Кластерный анализ: оценка качества кластеризации

Алла Тамбовцева

3 мая 2018 г

На этом занятии мы попробуем сымитировать небольшое исследование, основанное на кластерном анализе: реализуем иерархический кластерный анализ, проверим качество получившейся кластеризации, определим, сколько кластеров все-таки выбрать и проведем кластеризацию, используя метод k-средних (*k-means*).

### Данные

Загрузим базу данных reg\_elect.csv, содержащую результаты президентских выборов 2018 года:

df <- read.csv(file.choose())

Посмотрим на нее:

View(df)

head(df)

## X region total invalid valid Baburin Grudinin  
## 1 1 Алтайский край 1822203 14203 1177056 7581 281978  
## 2 2 Амурская область 635083 5383 389170 2358 73485  
## 3 3 Архангельская область 914218 4880 536082 4448 51868  
## 4 4 Астраханская область 736206 4340 440386 2185 64047  
## 5 5 Белгородская область 1218896 8749 883685 5218 93102  
## 6 6 Брянская область 978509 7177 772301 4472 68375  
## Zhirinovsky Putin Sobchak Suraikin Titov Yavlinsky turnout turnout\_perc  
## 1 84785 770278 11788 7855 5532 7259 1191259 65.37466  
## 2 37909 264493 4428 2466 2080 1951 394553 62.12621  
## 3 46925 407190 10588 3842 4982 6239 540962 59.17210  
## 4 19339 342195 5060 2823 2233 2504 444726 60.40782  
## 5 49685 711392 8474 6534 4835 4445 892434 73.21658  
## 6 43940 636087 7463 4265 4175 3524 779478 79.65977  
## Baburin\_perc Grudinin\_perc Zhirinovsky\_perc Putin\_perc Sobchak\_perc  
## 1 0.6363855 23.670587 7.117260 64.66083 0.9895413  
## 2 0.5976383 18.624874 9.608088 67.03611 1.1222827  
## 3 0.8222389 9.588104 8.674362 75.27146 1.9572539  
## 4 0.4913138 14.401452 4.348520 76.94513 1.1377792  
## 5 0.5846931 10.432368 5.567358 79.71368 0.9495380  
## 6 0.5737173 8.771896 5.637106 81.60423 0.9574356  
## Suraikin\_perc Titov\_perc Yavlinsky\_perc  
## 1 0.6593864 0.4643826 0.6093553  
## 2 0.6250111 0.5271789 0.4944836  
## 3 0.7102162 0.9209519 1.1533158  
## 4 0.6347729 0.5021069 0.5630433  
## 5 0.7321550 0.5417768 0.4980760  
## 6 0.5471610 0.5356149 0.4520974

В базе данных 87 наблюдений, так как в ней помимо регионов сохранены результаты выборов на территории за пределами РФ и в городе Байконур.

**Пояснения по переменным:**

* total -- общее число зарегистрированных избирателей
* invalid -- число недействительных бюллетеней
* valid -- число действительных бюллетеней
* turnout -- явка (сумма действительных и недействительных бюллетеней)
* далее -- число голосов за каждого кандидата (по фамилиям)
* столбцы с \_perc -- явка в процентах и проценты голосов за кандидатов

### Иерархический кластерный анализ

Выберем переменные, на основе которых мы будем кластеризовать регионы. Нам понадобятся все столбцы, которые заканчиваются на \_perc: явка в процентах и проценты голосов за кандидатов. Воспользуемся удобной функцией ends\_with из библиотеки dplyr.

library(dplyr)  
to\_clust <- df %>% select(ends\_with("\_perc"))

**Пояснения к коду:** логика использования оператора %>% такая: взять то, что слева, и подать этот объект на вход функции, которая стоит справа от оператора. В нашем случае мы берем базу df и подаем ее на вход функции select для выбора столбцов. Внутри скобок у select мы могли бы через запятую перечислить нужные нам столбцы, но сейчас мы поступим более хитро: выберем все столбцы, названия которых заканчиваются на \_perc. Выбранные столбцы сохраним в новую маленькую базу to\_clust.

Назовем строки по названиям региона (да, можно было бы вместо длинных названий создать числовой *id* регионов, но оставим названия для наглядности, чтобы не тратить время на "расшифровку" дендрограммы).

rownames(to\_clust) <- df$region

Теперь все готово к работе. Как всегда, стандартизируем данные (scale) и создадим матрицу расстояний m.

m <- dist(scale(to\_clust))

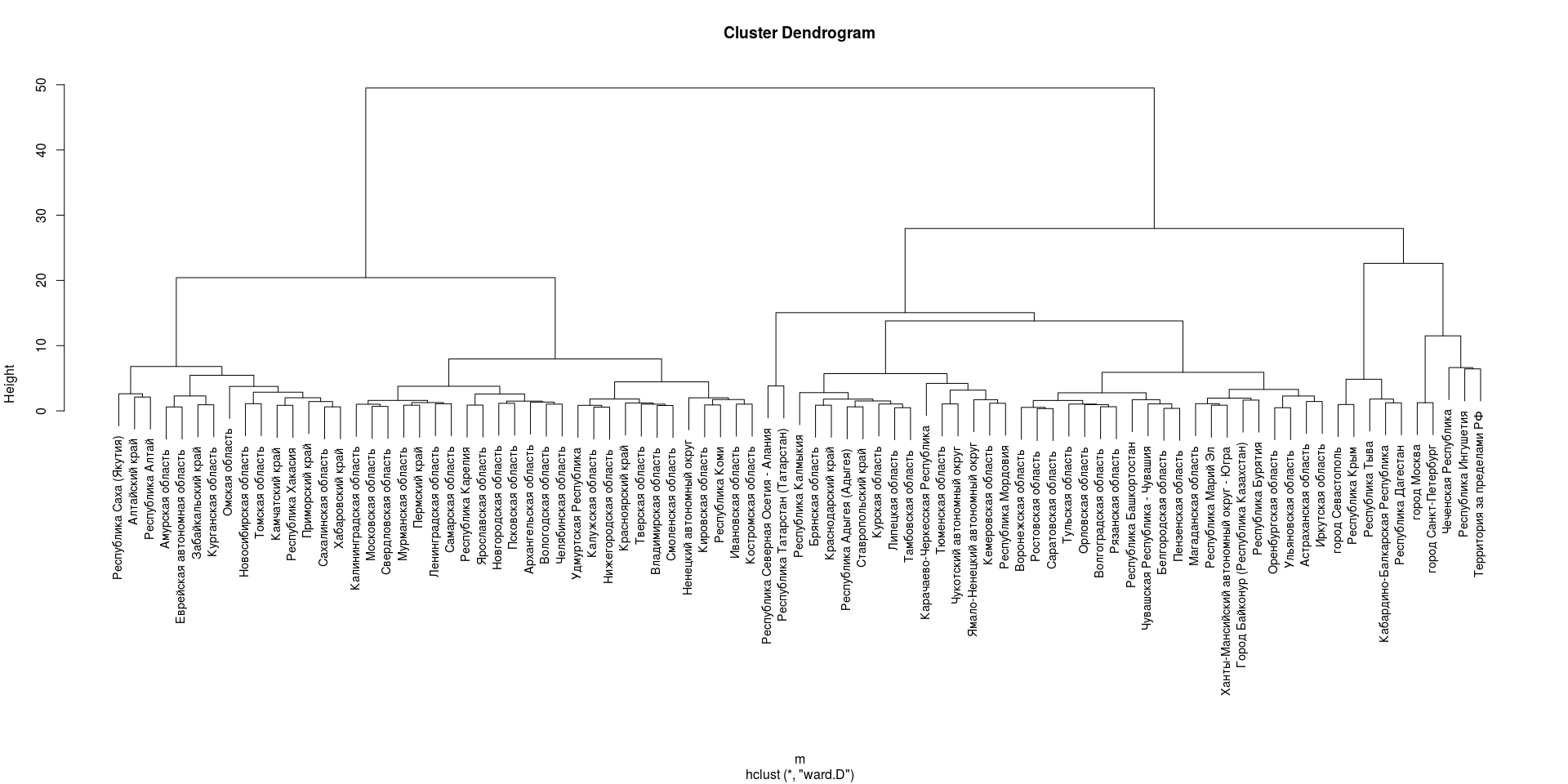
Проведем кластерный анализ, используя метод Варда в качестве метода агрегирования (да, по правилам нам нужна матрица с квадратами евклидова расстояния, но давайте пока оставим матрицу с обычным евклидовым расстоянием, чтобы можно было использовать ту же матрицу для другого метода агрегирования).

