‘ R’

2022-03-30

Contents

R, “ R
”. , R ,
. . . ,
.

, , . ,
R , R ,
, .
R, , .

R .
R: `for`, `while`, `repeate`.
, .
`apply()`. ,
`purrr`, `tidyverse`
`apply()`. ,
.
, `foreach`,
`pbapply` `furrr`. R,
`future`.

, -
R . “ R” ,
R. . . “ R Excel”,
.

7 2 55 , 7
30 . ,
. ,
. , , ,
, .
, ,
. .
.
, . .

! . ,
.
, .
, telegram :

- R ( )
- R

,
, .
, , YouTube,
, .
, “ R”,
selesnow@gmail.com. ,
.

, 2008 .
- Netpeak.
R : `rgoogleads`, `ryandexdirect`, `rfacebookstat`,
`timeperiodsR`, `rvkstat` . CRAN
150 000 .
“ R - ” Web Promo Experts.

Telegram YouTube R4marketing. .

, Netpeak Journal.

, , Go-Analytics, Analyze, eCommerce, 8P .

2016 R . .

R, , , , . :

- Telegram R4marketing
- Youtube R4marketing

1. for, while repeat
2. : try() tryCatch()
3. apply
4. purrr
5. : safely(), possibly(), quietly()
6. , : foreach, doFuture, pbapply, furrr
7. future

, , .

:

Chapter 1

for, while repeat

1.1

[1] — ,
.

. , 5
, . . .
, , .
:

- ;
- R;
- R;
- ;
- next break.

, .

1.2

1.3

00:00 . 00:28 00:58 R. 01:44 for.
02:40 for. 03:34 next.

[1] : https://ru.wikipedia.org/wiki/ _( )

04:50 for. 06:52 for (data.frame). 09:38 for. 10:55 . csv data.frame. 14:11 while. 15:25 break. 17:12 repeat. 18:30 while repeat R. 19:47 R . 21:17 .

1.4

```
#                  R


# for ---------------------------------------------------------------------
##                ,
##
##

##
week <- c('Sunday',
          'Monday',
          'Tuesday',
          'Wednesday',
          'Thursday',
          'Friday',
          'Saturday')

for ( day in week ) {

  print(n)
  Sys.sleep(0.25)

}

##
persons <- list(
  list(name = "Alexey", age = 36),
  list(name = "Justin", age = 27),
  list(name = "Piter",  age = 22),
  list(name = "Sergey", age = 39))

##      next
for ( person in persons ) {

  if ( person$age < 30 ) next
  print( paste0( person$name, " is ", person$age, " years old") )

}

##
for ( col in mtcars ) {
  print(mean(col))
  names(col)
}

##
for ( row in 1:nrow(mtcars) ) {
  print(mtcars[row, c('cyl', 'gear')])
}

##          for
x <- 1:5
y <- letters[1:5]

for ( int in x ) {

  for ( let in y ) {

    print(paste0(int, ": ", let))

  }

}

##
setwd('docs')
files <- dir()
result <- list()

for ( file in files ) {

  temp_df <- read.csv(file)

  result <- append(result, list(temp_df))

}

#
result <- do.call('rbind', result)
```

```
# while --------------------------------------------------------------------
##                 ,
##
x <- 1

while ( x < 10 ) {

  print(x)
  x <- x + 1

}

#       break
x <- 1
while ( x < 20 ) {

  print(x)

  if ( x / 2 == 5 ) break

  x <- x + 1

}

# repeate ------------------------------------------------------------------
##                 ,
##             break
x <- 1

repeat {

  print(x)

  if (x / 2 == 5) break

  x <- x + 1
}
```

1.5

R from

1.6

Chapter 2

: try() tryCatch()

2.1

R, , , .

try() tryCatch() R.

