Иерархический кластерный анализ и метод k-means

Алла Тамбовцева

Установка библиотек и загрузка данных

Установим необходимые библиотеки (считаем, что `tidyverse`, которая включает в себя `ggplot2` для графиков уже установлена):

```
install.packages("factoextra")
install.packages("NbClust")
install.packages("fossil")
```

Загрузим данные по ценам на квартиры в Москве из файла `flats.csv` и удалим строки с пропущенными значениями (их нет, но оставим код для универсальности):

```
flats <- read.csv(file.choose())
flats <- na.omit(flats)
```

Переменные в файле:

- `price`: цена квартиры, в 1000$;
- `totsp`: общая площадь, в кв.метрах;
- `livesp`: жилая площадь, в кв.метрах;
- `kitsp`: площадь кухни, в кв.метрах;
- `dist`: расстояние до центра города, в км;
- `metrdist`: расстояние до метро, в мин;
- `walk`: шаговая доступность до метро, 0 или 1;
- `brick`: дом из кирпича или аналогичного «капитального» материала, 0 или 1;
- `floor`: удобный ли этаж (не первый и последний), 0 или 1.

Посмотрим на переменные и проверим, что все переменные корректных типов (например, числа считаны как числа, а не как текст):

```
str(flats)
```

```
## 'data.frame':    2040 obs. of  9 variables:
##  $ price   : int  730 477 350 410 50 340 285 470 135 470 ...
##  $ totsp   : int  134 119 80 102 63 83 90 146 105 120 ...
##  $ livesp  : int  102 93 44 75 47 49 61 86 58 75 ...
##  $ kitsp   : num  10 15 14 10 7 19 6 16 21 18 ...
##  $ dist    : num  9 9 10.5 6.5 12 9 14.5 14.5 13.5 9 ...
##  $ metrdist: int  10 1 10 5 20 5 10 5 15 10 ...
##  $ walk    : int  1 1 1 1 0 1 1 1 1 1 ...
##  $ brick   : int  1 0 1 1 1 1 0 1 1 1 ...
##  $ floor   : int  1 1 0 1 0 1 1 1 1 1 ...
```

Примечание: тип `int` — целочисленный (`integer`), тип `num` — просто числовой (`numeric`), может включать в себя как дробные, так и целочисленные значения.

Описание данных

Посмотрим на описательные статистики по всем столбцам:

```
summary(flats)
```

```
##      price            totsp           livesp           kitsp      
##  Min.   : 50.0   Min.   : 44.00   Min.   : 28.00   Min.   : 5.000  
##  1st Qu.: 95.0   1st Qu.: 62.00   1st Qu.: 42.00   1st Qu.: 7.000  
##  Median :115.0   Median : 73.50   Median : 45.00   Median : 9.000  
##  Mean   :127.5   Mean   : 73.08   Mean   : 46.34   Mean   : 8.899  
##  3rd Qu.:142.0   3rd Qu.: 79.00   3rd Qu.: 50.00   3rd Qu.:10.000  
##  Max.   :730.0   Max.   :192.00   Max.   :102.00   Max.   :25.000  
##       dist          metrdist          walk            brick       
##  Min.   : 3.00   Min.   : 1.000   Min.   :0.0000   Min.   :0.000  
##  1st Qu.: 9.00   1st Qu.: 5.000   1st Qu.:0.0000   1st Qu.:0.000  
##  Median :12.00   Median : 7.000   Median :1.0000   Median :0.000  
##  Mean   :11.02   Mean   : 8.117   Mean   :0.6858   Mean   :0.323  
##  3rd Qu.:13.50   3rd Qu.:10.000   3rd Qu.:1.0000   3rd Qu.:1.000  
##  Max.   :17.00   Max.   :20.000   Max.   :1.0000   Max.   :1.000  
##      floor       
##  Min.   :0.0000  
##  1st Qu.:1.0000  
##  Median :1.0000  
##  Mean   :0.7907  
##  3rd Qu.:1.0000  
##  Max.   :1.0000
```

С описательными статистиками все знакомы, отметим здесь только то, что в бинарных показателях среднее корректно интерпретировать как долю «единиц», то есть мы можем определить выборочную долю квартир в шаговой доступности от метро и проч.

Загрузим библиотеку `tidyverse` и визуализируем количественные показатели.

```
library(tidyverse)
```

Пояснения к коду для графиков с `ggplot()`

В аргумент `data` записываем название датафрейма, внутри `aes()` указываем, какие переменные идут по осям `x` и `y`. Далее через `+` добавляем слои, отвечающие за тип и внешний вид графика:

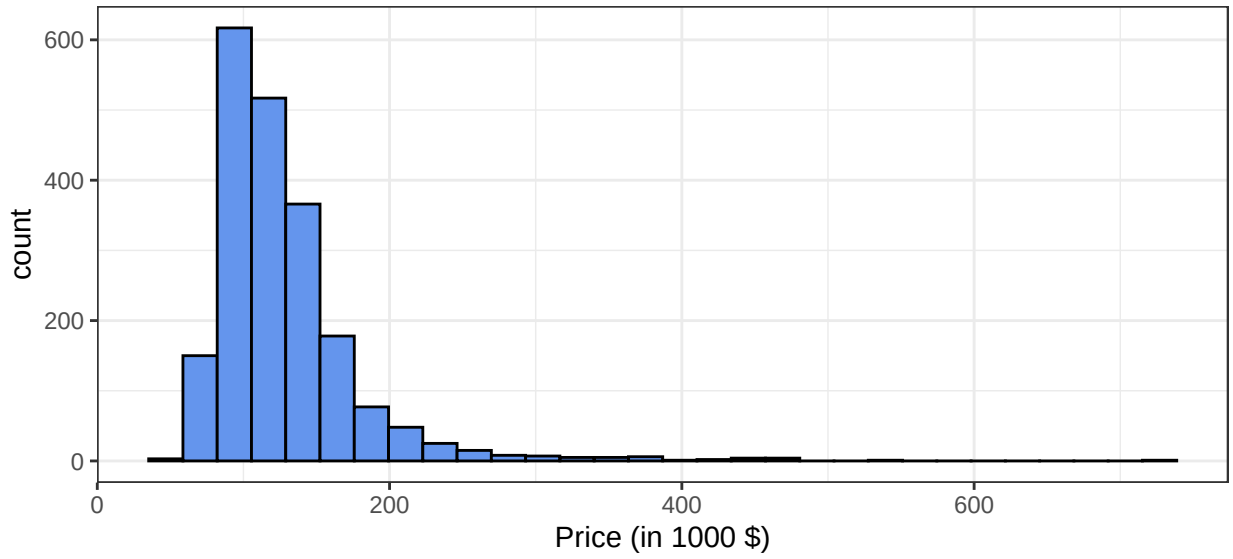
— `geom_histogram()`: гистограмма, фиксируем цвет заливки `fill` и цвет границ столбцов `color`;

— `theme_bw()`: чёрно-белая тема для фона и разметки;

— `labs()`: заголовки и подписи к осям.

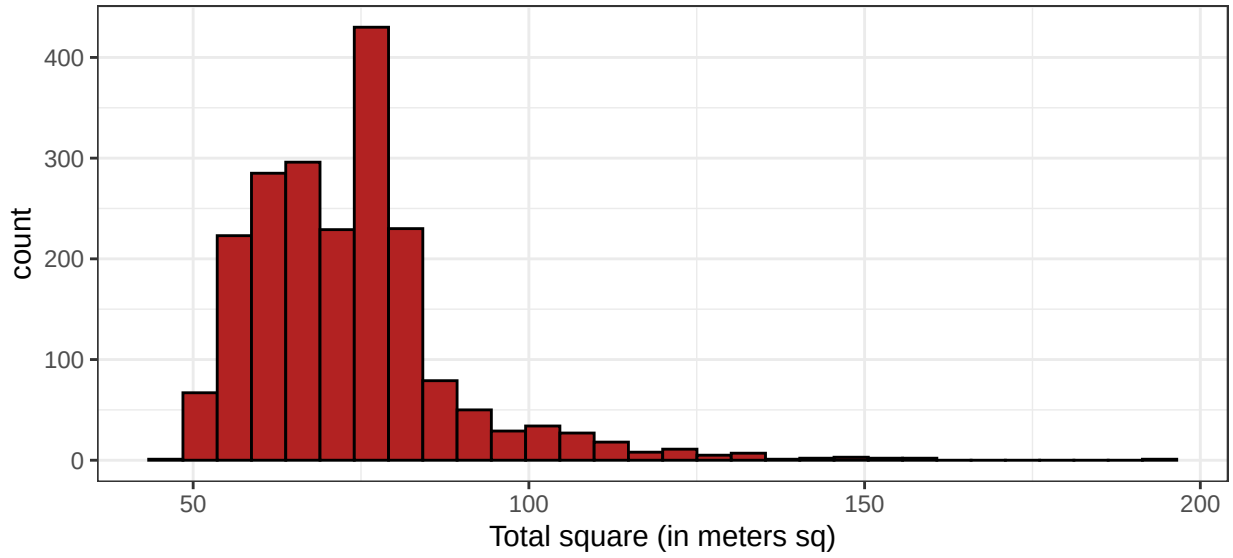
Построим гистограмму для цен на квартиры:

```
ggplot(data = flats, aes(x = price)) +
  geom_histogram(fill = "cornflowerblue", color = "black") +
  theme_bw() +
  labs(x = "Price (in 1000 $)")
```



