Лабораторная по многомерной статистике

Дмитрий Пасько

Table of Contents

## Описание

Для выполнения работы использовался язык **R** версии 3.6.1. Необходимые пакеты скачиваются командой

install.packages(c("readxl", "dplyr", "ggplot2", "ggpubr", "corrplot", "psych",   
 "MASS", "tree", "randomForest", "TeachingDemos", "mice", "corrgram", "magrittr",   
 "plotly", "factoextra"))

и подключаются

library(readxl) #чтение из excel  
library(dplyr) #современные средства программирования, включая функциональное  
library(ggplot2) #красивые удобные графики  
library(ggpubr) #группировка изображений  
library(corrplot) #картинка для корреляционной матрицы  
library(psych) #факторный анализ  
library(MASS) #линейный дискриминантный анализ  
library(tree) #визуализация деревьев  
library(randomForest) #случайные леса  
library(TeachingDemos) #пиктограммы  
library(mice) #обработка отсутствующих значений  
library(corrgram) #коррелограммы  
library(magrittr) #конвеерный оператор  
library(plotly) #интерактивная графика  
library(factoextra) #графика по главным компонентам

Также использовался номер варианта

nv = 7 #номер варианта

Дополнительная информация: [сайт по статистике в R](http://www.sthda.com/english/), [огромная статья по построению моделей в R](https://ranalytics.github.io/data-mining/01-Data-Mining-Models-in-R.html), [cтатья по кластерному, факторному и дискриминантному анализу в R](https://rpubs.com/iezepov/e502lec7), [курс по анализу данных в R](https://stepik.org/course/129/syllabus), [курс по основам R как языка программирования](https://stepik.org/course/497/syllabus), [курс по статистике в R](https://stepik.org/course/524/syllabus), [перечень основых функций языка](https://aakinshin.net/ru/posts/r-functions/), [математические операции в R](https://ru.wikibooks.org/wiki/Язык_программирования_R/Математика), [книги по R на моём GitHub](https://github.com/PasaOpasen/LittleHelps/tree/master/книги%20по%20языкам/R), [набор ссылок](https://ru.stackoverflow.com/questions/506597/Книги-и-учебные-ресурсы-по-языку-r), [отличный сайт по R](https://www.statmethods.net).

## Многомерный анализ

### Задание 1

Подготовим данные:

datacrude = data.frame(read\_excel("Таблица 1.xlsx")) #считывание таблицы  
data = datacrude[5:nrow(datacrude), -1] #удаление лишних строк и столбцов  
data = data[-nv, ] #удаление строки в соответствиии с номером варианта  
colnames(data) = c("Country", "Doctors", "Deaths", "GDP", "Costs") #переименование столбцов (для лаконичности)  
data[, 1] = factor(data[, 1]) #первая переменная из количественной преобразуется в номенативную  
row.names(data) = as.character(1:nrow(data)) #наблюдения нумеруются  
  
data %>% tbl\_df() #таблица старого образца переводится в более удобный и выводится

## # A tibble: 19 x 5  
## Country Doctors Deaths GDP Costs   
## <fct> <chr> <chr> <chr> <chr>   
## 1 Россия 44.5 84.98 20.399999999~ 3.2   
## 2 Австралия 32.5 30.58 71.400000000~ 8.5   
## 3 Австрия 33.9 38.42 78.7 9.1999999999~  
## 4 Азербайджан 38.79999999999~ 60.34 12.1 3.3   
## 5 Армения 34.4 60.22 10.9 3.2   
## 6 Беларусь 43.6 60.79 20.399999999~ 5.4   
## 7 Болгария 36.4 70.56999999999~ 17.3 5.4   
## 8 Великобрита~ 17.89999999999~ 34.51 69.7 7.1   
## 9 Венгрия 32.1 64.73 24.5 6   
## 10 Германия 38.1 36.63000000000~ 76.2 8.6   
## 11 Греция 41.5 32.84000000000~ 44.4 5.7   
## 12 Грузия 55 62.64 11.3 3.5   
## 13 Дания 36.70000000000~ 34.07 79.2 6.7   
## 14 Ирландия 15.8 39.27000000000~ 57 6.7   
## 15 Испания 40.9 28.46 54.8 7.3   
## 16 Италия 49.4 30.27 72.099999999~ 8.5   
## 17 Казахстан 38.1 69.04000000000~ 13.4 3.3   
## 18 Канада 27.6 25.42 79.900000000~ 10.199999999~  
## 19 Киргизия 33.20000000000~ 53.13 11.2 3.4

data[, 2:5] = apply(data[, 2:5], 2, function(x) scale(as.numeric(x))) #тут переменные из текста преобразуются в числа и стандартизируются