2.2

2.3

1. try() (0:37)
2. try() for (2:54)
3. tryCatch() (7:16)
4. tryCatch() (12:32)
5. tryCatch() for (13:39)
6. finally tryCatch() (15:27)
7. (19:09)
8. tryCatch() lapply() (24:11)

2.4

```
#
setwd(r'(C:\Users\Alsey\Documents\try_catch_lesson)')

#         try
res <- try( 10 / 'u' )

#
class(res)

#
attr(res, 'condition')

#
values <- list(3, 6, 2, 'x', 7, 3, 't', 9)

for ( val in values ) {

  res <- try( val / 10, silent = TRUE )

  if ( class(res) == 'try-error' ) {

    print(attr(res, 'condition'))

  } else {

    print( paste0( val, " / 10 = ", res))

  }

}


#         tryCatch
##
###
div <- function(x, y) {

  if ( is.na(y) ) {

    warning("Y is NA")

  }
  return( x / y )

}

###
val <- "just text"

###
result <-
  tryCatch(
    expr = {

      y <- div(10, val)

    },
    error = function(err) {

      message(err$message)
      y <- 0

    },
    warning = function(war) {

      message(war$message)
      y <- 1

    })


###
if ( 'error' %in% class(result)  ) {

  message("Catch")

}

###
values <- list(1, 3, NA, 8, "text")

for ( val in values ) {

  temp <-
    tryCatch({
      div(10, val)
    },
    error = function(err) {

      print(err$message)

    })

  if ( 'error' %in% class(temp) ) next
}


#    finnaly
library(DBI)
library(RSQLite)

##
con <- dbConnect(SQLite(), 'my.db')
##
df <- data.frame(a = 1:5,
                 b = letters[1:5])

##
out <-
  tryCatch(
    {

      dbWriteTable(con,
                   'my_data',
                   df)

    },

    error = function(err) {
      print(err$message)
      return(err)
    },

    finally = {

      print("          ")
      dbDisconnect(con)
    }
  )

#
##
```

```
exception <- function(class, msg)
  {
    stop(errorCondition(msg, class = class))
  }

##
divideByX <- function(x){
  #
  if (length(x) != 1) {
    exception("NonScalar", "x is not length 1")
  } else if (is.na(x)) {
    exception("IsNA", "x is NA")
  } else if (x == 0) {
    exception("DivByZero", "divide by zero")
  }
  #
  10 / x
}

##
val <- 0

tryCatch(
  {
    divideByX(val)
  },
  IsNA = function(x) {
    print("Catch")
  },
  NonScalar = function(x) {
    print("Catch2")
  },
  DivByZero = function(x) {
    print('Catch3')
  }
)

#
lapply(list(NA, 3:5, 0, 4, 7),
       function(x) tryCatch({

           divideByX(x)

       },
       IsNA=function(err) {
            warning(err)  # signal a warning, return NA
            NA
       },
       NonScalar=function(err) {
            message(err)     # fail
       },
       DivByZero=function(err) {
            message(err)
       })
)
```

2.5

Chapter 3

apply

3.1

R, ,
R. , ? .
R, `apply()`.
- , .

3.2

3.3

00:00 . 00:48 `apply`. 02:22 `apply()`. 07:57
`apply()` . 09:05 `lapply()`,
`sapply()` `vapply()`. 12:09 `apply`.
13:23 csv `lapply()`. 15:40 `mapply()`.
18:00 .

3.4

```
# apply family

#           ----------------------------------------------------------
#
```

```
for ( x in seq_along(1:nrow(mtcars)) ) {
  cat(rownames(mtcars[x,]), ":", sum(mtcars[x,]), "\n")
}

#
col_num <- 1

for ( x in mtcars ) {
  cat(names(mtcars)[col_num], ":", sum(x), "\n")
  col_num <- col_num + 1
}

# apply ------------------------------------------------------------------
# 1 -
# 2 -
apply(mtcars, 1, sum)
apply(mtcars, 2, sum)

sum(mtcars[3, ])
sum(mtcars[ ,3])
# row operation ----------------------------------------------------------
rowSums(mtcars)
rowMeans(mtcars)
#                     -------------------------------------
apply(mtcars, 2, quantile, probs = 0.25)
quantile(mtcars[, 3], probs = 0.25)

# lapply -----------------------------------------------------------------
values <- list(
  x = c(4, 6, 1),
  y = c(5, 10, 1, 23, 4),
  z = c(2, 5, 6, 7)
)

lapply(values, sum)
sapply(values, sum)
vapply(values, sum, FUN.VALUE = 7)

# lapply              ------------------------------------------
fl <- function(x) {
  num_elements <- length(x)
  return(x[1] + x[num_elements])
}

lapply(values, fl)
```

```r
#               ----------------------------------------------------
directory <- 'C:/Users/Alsey/Documents/docs/'
files <- dir(path = directory, pattern = '\\.csv$')
all_data <- list()

#
for ( file in files ) {
  data <- read.csv(paste0(directory, file))
  all_data <- append(all_data, list(data))
}

dplyr::bind_rows(all_data)

# lapply
file_paths <- paste0(directory, files)
all_data <- lapply(file_paths, read.csv)
dplyr::bind_rows(all_data)


# mapply ------------------------------------------------------------------
mapply(rep, 1:4, times=4:1)
mapply(rep, times = 1:4, x = 4:1)
```

