Лабораторная работа № 1.2
Ввод и визуализация данных в R. Простейшие графические построения в R.

Цель:

- исследовать возможности языка R для визуализации данных в R;

Время: 2 часа

Лабораторное оборудование: персональные компьютеры, выход в сеть Internet, RStudio.

Краткие теоретические сведения

Простейшие графические построения в R.

Основные графические функции пакета R:
Рисование графиков

- plot(x) — график x
- plot(x, y) — график зависимости y от x
- hist(x) — гистограмма
- barplot(x) — столбчатая диаграмма
- dotchart(x) — диаграмма Кливленда
- pie(x) — круговая диаграмма
- boxplot(x) — график типа "коробочки с усами"
- sunflowerplot(x, y) — то же, что и plot(), однако точки с одинаковыми координатами изображаются в виде "ромашек", количество лепестков у которых пропорционально количеству таких точек
- coplot(x~y | z) — график зависимости y от x для каждого интервала значений z
- interaction.plot(f1, f2, y) — если f1 и f2 — факторы, эта фукнция создаст график со средними значениями y в соответствии со значениями f1 (по оси x) и f2 (по оси y, разные кривые)
- matplot(x, y) — график зависимости столбцов y от столбцов x
- fourfoldplot(x) — изображает (в виде частей окружности) связь между двумя бинарными переменными в разных совокупностях
- assocplot(x) — график Кохена-Френдли
- mosaicplot(x) — мозаичный график остатков лог-линейной регрессии
- pairs(x) — если x - матрица или таблица данных, эта функция изобразит диаграммы рассеяния для всех возможных пар переменных из x
- plot.ts(x), ts.plot(x) — изображает временной ряд
- qqnorm(x) — квантили
- qqplot(x, y) — график зависимости квантилей y от квантилей x
- contour(x, y, z) — выполняет интерполяцию данных и создает контурный график
- filled.contour(x, y, z) — то же, что contour(), но заполняет области между контурами определёнными цветами
- image(x, y, z) — изображает исходные данные в виде квадратов, цвет которых определяется значениями x и y
- persp(x, y, z) — то же, что и image(), но в виде трехмерного графика
- stars(x) — если x — матрица или таблица данных, изображает график в виде "звезд" так, что каждая строка представлена "звездой", а столбцы задают длину сегментов этих "звезд"

- symbols(x, y, ...) — изображает различные символы в соответствии с координатами
- termplot(mod.obj) — зображает частные эффекты переменных из регрессионной модели

Рисование графиков на низком уровне

- points(x, y) — рисование точек
- lines(x, y) — рисование линии
- text(x, y, labels, ...) — добавление текстовой надписи
- mtext(text, side=3, line=0, ...) — добавление текстовой надписи
- segments(x0, y0, x1, y1) — рисование отрезка
- arrows(x0, y0, x1, y1, angle= 30, code=2) — рисование стрелочки
- abline(a,b) — рисование наклонной прямой
- abline(h=y) — рисование вертикальной прямой
- abline(v=x) — рисование горизонтальной прямой
- abline(lm.obj) — рисование регрессионной прямой
- rect(x1, y1, x2, y2) — рисование прямоугольника
- polygon(x, y) — рисование многоугольника
- legend(x, y, legend) — добавление легенды
- title() — добавление заголовка
- axis(side, vect) — добавление осей
- rug(x) — рисование засечек на оси X
- locator(n, type = "n", ...) — возвращает координаты на графике, в которые кликнул пользователь

Графические параметры

Многие характеристики диаграмм (шрифты, цвета, оси, названия) можно изменять при помощи опций, которые называются «графические параметры».

Один способ назначить эти параметры – использовать функцию `par()`. Значения параметров, заданные таким способом, будут действовать на протяжении всей сессии, пока вы не измените их. Формат применения функции таков: `par(название параметра=назначение, название параметра=назначение, ...)`. Функция `par()` без аргументов выводит на экран действующие значения графических параметров. Добавление аргумента `no.readonly=TRUE` позволяет увидеть только те графические параметры, которые можно изменять.

Можно использовать столько функций `par()`, сколько нужно, так что команда может быть также записана в виде

```
par(lty=2)
par(pch=17)
```

Второй способ задать графические параметры – это включить записи типа `название параметра=назначение` внутрь графической функции высокого уровня. В этом случае заданные параметры будут действовать только для конкретной диаграммы. Можно было бы построить тот же график при помощи следующего программного кода:

```
plot(dose, drugA, type="b", lty=2, pch=17)
```

Не во всех графических функциях высокого уровня можно изменять все возможные графические параметры.