Полученная таблица (**стандартизованные данные**):

data %>% tbl\_df()

## # A tibble: 19 x 5  
## Country Doctors Deaths GDP Costs  
## <fct> <dbl> <dbl> <dbl> <dbl>  
## 1 Россия 0.869 2.06 -0.807 -1.25   
## 2 Австралия -0.408 -0.992 0.981 1.06   
## 3 Австрия -0.259 -0.552 1.24 1.37   
## 4 Азербайджан 0.262 0.678 -1.10 -1.20   
## 5 Армения -0.206 0.671 -1.14 -1.25   
## 6 Беларусь 0.773 0.703 -0.807 -0.289   
## 7 Болгария 0.00672 1.25 -0.915 -0.289   
## 8 Великобритания -1.96 -0.771 0.921 0.451   
## 9 Венгрия -0.451 0.924 -0.663 -0.0275  
## 10 Германия 0.188 -0.653 1.15 1.10   
## 11 Греция 0.550 -0.865 0.0345 -0.158   
## 12 Грузия 1.99 0.807 -1.13 -1.12   
## 13 Дания 0.0387 -0.796 1.25 0.277   
## 14 Ирландия -2.19 -0.504 0.476 0.277   
## 15 Испания 0.486 -1.11 0.399 0.538   
## 16 Италия 1.39 -1.01 1.01 1.06   
## 17 Казахстан 0.188 1.17 -1.05 -1.20   
## 18 Канада -0.930 -1.28 1.28 1.80   
## 19 Киргизия -0.334 0.273 -1.13 -1.16

**Средние и стандартные отклонения**:

means = apply(data[, -1], 1, mean)  
sds = apply(data[, -1], 1, sd)  
data\_frame(countries = data[, 1], means, sds)

## # A tibble: 19 x 3  
## countries means sds  
## <fct> <dbl> <dbl>  
## 1 Россия 0.219 1.53   
## 2 Австралия 0.160 1.02   
## 3 Австрия 0.448 0.994  
## 4 Азербайджан -0.340 0.952  
## 5 Армения -0.480 0.899  
## 6 Беларусь 0.0952 0.772  
## 7 Болгария 0.0137 0.911  
## 8 Великобритания -0.340 1.30   
## 9 Венгрия -0.0543 0.704  
## 10 Германия 0.447 0.857  
## 11 Греция -0.110 0.586  
## 12 Грузия 0.138 1.53   
## 13 Дания 0.193 0.844  
## 14 Ирландия -0.484 1.21   
## 15 Испания 0.0780 0.795  
## 16 Италия 0.612 1.09   
## 17 Казахстан -0.225 1.12   
## 18 Канада 0.217 1.55   
## 19 Киргизия -0.587 0.689

