### Исследование методов прогнозирования с помощью среды R. Часть 2. Основы языка R.

##### Типы данных языка R

Все объекты данных (а, следовательно, и переменные) в R можно разделить на следующие классы (т.е. типы объектов):

° numeric – объекты, к которым относятся целочисленные (integer) и действительные числа (double);

° logical – логические объекты, которые принимают только два значения: FALSE (сокращенно F) и TRUE (T);

° character – символьные объекты (значения переменных задаются в двойных, либо одинарных кавычках).

В R можно создавать имена для различных объектов (функций или переменных) как на латинице, так и на кириллице, но следует учесть, что а (кириллица) и a (латиница) – это два разных объекта. Кроме того, среда R чувствительна к регистру, т.е. строчные и заглавные буквы в ней различаются. Имена переменных (идентификаторы) в R должны начинаться с буквы (или точки .) и состоять из букв, цифр, знаков точки и подчёркивания.

При помощи команды ? <имя> можно проверить, существует ли переменная или функция с указанными именем.

Проверка на принадлежность переменной к определенному классу проверяется функциями is.numeric(<имя\_объекта>), is.integer(<имя>), is.logical(<имя>), is.character(<имя>), а для преобразования объекта в другой тип можно использовать функции as.numeric(<имя>), as.integer(<имя>), as.logical(<имя>), as.character(<имя>).

В R существует ряд специальных объектов:

° Inf – положительная или отрицательная бесконечность (обычно результат деления вещественного числа на 0);

° NA – "отсутствующее значение" (Not Available);

° NaN – "не число" (Not a Number).

Проверить, относится ли переменная к какому-либо из этих специальных типов, можно, соответственно, функциями is.nite(<имя>), is.na(<имя>) и is.nan(<имя>).

##### Векторы и матрицы

В то время как векторы представляют одномерную последовательность данных определённых типов, во многих случаях удобнее оперировать многомерными данными. Например, матрицы представляют собой двумерные векторы. Для преобразования вектора в матрицу можно присвоить вектору атрибут размерности с помощью функции *dim(x).* Данные в матрице хранятся по столбцам.

> x = 1:16

> dim(x) = c(16)

> x

[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Другим способом создания матриц является использование функции *matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE)*. Параметр *data* принимает вектор, элементами которого будет проинициализирована созданная матрица. Параметры *nrow* и *ncol* определяют количество строк и столбцов соответственно, *byrow* определяет будет ли заполнена матрица элементами из *data* по строкам или по столбцам.

|  |
| --- |
| >x = 1:16  > y = matrix(data = x , nrow = 2, ncol = 8)  > x  [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  > y  [ ,1] [ ,2] [ ,3] [ ,4] [ ,5] [ ,6] [ ,7] [ ,8]  [1 ,] 1 3 5 7 9 11 13 15  [2 ,] 2 4 6 8 10 12 14 16  > dim(y)  [1] 2 8 |

Узнать количество строк и столбцов матрицы можно, получив весь вектор размерностей с помощью функции dim, либо используя функции *nrow(x)* и *ncol(x).*

Доступ к элементам матрицы *A* происходит по индексу. *A[i, j]* ссылается на элемент *i*-й строки и *j*-го столбца матрицы *A*. На месте индексов *i* и *j* могут стоять векторы. Векторы *i* и *j* могут быть логическими или символьными. В последнем случае строкам и столбцам матрицы должны быть присвоены имена функций *rownames(x)* и *colnames(x).* [1, c. 446]

*A[i, ]* эквивалентно *A[i, 1:ncol(A)]*, а *A[, j]* эквивалентно *A[1:nrow(A), j]*. Возможен доступ к элементам матрицы с помощью одного индекса. Например, *A[5]* означает 5-й элемент матрицы *A.*

|  |
| --- |
| > m = matrix(1:6 , nrow = 2, ncol = 3)  > m  [ ,1] [ ,2] [ ,3]  [1 ,] 1 3 5  [2 ,] 2 4 6  > m [1 , 3]  [1] 5  > A = matrix(c(23 , 31 , 58 , 16) , nrow = 2)  > rownames(A) <- c("petal", "sepal")  > colnames(A) <- c("length", "width")  > A  length width  petal 23 58  sepal 31 16  > A["petal", "length"]  [1] 23 |

Для того, чтобы объединить несколько матриц в одну, используются функции *rbind(...)* (объединяет матрицы друг с другом снизу) и *cbind(...)* (объединяет матрицы друг с другом справа).

|  |
| --- |
| > A = matrix(1:4 , nrow = 2, ncol = 2)  > B = matrix(5:8 , nrow = 2, ncol = 2)  > cbind(A , B)  [ ,1] [ ,2] [ ,3] [ ,4]  [1 ,] 1 3 5 7  [2 ,] 2 4 6 8  > rbind(A , B)  [ ,1] [ ,2]  [1 ,] 1 3  [2 ,] 2 4  [3 ,] 5 7  [4 ,] 6 8 |

Рассмотрим еще несколько функций, часто используемых при работе с матрицами:

* t(A). Транспонирует матрицу.
* diag(A). Возвращает вектор диагональных элементов матрицы, либо создает диагональную матрицу с указанными значениями диагонали.
* det(A). Вычисляет определитель матрицы.
* solve(A, b). Решает систему линейных уравнений Ax = b с квадратной и невырожденной матрицей A. При этом solve(A) может применяться для отыскания обратной матрицы.
* eigen(A). Вычисляет собственные числа и векторы. Возвращает список, содержащий собственные числа и матрицу собственных векторов.
* chol(A). Выполняет разложение Холецкого для симметричной положительно определенной матрицы.
* qr(A). Выполняет QR-разложение матрицы. Возвращает список, содержащий полученные в результате разложения матрицы.
* qr.solve(A, b, tol = 1e-7). Решает систему линейных уравнений через QR-разложение.

