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 обладана 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.

# 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.

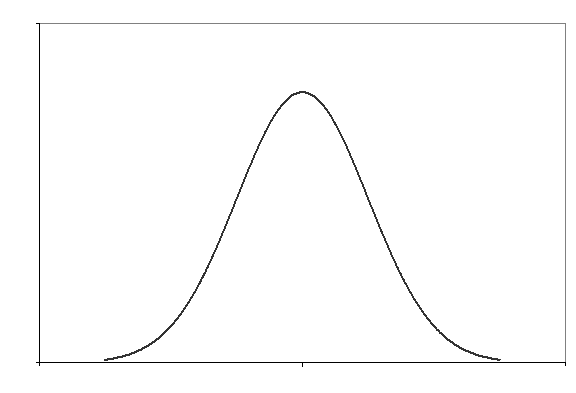
Математика - это всегда не просто, поэтому начнём издалека.

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.

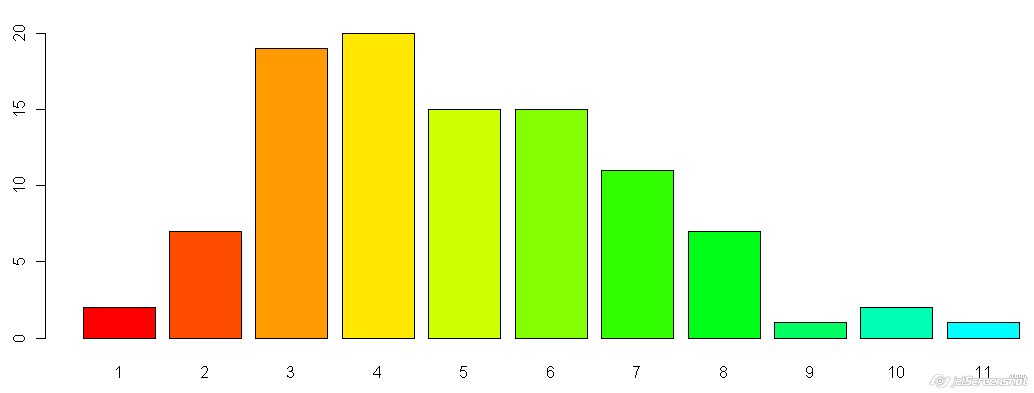
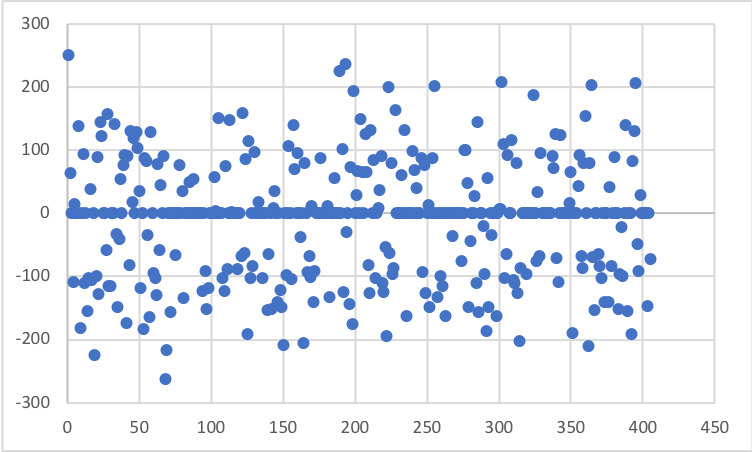
За первый пункт отвечает [Коэфиициент корреляции Пирсона](https://en.wikipedia.org/wiki/Pearson_correlation_coefficient), за второй - [Дисперсионный анализ](https://en.wikipedia.org/wiki/Analysis_of_variance), а за третий - [Точный тест Фишера](https://en.wikipedia.org/wiki/Fisher%27s_exact_test)

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 :)