Для решения задачи создается **матрица (евклидовых) расстояний**

(d = dist(data[, 2:5], method = "euclidean")) #матрица расстояний

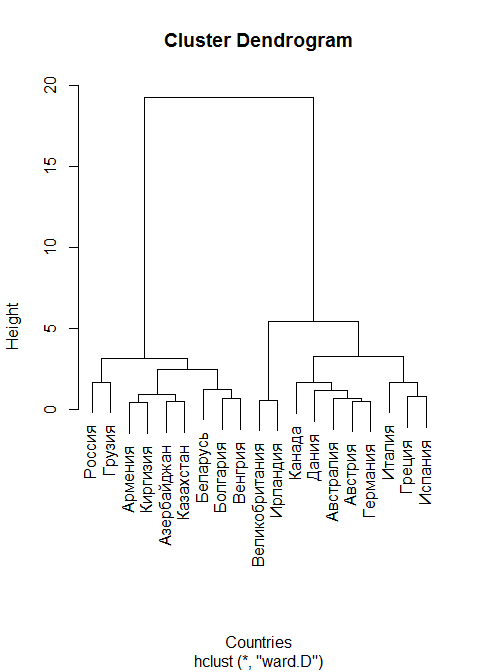
## 1 2 3 4 5 6 7  
## 2 4.4120797   
## 3 4.3697012 0.6115859   
## 4 1.5382991 3.5610208 3.7186612   
## 5 1.7880599 3.5531815 3.7371442 0.4723246   
## 6 1.6638732 3.0472614 3.0908033 1.0870717 1.4099074   
## 7 1.5250963 3.2594515 3.2699370 1.1240009 1.1617940 0.9488992   
## 8 4.6803367 1.6849098 1.9707704 3.7231844 3.5064730 3.6321676 3.4486947  
## 9 2.1305565 2.7495920 2.7866134 1.4625949 1.3551118 1.2791960 0.6699219  
## 10 4.1441528 0.7076818 0.5346512 3.4849263 3.5594776 2.8188389 3.1403530  
## 11 3.2486216 1.8206319 2.1256680 2.1992479 2.3440153 1.7984619 2.3867169  
## 12 1.7145100 4.2598610 4.3159864 1.7317365 2.2010009 1.5063236 2.2014486  
## 13 3.9266511 0.9626679 1.1543992 3.1534712 3.2038842 2.7121034 3.0370283  
## 14 4.4585820 2.0655695 2.3403208 3.4726012 3.1990304 3.4902301 3.1859046  
## 15 3.8527282 1.1937521 1.5010755 2.9190773 3.0342823 2.3476421 2.8678297  
## 16 4.2780349 1.7992641 1.7544006 3.6966667 3.9110938 2.9014521 3.5410103  
## 17 1.1516395 3.7772898 3.8710270 0.4959369 0.6400378 1.2052633 0.9456070  
## 18 5.2950469 0.9960504 1.0830876 4.4639940 4.4123030 3.9438542 4.0589639  
## 19 2.1800493 3.3145362 3.5576672 0.7224496 0.4269473 1.5073417 1.3700879  
## 8 9 10 11 12 13 14  
## 2   
## 3   
## 4   
## 5   
## 6   
## 7   
## 8   
## 9 2.8106515   
## 10 2.2618696 2.7310492   
## 11 2.7345527 2.1695599 1.7354273   
## 12 4.9733098 2.7118614 3.9329969 2.6691311   
## 13 2.0363879 2.6398079 0.8589113 1.3939745 3.7375233   
## 14 0.5914261 2.5381635 2.6064969 2.8282744 4.8619004 2.3749627   
## 15 2.5278567 2.5431818 1.0871038 0.8260148 3.3153464 1.0481624 2.7533921  
## 16 3.4174312 3.3313885 1.2636674 1.7764577 3.5961051 1.5965054 3.7338925  
## 17 3.8739706 1.4137608 3.6706682 2.5551808 1.8380458 3.3737126 3.5987458  
## 18 1.8095780 3.4938982 1.4651779 2.7833382 5.2109929 1.8695011 2.2686056  
## 19 3.2468663 1.3911303 3.3826377 2.1055139 2.3815860 3.0044314 2.9454487  
## 15 16 17 18  
## 2   
## 3   
## 4   
## 5   
## 6   
## 7   
## 8   
## 9   
## 10   
## 11   
## 12   
## 13   
## 14   
## 15   
## 16 1.2122349   
## 17 3.2265303 3.9414781   
## 18 2.0977766 2.4660759 4.6574785   
## 19 2.7937204 3.7553911 1.0377559 4.1630214

которая используется функцией кластеризации по *методу ближайшего соседа* с расстоянием Варда между кластерами:

fit <- hclust(d, method = "ward.D")

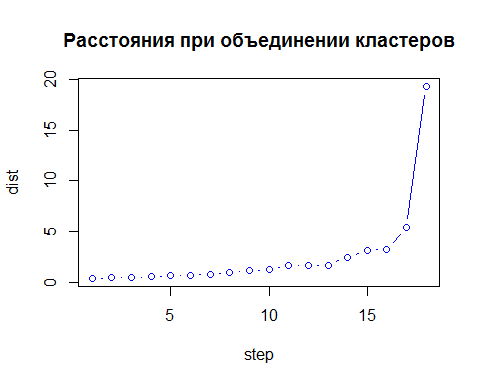
**Дендрограмма полученной кластеризации**:

plot(fit, labels = data$Country, xlab = "Countries")