hc <- hclust(m, method = "ward.D")

**Примечание:** Вместо того, чтобы возводить значения матрицы расстояний в квадрат, можно взять исходную матрицу m и использовать метод ward.D2, который сам возводит значения расстояний из матрицы расстояний m в квадрат. Другими словами, код hclust(m^2, method = "ward.D")и код hclust(m, method = "ward.D2") должны дать одинаковые результаты.

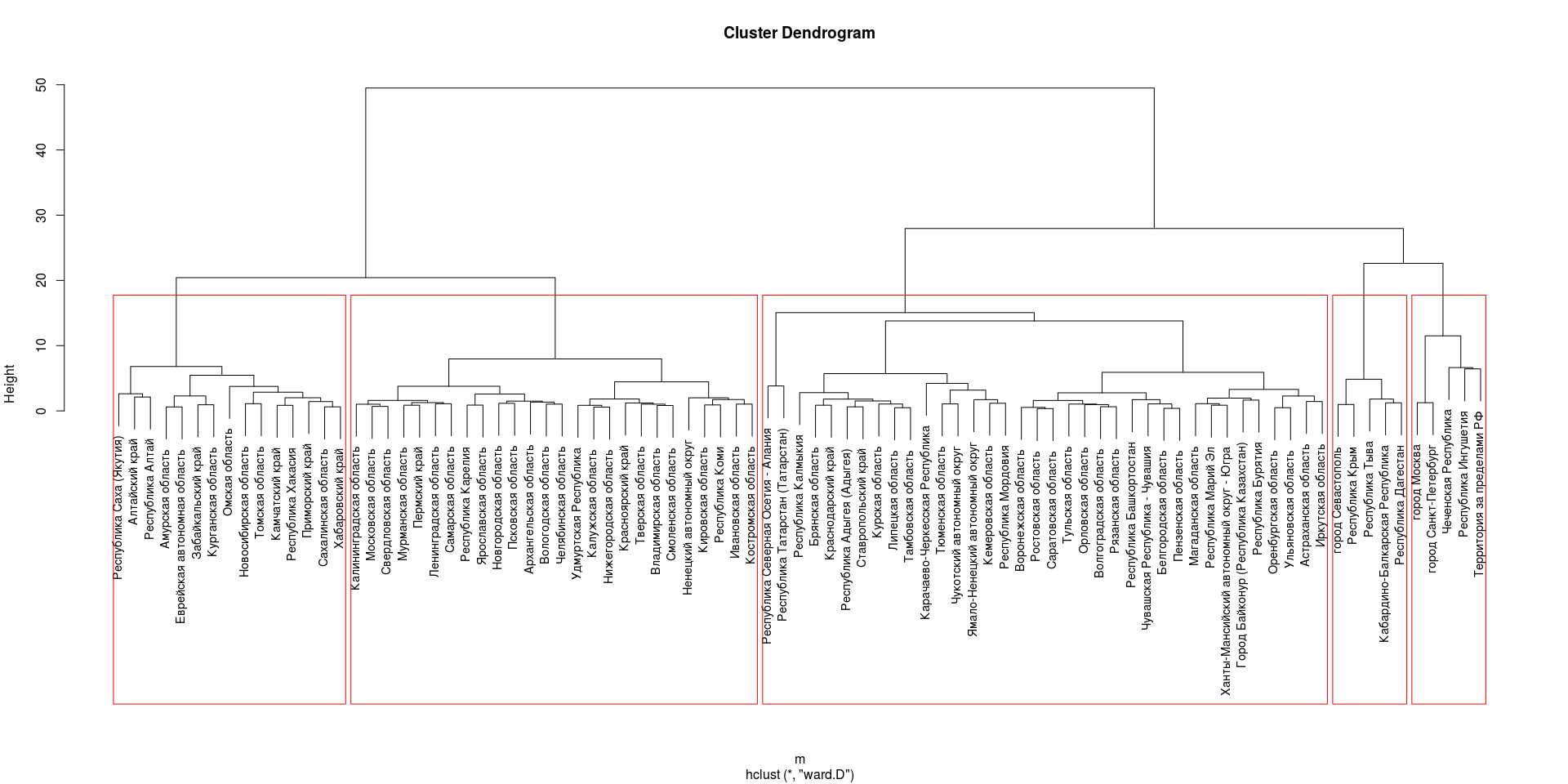
Посмотрим на дендрограмму:

plot(hc, cex = 0.9)



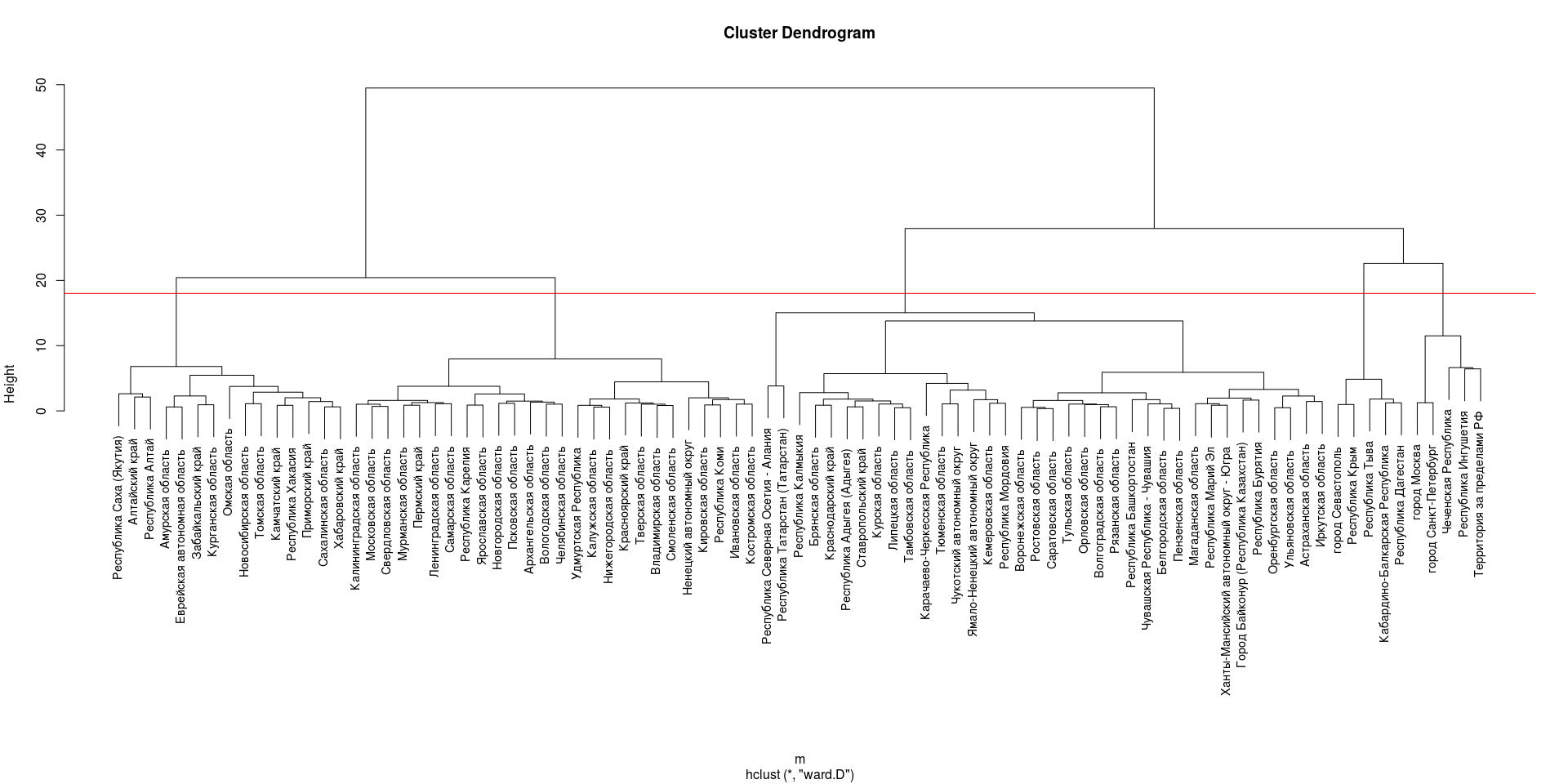
Давайте, чтобы не делать слишком общую классификацию, выделим 5 кластеров. Может быть, такой выбор окажется неидеальным, но зато будет интереснее оценивать качество кластеризации.