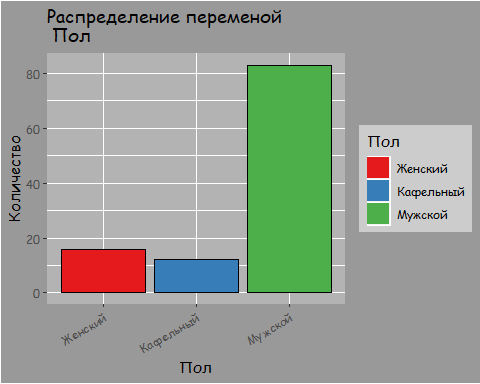
# Распределения ответов

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

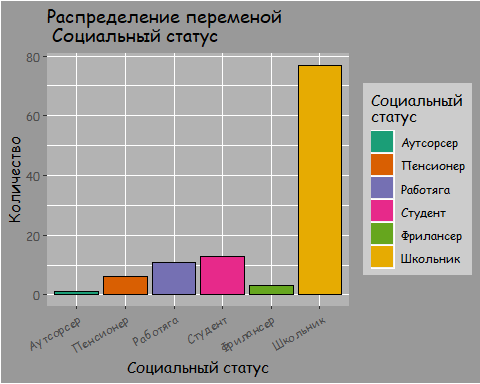
### Пол



|  |  |
| --- | --- |
| 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.

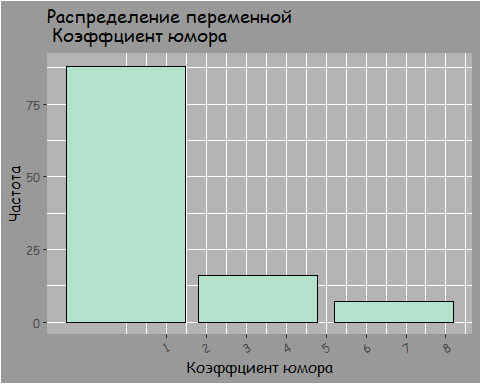
### Социальный статус



|  |  |
| --- | --- |
| 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*:

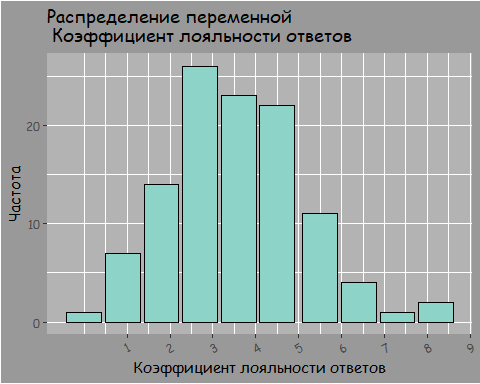
Неоднозначность названия критерия объясняется тем, что высший балл достался бы тому, чьи ответы были бы “кафельный пол”, “пенсионер” и “нефильтрованная тема”. Психология подсказывает три возможные причины таких ответов:

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*:

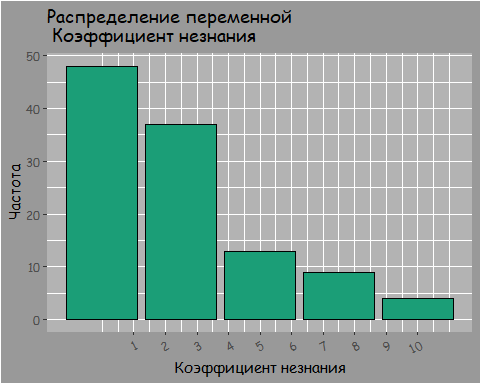
1. ~~Хорошо, что~~ В данных не попалось ни одного “пенсионера кафельного пола, предпочитающего нефильтрованную тему”. Это говорит о неокончательной маргинализации программистов.
2. Программисты очень ответственно относятся к заполнению самых абсурдных форм и не находят юмор всегда уместным.

