Основы статистики. Часть 2

stepic.org

18 апреля 2016 г.

В том документе представлена небольшая памятка, которая может помочь вам при решении заданий.

Работа с dataframe

Dataframe - основной тип данных, с которым мы будем работать в R. Обычно результаты нашего исследования сохраняются в dataframe следующим образом: каждая строка это - наблюдение, каждый столбец - это переменная. Давайте посмотрим на встроенные в R данные под названием iris.

```
# выполните эту команду чтобы данные отобразились в глобальном окружении
data(iris)

# начнем с общей информации о данных
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
# посмотрим на первые пять строк наших данных
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Функция str() вернет вам описание данных: число наблюдение, список переменных и их тип. Мы видим, что в данных 150 наблюдений (цветков Ириса), у каждого из них измерена длина и ширина лепестка и чашелистика, а также указано к какому виду относится каждый цветок. Команда head() позволяет взглянуть на сами данные.

Индексация dataframe

```
# каждое наблюдение имеет номер строки и столбца, в котором оно находится
iris[1, 4] # такая команда напечатает наблюдение в первой строке в четвертом столбце

# к переменной в данных можно обратиться по ее имени при помощи знака $
iris$Sepal.Length

# или при помощи номера столбца
iris[, 1] # тоже самое, что и iris$Sepal.Length, обратите внимание что мы не указал никакого
 индекса на месте строк, это означает, что мы отбираем все строки, но только первый столбец

# чтобы обратиться сразу к нескольким переменным можно использовать вектор имен или индексов
iris[, c(1, 3)]
iris[, c("Sepal.Length", "Petal.Length")] # тоже самое

# чтобы отобрать все переменные кроме некоторой, используйте отрицательную индексацию
iris[, -1] # все кроме первой
iris[, -c(1, 3)] # все кроме первой и третьей
```

Работа с факторами

В заданиях этой недели вы не раз столкнетесь с факторами. Это специальный тип данных в R для хранения номинативных переменных.

```
# В наших данных переменная Species - это фактора с тремя градациями (levels).
str(iris$Species)
```

```
##  Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
# Давайте посмотрим на частоту встречаемости каждой из градации при помощи функции table()
table(iris$Species)
```

```
##
##     setosa versicolor  virginica
##         50         50         50
```

Часто возникает путаница между levels и labels фактора в R. Давайте рассмотрим пример.

```
# Создадим вектор из целых чисел, где каждое число - это один из авторов этого курса. Предпол
ожим, это данные о том, сколько уроков создал каждый из авторов. Видим, что каждый из трех ав
торов сделал по три урока.
v <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
str(v)
```

```
##  num [1:9] 1 1 1 2 2 2 3 3 3
```

```
# Давайте сделаем из вектора фактор при помощи функции factor()
f1 <- factor(v)
str(f1)
```

```
##  Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 3 3 3
```

```
# Теперь у нашего фактора три уровня Levels: 1 2 3. R считает, что переменная f1 - это факто
р, который может принимать только три градации 1, 2 или 3. По умолчанию R создает фактор с чи
слом уровней равным числу уникальных элементов в векторе.

# Для удобства работы с фактором мы можем дать имена нашим градациям, например, Ivan, Tolik,
 Polina при помощи аргумента labels.
f1 <- factor(v, labels = c("Ivan", "Tolik", "Polina"))
str(f1)
```

```
##  Factor w/ 3 levels "Ivan","Tolik",..: 1 1 1 2 2 2 3 3 3
```

```
table(f1)
```

```
## f1
##   Ivan  Tolik Polina
##      3      3      3
```

```
# Но на практике может возникнуть следующая ситуация, в нашей выборке будут представлены не в
се возможные градации фактора. Тогда мы можем указать, какие градации возможны при помощи арг
умента levels.
f1 <- factor(v, levels = c(1, 2, 3, 4))
str(f1)
```