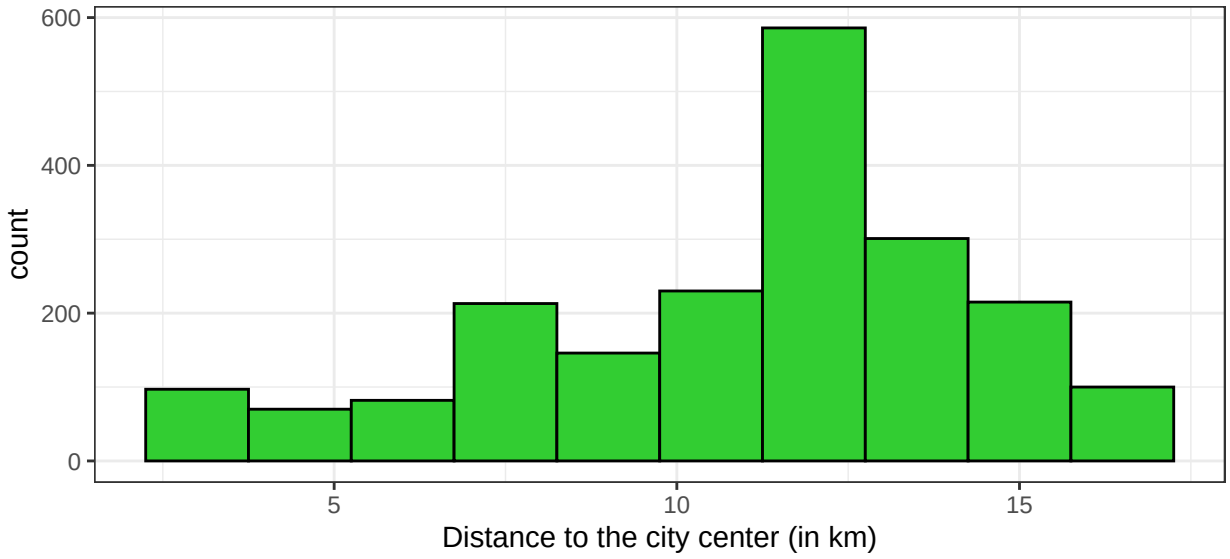
Гистограмма для площади квартир:

```
ggplot(data = flats, aes(x = totsp)) +
  geom_histogram(fill = "firebrick", color = "black") +
  theme_bw() +
  labs(x = "Total square (in meters sq)")
```



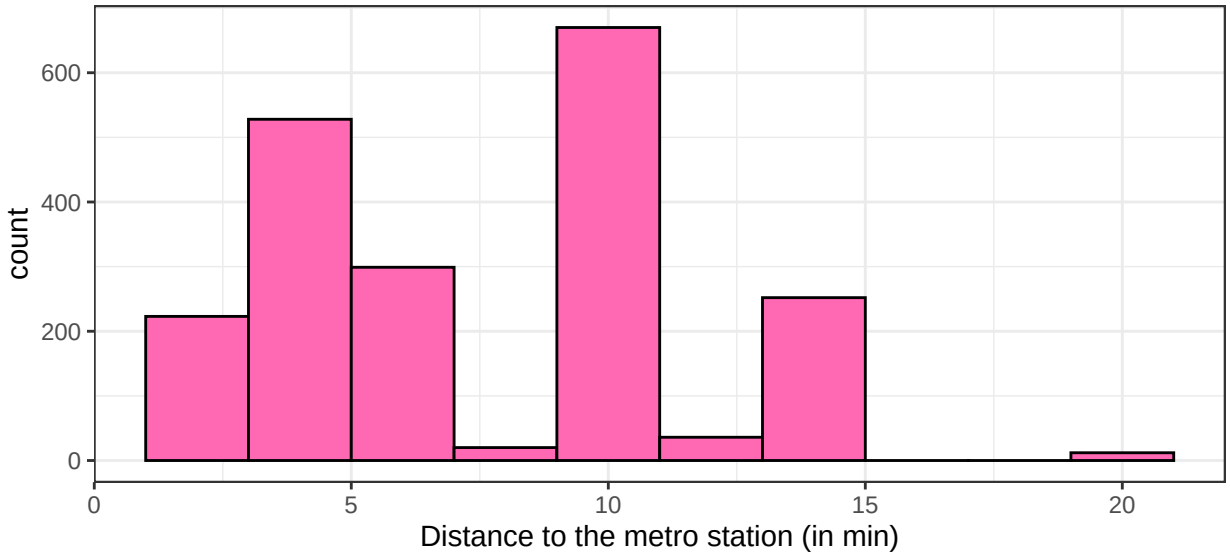
Гистограмма для расстояния до центра:

```
ggplot(data = flats, aes(x = dist)) +
  geom_histogram(binwidth = 1.5, fill = "limegreen", color = "black") +
  theme_bw() +
  labs(x = "Distance to the city center (in km)")
```



Гистограмма для расстояния до метро:

```
ggplot(data = flats, aes(x = metrdist)) +
  geom_histogram(binwidth = 2, fill = "hotpink", color = "black") +
  theme_bw() +
  labs(x = "Distance to the metro station (in min)")
```



Иерархический кластерный анализ

Теперь перейдём к иерархическому кластерному анализу. Сформируем матрицу расстояний через `dist()`, предварительно прошкалировав наши данные с помощью `scale()`:

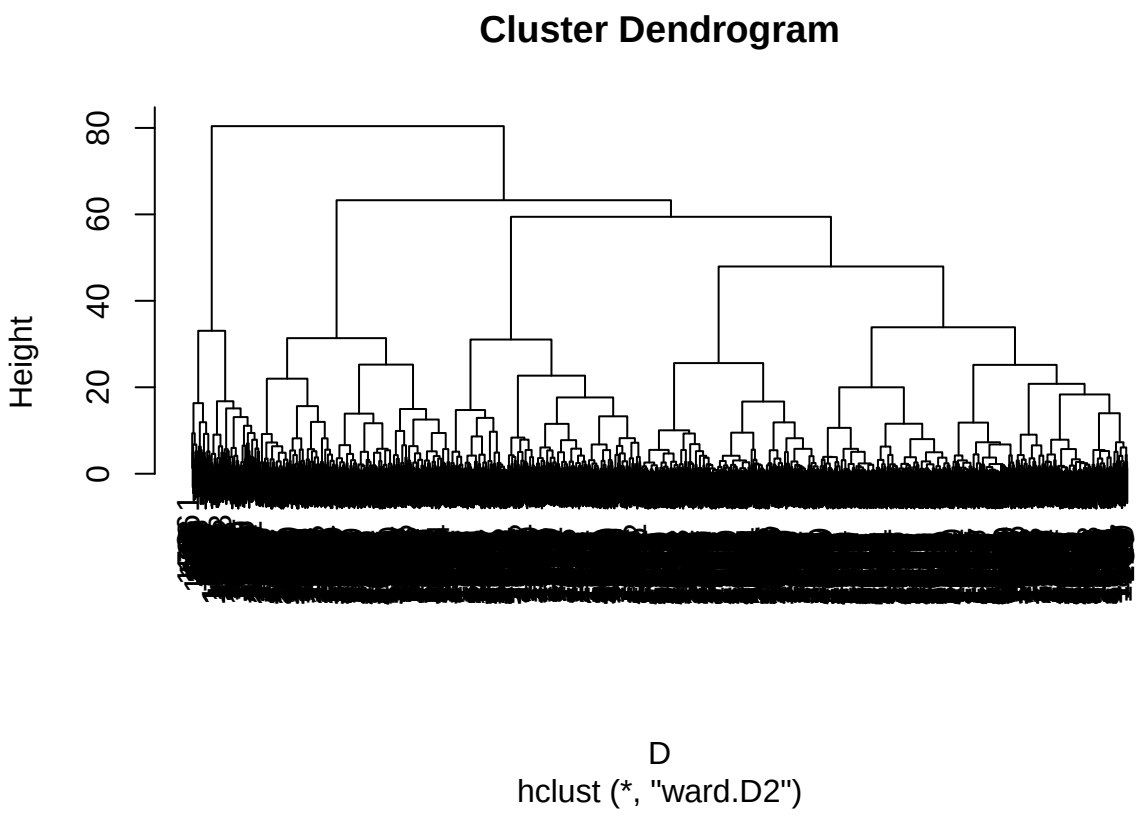
```
D <- dist(scale(flats))
```

По умолчанию функция `dist` вычисляет евклидовы расстояния, они нас устраивают. В качестве метода агрегирования возьмём метод Уорда как наиболее эффективный, но учтём, что этот метод на входе требует квадраты евклидовых расстояний (опция `ward.D2`, не просто `ward.D`):

```
hc_ward <- hclust(D, method = "ward.D2")
```

Построим дендрограмму (она будет довольно дикая из-за большого числа наблюдений, но нам её хватит для определения крупных кластеров):

```
plot(hc_ward)
```



Если мы посмотрим внимательно на дендрограмму, мы сможем выделить разное число кластеров. Выбор в данном случае зависит от того, насколько общее мы хотим получить деление (на мой взгляд, остановиться на двух группах здесь было бы слишком просто), и от содержательных соображений.

Выберем три кластера и «разрежем» дендрограмму на высоте *Height*, которое соответствует делению на три группы. R сам его посчитает с помощью функции `cutree()` и припишет наблюдениями, относящимся к разным кластерам, соответствующие метки:

```
ward <- cutree(hc_ward, k = 3)
```

Сделаем эти метки факторными (качественными) и добавим их в исходный датафрейм:

```
flats$ward <- factor(ward)
```

Содержательная интерпретация кластеров

Оценим, что получилось. Посмотрим на описательные статистики по группам.

Пояснения к коду для описания групп

Функция `summarise_at()` позволяет применить функции для описания или агрегирования данных, указанные внутри `.funs()` к фиксированному набору переменных, которые находятся внутри `vars()`.

Здесь взяты переменные с `price` до `floor`, и применяем мы функции `median()` и `mean()`. Можно было бы применить их одновременно, записать несколько функций в виде вектора, но тогда выдача была бы менее удобной, плюс, всё равно состояла бы из двух частей.

Сначала посмотрим на медианные значения показателей:

```
flats %>% group_by(ward) %>% summarise_at(vars(price:floor),
                                          .funs = median)
```

```
## # A tibble: 3 x 10
##   ward  price totsp livesp kitsp  dist metrdist  walk brick floor
##   <fct> <dbl> <dbl>  <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl> <dbl>
## 1 1      210. 106.    62.5    15    10        7     1     1     1
## 2 2      105   68     44       8    12        8     1     0     0
## 3 3      115   72.5   45       8    12        7     1     0     1
```

Что мы здесь видим? Однозначно одно: в первом кластере собраны самые дорогие и самые большие по площади квартиры. Про остальные два кластера пока всё не так ясно. Поэтому теперь посмотрим на средние значения:

```
flats %>% group_by(ward) %>% summarise_at(vars(price:floor),
                                          .funs = mean)
```

```
## # A tibble: 3 x 10
##   ward  price totsp livesp kitsp  dist metrdist  walk brick floor
##   <fct> <dbl> <dbl>  <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl> <dbl>
## 1 1      240. 108.    63.2 15.1   9.94     6.91 0.757 0.622 0.926
## 2 2      112.  68.7   44.4  8.07 10.9      8.33 0.700 0.363 0
## 3 3      121.  70.8   45.2  8.51 11.2      8.18 0.675 0.282 1
```

Вот здесь уже всплывает очень интересный факт: так как для бинарных показателей среднее совпадает с долей «единиц», получается, что во втором кластере 100% квартир находятся на неудобных этажах, а в третьем — на удобных (и да, квартиры во втором кластере логичным образом, в среднем, дешевле).