### 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

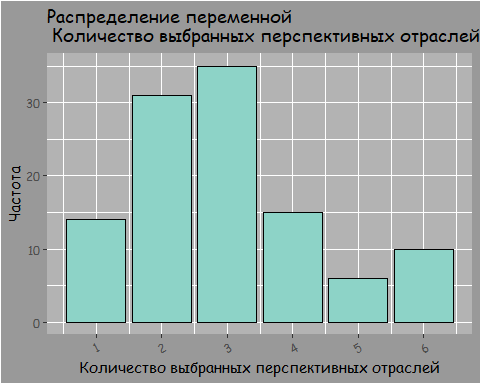
 *Примечание*:

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

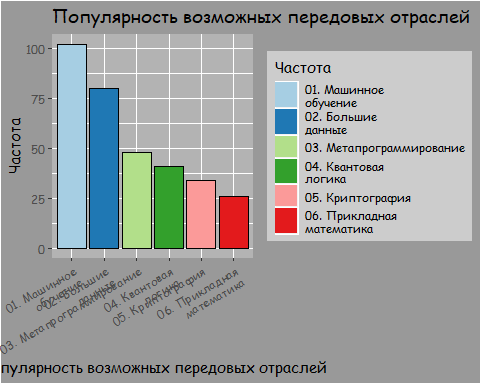


|  |  |
| --- | --- |
| x | Freq |
| 1 | 0.126 |
| 2 | 0.279 |
| 3 | 0.315 |
| 4 | 0.135 |
| 5 | 0.054 |
| 6 | 0.090 |

*Выводы*:

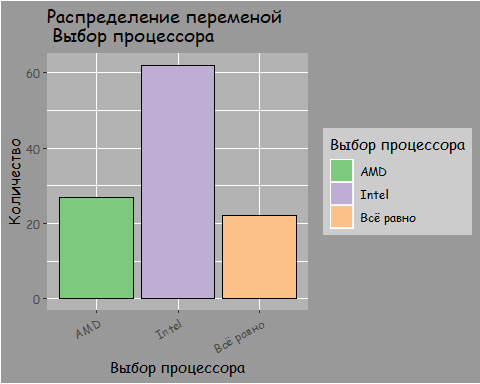
1. Часть испытуемых тупо выбрала всё, что не есть хорошо
2. Оптимальное число возможных перспективных отраслей - 3

### Самые популярные возможные перспективные отрасли

 *Выводы*:

## Железо

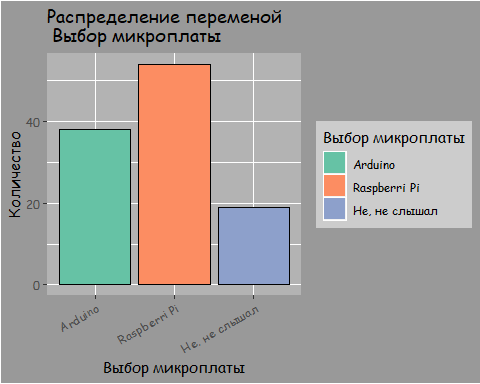
### Выбор процессора



|  |  |
| --- | --- |
| 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

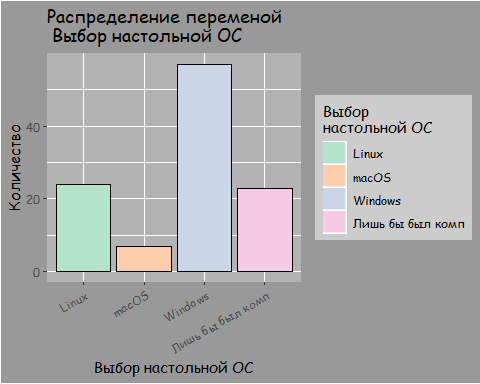


|  |  |
| --- | --- |
| x | Freq |
| Arduino | 0.342 |
| Raspberri Pi | 0.486 |
| Не, не слышал | 0.171 |
| *Выводы*: |  |

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

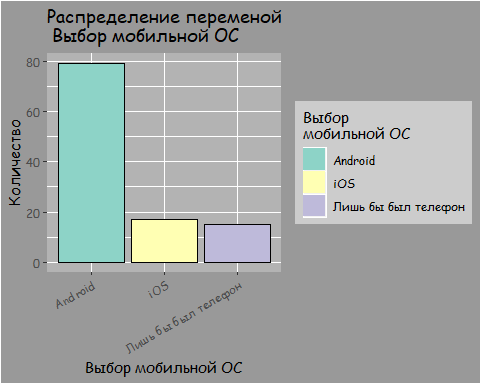
### Предпочитаемая настольная ОС



|  |  |
| --- | --- |
| 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.