3.5

apply from

3.6

3.7

- “ R apply- ” ( ).

Chapter 4

purrr

4.1

apply for,
. , purrr.

:

- purrr apply.
- map, map2, pmap, invoke.
- purrr.

4.2

4.3

00:00 . 00:57 purrr. 02:15
purrr. 03:29 map. 04:26 purrr. 05:20
map(). 08:23 map() for. 08:56
map_dfr(), map_dfc(). 13:01 , map2
pmap. 15:01 purrr. 20:05 walk. 22:31 keep()
discard(). 26:27 invoke. 29:12 reduce()
accumulate(). 34:23 .

4.4

```
# install.packages('purrr')
library(purrr)
library(dplyr)

#      map_*-------------------------------------------------------------
##
v_sizes <- c(5, 12, 20, 30)
map(v_sizes, rnorm)

#
rnd_list <- map(v_sizes, runif, min = 10, max = 25)
#
map_dbl(rnd_list, mean)
#
for ( i in rnd_list ) cat(mean(i), " ")

#
products <- tibble(
  product_id = 1:10,
  name = c('Notebook',
           'Smarthphone',
           'Smart watch',
           'PC',
           'Playstation',
           'TV',
           'XBox',
           'Wifi router',
           'Air conditioning',
           'Tablet'),
  price = c(1000, 850, 380, 1500, 1000, 700, 870, 80, 500, 150)
)

managers <- c("Svetlana", "Andrey", "Ivan")
clients  <- paste0('client ', 1:30)

create_transaction <- function(
  transaction_id,
  products_number = 3,
  product_dict,
  counts = c(1, 3),
  dates = c(Sys.Date() - 30, Sys.Date()),
  managers,
  clients
) {

  transaction <- sample_n(product_dict, size = products_number, replace = F) %>%
                mutate(date = sample( seq(dates[1], dates[2], by = 'day'), size = 1 ),
                       manager  = sample(managers, 1),
                       clients  = sample(clients, 1),
                       count    = sample(seq(counts[1], counts[2]), products_number, replace =
                       sale_sum = price * count,
                       transaction_id)

  return(transaction)

}

#        5
map_dfr(1:5,
        create_transaction,
            products_number = sample(1:10, 1),
            product_dict = products,
            counts = c(1, 3),
            dates = c(Sys.Date() - 30, Sys.Date()),
            managers = managers,
            clients = clients,
        .id = 'transaction_id')

#      pmap_* ----------------------------------------------------------
#                                    map2_*
x <- list(1, 1, 1)
y <- list(10, 20, 30)

map2(x, y, ~ .x + .y)

#                                          pmap_*
params <- tibble(
  transaction_id  = 1:3,
  products_number = c(4, 2, 6),
  product_dict    = list(products, products, products),
  counts          = list(c(1, 3), c(7, 10), c(2, 7)),
  dates           = list(c(as.Date('2021-11-01'), as.Date('2021-11-04')),
                         c(as.Date('2021-11-05'), as.Date('2021-11-08')),
                         c(as.Date('2021-11-09'), as.Date('2021-11-14'))),
  managers        = list(managers, managers, managers),
  clients         = list(clients, clients, clients)
)
```

```
tranaction_df <- pmap_df(params, create_transaction)

#      walk ------------------------------------------------------------
#       7
transactions <- map(1:7,
                    create_transaction,
                    products_number = sample(1:10, 1),
                    product_dict = products,
                    counts = c(1, 3),
                    dates = c(Sys.Date() - 30, Sys.Date()),
                    managers = managers,
                    clients = clients)

file_names <- paste0('transaction_', 1:7, ".csv")

walk2(
  .x = transactions,
  .y = file_names,
  write.csv
)

#      keep   discard --------------------------------------------------
#
map_dbl(transactions, ~ sum(.x$sale_sum))
#                       3000
transactions %>%
  keep(~ sum(.x$sale_sum) >= 3000)
#                       4000
transactions %>%
  discard(~ sum(.x$sale_sum) >= 4000)

#                        keep   walk
transactions %>%
  keep(~ sum(.x$sale_sum) >= 3000) %>%
  walk2(
    .x = .,
    .y = paste0('transaction_3k_', seq_along(.), ".csv"),
    write.csv
  )


#                        invoke ----------------------------
fun <- c('mean', 'sum', 'length')
params <- list(
  list(x   = tranaction_df$sale_sum),
  list(... = tranaction_df$sale_sum),
  list(x   = tranaction_df$sale_sum)
)

invoke_map_dbl(fun, params)


df <- tibble::tibble(
  f = c("runif", "rpois", "rnorm"),
  params = list(
    list(n = 10),
    list(n = 5, lambda = 10),
    list(n = 10, mean = -3, sd = 10)
  )
)

df

invoke_map(df$f, df$params)


#      reduce   accumulate ---------------------------------------------
#
#
managers_dict <- tibble(
  manager = managers,
  department = c('Sale', 'Sale', 'Marketing'),
  salary_percent = c(0.1, 0.12, 0.2)
)

clients_dict <- tibble(
  clients = clients,
  discount = runif(length(clients), min = 0, max = 0.4)
)

data_model <- list(tranaction_df, managers_dict, clients_dict)

reduce(transaction_data, left_join) %>%
  mutate(manager_bonus = sale_sum * salary_percent,
         total_sum = sale_sum - (sale_sum * discount),
         cumulate_minuses = accumulate(sale_sum - total_sum + manager_bonus, sum))

#              dplyr
tranaction_df %>%
  left_join(managers_dict) %>%
  left_join(clients_dict) %>%
  mutate(manager_bonus = sale_sum * salary_percent,
         total_sum = sale_sum - (sale_sum * discount),
         cumulate_minuses = cumsum(sale_sum - total_sum + manager_bonus))
```