plot(hc, cex = 0.9)  
rect.hclust(hc, k = 5)



Если график просто появляется в окне *Plots*, то у него границы кластеров могут "уходить вверх" из-за того, что подписи наблюдений слишком длинные. Есть альтернатива -- прочертить линию на том уровне (на том расстоянии между кластерами), который нам нужен. По дендрограмме выше видно, что для того, чтобы получить пять кластеров, нужно "разрезать" дендрограмму на высоте примерно 18.

plot(hc, cex = 0.9)  
abline(h = 18, col = "red") # h - horizontal line, col - color



Вытащим из полученной кластеризации метки кластеров. Добавим их отдельным столбцом в базу данных, но перед этим сделаем эти метки факторными, то есть скажем R, что метки 1, 2, 3, 4, 5 -- это не числа, где 5 -- самое большое, а условные обозначения, коды (как в случае, когда мы кодируем рспондентов мужского пола цифрой 2, а женского -- цифрой 1).

groups5 <- cutree(hc, k = 5)  
df$groups5 <- factor(groups5)

### Оценка кластеризации: содержательно

Теперь посмотрим на каждый кластер в отдельности -- будем выбирать из базы строки, где значения groups5 равны то 1, то 2, и так далее, а потом оценивать содержательно, насколько разумными у нас получились кластеры. Воспользуемся функцией filter() из библиотеки dplyr.

df %>% filter(groups5 == 1) %>% View  
df %>% filter(groups5 == 2) %>% View  
df %>% filter(groups5 == 3) %>% View  
df %>% filter(groups5 == 4) %>% View  
df %>% filter(groups5 == 5) %>% View

В первом кластере оказались регионы, которые относительно близки географически (Дальний Восток, Сибирь) и в которых достаточно высокий процент за Грудинина (КПРФ). Второй кластер тоже получился достаточно "географичным": в основном, в нем находятся северные регионы и регионы Центральной России. В этих регионах наблюдается относительно высокий процент у Жириновского и у Собчак. В третьем кластере находятся южные регионы и регионы так называемого "красного пояса". В этих регионах высока поддержка Путина и Грудинина. В четвертом кластере находятся всего пять наблюдений: регионы с запредельно высокой явкой и процентом голосов за Путина (Москва, Санкт-Петербург, Чеченская республика, Республика Ингушетия) и территория за пределами РФ. В пятом кластере характеристики регионов похожи на характеристики элементов предыдущего кластера, но значения явки и процента за Путина пониже.

### Оценка кластеризации: визуально

Теперь давайте посмотрим на описательные статистики по группам (кластерам), точнее, на средние значения разных переменных по группам. Сначала сгруппируем данные по переменной groups5, то есть по кластерам. Потом запросим описательные статистики по конкретным переменным (общая функция для описательных статистик -- summarise, но мы используем summarise\_at, так как нас интересуют определенные столбцы). Перечисляем нужные переменные в .vars = vars(), а нужные статистики в .funs = funs(). Дальше хотим посмотреть на полученную таблицу в отдельном окне -- View.

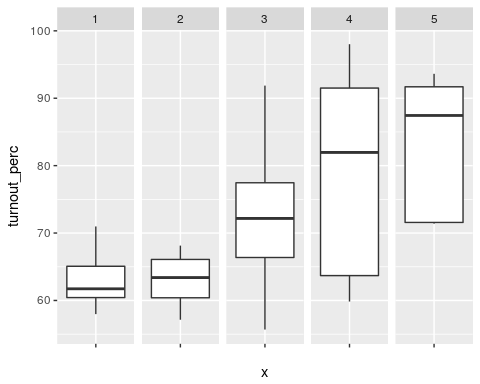
df %>% group\_by(groups5) %>%   
 summarise\_at(.vars = vars(ends\_with("\_perc")), .funs = funs(mean)) %>%   
 View

## Warning in View(.): X cannot set locale modifiers

Видно, что средние значения показателей по группам отличаются.

А теперь посмотрим на распределения разных переменных по кластерам и проверим, правда ли, что они отличаются. Начнем с явки. Построим "ящики с усами" -- для этого нам понадобится библиотека ggplot2.

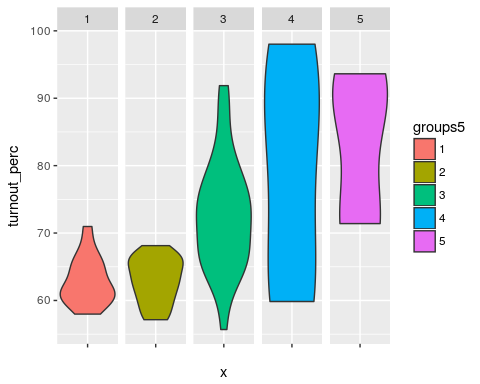
library(ggplot2)  
ggplot(data = df, aes(x = "", y = turnout\_perc)) + geom\_boxplot() + facet\_grid(~groups5)



**Пояснения к коду:** Графики, построенные с помощью ggplot2, создаются по слоям. Сначала в "главном" слое указывается база данных, из которой нужно выбирать переменные (data = df), затем в aes() указываются переменные, значения которых будут идти по осям x и y. Далее следует слой с типом графика (geom\_boxplot) и слой, который позволит нарисовать графики для каждой группы в отдельном окне-фасетке (facet\_grid).

