Занятие первое. Hello World или знакомимся с R.

Предисловие.

Сразу уточню, что полезным будет найти какую-либо книгу по R и подглядывать в нее в случае сложностей. Не могу назвать хорошей книги по R, достаточно неплоха (по-крайней мере весьма объемна) книга "Modern Applied Statistics with S" авторов W.N. Venables and B.D. Ripley.

Существенным недостатком названия R является то, что при попытке найти в поисковой системе, скажем, "Как записать цикл в R" вы едва ли обнаружите нужный ответ на первых 100 страницах выдачи. Тем не менее язык крайне популярен и вы легко найдете сотни книг, сотни ответов на stackexchange, а также множество документов наподобие FAQ или manual.

Учитывая, что у R бесчисленное множество библиотек, очень полезным является сайт http://www.rdocumentation.org/, в котором можно искать нужные вам функции или изучать отдельные библиотеки.

Довольно качественный подбор базовых функций и их описаний: https://github.com/AndreyAkinshin/aakinshin.net/blob/master/ru/_posts/r/functions.md

Сайт разработчиков: http://cran.r-project.org/, там вы можете скачать дистрибутивы и разобраться как их установить.

Полезным также является использование оболочки RStudio, значительно улучшающей интерфейс и позволяющий использовать некоторые графические возможности.

Нам понадобятся базовые библиотеки: base, stats, graphics, datasets, дополнительные библиотеки будут упоминаться по мере продвижения по материалу.

F.A.Q. Как использовать R.

1. Как создать программу?

Программа в R называются скриптом, создать ее можно командой "создать скрипт". Сама рабочая среда R устроена как командная строка, но удобнее писать целые программы в файле скрипта, а потом запускать их в командную строку.

2. Как запустить программу?

Если вы написали что-то в скрипте, а затем щелкнули на консоль R, то в верхнем меню появится возможность запустить скрипт целиком или запустить выделенные строки.

3. Как пользоваться переменными?

Присваивание осуществляется операцией <-. Скажем, Temp<-5 положит в переменную Temp число 5. Объявления переменной перед этим не требуется. Впрочем, доступно и использование обычного оператора =.

4. Как вывести на экран информацию?

Информация выведется в консоль. Два наиболее простых варианта таковы:

а) Для вывода значения переменной Temp просто написать отдельной строкой Temp. R увидит, что вы не используете присваивания и распознает, что вам хочется увидеть содержимое переменной Temp. Аналогичным образом можно производить операции и

смотреть на результат, например 2*sin(Temp) + Temp1 подсчитает вам соответствующее выражение и выдаст в консоль.
б) Использовать print(). Если аргументом print будет переменная, например, print(Temp), то вы увидите ее значение, а если строка, то в консоли просто появится соответствующий текст, например, print("Temp") выдаст слово Temp.

5. Какие есть классы переменных?

Основные классы переменных в R — "character"(строка), "numeric"(действительное число), "integer"(целое число), "logical"(булева величина), "complex"(комплексная величина), "list"(список). При этом по умолчанию R сам выдает тип переменным при присваивании им какой-либо информации.

6. Как работать с векторами?

Простейший вектор задается командой двоеточие, скажем, 1:5 — это вектор из 5 элементов 1,2,3,4,5.
Можно создать вектор командой c(). Так c(5,2,3) создаст вектор с элементами 5, 2, 3, c(1:5,7) — вектор с числами от 1 до 5, а затем 7, а c(x,y) склеивает векторы x и y.
И наконец можно создать вектор командой vector() с двумя аргументами, первая отвечает за класс данных, а вторая за их число. То есть x<-vector("integer 5) сделает из x вектор с 5 числовыми компонентами (по умолчанию равными 0).
Удобно также пользовать командой seq, создающей арифметическую прогрессию. Основные ее аргументы from, to, by, задающие первый и последний член и шаг. Скажем seq(from = 3, to = 103, by= 2) выдаст все нечетные числа от 3 до 103, а seq(from = 2, to = 7, by = 2) выдаст 2,4,6.
Обращение к отдельным компонентам вектора ведется с помощью [], например Vec[2] обратится ко второму элементу вектора Vec.
Можно задавать вектор на основе любой заданной переменной, обращаясь к ней с помощью []. Например, если я напишу Vec<-5, а затем Vec[2]<-4, то получу вектор Vec, у которого первая координата 5, а вторая 4. Если написать Vec<-5, а затем Vec[3]<-5, не задавая Vec[2], то получите вектор длины 3, у которого вторая координата NA — не задана.

