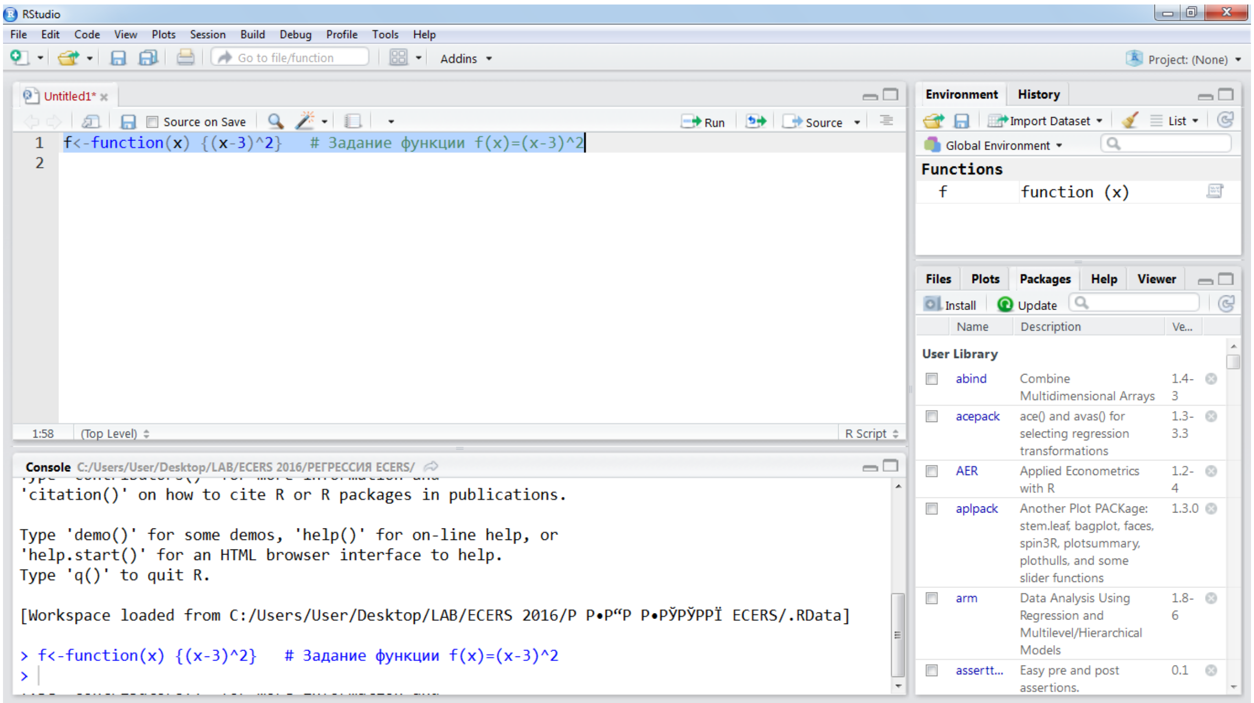
Графика в R

Задание математических функций и построение графиков

Запустим RStudio, создадим новый лист программы (например, по сочетанию Ctrl+Shift+N) и наберем или скопируем следующий код:

```
f <-function(x) {(x-3)^2}          # Задание функции f(x)=(x-3)^2
```

Что произойдет, если мы отправим этот код на компиляцию (Ctrl+Enter)?



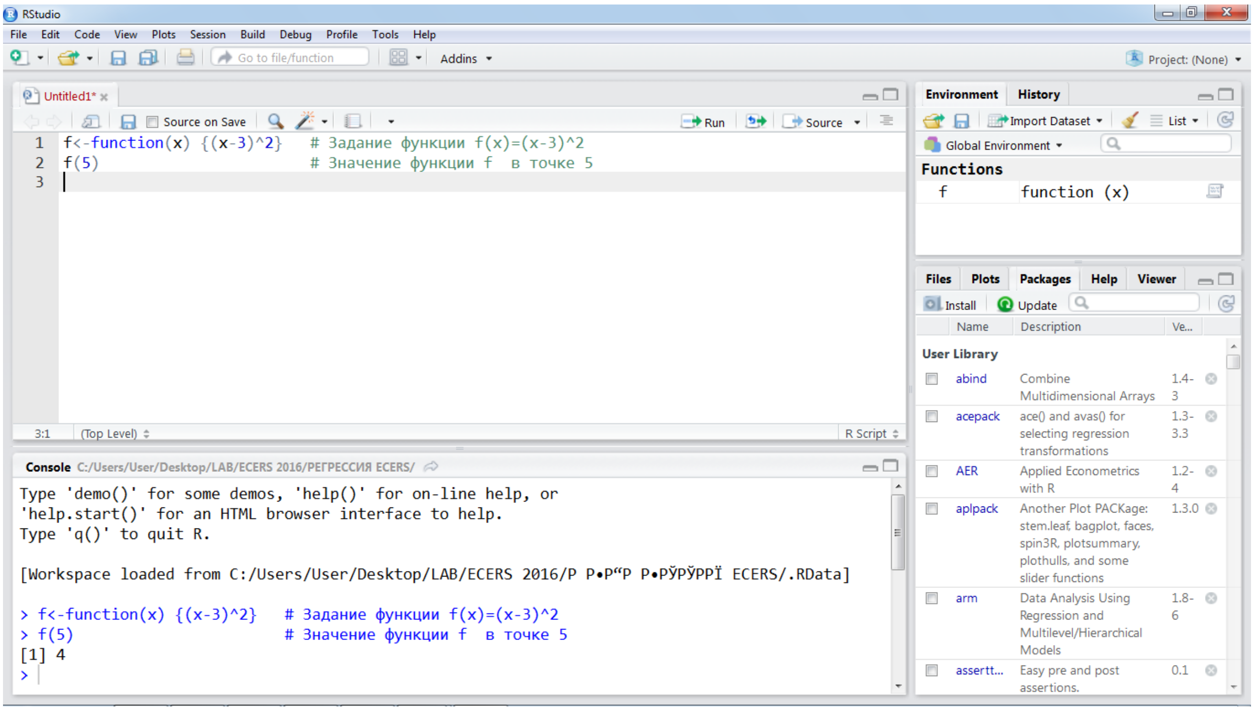
На первый взгляд R никак не отреагировал на введенный текст, однако в действительности, в памяти R создалась функция под именем f, значения которой, согласно нашему коду в фигурных скобках {(x-3)^2}, зависят от одной переменной следующим образом:

$$f(x) = (x-3)^2.$$

Обратите внимание на правое верхнее окно RStudio: в нем появилась только что созданная функция $f(x)$.

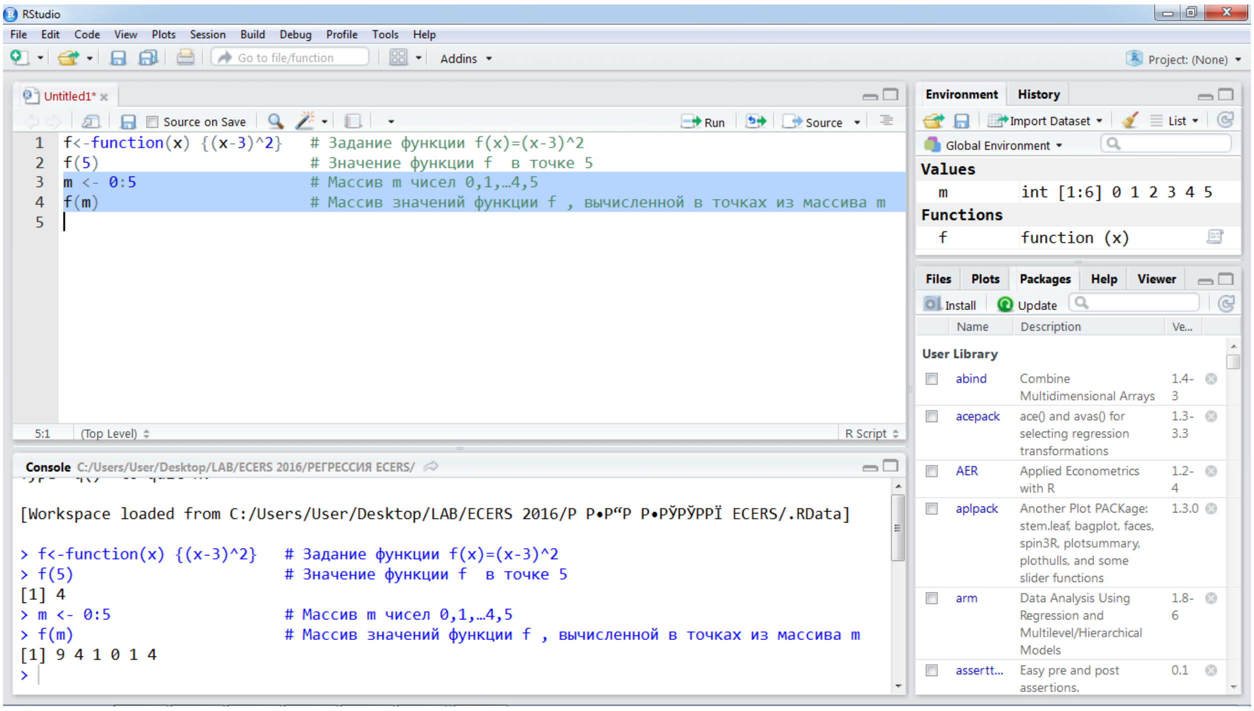
Это означает, что теперь R будет понимать обращение к этой функции для конкретных чисел. Например, вычислим значение этой функции в точке $x = 5$, набрав:

```
f(5)        # Значение функции f  в точке 5
```



В качестве аргумента введенной функции f можно подставлять и целые массивы, тогда функция будет вычислена в каждой точке массива:

```
m <- 0:5        # Массив m целых чисел 0,1,…4,5
f(m)            # Массив значений функции f , вычисленной в точках из массива m
```



Как видим, строка внизу консоли R перечисляет все значения функции. Таким образом, можно легко задавать/генерировать конечные последовательности значений функции для заданных последовательностях значений аргументов.

Задание 1. Объявить в R функцию

$$g(x,a,b) = \frac{x^2-a}{x-b},$$

где a и b – параметры (по умолчанию считать параметры равными единицам) и вычислить значения $g(2;1{,}1)$, $g(-3;5{,}1)$ и $g(6;9{,}3)$.

