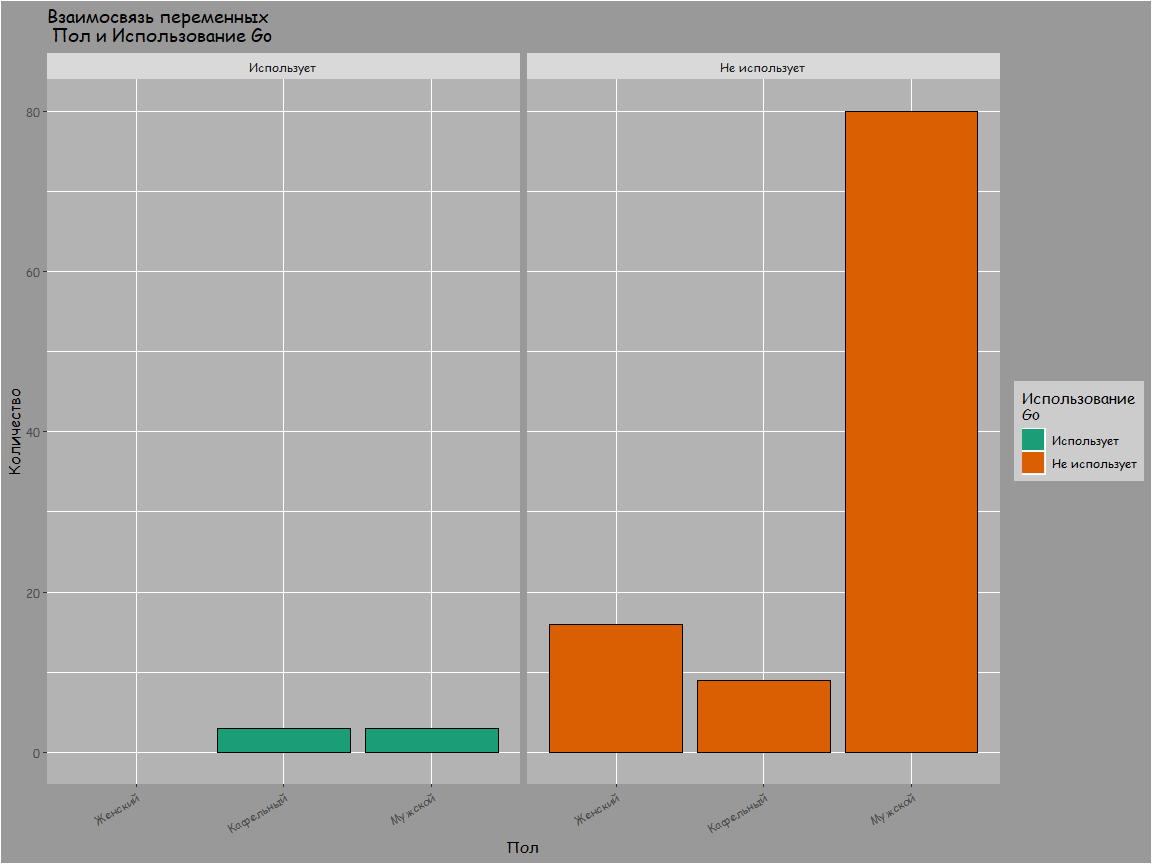


|  |  |  |  |
| --- | --- | --- | --- |
|  | Женский | Кафельный | Мужской |
| Использует | 0.009 | 0.063 | 0.162 |
| Не использует | 0.135 | 0.045 | 0.586 |

*Выводы*:

1. Самый большой процент использующих шарпы обнаружен среди кафельников
2. Программистки знают си-шарп меньше всего