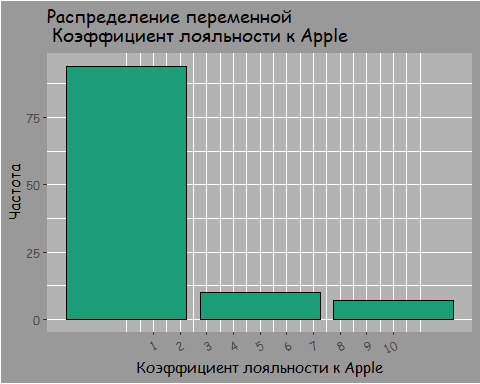
### Предпочитаемая мобильная ОС



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

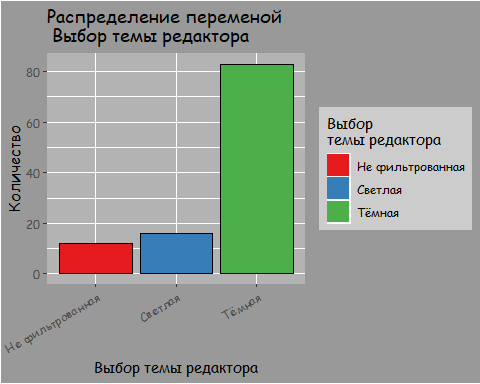
1. Доля предпочтения андроида указывает на успешность социалистической революции и процветание пролетариата. Доля ~~буржуев~~ пользователей Apple в мобильном секторе значительно меньше.

### 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

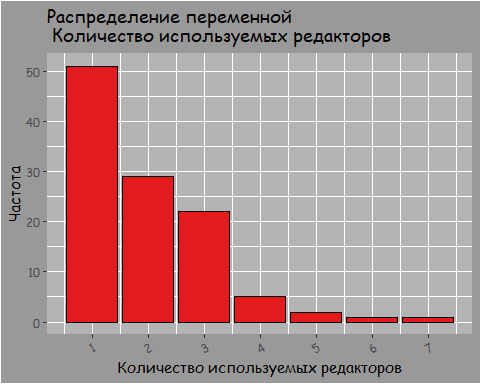
### Тема редактора кода



|  |  |
| --- | --- |
| 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.

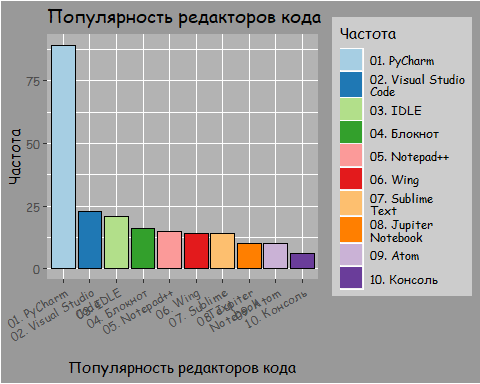
### Количество редакторов, используемых программистом



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

*Выводы*:

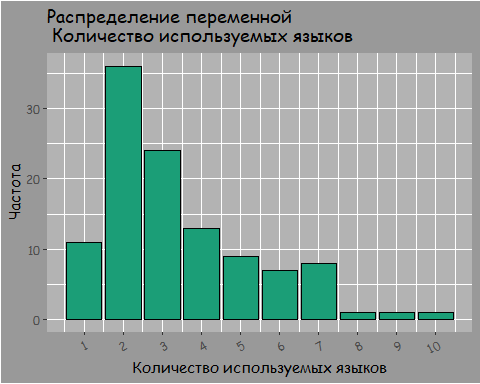
### Самые популярные редакторы



*Выводы*:

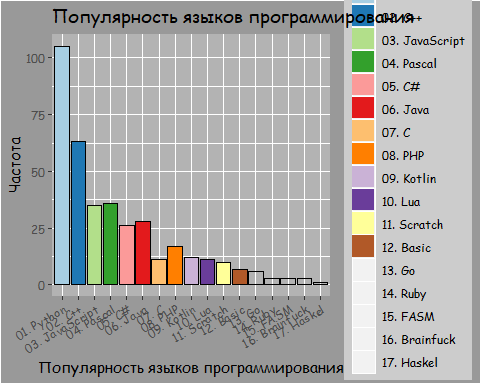
## Общепрограммистские вопросы

### Количество языков программирования, которое знает программист

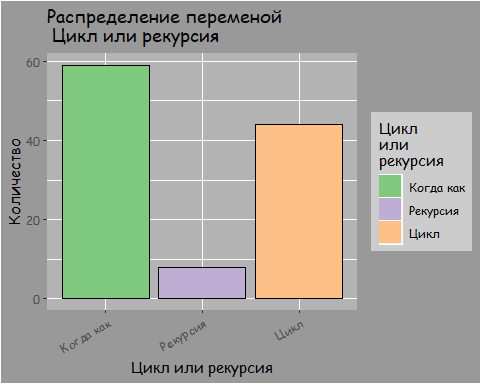


|  |  |
| --- | --- |
| x | Freq |
| 1 | 0.099 |
| 2 | 0.324 |
| 3 | 0.216 |
| 4 | 0.117 |
| 5 | 0.081 |
| 6 | 0.063 |
| 7 | 0.072 |
| 8 | 0.009 |
| 9 | 0.009 |
| 10 | 0.009 |
| \*Выво | ды\*: |

### Самые популярные языки программирования

 *Выводы*:

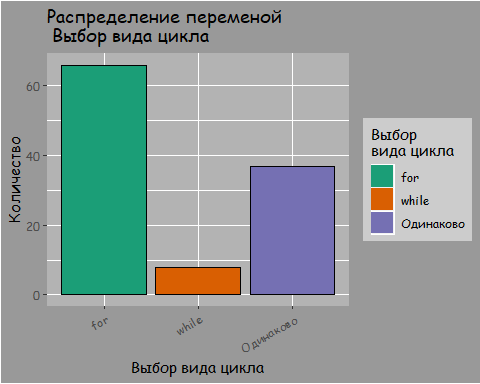
### Цикл или рекурсия?



|  |  |
| --- | --- |
| 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.

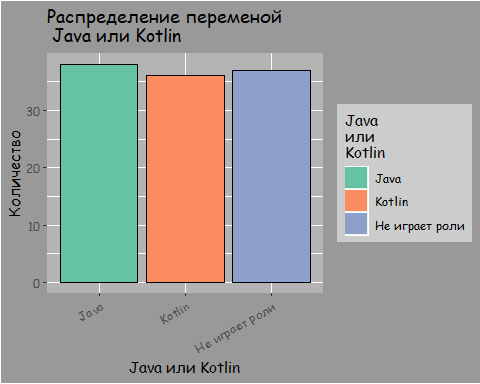
### For или While?



|  |  |
| --- | --- |
| 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.

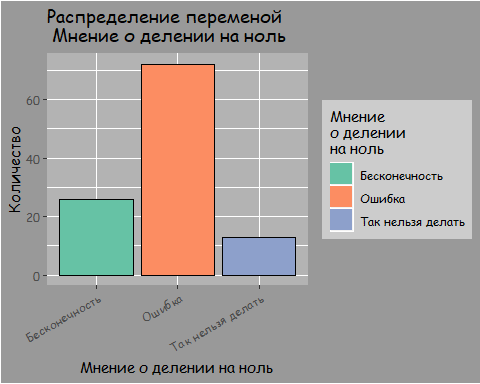
### Java или Kotlin?



|  |  |
| --- | --- |
| 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.