Решение. Наберем следующий код в окне программы:

```
g <-function(x , a = 1, b = 1) {(x^2-a)/(x-b)}            # Задание функции g(x,a,b)
```

Если теперь в следующей строчке набрать

```
g (2)                              # Вычисление функции  g(x=2,a=1,b=1)
```

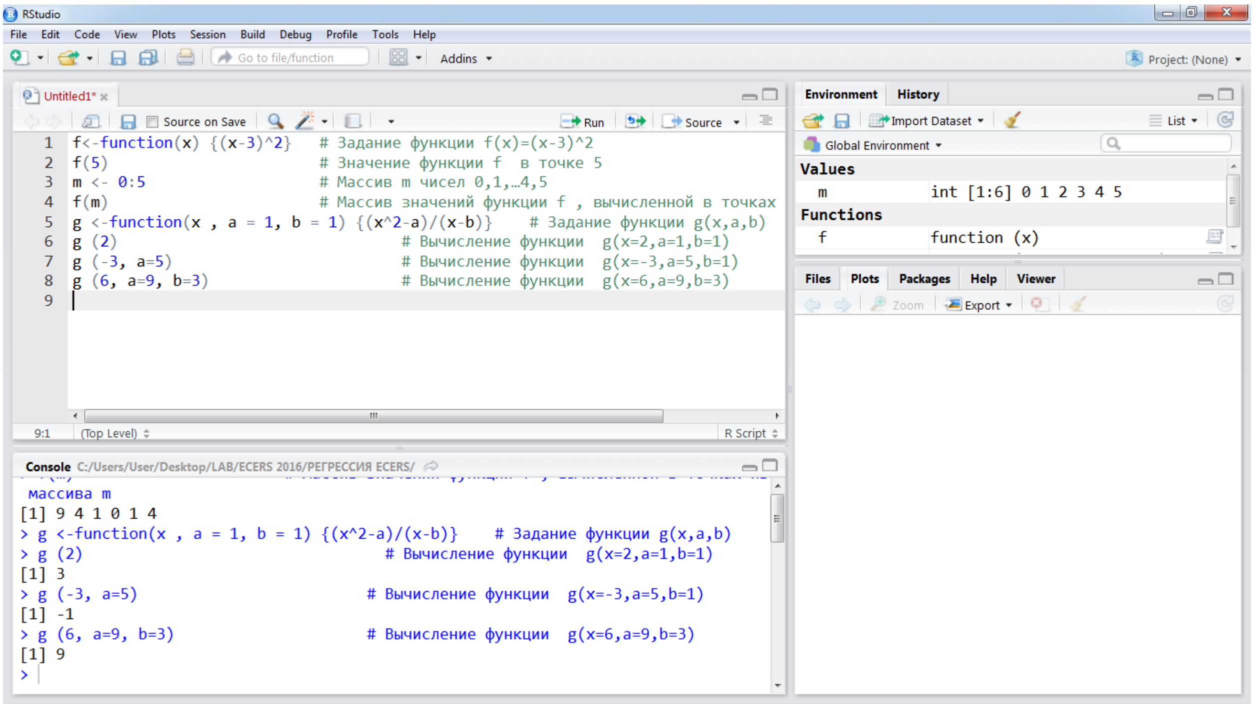
мы получим вычисленное значение функции $g(x)$ для $x = 2$, при этом опущенные нами параметры a и b будут приравнены к значениям по умолчанию, т.е. 1, объявленным ранее в аргументах function(x , a = 1, b = 1). Вызов функции в формате:

g (-3, a=5) # Вычисление функции g(x=-3,a=5,b=1)

даст нам значение функции g для $x=-3$. При этом параметр $a=5$, а опущенное значение параметра b будет приравнено к значению по умолчанию, т.е. 1.

Естественно, возможно и полное обращение к нашей функции, с указанием всех ее аргументов:

g (6, a=9, b=3) # Вычисление функции g(x=6,a=9,b=3)



Замечание 1. При вызове функции $g(x, a, b)$ мы опускали название переменной, но не было бы ошибкой записать $g(x=2)$. Вообще, в R можно опускать названия аргументов функции, если только помнить в каком порядке они перечисляются. Так, например, команда $g(2,5)$ будет эквивалента вызову функции $g(x=2, a=5, b=1)$.

Замечание 2. Попробуйте объяснить результат NaN («невычисляемо», «не число»), если ввести команду:

g (3, 9, 3) # Вычисление функции g(x=6,a=9,b=3)

Построение графиков функций

Используем возможности, описанные в предыдущем пункте, для построения графиков функций.

Задание 2. Построить график функции $f(x) = (x-3)^2$ на отрезке $[-5; 8]$.