Как можно заметить, распределения явки в разных кластерах совсем не похожи друг на друга: отличаются не только медианные значения, но и разброс значений. Однако стоит помнить, что "ящики с усами" наиболее информативны в случае, когда распределение данных нормальное или близко к нормальному. Посмотрим теперь на скрипичные диаграммы (*violin plots*) для явки (в aes() мы добавили fill = groups5, чтобы графики были разноцветными, в зависимости от кластера):

ggplot(data = df, aes(x = "", y = turnout\_perc, fill = groups5)) + geom\_violin() + facet\_grid(~groups5)

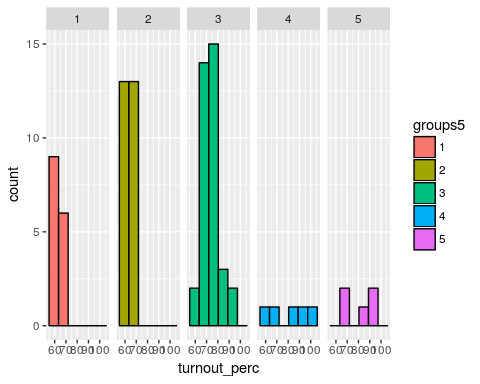


Картина стала еще более разнообразной. На скрипки, правда, графики особо не похожи, но зато они позволяют увидеть интересные особенности распределений. Так, в первом кластере распределение явки скошено вправо ("хвост" распределения справа), значения явки сконцентрированы на отрезке от 55% до 65%. Во втором кластере "хвостов" у распределения почти нет, значения явки сосредоточены на участке от 65% до 70%. В третьем кластере распределение явки очень растянутое, широкое, есть регионы с явкой 90% и, возможно, выше. В четвертом кластере распределение явки очень интересное, похоже на равномерное: доля регионов с явкой от 60% до 70% примерно такая же, как доля регионов с явкой от 70% до 80%. В пятом кластере находятся регионы с очень высокой явкой, более 85%, причем преобладают регионы с явкой 90%-95%.

**Примечание:** интерпретация -- это хорошо, но не забывайте про здравый смысл: в четвертом и пятом кластерах всего 5 регионов, поэтому все выражения "преобладают регионы" и "большинство регионов в кластере" очень условны, поскольку общее число регионов очень мало.

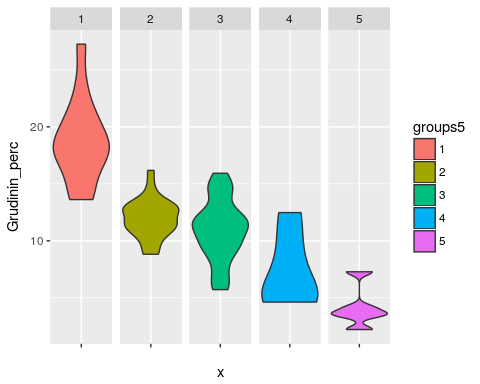
Для тех, кто любит менее экстравагантные графики: можно просто построить гистограммы:

# fill - цвет заливки  
# col - цвет границ графика, общий для всех кластеров  
# bins - число интервалов (столбцов) в гистограмме  
ggplot(data = df, aes(x = turnout\_perc , fill = groups5)) + geom\_histogram(bins = 6, col = "black") + facet\_grid(~groups5)



Конечно, для того, чтобы делать какие-то выводы, мало сравнить распределения явки, нужно посмотреть и на другие показатели. Рассматривать проценты за всех кандидатов мы сейчас не будем, можно подставить интересующие переменные в код для графиков, посмотрим только на Грудинина, кандидата от КПРФ, и Путина:

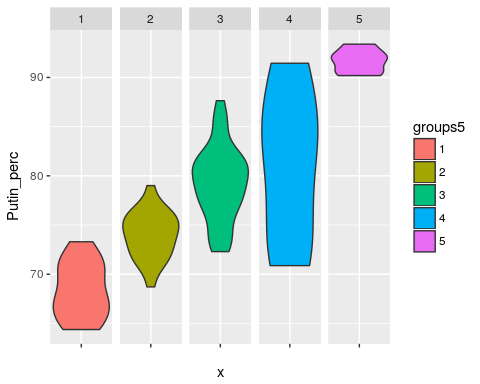
ggplot(data = df, aes(x = "", y = Grudinin\_perc, fill = groups5)) + geom\_violin() + facet\_grid(~groups5)



В первом кластере сосредоточены регионы с относительно высоким процентом за Грудинина. Во втором кластере процент за кандидата более низкий и "неровный", в третьем -- похожая ситуация, только разброс значений выше. В четвертом кластере находятся регионы, менее склонные голосовать за Грудинина, распределение скошено вправо, больше районов, где процент за этого кандидата около 5. В пятом кластере распределение очень специфическое. В целом, процент за Грудинина в этом регионе очень маленький, но при этом есть нетипичные регионы с процентом повыше.

И, наконец, графики с процентами голосов за Путина:

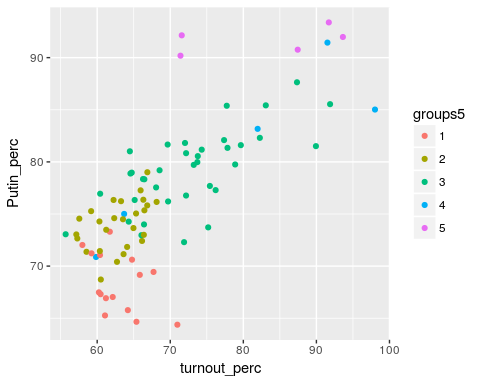
ggplot(data = df, aes(x = "", y = Putin\_perc, fill = groups5)) + geom\_violin() + facet\_grid(~groups5)



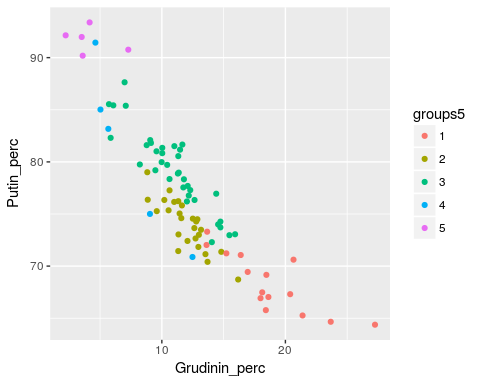
Картина получилась обратная тому, что мы видели на графиках для Грудинина, но похожая на то, что мы видели на графиках для явки. Можем посмотреть на распределение остальных показателей и получить "профиль" каждого кластера: например, кластер 1 содержит регионы с относительно низкой явкой и низким процентом за Путина, но высоким процентом за Грудинина, и так далее.

Теперь посмотрим на диаграммы рассеяния (это мы уже делали на самой первой лекции, только по показателям Всемирного банка).

ggplot(data = df, aes(x = turnout\_perc, y = Putin\_perc)) + geom\_point(aes(color = groups5))



ggplot(data = df, aes(x = Grudinin\_perc, y = Putin\_perc)) + geom\_point(aes(color = groups5))



В целом, точки сгруппированы по цветам, то есть, по кластерам.

### Оценка кластеризации: формально

Пока мы исследовали распределения только визуально. Давайте теперь применим какой-нибудь формальный критерий. Так как у нас пять групп, и понятно, что распределение разных показателей по группам не является нормальным, применим критерий Краскела-Уоллиса. В качестве примера возьмем явку и проценты за Грудинина и Путина.

kruskal.test(df$turnout\_perc ~ df$groups5)

##   
## Kruskal-Wallis rank sum test  
##   
## data: df$turnout\_perc by df$groups5  
## Kruskal-Wallis chi-squared = 38.763, df = 4, p-value = 7.797e-08

kruskal.test(df$Grudinin\_perc ~ df$groups5)

##   
## Kruskal-Wallis rank sum test  
##   
## data: df$Grudinin\_perc by df$groups5  
## Kruskal-Wallis chi-squared = 48.044, df = 4, p-value = 9.243e-10

kruskal.test(df$Putin\_perc ~ df$groups5)

##   
## Kruskal-Wallis rank sum test  
##   
## data: df$Putin\_perc by df$groups5  
## Kruskal-Wallis chi-squared = 57.537, df = 4, p-value = 9.543e-12

Нулевая гипотеза состоит в том, что данные в выборках взяты из одного распределения (медианы распределений равны). Так как p-value близко к 0, на уровне значимости 5% можно отвергнуть эту гипотезу: распределения явки/процентов за Грудинина/процентов за Путина отличаются по кластерам (медианы этих показателей по разным кластерам не равны).

