Как начать применять R в Enterprise

Пример практического подхода

Шутов Илья, Devoteam

2022/02/20

План презентации

- Насколько этот вопрос актуален?
- Специфика задач в Enterprise
- Практика создания собственных команд

[1] На вопрос “Почему R, а не Python?” ответ есть, но за рамками доклада

Почему этот вопрос актуален?

Бизнес-кейсы различны, техническая суть одинакова

- Аналитика работы колл-центра
- Аналитика продаж, включая прогнозы
- Антифрод системы
- Business process mining
- Различные аудиты (технические, финансовые)
- Складские и логистические задачи
- Activity-based costing
- Business-process monitoring
- Log-based аналитика
- Capacity management
- Текстовая аналитика (e-mail, service-desk)
- "Гибкие" дашборды и отчеты
- "интеллектуальные шины" между учетными системами (1С, СКУД, SAP, ...) и исполнительными
- ...

Практические наблюдения

- очень многие подобные задачи сводятся к математической манипуляции с данными (CRUD системы за рамками, рассматриваем именно разнообразный процессинг и преобразование);
- 80% задач по манипуляции данными могут быть быстро и эффективно решены "под ключ" путем применения инструментария R;
- в бизнесе, как правило, задачи и требования быстро корректируются, в т.ч. из-за внешних факторов или полученных промежуточных результатов;
- "модульные" технологии хорошо приживаются и в ИТ; строительство "монолита" может занять 2-3 года, что сопоставимо со сроком жизни небольшого решения. Гораздо эффективнее быстро собрать "модульную" конструкцию, накопить практического опыта и через 2-3 года скомпоновать новое решение с учетом полученных знаний и прошедших изменений в ИТ и бизнесе.

Типичные “городские легенды” про R

- R медленный
- R трудночитаемый
- R предназначен для стат. расчетов по сложных алгоритмам
- R предназначен для интерактивной работы

Все это возникает из-за поверхностного изучения темы и используемых инструментов.

Городские легенды — заблуждения из "90-х"

- R -- полноценный язык программирования, а не консольный калькулятор.
- R хорошо выступает в качестве универсального “клея” между различными платформами и C компонентами -- считает быстро!
- Читаемость кода зависит от опыта разработчика. Современный стиль R -- метапрограммирование. Код компактен и быстр.
- R -- это экосистема, позволяет реализовать полный цикл обработки данных от импорта данных до предоставления API и подготовки презентаций.

Специфика решения задач в Enterprise

Аналитические приложения не исключения

- железа обычно дают мало
- часто мало времени (как на разработку, так и на починку)
- нужна большая скорость работы
- нужна стабильность в режиме 24x7
- безопасность (LDAP, логирование, управление доступами)
- интеграция непонятно с чем и непонятно как
- требуется разработка APM
- многократная перегрузка (10-кратное превышение расчетной нагрузки в порядке вещей) + завсегда умудряются подсунуть кривые данные
- данных может быть как мало, так и очень много
- DevOps (git, автотесты, автосборки, ведение нескольких релизов, формирование тестового окружения); деплоймент в продуктив (как правило, только Linux)

Решения на R вполне вписываются в эти требования 1/2

- быстрые пакеты позволяют использовать мало железа
- метапрограммирование позволяет ускорить процесс разработки
- развитая поддержка методов функционального программирования
- NSE как средство для минимизация кода и устранения дублирования (чем меньше кода, тем меньше ошибок)
- наличие адаптеров/пакетов для типовых систем/протоколов в Enterpise
- возможность использования всей мощи ML (встроенная и внешняя)

Решения на R вполне вписываются в эти требования 2/2

- Shiny (пакет + сервер) для разработки Web APM с поддержкой подходов реактивного программирования
- развитые пакеты для реализации подхода Defensive Programming (ассерты, валидация)
- пакеты для логирования, технологии авторизации APM через LDAP, возможность управления доступами на уровне приложений
- пакеты и технологии для поддержки элементов DevOps (работа с git, автотесты, автодокументирование), широкий спектр технологий деплоймента в продуктив

Звучит интригующе, а делать то что?

Для работы над реальными задачами вполне возможно создать собственную команду:

- 2-3 человека на этапе старта;
- 1 месяц на "погружение" в тему;
- постоянное развитие в доп. темах и предметной области;
- 1 год на активное решение практических задач под руководством опытного человека и при коммуникациях с сообществом);
- подведение промежуточных итогов, корректировка планов на будущее.

Как создать свою команду R разработчиков

Стили разработки

Популярные стили:

- “У нас в школе были уроки информатики”
- Base R
- DB разработчик
- ООП разработчик

Посмотрим, оказывает ли влияние выбранного стиля на конечный результат стиль.