**сумма внутригрупповых расстояний** по мере объединения кластеров:

plot(fit$height, xlab = "step", ylab = "dist", type = "b", col = "blue", lwd = 1,   
 main = "Расстояния при объединении кластеров")



**Схема объединения** по шагам:

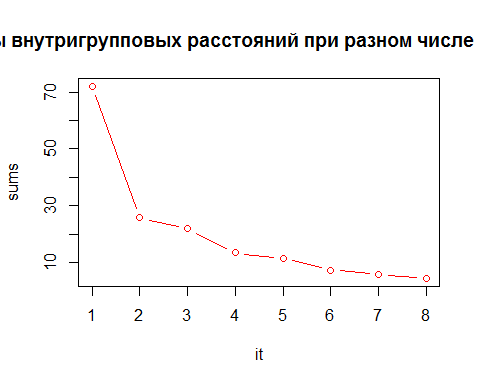
mat = fit$merge  
resu = list()  
countries = as.character(data$Country)  
for (i in 1:nrow(mat)) {  
   
 if (mat[i, 1] < 0) {  
 a = countries[-mat[i, 1]]  
 } else {  
 a = as.character(resu[[mat[i, 1]]])  
 }  
   
 if (mat[i, 2] < 0) {  
 b = countries[-mat[i, 2]]  
 } else {  
 b = as.character(resu[[mat[i, 2]]])  
 }  
   
 resu[[i]] = c(a, b)  
}  
names(resu) = paste("Шаг", 1:nrow(mat), "расстояние", fit$height)  
print(resu)

## $`Шаг 1 расстояние 0.426947302377`  
## [1] "Армения" "Киргизия"  
##   
## $`Шаг 2 расстояние 0.495936862326806`  
## [1] "Азербайджан" "Казахстан"   
##   
## $`Шаг 3 расстояние 0.534651158339984`  
## [1] "Австрия" "Германия"  
##   
## $`Шаг 4 расстояние 0.59142613070119`  
## [1] "Великобритания" "Ирландия"   
##   
## $`Шаг 5 расстояние 0.669921900930876`  
## [1] "Болгария" "Венгрия"   
##   
## $`Шаг 6 расстояние 0.701294749091246`  
## [1] "Австралия" "Австрия" "Германия"   
##   
## $`Шаг 7 расстояние 0.826014818116822`  
## [1] "Греция" "Испания"  
##   
## $`Шаг 8 расстояние 0.974841826914247`  
## [1] "Армения" "Киргизия" "Азербайджан" "Казахстан"   
##   
## $`Шаг 9 расстояние 1.17900272101096`  
## [1] "Дания" "Австралия" "Австрия" "Германия"   
##   
## $`Шаг 10 расстояние 1.26208946988791`  
## [1] "Беларусь" "Болгария" "Венгрия"   
##   
## $`Шаг 11 расстояние 1.68253703237057`  
## [1] "Канада" "Дания" "Австралия" "Австрия" "Германия"   
##   
## $`Шаг 12 расстояние 1.71451000628185`  
## [1] "Россия" "Грузия"  
##   
## $`Шаг 13 расстояние 1.71712349824102`  
## [1] "Италия" "Греция" "Испания"  
##   
## $`Шаг 14 расстояние 2.49230286327314`  
## [1] "Армения" "Киргизия" "Азербайджан" "Казахстан" "Беларусь"   
## [6] "Болгария" "Венгрия"   
##   
## $`Шаг 15 расстояние 3.16150026414533`  
## [1] "Россия" "Грузия" "Армения" "Киргизия" "Азербайджан"  
## [6] "Казахстан" "Беларусь" "Болгария" "Венгрия"   
##   
## $`Шаг 16 расстояние 3.29068732365053`  
## [1] "Канада" "Дания" "Австралия" "Австрия" "Германия" "Италия"   
## [7] "Греция" "Испания"   
##   
## $`Шаг 17 расстояние 5.42357100170829`  
## [1] "Великобритания" "Ирландия" "Канада" "Дания"   
## [5] "Австралия" "Австрия" "Германия" "Италия"   
## [9] "Греция" "Испания"   
##   
## $`Шаг 18 расстояние 19.2875906745862`  
## [1] "Россия" "Грузия" "Армения" "Киргизия"   
## [5] "Азербайджан" "Казахстан" "Беларусь" "Болгария"   
## [9] "Венгрия" "Великобритания" "Ирландия" "Канада"   
## [13] "Дания" "Австралия" "Австрия" "Германия"   
## [17] "Италия" "Греция" "Испания"