### Пол и использование Go

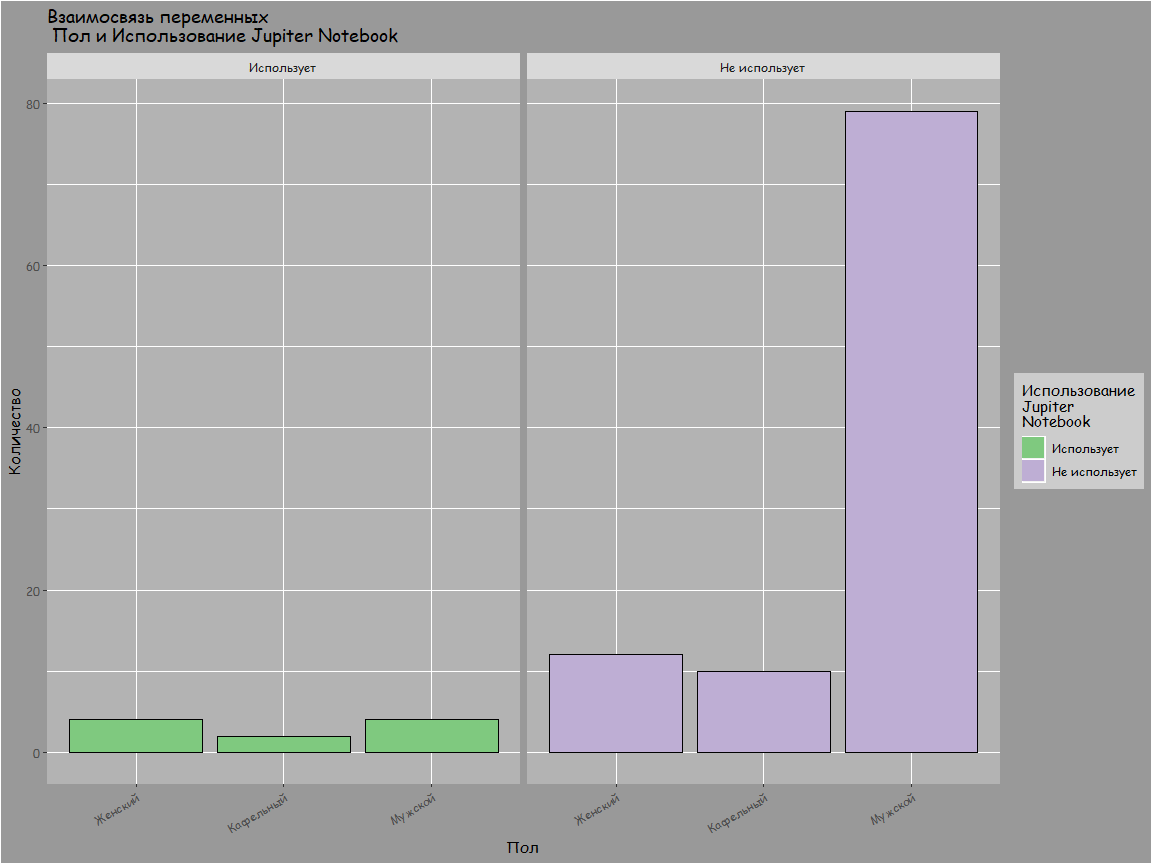


|  |  |  |  |
| --- | --- | --- | --- |
|  | Женский | Кафельный | Мужской |
| Использует | 0.000 | 0.027 | 0.027 |
| Не использует | 0.144 | 0.081 | 0.721 |

*Выводы*:

1. Знающих Go оказалось не так уж и много, но стоит отметить, что среди них нет женщин.

### Пол и использование Jupyter Notebook

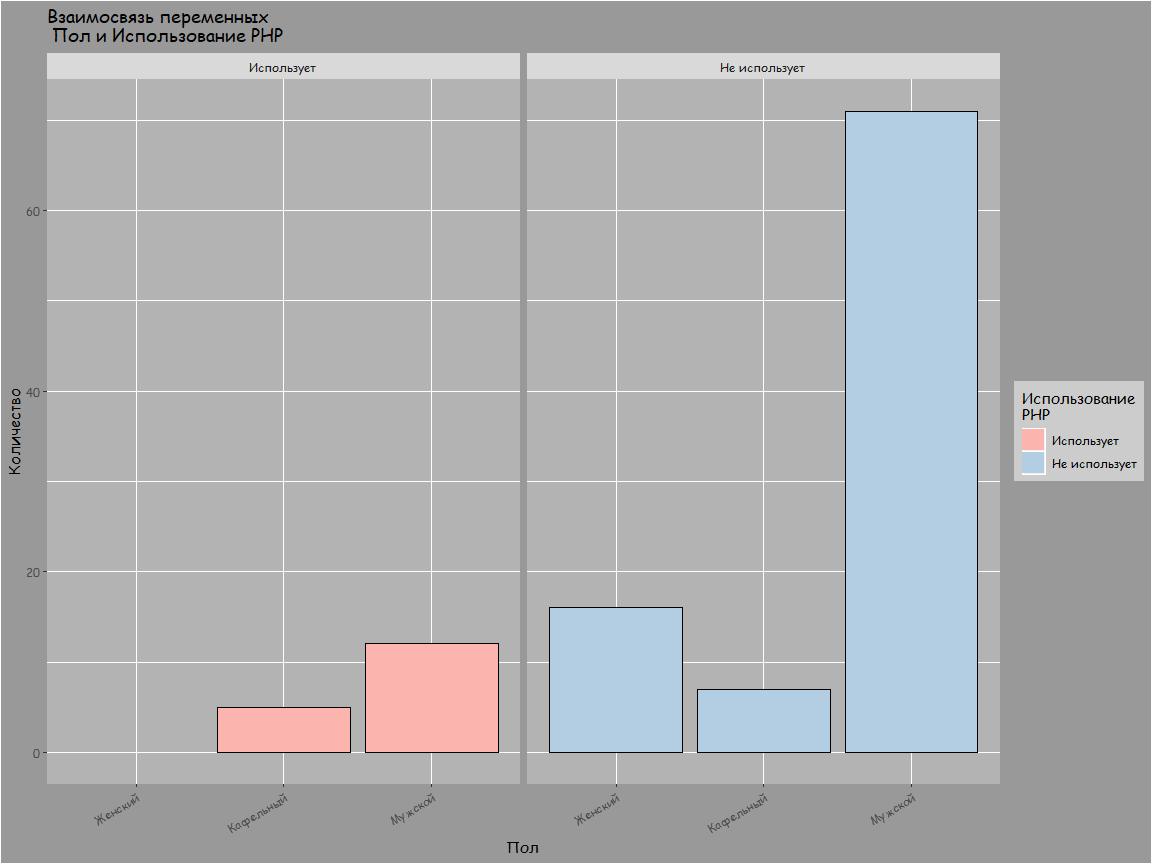


|  |  |  |  |
| --- | --- | --- | --- |
|  | Женский | Кафельный | Мужской |
| Использует | 0.036 | 0.018 | 0.036 |
| Не использует | 0.108 | 0.090 | 0.712 |

*Выводы*:

1. А вот тут дамы молодцы. У них самый большой процент использования юпитера, возможно, из-за его наглядности.

