**Практическая работа 2**

**R - калькулятор**

**Оператор комментария #**

Именно оставленные нами пояснения делают текст программ в каком-то смысле интерактивным: ремарки напоминают нам о смысле введенных ранее команд. В языке R таким оператором является символ решетки: #, т.е. компилятор R не будет воспринимать в качестве кода все, что написано после символа решетки «#» в текущей строке. Например, строка

# Здравствуй, мир! :)

послужит прекрасным заголовком, но само приветствие исполнено в R не будет.

**Загрузка библиотек**

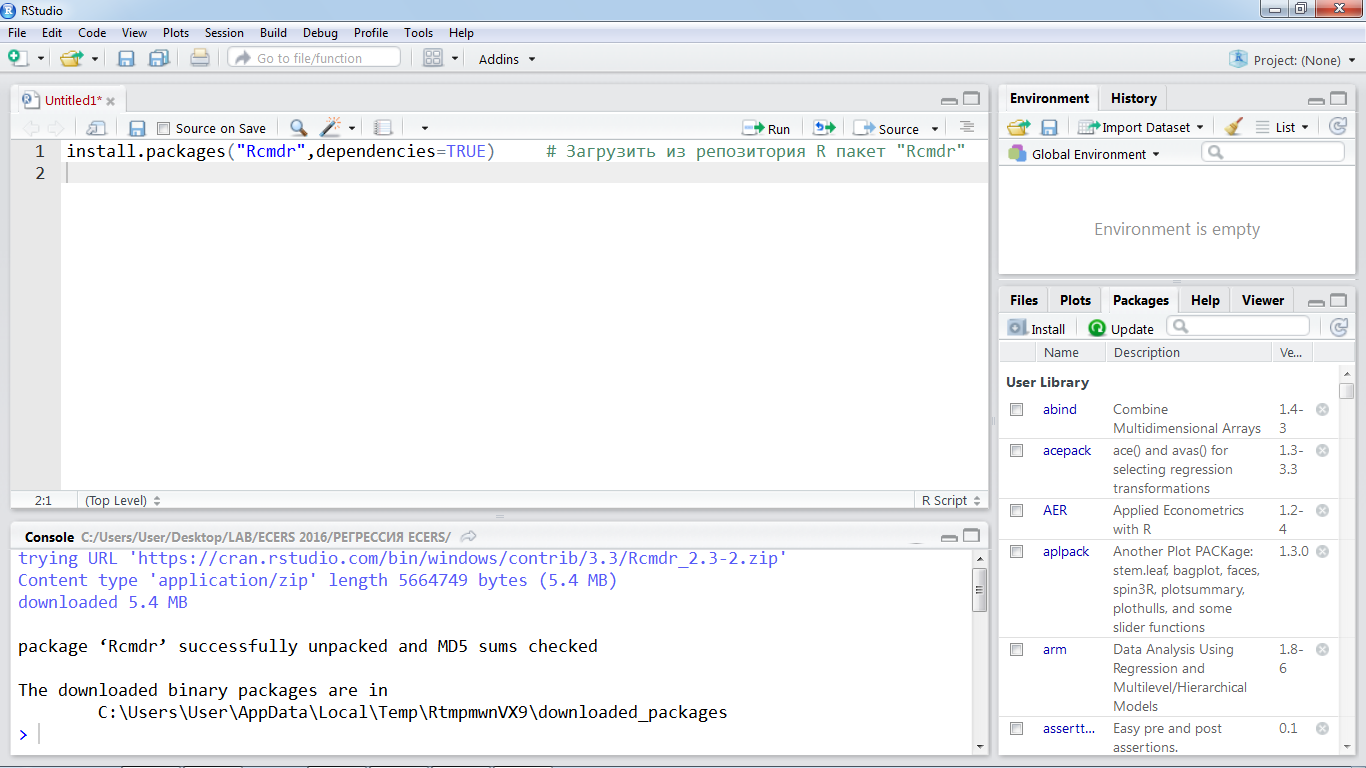
Необходимо подгрузить из интернета актуальные библиотеки используемых процедур и функций в R. Это можно сделать двумя способами.

*Первый способ:* набрать специальную команду загрузки в R требуемого пакета. Например, обязательной первичной загрузкой основной базы программных пакетов в R является загрузка библиотеки "Rcmdr". Введем в первой строке рабочего листа команду (можно просто скопировать текст через буфер обмена):