### Оценка кластеризации: валидация

Мы говорили о том, что результаты кластерного анализа, полученные с использованием одного расстояния/метода агрегирования можно сравнивать с реализацией кластерного анализа, полученного с помощью другого расстояния/метода агрегирования. Сравним результаты нашей реализации (метод Варда) с результатами кластерного анализа методом средней связи, используя индекс Ранда ([Rand Index](https://en.wikipedia.org/wiki/Rand_index)).

# groups5 - наши метки кластеров  
# groups5\_2 - метки по результатам КА с методом средней связи  
hc2 <- hclust(m, method = "average")  
groups5\_2 <- cutree(hc2, k = 5)

Установим и загрузим библиотеку fossil, в которой хранится функция для расчета индекса Ранда.

install.packages("fossil")

Посчитаем сам индекс:

library(fossil)  
rand.index(groups5, groups5\_2)

## [1] 0.4028335

Значение 0.4 - достаточно невысокое. Но выводы о том, что кластеризация получилась не очень, делать еще рано. Все-таки у метода средней связи и метода Варда разные особенности.

Еще показатели согласованности результатов кластерного анализа можно получить с помощью библиотеки fpc. Установим ее:

install.packages("fpc")

Обратимся к библиотеке и запросим все возможные статистики:

library(fpc)  
# m - матрица расстояний  
cluster.stats(m, groups5, groups5\_2)

## $n  
## [1] 87  
##   
## $cluster.number  
## [1] 5  
##   
## $cluster.size  
## [1] 15 26 36 5 5  
##   
## $min.cluster.size  
## [1] 5  
##   
## $noisen  
## [1] 0  
##   
## $diameter  
## [1] 4.734900 3.178691 7.976310 9.653249 3.049545  
##   
## $average.distance  
## [1] 2.391968 1.732994 2.543869 6.460859 2.222078  
##   
## $median.distance  
## [1] 2.231326 1.696577 2.094836 6.591171 2.516464  
##   
## $separation  
## [1] 0.9669200 0.8650578 0.8650578 4.3618021 2.3563008  
##   
## $average.toother  
## [1] 3.968354 3.806401 3.978244 6.944599 5.987076  
##   
## $separation.matrix  
## [,1] [,2] [,3] [,4] [,5]  
## [1,] 0.000000 1.0860469 0.9669200 5.517726 5.213241  
## [2,] 1.086047 0.0000000 0.8650578 4.361802 4.683634  
## [3,] 0.966920 0.8650578 0.0000000 4.447778 2.356301  
## [4,] 5.517726 4.3618021 4.4477778 0.000000 4.928625  
## [5,] 5.213241 4.6836344 2.3563008 4.928625 0.000000  
##   
## $ave.between.matrix  
## [,1] [,2] [,3] [,4] [,5]  
## [1,] 0.000000 3.023418 3.718794 7.625167 7.022042  
## [2,] 3.023418 0.000000 3.392199 6.217148 6.726854  
## [3,] 3.718794 3.392199 0.000000 7.040057 4.742212  
## [4,] 7.625167 6.217148 7.040057 0.000000 7.998345  
## [5,] 7.022042 6.726854 4.742212 7.998345 0.000000  
##   
## $average.between  
## [1] 4.308308  
##   
## $average.within  
## [1] 2.318376  
##   
## $n.between  
## [1] 2661  
##   
## $n.within  
## [1] 1080  
##   
## $max.diameter  
## [1] 9.653249  
##   
## $min.separation  
## [1] 0.8650578  
##   
## $within.cluster.ss  
## [1] 344.7903  
##   
## $clus.avg.silwidths  
## 1 2 3 4 5   
## 0.17065402 0.39548843 0.16263808 -0.08154182 0.52762879   
##   
## $avg.silwidth  
## [1] 0.2405507  
##   
## $g2  
## NULL  
##   
## $g3  
## NULL  
##   
## $pearsongamma  
## [1] 0.4476113  
##   
## $dunn  
## [1] 0.08961313  
##   
## $dunn2  
## [1] 0.4679592  
##   
## $entropy  
## [1] 1.357491  
##   
## $wb.ratio  
## [1] 0.5381175  
##   
## $ch  
## [1] 25.51928  
##   
## $cwidegap  
## [1] 2.298542 1.414098 3.830797 6.511035 2.504136  
##   
## $widestgap  
## [1] 6.511035  
##   
## $sindex  
## [1] 0.974445  
##   
## $corrected.rand  
## [1] 0.07577165  
##   
## $vi  
## [1] 1.261643

Функция cluster.stats() выдает целый ряд разных показателей, почитать про них можно в документации, вызвав *help* через ?cluster.stats().

### Оценка кластеризации: "нужные" кластеры vs случайный шум

Проверим, правда ли кластеры, которые мы видели на дендрограмме, получились обосновано, исходя из данных, а не случайно, из-за нескольких выбросов или какого-то шума в данных. Для этого нам потребуется [библиотека](http://stat.sys.i.kyoto-u.ac.jp/prog/pvclust/) pvclust (*pv - от p-value*).

install.packages("pvclust")

Функция pvclust реализует кластерный анализ по столбцам, а не по строкам таблицы, поэтому таблицу сначала нужно транспонировать (t перед to\_clust).