### Пол и использование PHP

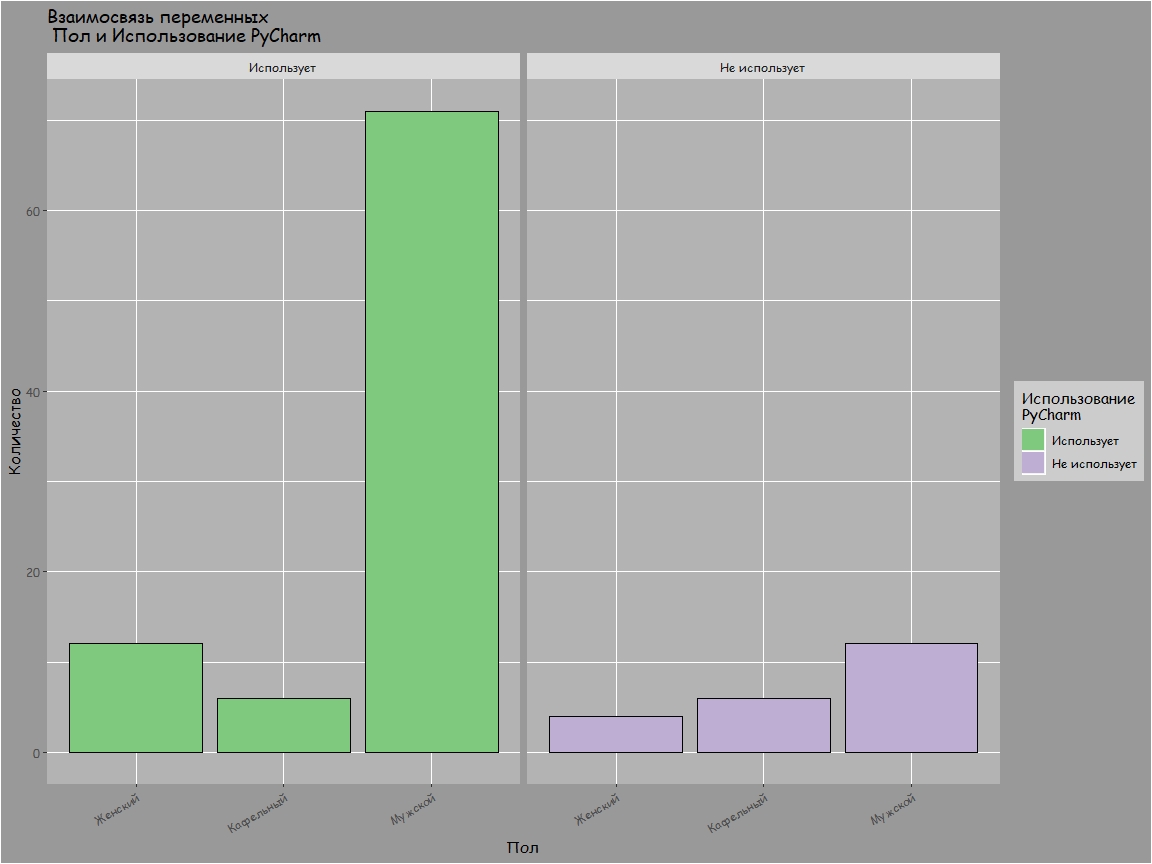


|  |  |  |  |
| --- | --- | --- | --- |
|  | Женский | Кафельный | Мужской |
| Использует | 0.000 | 0.045 | 0.108 |
| Не использует | 0.144 | 0.063 | 0.640 |

*Выводы*:

1. Ни одна женщина-программист из опроса не использует пэ-хэ-пэ, что довольно практично (а кому это вообще надо?)