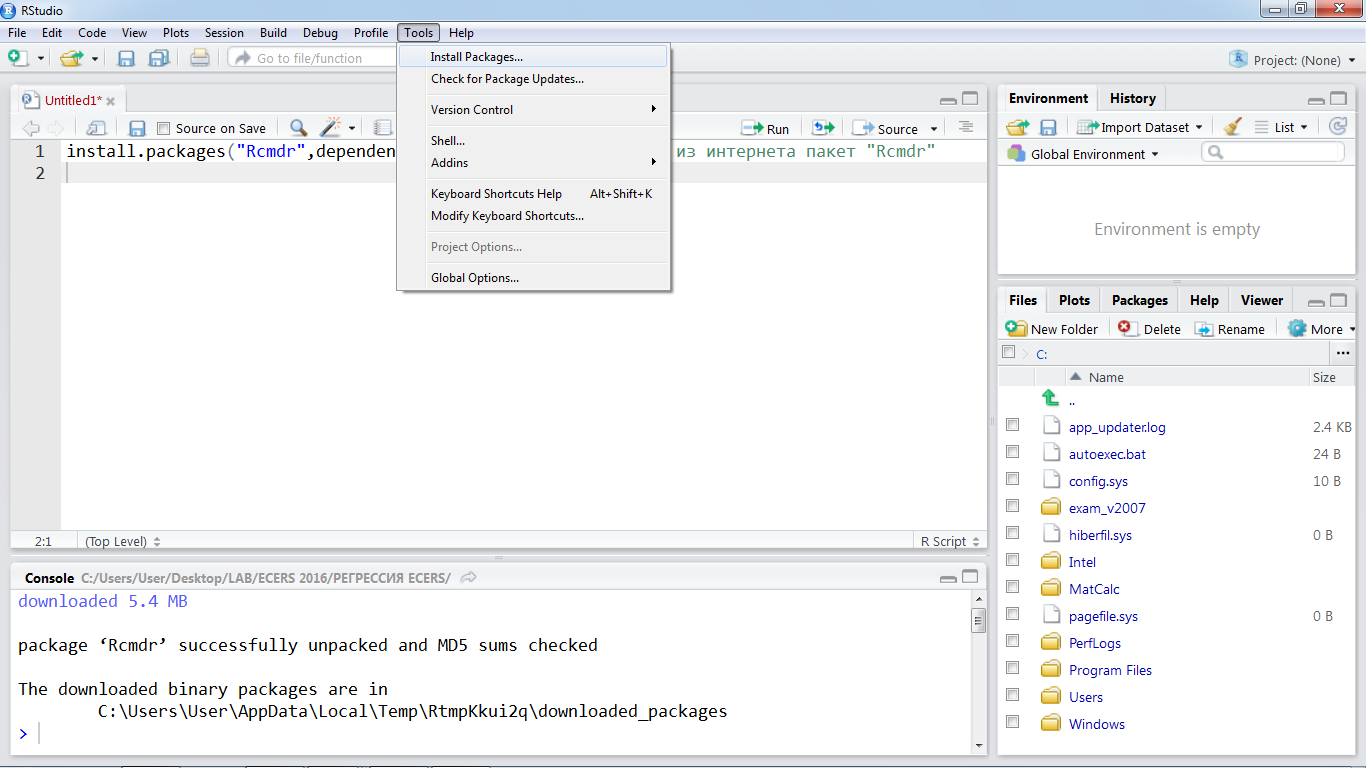
install.packages("Rcmdr") # Загрузить из репозитория R пакет "Rcmdr"

Важно знать, что в языке R различаются строчные и прописные буквы, то есть символы «a» и «A» – разные! Здесь Rcmdr не равно rcmdr!

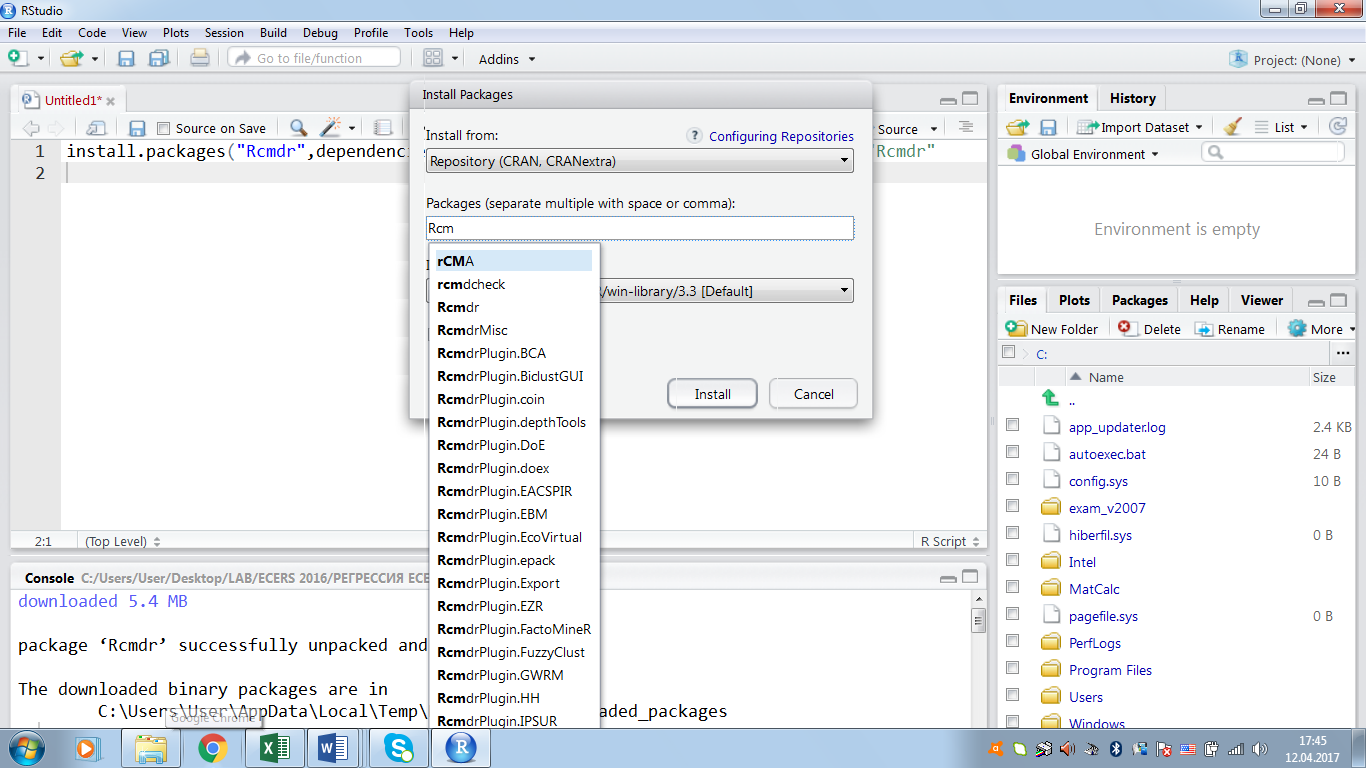
Если теперь после набора этой строки нажать Enter, то курсор перескочит на новую строку и ничего более не произойдет. Но если нажать сочетание Ctrl+Enter, то курсор также перескочит на новую строку, но код предыдущей строки при этом будет направлен на компиляцию, что приведет к выполнению указанной команды. Можно будет заметить, как R подгружает из интернета необходимые модули, это занимает примерно 5-7 минут. В итоге мы увидим сообщение об успешной установке:



*Второй способ:* вызвать интерактивное окно загрузчика библиотек, пройдя по меню Tools –> Install Packages…



В появившемся окне остается только вводить необходимые имена пакетов (если несколько сразу, то через пробел) и нажимать кнопку Install:



Однако, даже когда библиотека загружена на компьютер одним из описанных способов, обращение к ее процедурам и функциям в текущем сеансе (в текущем запуске RStudio) остается недоступным, пока мы не подключим (не активизируем) в текущей сессии эту библиотеку командой library(name). Это сделано в R для экономии оперативной памяти.

Например, мы хотим использовать расширенные возможности алгебры матриц с помощью библиотеки "Matrix". Тогда мы должны один раз загрузить этот пакет на компьютер командой:

install.packages("Matrix") # Загрузить из репозитория R пакет "Matrix"

и всякий раз, когда собираемся использовать этот пакет, нам необходимо в начале сессии запускать команду активации:

library(Matrix) # Активизировать загруженный в R пакет "Matrix"

**Задание 1.** В R есть встроенные данные о зависимости скорости автомобилей от тормозного пути (исследование Ford, 1920 год). Эти данные хранятся в зарезервированной переменной под именем «cars». Для того, чтобы понять, как хранятся данные о скоростях и дистанциях тормозных путей в cars требуется запустить команду glimpse(cars), которая входит в библиотеку dplyr.

**Решение**

Фактически нам необходимо сделать три небольшие операции:

1. Загрузить на компьютер библиотеку dplyr (это очень удобная и популярная библиотека для манипуляций с различными данными):

install.packages("dplyr") # Загрузить из репозитория R пакет "dplyr"

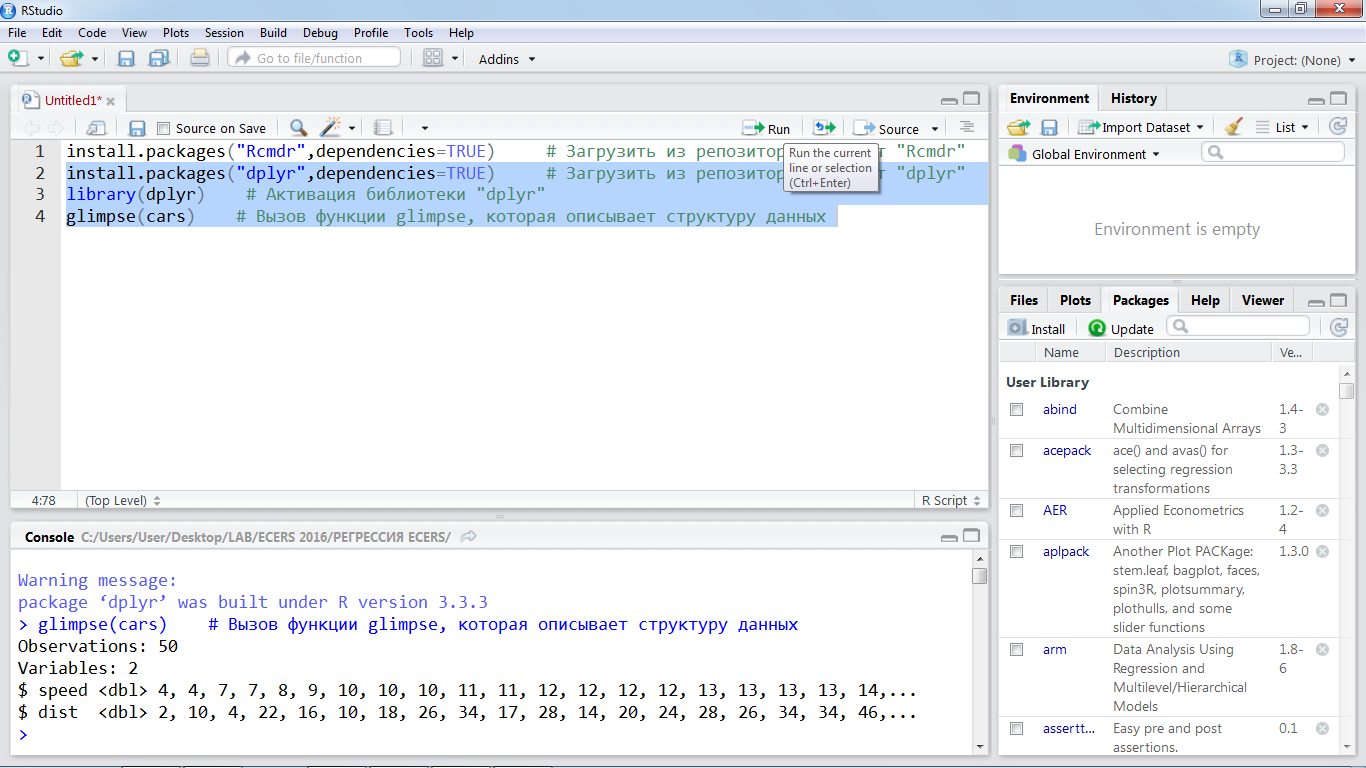
1. Активизировать эту библиотеку в текущей сессии:

library(dplyr) # Активация библиотеки "dplyr"