```
##  Factor w/ 4 levels "1","2","3","4": 1 1 1 2 2 2 3 3 3
```

```
# таким образом наш фактор теоретически может принимать 4 градации, но в выборке представлены
только три. Мы также теперь можем назвать каждую градацию.
f1 <- factor(v, levels = c(1, 2, 3, 4), labels = c("Ivan", "Tolik", "Polina", "Misha"))
table(f1)
```

```
## f1
##   Ivan  Tolik Polina  Misha
##      3      3      3      0
```

```
# теперь R понимает, что Misha есть среди авторов курса, но просто не встречается в этой выбо
рке.
```

Мораль: когда создаете фактор в R иногда важно сразу указать, какие градации он может принимать.

Функции для анализа номинативных данных

```
# Рассмотрим функцию для применения теста Хи - квадрат
# вы можете вызвать справку о функции chisq.test

# chisq.test ожидает на вход вектор или таблицу. В первом случае будет проверена гипотеза о р
авномерности распределения частот каждого элемента вектора, во втором случае будет проверена
 гипотеза о независимости двух признаков.

# рассмотрим пример на других встроенных данных mtcars
# в переменной am хранится тип коробки передач машины
# в переменной vs - тип двигателя

# проверяем гипотезу о равномерном распределении типа коробки передач
chisq.test(mtcars$am)
```

```
## Warning in chisq.test(mtcars$am): Chi-squared approximation may be
## incorrect
```

```
##
##  Chi-squared test for given probabilities
##
## data:  mtcars$am
## X-squared = 19, df = 31, p-value = 0.9549
```

```
# теперь о взаимосвязи двух признаков
t <- table(mtcars$am, mtcars$vs)
t
```

```
##
##      0  1
##   0 12  7
##   1  6  7
```

```
chisq.test(t)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  t
## X-squared = 0.34754, df = 1, p-value = 0.5555
```

```
# результат выполнения теста можно сохранить в переменную
fit <- chisq.test(t)

# теперь fit это список, в котором хранятся различные показатели анализа
str(fit)
```

```
## List of 9
##  $ statistic: Named num 0.348
##   ..- attr(*, "names")= chr "X-squared"
##  $ parameter: Named int 1
##   ..- attr(*, "names")= chr "df"
##  $ p.value  : num 0.556
##  $ method   : chr "Pearson's Chi-squared test with Yates' continuity correction"
##  $ data.name: chr "t"
##  $ observed : 'table' int [1:2, 1:2] 12 6 7 7
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "0" "1"
##   .. ..$ : chr [1:2] "0" "1"
##  $ expected : num [1:2, 1:2] 10.69 7.31 8.31 5.69
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "0" "1"
##   .. ..$ : chr [1:2] "0" "1"
##  $ residuals: table [1:2, 1:2] 0.401 -0.485 -0.455 0.55
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "0" "1"
##   .. ..$ : chr [1:2] "0" "1"
##  $ stdres   : table [1:2, 1:2] 0.952 -0.952 -0.952 0.952
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "0" "1"
##   .. ..$ : chr [1:2] "0" "1"
##  - attr(*, "class")= chr "htest"
```

```
# изучите справку, чтобы узнать подробее о том, что сохраняется в переменную fit