Решение. Данную функцию мы уже ранее объявили в R. Зададим последовательность аргументов x, пробегающих отрезок $[-5; 8]$ с достаточно малым шагом, например, 0.05. Это можно сделать с помощью команды «seq»:

```
x <- seq(-5, 8, 0.05)       # Последовательность чисел от -5 до 8 с шагом 0.05, записанная в x
```

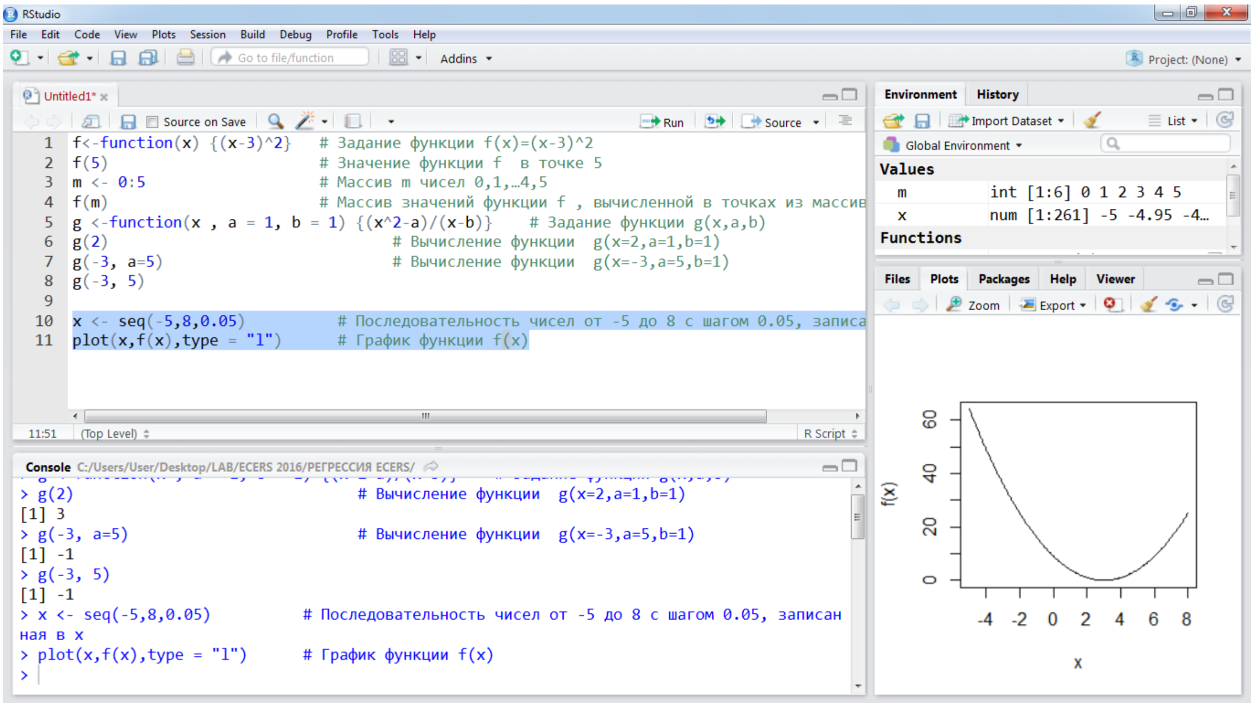
или в расширенном виде

```
x <- seq(from = -5, to = 8, by = 0.05) # То же самое
```

Теперь остается вызвать стандартную функцию построения графика:

```
plot(x,f(x), type = "l")    # График функции f(x)
```

Здесь в качестве значения параметра типа кривой «type» выбрано значение "l" от слова line – соединение точек прямыми линиями.



Функция seq имеет еще несколько полезных параметров. Например, часто бывает удобно задать последовательность от a до b не с помощью шага, а посредством указания общего количества точек в последовательности. Следующая ниже строчка кода задает последовательность из 5 чисел, эквидистантно пробегающим диапазон от 21 до 36:

```
x <- seq(from = 21, to = 36, length.out = 5) #  Последовательность из пяти чисел от 21 до 36
```

```
> x <- seq(from = 21, to = 36, length.out = 5) #  Последовательность из пяти
чисел от 21 до 36
> x
[1] 21.00 24.75 28.50 32.25 36.00
```

Задание 3. Построить графики функции $g(x,a,b) = \frac{x^2-a}{x-b}$ на отрезке $[-40; 40]$ при различных значениях параметров:

a) $a = 4,\ b = -5$;

b) $a = 4,\ b = 2$;