### Пол и использование PyCharm

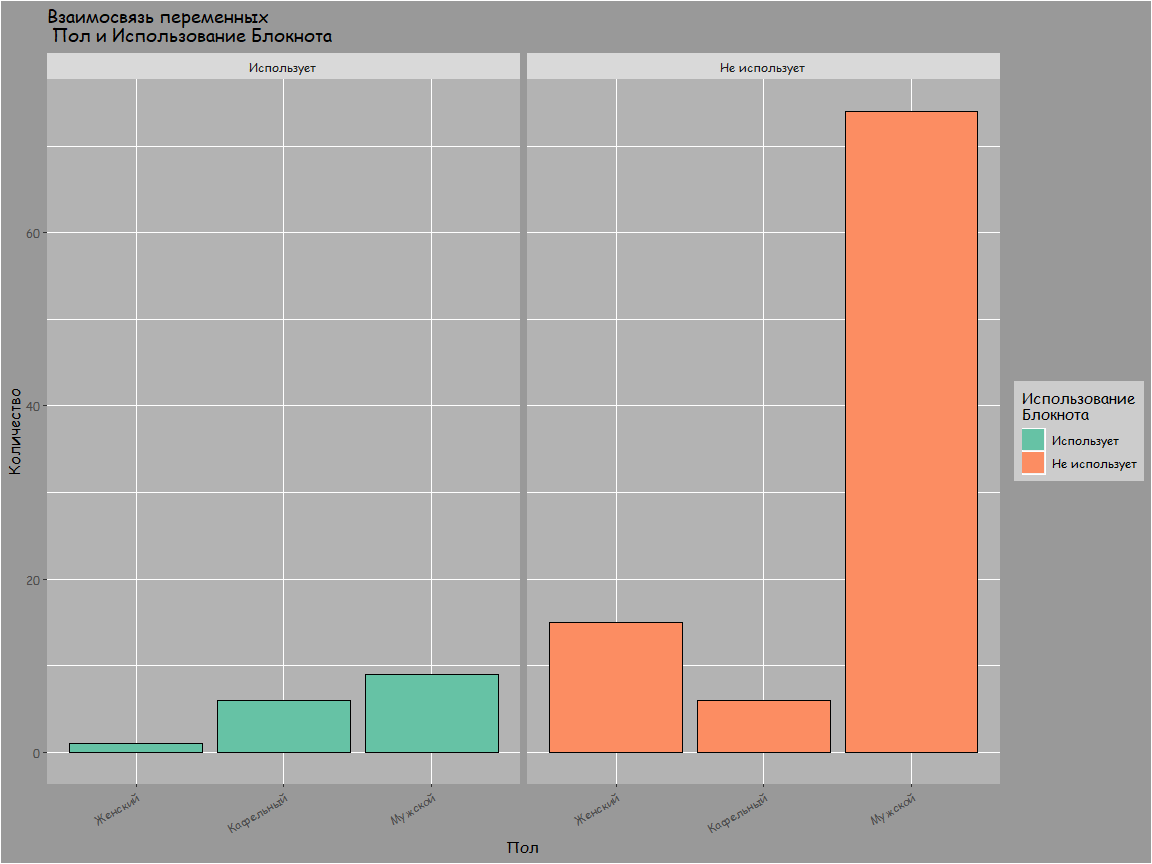


|  |  |  |  |
| --- | --- | --- | --- |
|  | Женский | Кафельный | Мужской |
| Использует | 0.108 | 0.054 | 0.640 |
| Не использует | 0.036 | 0.054 | 0.108 |

*Выводы*:

1. Самый большой процент использования пайчарма у лиц мужского пола, на втором месте женщины.
2. А вот кафельнопольщики используют джетбрейновский питонэдитор пятьдесят на пятьдесят. Скорее всего, в силу своей кафельности.

### Пол и использование Блокнота

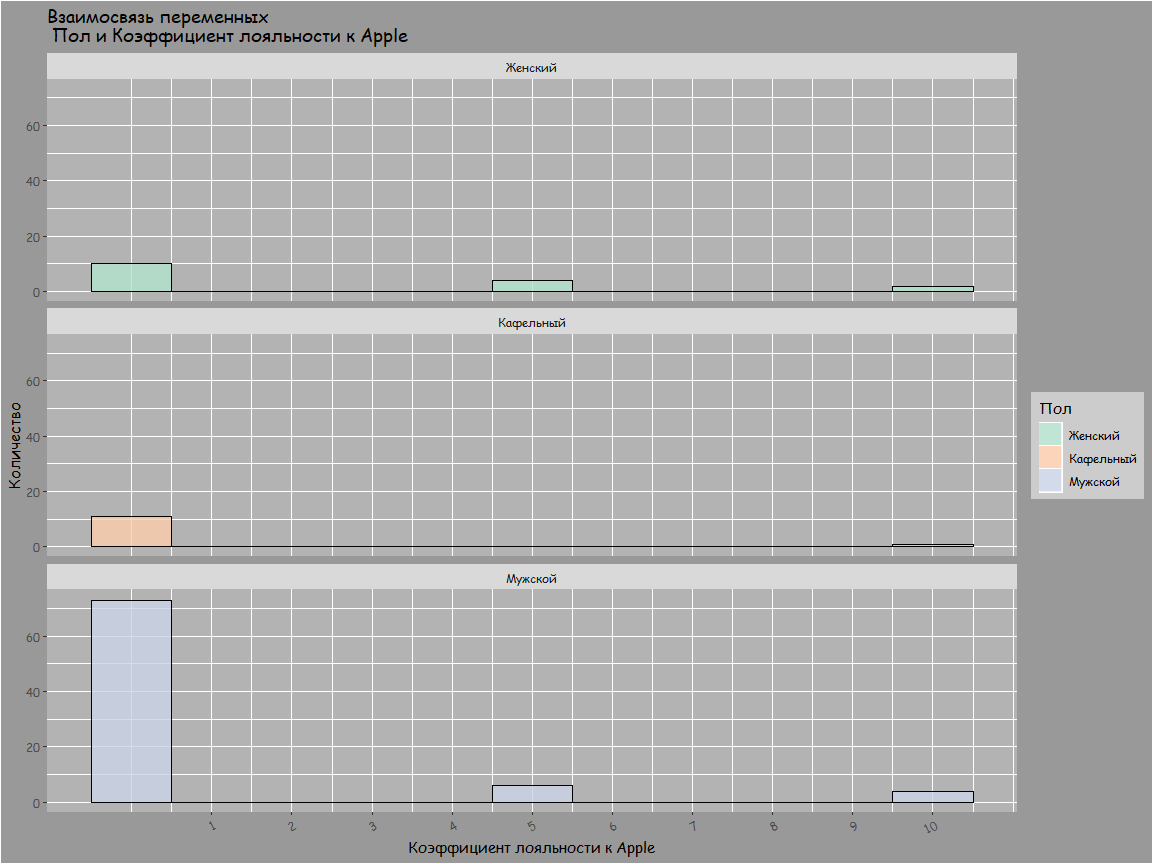


|  |  |  |  |
| --- | --- | --- | --- |
|  | Женский | Кафельный | Мужской |
| Использует | 0.009 | 0.054 | 0.081 |
| Не использует | 0.135 | 0.054 | 0.667 |

*Выводы*:

1. Активнее всего блокнот используют кафельнополые прогеры. Это подтверждает их гендерную самостоятельность в классификации.
2. Женщины и девушки не страдают подобной ерундой.