# Применить точный критерий Фишера также очень просто
fisher_result <- fisher.test(t)
fisher_result
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  t
## p-value = 0.4727
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##   0.3825342 10.5916087
## sample estimates:
## odds ratio
##   1.956055
```

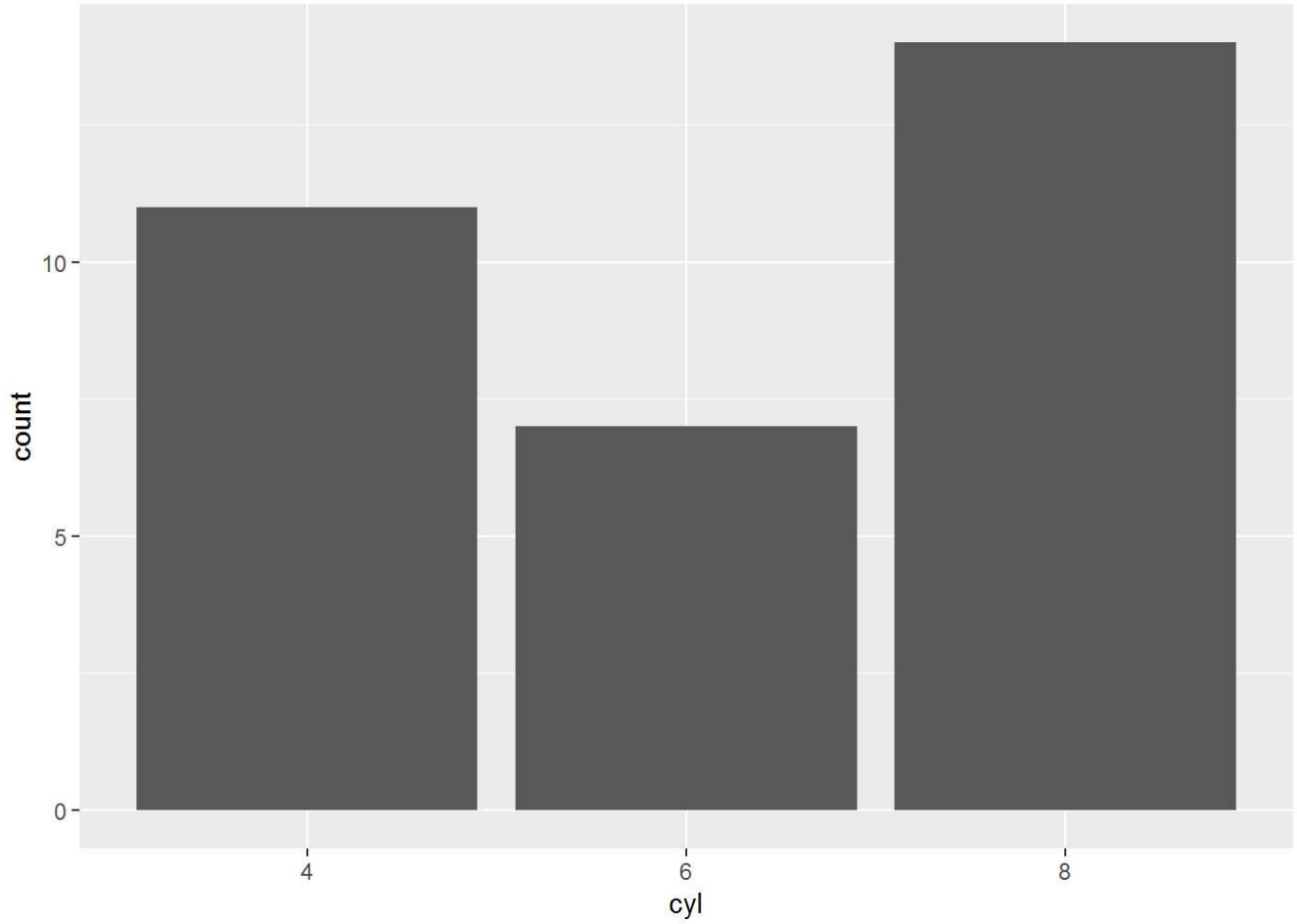
```
str(fisher_result)
```

