Основы программирования в R

Списки в R

Алла Тамбовцева, НИУ ВШЭ

Содержание

Создание списков

Если сравнивать списки (*lists*) в R со структурами данных в Python, списки можно сравнить со словарями: они могут обладать вложенной структурой и хранить в себе объекты разных типов. В то же время списки в Python и списки в R похожи. Списки в R – это вложенные списки в Python.

Посмотрим на пример списка, который содержит в себе два числовых вектора:

```
L <- list(c(1, 2, 3, 4), c(5, 6, 7))
L
```

```
## [[1]]
## [1] 1 2 3 4
##
## [[2]]
## [1] 5 6 7
```

А теперь на пример списка с векторами разных типов:

```
grades <- list(c("Ann", "Sam", "Tom"), c(8, 7, 5))
grades
```

```
## [[1]]
## [1] "Ann" "Sam" "Tom"
##
## [[2]]
## [1] 8 7 5
```

Что интересно, в список можно объединять не только вектора, но и структуры разных типов. Например, мы можем создать список с описанием датафрейма (строки, тип *character*) и самим датафреймом (тип *data.frame*). Для начала посмотрим на встроенный в R датафрейм `mtcars`:

```
mtcars
```

```
##                    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
```

```
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

А теперь создадим список `dat`, который содержит название датафрейма, его краткое описание и сам датафрейм:

```
dat <- list("mtcars", "from R", mtcars)
dat
```

```
## [[1]]
## [1] "mtcars"
##
## [[2]]
## [1] "from R"
##
## [[3]]
##                    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C         17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
```

```
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

Получилось!

Так как в списках может храниться большое число разных векторов, для удобства им можно давать названия. Список `grades` можно было записать и так:

```
grades <- list(names = c("Ann", "Sam", "Tom"), marks = c(8, 7, 5))
grades
```

```
## $names
## [1] "Ann" "Sam" "Tom"
##
## $marks
## [1] 8 7 5
```

Работа с элементами списка

Если у элементов списка нет названий, обращаться к ним можно по индексу, указав его в двойных квадратных скобках:

```
L[[1]]
```

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

Если мы напишем индекс в одинарных квадратных скобках, визуально мы получим почти такой же результат:

```
L[1]
```

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

Но разница есть. В первом случае (двойные квадратные скобки) мы получаем сразу вектор, а во втором (одинарные квадратные скобки) — маленький список, внутри которого вектор. Сравним типы результатов:

```
class(L[[1]])
```

```
## [1] "numeric"
```

```
class(L[1])
```

```
## [1] "list"
```

Если у элементов списка есть названия, их можно вызывать более удобным образом, с помощью $:

```
grades$names
```

```
## [1] "Ann" "Sam" "Tom"
```

```
grades$marks
```

```
## [1] 8 7 5
```

Если нужно обратиться к «элементу элемента» списка, нужно сначала указать номер вектора, в котором находится элемент, а затем номер самого элемента в этом векторе. Выберем второй элемент в первом векторе списка L:

```
L[[1]][2]
```

```
## [1] 2
```

Можно заметить, что список похож на матрицу: для того, чтобы обратиться к элементу, нужно указать «строку» (вектор) и «столбец» (положение в векторе).

А теперь выберем третий элемент вектора names:

```
grades$names[3]
```

```
## [1] "Tom"
```

Для того, чтобы добавить элемент в список, нужно чётко понимать положение элемента в этом списке: будет ли это элементом самого списка или «элементом элемента. Добавим в список L третий элемент:

```
L[[3]] <- c(0, 9)
L
```

```
## [[1]]
## [1] 1 2 3 4
##
## [[2]]
## [1] 5 6 7
##
## [[3]]
## [1] 0 9
```

А теперь аналогичным образом изменим первый элемент второго вектора:

```
L[[2]][1] <- 50
L
```

```
## [[1]]
## [1] 1 2 3 4
##
## [[2]]
## [1] 50  6  7
##
## [[3]]
```

```
## [1] 0 9
```

Запросим структуру списков с помощью функцию `str()`:

```
str(grades)
```

```
## List of 2
##  $ names: chr [1:3] "Ann" "Sam" "Tom"
##  $ marks: num [1:3] 8 7 5
```

```
str(L)
```

```
## List of 3
##  $ : num [1:4] 1 2 3 4
##  $ : num [1:3] 50 6 7
##  $ : num [1:2] 0 9
```

Списки в статистике

Со списками легко можно встретиться, создавая статистические модели. Многие статистические функции выдают результат в виде списков. Когда результаты выводятся на экран, это не всегда заметно, но если мы захотим заглянуть внутрь, то увидим, что та же регрессионная выдача представляет собой объект, похожий на список, из которого можно выбрать вектор коэффициентов (`coefficients`), вектор остатков (`residuals`), предсказанных значений (`fitted.values`) и так далее.