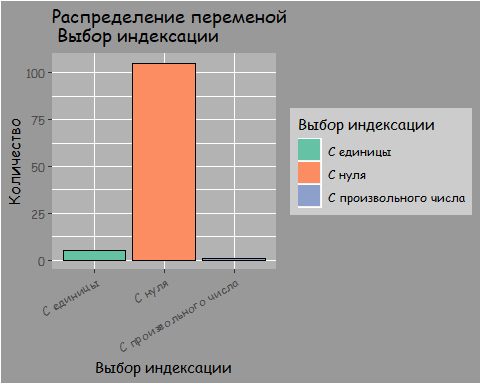
### Мнение о делении на ноль



|  |  |
| --- | --- |
| 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

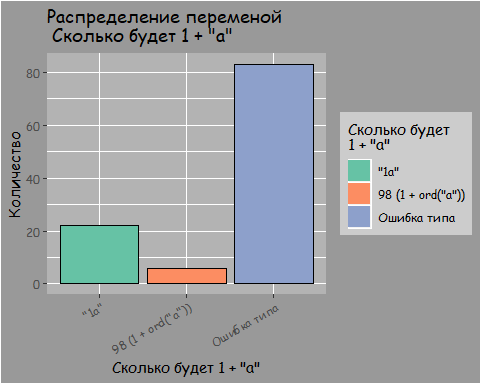


|  |  |
| --- | --- |
| x | Freq |
| С единицы | 0.045 |
| С нуля | 0.946 |
| С произвольного числа | 0.009 |
| *Выводы*: |  |

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.

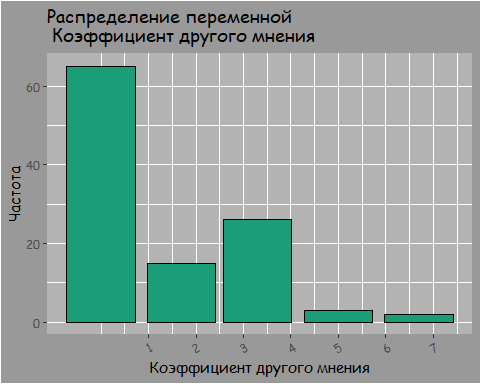
### Сколько будет 1 + “a”?



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

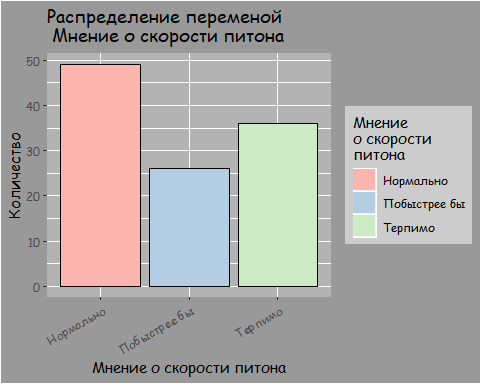
1. Большиство, вторя всяким питонам и джавам, говорит, что смешение типов невозможно.
2. Доля любителей слабой типизации достаточно велика.

### Другое мнение

 *Выводы*:

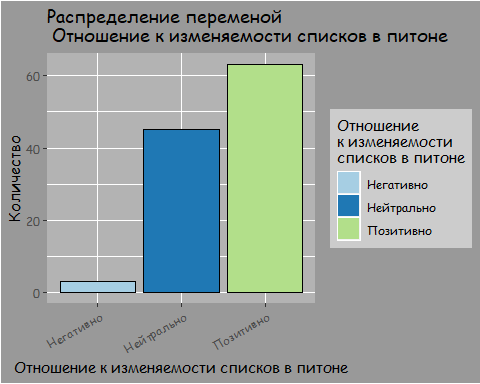
## Вопросы о питоне

### Мнение о скорости питона



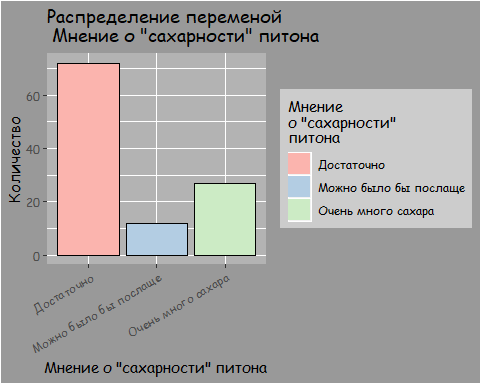
|  |  |
| --- | --- |
| x | Freq |
| Нормально | 0.441 |
| Побыстрее бы | 0.234 |
| Терпимо | 0.324 |
| *Выводы*: |  |