### Задание 2

Попробуем узнать, сколько кластеров будет достаточно. Для этого рассчитаем **суммы внутригрупповых расстояний**, когда число кластеров равно 1, 2, … 8, и изобразим их на графике:

it = 1:8  
sums = sapply(it, function(k) kmeans(data[, 2:5], k)$tot.withinss)  
plot(it, sums, type = "b", col = "red", main = "Суммы внутригрупповых расстояний при разном числе кластеров")



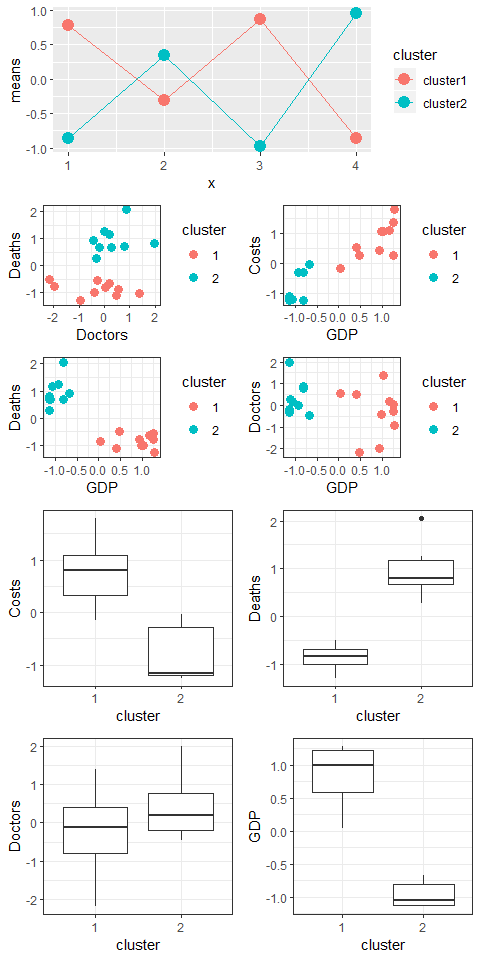
По принципу *метода каменистой осыпи* делаем вывод, что исходный набор данных естественно делится на 2 или 3 кластера. Напишем функцию, которая строит модель для заданного числа кластеров, проводит анализ этой модели и строит некоторые графики:

# функция, проводящая некоторый анализ и строящая графики для заданного  
# числа кластеров  
getimage = function(k) {  
   
 fit = kmeans(data[, 2:5], k) #строится модель  
 cat("Основная информация: \n")  
 print(fit)  
   
 # cat('Внутригрупповые суммы:',fit$withinss,'\n')#внутригрупповые суммы  
 # cat('Общая сумма:', fit$betweenss,'\n')  
 cat("Матрица расстояний:\n")  
 print(dist(fit$centers)) #матрица расстояний  
 cat("Центры:\n")  
 print(fit$centers)  
   
   
 # Добавляем кластер к фрейму данных  
 library(dplyr)  
 newdata = as\_data\_frame(data) %>% mutate(cluster = factor(fit$cluster))  
   
 # агрегирование данных по группам  
 means = newdata[, 2:6] %>% group\_by(cluster) %>% summarise(meanCosts = mean(Costs),   
 sdCosts = sd(Costs), meanDoctors = mean(Doctors), sdDoctors = sd(Doctors),   
 meanGDP = mean(GDP), sdGDP = sd(GDP), meanDeaths = mean(Deaths), sdDeaths = sd(Deaths))  
 print(means)  
   
   
 means = means[, c(1, 2, 4, 6, 8)] #берётся сабсет только из значений для средних  
   
 lbs = c("cluster1", "cluster2", "cluster3", "cluster4", "cluster5")  
   
 library(ggplot2)  
 library(ggpubr)  
   