```
## List of 7
##  $ p.value    : num 0.473
##  $ conf.int   : atomic [1:2] 0.383 10.592
##   ..- attr(*, "conf.level")= num 0.95
##  $ estimate   : Named num 1.96
##   ..- attr(*, "names")= chr "odds ratio"
##  $ null.value : Named num 1
##   ..- attr(*, "names")= chr "odds ratio"
##  $ alternative: chr "two.sided"
##  $ method     : chr "Fisher's Exact Test for Count Data"
##  $ data.name  : chr "t"
##  - attr(*, "class")= chr "htest"
```

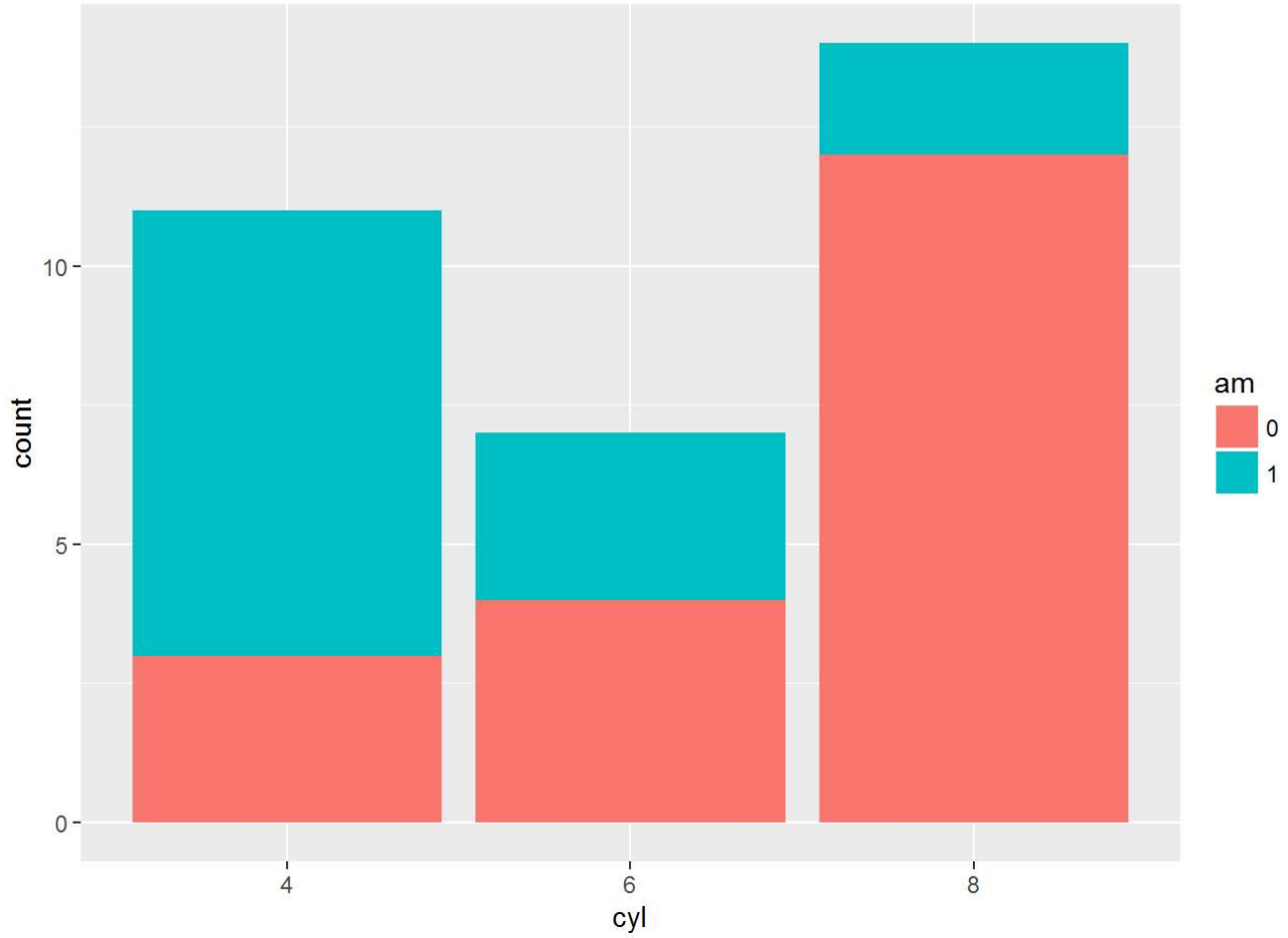
Визуализация номинативных переменных

Рассмотрим несколько примеров графикоф для анализа номинативных данных.

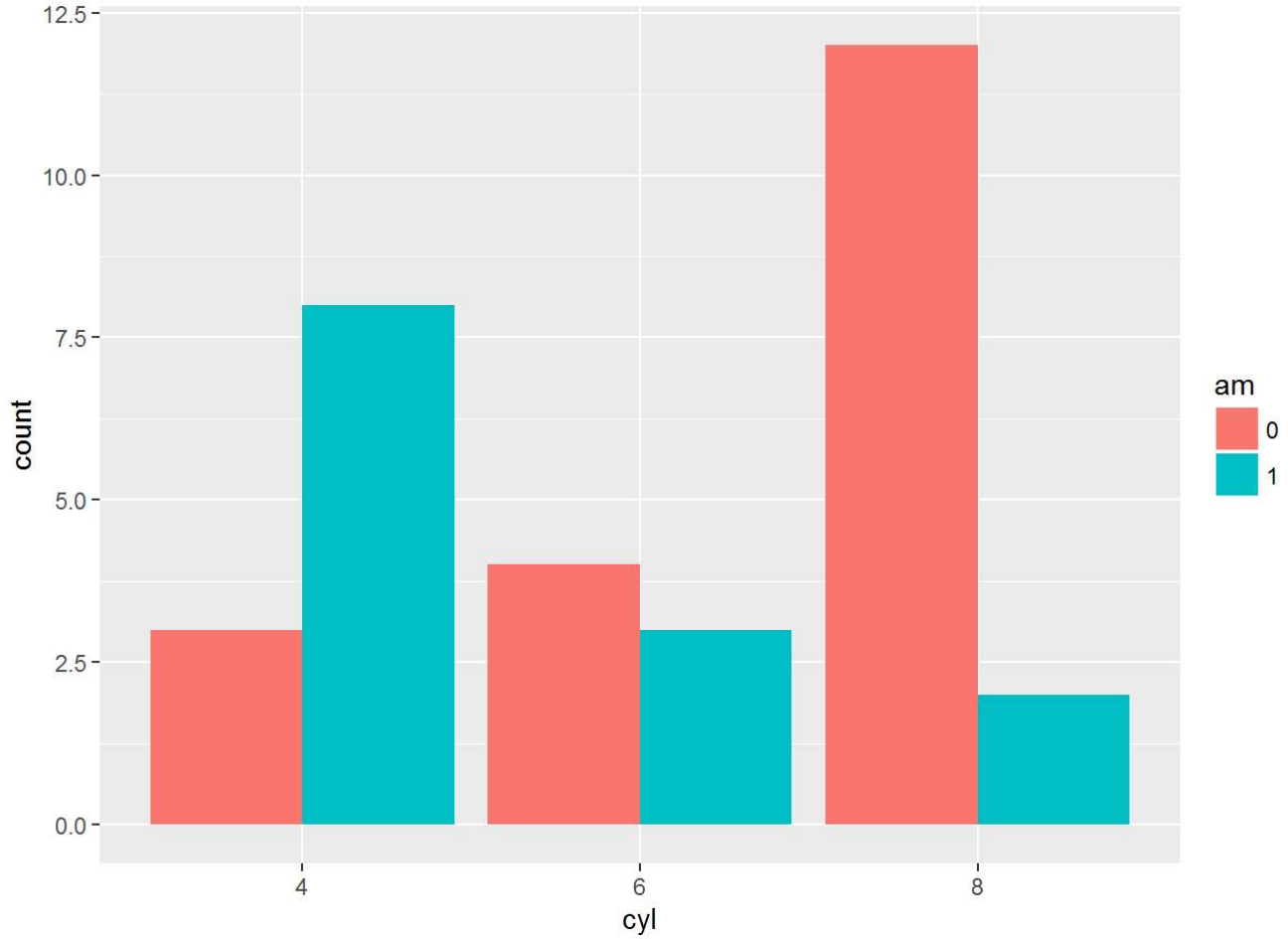
```
# сначала переведем наши номинативные переменные в фактор
mtcars$am <- factor(mtcars$am)
mtcars$vs <- factor(mtcars$vs)
mtcars$cyl <- factor(mtcars$cyl)