### Отношение к изменяемости списков



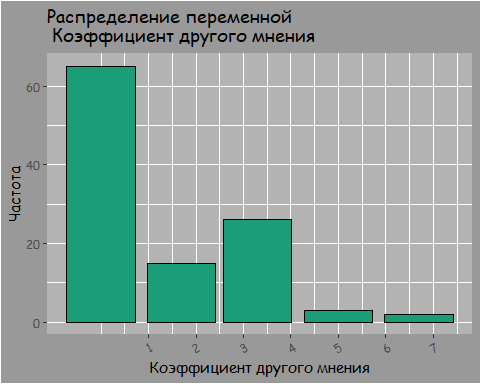
|  |  |
| --- | --- |
| x | Freq |
| Негативно | 0.027 |
| Нейтрально | 0.405 |
| Позитивно | 0.568 |
| *Выводы*: |  |

### Мнение о “сахарности” питона



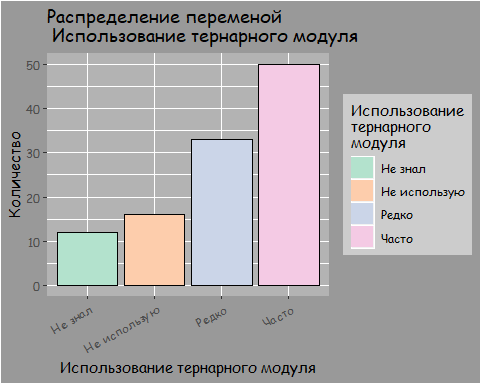
|  |  |
| --- | --- |
| x | Freq |
| Достаточно | 0.649 |
| Можно было бы послаще | 0.108 |
| Очень много сахара | 0.243 |
| *Выводы*: |  |

### Недовольство питоном

 *Выводы*:

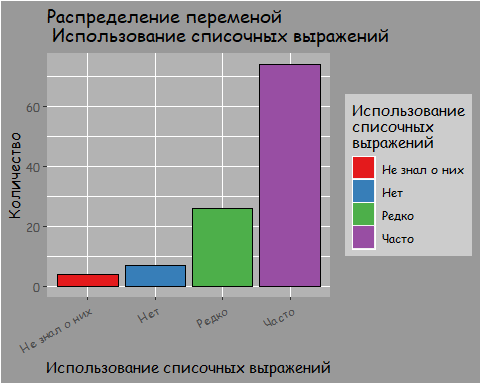
## Вопросы о использовании синтаскического сахара

### Использование тернарного модуля



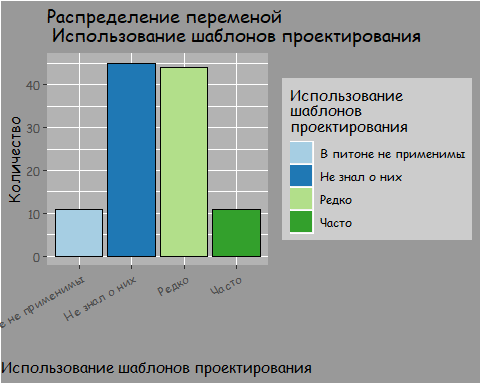
|  |  |
| --- | --- |
| x | Freq |
| Не знал | 0.108 |
| Не использую | 0.144 |
| Редко | 0.297 |
| Часто | 0.450 |
| *Выводы*: |  |

### Использование списочных выражений



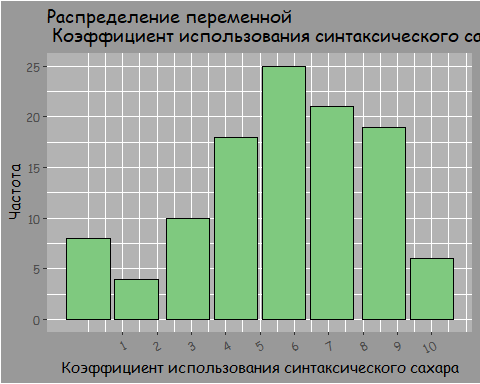
|  |  |
| --- | --- |
| x | Freq |
| Не знал о них | 0.036 |
| Нет | 0.063 |
| Редко | 0.234 |
| Часто | 0.667 |
| *Выводы*: |  |