Вернёмся к цене и визуализируем различия в её распределении по группам.

Пояснения к коду для графиков с `ggplot()`

В аргумент `data` записываем название датафрейма, внутри `aes()` указываем, какие переменные идут по осям `x` и `y`. В данном случае по оси `x` указаны группы по результатам кластерного анализа, а по оси `y` – цена (классический ящик с усами – вертикальный график). Кроме того, внутри `aes()` мы зафиксировали, что цвет заливки ящиков должен различаться в зависимости от группы.

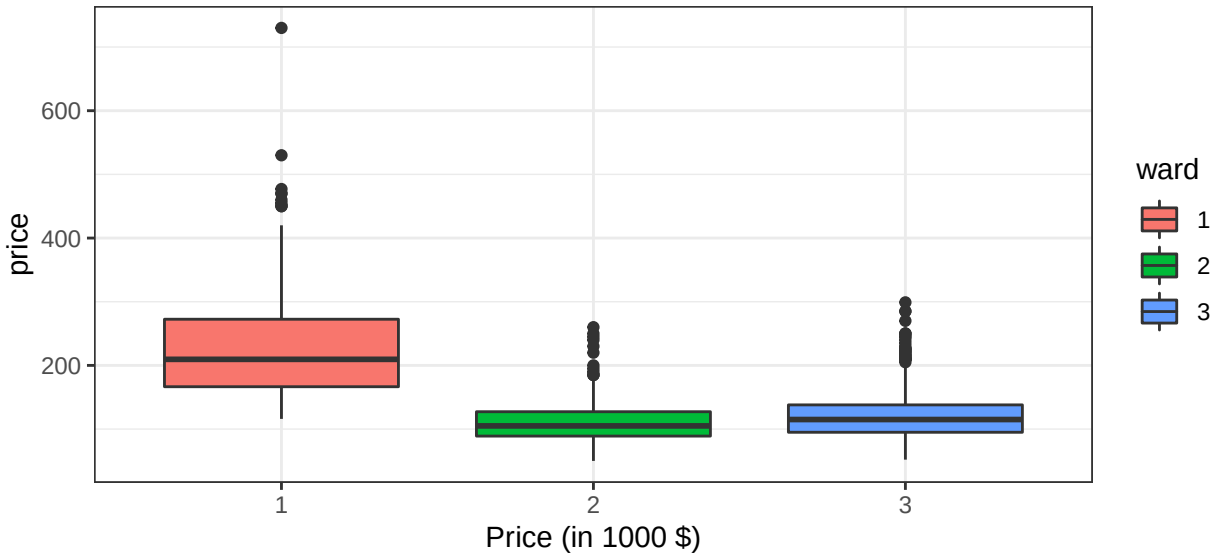
Далее добавляем слои, отвечающие за тип и внешний вид графика:

— `geom_boxplot()`: ящик с усами;

— `theme_bw()`: чёрно-белая тема для фона и разметки;

— `labs()`: заголовки и подписи к осям.

```
ggplot(data = flats, aes(x = ward, y = price, fill = ward)) +
  geom_boxplot() +
  theme_bw() +
  labs(x = "Price (in 1000 $)")
```



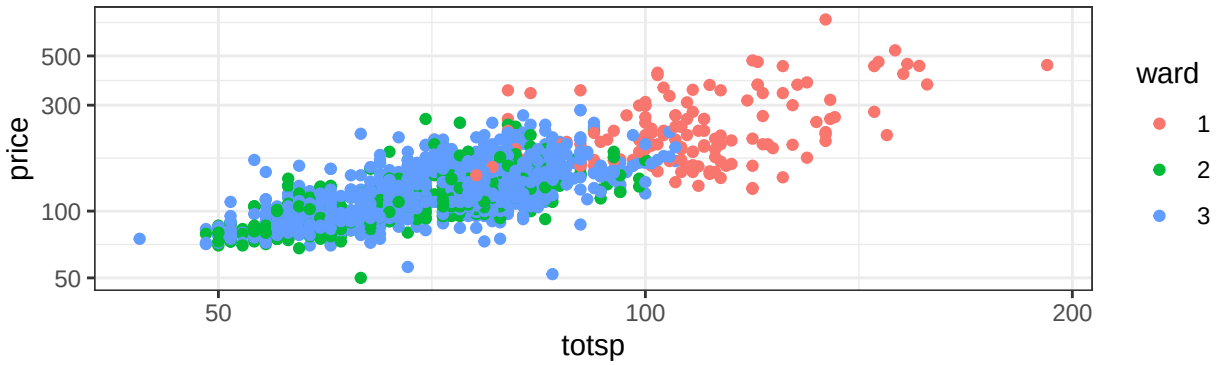
Разница в цене квартир во второй и третьей группе не очень большая, но это не означает, что кластеризация у нас получилась плохая. Мы изучаем методы многомерного анализа данных, и сами группы свободно могут отличаться друг от друга по другим измерениям.

Попробуем посмотреть, образуют ли точки, относящиеся к разным кластерам, группы на диаграмме рассеивания.