Символы и линии

Графические параметры можно использовать для того, чтобы указывать тип символов и линий на диаграммах. Соответствующие параметры перечислены в табл. 1.

Таблица1. Параметры для указания типов символов и линий

Параметр	Описание
`pch`	Определяет тип символа (см. рис.1)
`cex`	Определяет размер символа. `cex`– это число, обозначающее, как символы должны быть масштабированы по отношению к размеру по умолчанию. 1 = размер по умолчанию, 1.5 – на 50% крупнее, 0.5 – на 50% мельче и т. д.
`lty`	Определяет тип линии (см. рис. 2)
`lwd`	Определяет толщину линии по сравнению с толщиной линии по умолчанию (1). Например, lwd=2 делает линию в два раза толще, чем по умолчанию

Параметр `pch=` определяет тип символов, которые используются на диаграмме. Возможные значения приведены на рис. 1.

Для символов с 21 по 25 можно отдельно указывать цвет контура (`border=`) и заполнения (`bg=`).

Рисунок 1. Символы, назначаемые при помощи параметра `pch`.

Параметр `lty=` применяется для обозначения нужного типа линии. Значения параметра показаны на рис. 2.

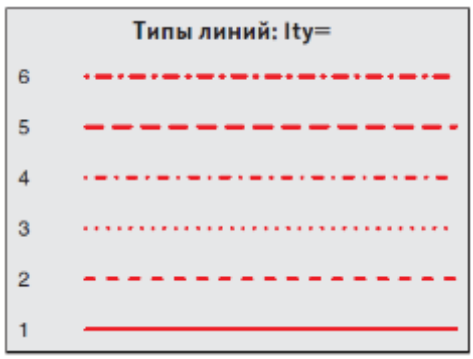


Рисунок 2. Типы линий, назначаемые при помощи параметра `lty`.

Программный код, объединяющий все эти параметры,

```
plot(dose, drugA, type="b", lty=3, lwd=3, pch=15, cex=2)
```

создаст график, на котором точечная линия в три раза шире, чем по умолчанию, соединяет наблюдения, представленные в виде заполненных квадратов в два раза большего размера, чем по умолчанию.

Результат представлен на рис. 3.

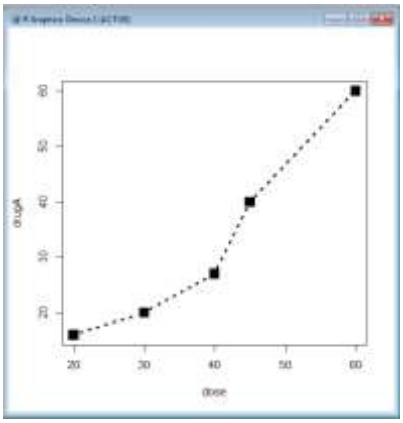


Рисунок 3. График зависимости реакции пациента от дозы лекарства А с измененными типом и шириной линии, а также типом и размером символов.

Цвета

В R есть несколько связанных с цветами параметров. В табл. 2 приведены некоторые из самых распространенных.

Таблица 2. Параметры для назначения цвета

Параметр	**Описание**
col	Цвет элементов на графике. Для некоторых функций (таких как `lines` и `pie`) можно указывать вектор из значений, которые используются по очереди. Например, если `col=c("red", "blue")` и изображены три линии, первая будет красной, вторая – синей и третья – красной.
col.axis	Цвет значений осей
col.lab	Цвет подписей осей
col.main	Цвет заголовков
col.sub	Цвет подзаголовков
fg	Цвет графика
bg	Цвет фона

В R цвета можно обозначать номером, названием, в шестнадцатеричной системе, а также в системах RBG или HSV. Например, `col=1`, `col="white"`, `col="#FFFFFF"`, `col=rgb(1,1,1)` и `col=hsv(0,0,1)` – взаимозаменяемые способы обозначить белый цвет. Функция `rgb`() определяет цвета по значениям красного, зеленого и синего, а `hsv`() основана на значениях оттенка и насыщенности.

Функция `colors`() выводит на экран список всех доступных цветов.

В R также реализован ряд функций, которые позволяют создавать векторы из близких цветов. К таким функциям относятся `rainbow`(), `heat.colors`(), `terrain.colors`(), `topo.colors`() и `cm.colors`(). Например, `rainbow`(10) создает 10 соседних "радужных" цветов. Оттенки серого создаются функцией gray().