Создадим линейную модель, которая предсказывает число лошадиных сил в зависимости от числа цилиндров:

```
model <- lm(data = mtcars, hp ~ cyl)
model
```

```
##
## Call:
## lm(formula = hp ~ cyl, data = mtcars)
##
## Coefficients:
## (Intercept)          cyl
##      -51.05        31.96
```

Интерпретировать полученную модель мы пока не будем, а лучше посмотрим на её структуру:

```
str(model)
```

```
## List of 12
##  $ coefficients : Named num [1:2] -51.1 32
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "cyl"
##  $ residuals    : Named num [1:32] -30.7 -30.7 16.2 -30.7 -29.6 ...
##   ..- attr(*, "names")= chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
##  $ effects      : Named num [1:32] -829.8 317.8 26.4 -25.7 -29.8 ...
##   ..- attr(*, "names")= chr [1:32] "(Intercept)" "cyl" "" "" ...
##  $ rank         : int 2
##  $ fitted.values: Named num [1:32] 140.7 140.7 76.8 140.7 204.6 ...
##   ..- attr(*, "names")= chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
##  $ assign       : int [1:2] 0 1
##  $ qr           :List of 5
##   ..$ qr   : num [1:32, 1:2] -5.657 0.177 0.177 0.177 0.177 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
```

```
##   .. .. ..$ : chr [1:2] "(Intercept)" "cyl"
##   .. ..- attr(*, "assign")= int [1:2] 0 1
##   ..$ qraux: num [1:2] 1.18 1.02
##   ..$ pivot: int [1:2] 1 2
##   ..$ tol  : num 1e-07
##   ..$ rank : int 2
##   ..- attr(*, "class")= chr "qr"
##  $ df.residual  : int 30
##  $ xlevels      : Named list()
##  $ call         : language lm(formula = hp ~ cyl, data = mtcars)
##  $ terms        :Classes 'terms', 'formula'  language hp ~ cyl
##   .. ..- attr(*, "variables")= language list(hp, cyl)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "hp" "cyl"
##   .. .. .. ..$ : chr "cyl"
##   .. ..- attr(*, "term.labels")= chr "cyl"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(hp, cyl)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "hp" "cyl"
##  $ model        :'data.frame':   32 obs. of  2 variables:
##   ..$ hp : num [1:32] 110 110 93 110 175 105 245 62 95 123 ...
##   ..$ cyl: num [1:32] 6 6 4 6 8 6 8 4 4 6 ...
##   ..- attr(*, "terms")=Classes 'terms', 'formula'  language hp ~ cyl
##   .. .. ..- attr(*, "variables")= language list(hp, cyl)
##   .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:2] "hp" "cyl"
##   .. .. .. .. ..$ : chr "cyl"
##   .. .. ..- attr(*, "term.labels")= chr "cyl"
##   .. .. ..- attr(*, "order")= int 1
##   .. .. ..- attr(*, "intercept")= int 1
##   .. .. ..- attr(*, "response")= int 1
##   .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. ..- attr(*, "predvars")= language list(hp, cyl)
##   .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. .. ..- attr(*, "names")= chr [1:2] "hp" "cyl"
##  - attr(*, "class")= chr "lm"
```

Внутри модель, действительно, выглядит как список!

Преобразование списков

При необходимости структуру списка можно упростить — превратить его в вектор с помощью функции `unlist()`. Если в списке всего один вектор, он таковым и останется:

```
small <- list(c("a", "b", "c"))
small
```

```
## [[1]]
```

```
## [1] "a" "b" "c"
```

```
unlist(small)
```

```
## [1] "a" "b" "c"
```

А если в списке несколько векторов, тогда все склеится в один длинный вектор:

```
unlist(L)
```

```
## [1]  1  2  3  4 50  6  7  0  9
```