Newbie. “У нас в школе были уроки информатики”

1/6 кода, почти бесконечное время исполнения, невозможность осознать логику переходов.

```
for (i in 1:rownum) {
    print(i)
    a ← res[i,1]
    print(res[i,1])

  if (is.na(res[i,10])) {break}

   if (res[i,1] == res[i+1,1]) {  # Маркер начала и конца относиться к разным точкам?

                if (res[i,10] == mst) {#Берем 1 строку маркера начала, проверяем с последующей
                      if (res[i+1,10] == msu) {
                          one←res[i,]
                          two←res[i+1,]
                          r ← bind_rows(bind_cols(list(one,two),.id = i),r)   #Объединяем
                          r$STATUS[1] ← 'Успех'
                          i = i + 2 #  Встаем на позицию следущей операции начала маркера и конца окончания
                      } else if (res[i+1,10] == mst){
                          bindr(r,i)
                          r$STATUS[1] ← 'Нет маркера окончания'
                      } else  {
                          bindr(r,i)
                          r$STATUS[1] ← 'Нет окончания маркера'
                          bindr(r,i+1)
                          r$STATUS[1] ← 'Неизвестный маркер'
                          i = i + 2
                      }
                } else if (res[i,10] == msu) {
                  # Допущение, при проверки маркера окончания (не обрабатываю,
                  # если после маркера окончания идет маркер начала, при неправильной сортировки,
                  # подразумевается  сортировка данных корректная)
                            if (res[i+1,10] == mst) {
                                          bindr(r,i)
                                          r$STATUS[1] ← 'Нет маркера начала'
                             } else if (res[i+1,10] == msu) {
                                          bindr(r,i)
                                          r$STATUS[1] ← 'Нет окончания маркера'
                               } else {
                                          bindr(r,i)
```

Newbie: как можно переписать ("Tidyverse way")

```
df <- df0 %>%
  # сортировка по времени от раннего к более позднему
  arrange(point, pos, timestamp) %>%
  mutate(grp = if_else(stri_detect_fixed(msg, "***  REQUEST ***"),
                       row_number(), as.integer(NA))) %>%
  # спускаем номер группы на все терминальные секции ниже
  fill(grp, .direction = "down") %>%
  # оставляем только первые две записи в полных группах
  group_by(grp) %>%
  filter(n() >= 2) %>%
  slice(1:2) %>%
  ungroup() %>%
  # развесим маркеры
  mutate(type = if_else(row_number() %% 2 == 1, "begin", "end")) %>%
  # разворачиваем и считаем разницу во времени
  select(-msg) %>%
  nest(point, pos) %>%
  tidyr::spread(key = "type", value = timestamp) %>%
  unnest() %>%
  mutate(delta = end - begin) %>%
  arrange(desc(delta))
```

Base R

```
DF3 ←data.frame(ifelse
(total_part_buffer_185_present_time[3]>0&total_part_buffer_185_previous_time[3]<0
, (abs(total_part_buffer_185_present_time[,3])+abs(total_part_buffer_185_previous
_time[,3]))/abs(total_part_buffer_185_previous_time[,3]),
                        ifelse
(total_part_buffer_185_present_time[3]<0&total_part_buffer_185_previous_time[3]>0
, (total_part_buffer_185_present_time[,3]-total_part_buffer_185_previous_time[,3]
)/total_part_buffer_185_present_time[,3]*(-1),
                                ifelse
(total_part_buffer_185_present_time[3]<0&total_part_buffer_185_previous_time[3]<0
&(total_part_buffer_185_present_time[3]<total_part_buffer_185_previous_time[3]),
(total_part_buffer_185_present_time[,3]-total_part_buffer_185_previous_time[,3])/
total_part_buffer_185_previous_time[,3]*(-1),
                                        ifelse
(total_part_buffer_185_present_time[3]<0&total_part_buffer_185_previous_time[3]<0
&(total_part_buffer_185_present_time[3]>total_part_buffer_185_previous_time[3]),
(total_part_buffer_185_present_time[,3]-total_part_buffer_185_previous_time[,3])/
total_part_buffer_185_present_time[,3],

(total_part_buffer_185_present_time[,3]-total_part_buffer_185_previous_time[,3])/
total_part_buffer_185_previous_time[,3])))))
```

Base R: как можно переписать ("Tidyverse way")

```
marginTest <- tibble(
  margin1 = c(-100,-200, -100, 100),
  margin0 = c(-200,-100,200, -100)
)

marginTest %>%
  mutate(
    deviation = (pmax(margin0, margin1) - pmin(margin0, margin1)) /
      pmin(abs(margin0), abs(margin1)) * sign(margin1)
  )
```