В этом случае задаются оттенки серого в виде вектора чисел от 0 до 1. Команда `gray(0:10/10)` создаст 10 оттенков серого.

Пример: чтобы увидеть, как это работает можно запустить программный код

```
n <- 10
mycolors <- rainbow(n)
pie(rep(1, n), labels=mycolors, col=mycolors)
mygrays <- gray(0:n/n)
pie(rep(1, n), labels=mygrays, col=mygrays)
```

Характеристики текста

Графические параметры также используются для определения размера, шрифта и стиля текста. Параметры, определяющие размер шрифта, приведены в табл. 3. Параметры, при помощи которых можно указать тип шрифта, перечислены в табл. 4.

Таблица 3 Параметры, определяющие размер шрифта

Параметр	Описание
`cex`	Число, определяющее, как отображаемый на диаграмме текст будет масштабирован относительно размера по умолчанию (1). 1.5 – на 50% больше, 0.5 – на 50% меньше и т. д.
`cex.axis`	Размер значений на осях по отношению к `cex`
`cax.lab`	Размер названий осей по отношению к `cex`
`cex.main`	Размер заголовков по отношению к `cex`
`cex.sub`	Размер подзаголовков по отношению к `cex`

Например, на всех диаграммах, созданных после команды

```
par(font.lab=3, cex.lab=1.5, font.main=4, cex.main=2)
```

в 1.5 раза более крупные, чем по умолчанию, подписи осей будут выделены курсивом, а названия будут в два раза крупнее, чем по умолчанию, и еще выделены полужирным курсивом.

Таблица 4. Параметры, определяющие семейство, размер и стиль шрифта.

Параметр	Описание
`font`	Число, которое определяет шрифт для текста на диаграмме. 1 = обычный, 2 = полужирный, 3 = курсив, 4 = полужирный курсив, 5 = символы (в кодировке Adobe)
`font.axis`	Шрифт значений на осях
`font.lab`	Шрифт для подписей по осям
`font.main`	Шрифт для заголовков
`font.sub`	Шрифт для подзаголовков
`ps`	Размер точки в шрифте (приблизительно 0.3 мм)
`family`	Семейство шрифтов. Стандартные значения – `serif`, `sans` и `mono`

Размер и стиль шрифта установить просто, тогда как с семейством шрифтов дело обстоит немного сложнее. Это происходит потому, что отображение `serif`, `sans` и `mono` зависит от устройства. Например, под Windows `mono` отображается как TT Courier New, `serif` – как TT Times New Roman, а `sans` – как TT Arial (TT обозначает шрифт типа True Type). Если вы удовлетворены таким отображением семейств шрифтов, то можете использовать параметры типа `family="serif"`, чтобы добиться желаемого результата. Если вы не удовлетворены, вам нужно создать новую систему соответствий. Под Windows можете назначать эти соответствия при помощи функции `windowsFont`().

Например, после выполнения команды

```
windowsFonts(
A=windowsFont("Arial Black"),
```

```
B=windowsFont("Bookman Old Style"),
C=windowsFont("Comic Sans MS")
)
```

можно использовать A, B и C как названия семейств шрифтов.

В этом случае `par(family="A")` назначит шрифт Arial Black

Размеры диаграммы и полей

Можно определять размер диаграммы и полей при помощи параметров, приведенных в табл. 5.

Таблица 5. Параметры для определения размеров диаграммы и полей

Параметр	**Описание**
`pin`	Размер диаграммы (ширина, высота) в дюймах
`mai`	Числовой вектор, задающий размеры полей, где параметры `c(низ, лево, верх, право)` измеряются в дюймах
`mar`	Числовой вектор, задающий размеры полей, где `c(низ, лево, верх, право)` измеряются в числе строк. По умолчанию это `c(5, 4, 4, 2) + 0.1`

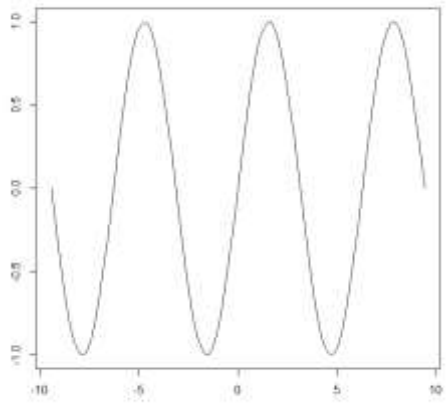
Команда `par(pin=c(4,3), mai=c(1,.5, 1, .2))` позволяет создавать диаграммы размером 4 дюйма в ширину и 3 дюйма в высоту с шириной полей сверху и снизу по одному дюйму, слева 0.5 дюйма и справа 0.2 дюйма.

