**Министерство науки и высшего образования Российской Федерации**

**ФГАОУ ВО «Севастопольский государственный университет»**

**Институт информационных технологий**

**Лабораторная работа №1**

**«Среда разработки R: визуализация данных»**

по дисциплине «Интеллектуальный анализ данных (методы Data Mining)»

для студентов всех форм обучения направления подготовки

09.04.02 «Информационные системы и технологии

Севастополь

2024

**Среда разработки R: визуализация данных.** Методические указания к лабораторным занятиям по дисциплине «Интеллектуальный анализ данных» / Сост.: И.П. Шумейко, О.А. Сырых – Севастополь: Изд-во СевГУ, 2024 – 16 с.

Методические указания предназначены для проведения лабораторных работ по дисциплине «Интеллектуальный анализ данных». Целью методических указаний является помощь студентам в изучении возможностей системы RStudio. Излагаются практические сведения необходимые для выполнения лабораторной работы, требования к содержанию отчета.

Лабораторная работа № 1  
Среда разработки R: визуализация данных.

**Цель:**

* исследовать возможности языка R для визуализации данных в R;

**Время:** 4 часа

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

**Краткие теоретические сведения**

**Простейшие графические построения в R.**

**Базовая графика в R**

В R много графических пакетов. В базовой комплектации за построение графики отвечает пакет graphics.

Функции, определенные в нем разбиты на три уровня:

* функции высокого уровня;
* функции низкого уровня;
* интерактивные функции.

К функциям высокого уровня относятся функции:

* par() – создание графического окна с заданными параметрами
* barplot() – построение мозаичной диаграммы (статистика);
* boxplot() – построение boxplot– особой диаграммы для статистического анализа данных относительно предполагаемого распределения(статистика);
* boxplot.matrix() – аналогично boxplot;
* bxp() – аналогично boxplot;
* contour() –построение графика с контурными линиями (линиями уровня);
* curve() – построение линии (кривой);
* dotchart() – точечные диаграммы (статистика);
* frame() – задание нового графического окна;
* plot.new() – задание нового графического окна;
* hist() – построение гистограммы (статистика);
* image() – построение цветной прямоугольной сетки согласно заданным цветам;
* matplot() – изображение элементов столбцов одной матрицы относительно элементов соответствующих столбцов другой матрицы;
* mosaicplot() – построение мозаичной диаграммы (статистика);
* persp() – построение трёхмерных графиков;
* pie() – построение круговой диаграммы;
* plot()– основная функция построения двумерных графиков;
* plot.data.frame() – графический анализ таблиц данных;
* plot.default() – построение диаграммы рассеивания;
* plot.factor() – построение диаграммы рассеивания для факторов;
* plot.formula()– построение диаграммы рассеивания с помощью структуры formula;
* plot.histogram()– построение гистограммы (статистика);
* plot.table()– графический анализ таблицы данных (построение мозаичных диаграмм);
* screen– управление графическим окном;
* stem()– построение древесной диаграммы (статистика);
* stripchart()– построение диаграммы рассеивания (аналог boxplot())

Основной функцией для построения двумерных графиков является plot(x, y, …). Аргументы функции:

x – координаты точки на графике.

y – координаты по оси у, если заданы соответствующие значения по оси х.

... – дополнительные аргументы. К ним относятся:

− type – символьный аргумент, определяющий тип построения графика. Возможные значения: «p»– точки (круги), «l» – линии, «b» – линии и точки, «c»– строятся только линии из «b», «o»– точки и линии пересекаются, «h» – гистограммо-подобные вертикальные линии, «s»–ступенчатая линия, «S» – другая ступенчатая линия, «n» – ничего не строится (но определяются области по осям).

− log – символьный аргумент — задаётся название оси («x», «y» или «xy»), координаты по которой переводятся в логарифмическую шкалу

− lty – стиль линии (должен быть задан параметр type= «l»). Возможные значения: «solid» (1) – сплошная линия; «blank» (0) – отсутствует линия; «dashed» (2) – штриховая линия; «dotdash» (4) – штрих-пунктир; «dotted» (3) – пунктирная линия; «longdash» (5) – длинный штрих; «twodash» (6) –двойной штрих.

− xlab, ylab, main и sub–символьные переменные – названия осей, основного и дополнительного заголовков графика.

− col – символьный или числовой аргумент – цвет графика.

− bg – символьная переменная, отвечающая за цвет фона графического окна. По умолчанию цвет белый – bg= «white».

− col.axis – символьная переменная – цвет осей. По умолчанию чёрный – «black».

− col.main – символьная переменная – цвет основного заголовка. По умолчанию чёрный –«black».

− col.sub – символьная переменная – цвет дополнительного заголовка. По умолчанию цвет чёрный –«black».

− fg – символьная переменная – цвет рамки вокруг графика.

− font.axis, font.lab, font.main и font.sub– числовые аргументы – тип шрифта для названия осей, основного и дополнительного заголовков. Возможные значения: 1 – простой (по умолчанию), 2– жирный; 3 – курсив и 4 – жирный курсив.

− lwd – положительное целое число – толщина линии; по умолчанию lwd=1.

− xlim – числовой вектор с(x1, x2) – пределы по оси x.

− ylim – числовой вектор с(y1, y2) – пределы по оси y.

− frame – логический аргумент – нужно ли строить рамку вокруг графика.

Функция curve() строит график заданной функции на некотором интервале. Её синтаксис:

curve(expr, from = NULL, to = NULL, n = 101, add =FALSE, type = "l", ylab = NULL, log = NULL, xlim =NULL, ...)

Аргументы этой функции:

expr − выражение как функция от х или имя функции, график которой нужно построить;

from и to − числовые аргументы − границы отрезка, на котором строится график. Если не заданы эти параметры, то границы определяются значением аргумента x заданного выражения. Если же и x не определён, то график строится на отрезке [0;1];

n − числовой аргумент − число точек, в которых определяется значение expr;

add − логический аргумент − нужно ли добавлять строящийся график к уже существующему (add=TRUE) или нужно создать новое графическое окно (add=FALSE).

**Управление графическим окном**

Управление графическим окном (разбиение на несколько подокон) можно осуществлять не только при помощи функции par() с её аргументами mfrow и mfcol, так и при помощи следующих функций:

− split.screen(figs, screen, erase = TRUE) – графическое окно разбивается на заданное число окон, каждое из которых рассматривается как отдельное графическое окно; полезно при построении большого количества графиков в одном окне.

− screen(n = , new = TRUE) – выбор подокна, в котором будет построен график.

− erase.screen(n =) – очистка определённого подокна (закрашивается цветом фона)

− close.screen(n, all.screens = FALSE) – удаление всех выбранных подокон.

Аргументы этих функций:

− figs – числовой вектор, состоящий из двух элементов – число подокон по строкам и по столбцам. Например, аргумент figs=c(2,3) означает, что графическое окно разбивается на 2 строки, в каждой из которых 3 подокна.

− screen – номер разбиваемого подокна.

− erase – логический аргумент, отвечающий нужно ли очищать данное окно.

− n – номер окна, в котором строится график (функция screen(n)), либо которое очищается (erase.screen()) или закрывается (close.screen()).

− new – логический аргумент; при значении TRUE графическое окно будет очищено перед построением нового графика.

− all.screens– логический аргумент. Он указывает на закрытие всех окон.

**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=).



Рисунк 1. Символы, назначаемые при помощи параметра pch.

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

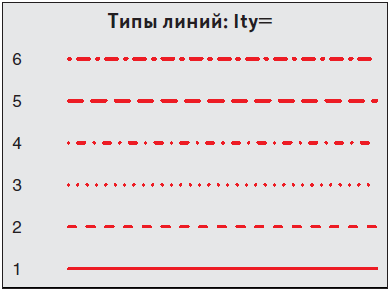


Рисунок 2. Типы линий, назначаемые при помощи параметра lty.

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.