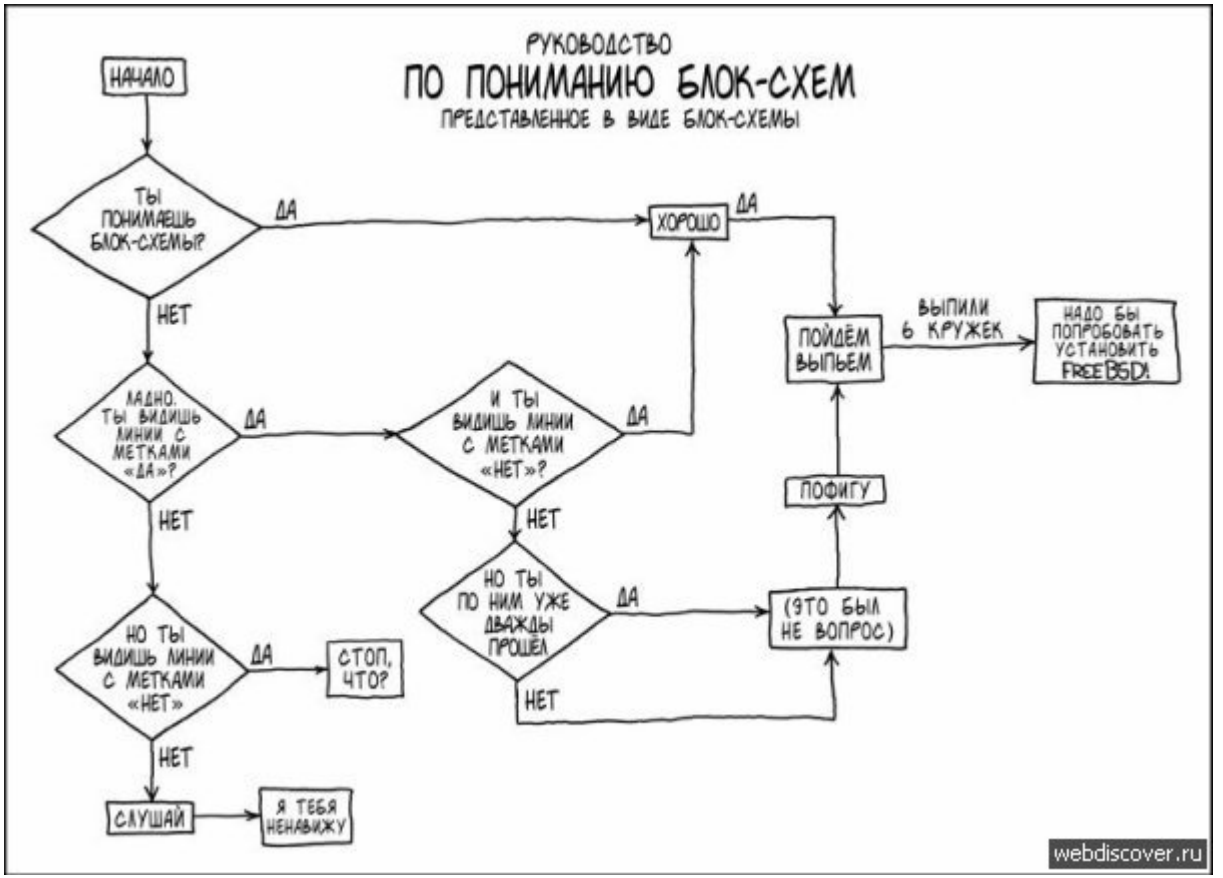
```
## # A tibble: 4 x 3
##   margin1 margin0 deviation
##     <dbl>   <dbl>     <dbl>
## 1    -100    -200        -1
## 2    -200    -100        -1
## 3    -100     200        -3
## 4     100    -100         2
```

DB разработчик

```
# Step 5-1: Level 1 _________________________
rows_processed <- NULL
res_level_1 <- base %>%
  inner_join(off_by_level[["level_1"]], by = c(
    "name", "domain"="terr", "modelunit")) %>%
  share_ff() %>%
  mutate(level="name")

# Step 5-2: Level 2 _________________________
# аккумулируем уже запроцессированные строки
rows_processed <- res_level_1 %>%
  select(ticker, sign, domain, name) %>%
  distinct()

res_level_2 <- base %>%
  anti_join(rows_processed, by = c(
    "ticker", "sign", "domain", "name")) %>%
  inner_join(off_by_level[["level_2"]], by = c(
    "group", "domain"="terr", "sign")) %>%
  share_ff() %>%
  mutate(level="middle")

# Step 5-3: Level 3 _________________________
# аккумулируем уже запроцессированные строки
```

DB разработчик: как можно переписать

Рисуем блок-схему, описываем булевой логикой, считаем.



http://webdiscover.ru/59675.html

Простынка флагов

```
# FullSimplify[lleaf || mleaf || rleaf ]
# # f21 || (f22 && f27) || (! f22 && f23 && f24 && f25 && f26) ||
# f28 || f29 || f21 || (f22 && (f210 || f27 || f28 || f29) ) ||
# f211 || f212 || (! f22 && f23 && f24 && f25 && f26)
```

flag_1_1	flag_1_2	flag_1_3	flag_2_1	flag_2_2	flag_2_3	flag_2_4	flag_2_5	flag_2_6	flag_2_7	flag_2_8	flag_2_9	flag_2_10	flag_2_11	flag_2_12	DVT_is_validated	type	bp_2_1_in	bp_2_2_in
FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE	TRUE	T: Валидный	TRUE	FALSE
FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	T: Валидный	TRUE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	T: Валидный	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	T: Валидный	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	T: Валидный	TRUE	TRUE
FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	T: Валидный	TRUE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	T: Валидный	TRUE	TRUE
FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	T: Валидный	TRUE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	T: Валидный	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	T: Валидный	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	T: Валидный	TRUE	TRUE
FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	T: Валидный	TRUE	FALSE
FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	T: Валидный	TRUE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	T: Валидный	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	T: Валидный	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	T: Валидный	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	TRUE	T: Валидный	TRUE	TRUE
FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	T: Валидный	TRUE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	T: Валидный	TRUE	TRUE
FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	T: Валидный	TRUE	FALSE
FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	T: Валидный	TRUE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	T: Валидный	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	T: Валидный	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	T: Валидный	TRUE	TRUE
FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	TRUE	T: Валидный	TRUE	FALSE
FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	TRUE	T: Валидный	TRUE	FALSE