4.5

purrr from

4.6

4.7

- 17 “ R ”.

Chapter 5

: safely(), possibly(), quietly()

5.1

R.
`retry`, `purrr`,
.

5.2

5.3

: 1. retry (0:36) 2. purrr (5:58) 3. safely() (8:05) 4. possibly() (9:40) 5. quietly() (10:53) 6. (12:50)

5.4

```
library(retry)

#
fun <- function(p = 0) {
  x <- runif(1)

  if (runif(1) < 0.9) {

    print(paste0('X = ', x, ' is Error!'))

    Sys.sleep(p)

    stop("random error")
  }
  "Excellent"
}

#
retry(fun(), when = "random error")

#
retry(fun(), when = "random error", interval = 2)

#
retry(fun(), when = "random error", max_tries = 3)

#
retry(fun(4), when = "random error", timeout = 2)

#
# val
# cnd              val
retry(fun(), until = function(val, cnd) val == "Excellent")

library(purrr)

#
div <- function(x, y) {

  if ( is.na(x) ) warning("X is NA")
  return(x / y)

}

#                lapply
val <- list(1, 6, 3, NA, 'k', 3)
#
lapply(val, div, y = 2)
```

```
# ######### #
# safely    #
# ######### #
#
res <- lapply(val, safely(div), y = 2)

#
res <- res %>% transpose()

res$result #
res$error  #

# ######### #
# possibly  #
# ######### #
#
res <- lapply(val, possibly(div, 0), y = 2)

# ######### #
# quietly   #
# ######### #
#                   ,
val <- list(1, 6, 3, NA, 3)
res <- map(val, quietly(div), y = 2) %>% str
```

5.5

Chapter 6

R

6.1

, 8 . ,
8 , ,
. 4 ,
2 , 4 .