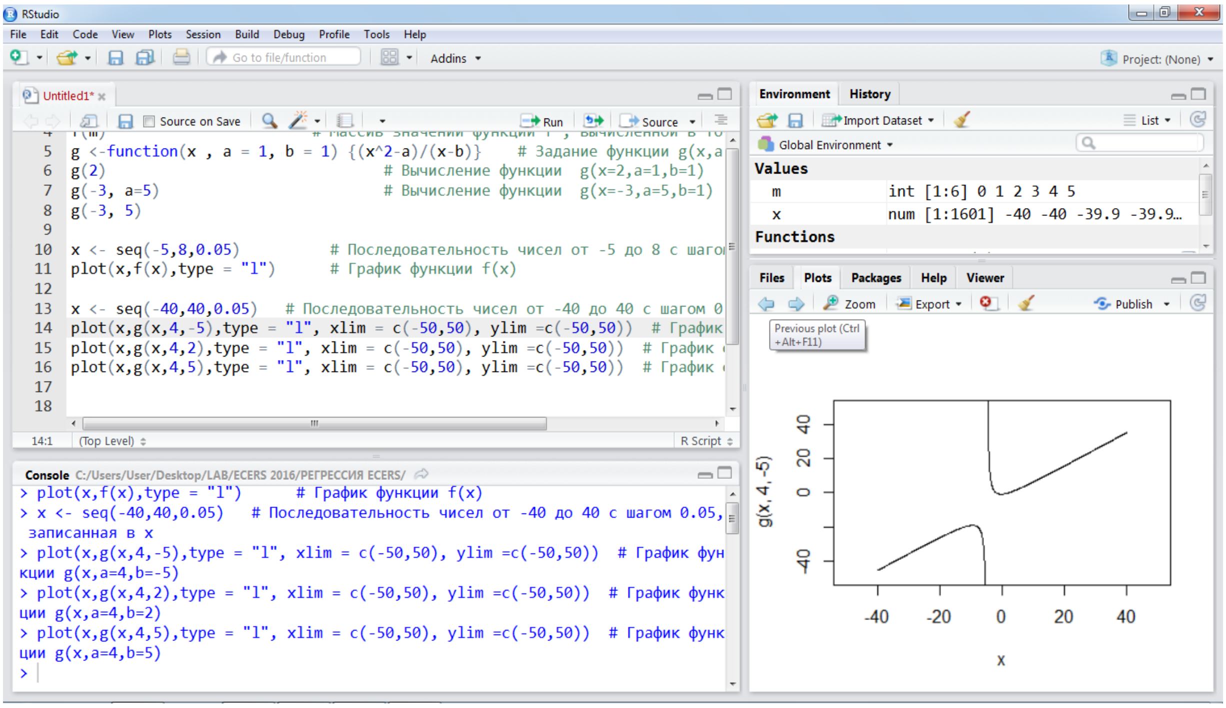
c) $a = 4,\ b = 5$.

Решение. Функция g нами была уже объявлена. Зададим, как и раньше, последовательность аргументов

```
x <- seq(-40, 40, 0.05) # Последовательность чисел от -40 до 40 с шагом 0.05, записанная в x
```

и поочередно вызовем построение графиков соответствующих функций:

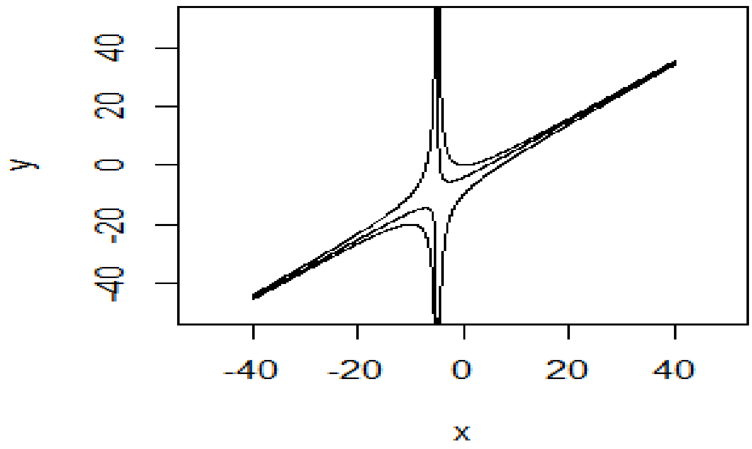
```
plot(x,g(x,4,-5),type = "l", xlim = c(-50,50), ylim =c(-50,50))  # График функции g(x,a=4,b=-5)
plot(x,g(x,4,2),type = "l", xlim = c(-50,50), ylim =c(-50,50))  # График функции g(x,a=4,b=2)
plot(x,g(x,4,5),type = "l", xlim = c(-50,50), ylim =c(-50,50))  # График функции g(x,a=4,b=5)
```



Для того, чтобы вернуться к предыдущему или последующему графику используйте в левой верхней части окна графика стрелки вперед и назад. Также, полезно нажать кнопку Zoom, находящуюся чуть правее, для увеличения картинки.

Кстати, часто бывает необходимым разместить несколько графиков на одном рисунке. Достигается это заменой последующих plot на lines. Например:

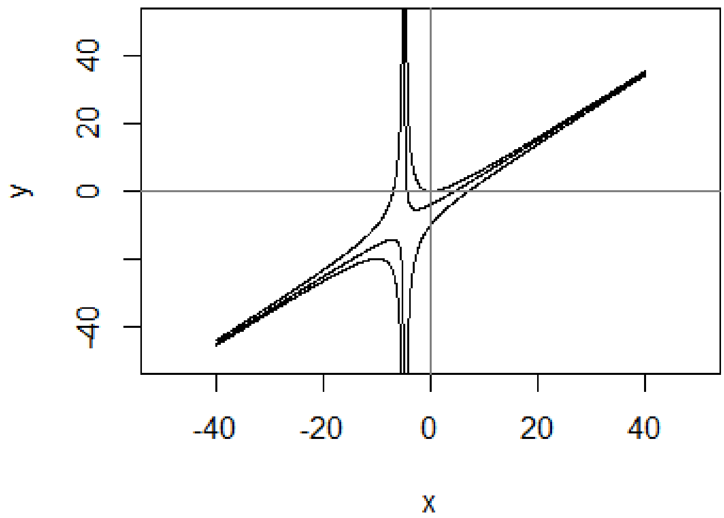
```
plot(x,g(x,0,-5),type = "l", xlim = c(-50,50), ylim =c(-50,50) ,ylab = "y")     # График g(x, 0, -5)
lines(x,g(x,20,-5),type = "l", xlim = c(-50,50), ylim =c(-50,50))  # Добавлен график g(x,20, -5)
lines(x,g(x,50,-5),type = "l", xlim = c(-50,50), ylim =c(-50,50))  # Добавлен график g(x,50, -5)
```



Как можно было заметить мы использовали дополнительные параметры в команде plot: xlim и ylim, устанавливающие границы изменения x и y на графике от -50 до 50.