Showing 1 to 32 of 1,811,502 entries, 59 total columns

ООП подход

```
ItemRatingMatrix ← R6Class(
  classname = "ItemRatingMatrix",

  public = list(
    initialize = function(inputData) {
      private$inputData ← inputData
    },

    get = function() { private$sparseRatingMatrix },
    getUserDict = function() { private$userDict },
    getItemDict = function() { private$itemDict },
    getUserCount = function() private$ratingMatrixStats$users_count,
    getItemCount = function() private$ratingMatrixStats$items_count,
    getSparsity = function() private$ratingMatrixStats$sparsity
  ),

  active = list(
    ratingMatrix = function(x) {
      if (missing(x)) {
        ratingMatrix ← private$sparseRatingMatrix
        if (!is.null(ratingMatrix)) {
          return(ratingMatrix)
        } else {
          stop("Sparse rating matrix is NULL")
        }
      } else {
        stop("Can't assign to sparse rating matrix")
      }
    }
  ),

  private = list(
    inputData = NULL,
    userDict = NULL,
    itemDict = NULL,
    sparseRatingMatrix = NULL,
    ratingMatrixStats = NULL,

    prepareSalesData = function() {
      private$userDict ← private$calculateUserDict()
      private$itemDict ← private$calculateItemDict()

      salesData ← private$inputData$sales_tbl %>%
```

ООП подход: как можно переписать

- В задачах обработки данных ООП становится гирей на ногах. В центре должны быть именно данные:
    - данные могут быть большими, необходимо избегать ненужного копирования данных;
    - процесс обработки должен быть понятен, относительно линеен и локализован;
    - данные должны быть доступны на этапе отладки в любой точке;
    - необходимо избегать side-effect, присущий методам при инкапсуляции данных в рамках ООП;
    - изменение структуры data.frame объекта внутри методов класса приводит к труднообъяснимой нетранзитивности процесса обработки.

Важен ли стиль программирования?

Для одноразовой работы стиль обычно не критичен. Главное получить результат. При решении сложных повторяющихся задач эффективность становится критичной. Поверхностное знакомство с инструментом либо приведет к медленному и неповоротливому решению (с возможными скрытыми ошибкам), либо вообще не позволит решить задачу.

Далее несколько частых задач и примеры их решения на R различными способами:

- Тайминг по копированию.
- Тайминг по итерированию по вектору.
- Тайминг по нахождению ТОП-5 по группам.
- Тайминг итерирования по строкам.
- Хорошо ли Вы знакомы с типом `POSIXct`?
- Решение задачи 76*4*51*3 (математика, 3-ий класс).
- Комбинаторика. 'Минусы в квадрате' (математика, 3-ий класс).

Тайминг по копированию

Высокуровневые многослойные концепции зачастую полностью уничтожают производительность решения. Отправная точка -- скрытые манипуляции с памятью: выделение, очистка, копирование. Сравним варианты.

```
library(data.table)
setDTthreads(0)

vec3 <- rep(3, 10^3)
vec4 <- rep(4, 10^4)
vec5 <- rep(5, 10^5)
vec6 <- rep(6, 10^6)
vec7 <- rep(7, 10^7)
vec8 <- rep(8, 10^8)
```

```
bench::mark(
  "10^3_copy" = {x <- vec3+1},
  "10^4_copy" = {x <- vec4+1},
  "10^5_copy" = {x <- vec5+1},
  "10^6_copy" = {x <- vec6+1},
  "10^7_copy" = {x <- vec7+1},
  "10^8_copy" = {x <- vec8+1},
  # ---- без присвоения -----
  "10^3" = {vec3 + 1; NULL},
  "10^4" = {vec4 + 1; NULL},
  "10^5" = {vec5 + 1; NULL},
  "10^6" = {vec6 + 1; NULL},
  "10^7" = {vec7 + 1; NULL},
  "10^8" = {vec8 + 1; NULL},
  check = FALSE,
  relative = TRUE
)
```

Тайминг по копированию, результаты

Точно учесть все происходящие под капотом процедуры, такие, как выделение памяти и запуск сборщика мусора, не представляется возможным. Но картина в целом выглядит весьма показательной. Простое присвоение (копирование) тоже **требует времени**. Несколько необдуманных копирований и порядок по скорости потерян.

