Урок 5. Простейшая графика

В этом уроке мы поговорим о визуализации данных стандартными средствами R, без загрузки дополнительных библиотек. Загрузим таблицу из csv-файла с данными по компаниям S&P 500 (Standard and Poor's 500):

```
dat <- read.csv("financials.csv")
View(dat)
```

Предположим, что наша задача — визуализировать распределение компаний по секторам, то есть наглядно показать, сколько компаний разного типа представлено в наших данных. Для описания неколичественного показателя `Sector` мы можем использовать частоты (абсолютные или относительные – в долях или процентах). Построим таблицу частот для показателя `Sector`:

```
table(dat$Sector) # секторы
```

```
##
##     Consumer Discretionary           Consumer Staples
##                         84                         34
##                     Energy                Financials
##                         32                         68
##                Health Care                Industrials
##                         61                         67
##     Information Technology                  Materials
##                         70                         25
##                Real Estate Telecommunication Services
##                         33                          3
##                  Utilities
##                         28
```

Теперь построим столбчатую диаграмму (*bar chart*), которая отразит эти частоты на графике. Обычная столбчатая диаграмма в R строится не для самого показателя, а для таблицы частот, вот так:

```
barplot(table(dat$Sector))
```

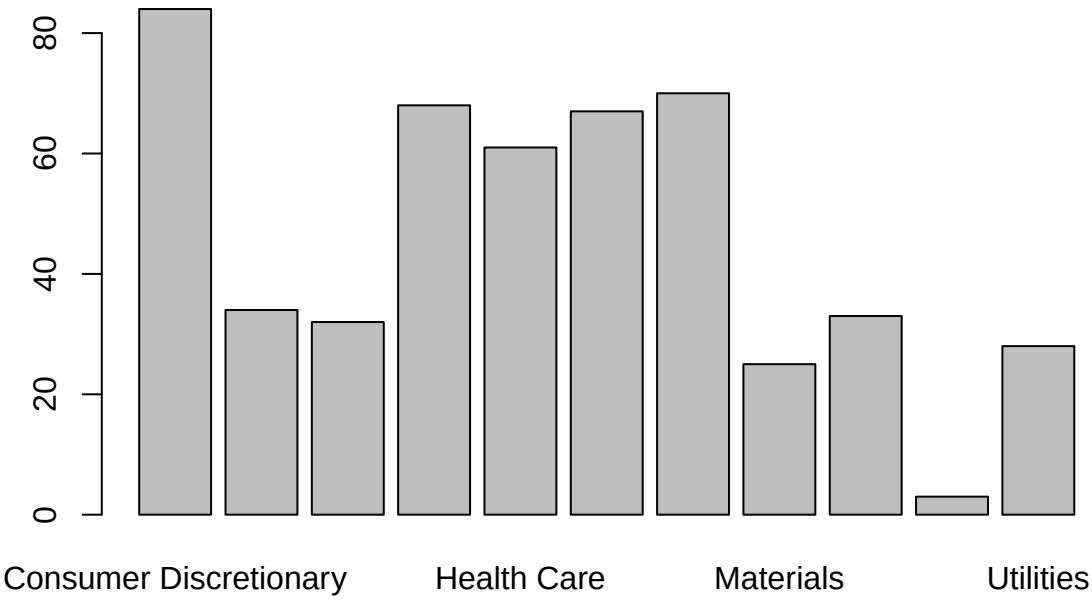
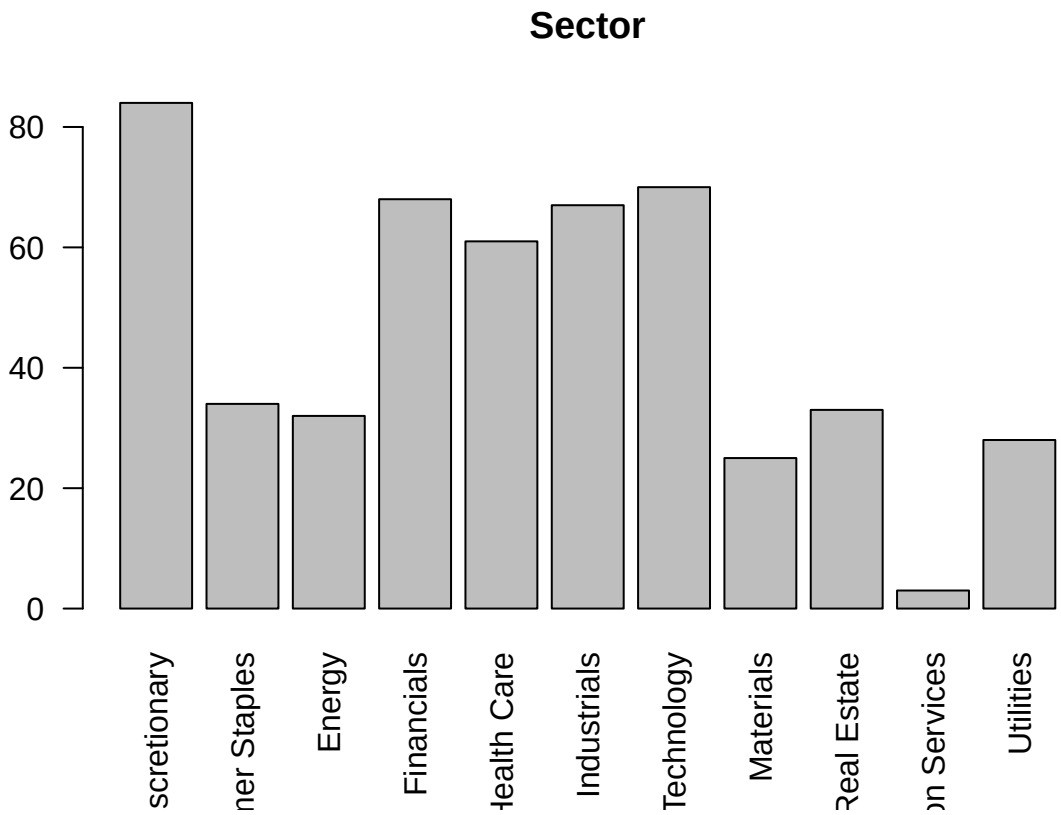