# будем использовать библиотеку ggplot для построения графиков

#install.packages("ggplot2")
library(ggplot2)

# построим простую гистограмму частот
ggplot(mtcars, aes(x = cyl)) +
    geom_bar()
```



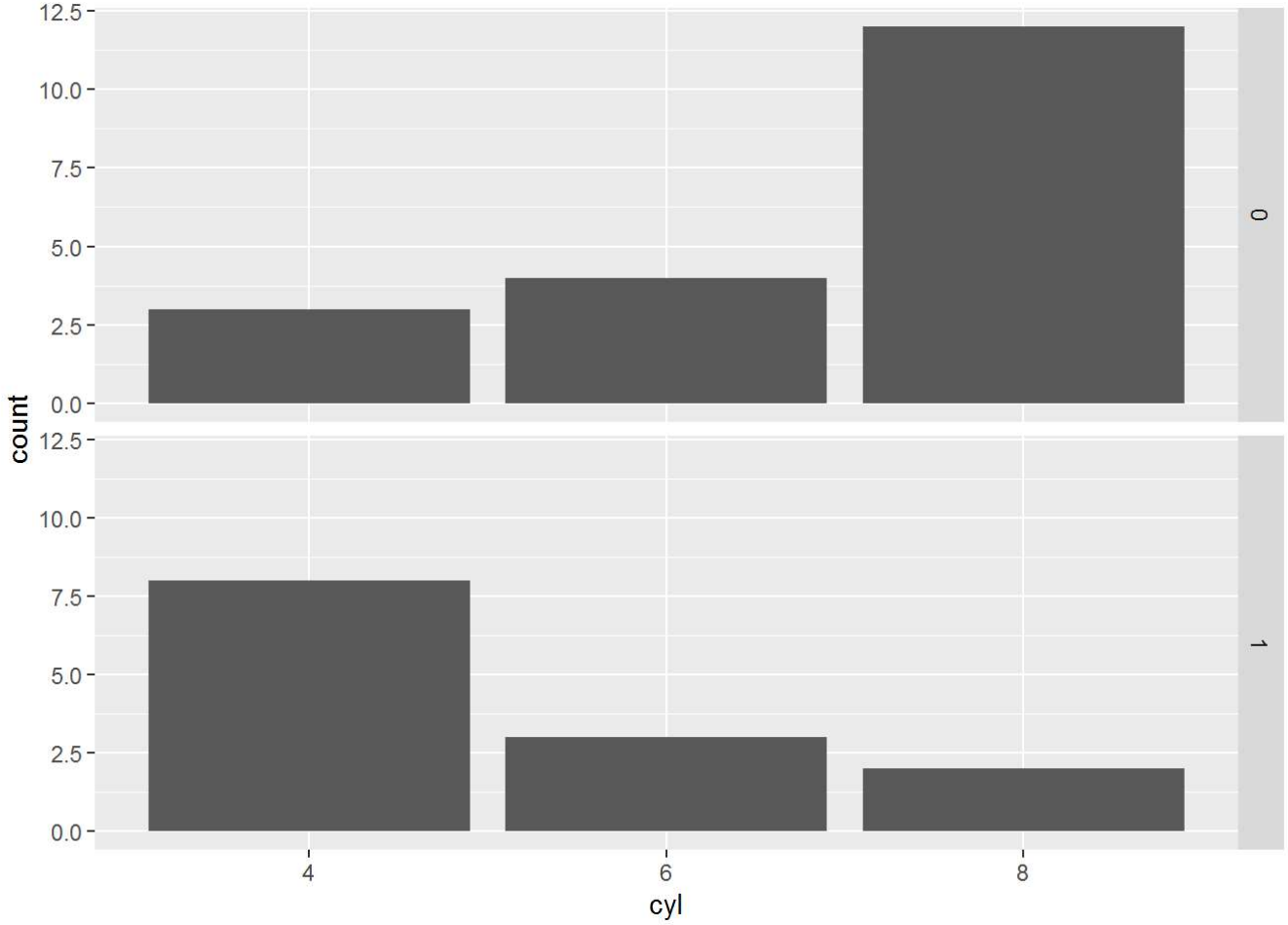