expression <chr>	min <dbl>	mean <dbl>	median <dbl>	max <dbl>	itr/sec <dbl>	mem_alloc <dbl>
10^3_copy	1.0	1.789252e+00	1.6	1.000000	2.485348e+05	1.000000
10^4_copy	59.5	2.472084e+01	44.0	1.431567	1.798852e+04	9.946322
10^5_copy	461.5	4.155248e+02	469.2	17.586369	1.070192e+03	99.409543
10^6_copy	7334.5	3.818280e+03	8126.6	28.294426	1.164638e+02	994.041750
10^7_copy	938763.5	1.579267e+05	375505.4	518.081402	2.815809e+00	9940.363817
10^8_copy	2643378.5	4.446913e+05	1057351.4	1458.818157	1.000000e+00	99403.584493
10^3	1.0	1.000000e+00	1.0	3.318157	4.446913e+05	1.000000
10^4	12.0	1.360891e+01	8.2	8.498344	3.267648e+04	9.946322
10^5	542.5	2.797245e+02	306.1	107.987583	1.589747e+03	99.409543
10^6	2107.0	8.011244e+02	1618.0	19.233996	5.550839e+02	994.041750
10^7	707522.0	1.190253e+05	283008.8	390.464680	3.736108e+00	9940.363817
10^8	1078257.0	1.813934e+05	431302.8	595.064570	2.451529e+00	99403.584493

Тайминг итерирования по вектору

```r
idx <- 1:500000

# вариант 1 -- цикл + растущий вектор
grow_cycle <- function(idx){
  res <- NULL
  for(i in idx){ res <- c(res, runif(1)) }
  res }

# вариант 2 -- цикл + заранее созданный вектор
fixed_cycle <- function(idx){
  res <- numeric(length = length(idx))
  for(i in idx){ res[[i]] <- runif(1) }
  res }

vec_init <- function(idx){ runif(length(idx)) }
```

expression <chr>	min <dbl>	mean <dbl>	median <dbl>	max <dbl>	itr/sec <dbl>	mem_alloc <dbl>
growing_array	36342.854	32640.2139	33833.2530	23033.6557	1.0000	250164.1585
preallocated_array	180.146	161.7926	167.7063	114.1743	201.7411	319.7921
fully_vectorized	1.000	1.0000	1.0000	1.0000	32640.2139	1.0000

Тайминг по нахождению ТОП-5 по группам (1/3)

Готовим сэмплы

```
library(tidyverse)
library(gapminder)

data <- gapminder %>%
  distinct(country) %>%
  mutate_at(vars(country), as.character) %>%
  # на каждую страну
  mutate(value = list(rnorm(10^4, mean = 100, sd = 10))) %>%
  unnest(value) %>%
  # физически перемешаем строки
  .[sample(nrow(.)), ]
```

Тайминг по нахождению ТОП-5 по группам (2/3)

```
dt <- as.data.table(data)
dt2 <- as.data.table(data) %>% setorder(country, -value)
data2 <- data %>% arrange(desc(value))

bench::mark(
  dplyr_naive = {
    data %>%
      group_by(country) %>%
      top_n(n = 5, wt = value) # есть нюансы %>%
      ungroup() },
  dplyr_smart = {
    data2 %>%
      group_by(country) %>%
      slice(1:5) %>%
      ungroup() },
  dt_naive = dt[, head(.SD, 5), by = country],
  dt_smart1 = dt2[, idx := seq_len(.N), by = country
                  ][idx <= 5, .(country, value)],
  dt_smart2 = dt2[dt2[, .I[seq_len(.N) <= 5L], country]$V1]
)
```

Тайминг по нахождению ТОП-5 по группам (3/3)