При желании, можно разместить на рисунке привычные оси координат *ox* и *oy*, дополнительно используя следующую команду:

```
abline(h = 0, v = 0, col = "gray50")        # Нанесение на график линий ох и оу
```



Полный перечень аргументов данной команды plot и их точное описание доступно по команде вызова справки:

```
?plot        # Вызов справки по команде plot
```

или нажатием клавиши F1 при поставленном курсоре перед первой буквой интересующей нас команды. Попробуйте найти графический параметр *tck* для функции plot, который автоматически рисует координатную сетку? Если задать его значение *tck = 1*.

Задание произвольных пользовательских функций

Обобщим создание математических функций на случай нескольких значений самой функции.

Задание 4. Запрограммировать функцию Power(x), возвращающую одновременно квадрат и куб своего аргумента x.

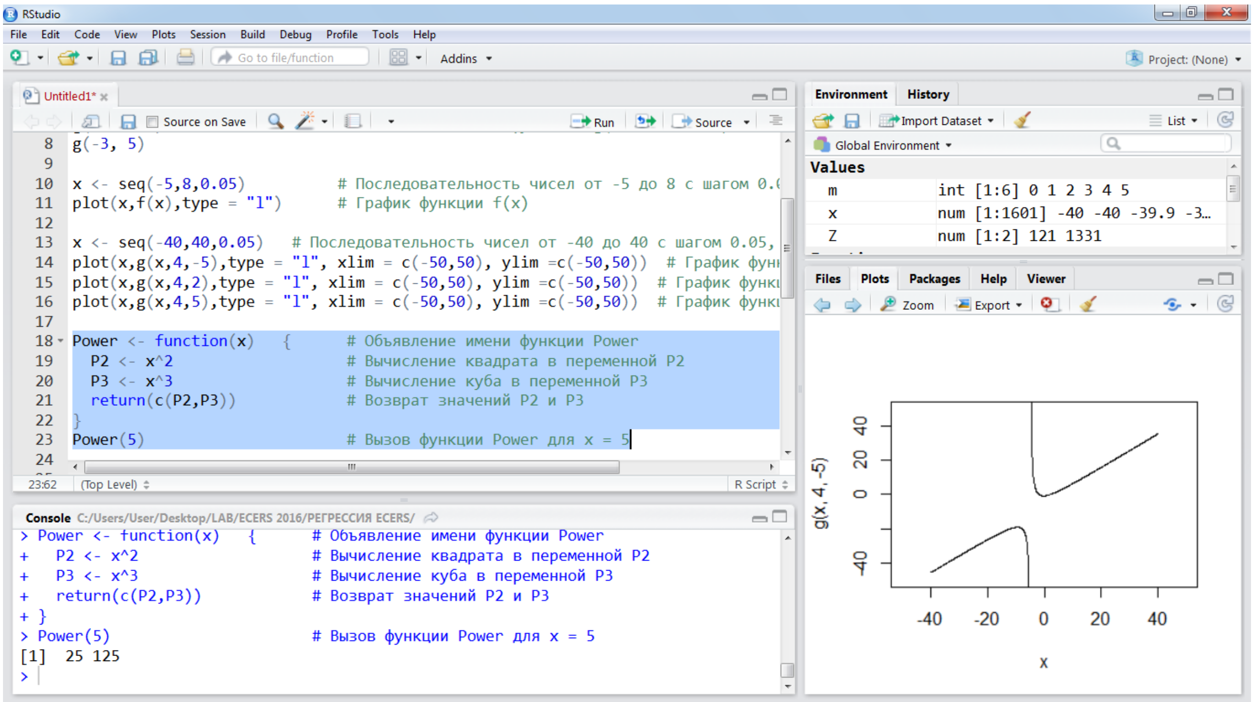
Решение. Введем на новых строчках следующий код:

```
Power <- function(x)   {                # Объявление имени функции Power
  P2 <- x^2                             # Вычисление квадрата в переменной P2
  P3 <- x^3                             # Вычисление куба в переменной P3
  return(c(P2, P3))                     # Возврат значений P2 и P3
}
Power(5)                                # Вызов функции Power для x = 5
```

Обратим внимание на то, что теперь фигурные скобки {…} после команды function(x) содержат не одно выражение, которое и считалось ранее по умолчанию значением нашей функции, а несколько команд. Во второй строчке мы вычисляем квадрат аргумента и записываем результат в переменную *P2*, а в третьей строке вычисляем куб и записываем в переменную *P3*. Следующей строкой мы должны указать компилятору R какое из вычисленных чисел мы хотим вернуть в качестве результата действия функции Power(x).