Пример 1. Рассмотрим построение графика функции одной переменной на следующем примере:

Построение графика функции *Sin(x)* на интервале (-3**pi*, 3**pi*)

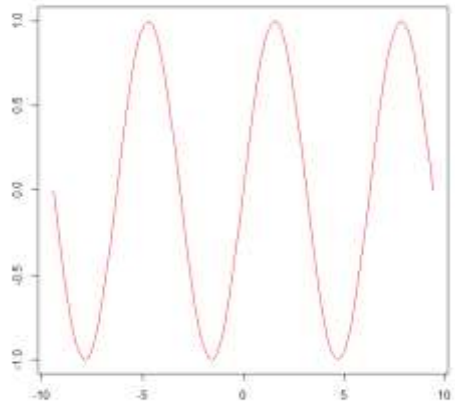
Построим график функции Sin на интервале (-3*pi, 3*pi)

plot(sin, -3*pi, 3*pi)

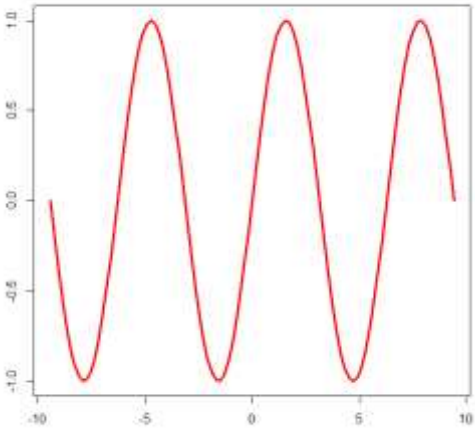


Зададим цвет линии (col - от англ.: color - цвет)

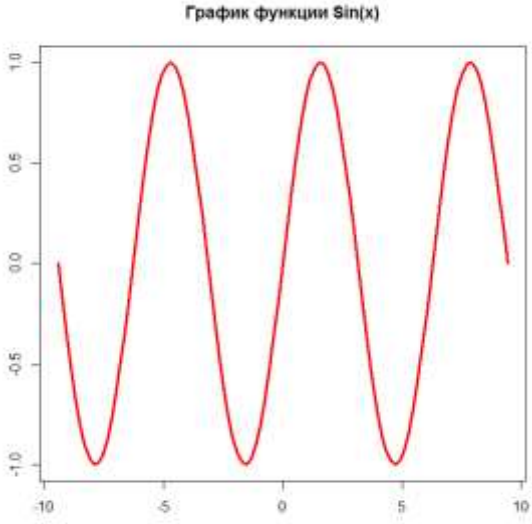
plot(sin, -3*pi, 3*pi, col = "red")



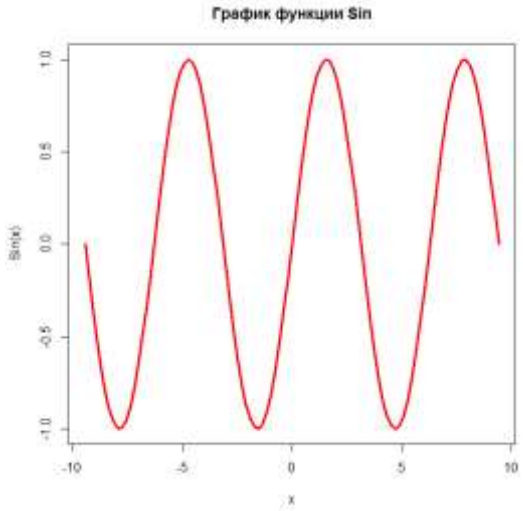
Зададим толщину линии (lwd - от англ.: line width - толщина линии)
plot(sin, -3*pi, 3*pi, col = "red", lwd = "3")



Создадим заголовок графика
plot(sin, -3*pi, 3*pi, col = "red", lwd = "3", main = "График функции Sin(x)")