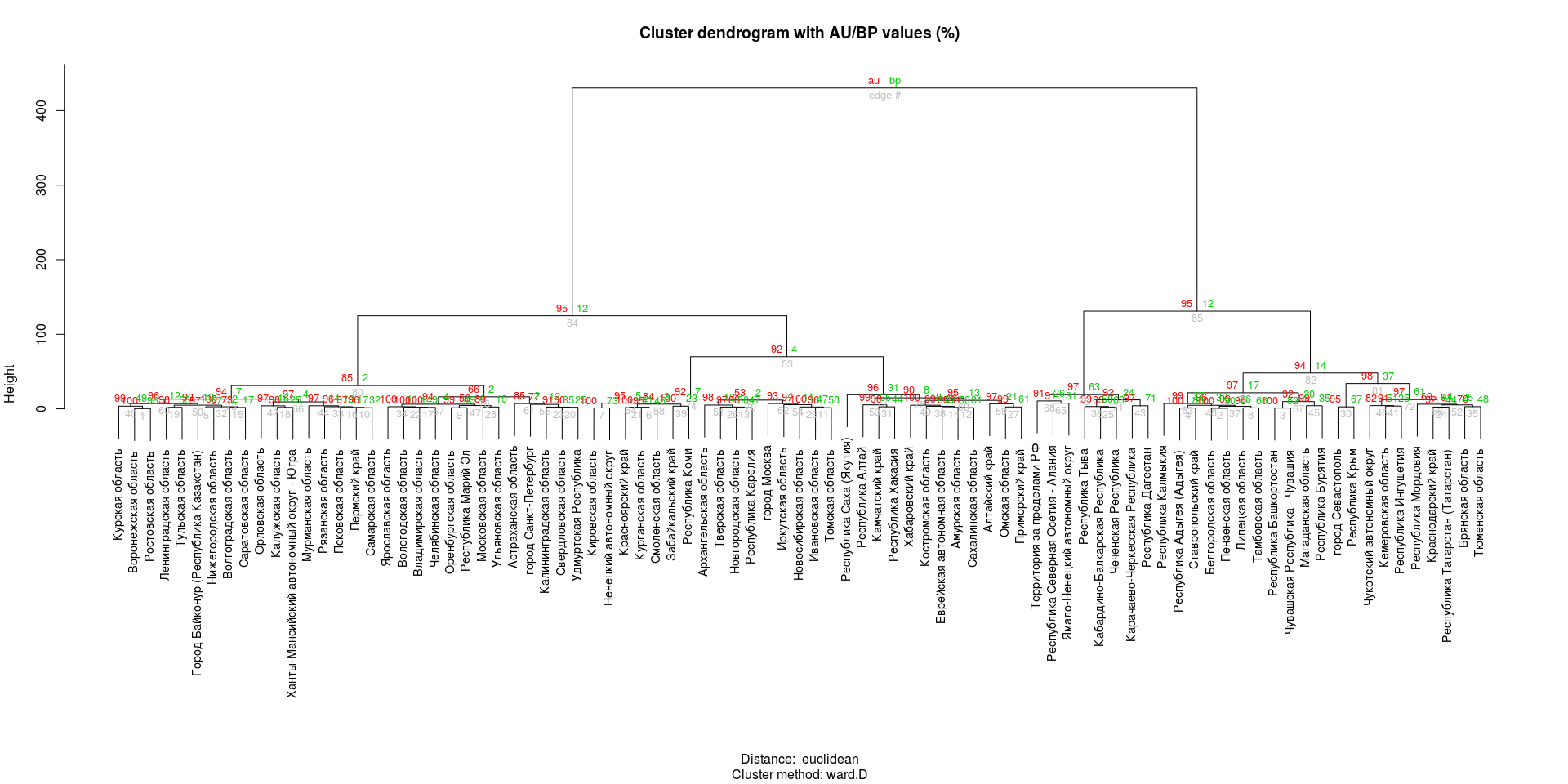
**Внимание:** исполнение следующего кода займет несколько минут, так как будет запускаться процедура бутстраппирования (*bootstrap*), которая заключается в повторяющемся "выдергивании" подвыборок из исходной выборки и оценивании параметров получившегося распределения.

library(pvclust)  
fit <- pvclust(t(to\_clust), method.hclust = "ward", method.dist = "euclidean")

## Bootstrap (r = 0.44)... Done.  
## Bootstrap (r = 0.56)... Done.  
## Bootstrap (r = 0.67)... Done.  
## Bootstrap (r = 0.78)... Done.  
## Bootstrap (r = 0.89)... Done.  
## Bootstrap (r = 1.0)... Done.  
## Bootstrap (r = 1.0)... Done.  
## Bootstrap (r = 1.11)... Done.  
## Bootstrap (r = 1.22)... Done.  
## Bootstrap (r = 1.33)... Done.

А теперь посмотрим на график.

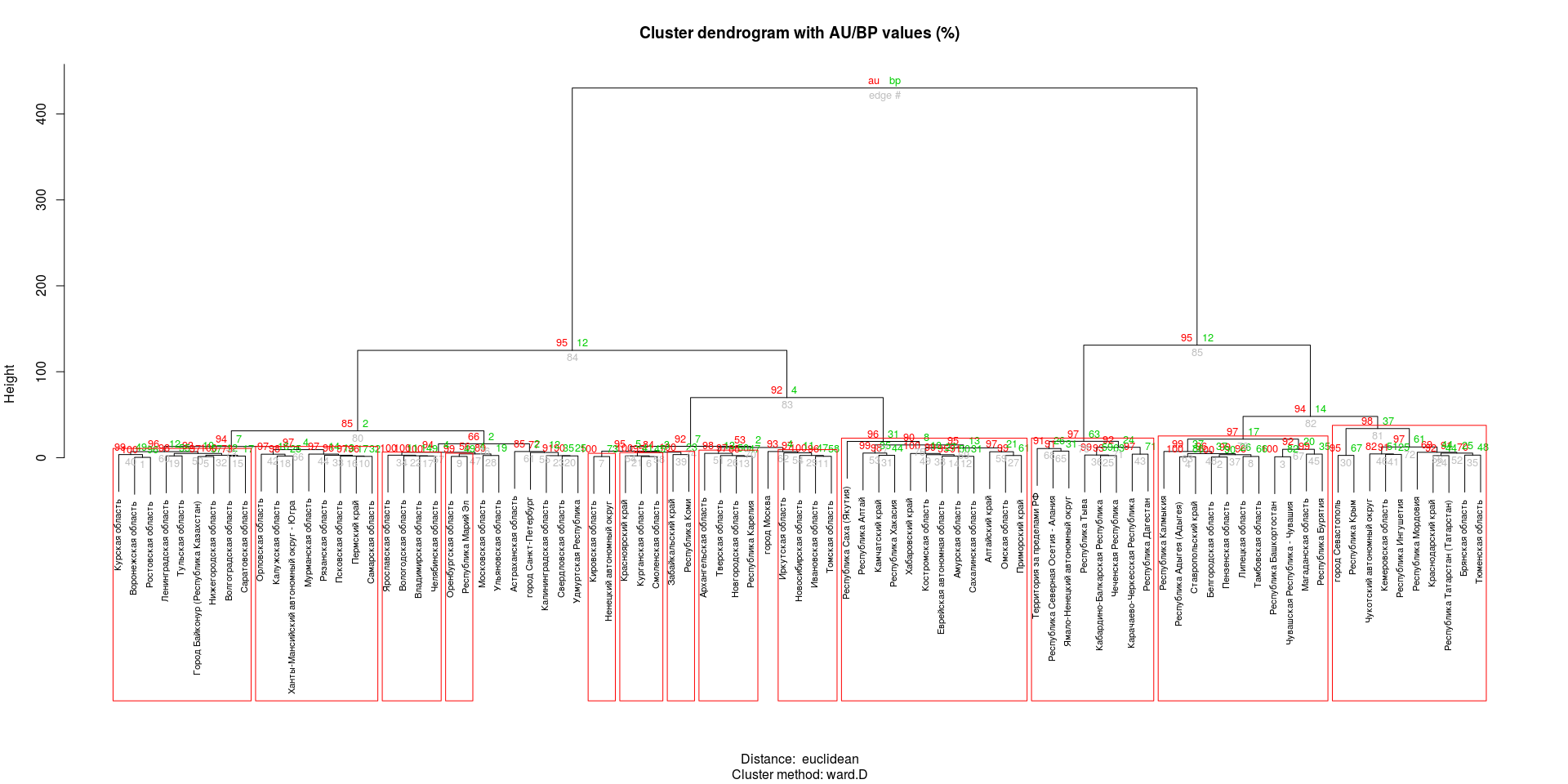
plot(fit, cex = 0.9)



На дендрограмме над каждым кластером (над каждой "веткой" дендрограммы) подписано p-value -- вероятность того, что кластер образован не случайно (например, из-за наличия в данных каких-то странных значений или шума), а обоснованно, на основе имеющихся показателей (*highly supported by data*). Значения подписаны двумя цветами. Зеленое число -- это почти несмещенное p-value, посчитанное на основе выборок, полученных в результате многошкального бутстраппирования (*multiscale bootstrap resampling*, *AU* -- от *Approximately Unbiased*), красное число -- то же p-value, но посчитанное по итогам обычного, нормального бутстраппирования (*normal bootstrap resampling*, *BP* -- от *Bootstrap Probability*).

Кроме того, можно выбрать желаемую вероятность -- при которой гипотеза о том, что кластер образован не случайно, не отвергается, и выделить такие кластеры на дендрограмме.

plot(fit, cex = 0.7)  
pvrect(fit, alpha = 0.95)



Если брать вероятность, равную 95%, то видно, сколько кластеров не случайны. Полученная дендрограмма, кстати, отличается от того, что мы видели в начале. Давайте думать дальше.