1. Вызвать соответствующую функцию из пакета:

glimpse(cars) # Вызов функции glimpse, которая описывает структуру данных

Далее мы выделяем эти три набранные строки левой клавишей мыши и запускаем код на компиляцию, щелкнув мышкой по клавише Run (или нажав Ctrl+Enter):



В результате в окне консоли получим полную информацию о переменной cars. Мы вернемся к этому отчету несколько позже.

**R – калькулятор**

Перечислим основные математические функции, набрав которые в качестве кода, мы мгновенно получим вычисленные значения:

sin(2); cos(2); tan(2); asin(0.5); acos(0.5); atan(2); log(0.3); log(1024, 2); exp(2); log10(4); sinh(2); cosh(2); tanh(2); asinh(2); acosh(2); atanh(1/2);

*Замечание. Здесь и далее условно в качестве аргументов функций взяты произвольные числа.*

Обратим особое внимание, что в R принято использовать точку в качестве десятичного разделителя (в Excel, кстати, более заумно: в ячейках используют запятую, а в макросах – точку).

Расшифруем некоторые из приведенных функций:

exp(2) # Экспонента от 2,

log(1024, 2) # Логарифм 1024 по основанию 2,

log(0.3) # Натуральный логарифм числа 0.3,

abs(-5) # Модуль от -5,

atan2(0,-3) # Угол между осью ox и вектором (-3,0) / здесь x,y наоборот!

2\*\*3; 2^3 # Возведение в степень 2 в 3, / возможны оба варианта

sqrt(4) # Корень из 4,

factorial(5) # Факториал числа 5, 5!=1\*2\*3\*4\*5

choose(5,3) # Число сочетаний 3 из 5:

pi # Число π

exp(1) # Число

Полезны также функции округления (представление чисел):

round(1.0023857, 6) # Округляет число 1.0023857 до 6 знака после запятой включ.

signif(1.0023857, 6) # Округляет число 1.0023857 до 6 значащих цифр включ.

trunc(-3.999) # Отсечение дробной части -3.999

floor(-4.2) # Наибольшее целое, не превосходящее -4.2

**Задание 2.** Вычислить с точностью не более четвертой цифры после запятой выражение

.

**Решение.** Разумеется, нам не следует сразу же бросаться набирать этого «крокодила» в строке. Надежнее всего разбить данное выражение на небольшие части, сохранить промежуточные вычисления в новых переменных и потом уже образовать ответ:

a <- tan(30.231^(1/3))^2 #Аргумент логарифма помещен в переменную a

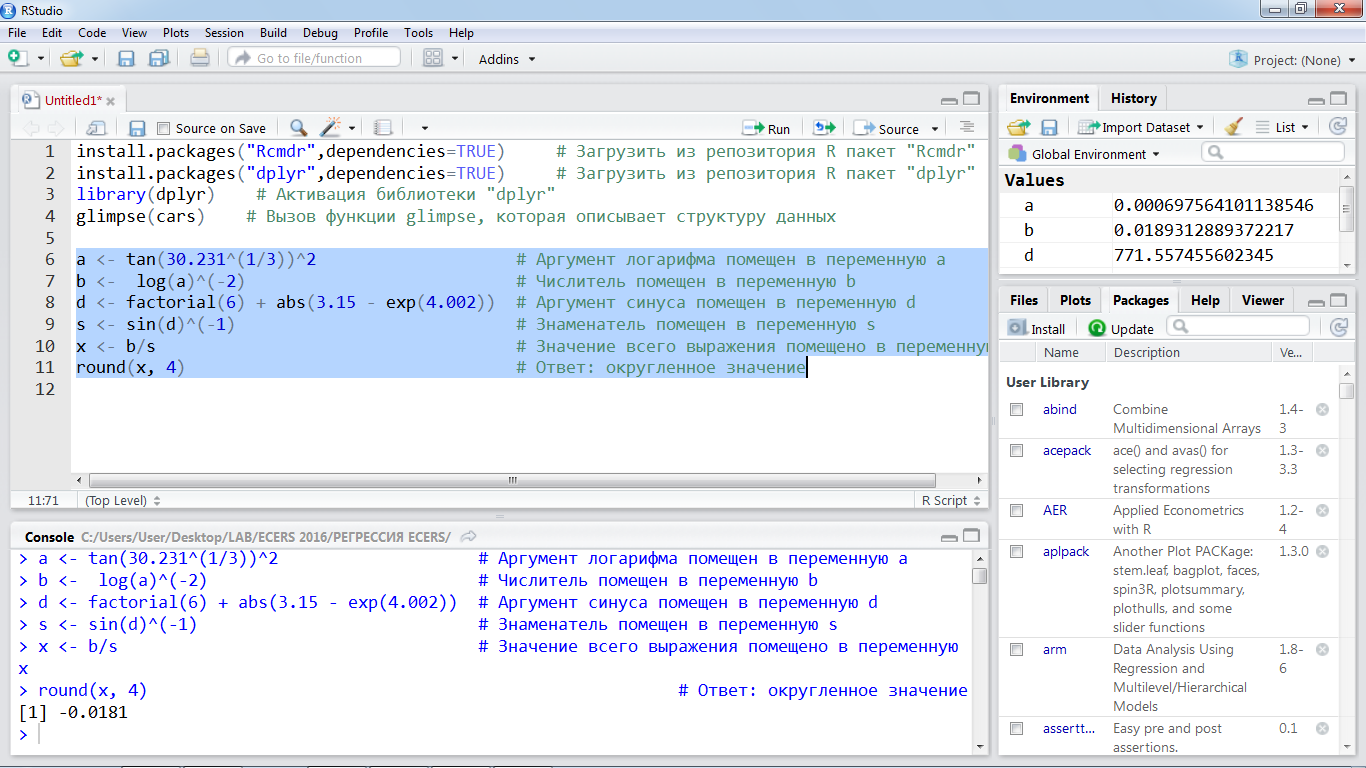
b <- log(a)^(-2) # Числитель помещен в переменную b