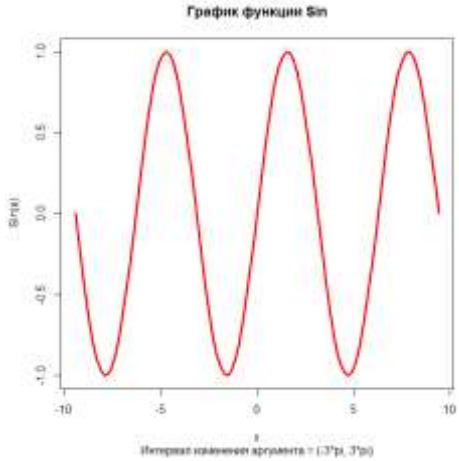


Подпишем оси графика
plot(sin, -3*pi, 3*pi, col = "red", lwd = "3", main = "График функции Sin", xlab = "x", ylab= "Sin(x)")



Создадим текст под графиком

plot(sin, -3*pi, 3*pi, col = "red", lwd = "3", main = "График функции Sin", xlab = "x", ylab= "Sin(x)", sub = "Интервал изменения аргумента = (-3*pi, 3*pi)")

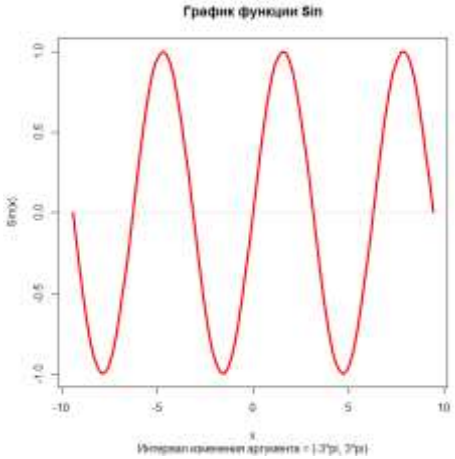


Добавим горизонтальный пунктир

Воспользуемся функцией abline, которая строит прямую y = a + bx

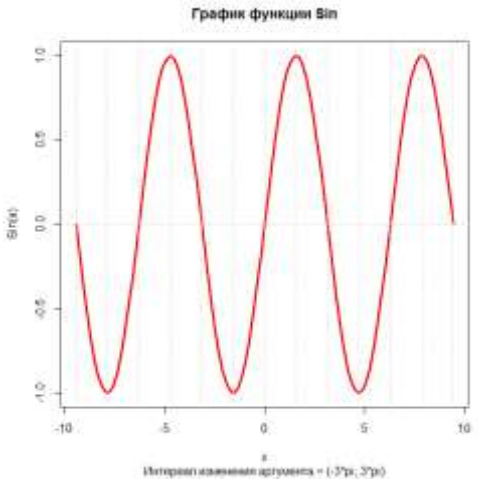
Параметр add=TRUE позволяет совместить прямую линию с графиком функции

abline(0, 0, col = "lightgray", lty = 2, add=TRUE)



Проведём вертикальные линии через точки (-3*pi, -5*pi/2, -2*pi, -3*pi/2, -pi, -pi/2, 0, pi/2, pi, pi/2, 3*pi/2, 2*pi, 5*pi/2, 3*pi)

abline(v=c(-3*pi, -5*pi/2, -2*pi, -3*pi/2, -pi, -pi/2, 0, pi/2, pi, pi/2, 3*pi/2, 2*pi, 5*pi/2, 3*pi), col = "light gray", lty = 2, add=TRUE)



Задание и порядок выполнения лабораторной работы №1.2

Задание 1.