### Пол и коэффициент лояльности к Apple

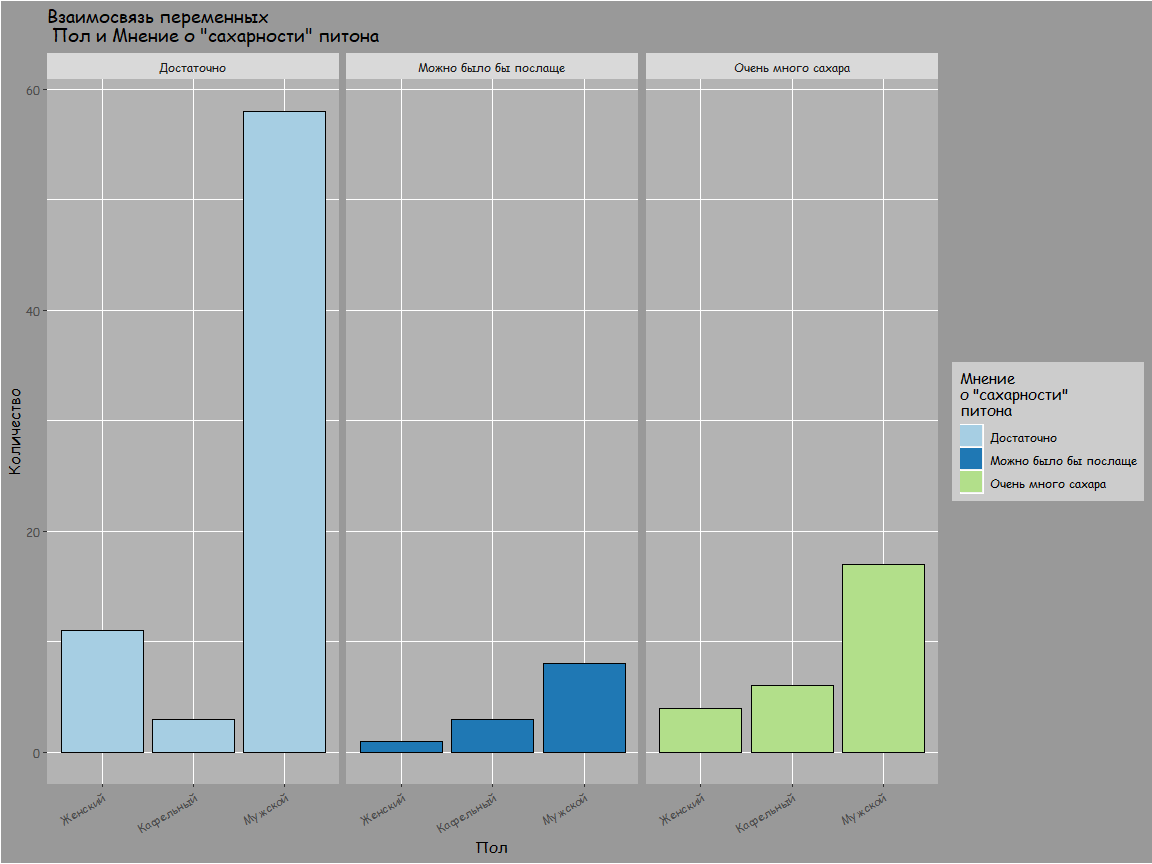


|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 5 | 10 |
| Женский | 0.090 | 0.036 | 0.018 |
| Кафельный | 0.099 | 0.000 | 0.009 |
| Мужской | 0.658 | 0.054 | 0.036 |

*Выводы*:

1. Самыми лояльными к продукции из силиконового забугорья оказались дорогие дамы, впрочем, почти такие же дорогие, как и цена этой самой продукции. Откуда у них деньги на это удовольствие? Скорее всего, сами заработали (особенно студентки и школьницы).
2. Только один человек из отряда кафельнополых поддался на соблазн заюзать эту чёртову яблочную электронику.