d <- factorial(6) + abs(3.15 - exp(4.002)) # Аргумент синуса помещен в переменную d

s <- sin(d)^(-1) # Знаменатель помещен в переменную s

x <- b/s # Значение всего выражения помещено в переменную x

round(x, 4) # Ответ: округленное значение



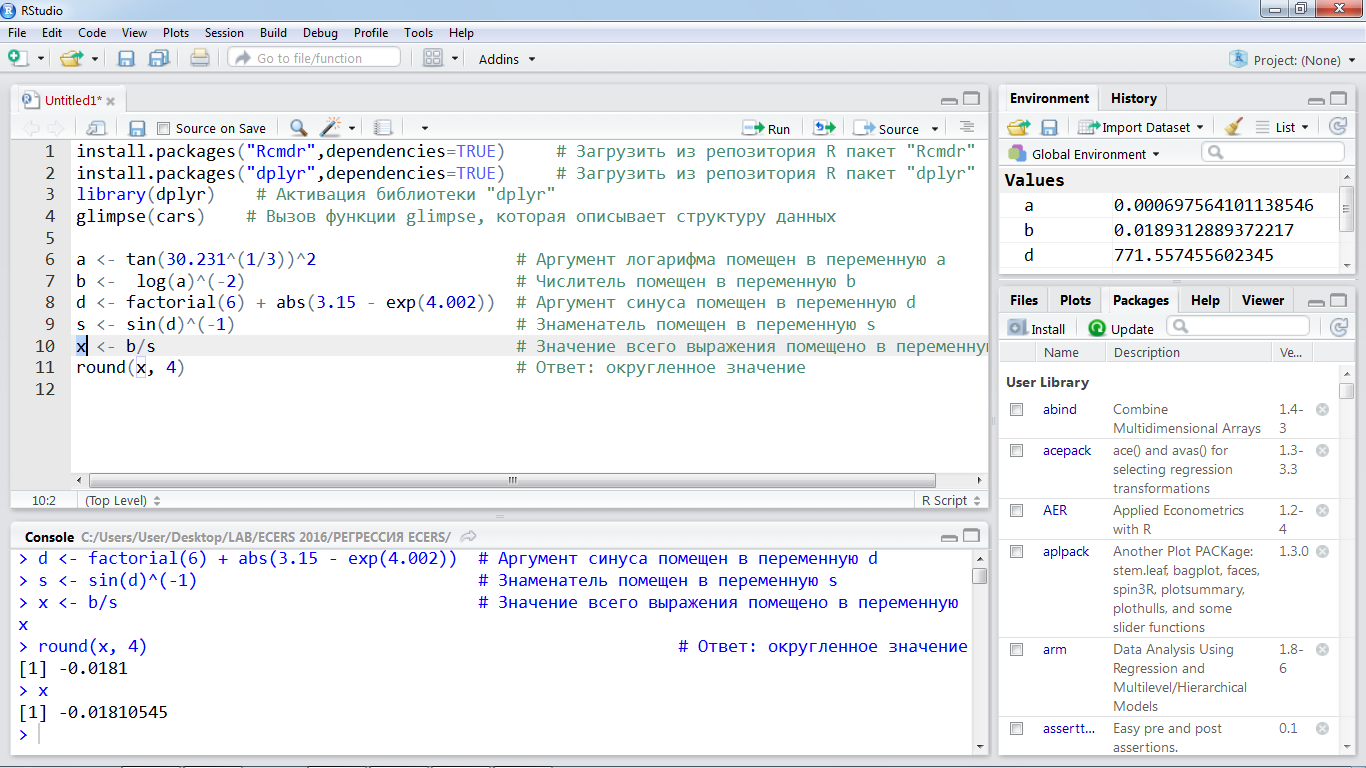
Обратите внимание, что значения наших промежуточных переменных отобразились в правом верхнем окне окружения. Если значений так много, что они не помещаются в данное окно, то узнать значение интересуемой переменной можно и по-другому. Как вариант просто написать ее на свободной строке и нажать Ctrl+Entr:

x

либо указать ее через точку с запятой с основным выражением:

x <- b/s ; x # Значение всего выражения помещено в переменную x

либо выделить мышкой именно эту переменную в тексте кода и также отправить на компиляцию Ctrl+Entr (или кликнуть по кнопке Run):



**Точность вычислений**

Помимо округления числовых результатов может встать обратная необходимость в повышении точности. Следующая команда позволяет повысить выводимую точность *double*-арифметики до 22 используемых цифр/разрядов в записи числа (параметр *digits*):

options(digits=12) # Установка максимального количества используемых цифр на уровне 12

Для получения еще большей точности следует использовать специализированные пакеты, например, пакет "Rmpfr", справка по которому станет доступна из RStudio после стандартной установки пакета из репозитория.