```
# добавим другие переменные на график
# три разных варианта
ggplot(mtcars, aes(x = cyl, fill = am)) +
    geom_bar()
```



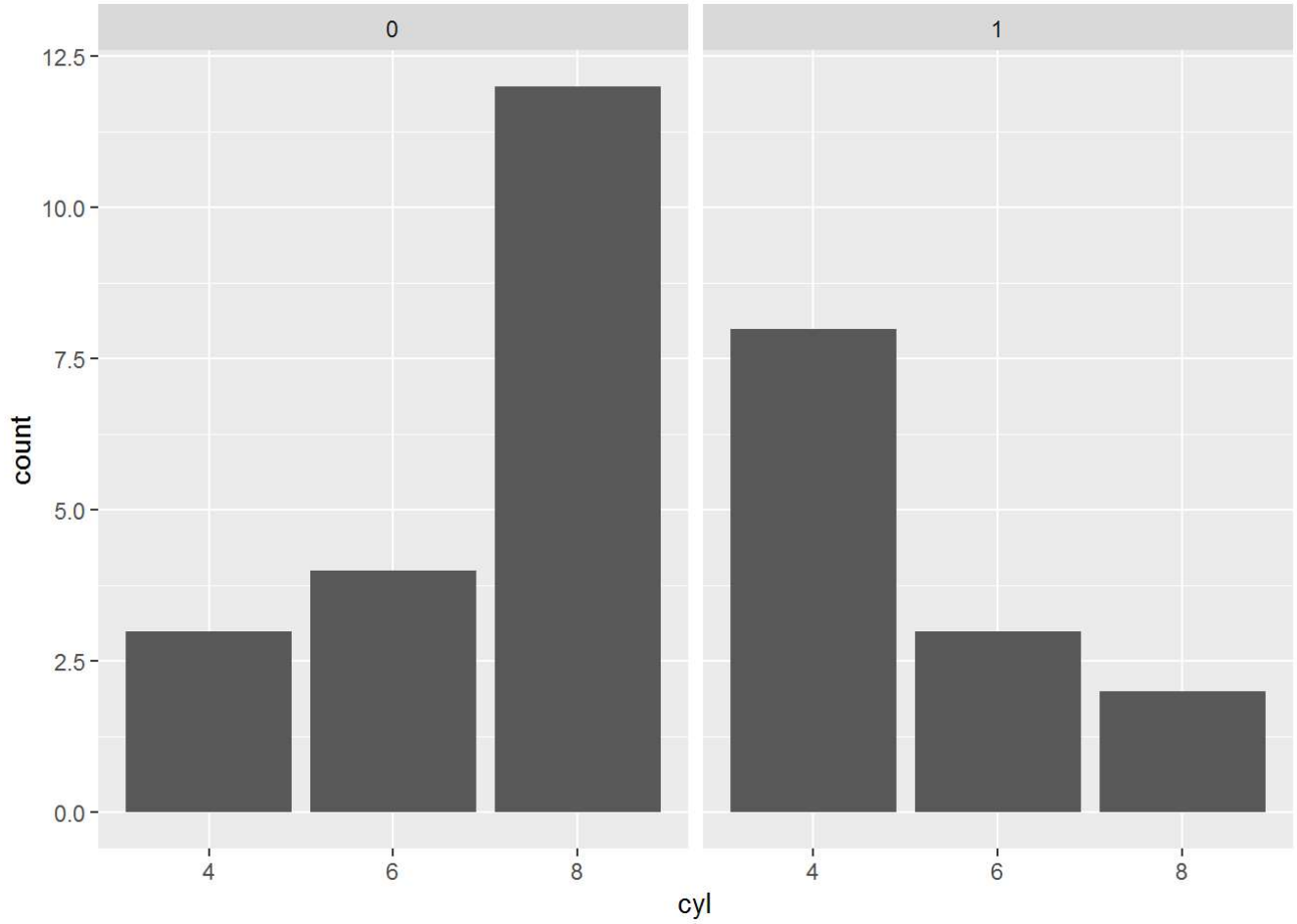