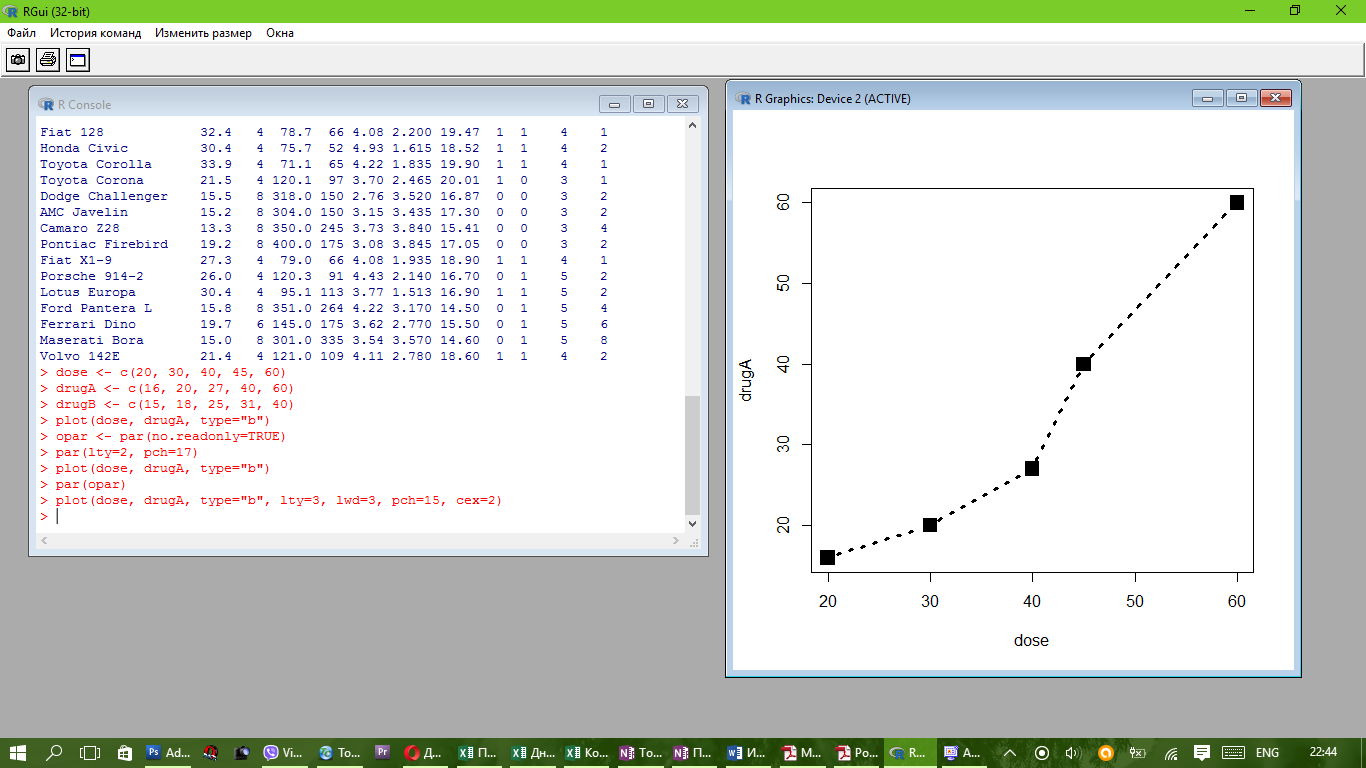


Рисунок 3.График зависимости реакции пациента от дозы лекарства A с измененными типом и шириной линии, а также типом и размером символов.

**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.

**Примеры использования простейшей графики в R**.

1. Использование функции plot(x, y,…) для построения графика функции y=cos(2x), с разным значением параметра type (рис. 4)
2. Создать вектор (переменную) х, на промежутке [-2π; 2π]с шагом 0,1.
3. Вычислить значение функции y=cos(2x).
4. Построить график функции y=cos(2x), используя разные параметры функции plot(x, y,…). Пример построения представлен на рисунке 4.

x<-seq(-2\*pi,2\*pi,0.1)

y=cos(2\*x)

plot(x,y,type="…")

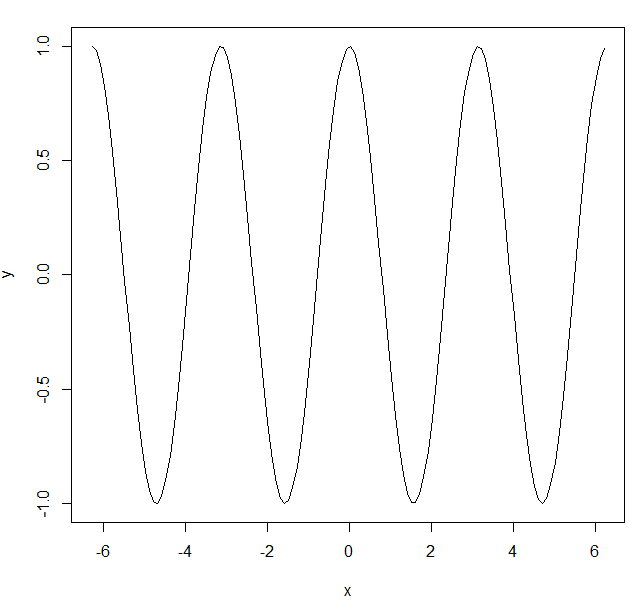
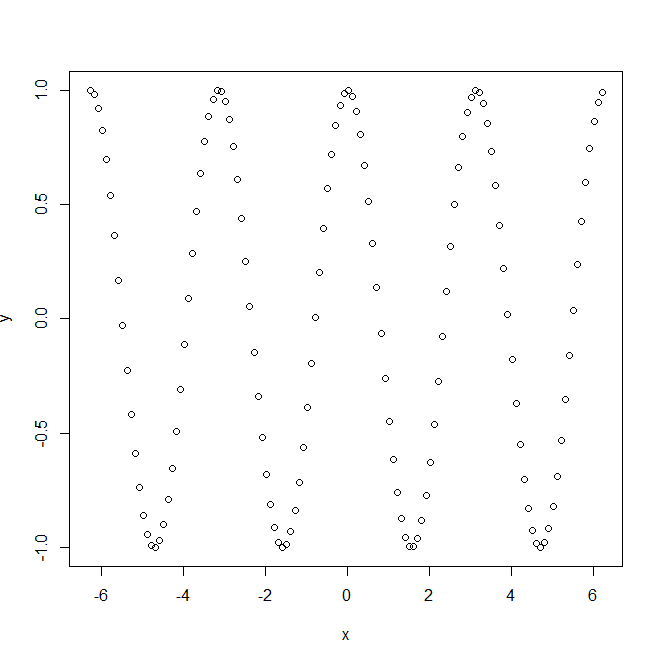
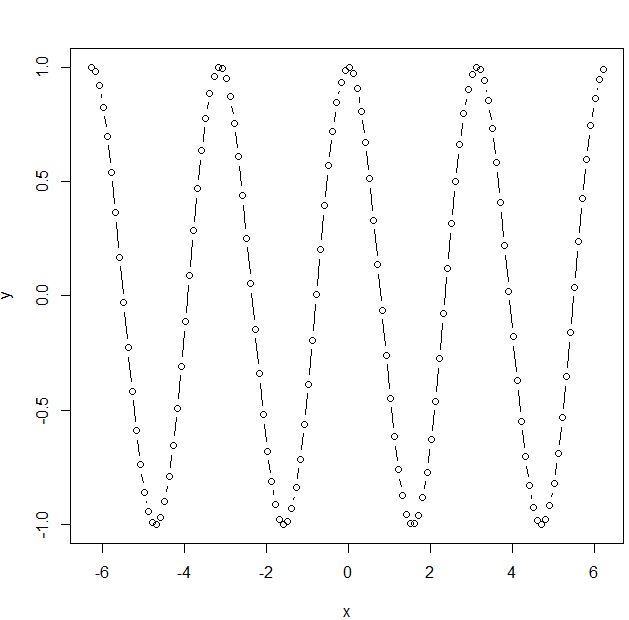


Рисунок 4 – График функции y=cos(2x)

1. Использование функции pie() для построения круговых диаграмм.
2. Создать переменную Salary, содержащую значения зарплат в тыс.руб.
3. Присвоить имена сотрудников соответствующим значениям зарплат.
4. Создать круговую диаграмму
5. Изменить ее параметры. Пример построения диаграммы представлен на рисунке 5.