Пояснения к коду для графиков с `ggplot()`

Также как и ранее, добавляем в `aes()` указание на то, что цвет точек зависит от группы в столбце `ward`. Плюс, чтобы сократить разброс точек на графике (есть нетипичные значения, супер-дорогая огромная квартира), добавляем слои `scale_x_log10()` и `scale_y_log10()`, чтобы по горизонтальной и вертикальной осям указывались логарифмированные показатели вместо исходных.

```
ggplot(data = flats, aes(x = totsp, y = price, color = ward)) +
  geom_point() + theme_bw() +
  scale_x_log10() + scale_y_log10()
```



Опять видим такую картину: первый кластер явно выделяется (большие и дорогие квартиры), два других перемешаны с точки зрения цены и площади.

Небольшое отступление по заявкам слушателей про трёхмерные графики.

По-хорошему, чтобы действительно показать различия между вторым и третьим кластером на диаграмме рассеивания такого вида, нам понадобится трёхмерный график. Установим библиотеку `rgl`:

```
install.packages("rgl")
```

Теперь построим сам график. Строки `setupKnitr()` и `rglwidget()` нужны для отображения графика HTML/Word/PDF при связывании файла RMarkdown, сам по себе он открывается в новом интерактивном окне. В Word/PDF график будет представлен в статичном виде, а в HTML сохранится интерактив, график можно будет поворачивать, увеличивать и прочее.

Сначала зафиксируем цвета точек и сохраним их в отдельный столбец датафрейма (это обычная операция для графиков за рамками `ggplot2`).

```
mycolors <- c('#F8766D', '#00BA38', '#619CFF')
flats$color <- mycolors[as.numeric(flats$ward)]
```

Выбранные цвета в `mycolors` — это цвета, которые по умолчанию использовались `ggplot()` на диаграмме выше, записанные в формате HEX.

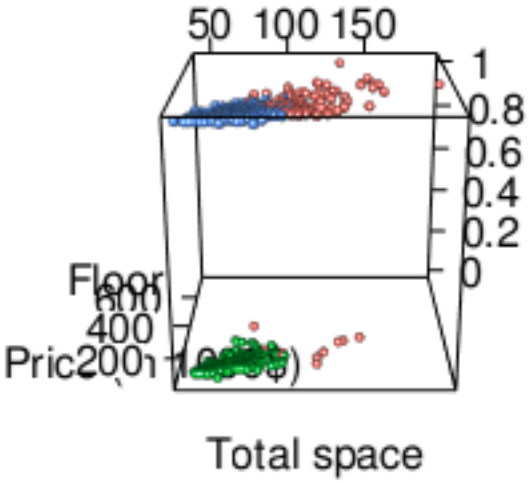
Наконец, переходим к графику (`type` - тип точек *sphere*, далее фиксируем радиус этих сфер):

```
library(rgl)

setupKnitr()

plot3d(
  x = flats$totsp,
  y = flats$price,
  z = flats$floor,
  col = flats$color,
  type = 's',
  radius = 8,
  xlab="Total space",
  ylab="Price (in 1000$)",
  zlab="Floor")

rglwidget()
```



Вот теперь точно видно, что у нас есть два «висящих» облака точек, в верхнем облаке, соответствующем квартирам на удобных этажах, находятся все наблюдения из третьего кластера, в нижнем — наблюдения из второго кластера. Ну, а наблюдения из первого кластера встречаются и там, и там (довольно жизненно: дорогие квартиры можно встреить на любом этаже).

Выбор количества кластеров

Рассмотрим два основных метода: метод согнутого локтя (*Elbow method*) и силуэтный метод (*Silhouette method*).

Пояснения к коду для графиков с `fviz_nbclust()`

Для обоих методов мы используем функцию `fviz_nbclust()` из библиотеки `factoextra`. Так как метки кластеров из столбца `ward` для кластеризации нам не нужны, а нужны только исходные данные, из датафрейма `flats` мы выбираем столбцы с первого по девятый.

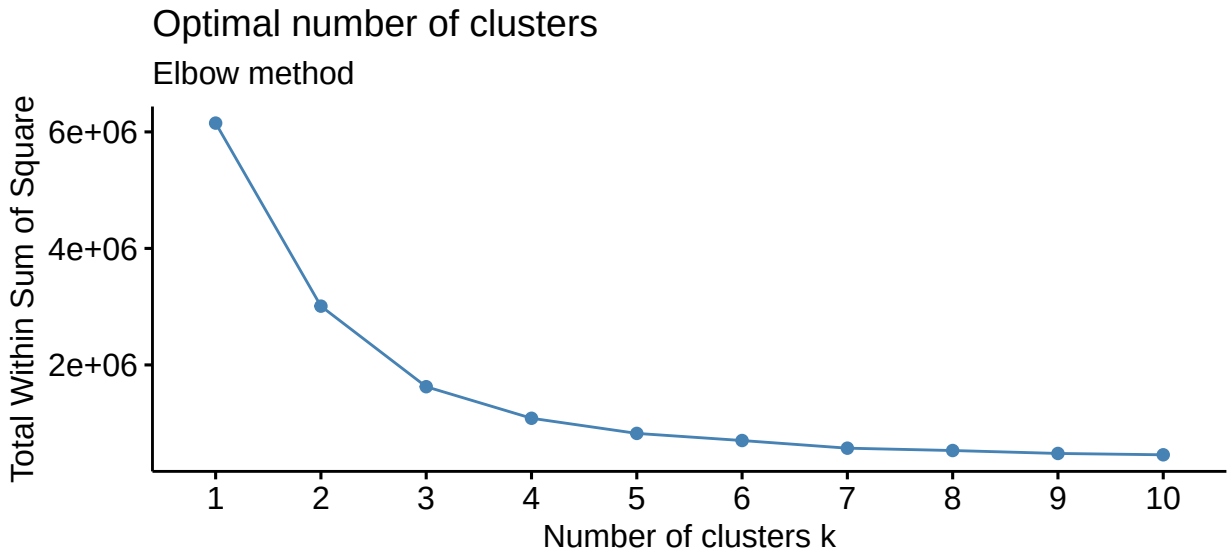
Чтобы выбрать количество кластеров, нам нужно оценить качество кластеризации при каждом возможном количестве кластеров (в разумных пределах, количество групп не более 10). Выбираем метод кластеризации `kmeans`, он является более точным и требует на входе количество кластеров (будет изменяться от 1 до 10).