**Массивы чисел в R**

Примером массива чисел может служить рассмотренный в задании №1 объект cars с данными о скоростях и тормозных путях. Напомним, что отчет в окне консоли о структуре объекта cars по функции **glimpse(cars)** имел вид:

Observations: 50

Variables: 2

$ speed <dbl> 4, 4, 7, 7, 8, 9, 10, 10, 10, 11, 11, 12, 12, 12, 12, 13, 13, 13, 13, 14,...

$ dist <dbl> 2, 10, 4, 22, 16, 10, 18, 26, 34, 17, 28, 14, 20, 24, 28, 26, 34, 34, 46,...

из которого следует, что переменная cars представляет собой массив данных (более точно таблицу данных – тип data.frame, но сути это не меняет):

«Оbservations: 50» дословно означает «Наблюдений 50 штук» и, если представлять себе эти данные как таблицу, то речь идет о 50 строках;

«Variables: 2» дословно означает «Переменных 2 штуки» и фактически это – количество столбцов;

«$ speed <dbl> 4,4,…» означает, что первый столбец в переменной cars называется «speed» (данные о скоростях), он относится к типу «dbl». double – двойная точность – вещественные числа, модуль которых не превышает , все что превосходит этот рубеж обозначается в R символом бесконечности «Inf» (не правда ли, сложно себе представить реальное количество чего-либо, выраженное единицей с 308 нулями);

«$ dist <dbl> 2, 10,…» то же, но с названием второго столбца как «dist».

Иными словами, данные о скорости и соответствующей дистанции сгруппированы в двух столбцах, а по строчкам соответствуют одному и тому же наблюдению (конкретной испытуемой машине).

В R это означает, что можно обратиться к различным наблюдениям, собранным в объекте cars, следующими способами:

cars # Сам объект-массив, вывод на экран консоли

cars$speed # Массив скоростей (вектор)

cars[ ,1] # То же самое (значения всех строк из первого столбца)

cars$dist # Массив тормозных путей (вектор)

cars[ ,2] # То же самое (значения всех строк из второго столбца)

cars[1, ] # Первое наблюдение (значения всех столбцов из 1-ой строки)

cars[23,] # 23-е наблюдение (значения всех столбцов из 23-тьей строки)

cars[6,1] # Скорость при 6-ом наблюдении

cars$speed[6] # То же самое

cars[5,2] # Тормозной путь при 5-ом наблюдении

cars$dist[5] # То же самое

Сравните построчечно выводы на консоли R, запустив каждую строку в отдельности Ctrl+Enter.

**Задание 3.** Вычислить среднюю скорость первых пяти наблюдений в объекте cars.

**Решение.** В свободной строке составим код, вычисляющий среднее арифметическое первых пяти скоростей:

(cars$speed[1] + cars$speed[2] + cars$speed[3] + cars$speed[4] + cars$speed[5])/5