Salary <- c (21, 19, 27, 11, 102, 25, 21)

names(salary) <- c ("Коля", "Женя", "Петя", "Саша", "Катя", "Вася", "Жора")

pie(salary)

pie(salary, col=3:9)

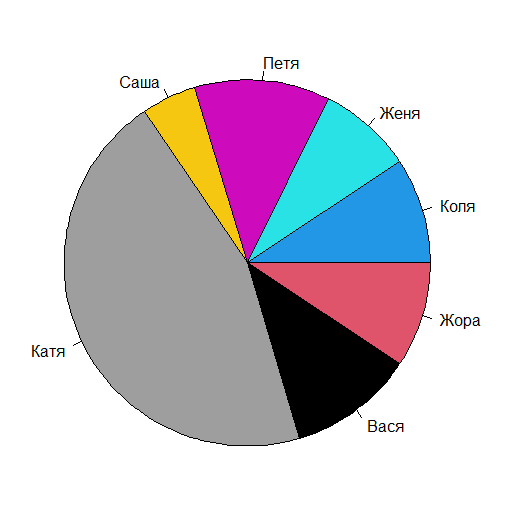


Рисунок 5 – Круговая диаграмма

1. Разбиение графического окна с использованием функции par().
2. Создать новое графическое окно.
3. Присвоить вектору х значения от -π до π с шагом 0,1
4. Присвоить переменной y значения функции sin2x
5. Построить в первом окне график функции y= sin2x, цвет линии – коричневый, подпись по оси х – «sin 2x»
6. Во втором окне построить график функции y= sinx, цвет линии – синий, подпись по оси х – «sin x». Пример построения представлен на рисунке 6.

par(mfrow=c(2,1))

x=seq(-pi,pi,by=0.1)

y=sin(2\*x)

plot(x,y,type="l",col="brown",sub="sin 2x")

plot(sin,-pi,pi,type="l",col="blue",sub="sin x")

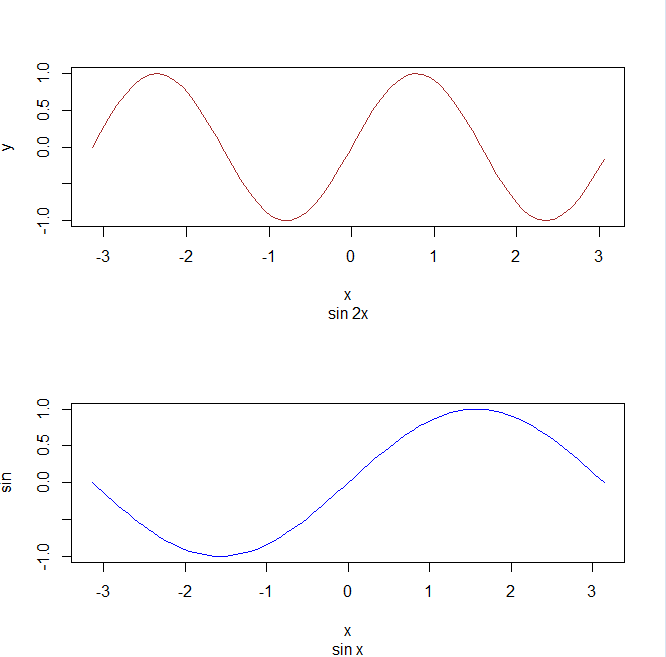


Рисунок 6 – Разбиение графического окна функцией par()

1. Построение столбчатых диаграмм.

Для категориальных данных, принимающих мало различных значений, используют столбчатые диаграммы, которые строят несколько столбцов с высотами, соответствующими количеству наблюдений.

1. Создать вектор имен командой salary.n <- c("Коля", "Женя", "Петя", "Саша", "Катя", "Вася", "Жора").
2. Векторизовать вектор имен при помощи функции factor
3. Построить диаграмму при помощи команды plot. Пример построения диаграммы представлен на рисунке 7.

salary.n<- c("Коля", "Женя", "Петя", "Саша", "Катя", "Вася", "Жора")

salary.o<-factor(salary.n)

plot(salary.o,salary,type="h")

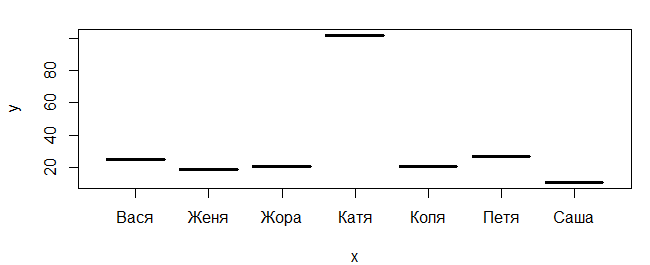


Рисунок 7 – Столбчатая диаграмма

1. Построение диаграмм при помощи функции barplot() с использованием аргументов а xlab, ylab, main, col.main и font.main.

С помощью команды barplot() построить диаграмму «Зарплата сотрудников», которая будет содержать подписи осей и название диаграммы.

Результат построения приведены на рисунке 8.

barplot(salary, col="blue", xlab="Имена сотрудников", ylab="Зарплата", main="Зарплата сотрудников",col.main="tomato1", font.main=3)

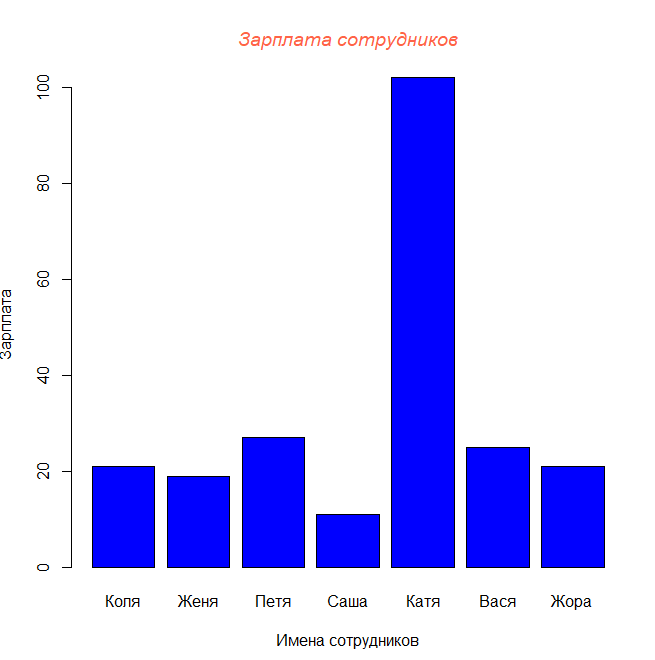


Рисунок 8 – Столбчатая диаграмма, построенная командой barplot()

1. Построение диаграмм при помощи функции hist().

Построить диаграмму «Распределение зарплаты» используя функцию hist(). Цвет диаграммы – желтый, подпись по оси y – «Количество людей». Результат построения приведен на рисунке 9.

hist(salary,col="yellow",main="Распределение зарплаты", ylab="количество людей")

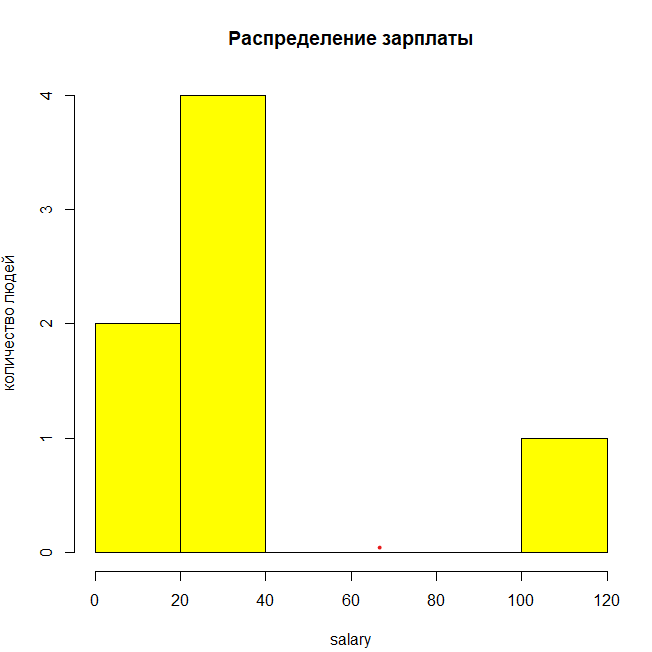


Рисунок 9 – Гистограмма

1. Построение трехмерного графика при помощи функции persp().
2. Создать вектор значений переменной x от -10 до 10 с шагом 0,5.
3. Создать вектор значений переменной y=х.
4. Ввести функцию, чей график будем строить командой f <- function(x, y) { r<- sqrt(x^2+y^2); 10 \* sin(r)/r}.
5. Командой z<-outer(x, y, f) создать матрицу, которая ставит в соответствие переменной z значения переменных x и y и функцию f(x,y).
6. Командой op<- par(bg = "white") создать графическое окно с белым фоном.
7. Построить трехмерное изображение функции z=f(x,y) командой persp(x,y,z,theta=30,phi=30,expand=0.5,col="lightblue"). Результат представлен на рисунке 10.
8. Изменяя значения аргументов theta, phi и expand, определите как изменение их значений влияет на график изображение