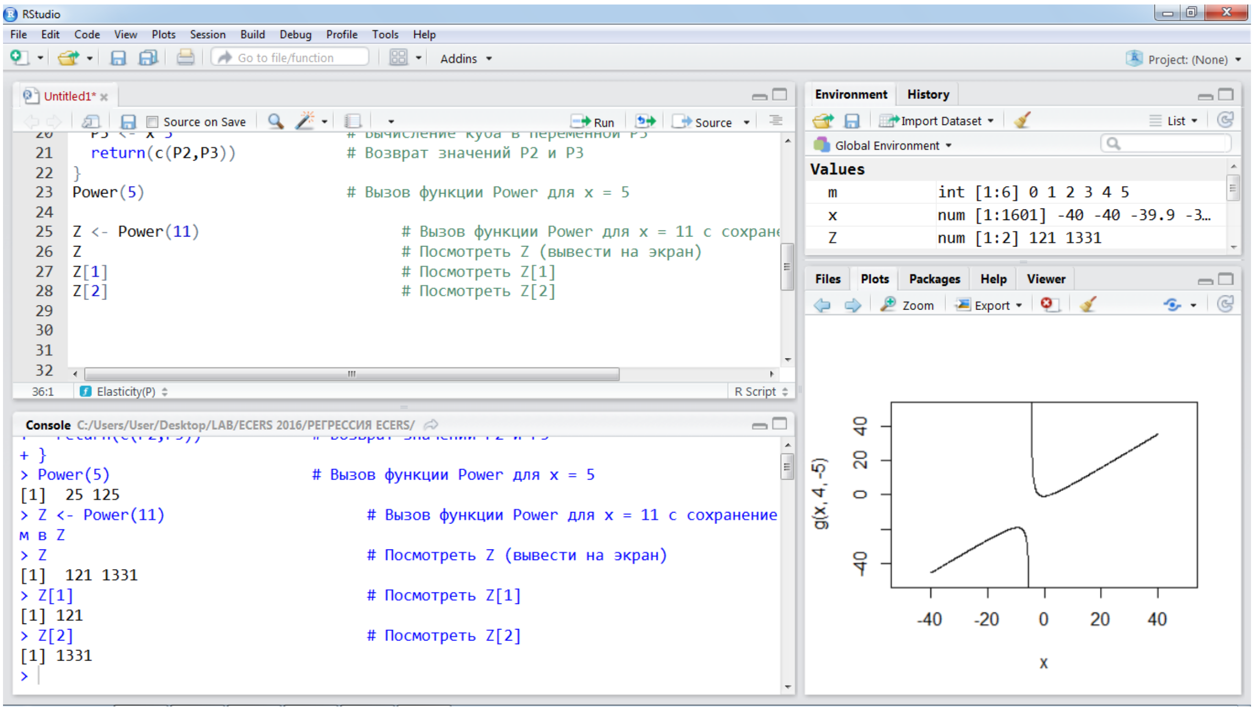
Здесь мы могли бы написать return(P2), и тогда функция вернула бы значение квадрата. Для случая return(P3) получили бы на выходе значение куба. Однако мы остановились на return(c(P2, P3)), и это означает, что будут выведены оба значения, т.к. результатом выполнения функции будет формирование массива из двух чисел: c(P2, P3). *Замечание:* вообще, оператор c(x,y,z,…) объединяет свои аргументы в вектор (массив), но об этом позднее.

В итоге после отработки набранного кода, действительно, получаем строку с квадратом и кубом для $x = 5$:



Если бы мы присвоили значение данной функции какой-нибудь переменной, то эта переменная представляла бы собой обычный массив из двух значений. Например,

```
Z <- Power(11)          # Вызов функции Power для x = 11 с сохранением в Z
Z                       # Посмотреть Z (вывести на экран)
Z[1]                    # Посмотреть Z[1]
Z[2]                    # Посмотреть Z[2]
```



На практике часто стоит задача не только вычисления значения какого-либо признака, но и ответ на вопрос о качественном характере исследуемого явления. Например, при изучении зависимости спроса от предложения нас больше интересует не значения самих зависимостей, а будет ли спрос в принципе эластичным или нет для конкретных цен.

Задание 5. Запрограммировать в R функцию, отвечающую на вопрос: будет ли спрос (Q) эластичным относительно цены предложения (P) для функции $Q(P) = \frac{1}{1+P^2}$. Здесь под спросом Q мы понимаем долю желающих приобрести товар по цене P.