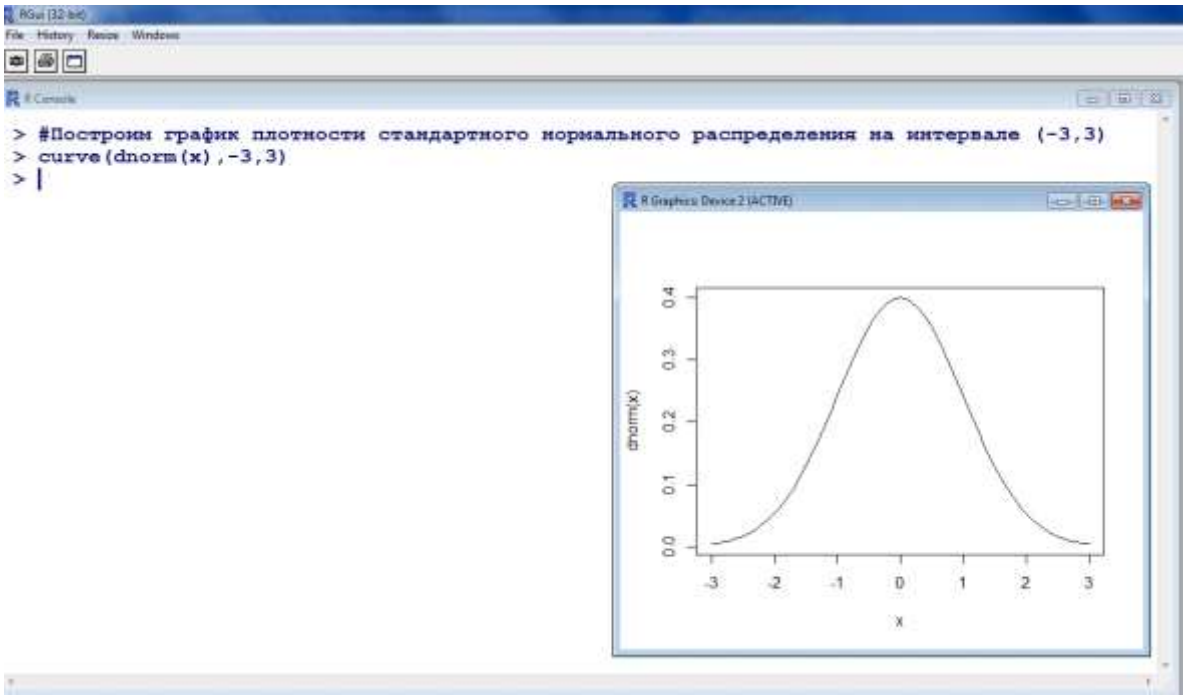
1. Задайте функцию одной переменной, отличную от линейной.
2. Постройте график функции на заданном интервале, с помощью параметров, использованных в рассмотренном выше примере, сделайте необходимые настройки.
3. При наличии особых точек (минимум, максимум, точка перегиба) отметьте их (на оси абсцисс) специальными символами и проведите через них пунктирные вертикальные и горизонтальные линии.

Задание 2.

Изучите приведённые ниже программы, выполняющий построение графиков плотности нормального определения (с различными параметрами). Ответьте на вопросы.

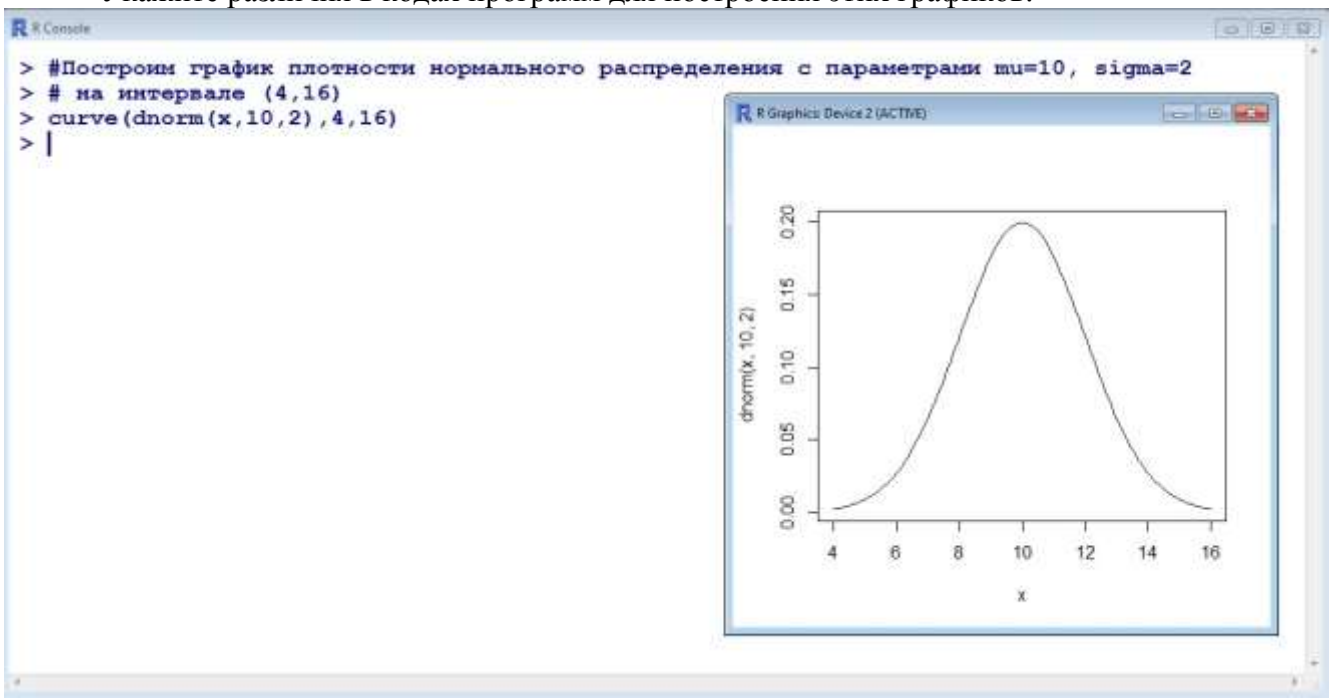
1. Каковы численные значения параметров этого закона распределения?