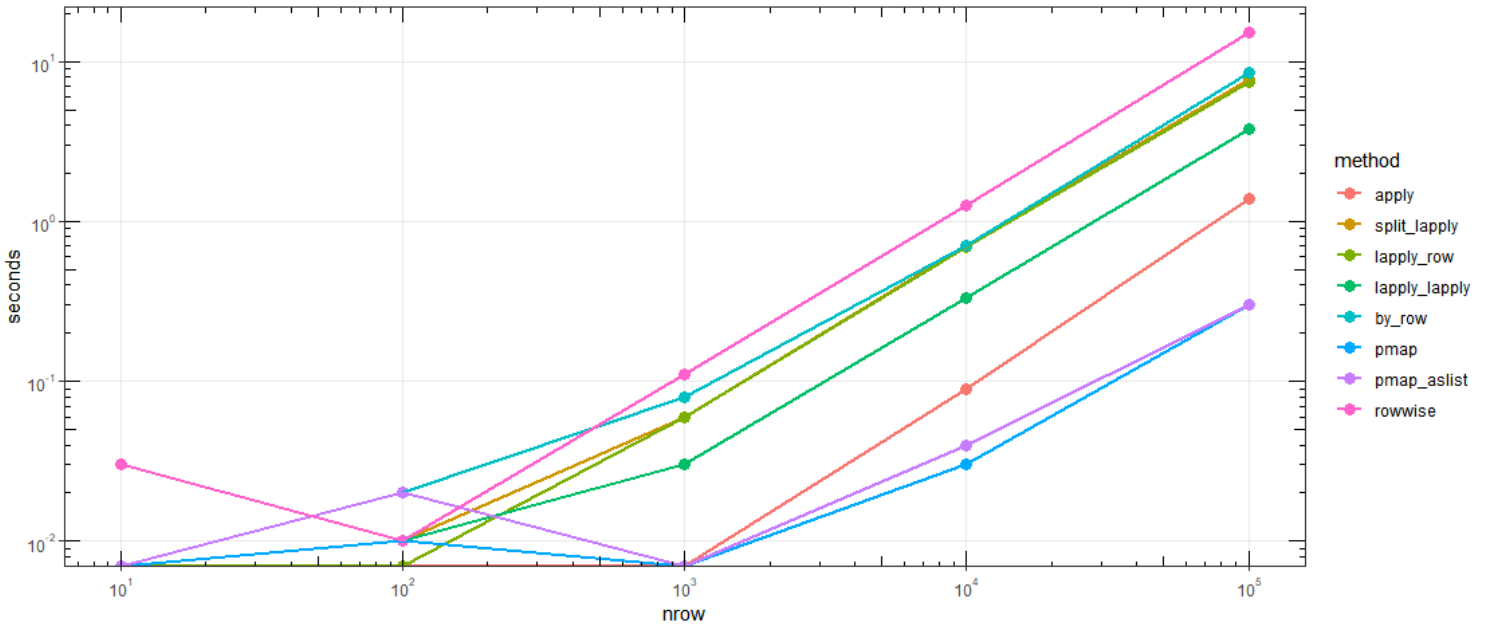
Тест имеет смысл только при совершении регулярных выборок из массива данных. Время на предварительную оптимизацию, которое в любом случае будет потрачено, при единичном исполнении может нивелировать алгоритмические трюки.

expression <chr>	min <dbl>	mean <dbl>	median <dbl>	max <dbl>	itr/sec <dbl>	mem_alloc <dbl>	n_gc <dbl>
dplyr_naive	20.240282	19.838844	18.604314	11.185455	1.000000	16.437930	2.36
dplyr_smart	5.380248	5.316222	4.827279	5.767941	3.731756	1.932270	1.00
dt_naive	2.276617	2.247050	2.094030	1.747078	8.828841	1.000000	1.08
dt_smart1	1.189364	1.627906	1.351681	1.788860	12.186728	4.829975	5.36
dt_smart2	1.000000	1.000000	1.000000	1.000000	19.838844	3.871297	7.08

Тайминг итерирования по строкам

- Applying a function over rows of a data frame by Winston Chang;
- Row-oriented workflows in R with the tidyverse by Jenny Brian

	nrow	apply	split_lapply	lapply_row	lapply_lapply	by_row	pmap	pmap_aslist	rowwise
1	1e+01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03
2	1e+02	0.00	0.01	0.00	0.01	0.02	0.01	0.02	0.01
3	1e+03	0.00	0.06	0.06	0.03	0.08	0.00	0.00	0.11
4	1e+04	0.09	0.71	0.69	0.33	0.70	0.03	0.04	1.25
5	1e+05	1.39	7.72	7.45	3.81	8.46	0.30	0.30	15.27



Хорошо ли Вы знакомы с типом `POSIXct`?

Задача типичная для аналитики распределенных событий. ИТ система регистрирует события по UTC, аналитика (например, аналитика телесмотрения) должна осуществлятся в рамках локального времени каждого пользователя.

Проблемка! `POSIXct` -- вектор с атрибутом таймзоны. Поэтому сложно векторизовать для разных tz, а в `data.frame` так вообще невозможно иметь колонку с временной меткой с учетом зоны. Собрано по мотивам Making Faceted Heatmaps with ggplot2

Сэмулируем набор случайных временных данных: 500,000 событий в 50 различных таймзонах.

```
head(events_tbl, n = 3)
```

```
## # A tibble: 3 x 2
##   timestring          tz
##   <chr>               <chr>
## 1 2019-04-16 14:19:02 America/Campo_Grande
## 2 2019-04-19 04:25:04 Asia/Thimbu
## 3 2019-04-18 01:04:54 America/Rainy_River
```

POSIXct: достаем локальные час и день событий

```
tic("Direct processing")
events1 <- events_tbl %>%
  mutate(rt = ymd_hms(timestring, quiet = FALSE)) %>%
  mutate(hour = map2_chr(.$rt, .$tz, ~format(.x, "%H", tz = .y)),
         wkday = map2_chr(.$rt, .$tz, ~weekdays(as.Date(.x, tz = .y))
toc()
```

```
## Direct processing: 133.64 sec elapsed
```

```
tic("Smart processing")
parseDFDayParts <- function(df, tz) {
  real_times <- ymd_hms(df$timestring, quiet = TRUE)
  tibble(wkday = weekdays(as.Date(real_times, tz = tz)),
         hour = format(real_times, "%H", tz = tz))}
events2 <- events_tbl %>%
  group_by(tz) %>% nest() %>%
  mutate(res = map2(data, tz, parseDFDayParts)) %>%
  unnest()
toc()
```

```
## Smart processing: 1.936 sec elapsed
```