Для метода согнутого локтя выбираем меру разброса `wss`, общую внутригрупповую сумму квадратов отклонений, так как логика использования этого метода предполагает работу с различиями внутри групп.

Для силуэтного метода логичным образом выбираем `silhouette`. Силуэт — мера сходства наблюдений в кластере. Подробнее про силуэты и их вычисление можно почитать здесь.

Метод согнутого локтя:

```
library(factoextra)
fviz_nbclust(flats[1:9], kmeans, method = "wss") +
  labs(subtitle = "Elbow method")
```

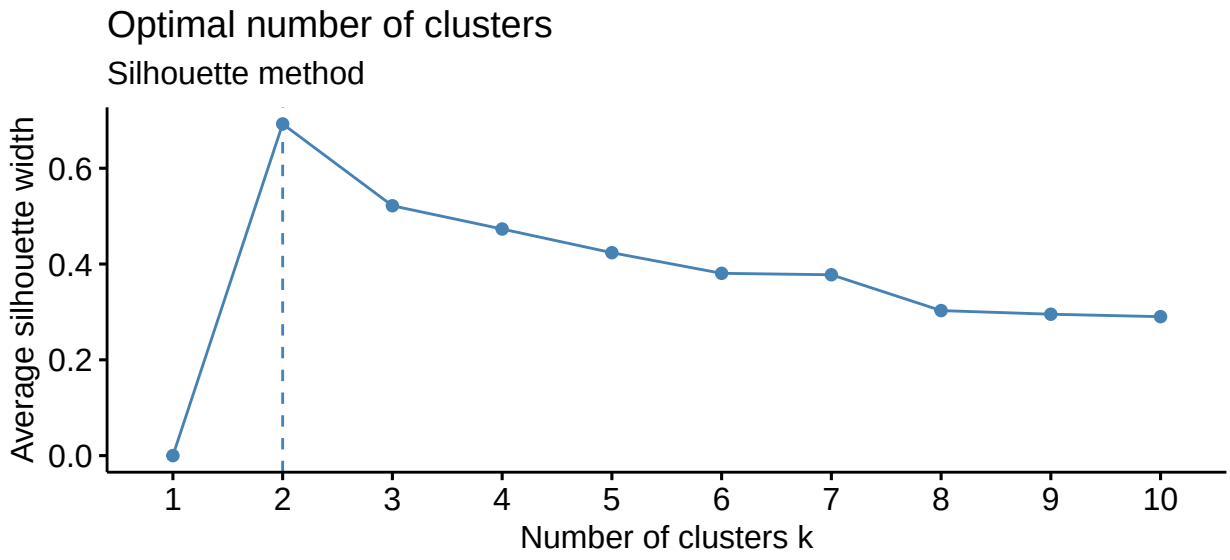


Как следует из названия этого метода, на графике мы должны найти «локоть» (или «колено»), то есть такое число кластеров, начиная с которого общий внутригрупповой разброс начинает незначительно уменьшаться. Другими словами, мы стараемся подобрать такое число групп, чтобы наблюдения внутри групп были минимально отдалены друг от друга и чтобы при этом число этих групп не было избыточно большим (зачем дробить наблюдения на большее число групп, если схожесть наблюдений внутри при этом несильно увеличивается).

Выбранное нами ранее количество кластеров $k = 3$ вполне согласуется с этим графиком. Слишком общее деление на две группы нам неинтересно, а «локоть» здесь находится в районе $k = 3$ и $k = 4$. Можете проверить самостоятельно, что при выделении четырёх кластеров они будут менее интерпретируемыми.

Силуэтный метод:

```
fviz_nbclust(flats[1:9], kmeans, method = "silhouette") +
  labs(subtitle = "Silhouette method")
```



На этом графике пик наблюдается при количестве кластеров 2, но мы снова пойдём дальше и выберем значение $k = 3$, при котором происходит следующий излом. После этого излома средняя ширина силуэта (специфическая мера различия наблюдений в группе) убывает несильно и довольно равномерно,

то есть, при делении на большее число групп эти группы не будут становиться более стабильными с точки зрения разброса значений.