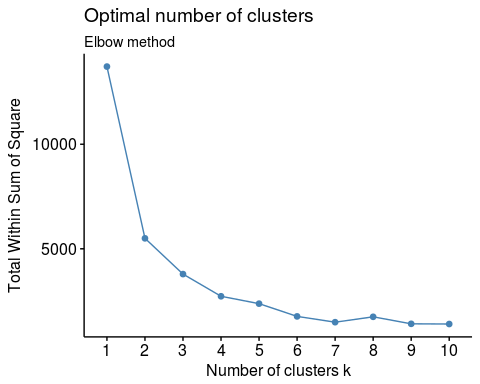
### Сколько кластеров выбрать?

Конечно, главное правило по выбору числа кластеров: выбрать столько, сколько можно интерпретировать содержательно. Но посмотрим на статистические методы. Для этого нам потребуется библиотека factoextra. Установим его:

install.packages("factoextra")

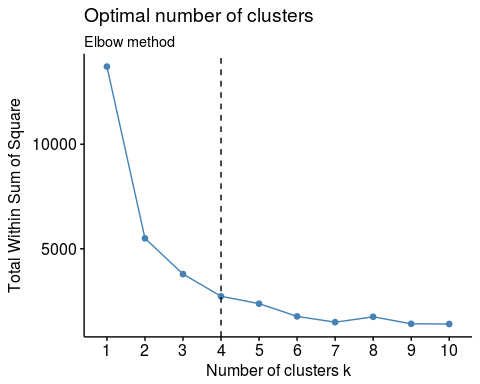
**Elbow method** ("метод согнутого колена", он же "метод каменистой осыпи"). Построим график, где по оси абсцисс отмечено число кластеров , а по оси ординат -- значения функции , которая определяет внутригрупповой разброс в зависимости от числа кластеров.

library(factoextra)  
fviz\_nbclust(to\_clust, kmeans, method = "wss") +  
 labs(subtitle = "Elbow method")



По графику видно, что "колено сгибается" при . Наверное, четыре кластера, лучше, чем пять. Но вообще, резкий изгиб наблюдается уже при - вопрос в том, достаточно ли нам только двух кластеров, или мы все же хотим более детальное деление на группы.

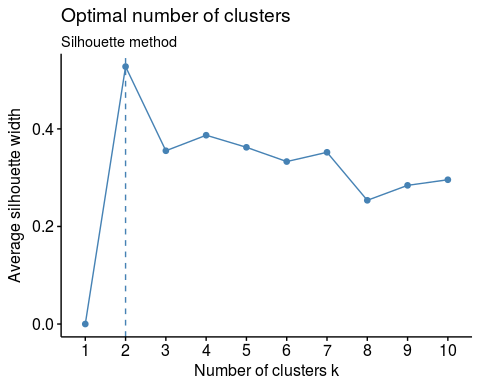
# geom\_vline - добавить вертикальную линию  
fviz\_nbclust(to\_clust, kmeans, method = "wss") +  
 labs(subtitle = "Elbow method") +  
 geom\_vline(xintercept = 4, linetype = 2)



Попробуем другой метод.

**Silhouette method** ("силуэтный метод").

fviz\_nbclust(to\_clust, kmeans, method = "silhouette") +  
 labs(subtitle = "Silhouette method")



И опять мы получаем только два кластера. Но тем не менее, кластерный анализ очень сильно зависит от исследователя :)

**Примечание:** что такое "силуэт" и как он считается, см. [здесь](https://en.wikipedia.org/wiki/Silhouette_(clustering)).

### K-means

Вот мы и подошли вплотную к методу *k-средних*. На предыдущем шаге неявно этот метод кластеризации уже был задействован: и Elbow method, и Silhouette method основаны на методе k-средних с разным k. Давайте попробуем реализовать кластерный анализ методом k-средних, взяв .

cl <- kmeans(to\_clust, 4)

**Обратите внимание:** в функции kmeans() мы указываем саму базу данных, матрица расстояний m нам уже не нужна.

Посмотрим, что внутри cl:

cl

## K-means clustering with 4 clusters of sizes 21, 13, 9, 44  
##   
## Cluster means:  
## turnout\_perc Baburin\_perc Grudinin\_perc Zhirinovsky\_perc Putin\_perc  
## 1 75.00119 0.5215231 9.301290 4.797676 81.35068  
## 2 63.44951 0.6637527 19.575591 7.561974 67.52520  
## 3 90.50254 0.3904947 6.044416 2.390205 88.07695  
## 4 63.48579 0.6928433 12.256333 6.973875 74.70153  
## Sobchak\_perc Suraikin\_perc Titov\_perc Yavlinsky\_perc  
## 1 1.2107819 0.6148411 0.5795316 0.6676709  
## 2 1.3950757 0.5978785 0.6270841 0.7281903  
## 3 0.9312035 0.5803068 0.4734051 0.5050261  
## 4 1.7456653 0.6787880 0.7789469 1.0482856  
##   
## Clustering vector:  
## Алтайский край   
## 2   
## Амурская область   
## 2   
## Архангельская область   
## 4   
## Астраханская область   
## 4   
## Белгородская область   
## 1   
## Брянская область   
## 1   
## Владимирская область   
## 4   
## Волгоградская область   
## 4   
## Вологодская область   
## 4   
## Воронежская область   
## 4   
## Город Байконур (Республика Казахстан)   
## 4   
## город Москва   
## 4   
## город Санкт-Петербург   
## 4   
## город Севастополь   
## 1   
## Еврейская автономная область   
## 2   
## Забайкальский край   
## 4   
## Ивановская область   
## 4   
## Иркутская область   
## 4   
## Кабардино-Балкарская Республика   
## 3   
## Калининградская область   
## 4   
## Калужская область   
## 4   
## Камчатский край   
## 2   
## Карачаево-Черкесская Республика   
## 3   
## Кемеровская область   
## 3   
## Кировская область   
## 4   
## Костромская область   
## 2   
## Краснодарский край   
## 1   
## Красноярский край   
## 4   
## Курганская область   
## 4   
## Курская область   
## 4   
## Ленинградская область   
## 4   
## Липецкая область   
## 1   
## Магаданская область   
## 4   
## Московская область   
## 4   
## Мурманская область   
## 4   
## Ненецкий автономный округ   
## 4   
## Нижегородская область   
## 4   
## Новгородская область   
## 4   
## Новосибирская область   
## 2   
## Омская область   
## 2   
## Оренбургская область   
## 4   
## Орловская область   
## 1   
## Пензенская область   
## 1   
## Пермский край   
## 4   
## Приморский край   
## 2   
## Псковская область   
## 4   
## Республика Адыгея (Адыгея)   
## 1   
## Республика Алтай   
## 2   
## Республика Башкортостан   
## 1   
## Республика Бурятия   
## 1   
## Республика Дагестан   
## 3   
## Республика Ингушетия   
## 1   
## Республика Калмыкия   
## 1   
## Республика Карелия   
## 4   
## Республика Коми   
## 4   
## Республика Крым   
## 1   
## Республика Марий Эл   
## 4   
## Республика Мордовия   
## 1   
## Республика Саха (Якутия)   
## 2   
## Республика Северная Осетия - Алания   
## 3   
## Республика Татарстан (Татарстан)   
## 1   
## Республика Тыва   
## 3   
## Республика Хакасия   
## 2   
## Ростовская область   
## 4   
## Рязанская область   
## 4   
## Самарская область   
## 4   
## Саратовская область   
## 4   
## Сахалинская область   
## 2   
## Свердловская область   
## 4   
## Смоленская область   
## 4   
## Ставропольский край   
## 1   
## Тамбовская область   
## 1   
## Тверская область   
## 4   
## Территория за пределами РФ   
## 3   
## Томская область   
## 4   
## Тульская область   
## 1   
## Тюменская область   
## 1   
## Удмуртская Республика   
## 4   
## Ульяновская область   
## 4   
## Хабаровский край   
## 2   
## Ханты-Мансийский автономный округ - Югра   
## 4   
## Челябинская область   
## 4   
## Чеченская Республика   
## 3   
## Чувашская Республика - Чувашия   
## 1   
## Чукотский автономный округ   
## 1   
## Ямало-Ненецкий автономный округ   
## 3   
## Ярославская область   
## 4   
##   
## Within cluster sum of squares by cluster:  
## [1] 870.6931 351.0564 360.8594 1131.2831  
## (between\_SS / total\_SS = 80.2 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss"   
## [5] "tot.withinss" "betweenss" "size" "iter"   
## [9] "ifault"