##### Массивы данных

Ресурсы среды R позволяют обрабатывать массивы данных любой размерности. Работа с массивами почти идентична работе с матрицами. Создать массив можно либо изменив размерность вектора или матрицы с помощью функции *dim(x),* либо используя функцию *array (data = NA, dim = length(data)).*

|  |
| --- |
| > x = 1:24  > dim(x) = c(2, 3, 4)  > x  , , 1  [ ,1] [ ,2] [ ,3]  [1 ,] 1 3 5  [2 ,] 2 4 6  , , 2  [ ,1] [ ,2] [ ,3]  [1 ,] 7 9 11  [2 ,] 8 10 12  , , 3  [ ,1] [ ,2] [ ,3]  [1 ,] 13 15 17  [2 ,] 14 16 18  , , 4  [ ,1] [ ,2] [ ,3]  [1 ,] 19 21 23  [2 ,] 20 22 24 |

Для проверки является ли некоторая переменная вектором (не атомарным, а общим), матрицей или массивом используются функции is.vector(x), is.matrix(x), is.array(x).

С помощью функции *colnames(x)* можно изменить имена столбцов фрейма, с помощью *rownames(x)* – имена строк. Для загрузки наборов данных из файла в структуру или таблицу может быть использована функция *read.table(file, header = FALSE, sep = "", ...),* в которой на вход необходимо задать путь к текстовому файлу с данными. Параметр *header* позволяет указать, следует ли интерпретировать первую строку файла, как имена столбцов таблицы.

##### Факторные данные

Факторы представляют собой структуру данных для хранения векторов категориальных данных (классов), т.е. величин, которые могут принимать значения из конечного и неупорядоченного множества.

Факторы создаются с помощью функции *factor(x = character(), levels, labels= levels).*

Предположим, что у нас имеется 3 класса *Yes, No, Perhaps*. И некоторая выборка из 6 объектов, каждый из которых принадлежит одному из этих классов.

> v = c("Yes", "No", "Yes", "Perhaps", "No", "Perhaps")

> f = factor(v)

> f

[1] Yes No Yes Perhaps No Perhaps

Levels : No Perhaps Yes

Для получения и изменения текстового вектора, содержащего имена уровней фактора, служит функция *levels(x).*

##### Структуры данных

Структуры данных (data frames) один из самых важных типов данных в R, позволяющий объединять данные разных типов вместе. Структура является специальной версией списка, где все элементы имеют одинаковую длину. Можно считать, что структура данных это двумерная таблица, в которой (в отличие от числовых матриц), разные столбцы могут содержать данные разных типов (но все данные в одном столбце имеют один тип).

Создать структуру данных можно с помощью функции *data.frame(...),* аргументами которой являются произвольное количество элементов (столбцов) фрейма. В качестве элементов структуры данных могут выступать векторы, факторы, матрицы, списки или другие структуры. При этом все векторы должны иметь одинаковую длину, а матрицы и фреймы одинаковое число строк. Функция *data.frame(...)* просто собирает все данные вместе. Символьные векторы конвертируются в факторы. Остальные данные собираются во фрейм такими, какие они есть, поэтому для правильной работы некоторых алгоритмов приходится конвертировать их самостоятельно.

|  |
| --- |
| > a = matrix(1:8 , nrow = 4, ncol = 2)  > b = c("a", "b", "c", "a")  > d = (1:4 %% 2 == 0)  > e = factor(c("soft", "hard", "soft", "medium"))  > f = data.frame(a , b , d , e ) |

С помощью функции *colnames(x)* можно изменить имена столбцов фрейма, с помощью *rownames(x)* – имена строк. Для загрузки наборов данных из файла в структуру или таблицу может быть использована функция *read.table(file, header = FALSE, sep = "", ...),* в которой на вход необходимо задать путь к текстовому файлу с данными. Параметр *header* позволяет указать, следует ли интерпретировать первую строку файла, как имена столбцов таблицы.

Последовательность команд для самостоятельного рассмотрения:

|  |
| --- |
| m<-50  u<-c(1:m)  ncol(u)  nrow(u)  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  u<-matrix(c(1:m),1,m)  ncol(u)  nrow(u)  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  k<-10  B<-matrix(rep(u,times=k),m,k)  ncol(B)  nrow(B)  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  B<-t(matrix(rep(u,times=k),m,k))  ncol(B)  nrow(B) |

##### Практическое задание

1) Создать квадратную матрицу размером 2х2, состоящую из элементов от 1 до 4, найти обратную матрицу, результат представить в отчете;

2) Создать квадратную диагональную матрицу, состоящую из единичных элементов. Найти обратную матрицу, результат представить в отчете.

3) Создать вектор, состоящий из чисел от 1 до 10 с шагом 1. Преобразовать его в вектор, в котором сначала идут четные, затем нечетные элементы в обратном порядке, т.е.

(1,2,3,4,5,6,7,8,9,10) – исходный вектор

(2,4,6,8,10,9,7,5,3,1) – вектор, который необходимо получить

Полученный скрипт представить в отчете.

3) Выполнить последовательность команд для самостоятельного рассмотрения, представленную выше. Описать, какие матрицы получаются в результате их выполнения

##### Содержание отчёта по практической работе

1. Фрагмент данных, использованных для выполнения работы.

2. Полученные результаты по каждому пункту (или «скриншоты»).

3. Выводы о проделанной работе.