или короче

(cars[1,1] + cars[2,1] + cars[3,1] + cars[4, 1] + cars[5,1])/5 # То же самое

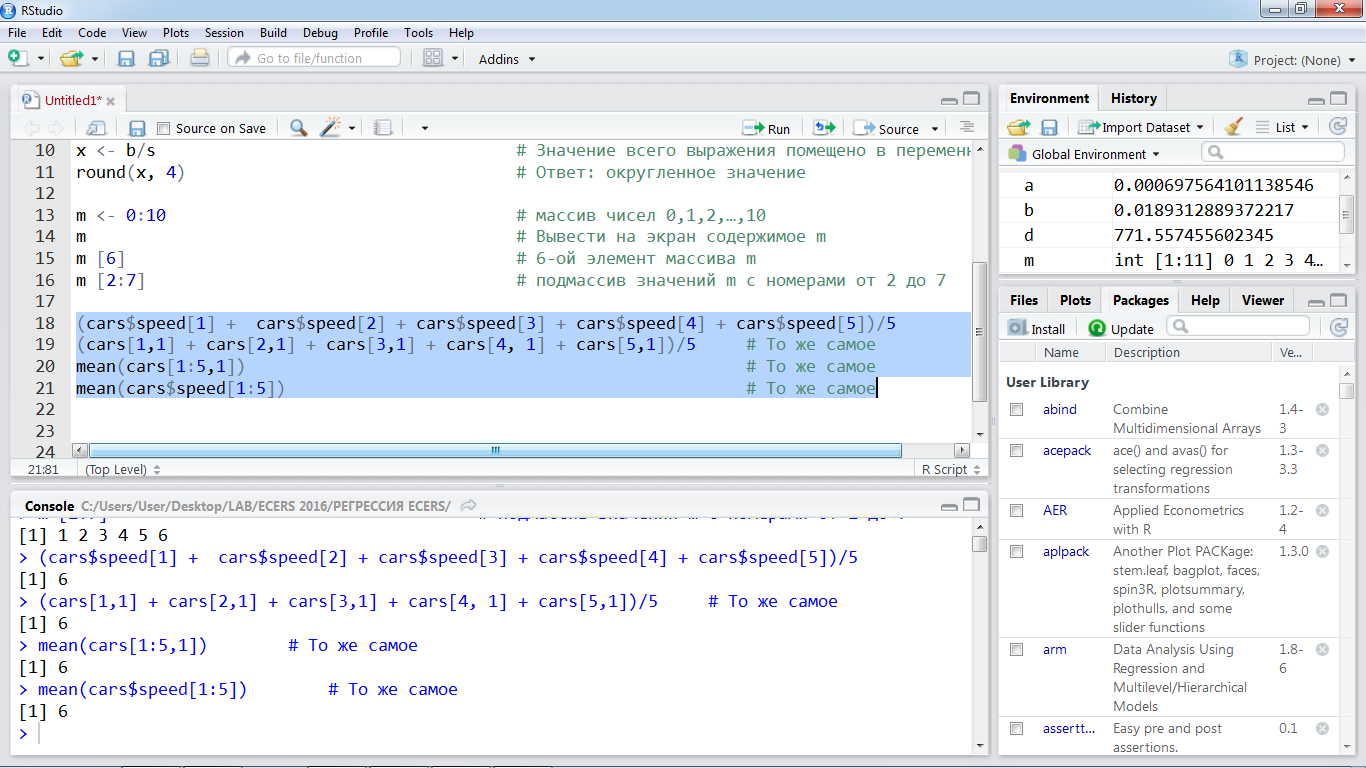
или еще короче

mean(cars[1:5,1]) # То же самое

или так

mean(cars$speed[1:5]) # То же самое

Последние две команды вызывают встроенную функцию вычисления среднего арифметического (mean) для элементов первого столбца (speed) массива cars, взятых из строк с 1-ой по 5-ую включительно.



Вообще интересно, что если набрать предыдущее выражение, не указав конкретный диапазон наблюдений, то мы вычислим среднюю скорость всех испытуемых машин (1920 г.)

mean(cars$speed) # Средняя скорость всех испытуемых машин

которая составит 15,4 mph, что эквивалентно 24,78 км/ч. (сравните со средними скоростями сегодня).

**Задания для самостоятельной работы**

1. Вычислить выражения с точностью в 6 значащих цифр

;

1. Вычислить выражение с точностью в 3 цифры после запятой

;

1. Вычислить среднее арифметическое значение длины тормозного пути для данных cars, выраженное в метрах. Использовать: в 1 футе 0,3048 метра. (Ответ: 13,1м.)
2. После загрузки библиотеки "ggplot2" в R становится доступной таблица данных под именем diamonds, в которой приведены статистические исследования алмазов. Определить сколько алмазов было исследовано и найти средний вес алмазов (в каратах).

*Указание.* Загрузить библиотеку "ggplot2" и использовать функцию glimps из пакета "dplyr", чтобы понять какую переменную из таблицы необходимо выбрать для дальнейших действий.

1. Проверить, действительно ли при очень малых значениях функция . На какую, в таком случае, функцию будет похож?