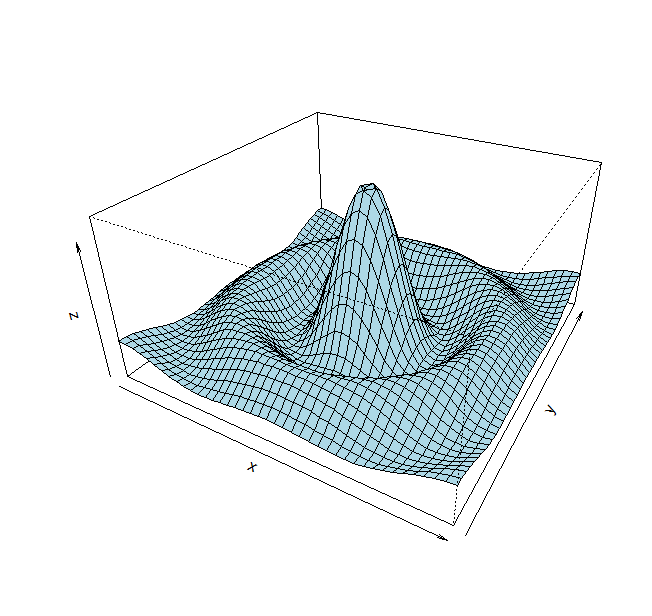


Рисунок 10 – Трехмерный график

1. Разбиение графического окна на части командой split.screen ().
2. Разбить графическое окно на четыре части.
3. Переместиться в верхнее левое окно и в нем построить график функции sin x.
4. В нижнем левом окне построить график функции arctg x.
5. В правом нижнем – график функции ex.
6. В правом верхнем – график функции cos x.

Результат построения представлен на рисунке 11

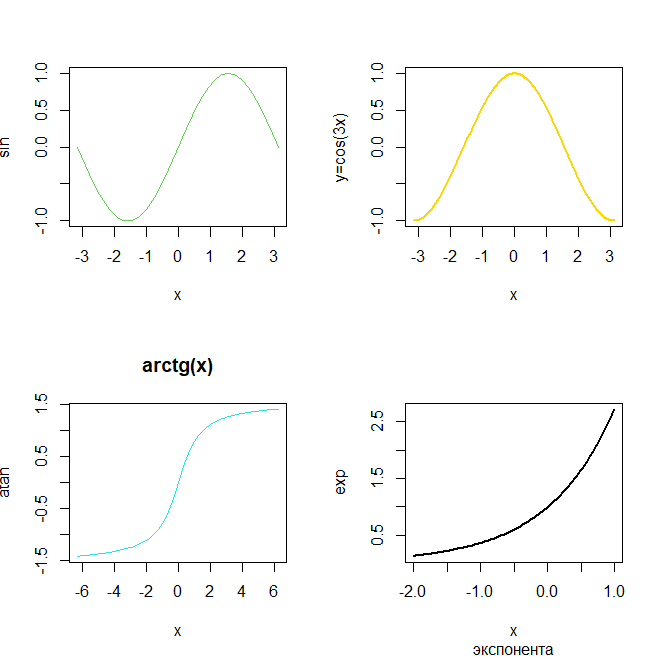


Рисунок 11 – Использование функции screen()

1. Построение нескольких графиков в одном графическом окне.
2. Определить значения переменной x = (-1,0, 1,1.5,2).
3. Определить значения переменной y, полученные экспериментально y=(0.8,0.1,1.05,2.3,3.8).
4. Вычислить значения функции y=x2.
5. Построим график функции по экспериментальным данным, увеличив толщину линий кругов до двух.
6. Командой par(new='TRUE') разрешить построение в графическом окне еще один график.
7. Построить график функции y=x2 сплошной линией синего цвета толщиной, равной двум,в этом же графическом окне.

Результаты выполнения построений приведены на рисунке 12.

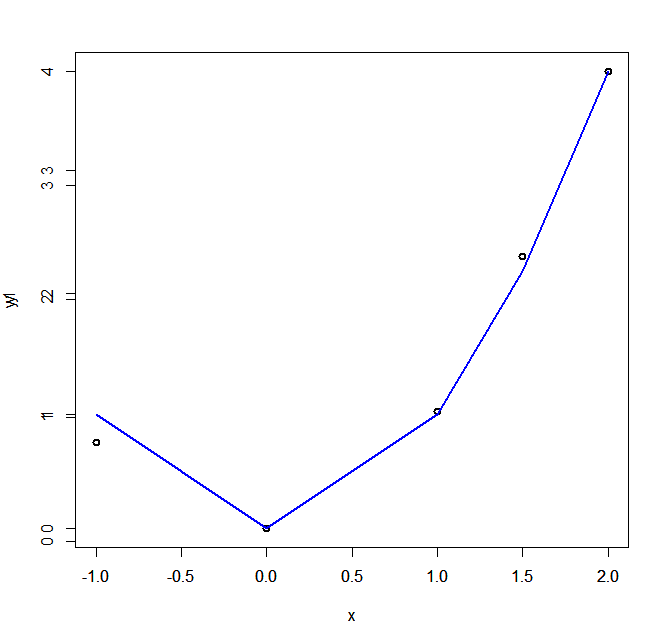


Рисунок 12 – Построение двух графиков в одном окне.

Выполнить предыдущее упражнение, используя для построения графика функции y=x2 функцию curve().

Результат представлен на рисунке 13.

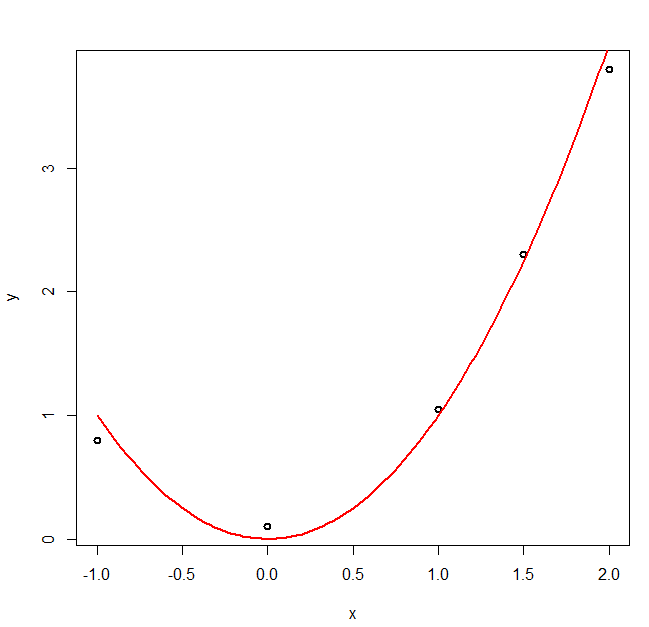


Рисунок 13 – Графики y(x) и x2

1. Построение графики с помощью пакета ggplot2

Диаграмма размаха (boxplot), также известная как ящик с усами. Этот график упрощает данные, оставляя только несколько параметров: медиану (такое число, что половина данных меньше его, а вторая половина больше), верхнюю и нижние квартили (т.е. такие числа, что четверть данных больше верхней квартили и четверть меньше нижней), существенный минимум и максимум (наименьшее и наибольшее значения, не считая выбросов), а также выбросы, которые существенно меньше или больше остальных чисел в ряду. Обычно этот способ применяется, если необходимо визуально сравнить несколько категорий данных.

В R этот график реализован с помощью обычной функции boxplot, или с использованием функции qplot пакета ggplot2 (рисунок 14). Базовая форма вызова функции

qplot(gear, mpg, data=mtcars, geom=’boxplot’, fill=factor(gear)),

где data – массив из которого берутся данные,

gear – переменная отвечающая за категорию данных,

mpg – переменная отвечающая за сами данные,

fill – раскрашивает диаграммы в соответствии со значением переменной gear.

Здесь к gear применили функцию factor, которая из числовой переменной сделала категориальную.