, .
, . . . , .
, 4 , . . ,
.

, ,
, , .

6.2

6.3

00:00 . 00:51 . 02:20 .
03:25 `foreach` . 07:42 `foreach`. 10:05
`foreach`. 11:05 `foreach`
. 12:41 . 13:52 ID .
14:56 `foreach`. 15:38 `%dorng%`. 18:10
`apply`. 20:52 `parallel` `pbapply`. 21:54 `furrr`. 23:10
`purrr` `furrr`. 23:50 .

6.4

```
#            -----------------------------------------------------
# install.packages("doSNOW")
# library(doSNOW)
# library(doParallel)
library(doFuture)

#
pause <- function(min = 1, max = 3) {
  ptime <- runif(1, min, max)

  Sys.sleep(ptime)

  out <- list(
    pid = Sys.getpid(),
    pause_sec = ptime
  )
}

test <- pause()

#       foreach
#
system.time (
  {test2 <- foreach(min = 1:3, max = 2:4) %do% pause(min, max)}
)

#
sum(sapply(test2, '[[', i = 'pause_sec'))

#
test3 <- foreach(min = 1:3, max = 2:4, .combine = dplyr::bind_rows) %do% pause(min, ma

#
#
#cl <- makeCluster(4)
#registerDoSNOW(cl)

options(future.rng.onMisuse = "ignore")
registerDoFuture()
plan('multisession', workers = 3)

#
```

```
system.time (
  {
    par_test1 <-
      foreach(min = 1:3, max = 2:4, .combine = dplyr::bind_rows) %dopar% {
      pause(min, max)
    }
  }
)

#
plan('sequential')

par_test1


#         apply -------------------

library(pbapply)
library(parallel)

#
cl <- makeCluster(3)

#   pbapply
par_test2 <- pblapply(rep(1, 3), FUN = pause, max = 3, cl = cl)
#   parallel
par_test3 <- parLapply(rep(1, 3), fun = pause, max = 3, cl = cl)

#
stopCluster(cl)

#    purrr ---------------------------
library(furrr)

plan('multisession', workers = 3)

par_test4 <- future_map2(1:3, 2:4, pause)

#
plan('sequential')
```

6.5

R from

6.6

6.7

- “ API R , API . (1)”

Chapter 7

future

7.1

R, future.

7.2

7.3

00:00 . 01:15 . 04:33 . 05:40
. 06:42 future. 08:20
future. 10:42 cluster. 12:09 . 18:00
. 19:03 future. 21:58
future . 26:07 future. 28:00
futureverse. 29:10 .