Решение. Фактически нам необходимо вычислить предельную эластичность функции $Q(P)$ по переменной P по формуле

$$E_{Q,P}(P) = \frac{E'(P)}{E(P)} P$$

и выяснить превышает ли ее модуль единицу (случай эластичного спроса) или нет:

$$\left|E_{Q,P}(P)\right| > 1.$$

В нашем случае

$$Q'(P) = \left(\frac{1}{1+P^2}\right)' = -\frac{2P}{(1+P^2)^2}.$$

Запишем следующий код в R:

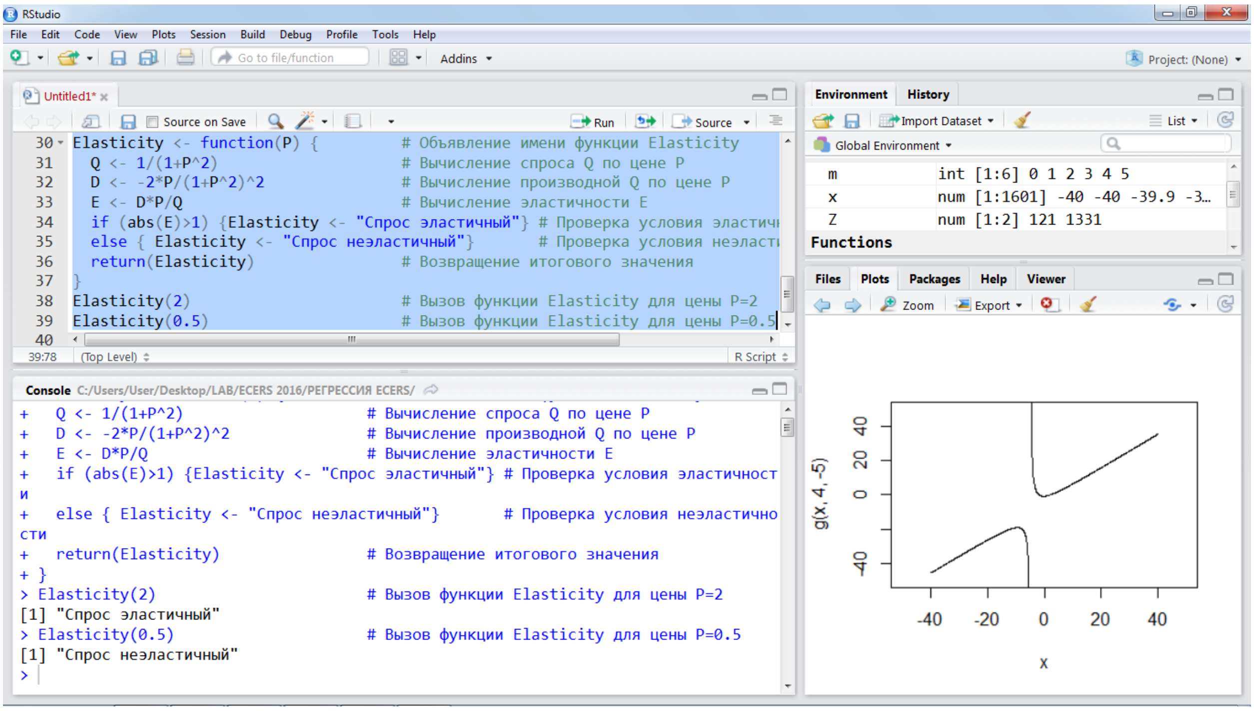
```
Elasticity <- function(P) {                    # Объявление имени функции Elasticity
  Q <- 1/(1+P^2)                               # Вычисление спроса Q по цене P
  D <- -2*P/(1+P^2)^2                          # Вычисление производной Q по цене P
  E <- D*P/Q                                   # Вычисление эластичности E
  if (abs(E)>1) {Elasticity <- "Спрос эластичный"}    # Проверка условия эластичности
  else { Elasticity <- "Спрос неэластичный"}           # Проверка условия неэластичности
  return(Elasticity)                                    # Возвращение итогового значения
}
Elasticity(2)                                  # Вызов функции Elasticity для цены P=2
Elasticity(0.5)                                # Вызов функции Elasticity для цены P=0.5
```

Здесь использован условный оператор if else. В синтаксисе языка R он имеет представление:

if (A) {…} else {…}

Если условие A выполняется, то производится группа операторов из первых скобок{…}, иначе – из последних. **Важно:** Если бы мы хотели записать условие «равно», то использовали бы двойной знак равенства if(abs(E) == 1)…, а если бы хотели «не равно», то соответствнно if (abs(E) != 1)…

Обратите внимание, что значением нашей функции в данном примере является строка, а не число.



Задания для самостоятельной работы

1. Вычислить значения $\sin x$ для первых ста целых чисел: 1..100.
2. Построить график функции $\operatorname{sign} x$ на отрезке $[-2, 2]$. *($\operatorname{sign} x$ – функция, возвращающая знак числа x, т.е. +1 для положительных и –1 для отрицательных, в нуле – ноль).* Указание: используйте встроенную в R функцию sign.
3. Объявить в R функцию $\operatorname{Separate}(x)$, которая возвращает два числа: целую и дробную части x. Построить их графики на отрезке $[-3, 3]$.
4. Объявить в R функцию $\operatorname{Sink}(x) = \frac{\sin x}{x}$ и построить ее график на отрезке$[-20, 20]$. Количество точек на отрезке взять равным 401. Будет ли данная функция непрерывна в нуле?
5. Запрограммировать функцию $\mathrm{W}(x)$, заданную системой:

6. $$W(x) = \begin{cases} x^2, & x \geq 0 \\ \sin(2x), & x < 0 \end{cases}$$

и построить ее график на отрезке $[-10;\ \ 5]$.