### Пол и мнение о “сахарности” питона

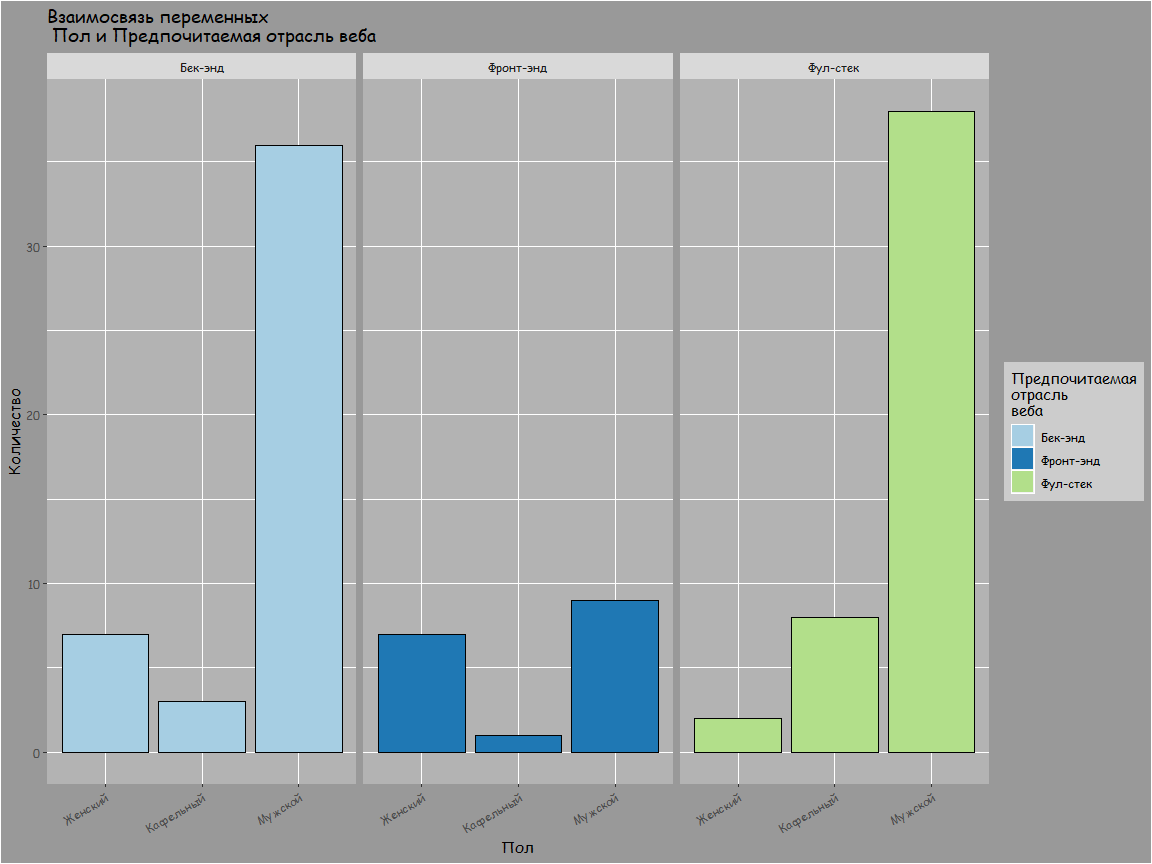


|  |  |  |  |
| --- | --- | --- | --- |
|  | Достаточно | Можно было бы послаще | Очень много сахара |
| Женский | 0.099 | 0.009 | 0.036 |
| Кафельный | 0.027 | 0.027 | 0.054 |
| Мужской | 0.523 | 0.072 | 0.153 |

*Выводы*:

1. Лица мужского пола живут в достатке сахаром питона, и в массе своей не жалуются. То же самое можно сказать о лицах женского пола.
2. Мнения кафельников разошлись. Их мало, но они есть. И четверть из них недовольна пресностью питона.

### Пол и предпочитаемая отрасль веба



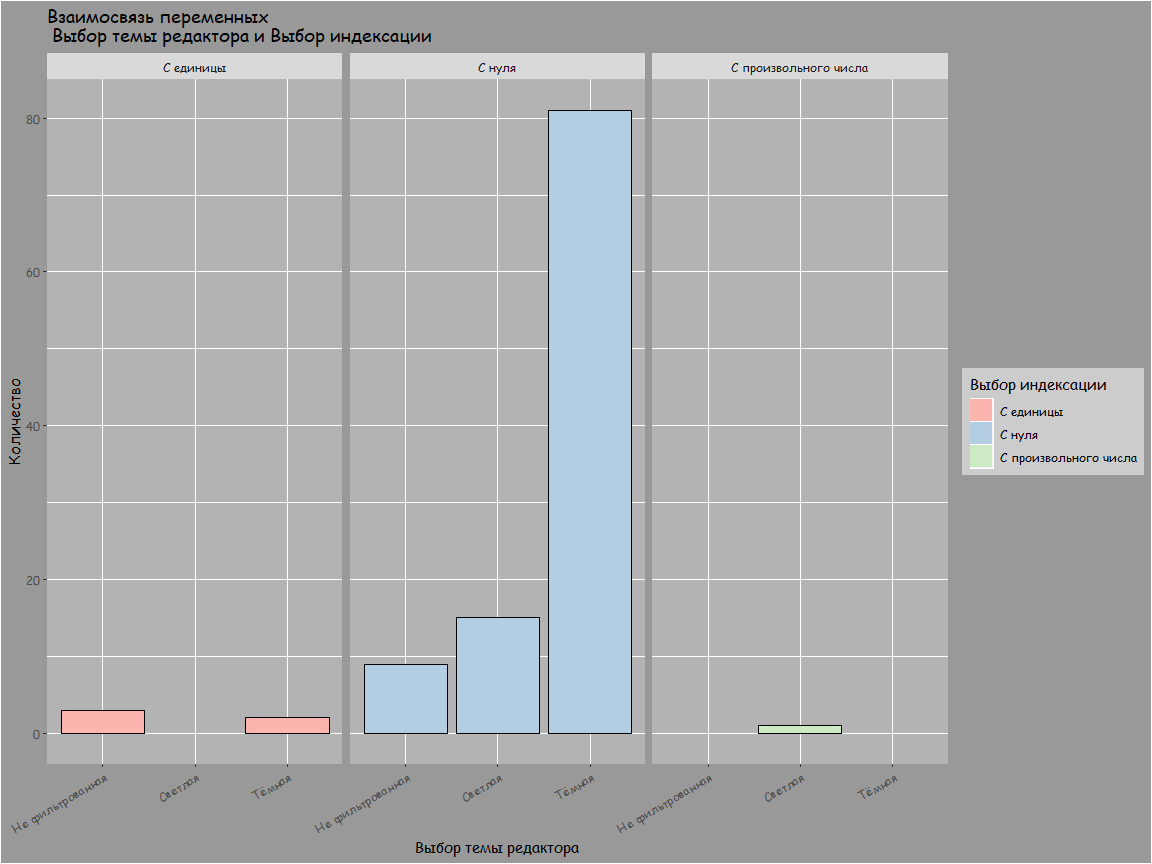
|  |  |  |  |
| --- | --- | --- | --- |
|  | Бек-энд | Фронт-энд | Фул-стек |
| Женский | 0.063 | 0.063 | 0.018 |
| Кафельный | 0.027 | 0.009 | 0.072 |
| Мужской | 0.324 | 0.081 | 0.342 |