Присутствуют ли эти значения в программе и почему?

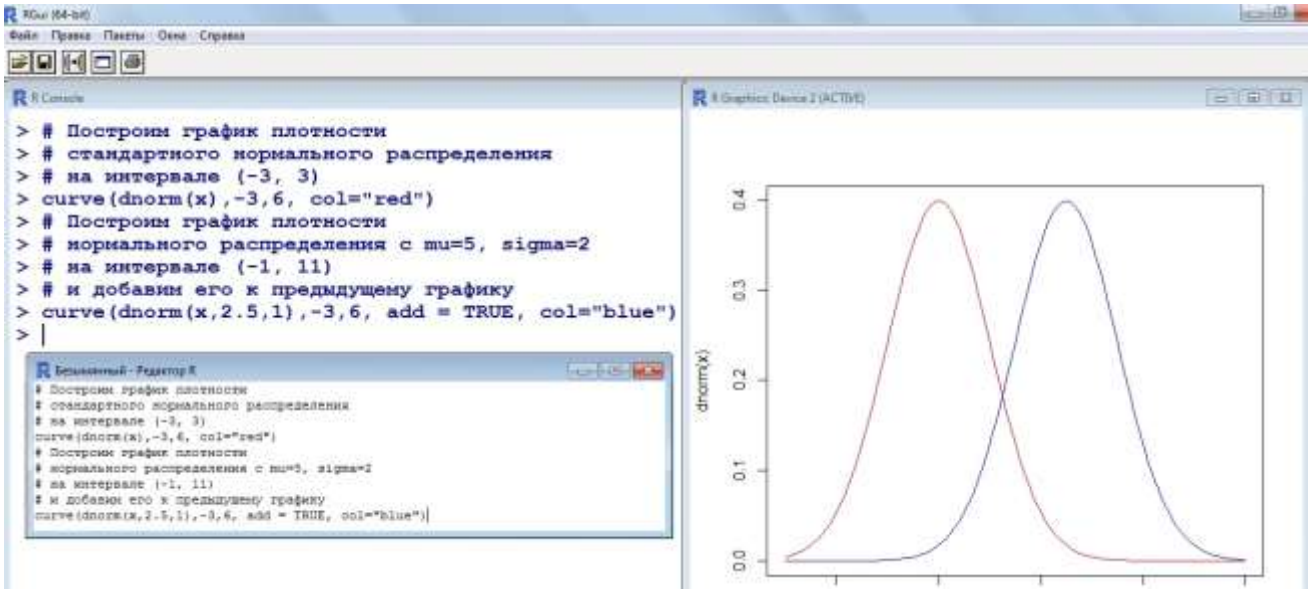


2. Сравните приведённый ниже график с графиком плотности стандартного нормального распределения.

Укажите различия в кодах программ для построения этих графиков.



3. Внимательно изучите код и исправьте комментарии к нему.



4. Изучить и выполнить код совмещения гистограммы и графика плотности распределения вероятностей

```
#
# Смоделируем выборку объёма 1000 из нормального распределения
# с параметрами: mu = 5; sigma = 2
n  = 1000
mu = 5
sigma = 2
sample = rnorm(n, mu, sigma)
#
# Построим гистограмму:
```

```
hist(sample, col = "red")

# Построим график плотности нормального распределения (с выбранными
параметрами)
# и совместим его с гистограммой:

curve(dnorm(x, mu, sigma)*n, lw="3", col="blue", add=TRUE)

# Обратите внимание: умножение на n нужно для форматирования
```

Задание 3. Ознакомьтесь с законами распределения вероятностей случайных величин, реализованными в R: https://r-analytics.blogspot.com/2012/12/r.html#.WbWaWshJaUk

Задание 4. построение графиков плотности нормального распределения с разными sigma

1. Постройте (на одном рисунке) графики плотности нормального распределения с одинаковыми средними значениями и разными среднеквадратическими отклонениями
2. Сделайте вывод о влиянии параметров нормального распределения на форму и расположение графика плотности.
3. Оформите задание как документ формата Word, включите в него заголовок (название задания), построенное изображение, значения параметров, использованных при построении графиков. Сформулируйте вывод.