В R также доступен менее классический, не очень устойчивый, способ выявления числа кластеров. Он доступен в библиотеке `NbClust` и делает следующее: запускает кластеризацию методом `kmeans` с разным числом кластеров (сами выбираем минимальное и максимальное), оценивает качество полученной кластеризации с помощью разных алгоритмов и сообщает количество кластеров, за которое «проголосовало» большинство используемых алгоритмов.

```
# takes time, not executed in this file

library(NbClust)
res <- NbClust(flats[1:9],
               min.nc = 2, max.nc = 8,
               method = "kmeans")
fviz_nbclust(res)
```

Оценка качества кластеризации

Проверим с помощью классических статистических методов, есть ли различия между полученными нами тремя кластерами.

Например, проверим, есть ли разница в медианной цене на квартиры в трёх кластерах, с помощью критерия Краскелла-Уолииса (нормальности распределения цены не наблюдается):

```
kruskal.test(flats$price ~ flats$ward)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  flats$price by flats$ward
## Kruskal-Wallis chi-squared = 356.8, df = 2, p-value < 2.2e-16
```

P-value примерно 0, отвергаем гипотезу об отсутствии различий медиан/распределений, следовательно, распределение цен на квартиры отличается в трёх кластерах.

Теперь поработаем с качественными показателями `walk`, `brick`, `floor`. Сравнивать доли «единиц» попарно в каждой паре групп (1 vs 2, 1 vs 3, 2 vs 3) неэффективно, поэтому просто перейдём к таблицам сопряжённости и критерию хи-квадрат.

```
tab_walk <- table(flats$ward, flats$walk)
tab_walk
```

```
##
##       0   1
##   1  36 112
##   2 125 291
##   3 480 996
```

```
chisq.test(tab_walk)
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab_walk
## X-squared = 4.6507, df = 2, p-value = 0.09775
```

```
tab_brick <- table(flats$ward, flats$brick)
tab_brick
```

```
##
##        0    1
##   1   56   92
##   2  265  151
##   3 1060  416
```

```
chisq.test(tab_brick)
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab_brick
## X-squared = 74.825, df = 2, p-value < 2.2e-16
```

```
tab_floor <- table(flats$ward, flats$floor)
tab_floor
```

```
##
##        0    1
##   1   11  137
##   2  416    0
##   3    0 1476
```

```
chisq.test(tab_floor)
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab_floor
## X-squared = 1978.5, df = 2, p-value < 2.2e-16
```

В целом, на 5%-ном (в первом случае на 10%-ном) уровне значимости мы можем считать, что есть связь между нашим делением квартир на группы и качественными характеристиками этих квартир, таких как шаговая доступность метро, материал дома, удоство этажа.

Кластеризация методом k-средних

Проведём кластеризацию методом k-средних, этот метод считается более точным, плюс, теперь с количеством групп мы точно определились, его использование стало возможным. Метки кластеров по итогам реализации алгоритма также сохраним в датафрейм в факторном виде:

```
kclust <- kmeans(flats[1:9], 3)
flats$k <- factor(kclust$cluster)
```

Сравнение различных реализаций кластерного анализа

Сравним деление на группы, полученное методом Уорда и методом k-means:

```
flats %>% group_by(ward) %>% summarise_at(vars(price:floor),
                                          .funs = c(mean))
```

```
## # A tibble: 3 x 10
##   ward  price totsp livesp kitsp  dist metrdist  walk brick floor
```

```
##   <fct> <dbl> <dbl>  <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl> <dbl>
## 1 1      240. 108.    63.2 15.1   9.94     6.91 0.757 0.622 0.926
## 2 2      112.  68.7   44.4  8.07 10.9      8.33 0.700 0.363 0
## 3 3      121.  70.8   45.2  8.51 11.2      8.18 0.675 0.282 1
```

```
flats %>% group_by(k) %>% summarise_at(vars(price:floor),
                                       .funs = c(mean))
```

```
## # A tibble: 3 x 10
##   k     price totsp livesp kitsp  dist metrdist  walk brick floor
##   <fct> <dbl> <dbl>  <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl> <dbl>
## 1 1      103.  67.0   43.3  8.00 11.9      8.51 0.635 0.229 0.759
## 2 2      164.  83.0   51.1 10.4   9.33     7.40 0.788 0.511 0.851
## 3 3      352. 118.    70.2 15.3   8.47     6.23 0.827 0.596 0.923
```

В целом, основания для кластеризации в двух методах похожи (главную роль играет цена и площадь), но решать, какая из кластеризаций нам подходит больше, стоит на содержательном уровне. С одной стороны, метод k-means более точный, с другой стороны, решение, предлагаемое методом Уорда, более понятное и интерпретируемое, если вспомнить чёткое деление квартир по удобству этажей.

Если хочется получить более формальный результат сравнения делений на группы, можно воспользоваться индексом Ранда (*Rand's index*). Он показывает, какая доля наблюдений группируется одинаково в двух вариантах кластеризации (его вычисление довольно интуитивное, идёт сравнение двух множеств с метками кластеров, можно почитать здесь).

```
library(fossil)

rand.index(as.integer(flats$ward),
           as.integer(flats$k))
```

```
## [1] 0.555905
```

Обратите внимание: функция `rand.index()` принимает на вход целочисленные метки кластеров.

Сходство не очень высокое, но это ожидаемо: мы уже выяснили, что в методе Уорда одним из ключевых оснований для деления на группы был этаж, а в кластеризации методом k-means — цена. Тем не менее, с учётом различий в логике используемых алгоритмов, сходство более 50% можно считать довольно высоким, поэтому заключаем, что деление на три кластера вышло удачным, как с содержательной, так и со статистической точки зрения.