 # здесь создаётся таблица со средними по каждой переменной и каждому классу  
 # в том виде, в каком удобней рисовать  
 tmpdata = data.frame(x = 1:4, means = as.numeric(means[1, 2:5]), cluster = rep(lbs[1],   
 4))  
 for (i in 2:k) {  
 tmpdata = rbind(tmpdata, data.frame(x = 1:4, means = as.numeric(means[i,   
 2:5]), cluster = rep(lbs[i], 4)))  
 }  
 tmpdata$cluster = factor(tmpdata$cluster)  
   
   
 ppp = ggplot(tmpdata, aes(x = x, y = means, col = cluster)) + geom\_line() +   
 geom\_point(size = 4)  
   
   
 pl1 = ggplot(newdata, aes(x = Doctors, y = Deaths, col = cluster)) + geom\_point(size = 3) +   
 theme\_bw()  
   
 pl2 = ggplot(newdata, aes(x = GDP, y = Costs, col = cluster)) + geom\_point(size = 3) +   
 theme\_bw()  
   
 pl3 = ggplot(newdata, aes(x = GDP, y = Deaths, col = cluster)) + geom\_point(size = 3) +   
 theme\_bw()  
 pl4 = ggplot(newdata, aes(x = GDP, y = Doctors, col = cluster)) + geom\_point(size = 3) +   
 theme\_bw()  
   
 costs = ggplot(newdata, aes(x = cluster, y = Costs)) + geom\_boxplot() +   
 theme\_bw()  
 deaths = ggplot(newdata, aes(x = cluster, y = Deaths)) + geom\_boxplot() +   
 theme\_bw()  
 doctors = ggplot(newdata, aes(x = cluster, y = Doctors)) + geom\_boxplot() +   
 theme\_bw()  
 gdp = ggplot(newdata, aes(x = cluster, y = GDP)) + geom\_boxplot() + theme\_bw()  
   
 p1 <- ggarrange(pl1, pl2, pl3, pl4, ncol = 2, nrow = 2)  
 p2 <- ggarrange(costs, deaths, doctors, gdp, ncol = 2, nrow = 2)  
 ggarrange(ppp, p1, p2, ncol = 1, nrow = 3, heights = c(1.3, 2, 3))  
}  
getimage2 = function(k) {  
 fit = kmeans(data[, 2:5], k) #строится модель  
   
 # Добавляем кластер к фрейму данных  
 library(dplyr)  
 newdata = as\_data\_frame(data) %>% mutate(cluster = factor(fit$cluster))  
 cat("Дисперсионный анализ для каждой переменной,", k, "кластеров \n")  
 cat("Затраты: \n")  
 print(summary(aov(Costs ~ cluster, newdata)))  
 cat("Смерти: \n")  
 print(summary(aov(Deaths ~ cluster, newdata)))  
 cat("Врачи: \n")  
 print(summary(aov(Doctors ~ cluster, newdata)))  
 cat("ВВП: \n")  
 print(summary(aov(GDP ~ cluster, newdata)))  
   
 # рисуются кластеры через главные компоненты library(cluster)  
 # print(clusplot(newdata, newdata$cluster, color=TRUE, shade=TRUE, labels=2,  
 # lines=0))  
 library(factoextra)  
 print(fviz\_cluster(fit, data[, -1], ellipse.type = "norm"))  
}

Используем эту функцию для **двух** кластеров:

getimage(2)