*Выводы*:

1. Женщины и девушки предпочитают фасадные работы внутренней отделке сильнее, чем мужчины, юноши и прочие.
2. Представители кафельного пола обожают сладко зафулстечить какой-нибудь проект.

## Любителям нефильтрованного посвящяется

### Тема редактора и выбор индексации

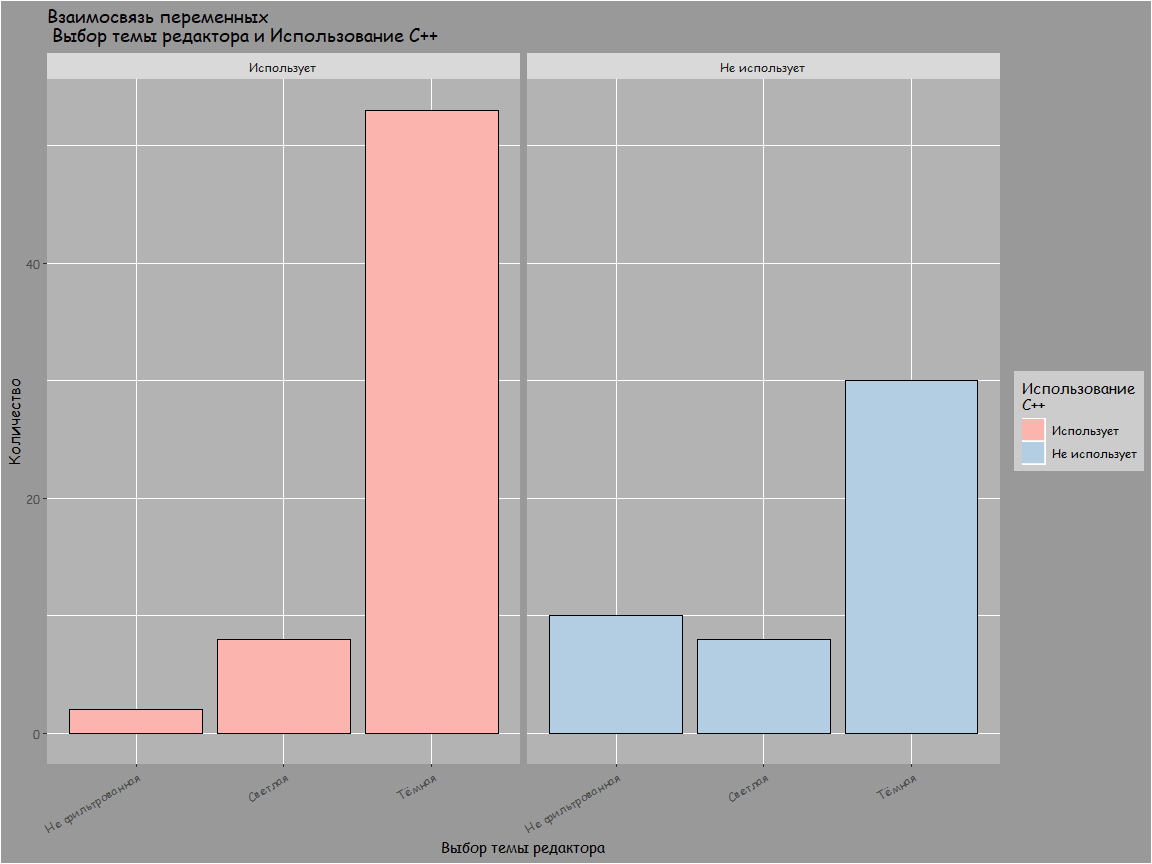


|  |  |  |  |
| --- | --- | --- | --- |
|  | С единицы | С нуля | С произвольного числа |
| Не фильтрованная | 0.027 | 0.081 | 0.000 |
| Светлая | 0.000 | 0.135 | 0.009 |
| Тёмная | 0.018 | 0.730 | 0.000 |

*Выводы*:

1. Действительно, только светлая голова со светлой темой могла предложить относительную индексацию. (Автор делает поклон этому статистическому выбросу)
2. Достаточно большой процент людей с нефильтрованной темой редактора предпочитает лицезреть ~~балтику 9~~ индексацию с единицы. Возможно, сказалось влияние нефильтрованного.

