Федеральное государственное образовательное бюджетное учреждение
высшего профессионального образования
«ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»
(Финансовый университет)

Департамент анализа данных, принятия решений
и финансовых технологий

Дисциплина «Программирование в среде R»

П.Б. Лукьянов

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ
ЛАБОРАТОРНОЙ РАБОТЫ № 5

Использование графики

Для студентов, обучающихся по направлению подготовки
«Прикладная информатика»
(программа подготовки бакалавра)

Москва 2021

Цель методических указаний – изучение графических возможностей среды R и способов построения графиков для наглядного представления результатов расчетов. При рассмотрении графических функций изучаются их параметры, а также особенности вызова основных и вспомогательных графических функций. В итоге информация на графиках обладает нужной информативностью и точно отображает зависимости исследуемых показателей.

Необходимый теоретический минимум

При необходимости выполнить графический анализ результатов у аналитика есть несколько возможностей (см. рис. 1):

- Использовать базовые графические функции R
- Подключить дополнительные специализированные пакеты и вызывать графические функции оттуда
- Разработать необходимые графические функции самостоятельно (рис. 2)

Основной подход к использованию в графических функций языка R такой же, как и при использовании любых других функций, встроенных в R или написанных самостоятельно – вся необходимая для работы информация должна передаваться через параметры, а в качестве параметра может быть вектор, матрица, список, таблица или массив.

Прежде чем какие-либо данные будут отображены на графике, сами эти данные должны быть сформированы в результате вычислений или обработки других, первичных данных, полученных в экспериментах или исследованиях (рис. 3).

Далее будут рассматриваться графические функции из стандартного дистрибутива R. Эти функции можно разбить на две категории: основные и вспомогательные. К основным функциям относятся функции, вызов которых приводит к рисованию того или иного графика. Названия некоторых основных функций приведены на рис. 3 справа.

Рис. 1. Варианты использования графических функций R

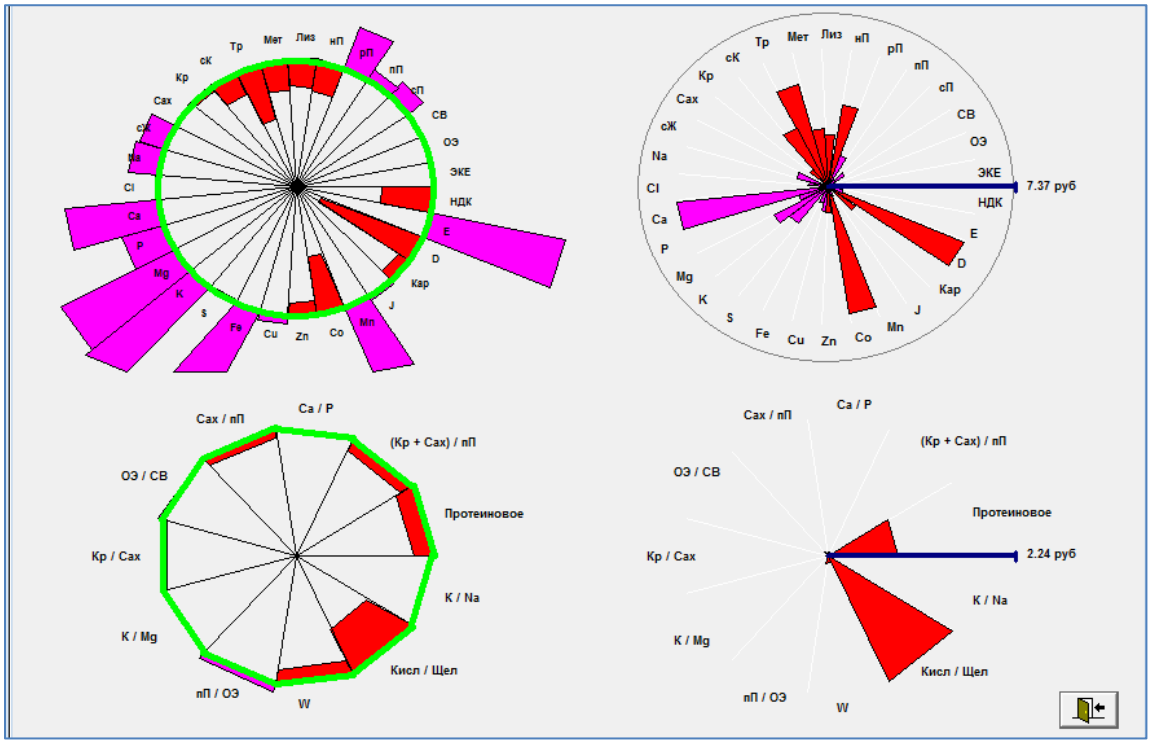

Рис. 2. Примеры графических функций, написанных Пользователем

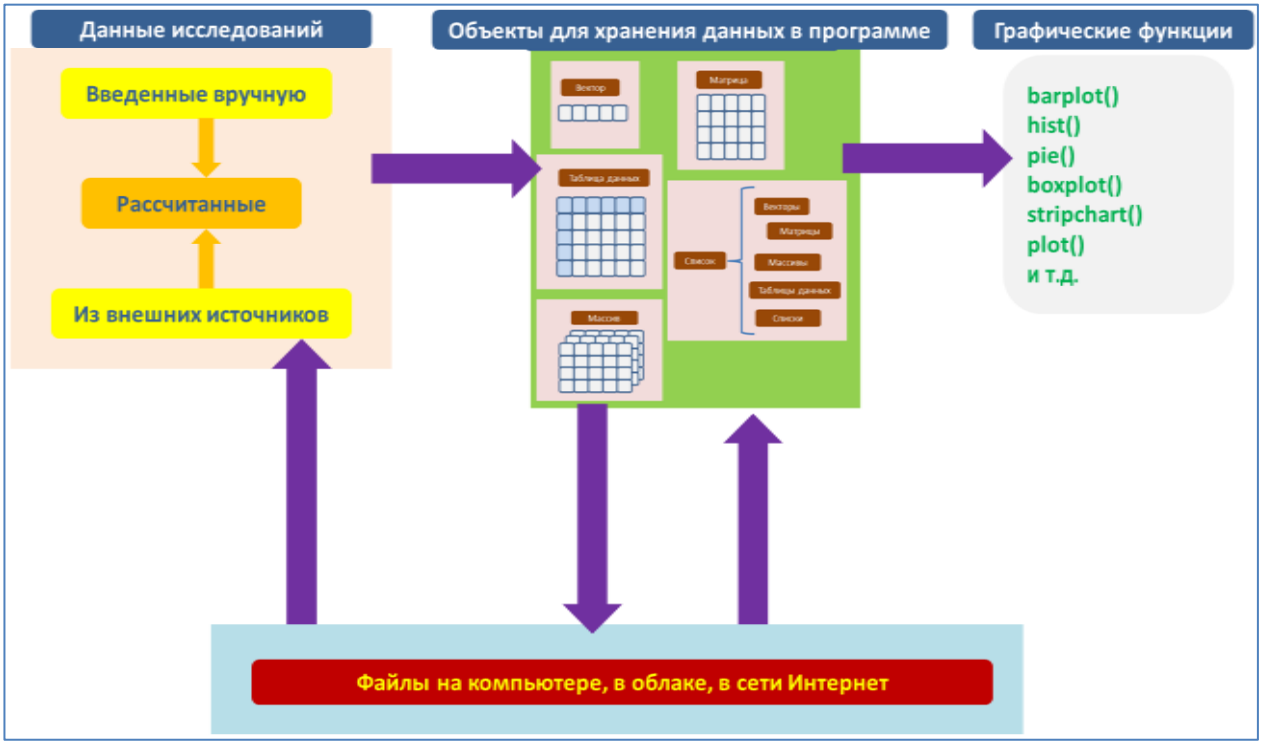

Рис. 3. Общая схема передачи данных в графические функции

Кроме основных функций существует определенное количество вспомогательных, которые предназначены для того, чтобы график, созданный ранее с помощью основной графической функции, дополнить новой информацией.

Использование вспомогательных функций существенно расширяет возможности основных функций, позволяя на уже нарисованную зависимость y1(x) накладывать зависимости y2(x), y3(x), …, yN(x), менять и добавлять координатные оси, разметку, поясняющие надписи и т.д. Связь основных и вспомогательных графических функций представлена на рис. 4.

Итак, вся информация для рисования передается через параметры графических функций (см. рис. 5). Передаются

- собственно данные
- всевозможные настройки для отображения данных

Чтобы не запутаться в очередности параметров при вызове функции, нужно в явном виде, используя формальные имена параметров функции, задавать им значения. Если значение параметра не указывать, график будет отрисован с использованием значения по умолчанию.

Рис. 4. Примеры и связь основных и вспомогательных функций

Все, что теоретически мы можем захотеть изменить, как правило, можно выполнить на практике, достаточно знать название параметра и его допустимые значения. Чаще всего требуется настраивать:

- Размер и толщину линий, точек, подписей, заголовков, разметки осей
- Цвет линий, точек, фона, масштабной сетки, текста надписей и подписей

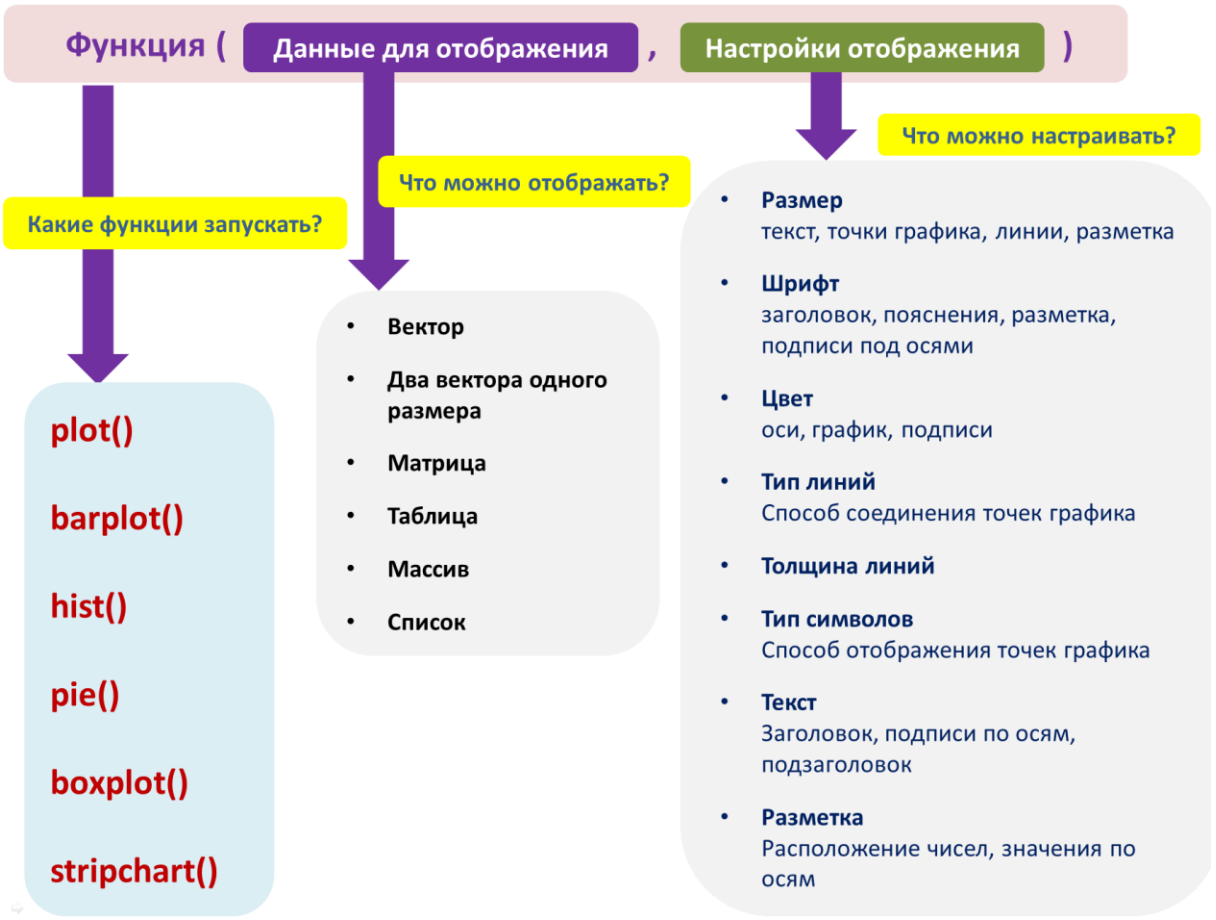

Рис. 5. Основные типы и назначение параметров графической функции

- Тип представления данных (только точки, точки, соединенные линиями, только линии, вид соединительных линий)
- Различные шрифты для вывода сообщений

Одной из наиболее часто используемых функций является универсальная и гибкая в настройке графическая функция **plot()**. Рассмотрим параметры и варианты вызова этой функции (см. рис. 6). Параметры для настройки отображения данных функции plot() работают аналогично и для других графических функций и могут быть использованы так же, как они используются в функции plot(). Напомним,

что информацию по всем возможным параметрам любой функции можно получить, запустив в консоли команду

?имя_графической_функции

где **имя_графической_функции** – название любой интересующей функции.

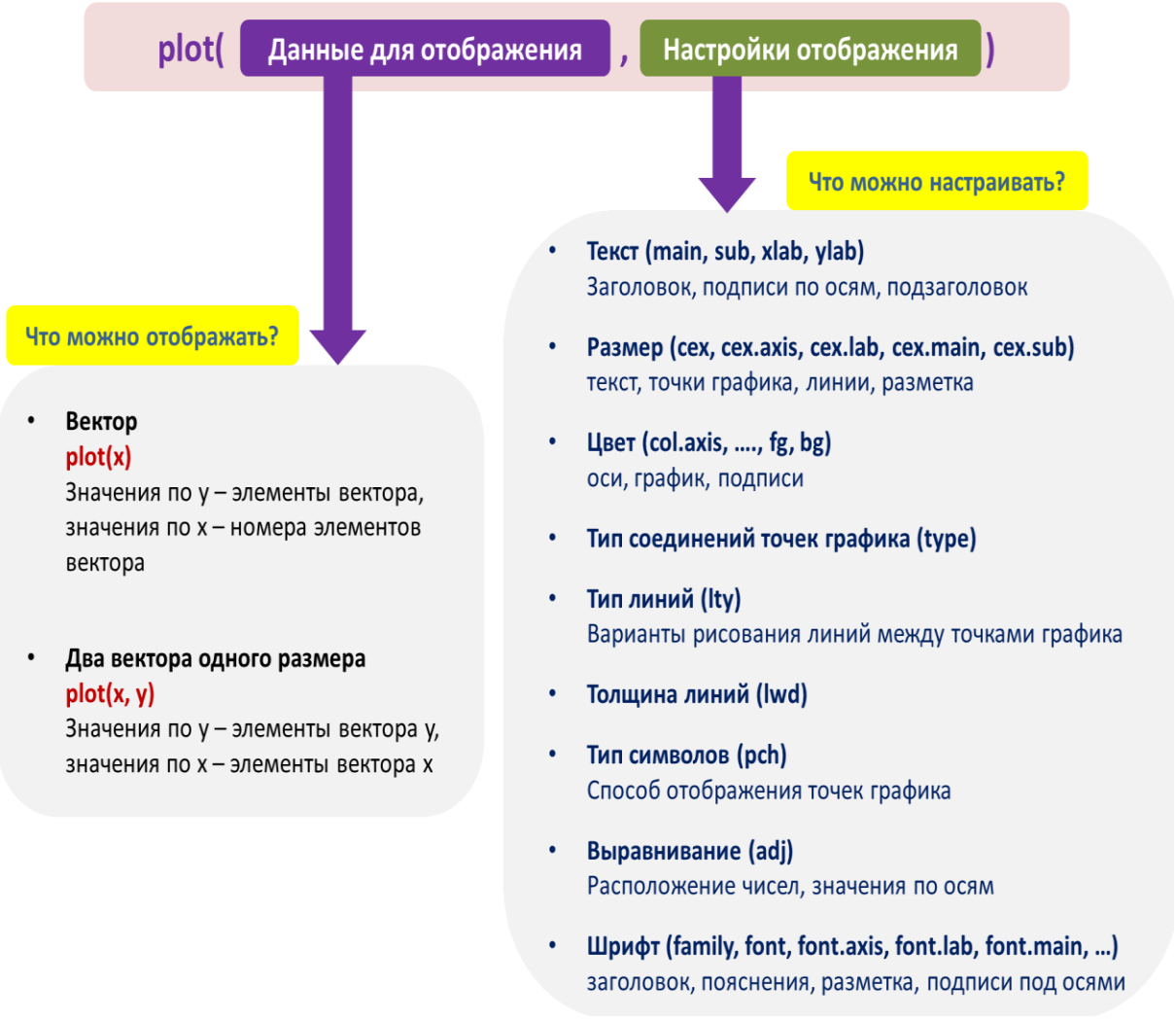

Рис. 6. Параметры графической функции plot()

Практические действия

Перейдем к практическому изучению графических функций, для чего подготовим числовой вектор res и передадим его как параметр в функцию plot() (рис. 7, 8).

```
res <- sample(x = -5:21, size = 30, replace = TRUE)

plot(x = res)
```

Рис. 7. Создание вектора случайных значений и вызов функции plot()

Проверьте, что произойдет, если запустить код res <- sample(..) без условия **replace = TRUE.** Программа должна выдать ошибку, так как среди 30 случайных значений в диапазоне от -5 до 21 обязательно будут повторяющиеся числа.

На графике (рис. 8) видим точки, случайным образом рассыпанные по прямоугольнику размером [1, 30] x [-5, 21].

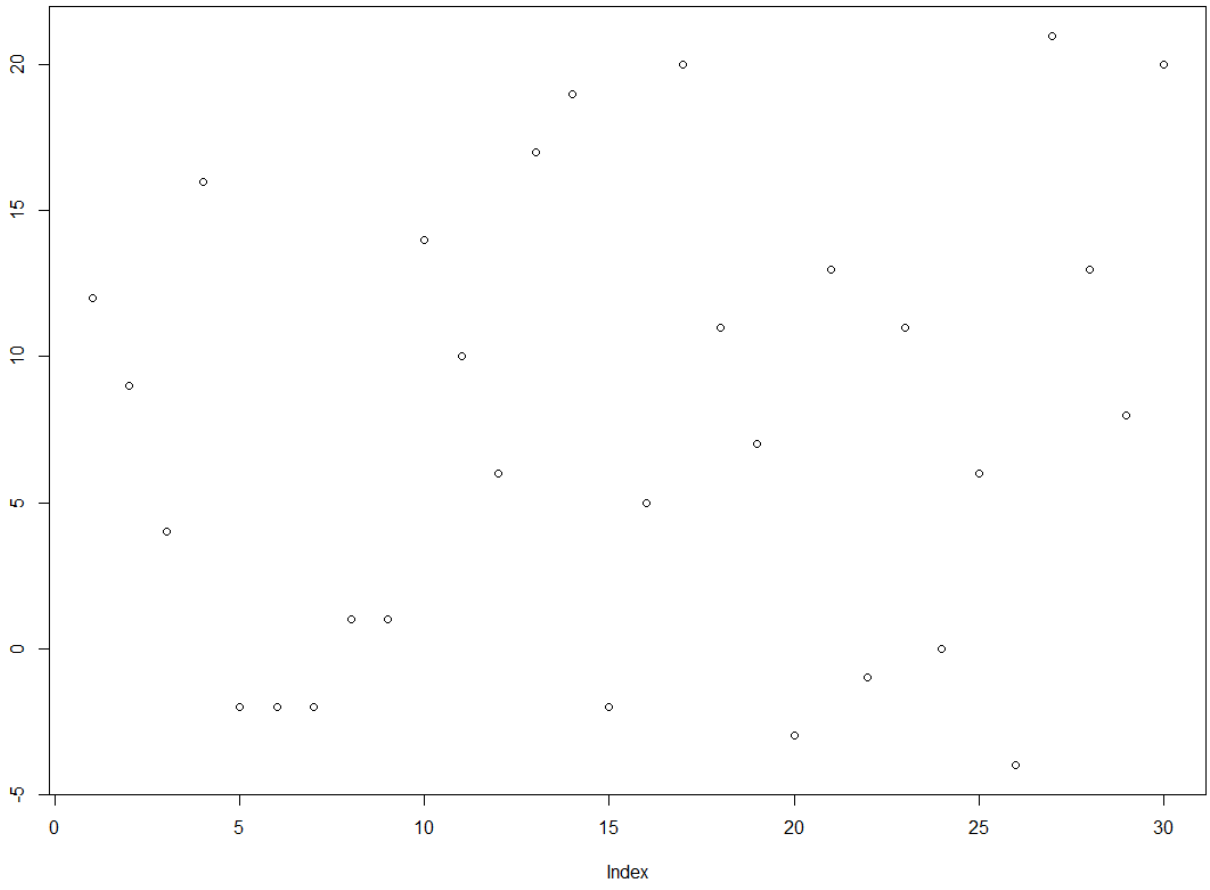

Рис. 8. Отображение вектора случайных значений на графике

Заметим, что точки располагаются точно в узлах невидимой масштабной сетки, образованной вертикальными и горизонтальными линиями, проходящими через целочисленные значения на осях OX и OY.

Масштабная сетка

Нарисуем масштабную сетку. Масштабная сетка строится посредством вызова функции, рисующей так называемые опорные линии. Для рисования опорной линии нужна вспомогательная графическая функция **abline()**.

В скрипте программы вызов **abline()** должен быть выполнен ПОСЛЕ вызова **plot()** (см. рис. 9), иначе **abline()** ничего не нарисует. Результат работы этой функции представлен на рис. 10.

```
res <- sample(x = -5:21, size = 30, replace = F)

plot(x = res)

# рисуем вертикальные линии on 1 до 30 синим цветом
abline(v = seq(1,30,1), col = 'blue')

# рисуем горизонтальные линии зеленым цветом
abline(h = seq(-5,21,1), col = 'green')
```

Рис. 9. Код рисования масштабной сетки на графике plot()

Вместо вызова функции abline() два раза ее можно вызвать один раз с тем же самым результатом. Подумайте, что нужно изменить в вызове abline(). Проверьте свои предположения.

Обратите внимание, что в параметрах функции **abline()** мы задаем тип линии – вертикальная (v = …) или горизонтальная (h = …), а также определяем цвет (color) через параметр col = ‘…’. Какие еще параметры есть у функции **abline()** ?

Выполните вызов функции с этими параметрами, объясните результат.

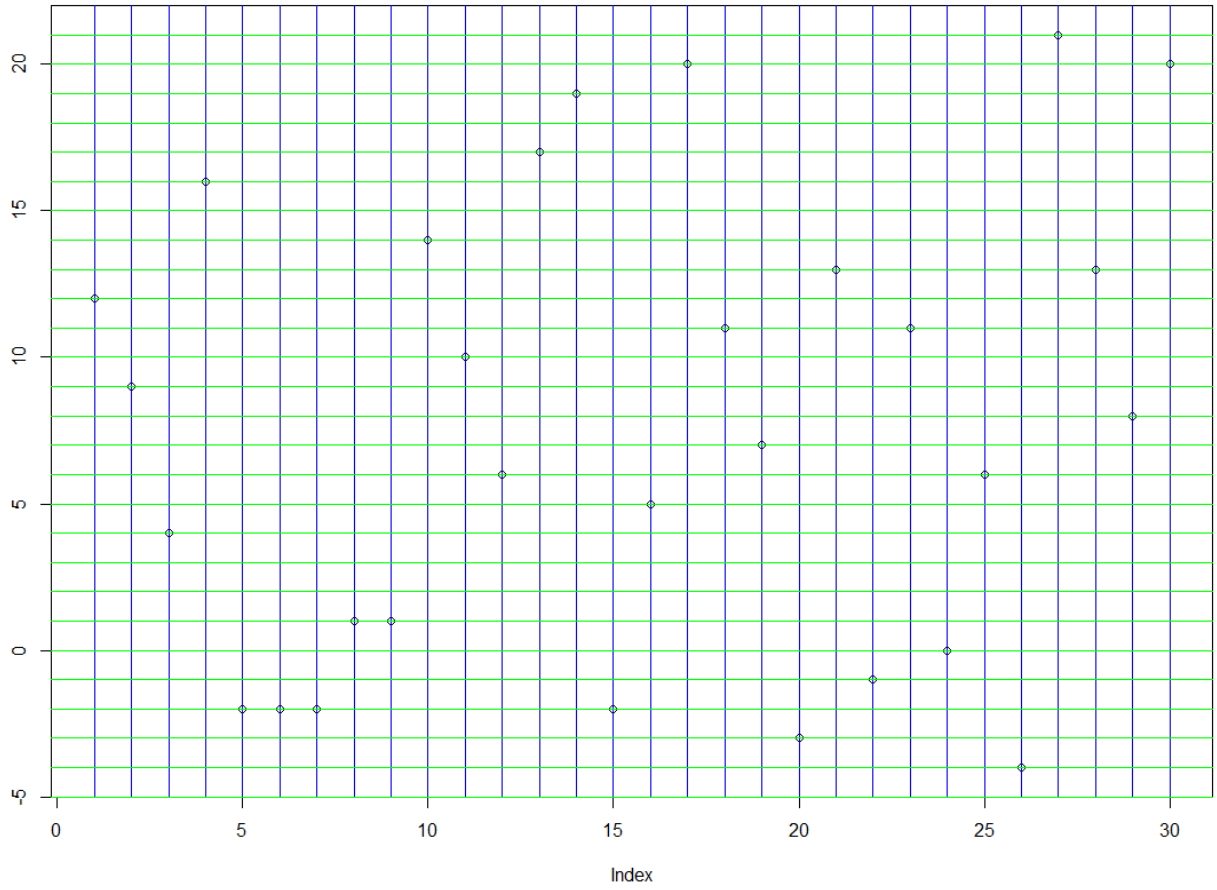

Рис. 10. Случайные точки в узлах масштабной сетки

Задание цвета

Все именованные цвета, доступные для использования в графике R, можно получить, вызвав специальную функцию **colors()**. Сколько всего цветов есть в нашем арсенале? Выберите любой цвет и раскрасьте наши случайные точки.

Каждый элемент графика может иметь свой цвет. Для задания цветов различных элементов графика используйте параметры, представленные в Таблице 1.

Вместо одного цвета можно указывать вектор цветов. В этом случае цвета будут присваиваться элементам по очереди. Если **col=c(“red”,“blue”)** и изображены три линии, первая будет красной, вторая – синей и третья снова – красной.

Таблица 1. Параметры, управляющие цветом различных элементов

col.* = , fg = , bg =

Параметр	Результат применения
col.axis	Цвет значений на осях
col.lab	Цвет подписей на осях
col.main	Цвет заголовков
col.sub	Цвет подзаголовков
fg	Цвет графика
bg	Цвет фона

Раскрасим точки в несколько цветов (см. рис. 11). То, что цвета различны, практически не видно, возникает следующая задача – задать нужный размер элементам графика.

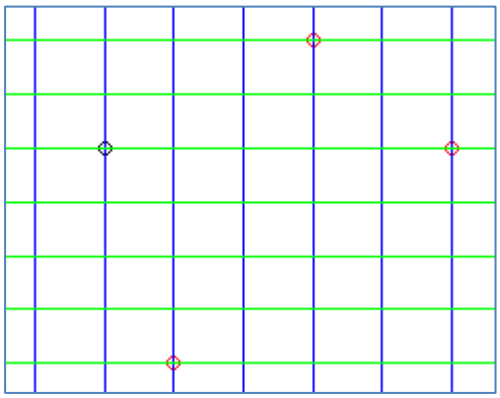

Рис. 11. Точки в узлах масштабной сетки раскрашены в разные цвета

Задание размера

На графике случайные точки видны плохо, нужно увеличить их размер. По умолчанию точки рисуются с некоторым коэффициентом, равным 1. Для изменения размера используется параметр cex, который будет увеличить размер точки, если cex>1, и уменьшать размер для cex<1. Так, для значения параметра cex=5 результат представлен на рис. 12.

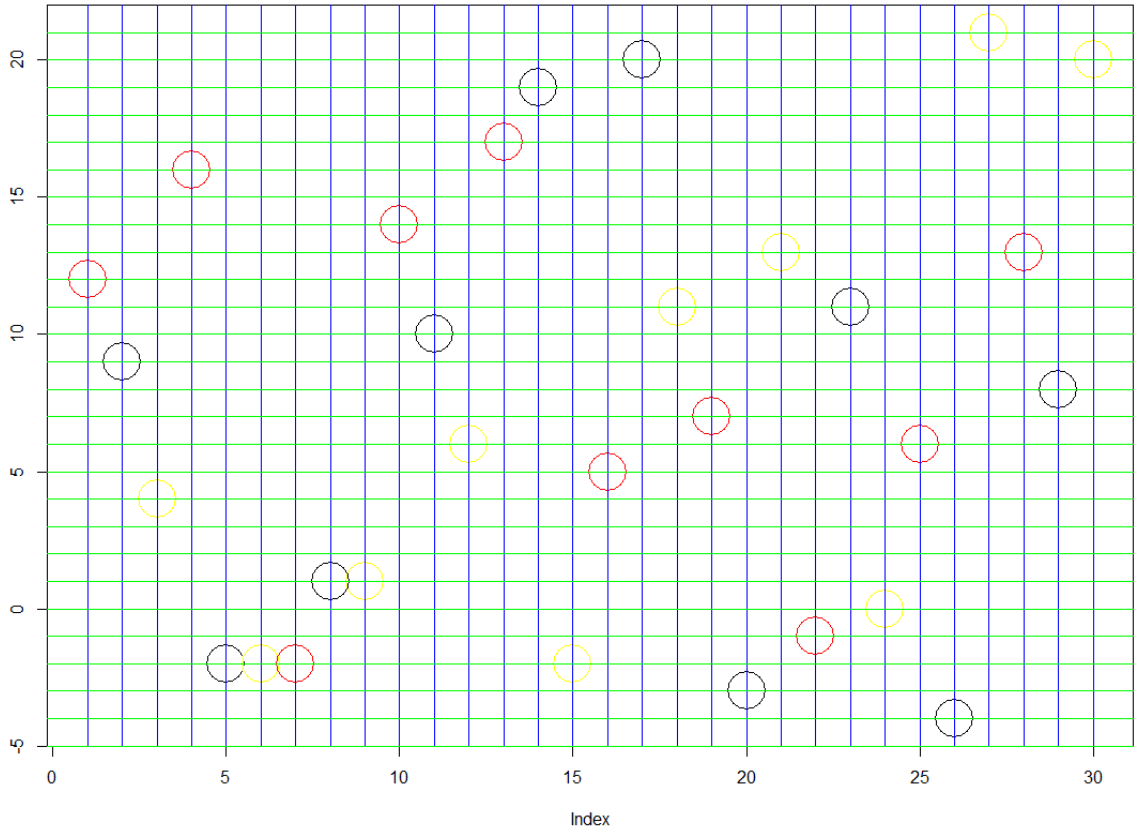

Рис. 12. Использование параметров col и cex

Варианты использования параметров, меняющих размеры элементов графика, представлены в таблице 2.

Задание вида отображения данных графика

Кроме окружностей (этот тип представления данных задан в R по умолчанию) значение может быть отрисовано разными вариантами отображения, что дает широкие возможности по размещению на одном

графике различных наборов данных. Управление представлением данных задается параметром pch=N, где N – целое число от 0 до 25 (рис. 13). Результат отображения на графике для N = 23 представлен на рис. 14.

Таблица 2. Параметры для изменения размера элементов графика

cex =

Параметр	Результат применения
cex	Коэффициент изменения размера элемента. По умолчанию **cex = 1** **cex = 1.5** означает, что элемент на 50% больше **cex = 0.5** – на 50% меньше
cex.axis	Размер цифр на осях по отношению к **cex**
cex.lab	Размер названий осей по отношению к **cex**
cex.main	Размер заголовков по отношению к **cex**
cex.sub	Размер подзаголовков по отношению к **cex**

Рис. 8. Варианты отображения точки на графике

Для символов с 21 по 25 можно отдельно указывать цвет заполнения (bg= ‘…’), (рис. 14). Самостоятельно протестируйте параметр с N от 1 до 25.

Интересно, что вместо N можно задавать в качестве значения pch любой символ, в этом случае параметр задается так: pch = ‘Ж’ (рис.. Измените значение параметра pch на любой символ и оцените результат.

Рис. 13. Варианты отображения значений на графике

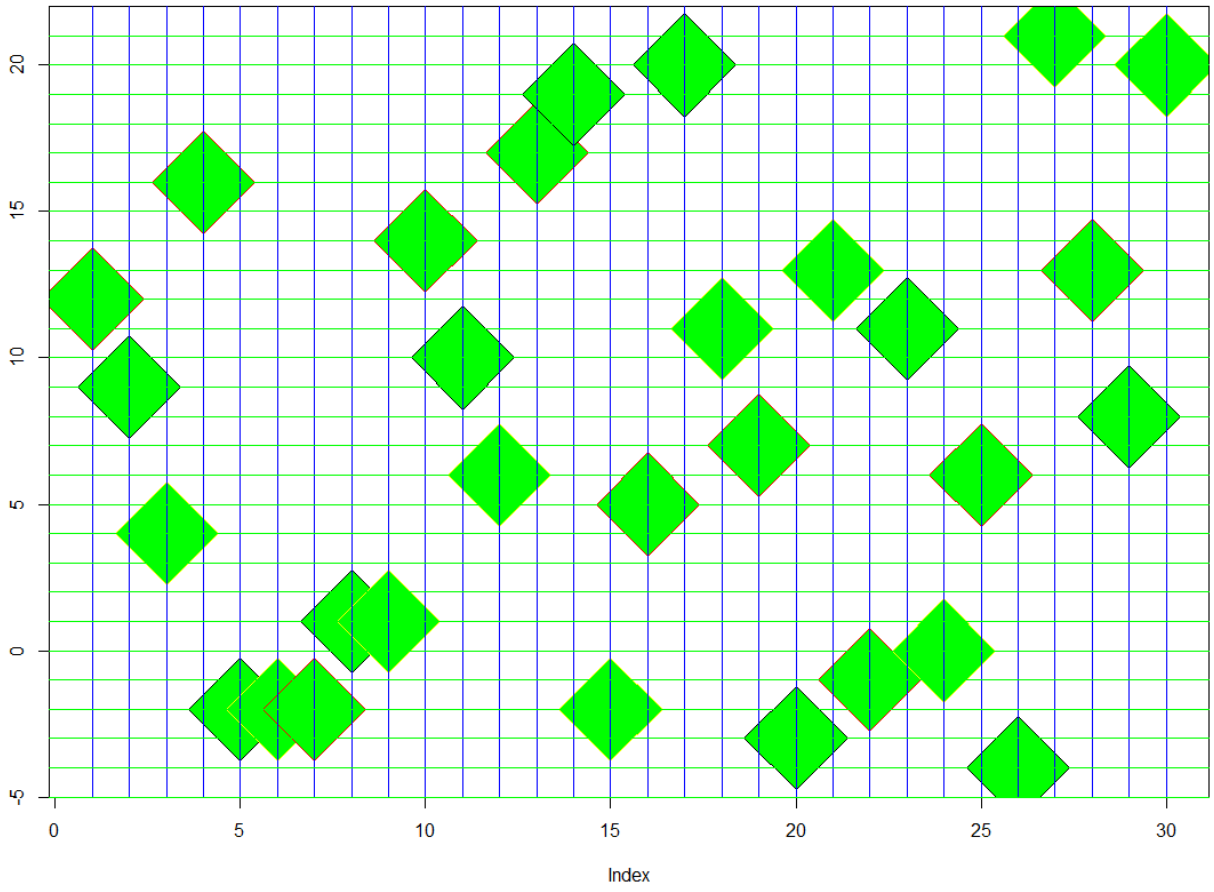

Рис. 14. Использование параметра pch для изменения вида графика

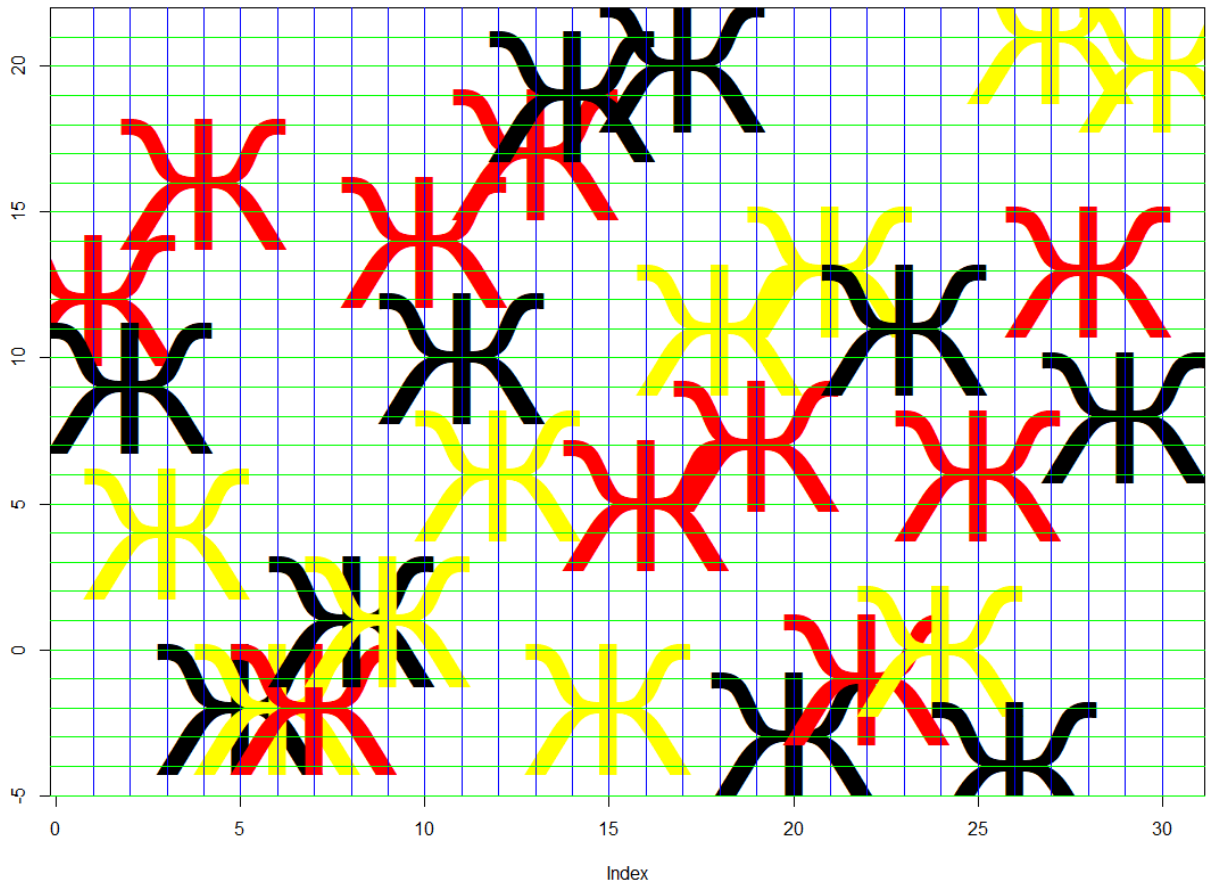

Рис. 15. Использование параметра pch со значением ‘Ж’

Соединение точек между собой

Рассмотрим способы соединения точек на графике. Основным параметром, фактически задающим тип графика, является параметр **type**. По умолчанию используется параметр type = ‘p’. Варианты значений для параметра **type** представлены на рис.16. Использование разных значений **type** дает большое разнообразие в оформлении данных, см. рис. 17.

На своих данных проверьте работу всех значений параметра **type**.

Вопросы возникают по поводу значения параметра type = ‘n’, зачем он нужен? Ответ следующий – порой нам нужно «пустой» график, чтобы затем размещать на нем необходимую информацию с помощью других функций.

type =

"p" (points) отображаем точки

"l" (lines) рисуем линии, проходящие через точки, точки не отображаются

"b" (both) отображаем точки, рисуем линии между ними

"c" рисуем линии, вместо точек - пробелы

"o" аналог “b”

"h" (high density) рисуем вертикальные линии до точек

"s" (steps) рисуем ступеньки между точками, вариант 1

"S" (steps) рисуем ступеньки между точками, вариант 2

"n" (no) вообще ничего не рисуем

Рис. 16. Типы соединения точек графика

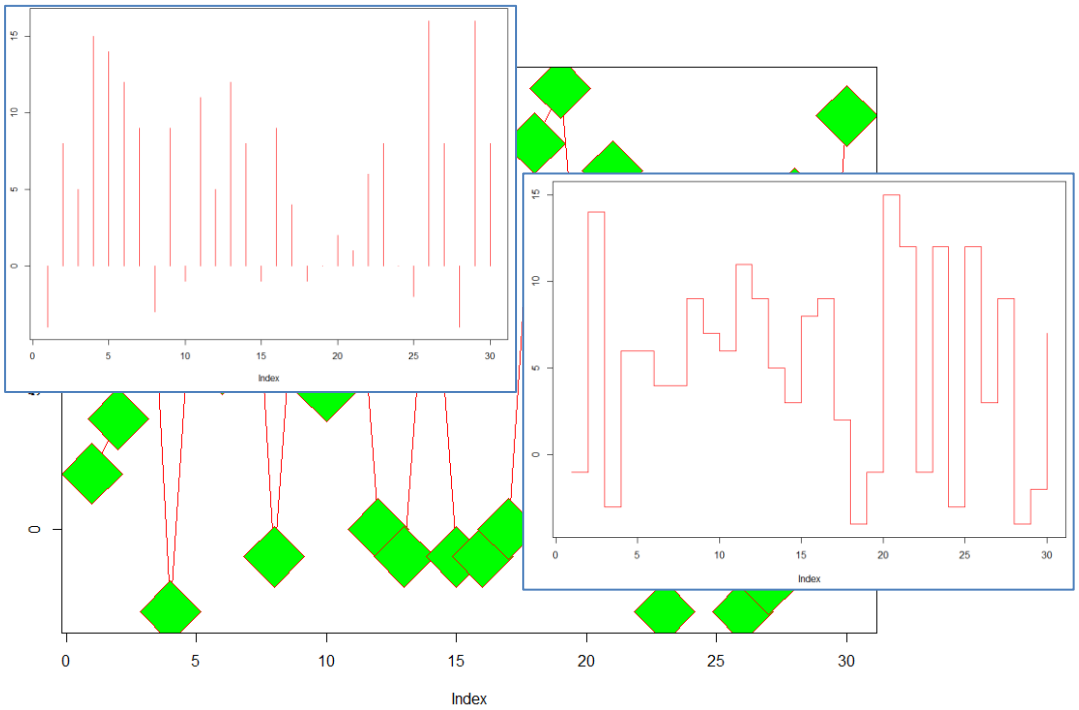

Рис. 17. Использование параметра type для изменения типа графика

Задание типа и толщины соединительной линии

По умолчанию точки на графике соединяются непрерывной линией. Вид этой линии можно менять, и отвечает за это параметр **lty**. Параметр может принимать 6 значений, значение по умолчанию = 1, все варианты представлены на рис. 18.

Рис. 18. Варианты соединительных линий между точками графика

Кроме задания типа линии мы можем менять ее толщину, используя параметр **lwd**. Параметр **lwd** – полный аналог параметра **cex** для точек. Вариант графика с использованием этих параметров представлен на рис. 19. Также на этом графике для функции **abline()** применен параметр **lty**. Подберите собственный набор параметров для отображения точек и линий. Добейтесь того, чтобы ваш график вам нравился.

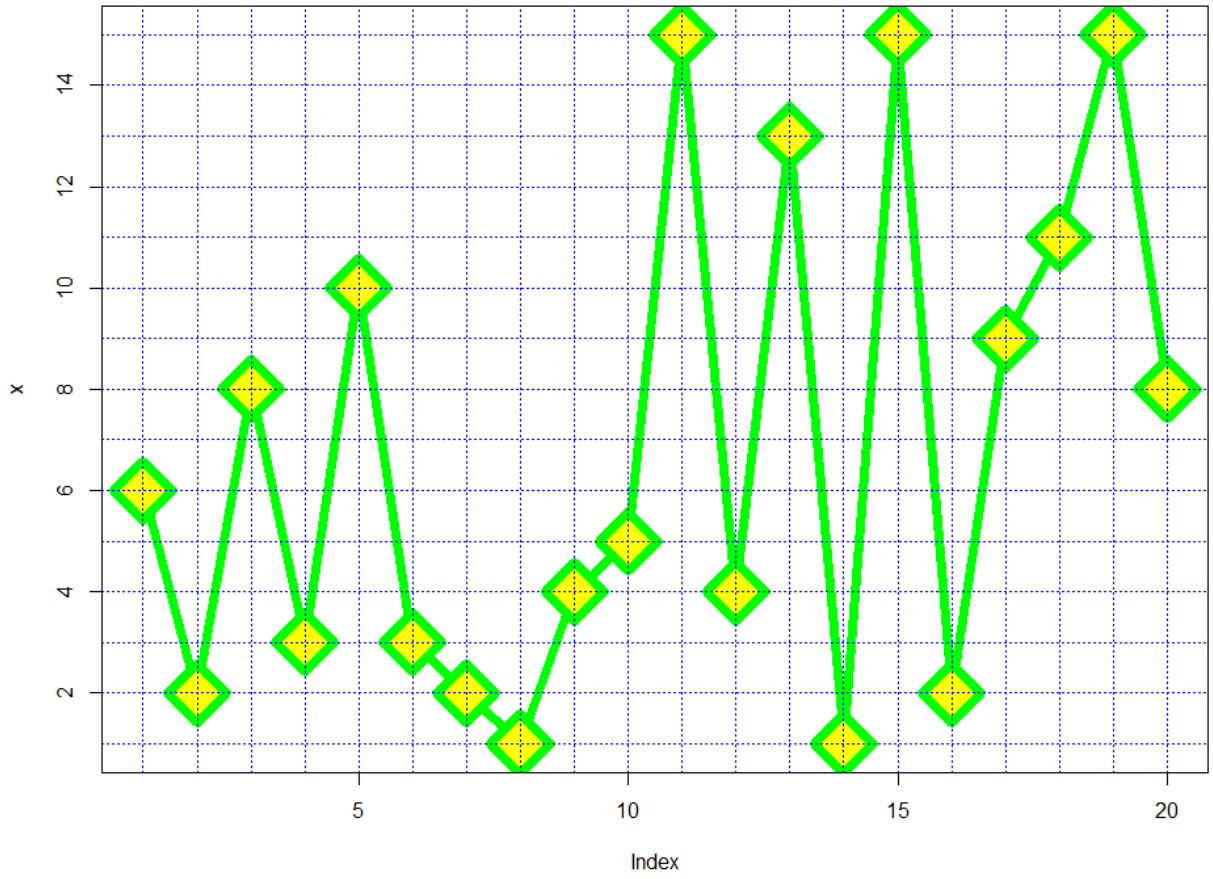

Рис. 19. Пример графика с различными значениями параметров

Размещение на графике дополнительных данных

Нередко на одном графике нужно отразить несколько наборов данных. Для размещения на готовом графике дополнительных наборов данных используются вспомогательные функции **lines()** и **points()**.

Параметры этих функций такие же, как и у функции **plot()**. Разместим на нашем графике функцию, формула которой представлена на рис. 20.

```
x2<-seq(-2,25, length.out=37)
y2<-2 + x2/3 * sin(x2^0.9)
```

Рис. 20. Задание векторов x2 и y2 для размещения на графике

Итоговый вид графика показан на рис. 21.

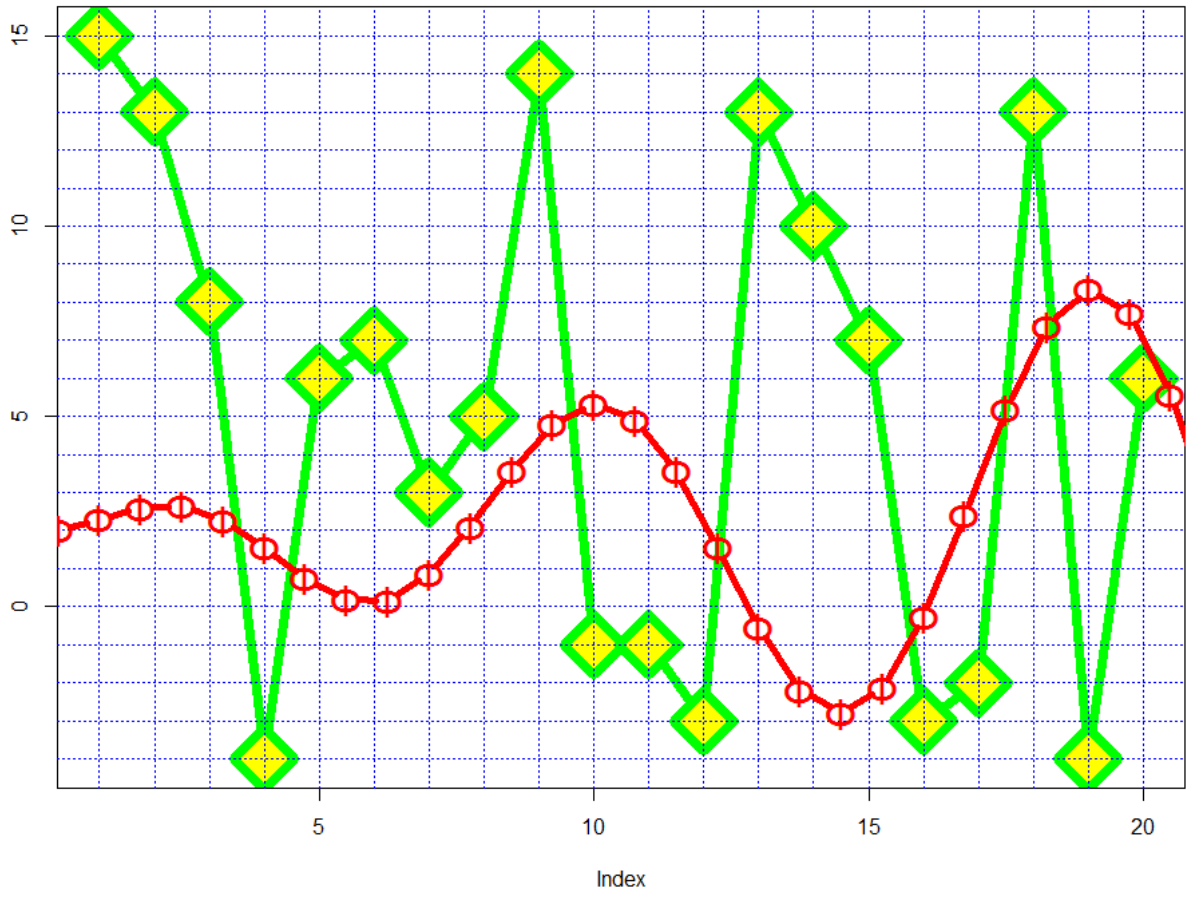

Рис. 21. Наложение двух наборов данных на одном графике

Создание заголовка, подписей осей

Одна из важных работ при оформлении графика – добавление поясняющего текста в заголовок, подписывание осей, создание подзаголовка, настройка размеров осей и т.д.

Для задания заголовка в plot() используется параметр **main**, например: main = ‘Динамика продаж по регионам’. Аналогичным образом для задания подписей по осям используются параметры **xlab**, **ylab**. Для задания подзаголовка (размещается под осью X) служит параметр **sub**. Эти параметры представлены в таблице 3.

Таблица 3. Поясняющие параметры графика

main = , sub = , xlab = , ylab =

Параметр	Результат применения
main	Задает заголовок графика
sub	Поясняющая надпись под графиком
xlab	Подпись оси X
ylab	Подпись оси Y

Хорошая практика подготовки отчетов – к каждому графику делать заголовки и необходимые подписи. У того, кто смотрит ваш график, не должно быть вопросов о том, что изображено на графике. Плохо подготовленный график – это график, по оформлению которого вам вынуждены задавать вопросы.

Примечание. В графических функциях используется множество параметров. В коде программы лучше записывать эти параметры так, чтобы на каждой строчке был один параметр. В этом случае их удобно менять, убирать в комментарии и видеть всю картину настроек. Пример см. на рис. 22.

Для того, чтобы автоматически выстроить параметры таким образом, в RStudio используют следующие команды: выделение кода (Ctrl-A) и его последующее форматирование (Ctrl-Shift-A).

Результат представлен на рис. 23.

```
plot(
  x = res,
  col = c('red','cyan', 'black', 'yellow'),
  cex = 10,
  pch = 23,
  bg = 'green',
  type = 'b',
  lwd = 12,
  lty = 3,
  main = 'Продуктивность рекламных компаний банка ПЕРИМЕТР',
  sub = 'Данные из открытых источников',
  xlab = 'Декады',
  ylab = 'Выручка, млн. руб'
)
```

Рис. 22. Пример оформления вызова функции plot()

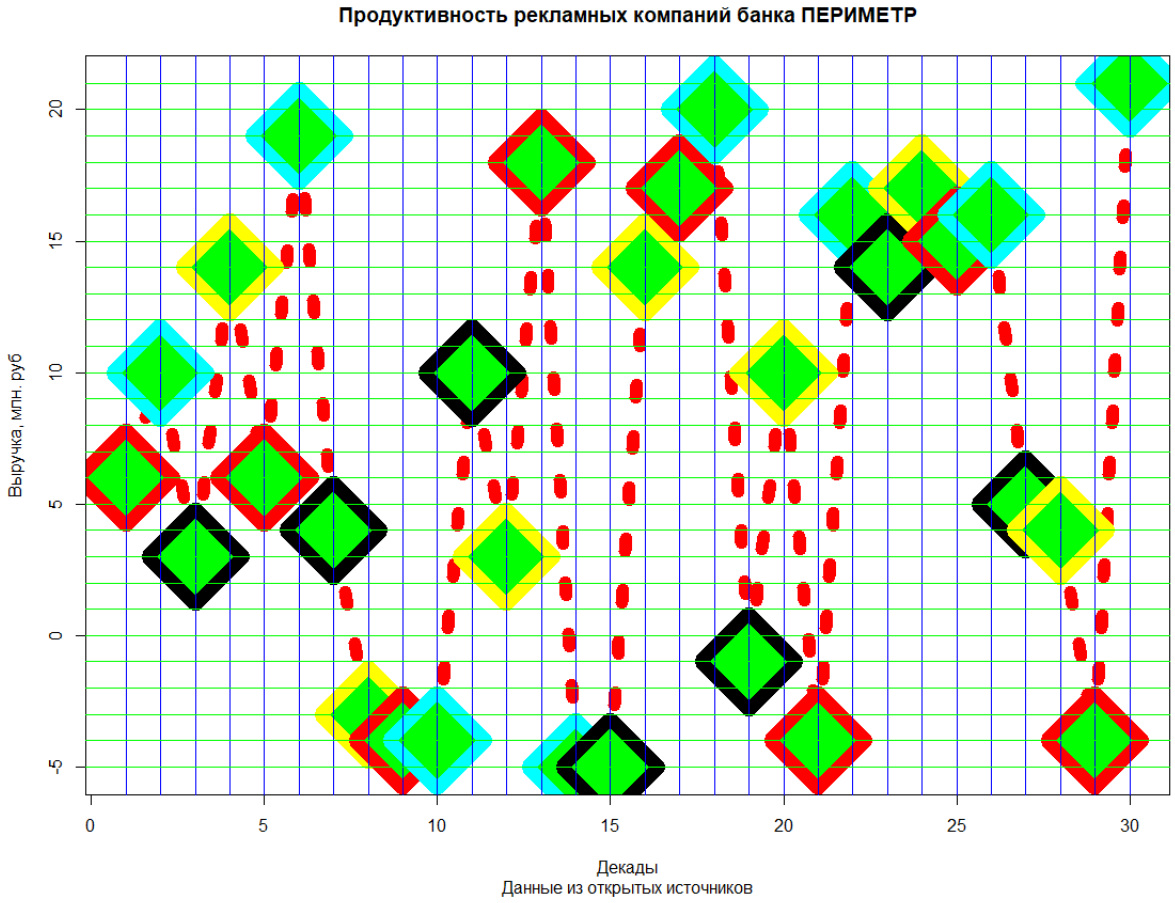

Рис. 23. Результат вызова функции plot() с набором параметров

Самостоятельные задания

1. Разработать функцию generate.supply(), возвращающую целочисленный вектор значений, соответствующих поставкам некоторого товара в магазин по дням. Параметры функции: n – количество дней (длина вектора поставок), min – минимальное количество поставки в день, max – максимальное кол-во поставки.
2. Вызвать функцию generate.supply() и использовать результат ее вызова для создания 3-5 графиков с различными вариантами оформления. Каждый график строится по одному товару и должен отображать динамику поставки товара в магазин по дням.
3. Используя вызов generate.supply() два раза, получить данные поставок по двум магазинам, отобразить эти данные на одном графике. Подготовить 2-3 графика с разным оформлением. Использовать функцию legend() для вывода поясняющих подписей.
4. Используя вызов generate.supply() пять раз, получить данные поставок по дням по пяти магазинам, отобразить эти данные на одном графике. Подготовить 2-3 графика с разным оформлением. Использовать функцию legend() для вывода поясняющих подписей. Отобразить на графике все данные по поставкам из всех магазинов.
5. Используя данные из Задания 4, подготовить график суммарных поставок товара в каждый магазин. Данные по магазинам разместить на одном графике. Вывести поясняющие подписи.
6. Разработать функцию generate.sale(), возвращающую целочисленный вектор значений, соответствующих продажам некоторого товара в магазин по дням. Параметр функции: in – вектор поставок товара по дням. Продажи в любой день не могут превышать поставку товара в этот день.

7. На одном графике отобразить динамику поставки и продажи товара в одном магазине.
8. Получить данные по поставкам и продажам товара по трем магазинам, отобразить эти данные на одном графике.
9. Построить график непроданных товаров по дням для трех магазинов. Использовать столбиковое представление, где высота итогового столбика формируется из трех столбиков, каждый из которых – непроданный товар отдельного магазина.

На графиках должна быть представлена вся информация, необходимая для понимания графика (подписи по осям, указание размерностей отображаемых величин, заголовки). Если по графику возникают вопросы, значит, график нельзя считать законченным.