7. Как считать функции от векторов?

К вектору можно применять как векторные функции (например, sum), так и скалярную. Если вы примените к вектору числовую функцию (например, sin), то получите вектор из синусов координат вектора. Например, sin(c(1,1))+2 это вектор (2+sin(1), 2+sin(1)). В том числе x+y для векторов их покомпонентно сложит, x*y — покомпонентно умножит и так далее.
Скалярно умножить векторы можно с помощью операции $\%*\%$ или просто (sum(x*y)).

8. Хорошо, а что с матрицами?

Матрица создается из вектора с помощью команды matrix(). Например matrix(c(1,2,3,4),nrow = 2) создаст матрицу

$$\begin{matrix} 1 & 3 \\ 2 & 4 \end{matrix}$$

nrow здесь задает количество строк, которые я желаю получить (можно вместо nrow использовать ncol и задавать таким образом число столбцов). По умолчанию заполнение идет по столбцам, но это можно изменить, дописав byrow = TRUE. Так matrix(c(1,2,3,4), byrow = TRUE, ncol = 2) создаст матрицу

1 2
3 4

9. А как работать с матрицами?

Если Matr — матрица, то обращение, скажем, к элементу 1,2 осуществляется командой Matr[1,2]. Matr[2:5, 3:4] выделит подматрицу из соответствующих элементов, Matr[1:2,] — две первых строки. С матрицами можно делать то же, что с векторами, например, складывать, считать от них синус и так далее, при этом операции будут осуществляться поэлементно.

Если написать умножение матриц Matr1*Matr2, то это также будет поэлементное умножение. Для обычного матричного умножения нужно записать Matr1%*%Matr2.

Транспонировать матрицу можно с помощью t(), найти определитель — с помощью det().

Для решения линейного уравнения используется solve(matr,vec), для обращения матрицы solve(matr).

10. Как склеивать векторы или матрицы?

Векторы можно склеивать в вектор с помощью все той же команды c(). Склеивание матриц производится командами cbind() или rbind(). Первая склеивает матрицы по строкам, вторая по столбцам.

11. Может ли у матрицы столбцы быть разных типов?

Тип matrix так не умеет, зато умеет тип data.frame. Создать объект такого типа можно с помощью команды data.frame(), скажем DF<-data.frame(Vec1,Vec2,...). Полученный объект будет матрицей со столбцами Vec1, Vec2,... Обращаться к нему можно как напрямую, скажем, DF[1:2,2:3], а можно вызвать отдельный столбец нашей таблицы командой DF$Vec2.

Если запустить команду в форме DF<-data.frame(TheFirst=Vec1, TheSecond = Vec2), то у DF столбцы будут носить названия TheFirst, TheSecond и обращаться к ним уже можно будет через DF$TheFirst.

Можно перевести матрицу в data.frame с помощью as.data.frame.

Можно обращаться к заголовкам с помощью команды names(). Скажем, names(Matr)[5]<-"МатСтат" переименует 5 столбец в МатСтат.

12. А можно делать списки из объектов разного типа и разной структуры.

Это позволит делать тип list. Команда list() позволяет их создавать, например, ListOne<-list(Matr1,Matr2) создаст мне список из двух матриц. Обращаться к ним можно с помощью ListOne[[1]] и ListOne[[2]]. Скажем, левый верхний элемент первой матрицы задается как List1[[1]][1,1].

13. Как работать с файлами?