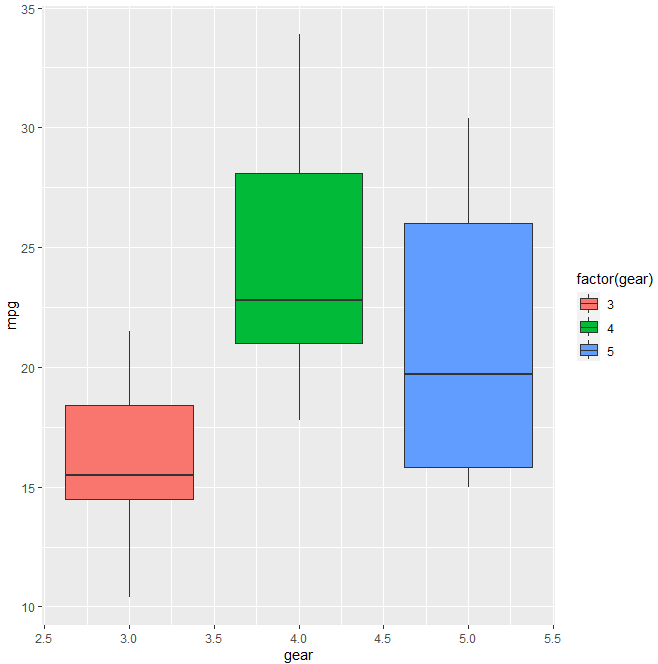


Рисунок 14 – Диаграмма размаха

Столбчатая диаграмма (Bar plot)

С использованием пакета ggplot2 столбчатую диаграмму можно построить, например, командой

qplot(gear, data=mtcars, geom=’bar’, fill=I(‘red’))

Результат построения представлен на рисунке 15.

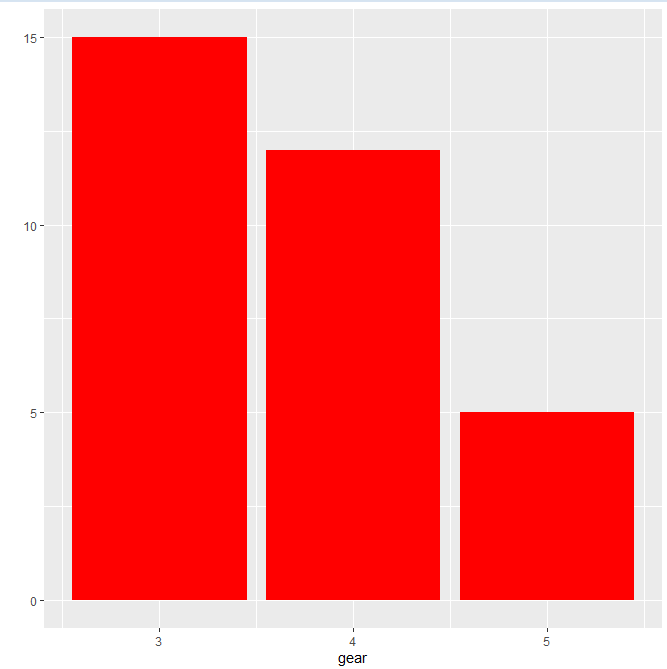


Рисунок 15 – Столбчатая диаграмма

Для параметра fill можно использовать не постоянное значение ”red”, а вектор значений, например, qplot(gear, data=mtcars, geom=’bar’, fill=factor(cyl)) (рис 16).

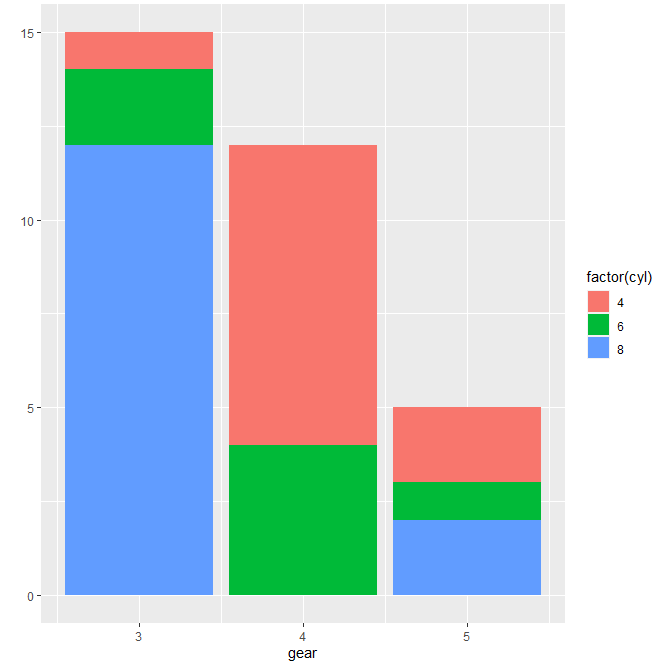


Рисунок 16 – Столбчатая диаграмма

Гистограмма (histogram). Гистограмма разбивает оси на интервалы, заменяет значения переменных на номер интервала попадания и строит по полученным данным столбчатую диаграмму.

Гистограмма оценивает плотность распределения соответствующей случайной величины. Высокие столбцы соотвествуют частой встречаемости значения, низкие – редкой.

С использованием пакета ggplot2 гистограмму можно построит, например, командой qplot(mpg, data=mtcars, geom=’histogram’, fill=I(‘red’)) (рис17).

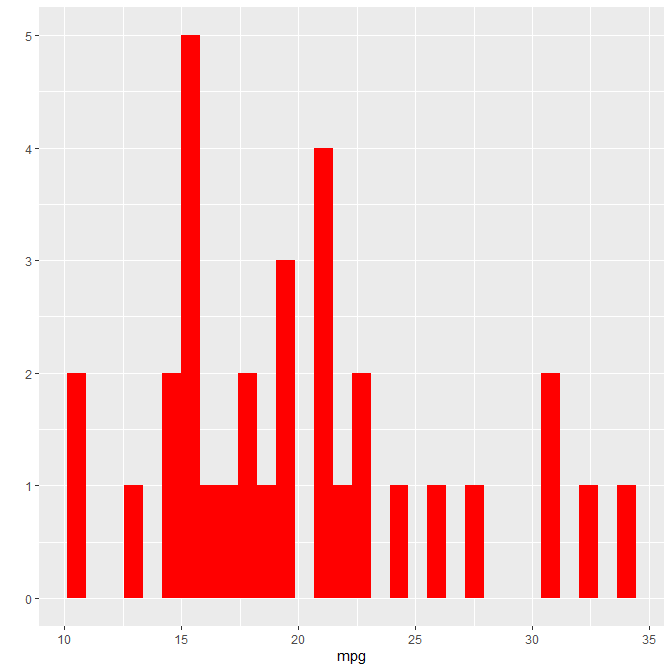


Рисунок 17 – Гистограмма

На рисунке 18 построена диаграмма с изменением параметров, например, выбраны ширина интервалов и цвет границы:

qplot(mpg, data=mtcars,geom=”histogram”,fill=I(‘red’),binwidth = 2,col=I(‘blue))

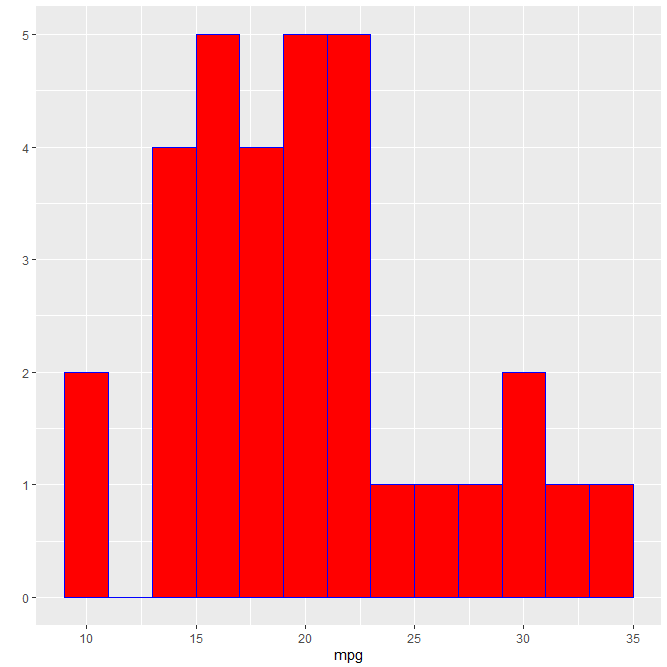


Рисунок 18 – Гистограмма

Ядерная оценка плотности (density plot). Гистограмма дает дискретизированное приближение функции плотности, между тем можно получить множество гладких оценок для этой функции. Популярным методом являются так называемые ядерные оценки (рисунок 19).

В R такой график можно построить с помощью

qplot(mpg, data=mtcars,geom=”density”,fill=factor(cyl), alpha=0.5),

где параметр alpha задает прозрачность.

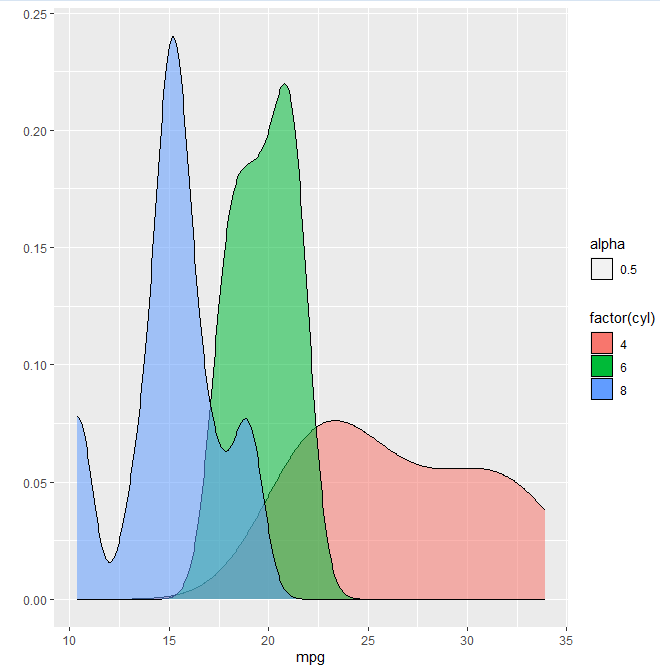


Рисунок 19 – Ядерная оценка плотности

Скрипичные графики (violin plot).

Если много графиков, то сравнивать их плотности очень тяжело – картинка получается перегруженной. В этом случае обычно используют box plot. Но существует также некоторый компромисс, называемый violin plot.

Violin plot строит вдоль вертикальной оси симметричную фигуру, чей контур соответствует плотности. Это позволяет и увидеть соотношение между несколькими категория данных, и увидеть больше, чем в boxplot (рисунок 20).

В R это реализуется с помощью команды

qplot(gear, mpg, data=mtcars,geom=c(‘violin’),fill=factor(gear))

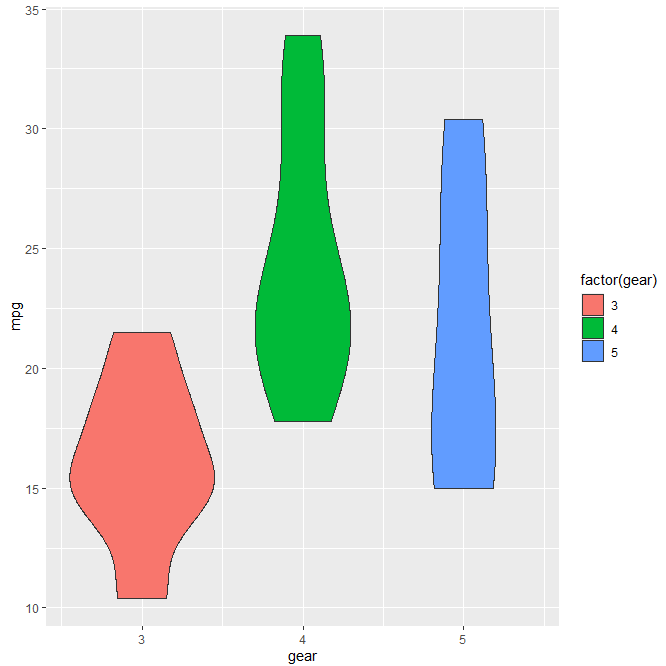


Рисунок 20 – Скрипичные графики

Чтобы понять, как этот график соотносится с диаграммой размаха, можно построить их вместе (рисунок 21)

qplot(gear, mpg, data=mtcars,geom=c(‘violin’,’boxplot’), fill=factor(gear))

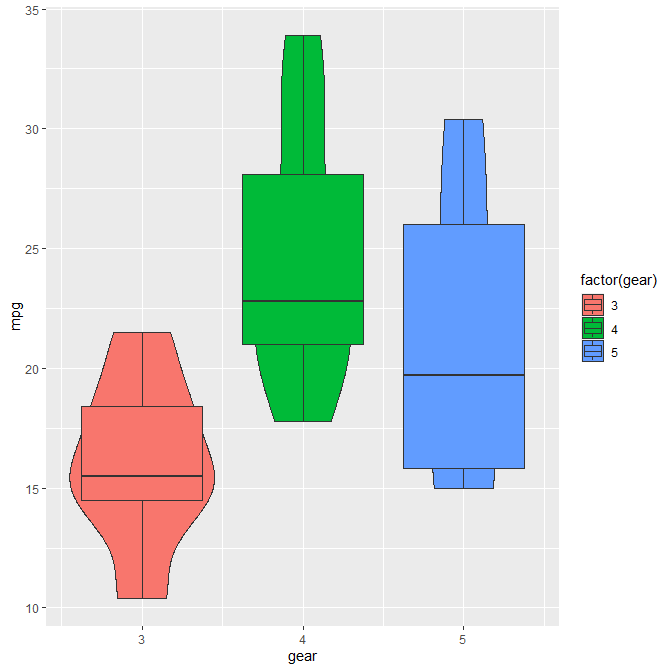


Рисунок 21 – Скрипичные графики с диаграммой размаха

Диаграмма рассеивания (scatterplot)

Scatterplot – это двумерный точечный график (рис 22). В R его можно создать просто функцией plot(x,y) или с использованием пакета ggplot2 командой

qplot(disp, mpg, data=mtcars, geom=”point”,col=factor(gear))

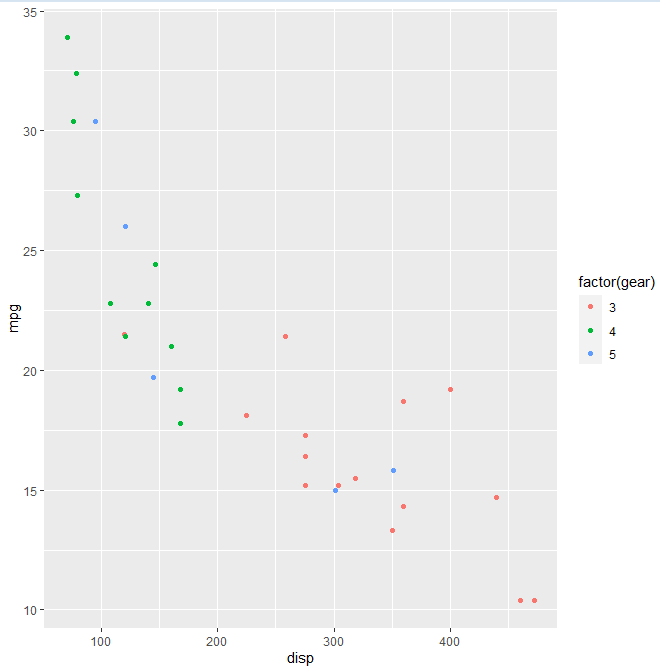


Рисунок 22 – Диаграмма рассеивания

Диаграмма рассеивания с дополнительным выделением (рис 23)

Можно использовать для такой визуализации scatterplot, добавляя третью размерность через цвет или размер

qplot(disp, mpg, data=mtcars, color=factor(hp))

qplot(disp, mpg, data=mtcars,size=factor(hp)).

qplot(disp, mpg, data=mtcars,size=factor(hp),col=factor(gear),shape=factor(am))

Таким образом, можно отобразить до 5 переменных

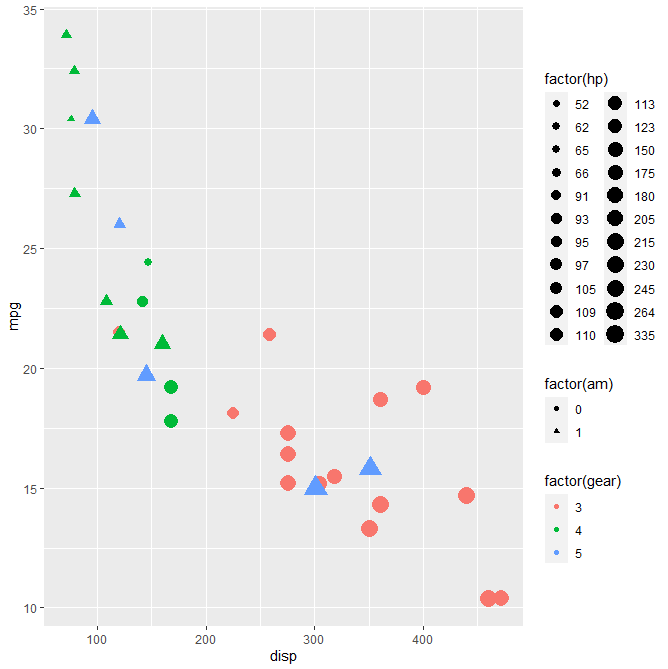
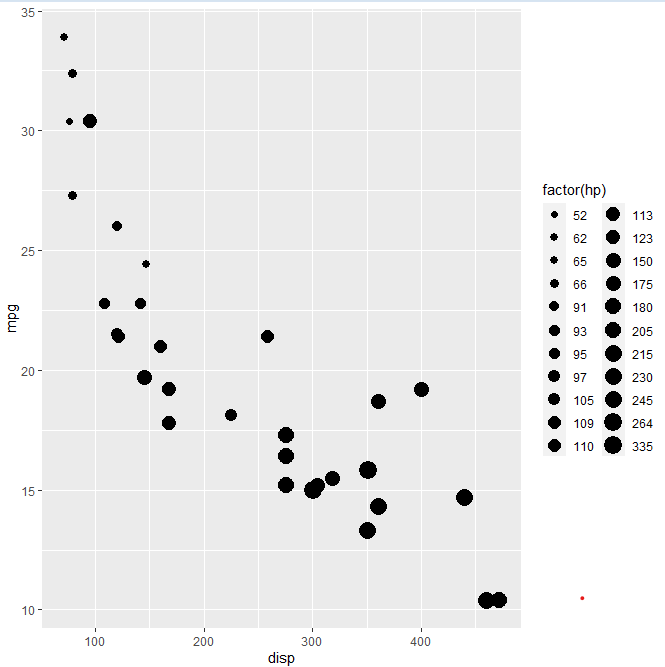
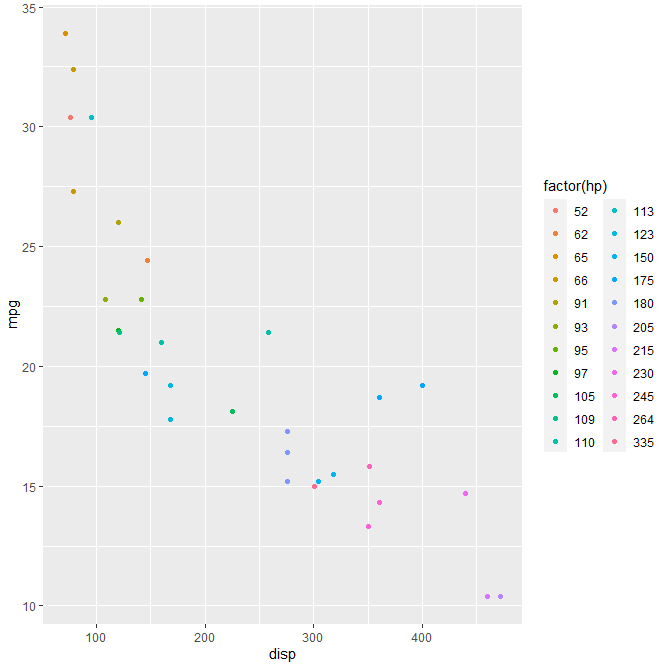


Рисунок 23 – Диаграмма рассеивания с дополнительным выделением

Матрица диаграмм рассеивания (scatterplot matrix)

Матрица диаграм рассеивания – это набор scatterplots, построенный по плоскостям, соответствующим каждой паре переменных (рис 24).

В R она строится с помощью команды

pairs(∼ mpg+disp+drat+wt,data=mtcars,main=’Simple Scatterplot Matrix’,col=mtcars$cyl)

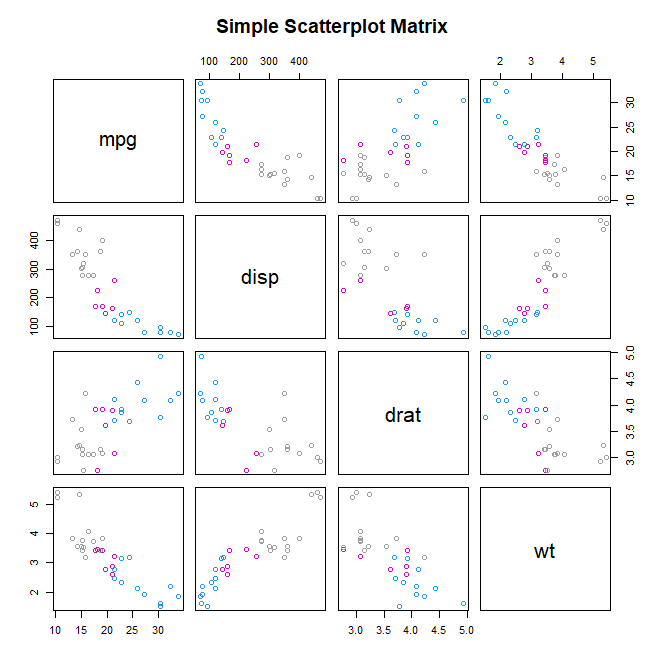


Рисунок 24 – Матрица диаграмм рассеивания

Параллельные координаты

Параллельные координаты предлагаются заменить данные 𝑥1, ..., 𝑥𝑑 на ломаную с вершинами (1, 𝑥1),...,(𝑑, 𝑥𝑑). Если размерность не слишком велика или данные хорошо сгруппированы, то мы сможем визуально отличить качественное отличие нескольких групп таких ломаных

В R реализацию предлагает, например, функция parcoord пакета MASS.

Пример ее использования (рис 25):

parcoord(iris[,c(3,4,2,1)],col=iris[,5])

При этом цвет задается вектором col, а сами данные – массивом iris.

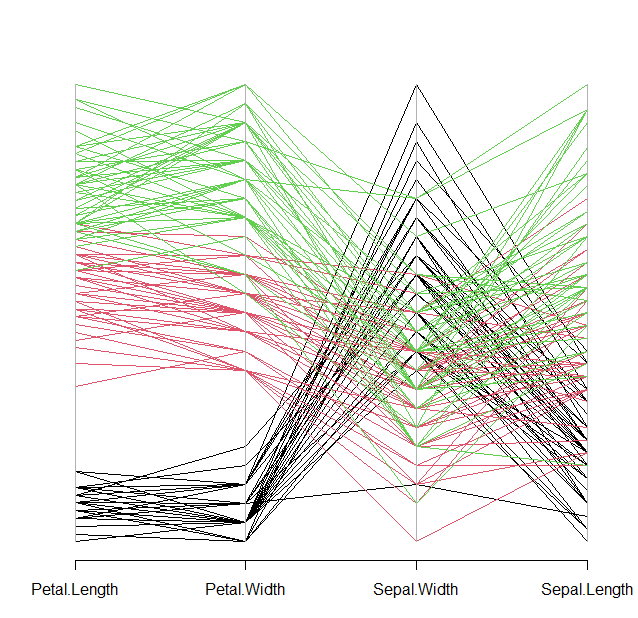


Рисунок 25 – Параллельные координаты

Радарная диаграмма (spider plot, radar chart, star plot, web plot)

В этом виде диаграмм строится аналог параллельных координат, но замкнутый по кругу. Иначе говоря, оси, соответствующие переменным, расположены через равные углы по кругу, каждой точке соответствует многоугольник (рис 26).

В R есть стандартная функция stars, реализующая эту процедуру в виде stars(mtcars[,1:11]).

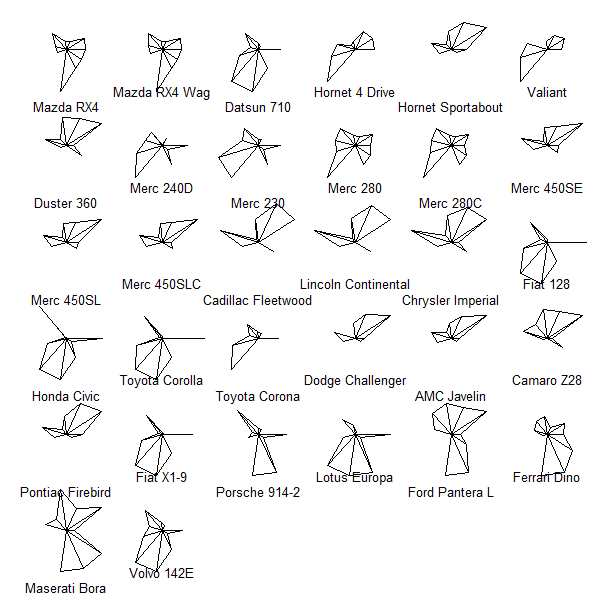


Рисунок 26 – Радарная диаграмма

Лица Чернова

Идея заключается в том, чтобы для каждой точки нарисовать лицо, различные параметры которого соответствуют значениям координат нашей точки.