Диаграмма показывает, сколько компаний каждого типа в датафрейме. Приведем график в порядок:

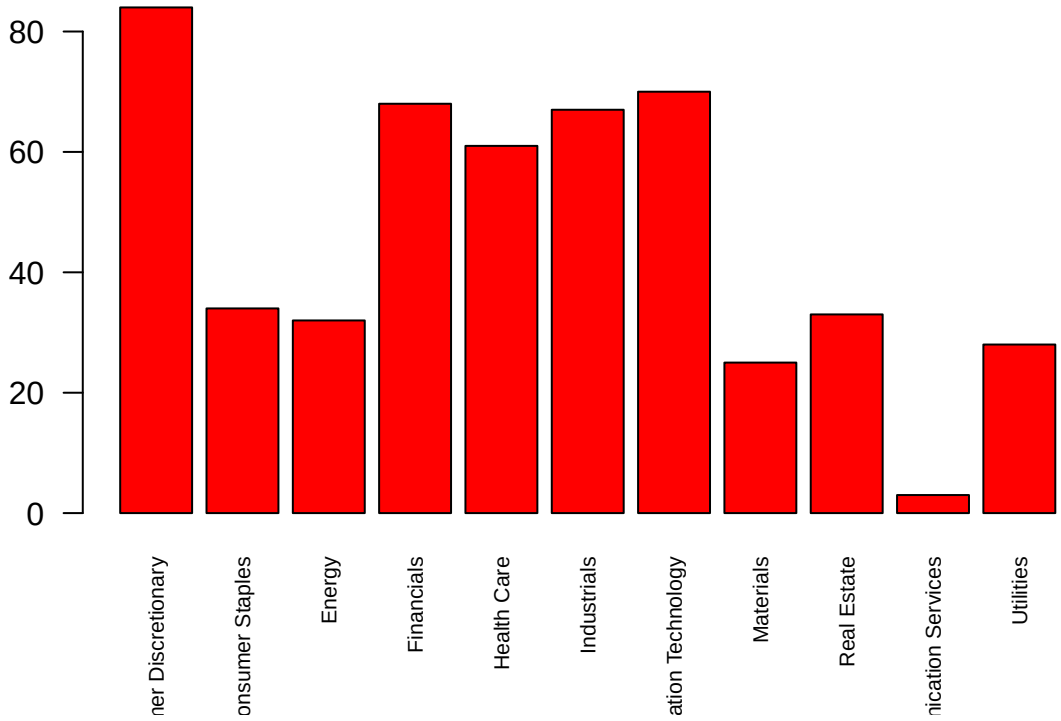
добавим заголовок (аргумент `main`), и повернем подписи к столбикам на 90 градусов (аргумент `las=2`):

```
barplot(table(dat$Sector),
main = "Sector",
las = 2)
```



Аргумент `las` отвечает за расположение подписей относительно осей, по умолчанию используется значение 0 (параллельно осям), мы выставили значение 2 (перпендикулярно осям). Уменьшим размер шрифта у подписей (аргумент `cex.names`, где `cex` отвечает за размер, а names за то, что мы настраиваем подписи к столбцам графика) и поменяем цвет столбцов на красный (аргумент `col`):

```
# cex - по умолчанию размер 1, здесь уменьшаем до 65% от него
barplot(table(dat$Sector),
main = "Sector",
las = 2,
cex.names = 0.65,
col = 'red')
```



Цветов в R много, и многие из них имеют очень интересные названия. (Ссылка на полный список цветов). Если интересует сам список цветов, только названия, можно воспользоваться функцией `colors()`:

```
colors()
```

Если мы хотим сделать столбцы разного цвета, вместо одного цвета в опции `col` нужно указать вектор цветов. Важно, чтобы число элементов в векторе цветов совпадало с числом столбцов, иначе R выдаст ошибку. Давайте сделаем что-то более продвинутое — добавим чередующиеся цвета: красный и синий. И тут нам пригодится изученная ранее функция `rep()`.

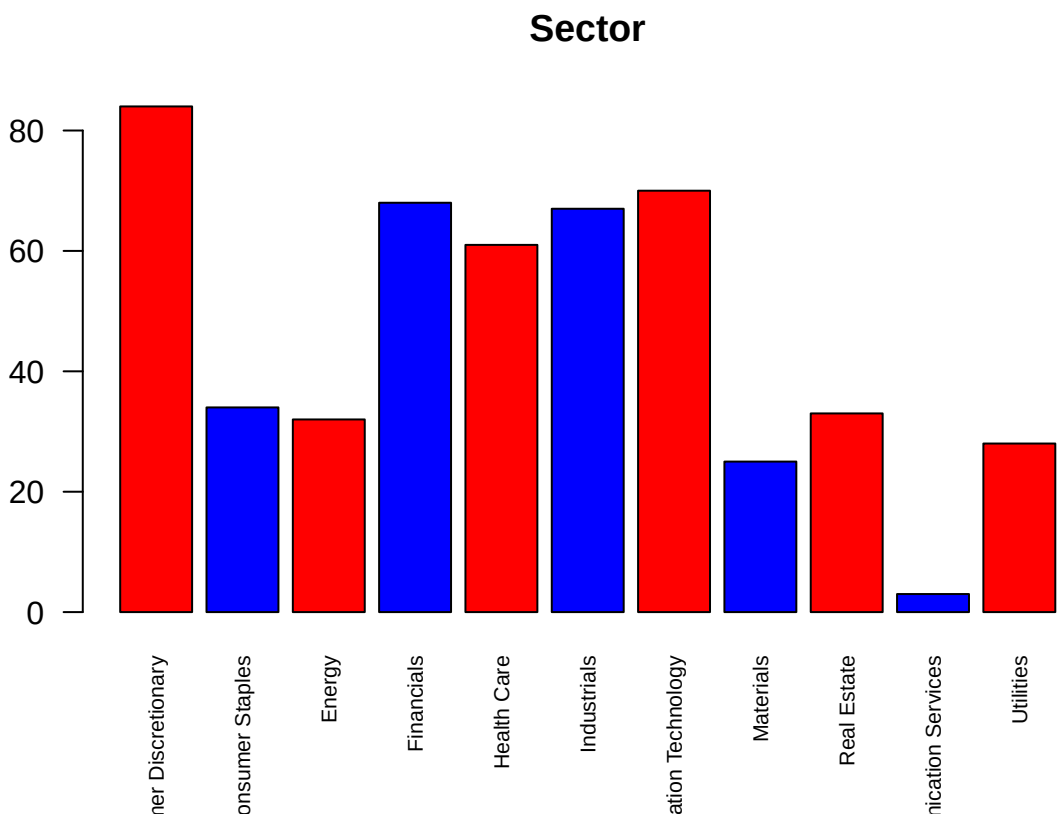
Так как столбцов у нас 11, нечетное число, мы можем поступить так: пять раз повторить пару цветов `c('red', 'blue')` и «доклеить» к полученному вектору последний цвет (красный).

```
cols <- rep(c('red', 'blue'), 5)
cols <- c(cols, 'red')
cols
```

```
##  [1] "red"  "blue" "red"  "blue" "red"  "blue" "red"  "blue" "red"  "blue"
## [11] "red"
```

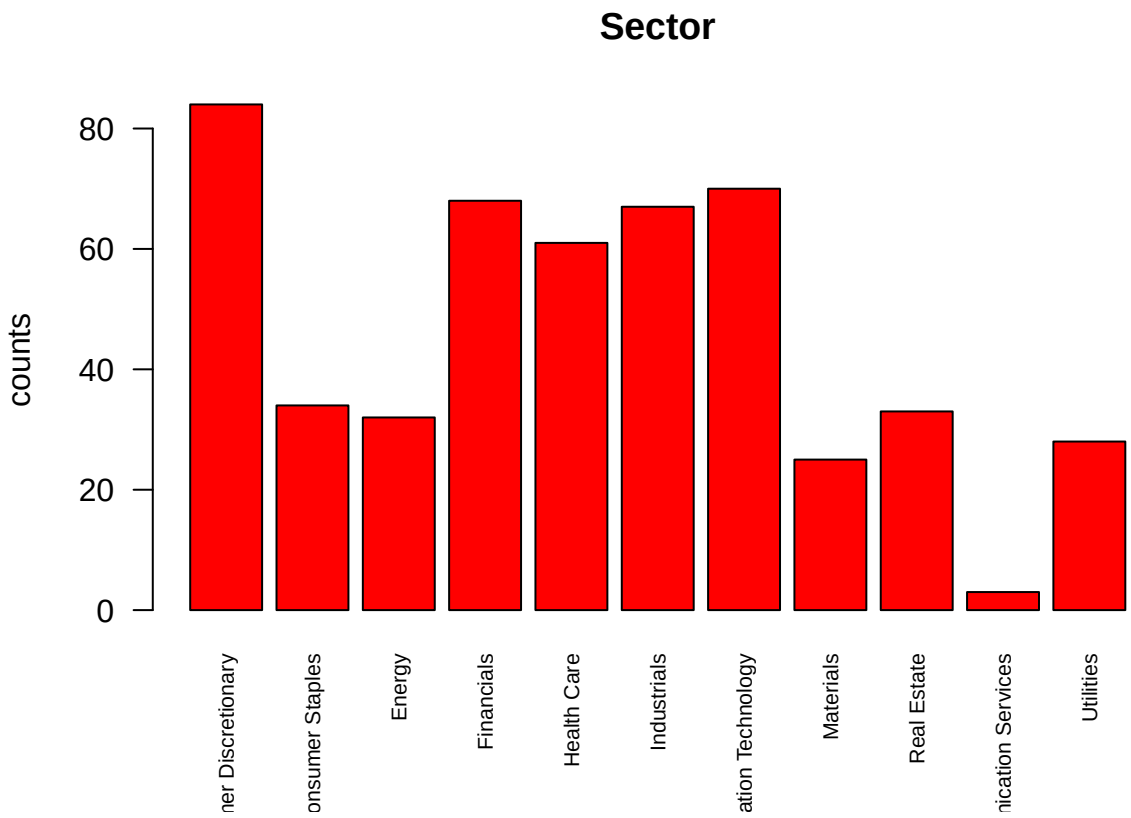
Теперь поменяем цвета на самом графике:

```
barplot(table(dat$Sector),
main = "Sector",
las = 2,
cex.names = 0.65,
col = cols)
```



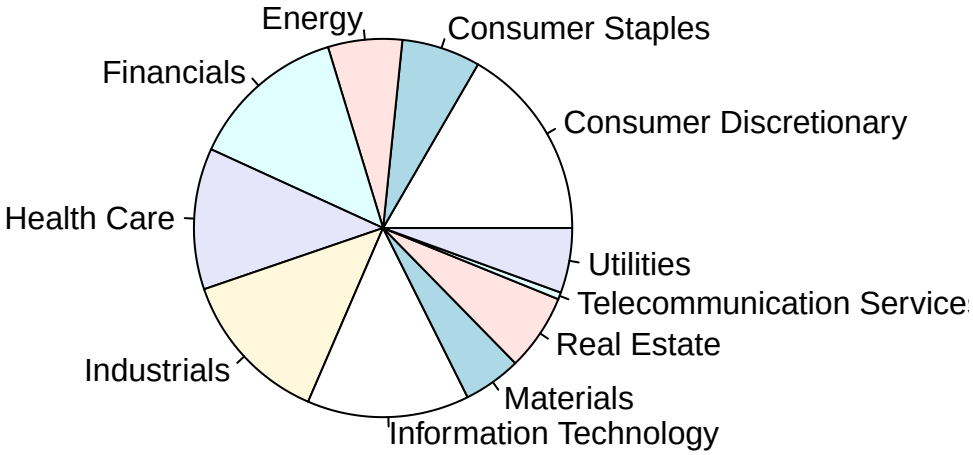
Добавим подпись к вертикальной оси (аргумент `ylab`, а для горизонтальной оси, для оси `x`, как можно догадаться, будет `xlab`, оба аргумента являются сокращениями от `x-label` и `y-label`):

```
barplot(table(dat$Sector),
main = "Sector",
las = 2,
cex.names = 0.65,
col = 'red',
ylab = "counts")
```



Теперь перейдем к другому графику – круговой диаграмме (*pie chart*), которая строится для относительных частот – процентов.

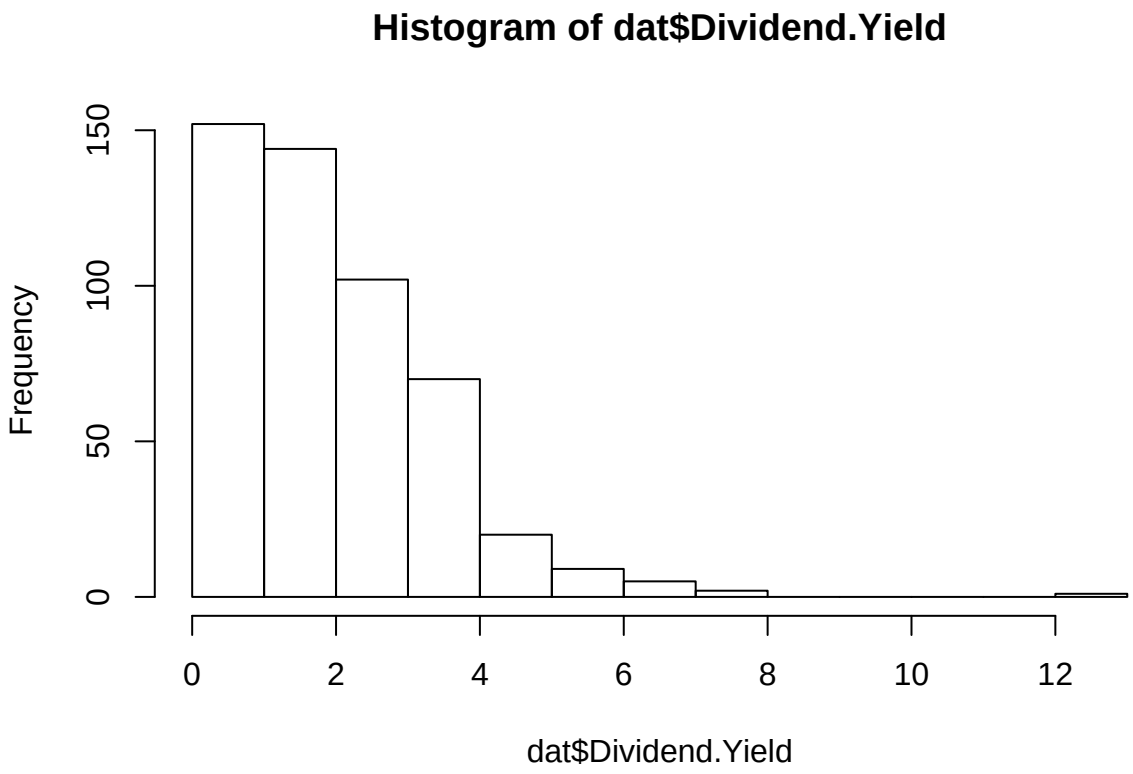
```
pie(table(dat$Sector))
```



Круговые диаграммы, построенные стандартными средствами R, не всегда симпатичные, чтобы сделать их красивыми, нужно постараться. Давайте пока оставим это до практического задания, а пока перейдем к другому графику — к гистограмме.

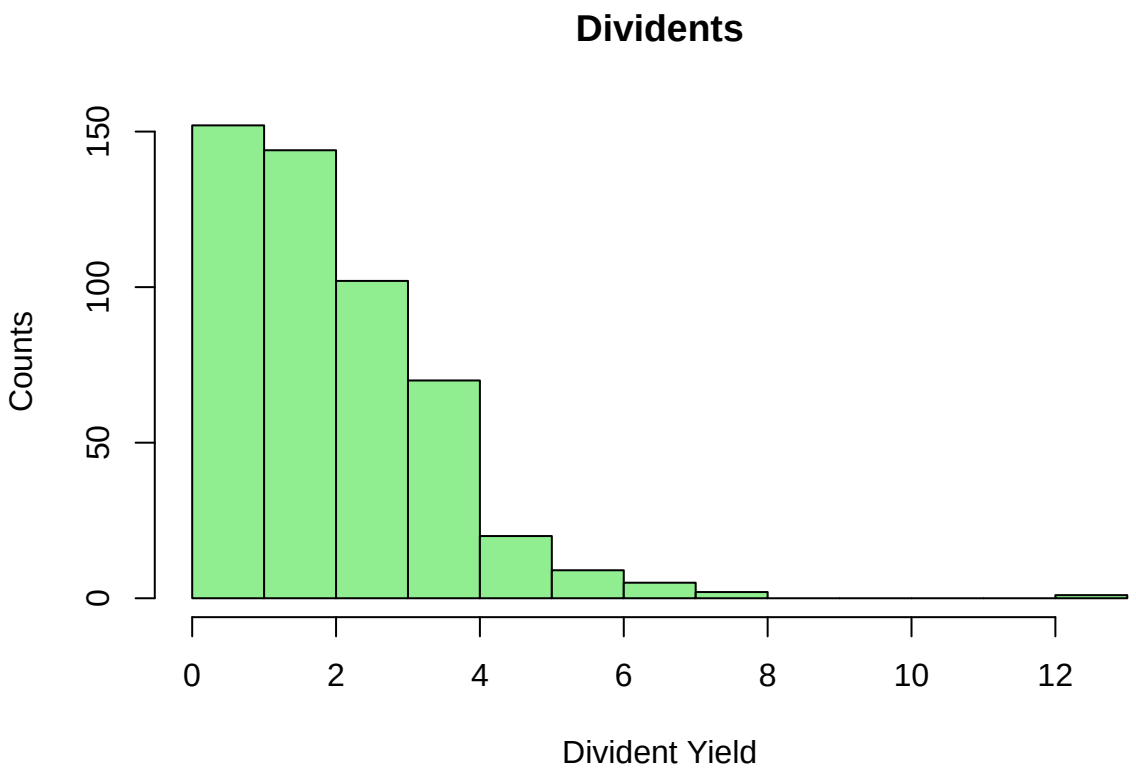
Гистограмма (*histogram*) – график, который иллюстрирует соответствие между значениями количественного показателя и частотой, с которыми эти значения встречаются в данных. Для начала построим самую простую гистограмму. Построим гистограмму для дивидентов.

```
hist(dat$Dividend.Yield)
```



Теперь сделаем ее красивой: добавим заголовок графика, поменяем цвет и подпишем оси.

```
hist(dat$Dividend.Yield,
main = "Dividents",
col = "lightgreen",
xlab = "Divident Yield",
ylab = "Counts")
```



Что показывает эта гистограмма? Форму распределения дивидентов. По гистограмме видно следующее. Распределение дивидентов скошено вправо — у графика есть длинный «хвост» справа. Это означает, что большинство значений сконцентрировано в районе нуля (часто встречающиеся значения), а компаний с большим числом дивидентов совсем мало.

В завершение урока посмотрим, как выгружать графики из RStudio. Первый способ простой: в окне *Plots* выбрать нужный график и кликнуть на кнопку *Export*. Выбрать нужный графический формат или pdf (особенно актуально для больших графиков или карт) и сохранить файл на компьютер. Второй способ более универсальный — выгрузить график с помощью кода. Для примера сохраним последний график в формате *png*:

```
# создаем и открываем файл graph.png для записи
dev.copy(png, "graph.png")
```

```
## quartz_off_screen
##                 3
```

```
# прогоняем код для графика
hist(dat$Dividend.Yield,
main = "Dividents",
col = "lightgreen",
xlab = "Divident Yield",
ylab = "Counts")

# записываем и закрываем файл
dev.off()
```

```
## pdf
##   2
```

Первая и последняя строки выше необходимы для сохранения графика в файл. Строка с `dev.copy()` позволяет выбрать устройство, с помощью которого мы создаем график. Здесь указано png, то есть,

мы хотим не просто вывести график на экран внутри RStudio, а сохранить его в виде png-файла. Если бы мы хотели сохранить его в формате *jpeg*, мы бы написали *jpeg*, если *pdf* – то *pdf*. Далее внутри `dev.copy()` мы указали название файла, в который мы хотим записать график. После этой строки мы исполняем весь код, относящийся к графику (как если бы мы ожидали его в отдельном окне), только теперь все элементы графика будут записываться в файл. В конце, когда работа над графиком закончена, мы пишем строку `dev.off()`, которая закрывает файл с графиком, сохраняет изменения, финализирует. Если этого не сделать, то график не сохранится, в папке будет лежать пустой png-файл, который, скорее всего, даже и не откроется. После строки `dev.off()`, если мы будем продолжать строить графики, они будут отображаться в отдельном окне RStudio, но записываться в файл не будут.

Теперь можем найти график в папке по умолчанию (здесь мы не прописывали особый путь к файлу):

```
getwd() # ищем файл здесь
```

```
## [1] "/Users/allat/Desktop"
```

На этом закончим обзор графики и перейдем к практическому заданию.