### Тема редактора и использование С++



|  |  |  |  |
| --- | --- | --- | --- |
|  | Не фильтрованная | Светлая | Тёмная |
| Использует | 0.018 | 0.072 | 0.477 |
| Не использует | 0.090 | 0.072 | 0.270 |

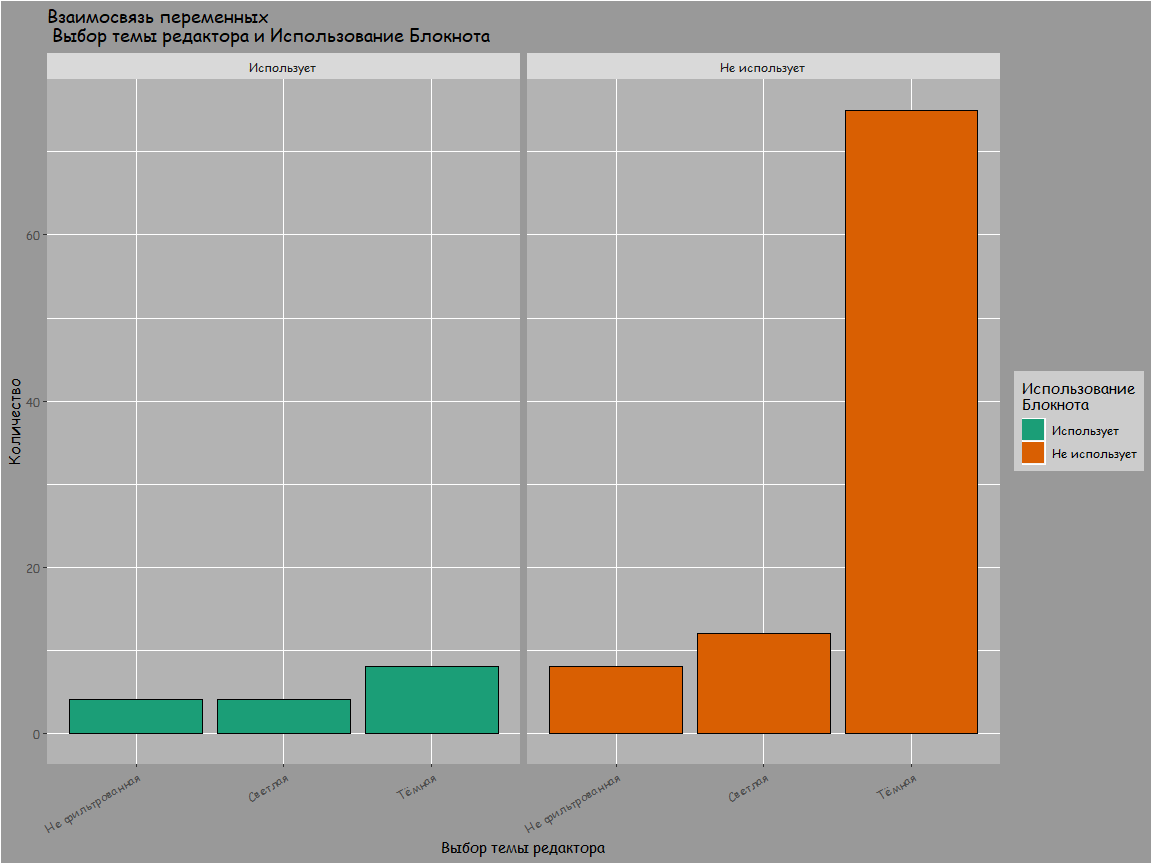
*Выводы*:

1. Использование плюсов “затемняет” редакторы программистов.
2. Также использование плюсов в разы увеличивает трезвость айти-пролетариата.

Да когда же это кончится?!

А вот сейчас и кончится. Внимание, последний график!

### Тема редактора и использование Блокнота



|  |  |  |  |
| --- | --- | --- | --- |
|  | Не фильтрованная | Светлая | Тёмная |
| Использует | 0.036 | 0.036 | 0.072 |
| Не использует | 0.072 | 0.108 | 0.676 |

*Выводы*:

1. Пользователи редактора Блокнот намного менее дружелюбны к тёмной теме.
2. Да и процент “нефильтрованный” тут заметно выше.

# Заключение

Программисты, как оказалось, народ непростой. В их обществе действуют свои невидимиые постороннему, а порой и программистскому, глазу закономерности и правила поведения. На их исследование можно было бы вбухать кучу лет, но так как автор этой статьи конченный меценат и раскидивает своё время налево и направо, что ему порядком надоело, то пусть этим займётся кто-нибудь другой.

## P. S.

Если во время прочтения вам показалось, что автор - пожилой тролль, то вы ошибаетесь.

Тролль он достаточно молодой.

Ваш коллега по цеху программного сотворения: [Павел Соломатин](https://vk.com/crave_ozer_man) - главный и единственный сотрудник компании Penza Street Analitics