Объект cl - это один большой список (*list*), к элементам которого можно обратиться, используя $. Например, запросим метки кластеров (cluster) и сохраним их отдельным столбцом в исходную базу данных df:

df$kmeans4 <- cl$cluster  
View(df)

Что можно сделать напоследок? Можно реализовать кластерный анализ, используя метод Варда, но выбрав 4 кластера, а потом сравнить результаты с k-means при .

# hc - результат кластеризации, который мы получили в самом начале  
# rand.index - из библиотеки fossil  
groups4 <- cutree(hc, k = 4)  
rand.index(groups4, cl$cluster)

## [1] 0.6482224

Согласованность результатов уже получше!

А что будет, если поверить графикам, и взять два кластера, как рекомедовали Месье Silhouette и Мистер Согнутое Колено (только не пишите так в домашних заданиях, пожалуйста!)?

groups2 <- cutree(hc, k = 2)  
cl2 <- kmeans(to\_clust, 2)  
rand.index(groups2, cl2$cluster)

## [1] 0.641807

Серьезных изменений в согласованности не произошло. Так что, действовать нужно по велению исследовательского сердца:)

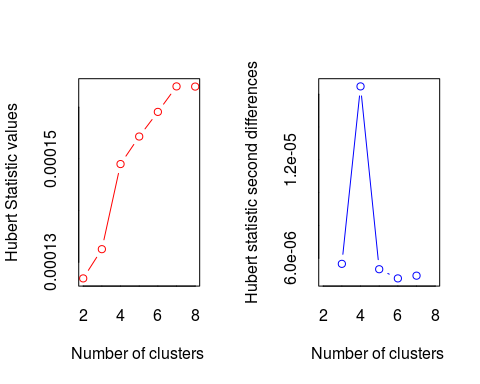
### Дополнение: NbClust

А для тех, кто еще не устал от кластерного анализа, можно попробовать процедуру Nbclust, которая включает 30 разных методов нахождения оптимального числа кластеров, и выдает результат, за который проголосовали большинство методов (*majority rule*).

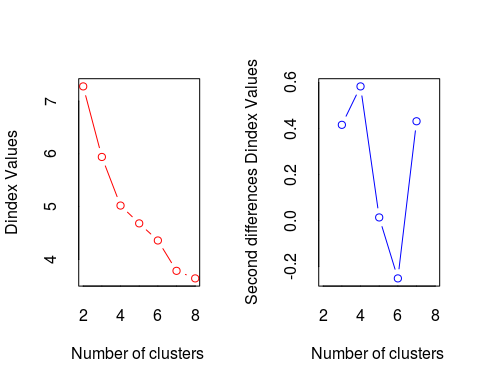
**Внимание:** не увлекайтесь, процедура достаточно специфичная и не всегда дает стабильные результаты! Исполнение кода займет некоторое время.

# поставить библиотеку  
install.packages("NbClust")

library(NbClust)  
res <- NbClust(to\_clust, min.nc = 2, max.nc = 8, method = "kmeans")



## \*\*\* : The Hubert index is a graphical method of determining the number of clusters.  
## In the plot of Hubert index, we seek a significant knee that corresponds to a   
## significant increase of the value of the measure i.e the significant peak in Hubert  
## index second differences plot.   
##



## \*\*\* : The D index is a graphical method of determining the number of clusters.   
## In the plot of D index, we seek a significant knee (the significant peak in Dindex  
## second differences plot) that corresponds to a significant increase of the value of  
## the measure.   
##   
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
## \* Among all indices:   
## \* 12 proposed 2 as the best number of clusters   
## \* 4 proposed 3 as the best number of clusters   
## \* 1 proposed 4 as the best number of clusters   
## \* 1 proposed 5 as the best number of clusters   
## \* 4 proposed 7 as the best number of clusters   
## \* 2 proposed 8 as the best number of clusters   
##   
## \*\*\*\*\* Conclusion \*\*\*\*\*   
##   
## \* According to the majority rule, the best number of clusters is 2   
##   
##   
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Все результаты (все 30 методов):

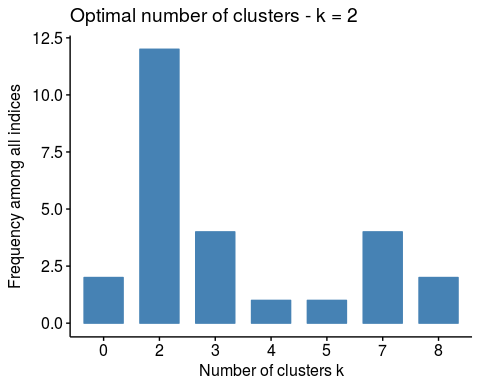
res$Best.nc

## KL CH Hartigan CCC Scott Marriot  
## Number\_clusters 7.0000 2.0000 4.0000 2.0000 3.0000 7.000000e+00  
## Value\_Index 4.5443 121.2382 16.7016 34.6366 97.5305 1.442519e+12  
## TrCovW TraceW Friedman Rubin Cindex DB  
## Number\_clusters 3 3.0000 8.00 7.0000 5.000 2.0000  
## Value\_Index 1120950 998.3253 41197.76 -81.5805 0.269 0.7328  
## Silhouette Duda PseudoT2 Beale Ratkowsky Ball  
## Number\_clusters 2.0000 2.0000 2.0000 2.0000 2.0000 3.000  
## Value\_Index 0.5483 1.1886 -6.6629 -0.8997 0.3331 1596.233  
## PtBiserial Frey McClain Dunn Hubert SDindex Dindex  
## Number\_clusters 2.0000 2.0000 2.0000 7.0000 0 2.000 0  
## Value\_Index 0.7165 3.0645 0.2131 0.1343 0 0.285 0  
## SDbw  
## Number\_clusters 8.0000  
## Value\_Index 0.0808

Число кластеров, выбраное большинством методов:

library(factoextra)  
fviz\_nbclust(res)

## Among all indices:   
## ===================  
## \* 2 proposed 0 as the best number of clusters  
## \* 12 proposed 2 as the best number of clusters  
## \* 4 proposed 3 as the best number of clusters  
## \* 1 proposed 4 as the best number of clusters  
## \* 1 proposed 5 as the best number of clusters  
## \* 4 proposed 7 as the best number of clusters  
## \* 2 proposed 8 as the best number of clusters  
##   
## Conclusion  
## =========================  
## \* According to the majority rule, the best number of clusters is 2 .



Все-таки 2! Или 3 в крайнем случае. Up to you. Решайте из содержательных соображений.