Решение задачи 76*4*51*3

Можно ли расставить знаки действий вместо *, чтобы значение выражения 76*4*51*3 равнялось 36?

```
c('*', '/', '+', '-') %>%
  {tidyr::crossing(op1 = ., op2 = ., op3 = .)} %>%
  mutate(data = glue::glue("76 {op1} 4 {op2} 51 {op3} 3"),
         expr = rlang::parse_exprs(data),
         res = purrr::map_dbl(expr, rlang::eval_bare)) %>%
  filter(between(res, 27, 56))
```

```
## # A tibble: 3 x 6
##   op1   op2   op3   data             expr         res
##   <chr> <chr> <chr> <S3: glue>      <list>     <dbl>
## 1 -     -     /     76 - 4 - 51 / 3 <language>    55
## 2 /     +     /     76 / 4 + 51 / 3 <language>    36
## 3 +     -     +     76 + 4 - 51 + 3 <language>    32
```

Минусы в квадрате 1/2

Можно ли расставить в клетках квадратной таблицы 4 на 4 десять минусов так, чтобы в каждом столбце было нечётное число минусов, а в каждой строке было чётное число минусов?

```
# пронумеруем все ячейки:
#  1  2  3  4
#  5  6  7  8
#  9 10 11 12
# 13 14 15 16

icomb <- arrangements::icombinations(1:16, k = 10, replace = FALSE)
df <- foreach(x = icomb, .combine = rbind) %do% {
  # x - вектор с номерами позиций, куда ставим знаки "минус"
  # считаем и проверяем условия для комбинаций
  v <- rep(0, 16) # создаем матрицу
  v[x] <- 1
  m <- matrix(v, nrow = 4, ncol = 4, byrow = TRUE)
  if (all(colSums(m) %% 2 == 1) && all(rowSums(m) %% 2 == 0))
    x else NULL
} %>% as_tibble(.name_repair = "minimal")
```

Минусы в квадрате 2/2

```
# а так можно посмотреть конкретную раскладку
v <- rep(0, 16)
# заполним единицами позиции, заполненные минусами
v[purrr::flatten_int(df[5, ])] <- 1
matrix(v, nrow = 4, ncol = 4, byrow = TRUE)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    1    1    1
## [2,]    1    0    1    0
## [3,]    1    1    0    0
## [4,]    0    1    1    0
```

Вопросы производительности и потребления памяти в комбинаторных задачах очень важны. С примером сравнительного анализа различных подходов и пакетов можно ознакомиться, например, здесь: Benchmarking ‘arrangements’.

Путь “быстрое погружение в R”

Ключевая цель — формирование практических навыков по (1) **быстрому** созданию (2) **качественного и эффективного** кода (3) с применением **оптимальных пакетов (= алгоритмов)**.

Подобный путь позволяет посредством интенсивной подготовки сформировать у новых сотрудников базовый набор знаний и навыков, минимально необходимых для начала решения задач с применением R.

- Акцент на инструментальную сторону вопроса; мат. основы и ML относим к предметной области и выносим за рамки.
- Широта рассматриваемых технических вопросов служит для формирования представления и «закладок» о возможностях экосистемы.
- Оптимальная продолжительность первичного погружения 1 блок = 1 день.
- В рамках каждой темы для закрепления материала необходимо разбирать практические задачи разной степени сложности и объемности.
- Поблочная разбивка и "джентельменский" набор пакетов не догма, а рекомендация.

Блок 1: основы языка

- Краткая история R. Основы синтаксиса и структуры языка. Основы применения IDE RStudio для анализа и разработки. Базовые типы и данных. Значения `NA`, `NULL`, `NaN`.
- Интерактивные вычисления и исполнение программного кода. Краткое знакомство с форматами R Markdown и R notebook.
- Принципы работы с библиотеками. Подготовка к аналитической работе, установка необходимых библиотек, создание проекта. Принципы профилировки вычислений, поиск узких (крайне длительных) мест и их устранение.
- Векторизация.
- Оценка скорости исполнения кода: `system.time`, `tictoc`, `benchmark`, `bench`, `profvis`.
- Оценка производительности системы: `benchmarkme`.

Блок 2: вселенная tidyverse

- Концепция и экосистема пакетов `tidyverse`. Краткий знакомство с входящими в него пакетами (импорт/обработка/визуализация/экспорт).
- Концепция tidy data как основа методов работы в `tidyverse`.
- `tibble` как переосмысленный `data.frame` в экосистеме `tidyverse`.
- Преобразования и манипуляции с данными. Синтаксис и принципы потоковой обработки (пакет `magrittr`). Группировка - вычисление - сборка. Пакеты `tibble`, `dplyr`, `tidyr`.

Блок 3: графика в ggplot

- Формирование графических представлений средствами ggplot (https://ggplot2.tidyverse.org/reference/index.html). Использование графических средств для анализа бизнес-данных.