Наиболее удобными форматами являются .txt и .csv. Функции, используемые для чтения

этих форматов — read.table(file,header=TRUE,sep=" "), где file — название файла, header — переменная, отвечающая за то, читать ли заголовки из первой строки, sep — формат разделителя. Для csv файлов есть также функция read.csv, более удобная для этого формата. Обе функции выдают data.frame.
Перед этим удобно установить директорию с помощью setwd, скажем setwd("C:/Work/R/").

14. Как в R устроены циклы и условия?

Довольно похоже на другие языки: условие задается с помощью if, else

```
if ()
{
}
else,
{
}
```

цикл for задается в формате

```
for ()
{
}
```

где условие цикла имеет вид Var in Seq, Var — переменная цикла, а Seq — некоторая последовательность, элементы которой будет пробегать Var. Скажем, for (i in 1:100){print("Hello!") } напишет слово Hello! 100 раз.
Цикл while задается в формате

```
while()
{
}
```

15. А как задавать функции?

С помощью команды function. Скажем,

```
f<-function(x,y,z){
w<-x+y+z
return (w+3)
}
```

создаст функцию f, которая трем числам/векторам/матрицам будет сопоставлять их сумму плюс 3. При этом никаких указаний типов не требуется, функции сработает для любых x,y,z, для которых определены операции внутри. Для вызова функции достаточно просто задать f(x,y,z).
Обратите внимания, что после return обязательно писать ().

16. Как строить графики?

plot() строит графики функций или массивов данных. Например, plot(x,y), где x,y — пара векторов одной длины, построит облако точек (x[i],y[i]).
plot(sin(x),-1,1) построит график синуса на -1,1.
Обратите внимание, что если вы сами задали функцию, то plot(f) не сработает, нужно писать plot(Vectorize(f)).

hist() строит гистограмму.
У этих функций есть множество параметров, про которые вы можете прочитать самостоятельно.
Более изящное решение представляет пакет ggplot2, о котором более подробно рассказано в отдельном файле.

17. А как сделать несколько графиков на одном экране?

Во-первых, можно использовать функции lines() и points(), они добавляют точки или линии на имеющийся график.
Во-вторых, бывает удобно также использовать у par параметр mfrow, позволяющий вам разбить поле для рисования на несколько. Например, par(mfrow = c(3,5)) сделает из вашего окна для рисования таблицу 3 на 5, каждый следующий график будет отправляться в новую ячейку.
Отмечу также команду windows(), создающую новое окно для рисования.
Команда par(new=TRUE) позволит вам следующий plot создать на новом листе.

18. Как получить случайные величины?

sample(vector,size) создает выборку размера size из вектора vector без повторения. Дописав replace = TRUE получим выборку с повторением.
Выборки из равномерного, нормального, экспоненциального, биномиального распределений создают командами runif, rnorm, rexp, rbinom. Первый аргумент отвечает за размер выборки, последующие — за параметры распределения.
Более подробная информация здесь.

19. Что насчет описательной статистики?

Базовую информацию о векторе выдает summary(). Полезно также пользоваться функцией str, чтобы получить базовую информацию о . Функции mean(), sd(), median() считают среднее, стандартное отклонение и медиану распределения.

Упражнения

Обратите внимание, что в разделе материалов практикума есть несколько готовых скриптов с подробными комментариями, чтобы вам было проще разобраться со структурой языка.

Упражнение 1. Написать "Hello, World!"

Упражнение 2. Загрузить базу данных cars из библиотеки datasets. Какую структуру имеет эта база данных? Построить график зависимости переменных из базы. Выделить автомобили со скоростью менее 15. Какое среднее значение имеют переменные? Какое среднее и дисперсия у отношения переменных?

Упражнение 3. Создать функцию композиции двух перестановок длины 20. Проверить для двух случайных перестановок, являются ли они коммутирующими. Визуально убедиться, что ваш алгоритм создания случайной перестановки равновероятно выбирает возможные значения для первого элемента перестановки.

Упражнение 4. Сгенерировать нормальную выборку с параметрами 1 и 1 размера 500. Найти ее выборочные среднее и дисперсию. Найти интервал, в который должны

попасть 95% наблюдений и подсчитать наблюдения, которые в него не попали.