```
ggplot(mtcars, aes(x = cyl, fill = am)) +
    geom_bar(position = 'dodge')
```



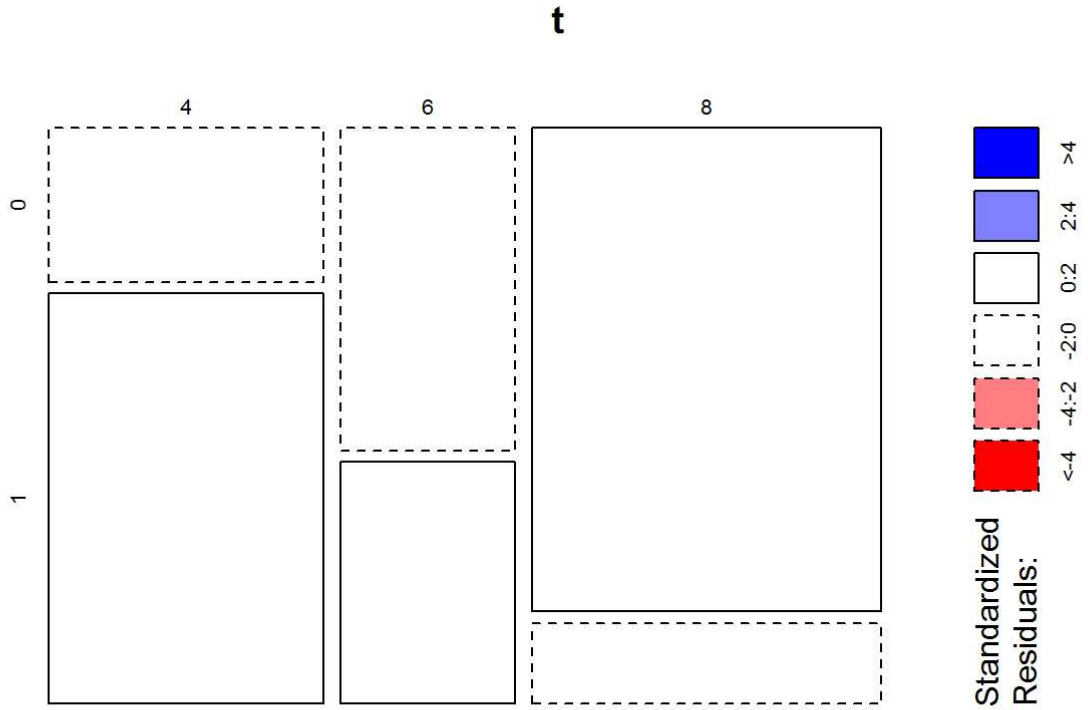
```
ggplot(mtcars, aes(x = cyl)) +
    geom_bar() +
    facet_grid(am ~ .)
```




```
ggplot(mtcars, aes(x = cyl)) +
    geom_bar() +
    facet_grid(. ~ am)
```




```
# построение мозаичного графика
t <- table(mtcars$cyl, mtcars$am)
mosaicplot(t, shade = T)
```


t
4
6
8
0
1
Standardized Residuals:
<-4
-4:-2
-2:0
0:2
2:4
>4