Задание 5. Построение графиков плотностей вероятностных распределений

Используя функции `dnorm`, `dexp`, `dchisq`, `dt` и `dunif`, постройте графики плотностей следующих вероятностных распределений:

- нормального,
- экспоненциального,
- Хи-квадрат,
- Стьюдента,
- равномерного (на отрезке).

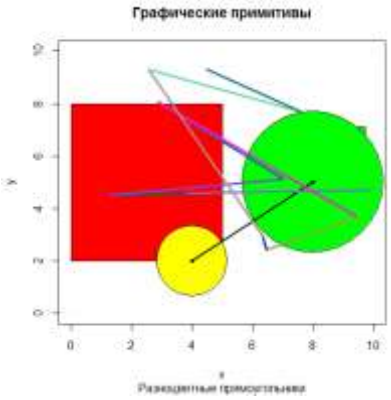
Параметры распределений, параметры графиков и интервалы построения выберите самостоятельно. Рассмотрите не менее двух вариантов значений каждого параметра. Объясните влияние параметров на график плотности. График плотности каждого распределения выполните своим цветом. Сохраните полученные изображения в графических файлах (щелкнув правой кнопкой мыши на изображении и выбрав опцию "Сохранить как растровый файл...") и включите их в отчёт.

Задание 6. Построение графических примитивов

1. Изучите пример построения графических примитивов.

```
# Задалим размеры области
X = 10
Y = 10
# Вызовем функцию plot для построения (пустой) области
# С помощью точек (0, 0) и (X, Y) определим размеры области
# Точки сделаем невидимыми (с помощью назначения pch = ".:" )
plot(x = c(0, X), y  = c(0,Y), main = "Графические примитивы", sub =
"Разноцветные прямоугольники", xlab="x", ylab="y", type = "p", pch = ".:")

# Теперь мы можем строить в этой области другие графические элементы
# Прямоугольник, где левый нижний угол = (0,2), правый верхний = (5, 8)
rect(0,2,5,8, col = "red")
```

```
# Построим ещё один прямоугольник, координаты и цвет которого выбираются
случайно
# Найдём случайные координаты левого нижнего угла
x1 = runif(1, 0, X)
x1
y1 = runif(1, 0, Y)
y1
# Найдём случайные координаты правого верхнего угла
x2 = runif(1, x1, X)
x2
y2 = runif(1, y1, Y)
y2

# Случайным образом сгенерируем 3 числа: значение красной, зелёной и синей
компоненты цвета, соответственно
rr = runif(1, 0, 1) # red
rg = runif(1, 0, 1) # green
rb = runif(1, 0, 1) # blue
rect(x1,y1,x2,y2, col = rgb(rr, rg, rb))

# Построим 2 круга - зелёного и жёлтого цвета
# Зададим координаты центров кругов
xc1 = 4
yc1 = 2
xc2 = 8
yc2 = 5
# Зададим радиусы кругов
r1 = 1
r2 = 2
symbols(x = c(xc1, xc2), y = c(yc1, yc2), circles=c(r1, r2), bg =
c("yellow","green"), add = TRUE)

# Отметим центры кругов точками
points(x = c(xc1, xc2), y = c(yc1,yc2), type = "p", pch = 20)

# Соединим центры кругов отрезком толщиной 2px
segments(xc1, yc1, xc2, yc2, lwd = 2)

# Построим несколько разноцветных отрезков
M= 10 # число отрезков
# Найдём координаты концов отрезков
xx = runif(M+1,0,X)
yy = runif(M+1,0,Y)

for(i in 1:M)
 {
  rr = runif(1, 0, 1) # red
  rg = runif(1, 0, 1) # green
  rb = runif(1, 0, 1) # blue
  segments(xx[i],yy[i],xx[i+1],yy[i+1],lwd = 3,col=rgb(rr,rg,rb))
 }
```



2. Создайте "абстрактную картину" с помощью различных графических примитивов.

Контрольные вопросы

1. Принцип построения диаграмм в языке R.
2. Функции изменения графических параметров диаграмм
3. Команды Добавление текста,
4. Команды настройки параметров осей
5. Команды добавления условных обозначений.

Функции построения графических примитивов