7.4

```
library(future)
library(dplyr)

#                                -------------------------------------
#
```

```
v <- {
  cat("Hello world!\n")
  3.14
}

#
v %<-% {
  cat("Hello world!\n")
  3.14
}

#
f <- future({
  cat("Hello world!\n")
  3.14
})
v <- value(f)
resolved(f)
#                                   -----------------
a <- 1

x %<-% {
  a <- 2
  2 * a
}

x

a

#                        ---------------------------------------
##
plan(sequential)
pid <- Sys.getpid()
pid

a %<-% {
  pid <- Sys.getpid()
  cat("Future 'a' ...\n")
  3.14
  }
b %<-% {
  cat("Future 'b' ...\n")
  Sys.getpid()
  }
```

```
c %<-% {
  cat("Future 'c' ...\n")
  2 * a
  }

b
c
a
pid

##
###                       R
plan(multisession)
pid <- Sys.getpid()
pid

a %<-% {
  pid <- Sys.getpid()
  cat("Future 'a' ...\n")
  cat('pid: ', pid)
  3.14
  }
b %<-% {
  cat("Future 'b' ...\n")
  Sys.getpid()
  }
c %<-% {
  cat("Future 'c' ...\n")
  2 * a
}

b

c

a

pid

plan(sequential)

#
availableCores()

###
```

```
library(parallel)
cl <- parallel::makeCluster(3)
plan(cluster, workers = cl)

pid <- Sys.getpid()
pid

a %<-% {
  pid <- Sys.getpid()
  cat("Future 'a' ...\n")
  cat('pid: ', pid)
  3.14
}
b %<-% {
  cat("Future 'b' ...\n")
  Sys.getpid()
}
c %<-% {
  cat("Future 'c' ...\n")
  2 * a
}

b

c

a

pid

parallel::stopCluster(cl)


#            ------------------------------------------------------
plan(list(multisession, sequential))
# plan(list(sequential, multisession))

#
# plan(list(tweak(multisession, workers = 2), tweak(multisession, workers = 2)))
pid <- Sys.getpid()
a %<-% {
  cat("Future 'a' ...\n")
  Sys.getpid()
  }
b %<-% {
  cat("Future 'b' ...\n")
  b1 %<-% {
    cat("Future 'b1' ...\n")
    Sys.getpid()
    }
  b2 %<-% {
    cat("Future 'b2' ...\n")
    Sys.getpid()
    }
  c(b.pid = Sys.getpid(), b1.pid = b1, b2.pid = b2)
  }

pid

a
b

plan(sequential)


#                   --------------------------------------------
plan(sequential)
b <- "hello"
a %<-% {
  cat("Future 'a' ...\n")
  log(b)
  } %lazy% TRUE

a

#
backtrace(a)

#                              ------------------------
#
manual_pause <- function(x) {
  Sys.sleep(x)
  out <- list(pid = Sys.getpid(), pause = x)
  return(out)
}

#
pauses <- c(0.5, 2, 3, 2.5)

#
```

```r
manual_pause(2)

#
plan("multisession", workers = 4)
#
futs <- lapply(pauses, function(i) future({ manual_pause(i) }))
#
sapply(futs, resolved)
#
res <- lapply(futs, value)

dplyr::bind_rows(res)


#       future        promises ----------------------------------
library(cli)
options(cli.progress_show_after = 0,
        cli.spinner = "dots")

#
pauses.1 <- sample(1:5, 4, replace = T)
pauses.2 <- sample(2:3, 4, replace = T)
pauses.3 <- sample(3:6, 4, replace = T)

#
plan(list(
  tweak(multisession, workers = 3),
  tweak(multisession, workers = 4)
  )
)

val1 <- future(
  {
    futs <- lapply(pauses.1, function(i) future({ manual_pause(i) }))
    res  <- lapply(futs, value)
    res  <- dplyr::bind_rows(res)
  }
)

val2 <- future(
  {
    futs <- lapply(pauses.2, function(i) future({ manual_pause(i) }))
    res  <- lapply(futs, value)
    res  <- dplyr::bind_rows(res)
  }
)

val3 <- future(
  {
    futs <- lapply(pauses.3, function(i) future({ manual_pause(i) }))
    res  <- lapply(futs, value)
    res  <- dplyr::bind_rows(res)
  }
)

#
cli_progress_bar("Waiting")
while ( ! (resolved(val1) | resolved(val2) | resolved(val3)) ) {
  cli_progress_update()
}

cli_progress_update(force = TRUE)

# result table
lapply(list(val1, val2, val3), value) %>%
  bind_rows() %>%
  mutate(main_pid = Sys.getpid()) %>%
  print() %>%
  pull(pause) %>%
  sum()  %>%
  cat("\n", "Sum of all pauses: ", ., "\n")

plan(sequential)
```

7.5

future from

7.6

7.7

- “ API R , API . (2)”.

“ R”!

, 42.
:

- 24 : , “ R Excel”, “ R”.
  2.
- 24 - 29 : , . .
  . **3**.
- 30 - 34 : , , ,
  . , .
  4.
- 35 - 42 : , ,
  . **5**.

,
R.

, ,
, :

Telegram YouTube.
R.

!

- email: selesnow@gmail.com
- telegram : R4marketing
- youtube : R4marketing
- telegram: AlexeySeleznev
- facebook: selesnow
- github: selesnow
- linkedin: selesnow
- : alexeyseleznev.wordpress.com