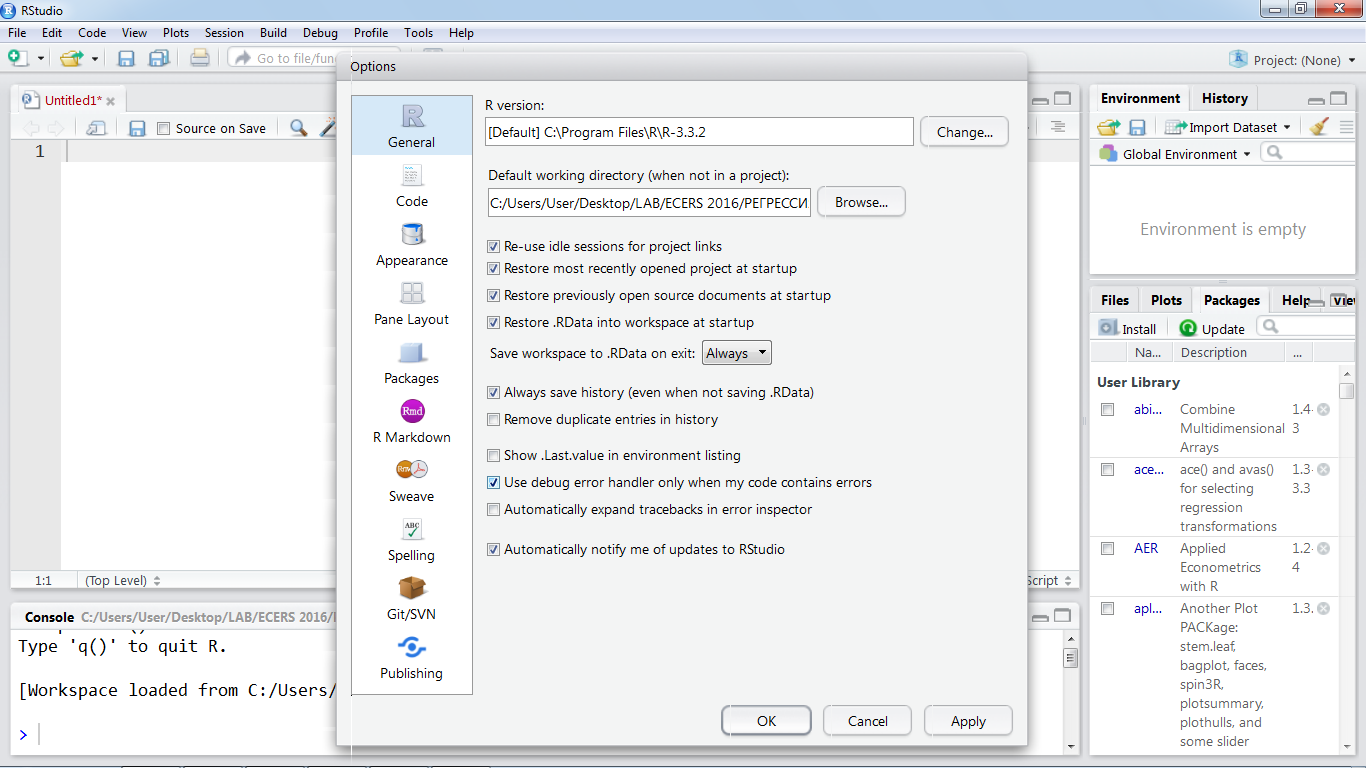
Приложение

**Некоторые дополнительные настройки RStudio**

Пройдите по пунктам основного меню Tools –> Global Options.

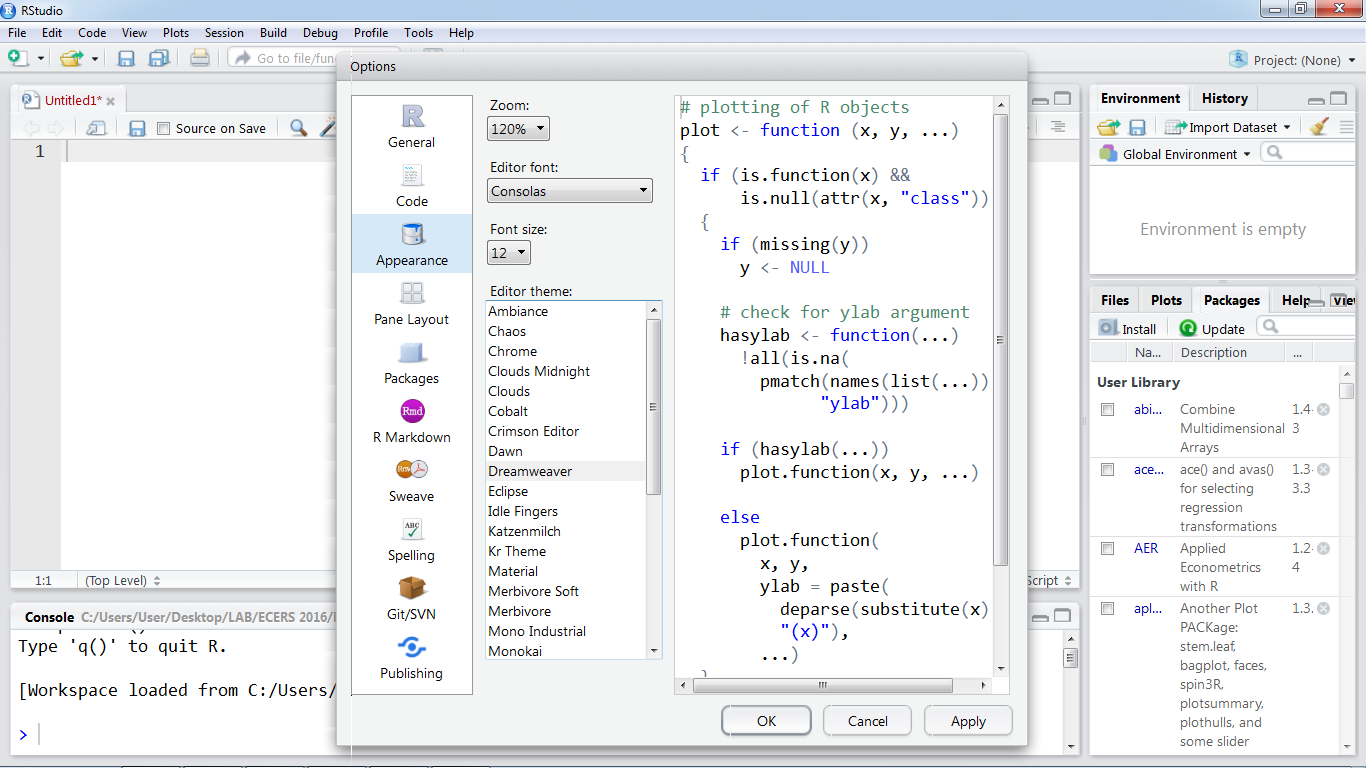
**Подменю General** В появившемся диалоговом окне обязательно укажите во второй строчке путь к рабочей директории R (default working directory). Именно сюда будут записываться все ваши данные по умолчанию и именно отсюда R будет предлагать вам открыть существующие файлы.

Полезно поставить галочки в соответствующие места, регламентирующие автосохранение и пр.



**Подменю Appearance** Данные настройки позволяют выбрать наиболее приятную цветовую палитру, сочетания шрифтов и масштаб отображения на экране. Смело выбирайте различные темы и экспериментируйте с Zoom и Editor font. Не забывайте по итогу нажимать клавишу Apply.

На рисунке приведена настройка, используемая автором данной методички:



*Замечание.* При выборе Editor font не увлекайтесь – не все шрифты распознают кодировки CP1251 и UTF8.

**Подменю Pane Layout** Данный раздел позволяет настроить формат вывода информации в четыре окна RStudio, перенастроив решительно все, что можно.