- Примеры виджетов и графиков

    - https://ggplot2.tidyverse.org/index.html
    - http://www.ggplot2-exts.org/gallery/
    - http://gallery.htmlwidgets.org
    - https://www.r-graph-gallery.com

- Мини Табло. Add-in esquisse

- Создание анимации. Пакет `gganimate`

Блок 4: строки и время

- Работа со строковыми и перечислимыми типами. Основы регулярных выражений. BaseR; пакеты `stringi`, `stringr`, `re2r`.
- Работа с датами и временем. Сущности: период, длительность, интервал. BaseR; пакеты `lubridate`, `anytime`.
- Работа с временными рядами (классика и tidy вариант).

Блок 5: импорт данных

- Расширенный импорт данных txt, csv, json, odbc, web scrapping (REST API), xlsx. (Пакеты `readr`, `opexlsx`, `jsonlite`, `curl`, `rvest`, `httr`, `readtext`, `DBI`, `data.table`).
- Рабочая директория. `getwd()`, `setwd()`. Применение пакета `here`
- Пакет и add-in datapasta.
- Введение в `readr`. Особенности специфицирования колонок.
- работа с Excel:
  - `readxl`
  - `openxlsx`
  - `tidyxl`
- Web-scrapping на демо-примере:
  - CSS Selector Reference
  - XPath Syntax

Блок 6: экспорт результатов

Пакеты `openxlsx`, `officer`, `DBI`, `jsonlite`, `readr`, `data.table`, `knitr`.

- Экспорт данных. Форматы rds, csv, json, xlsx, docx, pptx, odbc.
- Основы R Markdown и R Notebook. Экспорт отчетов путем knit -> LaTeX -> pdf
- Создание презентаций в формате html/pdf средствами R. Slide Presentations + `xaringan`
- Создание презентаций в формате PowerPoint средствами пакета `officer`.
- Прямой экспорт в Word средствами пакета `officer`.

Блок 7: основы программирования в R

Создание функций. Область видимости переменных. Просмотр объектов. Просмотр объектов, их структуры. Принципы работы «по ссылкам».

- Понятие функции. Создание собственных функций.
- Концепция окружения.
- Наконец-то, циклы. for loop, while loop
- `purrr` tutorial.
- Концепция "безопасных" вычислений, применение их в пакетном анализе (напр., safely).
- `Profvis` — Profiling tools for faster R code
- Подход lazy_evaluation, non-standard evaluation, пакет `lobstr`

Блок 8: первые шаги приложения

Направленность: Подходы к валидации промежуточных и финальных результатов. Shiny приложения, как целевого интерфейса для конечных потребителей. (Пакеты checkmate, reprex, futile.logger, shiny)

- Defensive programming. Валидация параметров функций. Пакет `checkmate` -- скорость и гибкость.
- Валидация входных данных:
  - `validate`: Data Validation Infrastructure;
  - `ruler`: Rule Your Data;
  - `janitor`: Simple Tools for Examining and Cleaning Dirty Data;
  - `tidyvrese` + `checkmate`
- Логирование средствами `futile.logger`.
- Знакомство с возможностями по созданию APM на базе Shiny на примерах Winners of the 1st Shiny Contest
- Принципы формирования воспроизводимых вычислений и культура оформления вопросов (пакет `reprex`).

Блок 9: даем нагрузку, выводим в продуктив

Методы и подходы в работе с данными «среднего» размера. Пакет `data.table`. Основные функции. Сравнительный экспериментальный анализ.

- Краткая вводная информация о деплойменте:
  - Rstudio Server. Полезные консоли Terminal и Jobs
  - работа в консоли Linux, базовы набор команд
  - работа с демонами в linux (перезапуск, статус)
  - контроль загрузки ресурсов в linux
  - технология docker

Блок 10 - ...

Здесь можно вписать дополнительные пакеты и подходы, которые крайне важны для решения задач в конкретной компании.

Как было указано в самом начале, базовая алгоритмика, статистика, линейная алгребра, AI/ML методы и пр. не входят в план быстрого старта:

- это ближе к предметной области, чем к базовым инструментальным знаниям;
- освоение этих вещей с 0 требует значительного времени; если это жизненно необходимо, то надо брать готовых специалистов с рынка;
- потребность в красивом математическом трюке появляется в рамках приложения (отнюдь не консольные вычисления) во второй половине проекта, когда решены все инфраструктурные вопросы (требования, железо, данные, ожидаемые результаты, архитектура).

В рамках Enterprise эти блоки расширяем по мере необходимости в парадигме смещения фокуса с увлекательного безответсвенного и бесконечного процесса по выбору вендорских решений (HP, SAS, SAP, ...) в сторону исполнения реальных насущных задач.

Быстро, дешево, на R open-source.

А теперь можно засучить рукава

Если остались вопросы

Slack ODS: `@ilya`, member_ID: `U7UTH8D5X`

habr: `https://habr.com/ru/users/i_shutov/`

T: `@iMissile`

skype: `ishutov`