### Использование шаблонов проетирования



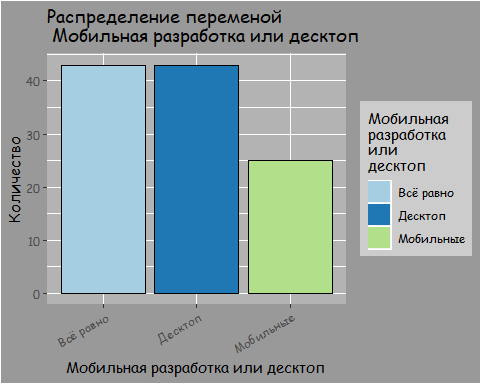
|  |  |
| --- | --- |
| x | Freq |
| В питоне не применимы | 0.099 |
| Не знал о них | 0.405 |
| Редко | 0.396 |
| Часто | 0.099 |
| *Выводы*: |  |

### Использование синтаксического сахара

 *Выводы*:

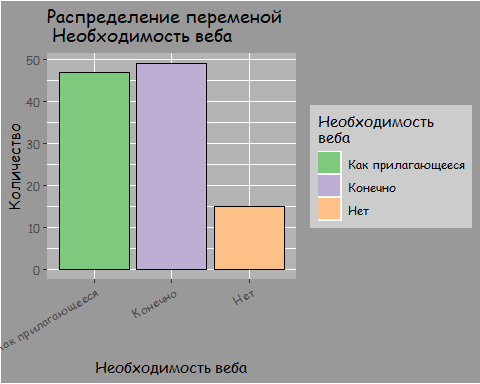
## Вопросы о сферах программирования

### Мобильная или десктоп разработка?



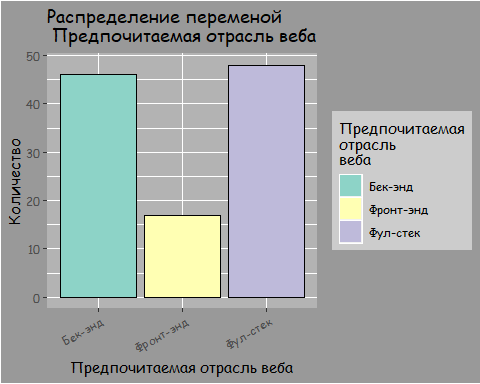
|  |  |
| --- | --- |
| x | Freq |
| Всё равно | 0.387 |
| Десктоп | 0.387 |
| Мобильные | 0.225 |
| *Выводы*: |  |

### Необходимость веба



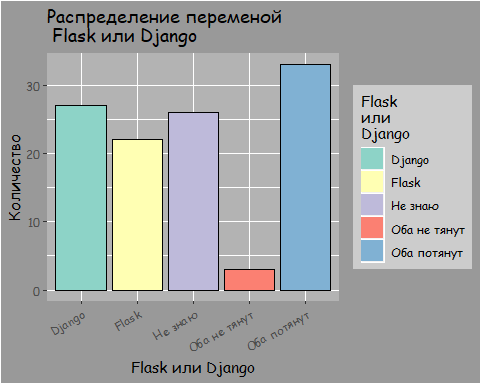
|  |  |
| --- | --- |
| x | Freq |
| Как прилагающееся | 0.423 |
| Конечно | 0.441 |
| Нет | 0.135 |
| *Выводы*: |  |

### Бек или фронт энд?



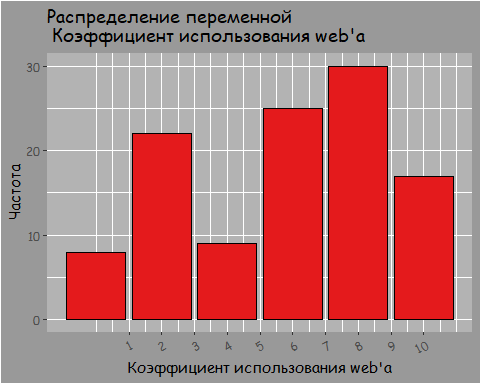
|  |  |
| --- | --- |
| x | Freq |
| Бек-энд | 0.414 |
| Фронт-энд | 0.153 |
| Фул-стек | 0.432 |
| *Выводы*: |  |

### 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*: |  |

### Использование веба

 *Выводы*: