Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Удмуртский государственный университет»
Институт экономики и управления

Визуализация данных на языке R

Учебно-методическое пособие

Ижевск
2024

УДК 004.43(075.8)
ББК 32.973.2р30
В428

Рекомендовано к изданию Учебно-методическим советом УдГУ

Рецензент: канд. экон. наук, доцент, директор института дополнительного образования ФГБОУ ВО «Удмуртский ГАУ» **О.В. Котлячков**

Составитель: Максимов Д.Г., Котлячкова Н.В.

В428 Визуализация данных на языке R : учеб.-метод. пособие : [Электрон. ресурс] / сост. Д.Г. Максимов, Н.В. Котлячкова. – Ижевск : Удмуртский университет, 2024. – 59 с.

Рассматриваются возможности языка R по визуализации данных. Учебно-методическое пособие предназначено для обучающихся очной, очно-заочной форм обучения экономических направлений. Уровень образования – бакалавриат.

УДК 004.43(075.8)
ББК 32.973.2р30

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Предлагаемое учебно-методическое пособие предназначено для обучающихся направления 38.03.03 «Управление персоналом» в рамках дисциплины «Компьютерные технологии в обработке социально-экономической информации».

Целью освоения дисциплины «Компьютерные технологии в обработке социально-экономической информации» является овладение студентами знаний о компьютерной технике и прикладных программах, связанных с подготовкой, редактированием, анализом и хранением данных, поиском информации (возможностями компьютерной техники, которые рационально использовать для решения задач, связанных с профессиональной деятельностью в области управления персоналом).

Изучение дисциплины «Компьютерные технологии в обработке социально-экономической информации» позволит сформировать компетенции обучающегося:

- ОПК-5 – способен использовать современные информационные технологии и программные средства при решении профессиональных задач;
- ОПК-6 – способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности.

Учебно-методическое пособие предназначено для обучения студентов визуализации экономических данных с помощью языка R и одного из наиболее популярных пакетов – *{ggplot2}*.

В учебно-методическом пособии пошагово описан порядок построения диаграмм с помощь языка R.

При выполнении заданий, представленных в данном учебно-методическом пособии, обучающийся получит практический опыт по обработке экономических данных и их визуализации.

1. Предварительные требования

Чтобы успешно выполнить задания необходимо установить на компьютер прикладное программное обеспечение, о котором будет сказано ниже.

Также необходимы минимальные знания по работе с языком *R* и базовые знания статистики.

Для выполнения приведённых в учебно-методическом пособии заданий и примеров необходимы следующие средства: *R, RStudio* и коллекция дополнительных пакетов.

1.1 Язык R

R – это язык программирования и среда для статистической обработки данных. Это инструмент с открытым исходным кодом, который поддерживается многими исследователями по всему миру.

R – это чувствительный к регистру клавиатуры интерпретируемый язык программирования.

В настоящее время существует огромное количество прикладных программных средств для статистической обработки данных, таких как *SAS, MS Excel, IBM SPSS, Stata*.

Преимущества R[1]:

1. *R* – бесплатный программный продукт.
2. *R* – мощная среда для статистических вычислений.
3. В *R* реализовано огромное количество статистических процедур.
4. Частое появление новых функций.
5. *R* имеет современные графические возможности.
6. *R* – это мощная платформа для исследования данных.
7. *R* представляет возможность программирования новых статистических методов.
8. Возможности R можно интегрировать в приложения, написанные на других языках, включая *C++*, *Java*, *Python*, *PHP*, *Pentaho*, *SAS* и *SPSS*.
9. *R* поддерживается наиболее популярными операционными системами (*Windows*, *Unix*, *Mac OS*).

1.2 Установка R

R можно бесплатно скачать из сетевого архива *R* (CRAN, Comprehensive R Archive Network), который располагается по адресу *https://www.r-project.org/*.

[1] Роберт И. Кабаков R в действии / пер. с англ. А.Н. Киселева. 3-е изд. – М.: ДМК Пресс, 2023. – 768 с.

Предварительно скомпилированные загрузочные файлы доступны для *Windows*, *Mac* и *Linux*.

1.3 RStudio (Posit)

RStudio – это интегрированная среда разработки (*IDE*) с открытым исходным кодом для *R* и *Python*. Она включает в себя консоль, редактор с подсветкой синтаксиса, который поддерживает прямое выполнение кода, а также инструменты для построения графиков, выполнения отладки, управления рабочим пространством и сохранят историю.

RStudio доступна в версиях с открытым исходным кодом и коммерческих версиях и работает на настольных компьютерах (*Windows*, *Mac OS* и *Linux*).

RStudio является свободной для использования и её можно скачать с официального сайта по адресу *https://posit.co/download/rstudio-desktop*.

1.4 Пакеты R

R в базовой версии уже обладает большими возможностями по обработки и графическому представлению данных. Однако некоторые наиболее впечатляющие возможности программы реализованы в дополнительных модулях, которые можно скачать и установить. В настоящее время существует более 19880 пакетов[2]. В них заключены огромные возможности – от анализа геостатистических данных до масс-спектроскопии белков и машинного обучения.

Пакеты – это собрание функций *R*, данных и скомпилированного программного кода в определённом формате.

Для визуализации данных будут использоваться базовые возможности языка *R* и пакета семейства `{tidyverse}`, в частности, один из наиболее популярных пакетов `{ggplot2}`.

[2] *https://cran.r-project.org/web/packages/* (дата обращения: 26.07.2024)

2. Диаграмма рассеяния

2.1 Введение

Диаграмма рассеяния представляет собой график, на котором значения двух переменных изображаются в виде точек (или каких-либо других символов), размещенных на декартовой плоскости. Координаты точек представляют собой пары значений соответствующих переменных. Диаграммы рассеяния позволяют увидеть взаимосвязь между двумя переменными.

2.2 Описание набора данных

Для построения диаграммы рассеяния будем использовать пакет *{ggplot2}*, входящий в *{tidyverse}* и набор данных *mpg*, который содержит данные о моделях автомобилей, выпускавшихся каждый год в период с 1990 по 2008 годы.

Переменные, входящие в набор данных *mtcars*:

1. *manufacturer* – наименование производителя автомобиля;
2. *model* – модель автомобиля;
3. *displ* – объём двигателя, в литрах;
4. *year* – год выпуска;
5. *cyl* – количество цилиндров;
6. *trans* – тип трансмиссии;
7. *drv* – тип привода (f = передний привод, r = задний привод, 4 = полноприводная);
8. *cty* – количество миль на одном галоне в городе;
9. *hwy* – количество миль на одном галоне за городом (на шоссе);
10. *fl* – тип топлива;
11. *class* – класс автомобиля.

2.3 Загрузим и обобщим набор данных

Подключим необходимые пакеты

```
library(tidyverse)
```

Загрузим набор данных

```
mpg
#> # A tibble: 234 × 11
#>    manufacturer model      displ  year   cyl trans drv     cty   hwy fl    c
lass
#>    <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <
chr>
#>  1 audi         a4           1.8  1999     4 auto… f        18    29 p     c
omp…
#>  2 audi         a4           1.8  1999     4 manu… f        21    29 p     c
omp…
#>  3 audi         a4           2    2008     4 manu… f        20    31 p     c
```

```
omp…
#>  4 audi         a4           2    2008     4 auto… f        21    30 p     c
omp…
#>  5 audi         a4           2.8  1999     6 auto… f        16    26 p     c
omp…
#>  6 audi         a4           2.8  1999     6 manu… f        18    26 p     c
omp…
#>  7 audi         a4           3.1  2008     6 auto… f        18    27 p     c
omp…
#>  8 audi         a4 quattro   1.8  1999     4 manu… 4        18    26 p     c
omp…
#>  9 audi         a4 quattro   1.8  1999     4 auto… 4        16    25 p     c
omp…
#> 10 audi         a4 quattro   2    2008     4 manu… 4        20    28 p     c
omp…
#> # ℹ 224 more rows
```

Посмотрим на набор данных для определения, содержащихся в ней переменных и их типа.

```
glimpse(mpg)
#> Rows: 234
#> Columns: 11
#> $ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi",
"…
#> $ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro",
"…
#> $ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0,
2.…
#> $ year         <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 2
00…
#> $ cyl          <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8
, …
#> $ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "au
to…
#> $ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4",
"4…
#> $ cty          <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17,
1…
#> $ hwy          <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25,
2…
#> $ fl           <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p",
"p…
#> $ class        <chr> "compact", "compact", "compact", "compact", "compact",
"c…
```

2.4 Визуализация данных

Построим диаграмму рассеяния на основе переменных *displ* и *hwy*.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```

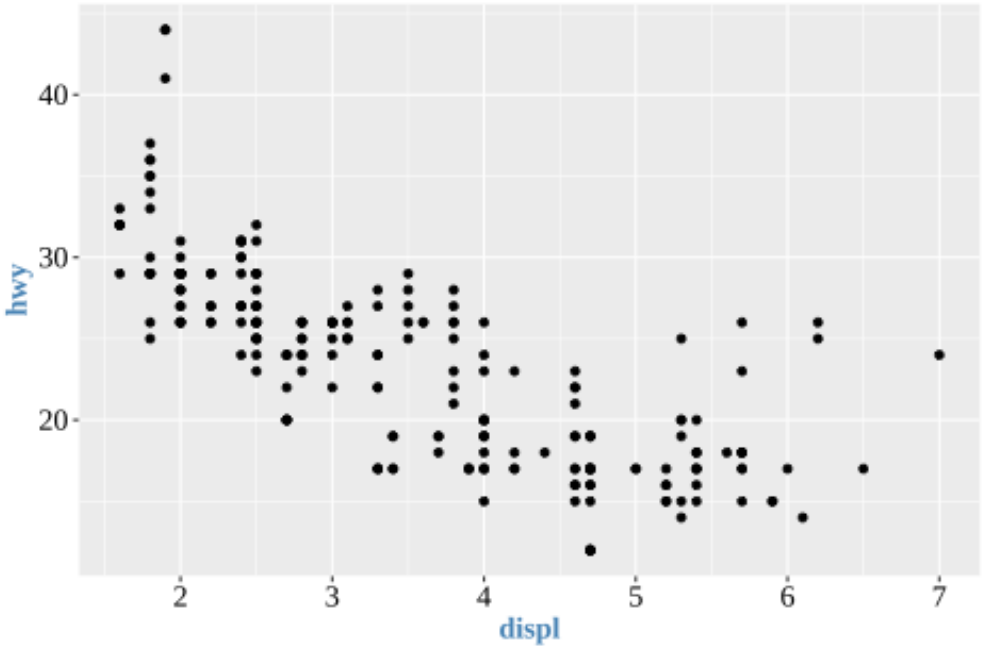

Рисунок 1– Диаграмма рассеяния

Первая строка этого кода определяет базовый слой с аргументом `data =`.

Вторая строка описывает слой `geom`, где будет использоваться точечный график.

Эстетика – это визуальные характеристики наблюдений на графике. Примерами могут служить координатные положения, форма, размер или цвет.

В случае представленного графика (*рисунок 1*) мы отображаем `displ` на оси `x` и `hwy` на оси `y`.

Чтобы завершить обсуждение, отметим ключевые моменты из представленного выше кода:

- есть один `ggplot` и один `geom_point`;
- функция `ggplot` предшествовала функции `geom_point`;
- знак плюс `+` соединяет две функции;
- `data` является аргументом спецификация `ggplot`;
- `mapping` определяет спецификацию `geom_point`.

Добавим цвет точкам (*рисунок 2*).

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```

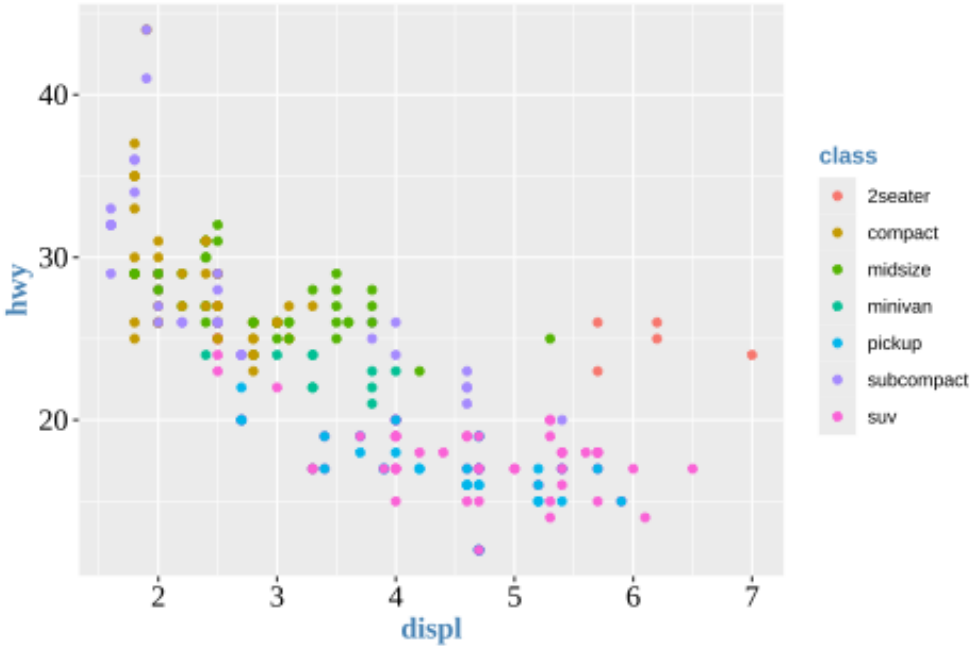

Рисунок 2 – Диаграмма рассеяния. Добавление цвета точкам

Представленная диаграмма позволяет сделать выводы о наблюдениях и их группировании.

2.5 Категориальные переменные

Возможно ли получить больше информации из категориальных переменных?

Используя функцию *mutate* создадим новую категориальную переменную *japen_make_car*, которая будет являться логической переменной и принимать два значения, *FALSE* и *TRUE*, то есть является ли производитель компанией из японии, то есть *honda*, *nissan*, *subaru*, *toyota*. Новый набор данных сохраним в *no_sports_cars*.

```
japan_cars <- mpg |>
  mutate(japanese_make_car =
           manufacturer %in% c("honda", "nissan", "subaru", "toyota"))
```

Построим диаграмму рассеяния с учетом производства автомобиля японскими компаниями (*рисунок 3*).

```
ggplot(data = japan_cars) +
  geom_point(
    mapping = aes(x = displ, y = hwy, color = japanese_make_car)
    )
```

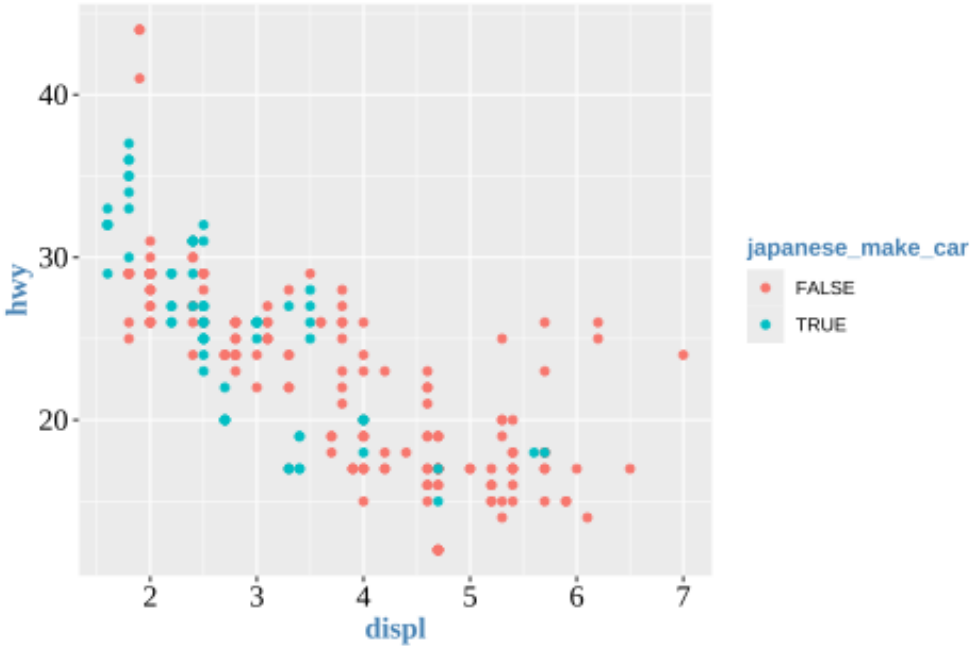

Рисунок 3 – Производства автомобилей японскими и не японскими компаниями

На основе представленной диаграммы можем наблюдать тенденцию к снижению выпуска автомобилей с ростом объёма двигателя.

Построим новую диаграмму и в качестве категориальной используем переменную `cyl` (*рисунок 4*).

```
ggplot(data = japan_cars) +
  geom_point(
    mapping = aes(x = cty, y = hwy, color = cyl)
    )
```

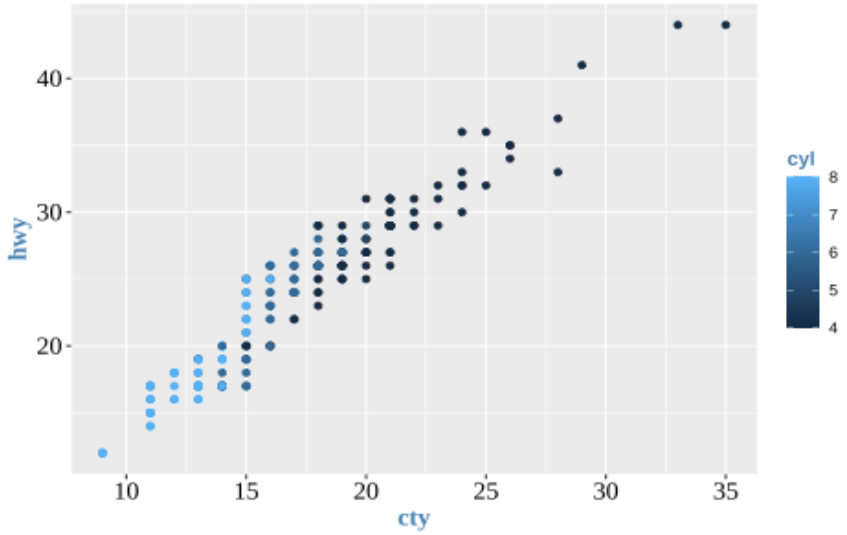

Рисунок 4– Диаграмма рассеяния. Количественная переменная

На основании представленной диаграммы можно отметить, что переменная `cyl` является количественной и содержит значения 4, 5, 6, 7 или 8 и ничего между ними. Как сообщить данную информацию `ggplot`?

Для этого нужно рассматривать значение *cyl* как фактор, который является утверждение, что переменная является категориальной. Функция, которую нужно использовать для преобразования называется *as_factor()*.

Применим данную функцию к переменной и снова построим график (*рисунок 5*), только в качестве переменной, которая будет располагаться по оси *x*, будем использовать *cty* (количество миль на одном галоне в городе).

```
ggplot(data = japan_cars) +
  geom_point(mapping = aes(x = cty, y = hwy, color = as_factor(cyl)))
```

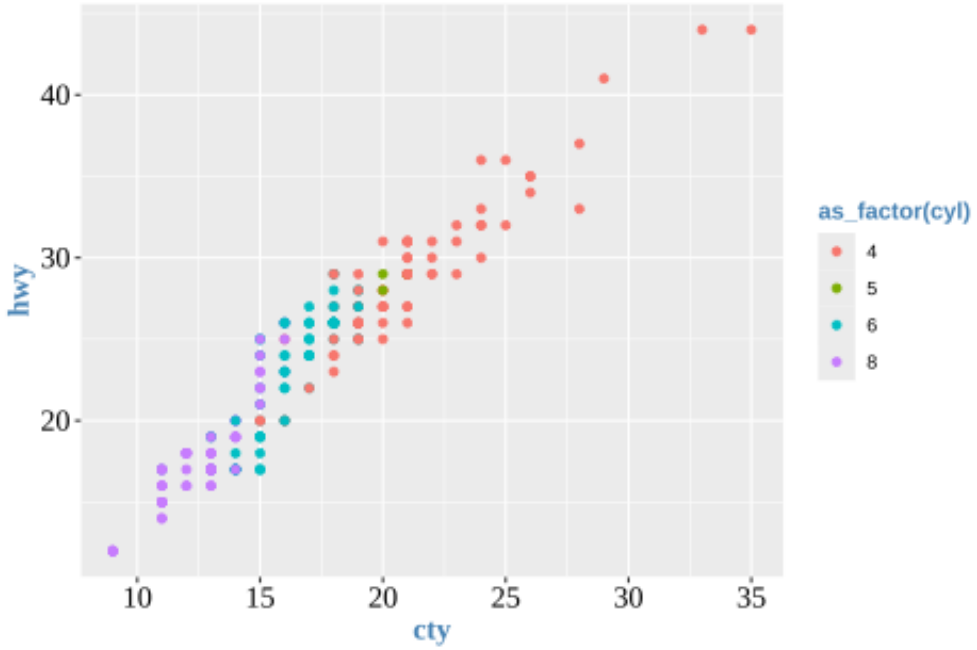

Рисунок 5– Диаграмма рассеяния. Применение факторной переменной

На полученном графике можно наблюдать точки выброса, как минимальные, так и максимальные. Посмотрим, на что указывают данные точки.

```
japan_cars |>
  filter(cty < 10 | cty >32.5) |>
  relocate(cty, .after = year) |>
  relocate(hwy, .before = cyl)
#> # A tibble: 7 × 12
#>   manufacturer model       displ  year   cty   hwy   cyl trans drv   fl    c
lass
#>   <chr>        <chr>       <dbl> <int> <int> <int> <int> <chr> <chr> <chr> <
chr>
#> 1 dodge        dakota pic…   4.7  2008     9    12     8 auto… 4     e     p
ick…
#> 2 dodge        durango 4wd   4.7  2008     9    12     8 auto… 4     e     s
uv
#> 3 dodge        ram 1500 p…   4.7  2008     9    12     8 auto… 4     e     p
ick…
#> 4 dodge        ram 1500 p…   4.7  2008     9    12     8 manu… 4     e     p
ick…
#> 5 jeep         grand cher…   4.7  2008     9    12     8 auto… 4     e     s
uv
#> 6 volkswagen   jetta         1.9  1999    33    44     4 manu… f     d     c
```

```
omp…
#> 7 volkswagen   new beetle    1.9  1999    35    44     4 manu… f     d     s
ubc…
#> # ℹ 1 more variable: japanese_make_car <lgl>
```

На основе полученного среза набора данных можно отметить, что всего 7 точек выброса и это “dodge” и “jeep”, крайние левые точки и “volkswagen” – крайние правые точки.

2.6 Непрерывная переменная

Ранее мы уже экспериментировали с переменной `cyl`. Посмотрим, что можно почерпнуть при визуализации, если использовать непрерывную переменную. Попробуем создать график с переменной `cty`, которая описывает количество прохождения миль на одном галоне в городе (*рисунок 6*).

```
ggplot(data = japan_cars) +
  geom_point(mapping = aes(x = displ, y = hwy, color = cty))
```

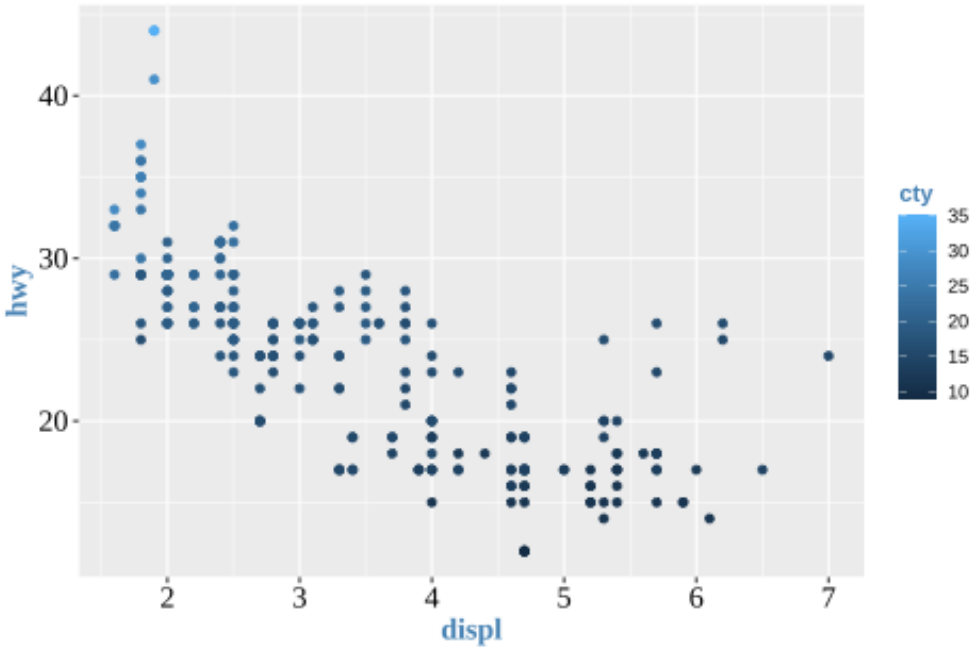

Рисунок 6– Диаграмма рассеяния. Применение непрерывной переменной

Синий градиент затрудняет просмотр изменения цвета. Будем использовать другой градиент. Для этого применим функцию `scale_color_gradient` (*рисунок 7*).

```
ggplot(data = japan_cars) +
  geom_point(mapping = aes(x = displ, y = hwy, color = cty)) +
  scale_color_gradient(low = "yellow2", high = "blue")
```

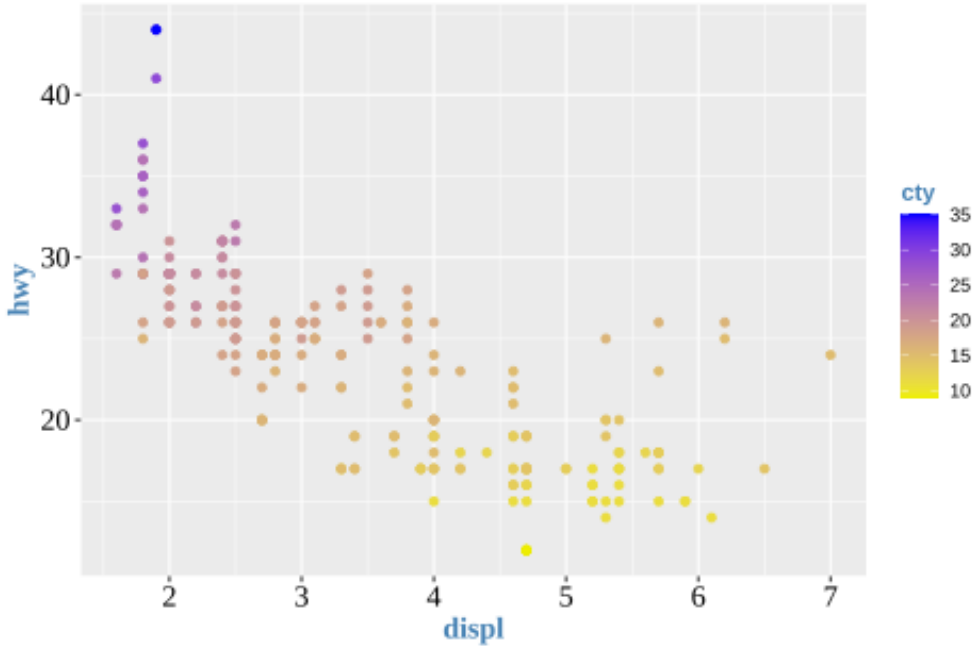

Рисунок 7– Диаграмма рассеяния. Применение непрерывной переменной и градиента

2.7 Другие возможности построения диаграммы рассеяния

Использование панелей

Вместо использования цвета для описания точек можно использовать другой инструмент пакета *{ggplot2}*, а именно панели (*facet*), которые разбивают график на несколько графиков, по одному для каждой категориальной переменно.

Используем данный инструмент, который представлен в виде функции *facet_wrap* пакета *{ggplot2}* (*рисунок 8*).

```
ggplot(data = japan_cars) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ as_factor(cyl), nrow = 2)
```

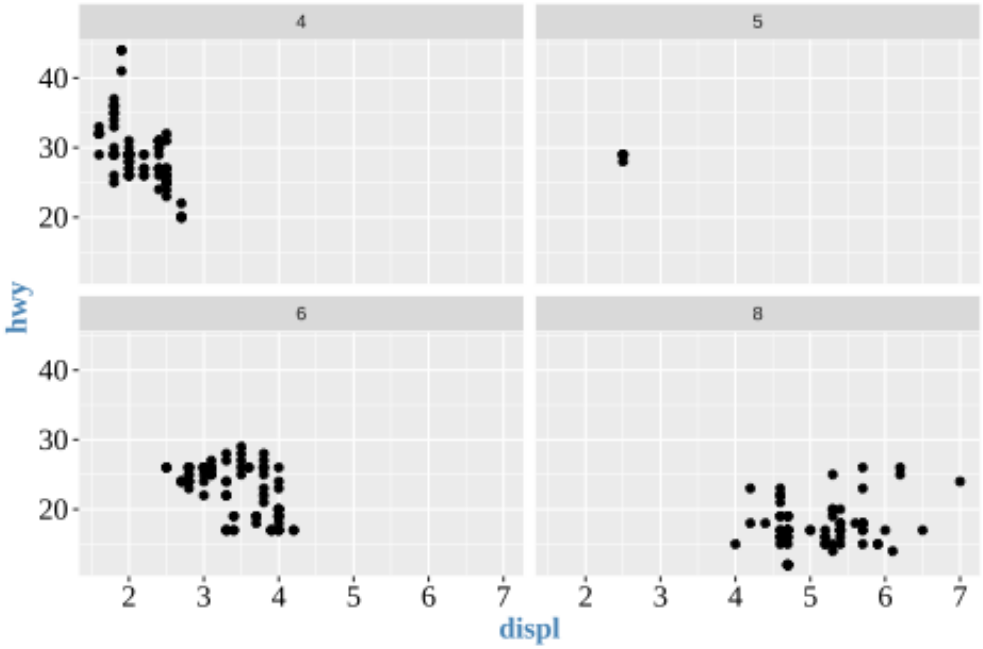

Рисунок 8 – Применение функции *facet_wrap()* в сочетании с аргументом *nrow = 2*

Использование значков и формы представления точке

Пакет *{ggplot}* позволяет по-разному представлять точки диаграммы рассеяния.

Использовать параметр прозрачность (*рисунок 9*).

```
ggplot(japan_cars) +
  geom_point(aes(x = cty, y = hwy, alpha = class))
```

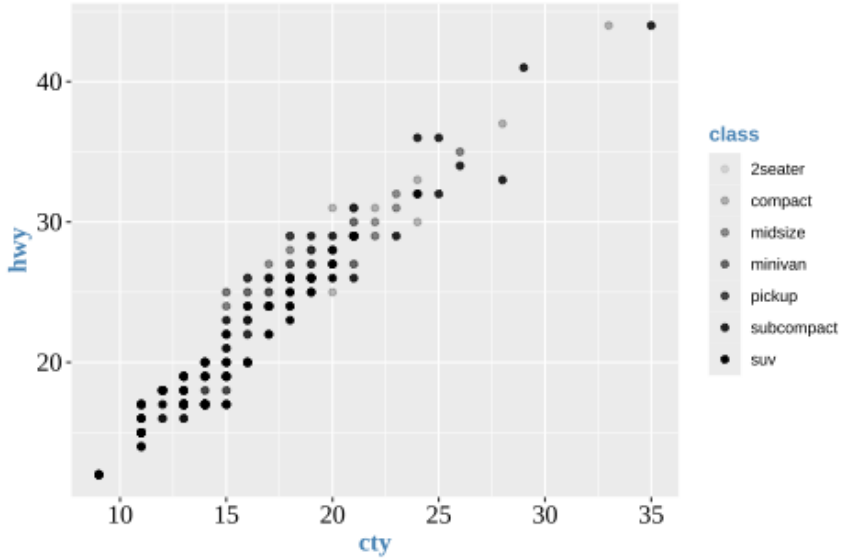

Рисунок 9 – Диаграмма рассеяния. Применение параметра прозрачность

Использование прозрачности и формы точек (*рисунок 10*).

```
ggplot(japan_cars) +
  geom_point(aes(x = cty, y = hwy, alpha = class, shape = class))
```

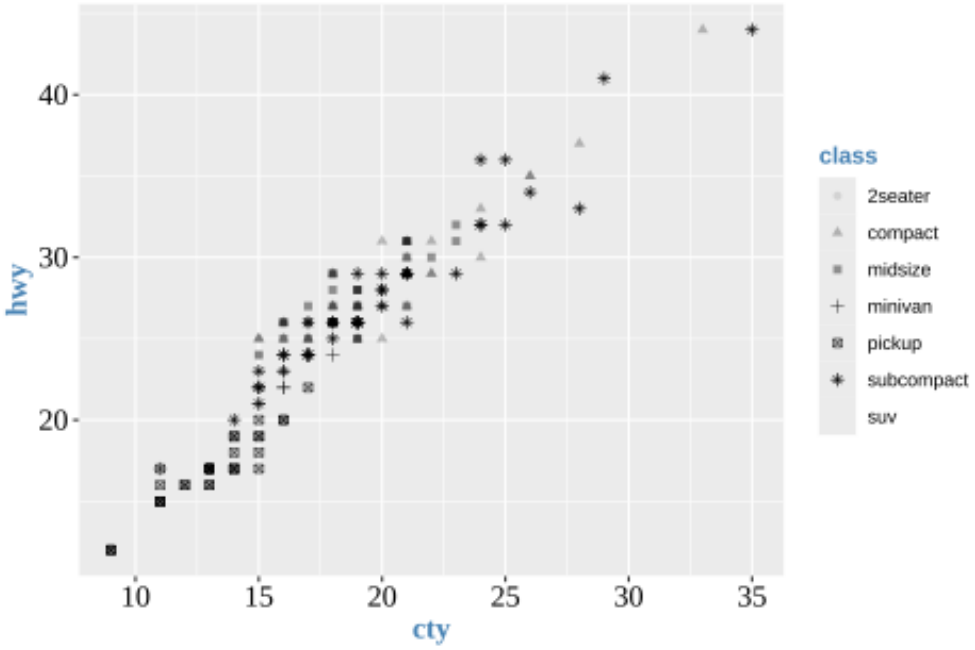

Рисунок 10 – Диаграмма рассеяния. Применение аргумента, определяющего форму точек

При использовании эстетики *shape* и *colour* для одной и той же переменной, легенды объединяются в одну (*рисунок 11*).

```
ggplot(japan_cars) +
  geom_point(aes(x = cty, y = hwy, shape = class, colour = class))
```

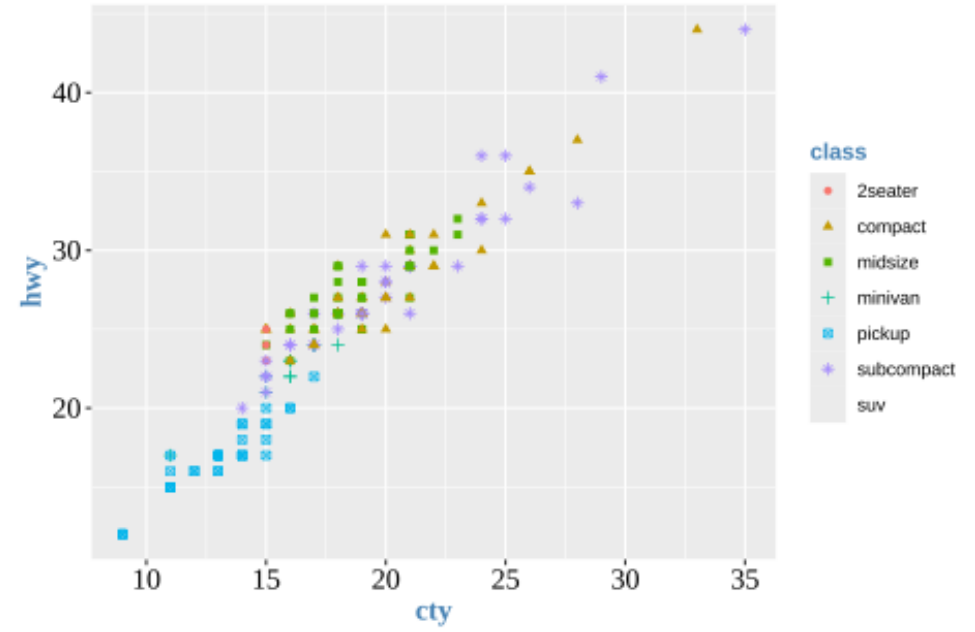

Рисунок 11 – Диаграмма рассеяния. Применение аргумента, определяющего форму точек

Задание

1. Использовать функцию *facet_grid()* для построения категоризированного графика.
2. Построить диаграмму рассеяния на основе набора данных *mpg* используя другие переменные.

3. Линейный график

3.1 Введение

Линейная диаграмма – это тип графика, который может показывать динамику по одному или нескольким показателям. Она позволяет соединить несколько отдельных точек данных в единую непрерывную прогрессию. Используется для визуализации непрерывных переменных, которые могут показывать тренды, сезонность и изменения во времени.

3.2 Описание набора данных

Создадим набор данных для реализации возможности построения линейных графиков.

```
df_study <- tibble(x = seq(-3, 3, 0.25),
                   g1 = 3 * x - x * x + 20,
                   g2 = 4 * x - 10,
                   g3 = -x + 40 * sin(x),
                   g4 = x*x*x - 2*x)
df_study
#> # A tibble: 25 × 5
#>        x    g1    g2     g3      g4
#>    <dbl> <dbl> <dbl>  <dbl>   <dbl>
#>  1 -3     2      -22  -2.64 -21
#>  2 -2.75  4.19   -21 -12.5  -15.3
#>  3 -2.5   6.25   -20 -21.4  -10.6
#>  4 -2.25  8.19   -19 -28.9   -6.89
#>  5 -2    10      -18 -34.4   -4
#>  6 -1.75 11.7    -17 -37.6   -1.86
#>  7 -1.5  13.2    -16 -38.4   -0.375
#>  8 -1.25 14.7    -15 -36.7    0.547
#>  9 -1    16      -14 -32.7    1
#> 10 -0.75 17.2    -13 -26.5    1.08
#> # ℹ 15 more rows
```

3.3 Построение линейной диаграммы

Для визуализации набора данных необходимо сделать его «длинным». Используем соответствующие функции пакета *{tidyr}*.

```
df_study_long <- df_study |>
  pivot_longer(c(g1, g2, g3, g4),
               names_to = "fun",
               values_to = "y") |>
  select(x, y, fun)
```

Визуализируем все диаграммы на одном графике (*рисунок 12*)

```
df_study_long |>
  ggplot() +
  geom_line(mapping = aes(x = x, y = y, group = fun))
```

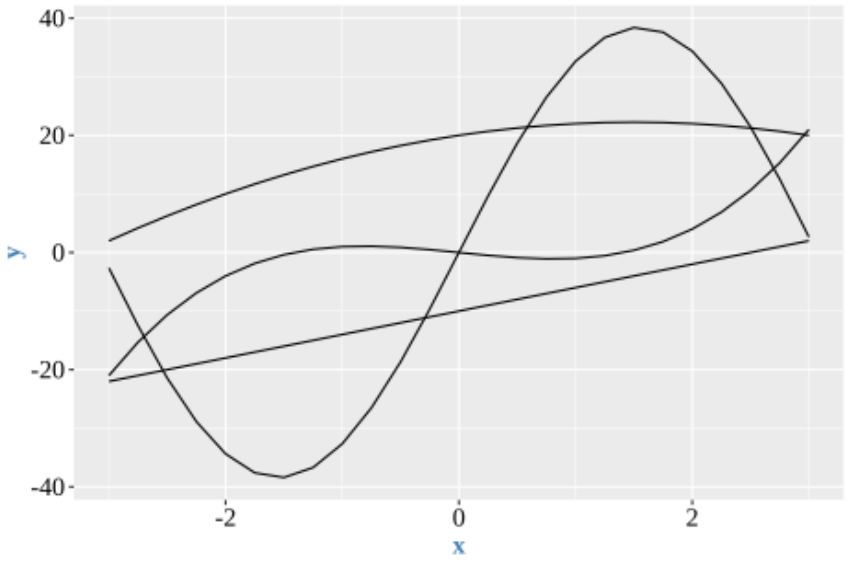

Рисунок 12 – Линейный график

Немного изменим график. Покрасим каждую линию в определённый цвет (*рисунок 13*).

```
df_study_long |>
  ggplot() +
  geom_line(mapping = aes(x = x, y = y, colour = fun))
```

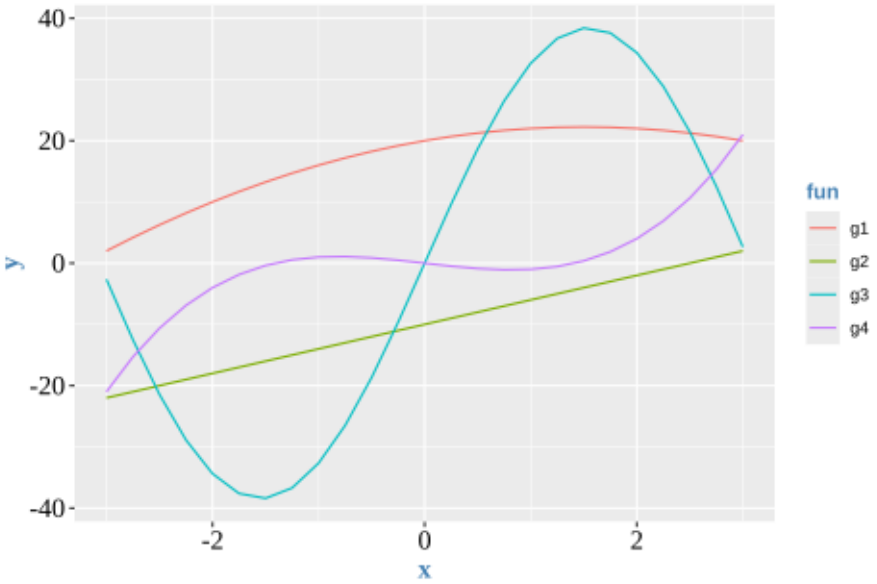

Рисунок 13 – Линейный график

Далее, поменяем тип линии и дадим название координатным осям (*рисунок 14*).

```
df_study_long |>
  ggplot() +
  geom_line(mapping = aes(x = x, y = y, colour = fun),
```

```
            size = 1.5, linetype = "twodash") +
  labs(x = "Значения x",
       y = "Значения y")
```

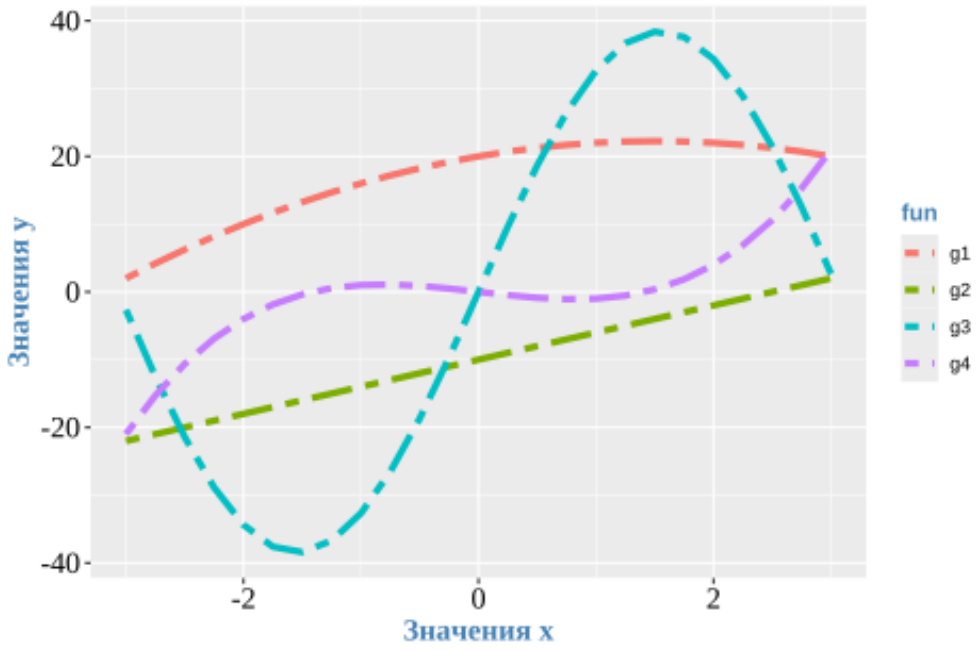

Рисунок 14 – Линейный график. Тип линии

3.4 Функция *geom_smooth*

Построим графики с использованием функции *geom_smooth* (*рисунок 15*).

```
df_study_long |>
  ggplot() +
  geom_smooth(aes(x = x, y = y, colour = fun))
```

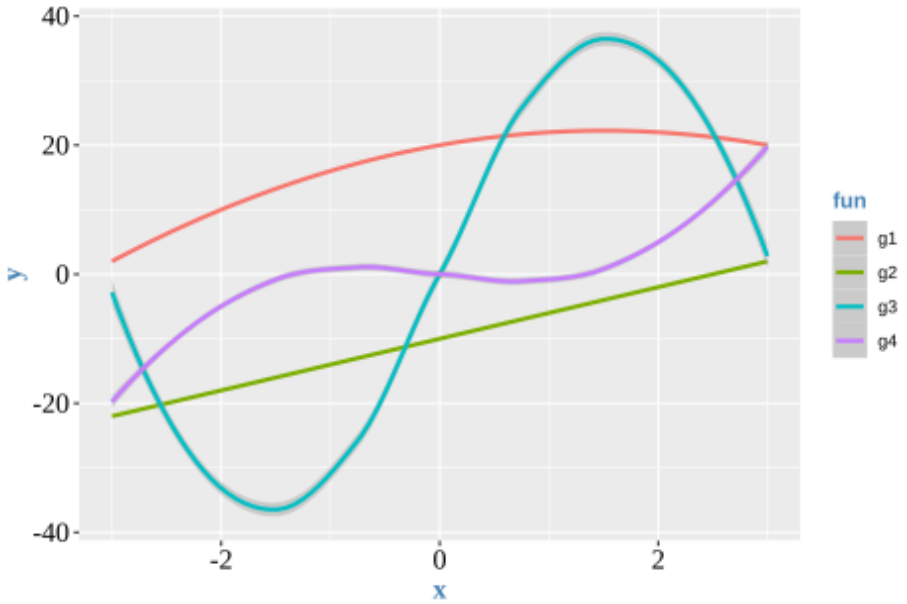

Рисунок 15 – Линейный график: функция *geom_smooth*

Графики немного отличаются от тех, которые были построены выше. Они выглядят как правильные кривые линии.

Объединим две функции (*рисунок 16*).

```
df_study_long |>
  ggplot() +
  geom_smooth(aes(x = x, y = y, colour = fun), se = FALSE) +
  geom_line(aes(x = x, y = y, colour = fun))
```

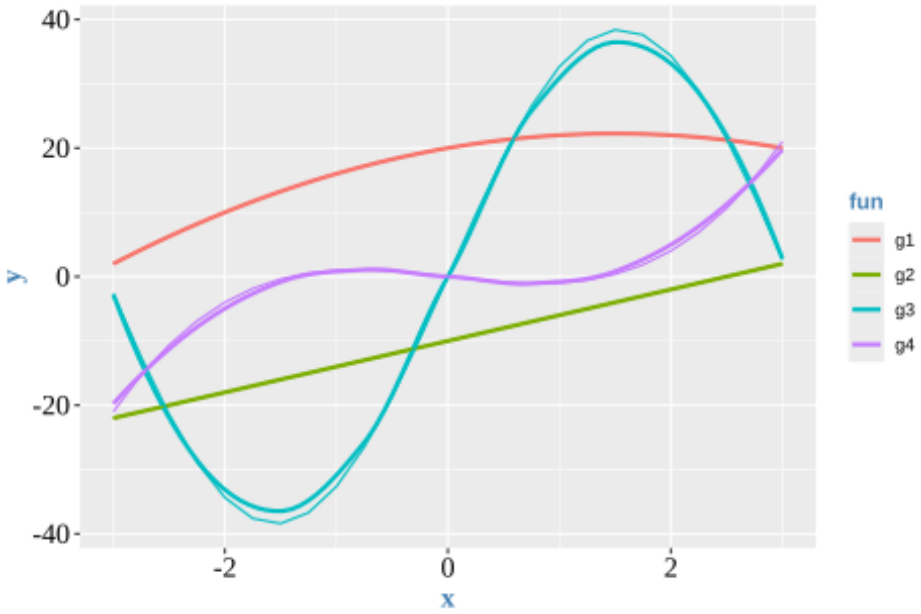

Рисунок 16 – Линейный график: функция *geom_smooth*

Можно заметить некоторые отклонения на получившейся диаграмме.

3.5 Объединение линейной и точечной диаграмм

В соответствии с философией, на основе которой разработан пакет *{ggplot2}*, одна диаграмма может содержать несколько графиков.

Для примера воспользуемся набором данных *mpg* и построим графики (*рисунок 17*).

```
mpg |>
  ggplot(aes(x = displ, y = hwy, colour = class)) +
  geom_point() +
  geom_smooth(se = FALSE, method = "lm")
```

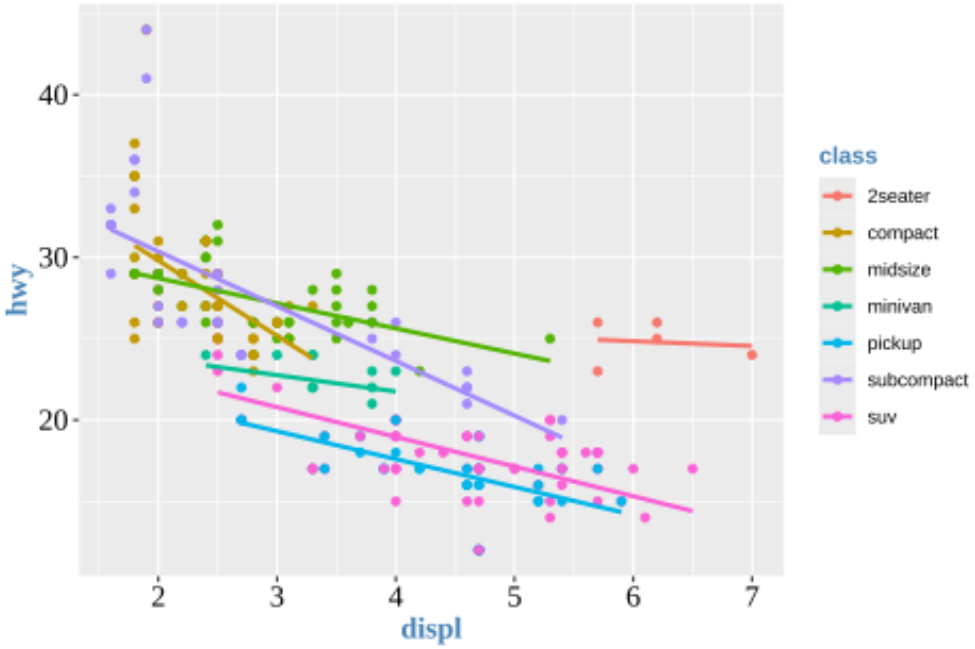

Рисунок 17 – Линейный график: функция *geom_smooth* и *geom_point*

Задание

1. дать название осям;
2. переименовать легенду;
3. оставить на диаграмме только одну линию.

4. Столбиковая диаграмма

4.1 Введение

Столбиковые диаграммы – это общепринятый визуальный инструмент для распределений как количественных, так и качественных переменных. В случае с качественной переменной – это отображения одной-единственной категориальной переменной, то есть изображение частот встречаемости каждого из её уровней (абсолютные или относительные). Данный вид диаграмм можно часто встретить в разнообразной литературе. Категории перечислены на оси *x*, а частоты или доли – на оси *y*.

Столбиковая диаграмма напоминает гистограмму; на столбиковой диаграмме ось *x* представляет разные категории факторной переменной, тогда как на гистограмме ось *x* представляет значения одной-единственной переменной на числовой шкале. На гистограмме столбики обычно изображаются впритык друг к другу, а разрывы указывают на то, что значения в данных отсутствовали. На столбиковой диаграмме столбики отображаются отдельно друг от друга.

4.2 Описание набора данных

В качестве анализируемых данных будем используем встроенный набор данных `mtcars`, который содержит измерения по 11 атрибутам для 32 различных автомобилей (для моделей 1973–1974 годов).

Переменные, входящие в набор данных *mtcars*:

1. *mpg* – количество миль на одном галлоне;
2. *cyl* – количество цилиндров;
3. *disp* – рабочий объем двигателя в зависимости от окружности цилиндра, глубины и общего количества цилиндров. Этот показатель дает хорошее представление об общей мощности, которую может генерировать двигатель;
4. *hp* – мощность двигателя (мощность в лошадиных силах измеряет теоретическую выходную мощность двигателя; примечательно, что полная мощность относится к работе двигателя в изолированной среде вне какого-либо конкретного транспортного средства. Установленные в автомобиле выхлопные системы, карбюратор, генератор переменного тока, системы питания и т. д. – все это влияет на мощность, которая фактически поступает в приводную систему. Более того, согласно онлайн-источникам, в начале 1970-х годов изменения в законодательстве повлияли на то, как измерялась общая мощность двигателя. Поскольку этот набор данных относится

к началу-середине 1970-х годов, неясно, можно ли использовать показатели *hp* в качестве надежных средств сравнения мощности двигателей разных моделей, так как неясно, как отчитывались производители);

5. *draft* – передаточное отношение задней оси (передаточное число заднего моста показывает количество оборотов приводного вала на каждый оборот колесной оси. Например, транспортное средство с высоким передаточным числом обеспечивало бы больший крутящий момент и, следовательно, большую тяговую способность. Конфигурация трансмиссии часто может влиять на передаточное число, установленное производителем);
6. *wt* – масса (фунт/1000 – общий вес автомобиля на 1000 фунтов);
7. *qsec* – время прохождения 1/4 мили;
8. *vs* – V/S (двоичная переменная, показывающая конфигурацию цилиндра двигателя в виде V-образной формы (vs=0) или прямой линии (vs=1). Конфигурация предлагает компромиссы в мощности / крутящем моменте, конструктивном использовании с точки зрения пространства /размера двигателя и производительности или центра тяжести транспортного средства);
9. *am* – вид коробки передач (двоичная переменная, сигнализирующая о том, имеет ли транспортное средство автоматическую (am=0) или ручную (am=1) конфигурацию коробки передач);
10. *gear* – число передач переднего хода (механические коробки передач имеют либо 4, либо 5 передач переднего хода; автоматические – либо 3, либо 4);
11. *carb* – количество карбюраторов (количество цилиндров карбюратора. Автомобиль может иметь несколько физических карбюраторов, но это встречается реже; этот показатель представляет собой сумму количества карбюраторов и количества цилиндров внутри карбюратора).

4.3 Загрузим и обобщим набор данных

Так как рассматриваемый набор является встроенным, загрузить его можно с помощь следующей команды

```
data(mtcars)
```

После этого необходимо посмотреть правильность загрузки набора данных. Мы можем взглянуть на первые шесть строк используя функцию `head()`, а для последних 6 строк – `tail()`.

```
head(mtcars)
#>                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
#> Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
#> Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
#> Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
#> Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
```

```
#> Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
#> Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Проанализируем, какие типы данных содержит таблица, то есть предварительно оценим возможности применения разнообразных статистик и графического представления информации.

Для этого выполним код, описанный ниже:

```
str(mtcars)
#> 'data.frame':    32 obs. of  11 variables:
#>  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
#>  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
#>  $ disp: num  160 160 108 258 360 ...
#>  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
#>  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
#>  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
#>  $ qsec: num  16.5 17 18.6 19.4 17 ...
#>  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
#>  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
#>  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
#>  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

Определим размер набора данных, то есть количество столбцов и строк (наблюдений).

```
dim(mtcars)
#> [1] 32 11
```

Видим, что набор данных состоит из 32 записей и 11 столбцов, что было отмечено раньше.

Используя функцию *names()* отобразим имена столбцов набора данных *mtcars*

```
names(mtcars)
#>  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
#> [11] "carb"
```

С помощью функции *summary()* вычислим сводную статистику для набора данных.

Для количественных переменных функция рассчитывает:

– Мин.: минимальное значение;

– 1st Qu: значение 1-го квартиля (25-го процентиля);

– Медиана: среднее значение;

– 3-й Qu: значение 3-го квартиля (75-го процентиля);

– Макс.: максимальное значение.

Для категориальных или факторных переменных будет подсчитано количество наблюдений для каждого уровня.

```
summary(mtcars)
#>       mpg             cyl             disp             hp
#>  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0
#>  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5
```

```
#>  Median :19.20   Median :6.000   Median :196.3   Median :123.0
#>  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7
#>  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0
#>  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0
#>       drat             wt             qsec             vs
#>  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000
#>  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000
#>  Median :3.695   Median :3.325   Median :17.71   Median :0.0000
#>  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
#>  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
#>  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
#>        am              gear            carb
#>  Min.   :0.0000   Min.   :3.000   Min.   :1.000
#>  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
#>  Median :0.0000   Median :4.000   Median :2.000
#>  Mean   :0.4062   Mean   :3.688   Mean   :2.812
#>  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
#>  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

4.4 Простые столбиковые диаграммы

Столбиковые диаграммы отражают распределение (частоту разных значений) категориальной переменной в виде вертикальных или горизонтальных столбиков. Самый простой способ применить для построения столбиковой диаграммы – применить базовую функцию *barplot()*

barplot(cyl)

где *cyl* – это вектор или матрица.

Если аргумент (переменная) *cyl* – вектор, его значение определяет высоту столбцов, и создается вертикальная столбиковая диаграмма. Добавление параметра *horiz=TRUE* позволяет получить горизонтальную столбиковую диаграмму. Также можно добавить параметры, создающие или меняющие надписи.

Функция *barplot()* содержит следующие аргументы

```
## Default S3 method:

barplot(height, width = 1, space = NULL,
        names.arg = NULL, legend.text = NULL, beside = FALSE,
        horiz = FALSE, density = NULL, angle = 45,
        col = NULL, border = par("fg"),
        main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
        xlim = NULL, ylim = NULL, xpd = TRUE, log = "",
        axes = TRUE, axisnames = TRUE,
        cex.axis = par("cex.axis"), cex.names = par("cex.axis"),
        inside = TRUE, plot = TRUE, axis.lty = 0, offset = 0,
        add = FALSE, ann = !add && par("ann"), args.legend = NULL, ...
        )

## S3 method for class 'formula'
barplot(formula, data, subset, na.action,
        horiz = FALSE, xlab = NULL, ylab = NULL, ...
        )
```

Полное описание аргументов можно посмотреть в *RStudio* при нажатии кнопки *F1* или найти описание в *Интернете*.

Построим столбиковую диаграмму в зависимости от количества цилиндров. Используя функцию `unique()` можно разделить все автомобили входящие в набор данных на три группы в соответствии с количеством цилиндров: 6, 4, 8.

В таблице данных (наборе данных) по видам автомобилей в переменной `cyl` указано количество цилиндров автомобилей

```
counts <- table(mtcars$cyl)
counts
#>  4  6  8
#> 11  7 14
```

Здесь видно, что в наборе данных присутствует 11 автомобилей с 4 цилиндрами, 7 – с 6 и 14 – с 8.

Построим простые вертикальную и горизонтальную столбиковую диаграммы (*рисунок 18 и рисунок 19*)

```
barplot(counts,
        main = "Вертикальная столбиковая диаграмма",
        xlab = "Количество цилиндров, шт",
        ylab = "Частота",
        font = 9
        )
```

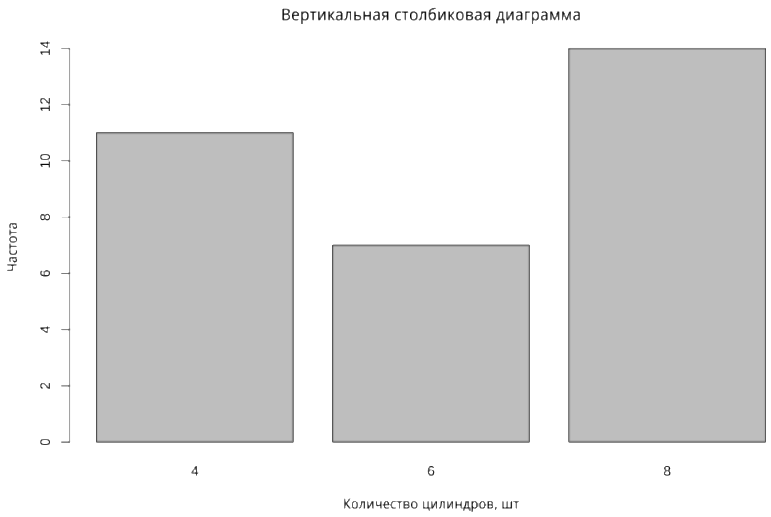

Рисунок 18 – Вертикальная столбиковая диаграмма

```
barplot(counts,
        main = "Горизонтальная столбиковая диаграмма",
        xlab = "Количество цилиндров, шт",
        ylab = "Частота",
        horiz = TRUE,
        font = 9
        )
```

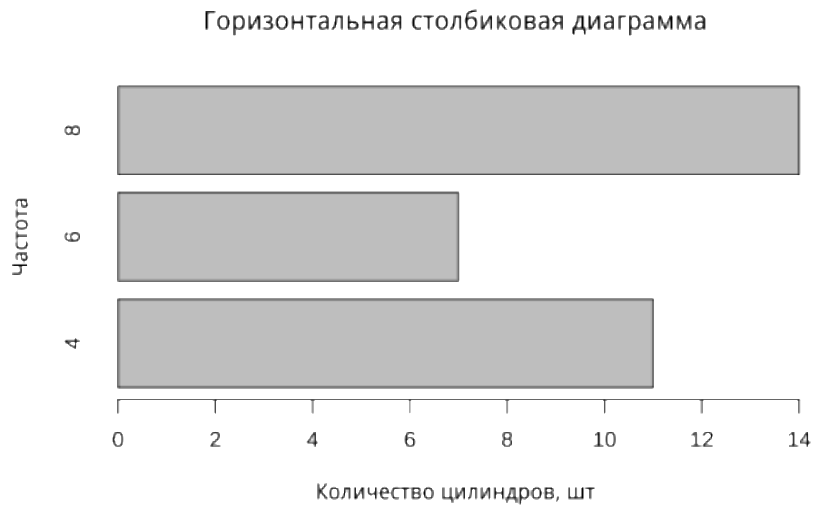

Рисунок 19 – Горизонтальная столбиковая диаграмма

Если категориальная переменная, которую необходимо изобразить, представляет собой фактор или упорядоченный фактор, можно быстро создать вертикальную или горизонтальную диаграмму с помощью функции `plot()` – *(рисунок 20).*

```
mtcars.my <- mtcars
mtcars.my$cyl.my <- factor(mtcars.my$cyl,
                           levels = c(4, 6, 8),
                           labels = c("Четыре", "Шесть", "Восемь"))
opar <- par(no.readonly = TRUE)
par(mfrow=c(1,2))
plot(mtcars.my$cyl.my, font=9)
plot(mtcars.my$cyl.my, font=9, horiz=TRUE)
par(opar)
```

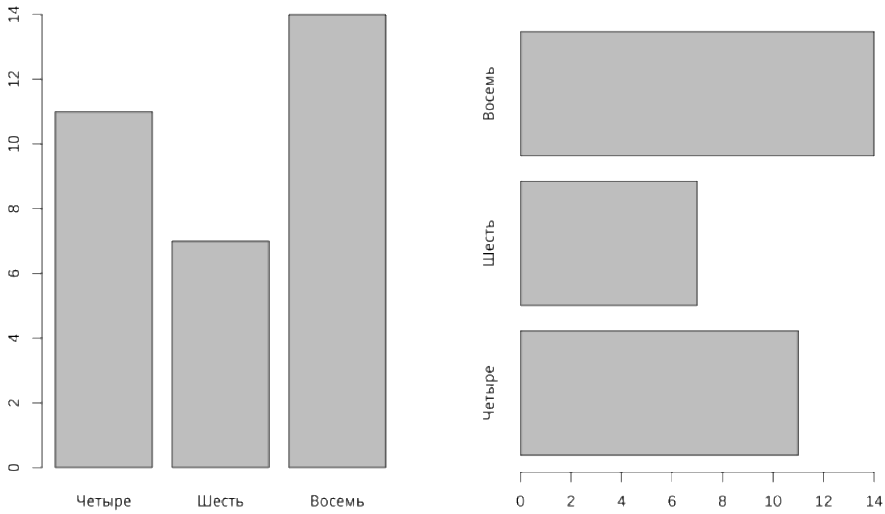

Рисунок 20 – Столбиковая диаграмма

4.5 Столбиковые диаграммы: составные и с группировкой

Если `cyl` – матрица, для неё с помощью функции `barplot()` будет построена составная столбиковая диаграмма или столбиковая диаграмма с группировкой. Если `based=FALSE` (по умолчанию), то каждой переменной матрицы соответствует столбец диаграммы.

Рассмотрим данные об автомобилях по количеству цилиндров и типу коробки передач.

```
counts <- table(mtcars$cyl, mtcars$am)
counts
#>
#>      0  1
#>   4  3  8
#>   6  4  3
#>   8 12  2
```

Первая строка показывает наличие трансмиссии: *0 – автоматическая, 1 – ручная*.

Первый столбец показывает количество цилиндров: *4 – четыре цилиндра, 6 – шесть, 8 – восемь*.

Полученные результаты можно представить в виде столбиковой диаграммы, составной или с группировкой *(рисунок 21)*

```
opar <- par(no.readonly = TRUE)
par(mfrow=c(1,2))
barplot(counts,
        main = "Составная столбикова диаграмма",
        xlab = "Тип корбоки передач",
        ylab = "Частота",
        col = c("red", "green", "blue"),
        legend = rownames(counts),
        font=9)
barplot(counts,
        main = "Столбиковая диаграмма с группировкой",
        xlab = "Тип корбоки передач",
        ylab = "Частота",
        col = c("red", "green", "blue"),
        legend = rownames(counts),
        beside = TRUE,
        font=9)
par(opar)
```

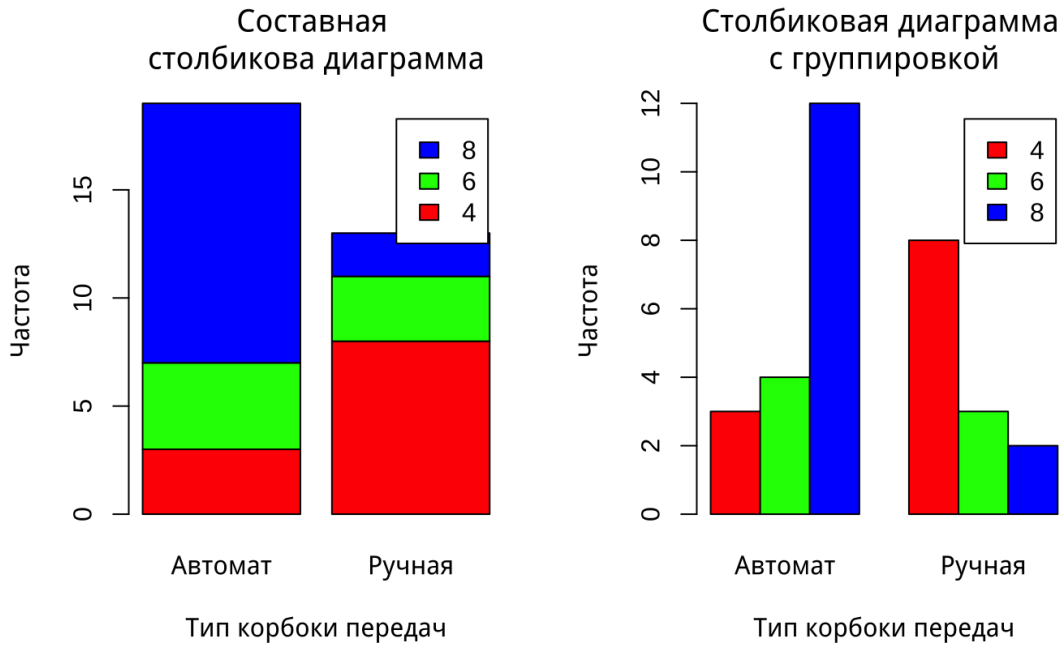

Рисунок 21 – Столбиковые диаграммы: составные и с группировкой

Недостатком данных диаграмм является то, что тип коробки представлен в виде цифры 0 и 1. Представим их виде строки, которая будет понятна человеку при анализе графика, а также отсортируем количество цилиндров от меньшего к большему *(рисунок 22)*.

```
mtcars2 <- within(mtcars, {
  vs <- factor(vs, labels = c("V", "S"))
  am <- factor(am, labels = c("Автомат", "Ручная"))
  cyl  <- ordered(cyl)
  gear <- ordered(gear)
  carb <- ordered(carb)
})

counts <- table(mtcars2$cyl, mtcars2$am)
counts
#>
#>     Автомат Ручная
#>   4       3      8
#>   6       4      3
#>   8      12      2

opar <- par(no.readonly = TRUE)
par(mfrow=c(1,2))
barplot(counts,
        main = "Составная столбиковая диаграмма",
        xlab = "Тип корбоки передач",
        ylab = "Частота",
        col = c("red", "green", "blue"),
        legend = rownames(counts),
        font = 9)
barplot(counts,
        main = "Столбиковая диаграмма с группировкой",
```

```
        xlab = "Тип корбоки передач",
        ylab = "Частота",
        col = c("red", "green", "blue"),
        legend = rownames(counts),
        beside = TRUE,
        font = 9)
par(opar)
```

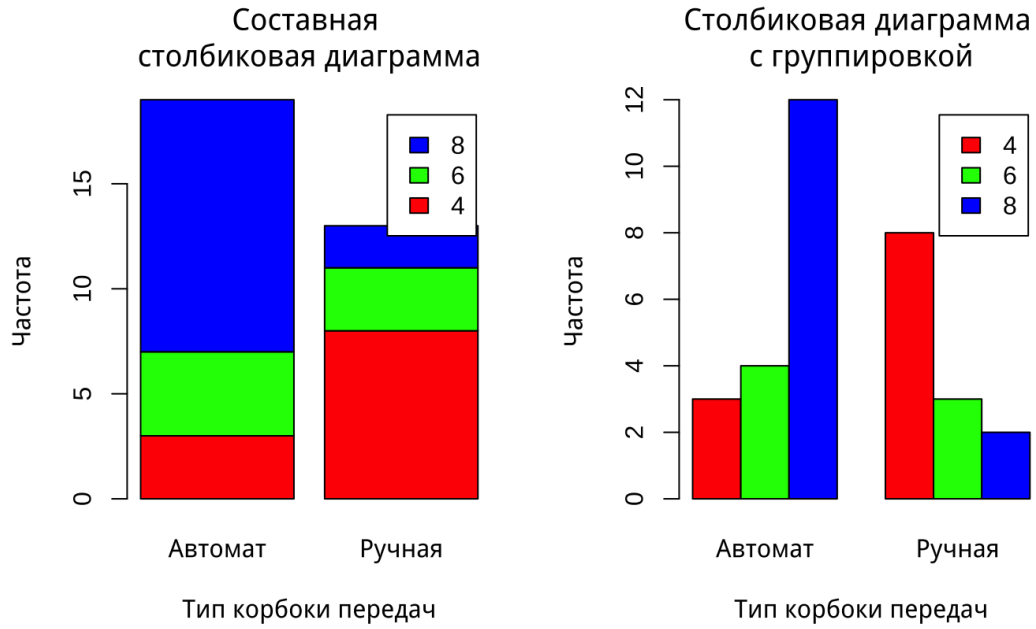

Рисунок 22 – Столбиковые диаграммы: составные и с группировкой

4.6 Оптимизация столбиковых диаграмм

Параметр font (шрифт)

Аргумент `font` – шрифт. Представлен в виде целого числа.

Параметр `font` может принимать любое целое число от 1 до 128 и от 160 до 254 *(рисунок 23)*.

```
barplot(counts,
        main = "Составная столбиковая диаграмма",
        xlab = "Тип корбоки передач",
        ylab = "Частота",
        col = c("red", "green", "blue"),
        legend = rownames(counts),
        font = 7
        )
```

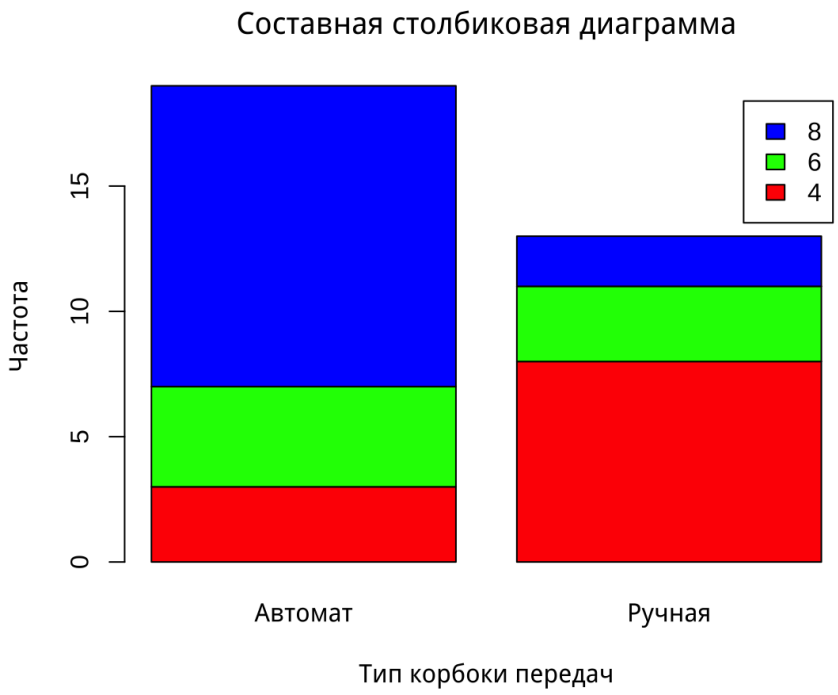

Рисунок 23 – Столбиковая диаграмма: шрифты

Размер символов

Размер символов задаётся при помощи аргумента `cex.names` (имена столбцов) и `cex.axis` (названия осей). По умолчанию равен 1.

Ширина линии обводки символа задаётся аргументом `lwd` *(рисунок 24).*

```
barplot(counts,
        main = "Столбиковая диаграмма ",
        xlab = "Тип корбоки передач",
        ylab = "Частота",
        col = c("red", "green", "blue"),
        legend = rownames(counts),
        font = 9,
        cex.names = 0.5,
        lwd = 3
        )
```

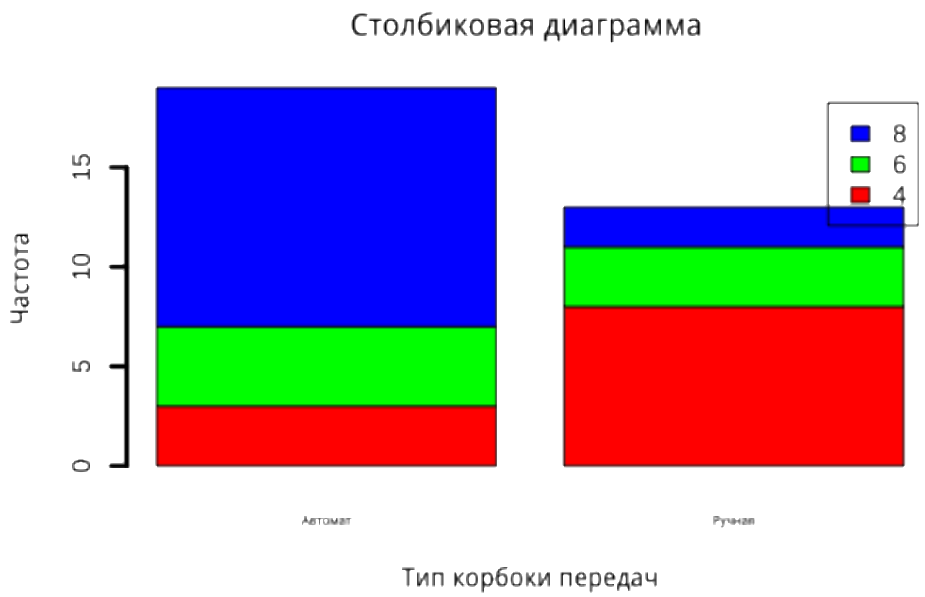

Рисунок 24 – Столбиковая диаграмма: символы и толщина линий обводки

Параметры xlab и ylab

Параметры `xlab` и `ylab` служат для указания названий осей *X* и *Y* соответственно.

Параметры axes и ann

Эти два параметра контролируют отображение осей и их названий соответственно.

Каждый из них может принимать одно из двух возможных значений – `TRUE` или `FALSE` *(рисунок 25).*

```
barplot(counts,
        main = "Столбиковая диаграмма",
        xlab = "Тип корбоки передач",
        ylab = "Частота",
        col = c("red", "green", "blue"),
        legend = rownames(counts),
        font = 7,
        axes = FALSE,
        ann = FALSE
        )
```

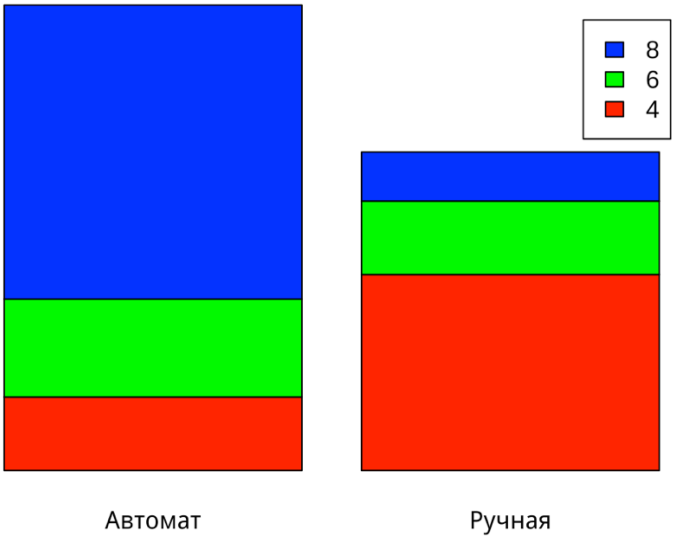

Рисунок 25 – Столбиковая диаграмма: координатные оси

4.7 Построение столбиковой диаграммы с помощью пакета *{ggplot2}*

Для построения столбиковой диаграммы будем использовать пакет *{ggplot2}* и встроенный набор данных *mtcars*.

Загрузим пакет *{tidyverse}*

```
library(tidyverse)
library(patchwork)
```

Прежде чем приступить к построению диаграммы необходимо определится с грамматикой построения графиков пакетом *{ggplot2}*.

Структура *{ggplot2}* – это многоуровневая грамматика.

Существует три типа слоёв, используемых в *{ggplot2}*:

- базовый слой – фон, система координат;
- слой *geom* – отдельные графики;
- слой для “украшения” – заголовки, легенды, надписи и т.д.

Для указания базового слоя мы используем функцию *ggplot()*. Использование данной функции отдельно не производит особого впечатления (*рисунок 26*).

```
ggplot()
```

Рисунок 26 – Пространство для визуализации

Подготовка данных

Немного преобразуем набора данных *mtcars*. Сохраним его под другим именем, преобразуем в *tibble*, а переменную *cyl* преобразуем в факторную. Для этого на основе переменной *cyl* с помощью функций *mutate()* и *factor()* создадим новую переменную

```
mtcars_job <- mtcars |>
  rownames_to_column(var = "car") |>
  tibble() |>
  mutate(cyl_factor = factor(cyl,
                             levels = c("4", "6", "8"),
                             labels = c("Четыре", "Шесть", "Восемь")
                             ))
```

Посмотрим данные с помощью функции *glimps()*.

```
glimpse(mtcars_job)
#> Rows: 32
#> Columns: 13
#> $ car        <chr> "Mazda RX4", "Mazda RX4 Wag", "Datsun 710", "Hornet 4 Dri
ve…
#> $ mpg        <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.
2,…
#> $ cyl        <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4,
4,…
#> $ disp       <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 1
40…
#> $ hp         <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180,
18…
#> $ drat       <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.9
2,…
```

```
#> $ wt         <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3
.1…
#> $ qsec       <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 2
2.…
#> $ vs         <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1,
1,…
#> $ am         <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
1,…
#> $ gear       <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4,
4,…
#> $ carb       <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2,
1,…
#> $ cyl_factor <fct> Шесть, Шесть, Четыре, Шесть, Восемь, Шесть, Восемь, Четыр
e,…
```

Задание. Опишите типы данных, которе содержит *mtcars_job*.

Построение диаграммы

Для построения столбиковой диаграммы можно использовать несколько функции: *geom_bar()* и *geom_col()*. В данной работе будем использовать функцию *geom_bar()*.

```
geom_bar(
  mapping = NULL,
  data = NULL,
  stat = "count",
  position = "stack",
  ...,
  just = 0.5,
  width = NULL,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)
```

где

– *mapping* – это аргумент, задающий эстетические атрибуты геометрических объектов (обычно при помощи функции *aes()*);

– *data* – имя таблицы с данными, которые необходимы для создания добавляемого слоя;

– *stat* – задаёт статистическое преобразование, которое применяется к изображаемым на слое данным;

– *position* – определяет способ взаимного расположения перекрывающихся геометрических объектов;

– *just* – регулировка размещения столбцов;

– *width* – ширина столбца;

– *na.rm* – логический аргумент, определяющий действие в отношение пропущенных данных (если *na.rm = FALSE*, по умолчанию, то пропущенный

значения будут удалены и пользователь увидит соответствующее предупреждение, `na.rm = TRUE`, то пропущенные значения также будут удалены, но без вывод предупреждения об этом);

– *orientation* – ориентация слоя;

– *show.legend* – логический аргумент, определяющий действие в отношение показа слоя легенды.

Эстетические атрибуты (аргументы функции `aes()`):

– *x* – переменная *X*;

– *alpha* – степень прозрачности цвета;

– *colour* – цвет линии, окаймляющей столбики;

– *fill* – цвет заливки столбцов;

– *linetype* – тип линии, окаймляющей столбики;

– *size* – толщина линии, окаймляющей столбики.

Построим столбиковую диаграмму (*рисунок 27*)

```
mtcars_job |>
  ggplot(aes(x = cyl_factor)) +
  geom_bar()
```

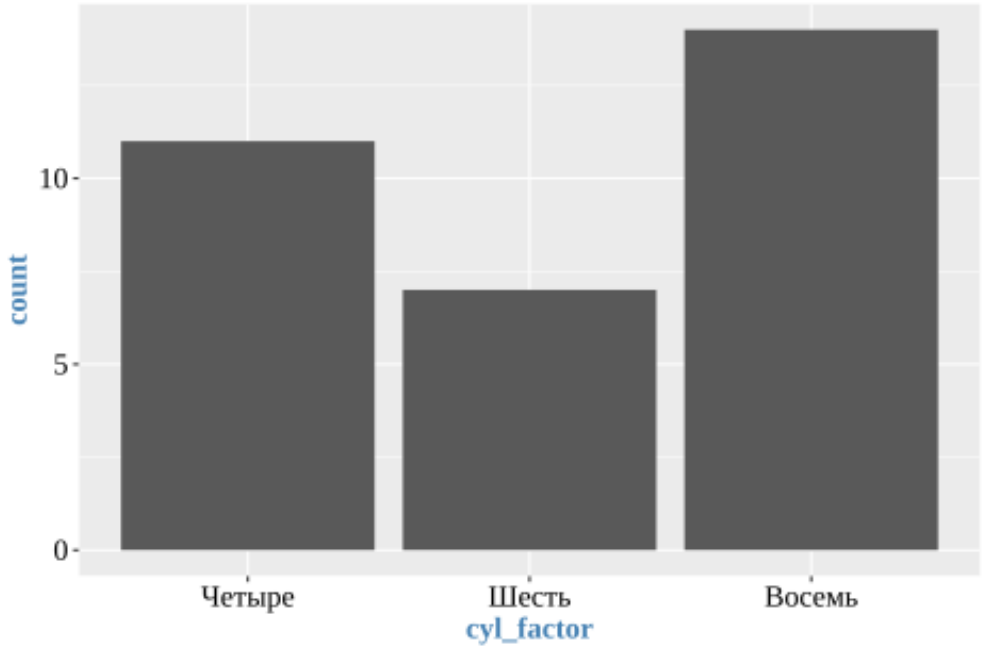

Рисунок 27 – Столбиковая диаграмма

Пакет `{ggplot2}` преобразует необработанную таблицу в новый набор категорий с соответствующими подсчётами. На основе полученной таблицы строится столбчатая диаграмма путем сопоставления `cyl_factor` с осью `X`, а `count` – с осью `Y`.

По умолчанию статистика `geom_bar()` использует для подсчёта функцию `stat_count()`, которая подсчитывает количество случаев.

Если подсчёты уже присутствуют в наборе данных, то их можно использовать для определения высоты столбиков. Для этого необходимо установить аргумент `stat = "identity"`. Рассмотрим пример о количестве проданных автомобилей по сегментам в 2023 году (*рисунок 28*).

```
store_car <- tribble(
  ~segment,                ~car_count,
  "Легковые",               1058.7,
  "Легкие \nкоммерческие",  90.4,
  "Грузовые",               143.7,
  "Автобусы",               16.3
  )
ggplot(data = store_car) +
  geom_bar(aes(x = segment, y = car_count),
           stat = "identity") +
  labs(x = "Сегмент",
       y = "Количество, шт.")
```

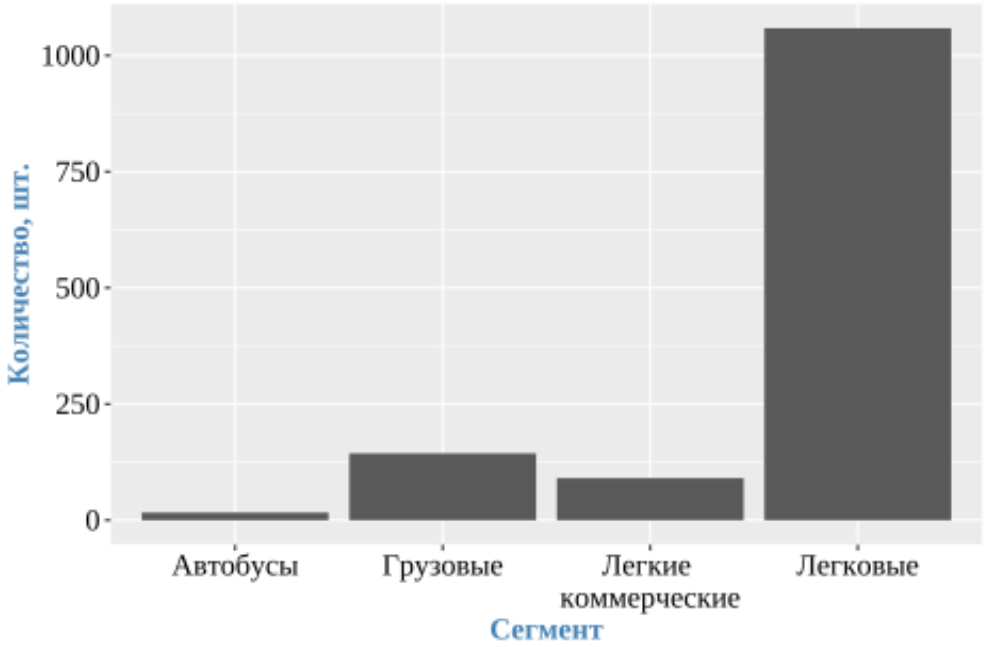

Рисунок 28 – Количество проданных автомобилей по сегментам в 2023 году в Российской Федерации

Часто бывает необходимость присвоить эстетическому атрибуту некоторое фиксированное значение, например цвет столбца. Это можно сделать, присвоив соответствующее значение параметру `fill` (*рисунок 29*).

```
mtcars_job |>
  ggplot(aes(x = cyl_factor)) +
  geom_bar(aes(fill = "green"))

mtcars_job |>
  ggplot(aes(x=cyl_factor)) +
  geom_bar(fill = "green")
```

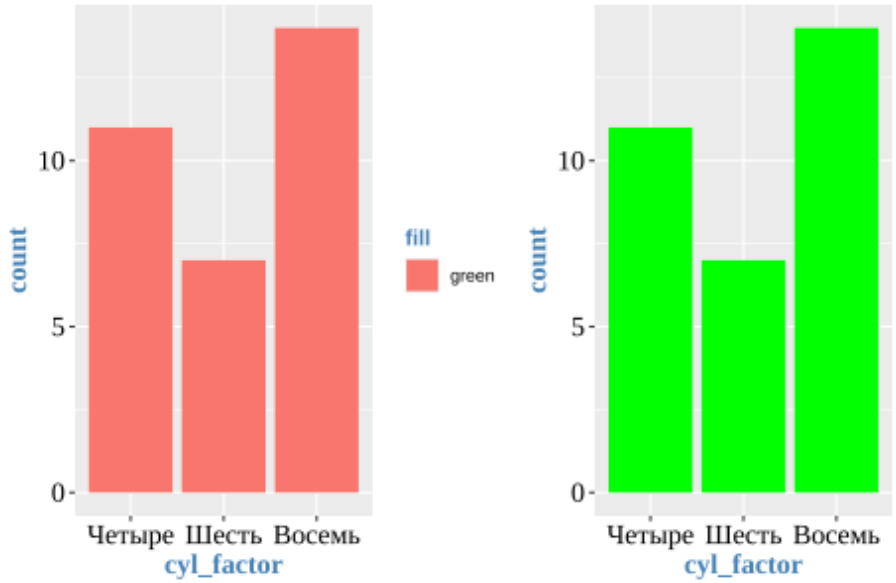

Рисунок 29 – Результаты построения одного и того же графика с использованием команды geom_bar(aes(fill="green")) и geom_bar(fill="green")

Заполнение столбцов отличается. При вызове функции в виде `geom_bar(aes(fill = "green"))` происходит автоматическое создание в таблице с данными новой качественной переменной с одним единственным значением – `"green"` и программа воспринимает это значение просто как уровень вновь созданной качественной переменной, а не как название цвета.

Переупорядочим столбцы в порядке убывания (*рисунок 30*).

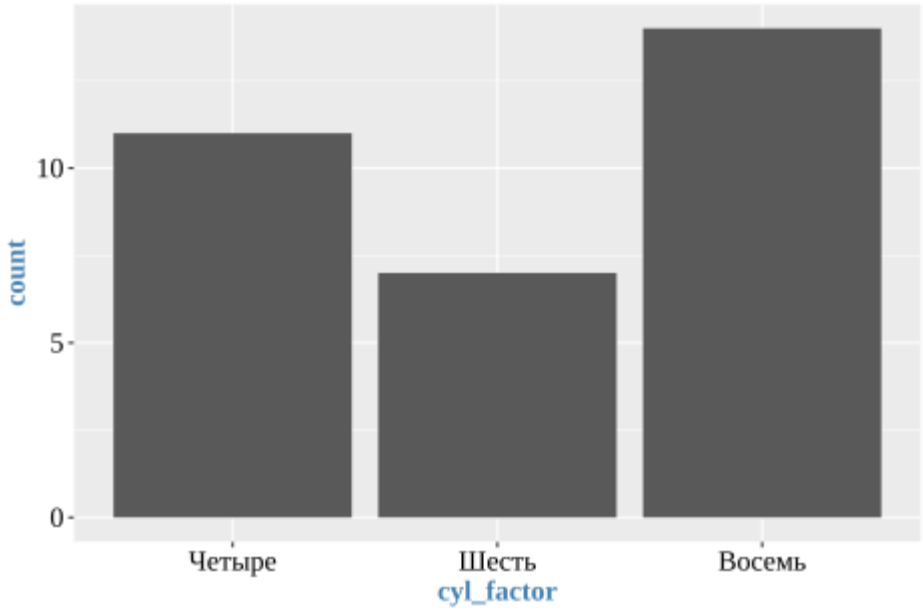

Рисунок 30 – Результаты построения одного и того же графика с использованием команды geom_bar(aes(fill="green")) и geom_bar(fill="green")

Построим горизонтальную столбиковую диаграмму. Существует два способа создания горизонтальной столбиковой диаграммы: с помощью `coord_flip()`

функции для поворота осей или передачи категориальной переменной в *y* аргумент `aes` (*рисунки 31 и 32*).

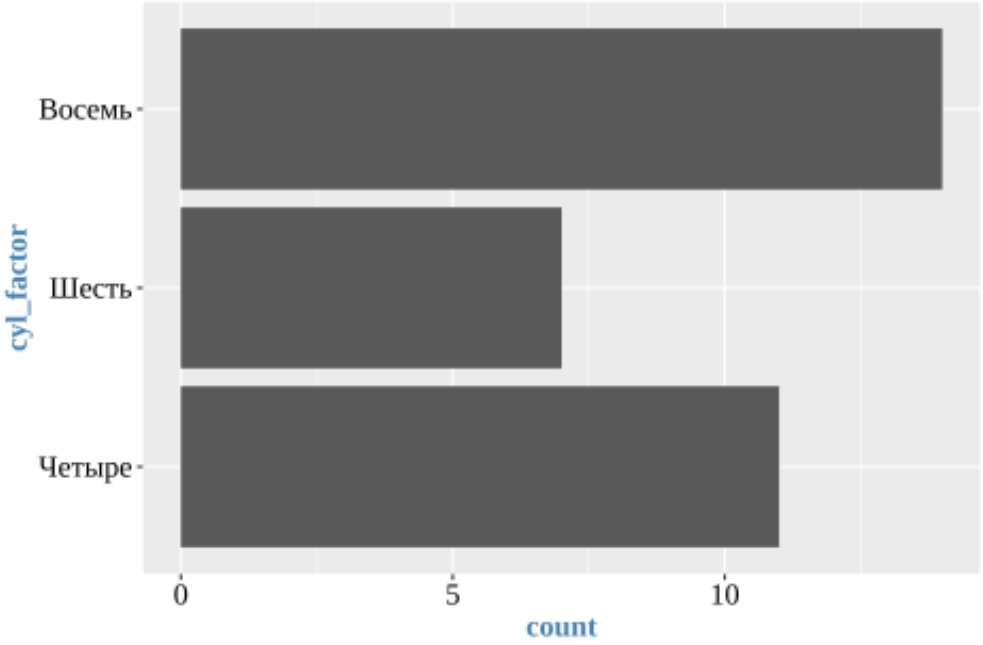

Рисунок 31 – Горизонтальная столбиковая диаграмма: `coord_flip())`

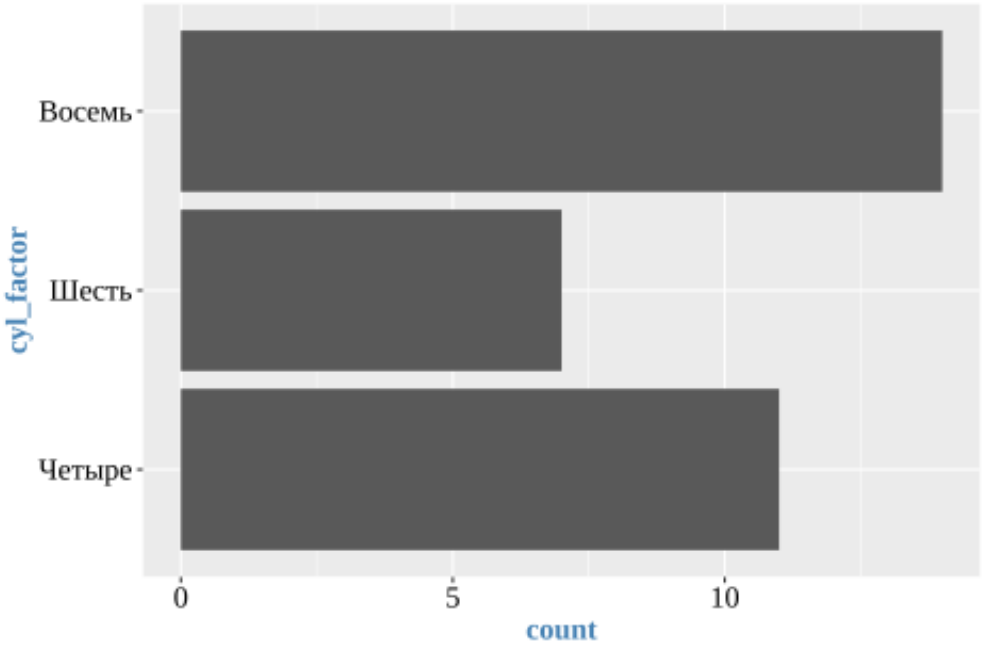

Рисунок 32 – Горизонтальная столбиковая диаграмма: смена осей

Обе столбиковые диаграммы являются идентичными.

Необходимо отметить, что у построенных диаграмм названия осей – это названия переменных, которые будут непонятны человеку, анализирующему графики. На языке *R*, как и любом другом, существует возможность переименовать их. Дать имена можно не одним, а нескольким способами. Одним из простых является применение функции `labs()` – (*рисунок 33*).

```
mtcars_job |>
  ggplot(aes(x = cyl_factor)) +
  geom_bar(stat = "count") +
  labs(x = "Число цилиндров",
       y = "Общее количество автомобилей")
```

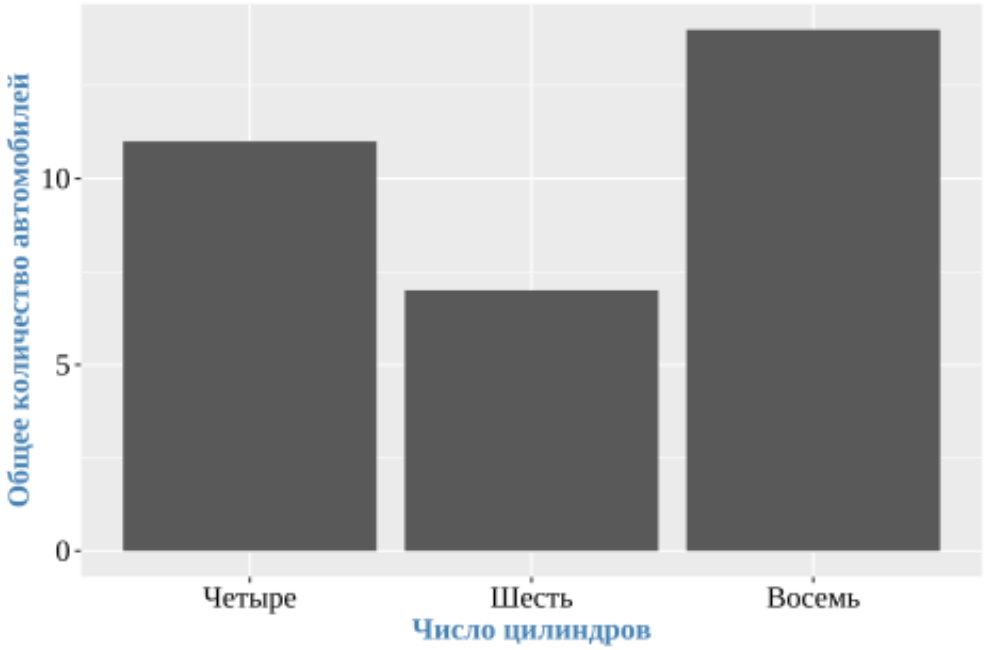

Рисунок 33 – Столбиковая диаграмма: названия осей

Используем дополнительные возможности языка *R*, дадим индивидуальный цвет каждому столбцу. Для этого возможно использовать несколько возможностей, однако необходимо их применять в зависимости от вида набора данных (*рисунок 34*).

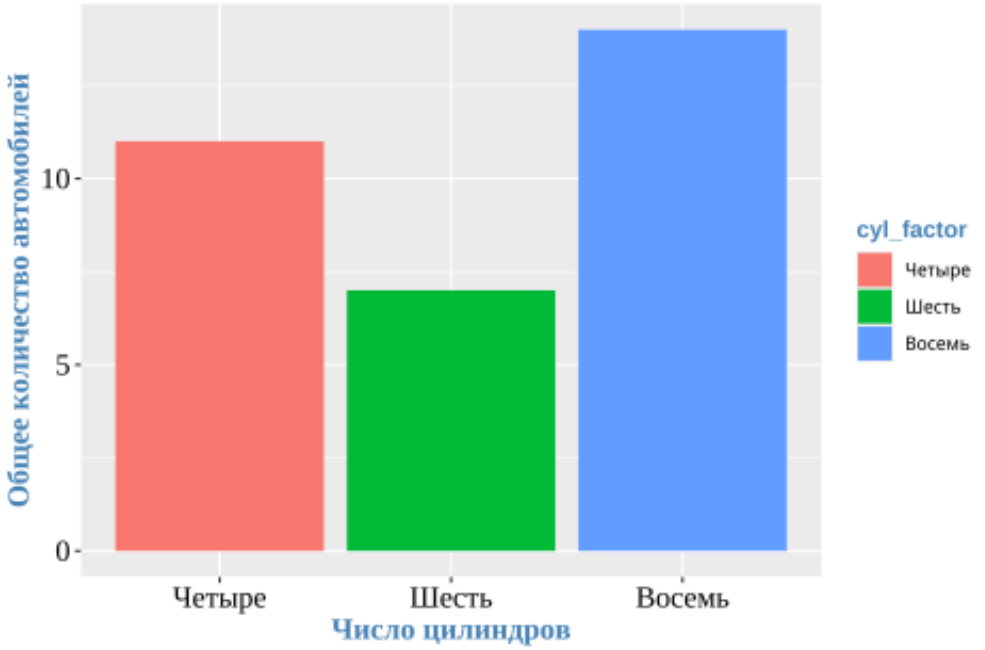

Рисунок 34 – Столбиковая диаграмма: названия осей

Сопоставим `cyl_factor` с другой переменной (*рисунок 35*).

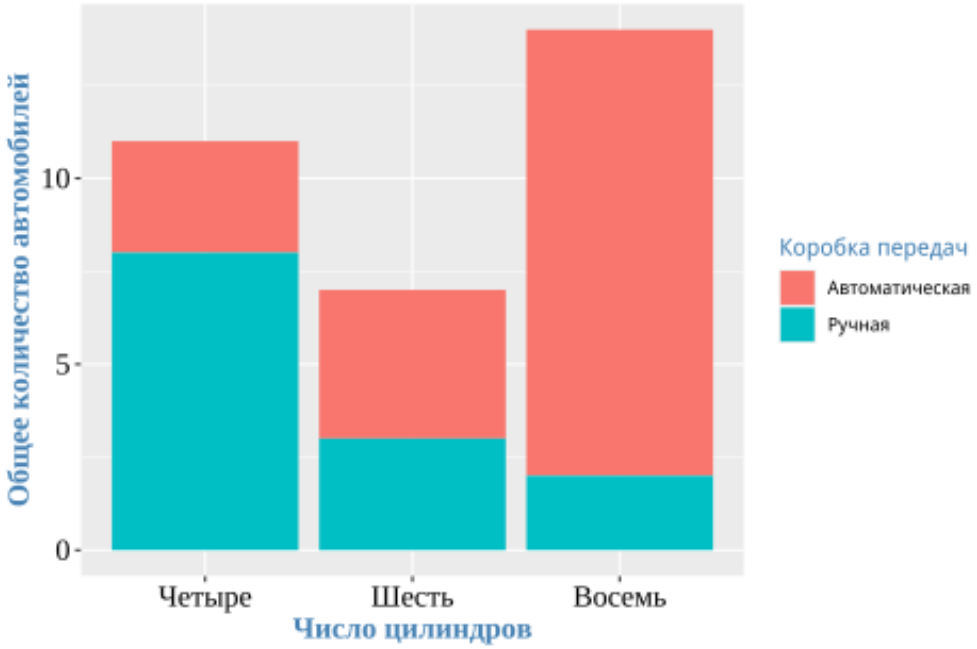

Рисунок 35 – Столбиковая диаграмма: сопоставление переменных

Эта визуализация создаёт «сложенную» (`stack`) столбчатую диаграмму. По умолчанию используется `position = "stack"` или `position = position_stack()`.

Посмотрим, как меняется столбчатая диаграмма с другими параметрами, так `position = fill`, столбики выстраиваются друг над другом, однако их длины пропорциональны доле общего числа наблюдений (*рисунок 36*).

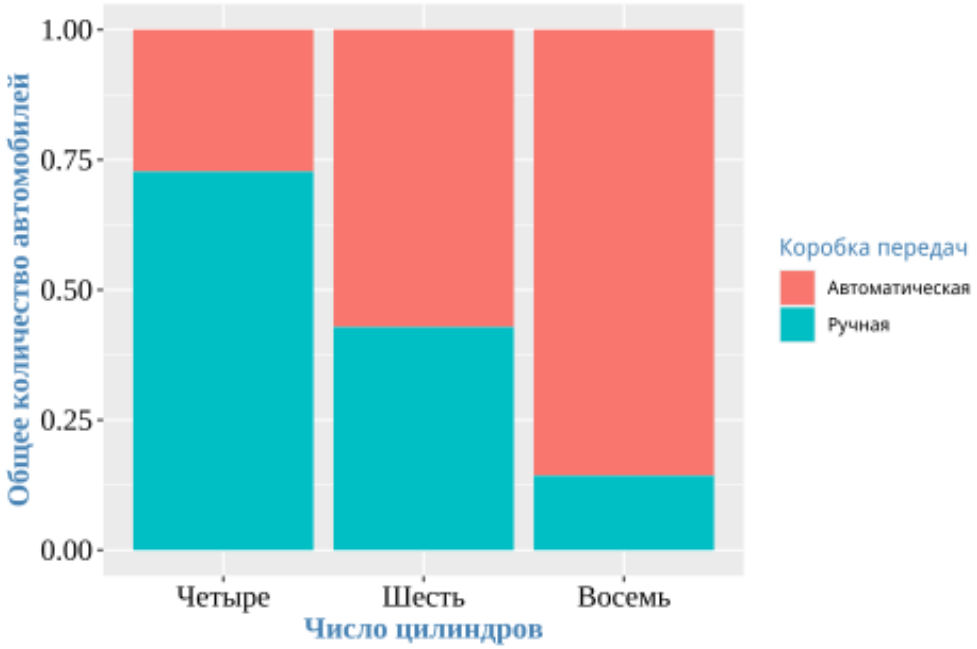

Рисунок 36 – Столбиковая диаграмма: `position = fill`

При использовании `position = "dodge"` столбики размещаются непосредственно друг рядом с другом. Это облегчает сравнение отдельных значений (*рисунок 37*).

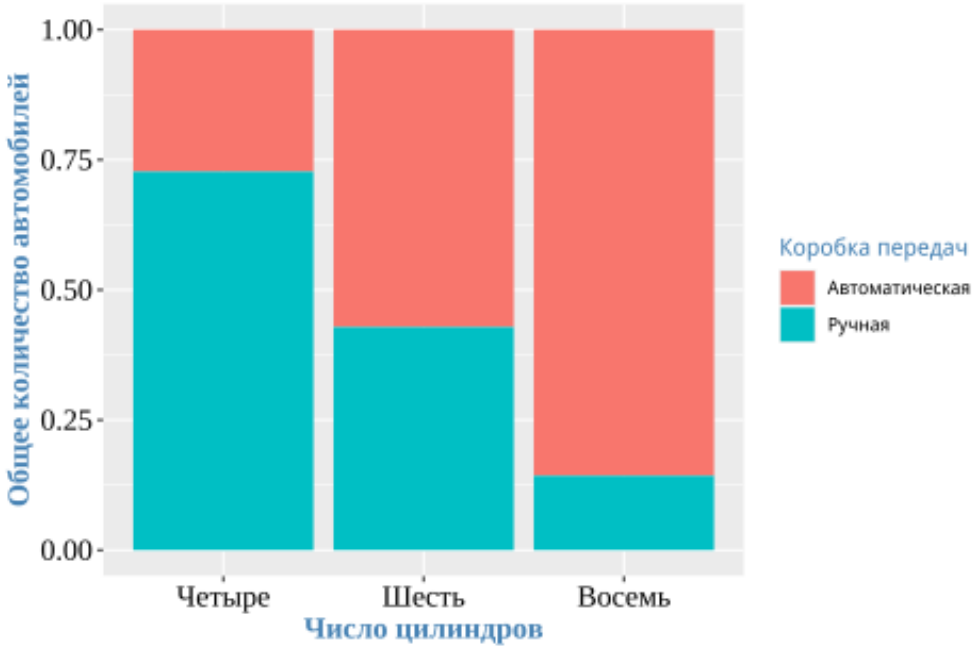

Рисунок 37 – Столбиковая диаграмма: `position = dodge`

При заполнении столбцов по группам цвета они будут соответствовать цветовым палитрам, которые по умолчанию содержаться в `{ggplot2}`. Чтобы переопределить эти цвета, можно указать их с помощью другой палитры или с помощью функции `scale_fill_manual()`, как с упорядоченным вектором цветов, так и с именованным вектором, где имена являются метками групп (*рисунок 38*).

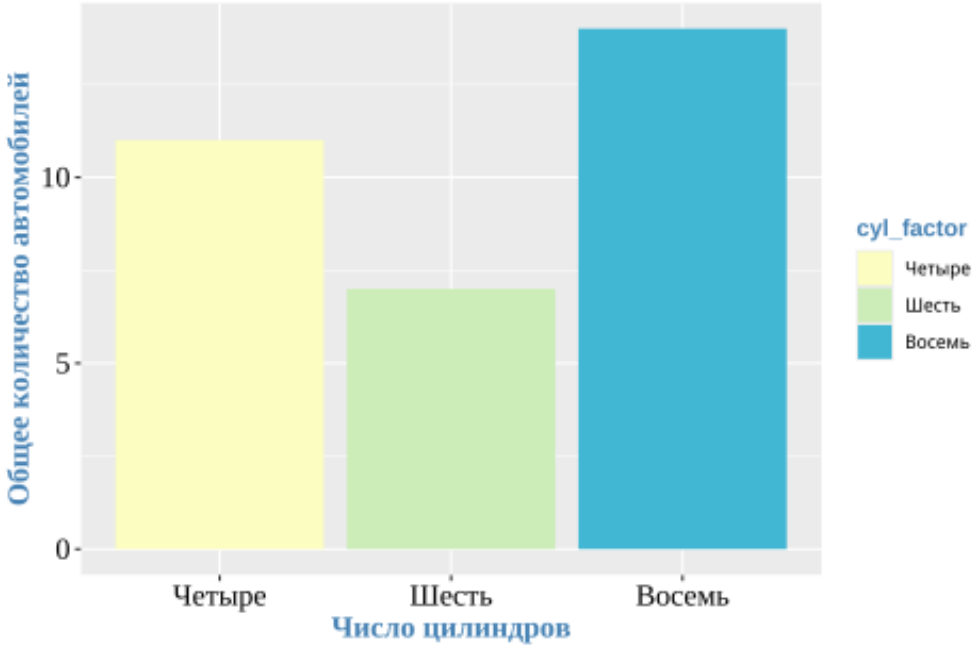

Рисунок 38 – Столбиковая диаграмма: `scale_fill_manual()`

Задание. Поэкспериментируйте с аргументом `colour` функции `aes()` и `ggplot()`.

Внешний вид графиков в `ggplot2` определяется системой стилей, или “тем”. В данном пакете содержится несколько стандартных стилей, которые задаются следующими функциями:

- `theme_gray()` – принятый по умолчанию “серый” стиль;
- `theme_dark()` – “темный” стиль;
- `theme_bw()` – “черно-белый” стиль;
- `theme_light()` – “светлый” стиль
- `theme_minimal()` – минималистический стиль;
- `theme_linedraw()` – стиль, похожий на `theme_light()`, однако с более выраженными линиями координатной сетки.

Пользователь имеет возможность создавать и свои собственные стили. Для этого служит функция `theme()`. У данной функции очень много аргументов, определяющих внешний вид разных элементов графика. В данной работе рассмотрим только несколько элементов. Более подробную информацию можно посмотреть в справочной информации пакета `ggplot2`, а также в книгах и в Интернете у таких авторов как Сергей Мастицкий[3], Роберт И. Кабаков[4] и создателя пакета – Хэдли Уикхема[5] (Hedley Wickham)

Оформим столбиковую диаграмму (*рисунок 39 и рисунок 40*).

```
mtcars_job |>
  ggplot(aes(x = cyl_factor, fill = cyl_factor)) +
  geom_bar(stat = "count") +
  labs(x = "Число цилиндров", y = "Общее количество автомобилей") +
  scale_fill_manual(values = c("#FCFEC1", "#CCEDB8", "#41B7D4")) +
  geom_text(aes(label = after_stat(count)), stat = "count", vjust = 2) +
  guides(fill = guide_legend(title = "Цилиндры")) +
  theme_bw() +
  theme(
    legend.position = "bottom",
    axis.title.x = element_text(size = 12, color = "blue"),
    axis.title.y = element_text(size = 12, colour = "blue"),
    axis.text = element_text(size = 12))
```

[3] Мастицкий С.Э. Визуализация данных с помощью ggplot2. – М.: ДМК Пресс, 2017. – 222 с.
[4] Роберт И. Кабаков R в действии / пер. с англ. А.Н. Киселева. 3-е изд. – М.: ДМК Пресс, 2023.
[5] https://ggplot2-book.org/

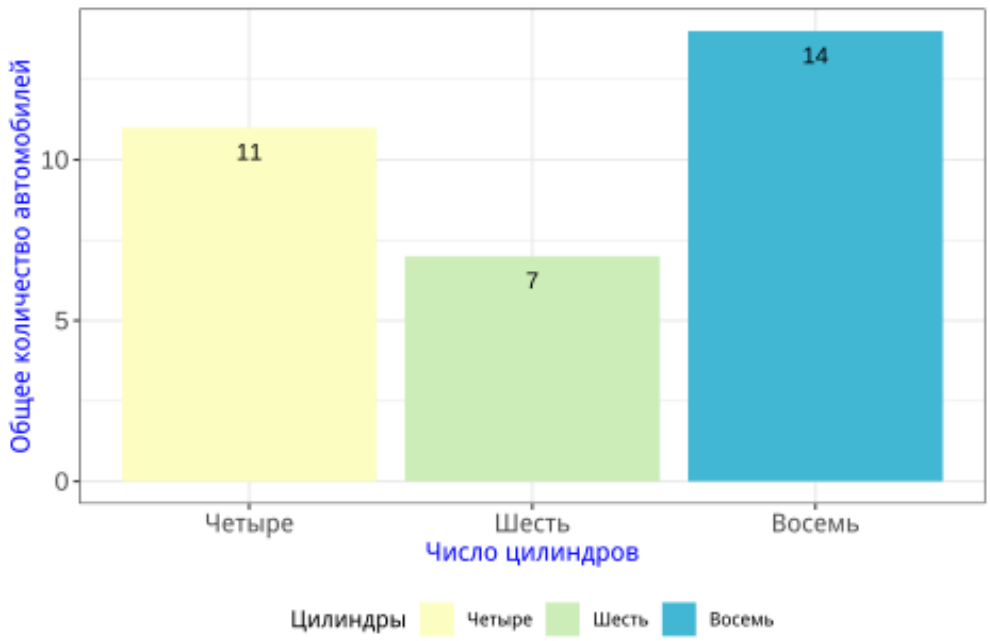

Рисунок 39 – Столбиковая диаграмма: настройка графика *theme()*

```
mtcars_job |>
  ggplot(aes(x = cyl_factor, fill = as_factor(am))) +
  geom_bar() +
  labs(x = "Число цилиндров", y = "Общее количество автомобилей") +
  scale_fill_manual(values = c("#FCFEC1", "#CCEDB8", "#41B7D4"), label = c("Авт
омат", "Ручная")) +
  geom_text(aes(label = after_stat(count)), stat = "count", position = position
_stack(vjust = 0.5)) +
  guides(fill = guide_legend(title = "Коробка передач", )) +
  theme_bw() +
  theme(
    legend.position = "bottom",
    axis.title.x = element_text(size = 12, color = "blue"),
    axis.title.y = element_text(size = 12, colour = "blue"),
    axis.text = element_text(size = 12))
```

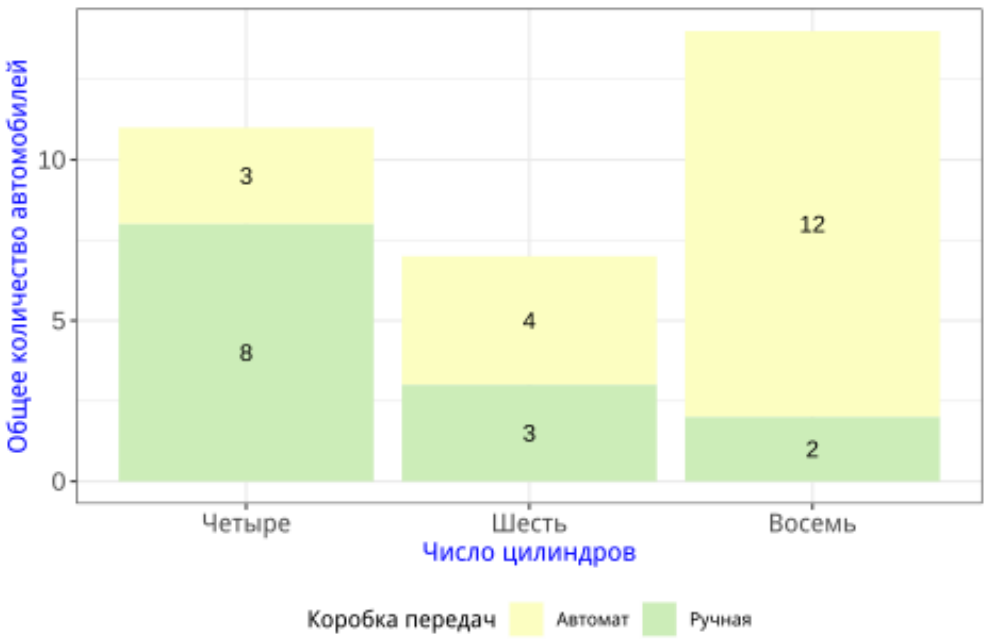

Рисунок 40 – Столбиковая диаграмма: настройка графика *theme()*

5. Гистограмма

5.1 Введение

Гистограммы графически отображают распределение значений непрерывных переменных, разделяя диапазон значений на заданное число отрезков по оси *x* и отображая частоту значений внутри каждого отрезка на оси *y*. При построении гистограммы значения анализируемой переменной упорядочиваются в соответствии с некоторым классовым промежутком. Получаемый в итоге график выглядит как совокупность из нескольких столбцов, ширина которых соответствует величине классового промежутка, а высота – частоте встречаемости соответствующего класса.

Гистограмма представляет собой столбчатый график, построенный по полученным опытным (эмпирическим) данным, которые разбиваются на ряд интервалов или дискретных значений показателя качества (например, середин интервалов), расположенных в порядке возрастания (ось абсцисс – ось X). Число данных (частота значений), попадающих в каждый интервал или соответствующих каждому дискретному значению, определяет высоту столбцов (ось ординат – ось Y) ~~[1]~~ [4].

Гистограмму можно рассматривать как аналог плотности распределения непрерывной случайной величины и по внешнему виду гистограммы подобрать соответствующий ей теоретический закон (модель) распределения. Чаще всего при анализе систем и процессов используют модель нормального распределения, график которой представляет собой холмообразную кривую, симметричную относительно среднего значения $\overline{X}$, ветви которой уходят в $\pm$ бесконечность.

5.2 Построение гистограммы с помощью базовых функций

Гистограмму можно создать при помощи команды

```
hist(x)
```

где *x* – это числовой вектор. Опция `freq=FALSE` позволяет построить гистограмму на основе плотности вероятности, а не частот значений. Параметры `breaks` определяет число отрезков. По умолчанию все отрезки имеют одинаковую длину.

В качестве набора данных будет рассмотрен уже знакомый `mtcars`.

Построим простую гистограмму (*рисунок 41*).

```
hist(mtcars$mpg)
```

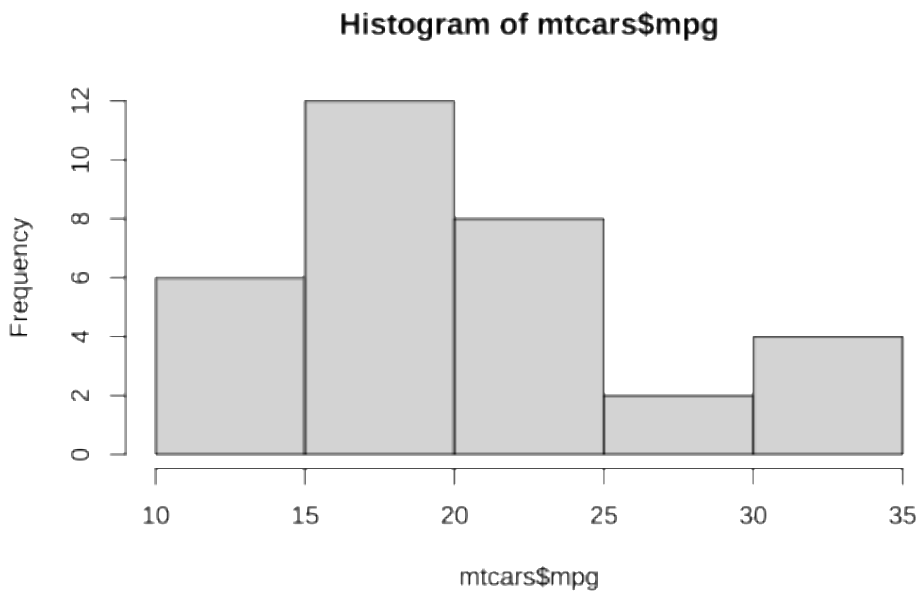

Рисунок 41 – Простая гистограмма

Построим гистограмму с заданным числом интервалов (*рисунок 42*)

```
hist(mtcars$mpg,
     col = "blue", breaks = 5,
     xlab = "Расход топлива",
     main = "Цветная гистограмма")
```

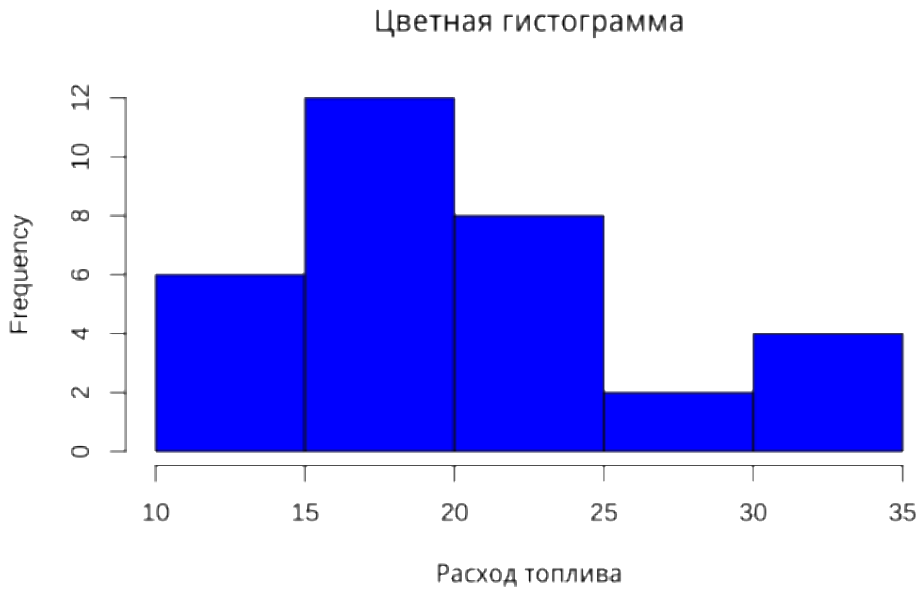

Рисунок 42 – Цветная гистограмма

Построим гистограмму с кривой распределения плотности. Функция `lines()` позволяют добавить кривую уже к созданной и изменить её размеры и цвет(*рисунок 43*).

```r
hist(mtcars$mpg,
     col = "green",
     breaks = 8,
     freq = FALSE,
     xlab = "Расход топлива",
     main = "Гистограмма с кривой нормального распределения"
     )
lines(density(mtcars$mpg), col = "red", lwd = 2)
```

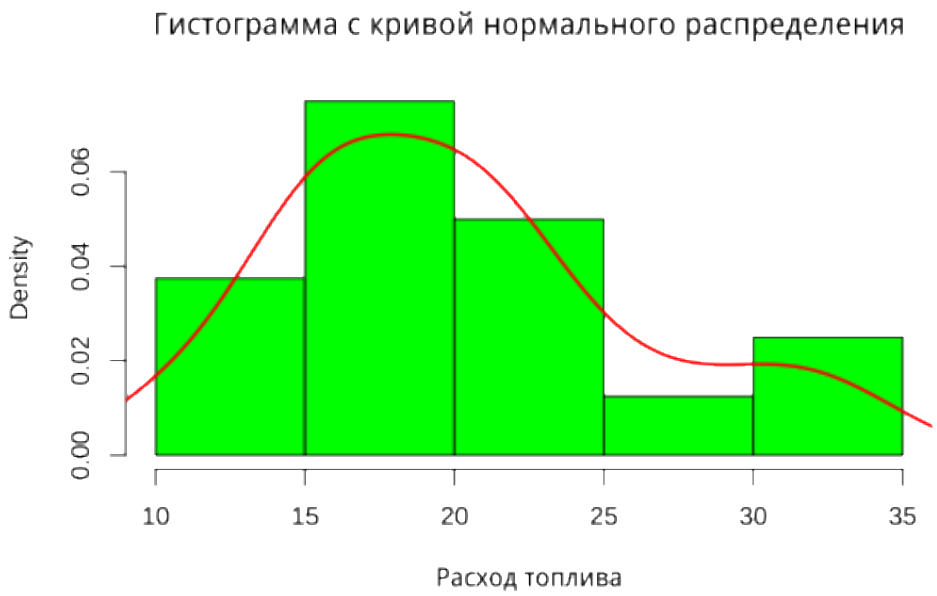

Рисунок 43 – Цветная гистограмма с кривой нормального распределения

5.3 Построение гистограммы с помощью пакета *{ggplot2}*

В пакете *{ggplot2}* содержится специальная функция *geom_histogram()*, которая служит для построения гистограммы

```r
geom_histogram(
  mapping = NULL,
  data = NULL,
  stat = "bin",
  position = "stack",
  ...,
  binwidth = NULL,
  bins = NULL,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)
```

где

– *mapping* – это аргумент, задающий эстетические атрибуты геометрических объектов (обычно при помощи функции *aes()*);

– *data* – имя таблицы с данными, которые необходимы для создания добавляемого слоя;
– *stat* – задаёт статистическое преобразование, которое применяется к изображаемым на слое данным;
– *position* – определяет способ взаимного расположения перекрывающихся геометрических объектов, по умолчанию `position = "stack"`;
– *binwidth* – размер классового промежутка:
– *bins* – число столбцов;
– *na.rm* – логический аргумент, определяющий действие в отношение пропущенных данных (если `na.rm = FALSE`, по умолчанию, то пропущенный значения будут удалены и пользователь увидит соответствующее предупреждение, `na.rm = TRUE`, то пропущенные значения также будут удалены, но без вывода предупреждения об этом);
– *orientation* – ориентация слоя;
– *show.legend* – логический аргумент, определяющий действие в отношение показа слоя легенды;
– *inherit.aes* – переопределяет эстетические параметры, по умолчанию `inherit.aes = TRUE`.

Эстетические атрибуты (аргументы функции `aes()`):
– *x* – переменная *X*;
– *alpha* – степень прозрачности цвета;
– *colour* – цвет линии, окаймляющей столбики;
– *fill* – цвет заливки столбцов;
– *linetype* – тип линии, окаймляющей столбики;
– *size* – толщина линии, окаймляющей столбики.

Загрузим необходимые пакет

```
library(tidyverse)
```

Простая гистограмма (*рисунок 44*).

```
mtcars |>
  ggplot(aes(x = mpg)) +
  geom_histogram()
```

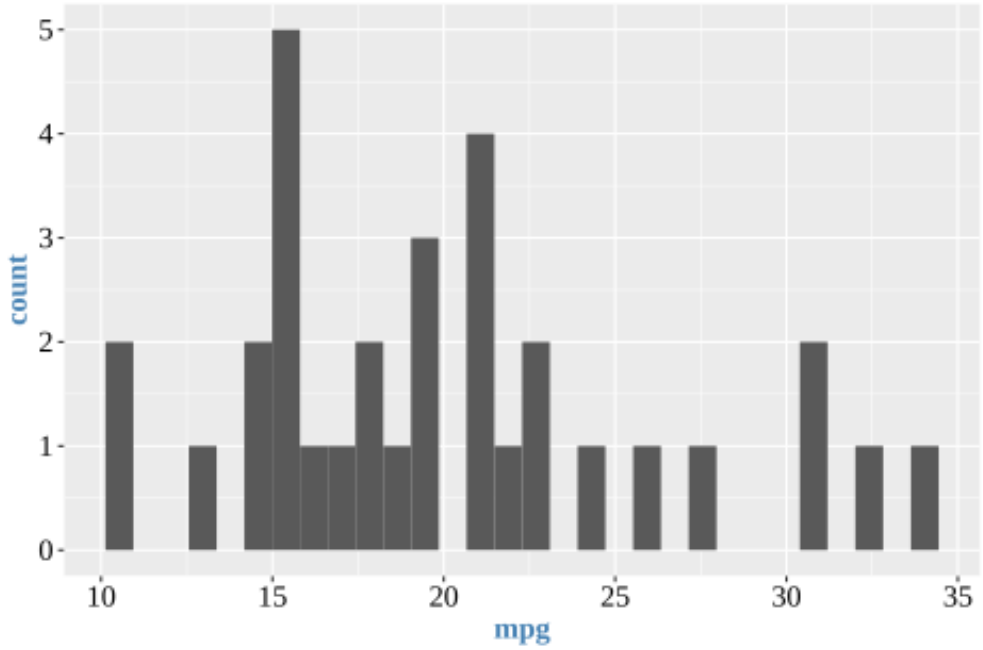

Рисунок 44 – Простая гистограмма: *geom_histogram()*

Цветная гистограмма (*рисунок 45*).

```
mtcars |>
  ggplot(aes(x = mpg)) +
  geom_histogram(bins = 15, fill = "red",
                 colour = "black", size = 1.3,
                 linetype = 3)
```

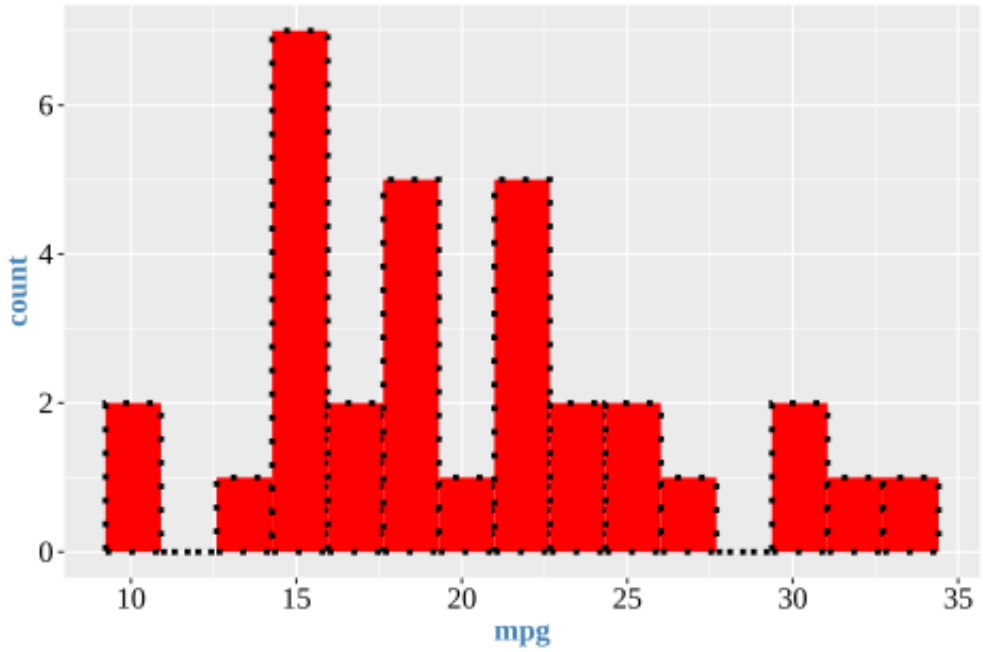

Рисунок 45 – Цветная гистограмма: *geom_histogram()*

Управление взаимным расположением столбиков на гистограмме.

Столбики располагаются друг над другом (*рисунок 46*).

```
mtcars_job <- tibble(mtcars)
```

```
mtcars_job |>
  mutate(cyl_factor = factor(cyl,
                             levels = c("4", "6", "8"),
                             labels = c("Четыре", "Шесть", "Восемь"))
         ) |>
  ggplot(aes(x = mpg)) +
  geom_histogram(aes(fill = cyl_factor)) +
  guides(fill = guide_legend(title = "Цилиндры"))
```

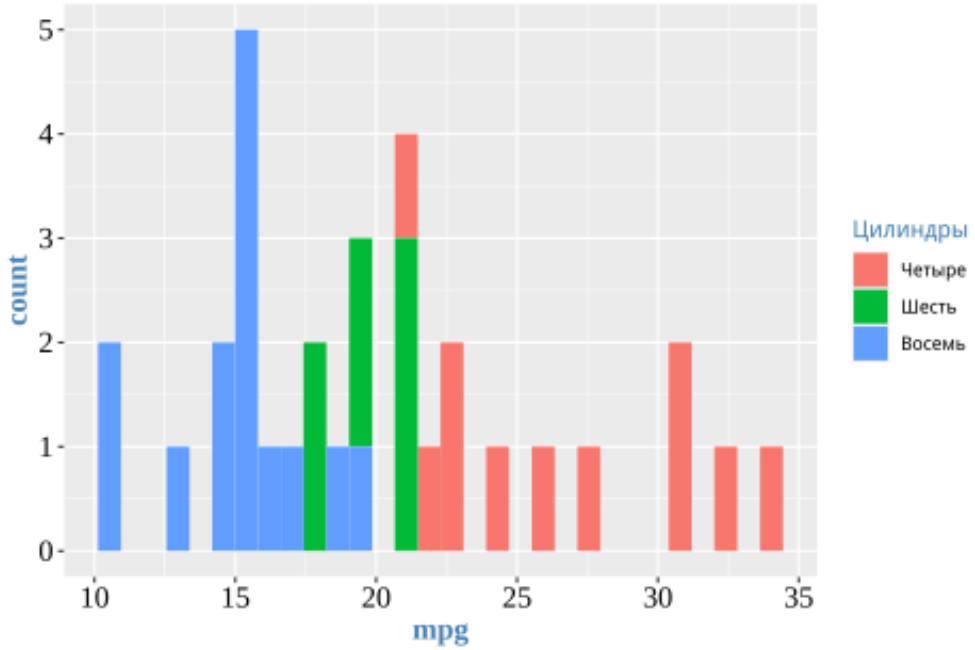

Рисунок 46 – Расположение столбиков гистограммы: *position="state"*

Столбики располагаются рядом (*рисунок 47*).

```
mtcars_job <- mtcars |>
  rownames_to_column(var = "car_names") |>
  as_tibble()

mtcars_job |>
  mutate(cyl_factor = factor(cyl,
                             levels = c("4", "6", "8"),
                             labels = c("Четыре", "Шесть", "Восемь"))
         ) |>
  ggplot(aes(x = mpg)) +
  geom_histogram(aes(fill = cyl_factor), position = "dodge") +
  guides(fill = guide_legend(title = "Цилиндры"))
```

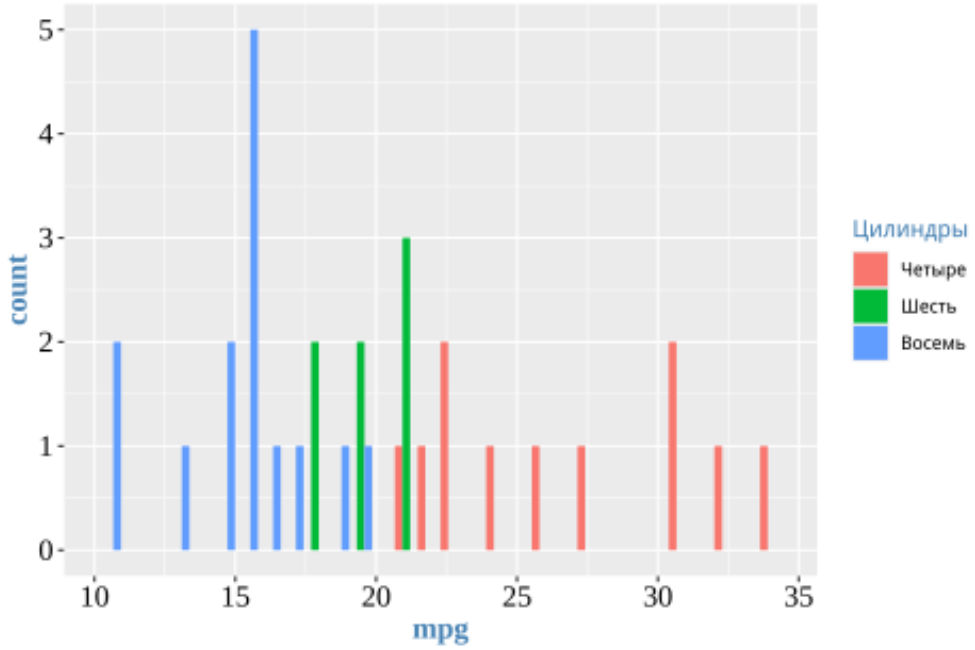

Рисунок 47 – Расположение столбиков гистограммы: *position="dodge"*

5.4 Ширина гистограммы

Гистограммы в некотором роде являются расширенной версией столбчатой диаграммы, где количество ячеек регулируется. Высоты представляют количество (или плотность), а ширина одинакова, либо варьируется для каждого столбца.

Для лучшей визуальной оценки мы можем использовать пользовательский набор интервалов в качестве ширины. Спецификация набора ширины гистограммы осуществляется посредством указания `breaks = BINS` в качестве аргумента в вызове `geom_histogram()`, где `BINS` – последовательность точек разрыва.

В качестве набора данных будем использовать `mpg`.

Построим гистограмму с шириной столбцов по умолчанию (*рисунок 48*).

```
ggplot(mpg, aes(x = hwy)) +
  geom_histogram(fill = "darkcyan",
                 colour = "gray")
```

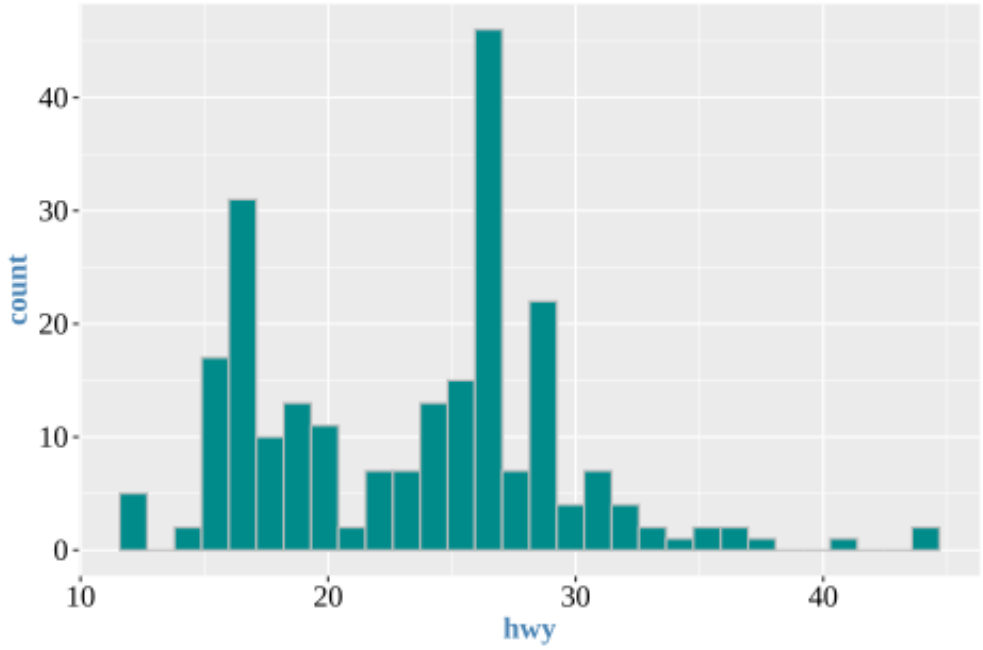

Рисунок 48 – Гистограмма: ширина столбцов по умолчанию

Построим гистограмму с шириной столбцов равных 1 (*рисунок 49*).

```
bins = seq(10, 50, 1)
ggplot(mpg, aes(x = hwy)) +
  geom_histogram(fill = "darkcyan",
                 colour = "gray",
                 breaks = bins)
```

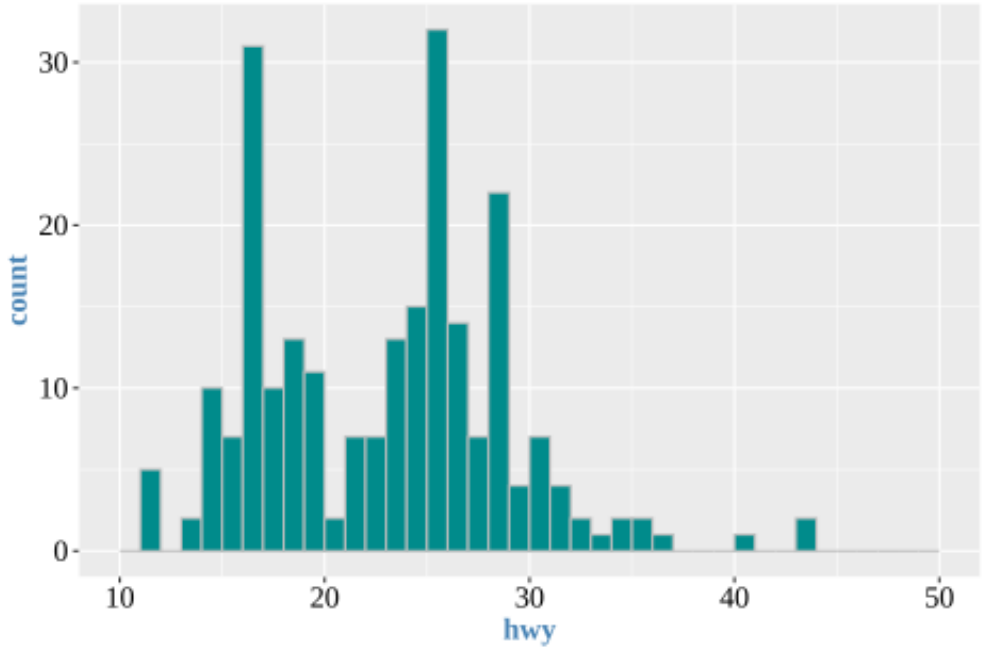

Рисунок 49 – Гистограмма: ширина столбцов равна 1

На основании полученной диаграммы (*рисунок 49*) можно отметить, что самый высокий столбец располагается в диапазоне (25;26].

Попробуем другую ширину шага, например, 5 (*рисунок 50*).

```
bins = seq(10, 50, 1)
ggplot(mpg, aes(x = hwy)) +
  geom_histogram(fill = "darkcyan",
                 colour = "gray",
                 breaks = bins)
```

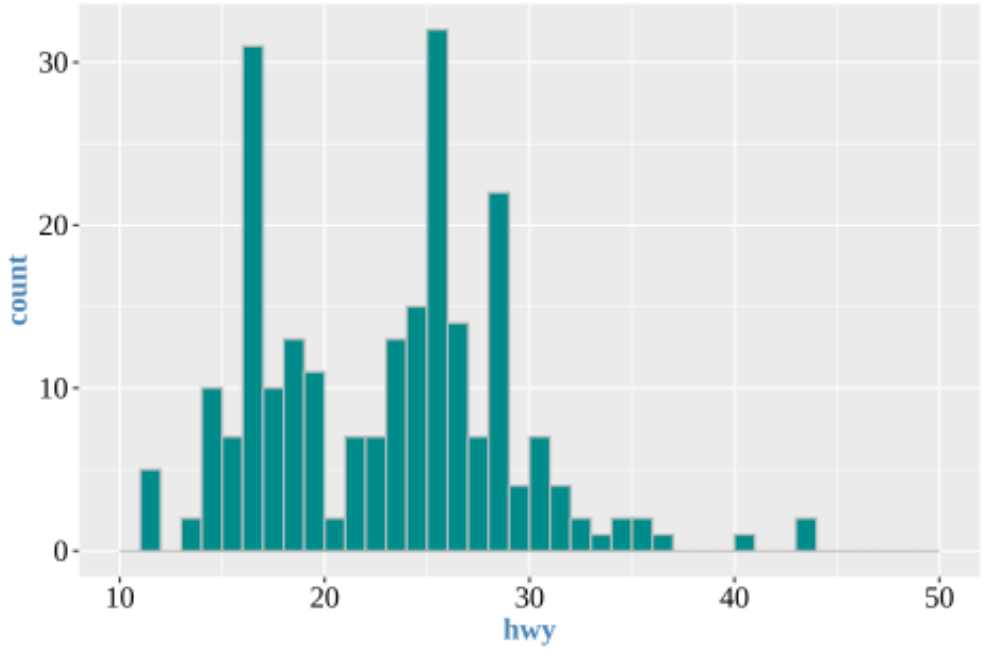

Рисунок 50 – Гистограмма: ширина столбцов равна 5

Можем отметить, что гистограмма показывает два высоких столбца в диапазонах (15,20] и (25,30] (*рисунок 50*).

5.5 Шкала плотности

Следует отметить некоторые расхождения между шкалой плотности и шкалой подсчёта. Хотя шкала подсчёта легко воспринимается визуально, чем шкала плотности, однако она может вводит в заблуждение.

Сопоставим две гистограммы (*рисунок 51*). Подключим пакет, который позволяет размещать несколько диаграммы рядом.

```
library(patchwork)

uneven_bins <- c(10, 15, 30, 45)

g1 <- ggplot(mpg, aes(x = hwy)) +
  geom_histogram(fill = "darkviolet", color = "grey",
                 breaks = uneven_bins, position = "identity")


g2 <- ggplot(mpg, aes(x = hwy)) +
  geom_histogram(aes(y = after_stat(density)),
                 fill = "darkviolet", color = "grey",
                 breaks = uneven_bins, position = "identity")
```

```
g1 + g2
```

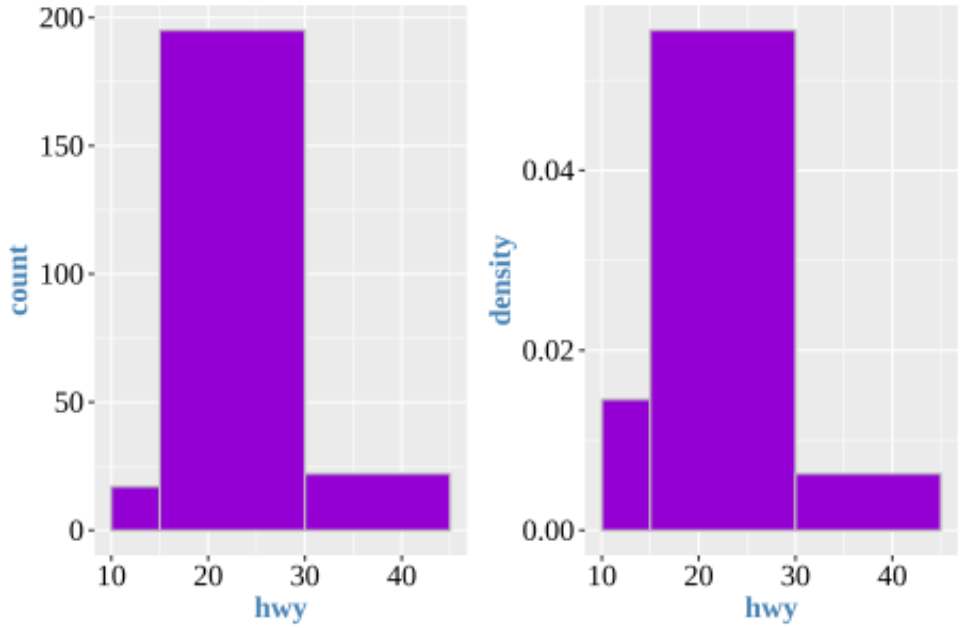

Рисунок 51 – Гистограмма. Шкалы плотности и подсчёта

Обратите внимание (*рисунок 51*), как гистограмма в шкале подсчета увеличивает высоту столбца (30,45]. Показанная высота – это просто количество моделей автомобилей в этом контейнере, без учета ширины столбца.

В то время как обе ширины столбца [10,15] и (30,45] могут иметь одинаковое количество моделей автомобилей в корзине, плотность столбца [10,15] больше, потому что большее количество элементов содержится в меньшей ширине ячейки.

Другими словами, (30,45] бар может обеспечить большее покрытие, поскольку он настолько «растянут», т.е. ширина его ячейки намного больше, чем у [10,15] бар.

Рассмотрим преимущество при использовании шкалы плотности.

В качестве примера рассмотрим два подмножества автомобилей из набора данных *mpg*.

```
first_subset <- mpg |>
  filter(class %in% c("minivan", "midsize", "suv"))

second_subset <- mpg |>
  filter(class %in% c("compact", "subcompact", "2seater"))
```

Создадим гистограмму для каждого созданного набора данных и сопоставим друг с другом (*рисунок 52*).

```
 bin_choise <- c(10, 15, 20, 22, 27, 30, 45)

g1 <- ggplot(first_subset, aes(x = hwy)) +
```

```
  geom_histogram(aes(y = after_stat(density)),
                 fill = "darkviolet",
                 colour = "grey",
                 breaks = bin_choise) +
  labs(title = "Средней вместимости")

g2 <- ggplot(second_subset, aes(x = hwy)) +
  geom_histogram(aes(y = after_stat(density)),
                 fill = "darkviolet",
                 colour = "grey",
                 breaks = bin_choise) +
  labs(title = "Малой вместимости")

g1 + g2
```

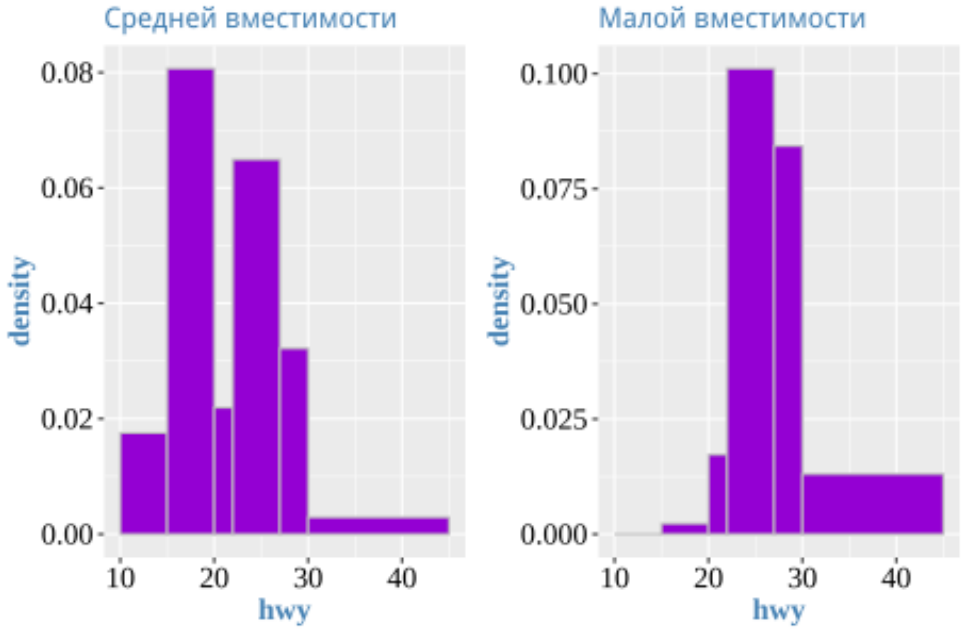

Рисунок 52 – Гистограммы с параметром *density*

На основании полученных диаграмм можно отметить, что в диапазоне (30,45], плотность наблюдений для автомобилей малой вместимости больше. Также отметим, что в первом подмножестве большее количество автомобилей сконцентрировано в диапазоне [10,30].

Наложим сглаженную кривую поверх каждой из гистограмм (*рисунок 53*). Площадь под кривой показывает, где находится основная часть данных.

```
bin_choise <- c(10, 15, 20, 22, 27, 30, 45)

g1 <- ggplot(first_subset, aes(x = hwy)) +
  geom_histogram(aes(y = after_stat(density)),
                 fill = "white",
                 colour = "grey",
                 breaks = bin_choise) +
  labs(title = "Средней вместимости") +
  geom_density(adjust = 3, fill = "darkcyan", alpha = 0.2)
```

```r
g2 <- ggplot(second_subset, aes(x = hwy)) +
  geom_histogram(aes(y = after_stat(density)),
                 fill = "white",
                 colour = "grey",
                 breaks = bin_choise) +
  labs(title = "Малой вместимости") +
  geom_density(adjust = 2, fill = "darkcyan", alpha = 0.2) +
  theme(
    axis.text.x = element_text(family = "serif")
  )

g1 + g2
```

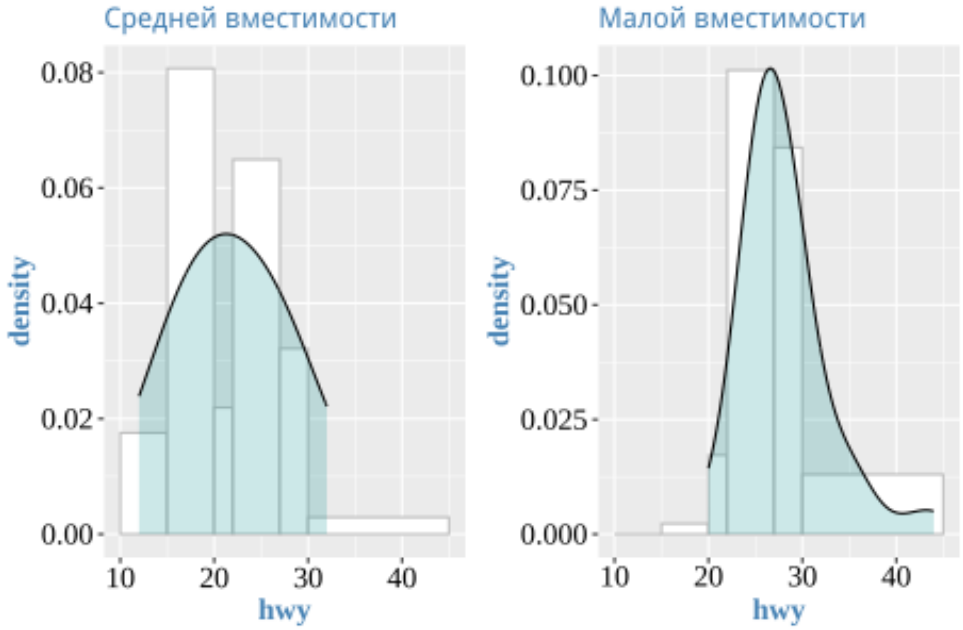

Рисунок 53 – Гистограмма с графиком плотности

Задание

Использовать встроенный набор данных *diamonds*. При построении использовать два направления: *базовые возможности* и возможности пакета *{ggplot2}*.

Задание 1. Оформите гистограмму, которая будет понятна исследователю.

Задание 2. Создайте диаграмму используя встроенный набор данных *diamonds*.

Задание 3. Построить столбиковую диаграмму.

Задание 4. Оформить столбиковую диаграмму, которая будет понятна исследователю.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Брюс П. Практическая статистика для специалистов Data Science: Пер. С англ. / П. Брюс, Э. Брюс, П. Гедек. – 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2021.
2. Кабаков И.Р. R в действии / пер. с англ. А.Н. Киселева. 3-е изд. – М.: ДМК, 2023.
3. Мастицкий С.Э. Визуализация данных с помощью ggplot2. – М.: ДМК, 2017.
4. Солонин С.И. *Метод гистограмм*. Екатеринбург: УрФУ, 2014.
5. Bache S.M., Wickham H. *magrittr: A Forward-Pipe Operator for R*. 2022.
6. ggplot2: Elegant Graphics for Data Analysis (3e). *https://ggplot2-book.org/*.
7. Müller K., Wickham H. *tibble: Simple Data Frames*. 2023.
8. R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing, 2023.
9. Wickham H. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag – New York, 2016.
10. Wickham H. и др. *dplyr: A Grammar of Data Manipulation*. 2023.

Учебное издание

ВИЗУАЛИЗАЦИЯ ДАННЫХ НА ЯЗЫКЕ R

Учебно-методическое пособие

Составители:
Максимов Даниил Геннадьевич
Котлячкова Наталья Владимировна

Авторская редакция

Издательский центр «Удмуртский университет»
426034, г. Ижевск, ул. Ломоносова, 4Б, каб. 021
Тел. : +7(3412)916-364 E-mail: editorial@udsu.ru