Реализация этого алгоритма есть в пакете TeachingDemos в функции faces (рис 27)

Применение достаточно просто:

faces(mtcars[,1:10])

В том же пакете есть аналогичная функция faces2 с другой системой кодирования лиц (рис 28)

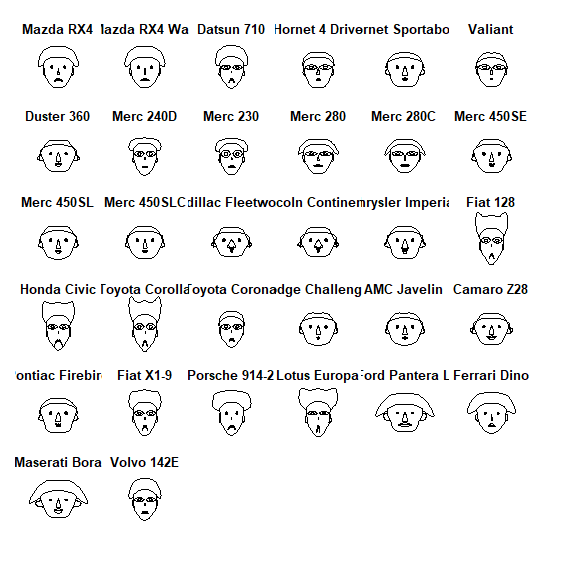


Рисунок 27 – Лица Чернова в функции faces

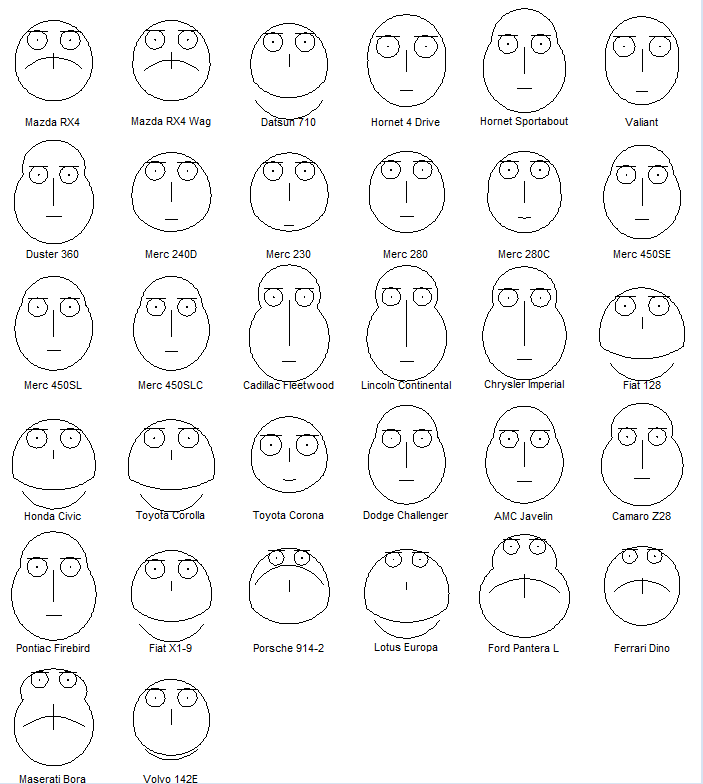


Рисунок 28 – Лица Чернова в функции faces2

**Задания для самостоятельного выполнения**

1. Выполнить все примеры, содержащие в методических рекомендация, изменяя параметры функций.
2. Смоделировать независимые случайные векторы (X,Y) , имеющие гауссовское распределение с заданным математическим ожиданием *a* и корреляционной матрицей *R* (Приложение А). Визуализировать данные на плоскости в виде точек.
3. Используя реальные статистические данные из заданного набора (Приложение Б) визуализировать данные на плоскости, используя 4-5 разных видов графиков.
4. Построить график функции y=f(x), заданной в приложении В.
5. Построить в том же графическом окне график функции y=f(x), заданной таблично в приложении В.
6. Постройте в новом графическом окне графики функций реализации запчастей 1, 2 и 3 групп за 2023 год по месяцам:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Группа** | **Январь** | **Февраль** | **Март** | **Апрель** | **Май** | **Июнь** |
| I группа | 30350,4 | 30350,4 | 94351,8 | 75810,6 | 57860,4 | 104943,6 |
| II группа | 6525 | 6525 | 73198,8 | 14310 | 29550 | 18716,4 |
| III группа | 10230 | 10230 | 69480 | 18874,8 | 43176,6 | 70900,2 |
| **Группа** | **Июль** | **Август** | **Сентябрь** | **Октябрь** | **Ноябрь** | **Декабрь** |
| I группа | 80710,5 | 91177,2 | 113397,6 | 89148 | 60709,2 | 49272,3 |
| II группа | 26282,4 | 39085,8 | 96808,8 | 215320,8 | 73904,4 | 75894 |
| III группа | 50358 | 77707,2 | 87267 | 86396,4 | 24232,8 | 17842,8 |

**Приложение А**

Варианты задания для моделирования данных в задании 2.

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | *n* | *a* | *R* |
| 1 | 100 | (-1,-1) |  |
| 2 | 1000 | (-1,0) |  |
| 3 | 10000 | (-1,1) |  |
| 4 | 100 | (0,-1) |  |
| 5 | 1000 | (0,1) |  |
| 6 | 10000 | (1,-1) |  |

**Приложение Б**

Варианты реальных наборов данных для задания 3

1. Ирисы Фишера

Ссылка: <https://archive.ics.uci.edu/ml/datasets/Iris>

Первый признак: Sepal.Length (столбец № 2)

Второй признак: Sepal.Width (столбец № 3)

1. Wine

Ссылка: <http://archive.ics.uci.edu/ml/datasets/Wine>

Первый признак: alcohol (столбец № 2)

Второй признак: color-intensity (столбец № 11)

1. Forest Fires

Ссылка: <http://archive.ics.uci.edu/ml/datasets/Forest+Fires>

Первый признак: ISI (столбец № 8)

Второй признак: wind (столбец № 11)

1. 1985 Auto Imports Database

Ссылка: <http://archive.ics.uci.edu/ml/datasets/Automobile>

Первый признак: length (столбец № 11)

Второй признак: width (столбец № 12)

1. Glass Identification Database

Ссылка: <http://archive.ics.uci.edu/ml/datasets/Glass+Identification>

Первый признак: RI (столбец № 2)

Второй признак: Al (столбец № 5)

1. Auto-Mpg Data

Ссылка: <http://archive.ics.uci.edu/ml/datasets/Auto+MPG>

Первый признак: mpg (столбец № 1)

Второй признак: horsepower (столбец № 4)

**Приложение В**

Варианты функций к заданию 4

1. *y=x2 – 4x + 3*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | -1 | -0,5 | 0 | 0,5 | 1 |
| y | 9 | 5,5 | 3 | 1,5 | 1 |

1. *y=2x2 – 4x + 1*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | -1 | 0 | 1 | 2 | 3 |
| y | 7 | 1 | -1 | 1 | 7 |

1. *y=x2 – 5x – 6*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | 1 | 3 | 5 | 7 | 9 |
| y | -10 | -12 | -6 | 8 | 30 |

1. *y=x2 – 3x – 3*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | 1 | 3 | 5 | 7 | 9 |
| y | -10 | -12 | -6 | 8 | 30 |

1. *y=x2 – 3x – 1*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | 0 | 0,5 | 2 | 4,5 | 8 |
| y | -1 | -2,25 | -3 | 5,75 | 39 |

1. *y=x2 – 2x + 2*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | -1 | 0 | 2 | 5 | 9 |
| y | 5 | 2 | 2 | 17 | 65 |

**Библиография**

1. Алексей Шипунов и др. Наглядная статистика. Используем R! – М.: ДМК Пресс, 2014. – 298 с. [Электронный ресурс]. Режим доступа: <http://ashipunov.info/shipunov/school/books/rbook.pdf>.
2. Зарядов И.С. Введение в статистический пакет R: типы переменных, структуры данных, чтение и запись информации, графика. М.: Издательство Российского университета дружбы народов, 2010. – 207 с.
3. Роберт И. Кабаков R в действии. Анализ и визуализация данных в программе R / пер. с англ. Полины А. Волковой. – М.: ДМК Пресс, 2014. – 588 с.: ил.
4. Оффициальный сайт RStudio. Режим доступа: <https://www.rstudio.com>.
5. Профессиональный информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и интеллектуальному анализу данных [Электронный ресурс]. Режим доступа: <http://machinelearning.ru>.
6. Мастицкий С.Э., Шитиков В.К. (2014) Статистический анализ и визуализация данных с помощью R. – Электронная книга. Режим доступа: http://r-analytics.blogspot.com