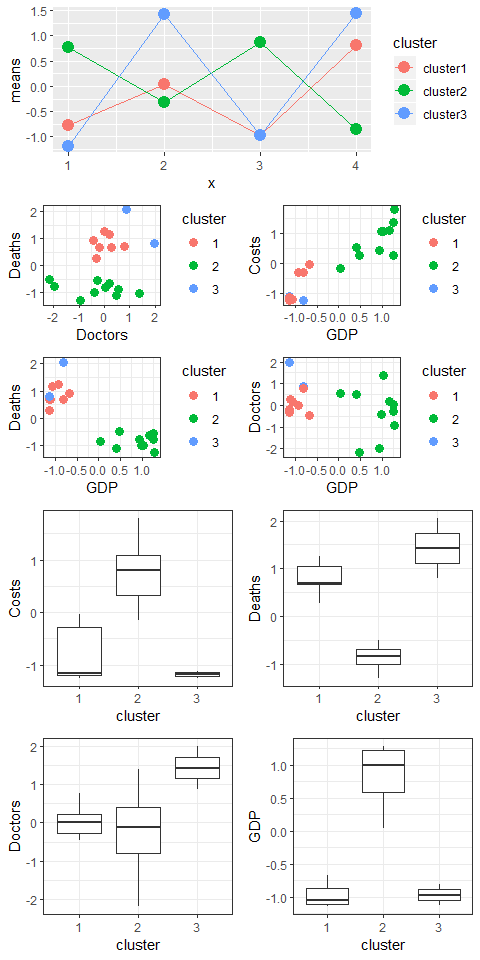
## Основная информация:   
## K-means clustering with 2 clusters of sizes 10, 9  
##   
## Cluster means:  
## Doctors Deaths GDP Costs  
## 1 -0.3094347 -0.8535616 0.8736311 0.7776878  
## 2 0.3438163 0.9484018 -0.9707013 -0.8640976  
##   
## Clustering vector:  
## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19   
## 2 1 1 2 2 2 2 1 2 1 1 2 1 1 1 1 2 1 2   
##   
## Within cluster sum of squares by cluster:  
## [1] 16.671286 9.045834  
## (between\_SS / total\_SS = 64.3 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss"   
## [5] "tot.withinss" "betweenss" "size" "iter"   
## [9] "ifault"   
## Матрица расстояний:  
## 1  
## 2 3.125833  
## Центры:  
## Doctors Deaths GDP Costs  
## 1 -0.3094347 -0.8535616 0.8736311 0.7776878  
## 2 0.3438163 0.9484018 -0.9707013 -0.8640976  
## # A tibble: 2 x 9  
## cluster meanCosts sdCosts meanDoctors sdDoctors meanGDP sdGDP meanDeaths  
## <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1 0.778 0.596 -0.309 1.12 0.874 0.426 -0.854  
## 2 2 -0.864 0.504 0.344 0.766 -0.971 0.177 0.948  
## # ... with 1 more variable: sdDeaths <dbl>



Видно, что исходные наблюдения хорошо разделяются на две группы. Если использовать **три** кластера, такое разделение будет уже сомнительным:

getimage(3)

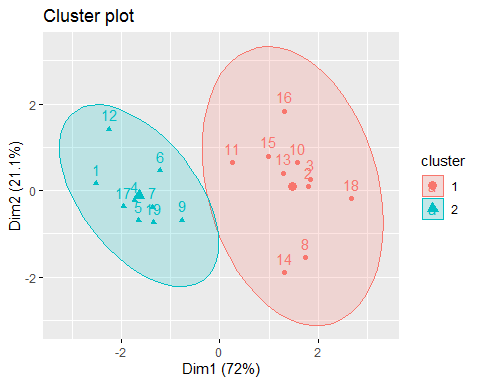
## Основная информация:   
## K-means clustering with 3 clusters of sizes 7, 10, 2  
##   
## Cluster means:  
## Doctors Deaths GDP Costs  
## 1 0.03409616 0.8097220 -0.9719809 -0.7735973  
## 2 -0.30943467 -0.8535616 0.8736311 0.7776878  
## 3 1.42783680 1.4337810 -0.9662225 -1.1808485  
##   
## Clustering vector:  
## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19   
## 3 2 2 1 1 1 1 2 1 2 2 3 2 2 2 2 1 2 1   
##   
## Within cluster sum of squares by cluster:  
## [1] 3.690518 16.671286 1.469772  
## (between\_SS / total\_SS = 69.7 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss"   
## [5] "tot.withinss" "betweenss" "size" "iter"   
## [9] "ifault"   
## Матрица расстояний:  
## 1 2  
## 2 2.949118   
## 3 1.580459 3.933316  
## Центры:  
## Doctors Deaths GDP Costs  
## 1 0.03409616 0.8097220 -0.9719809 -0.7735973  
## 2 -0.30943467 -0.8535616 0.8736311 0.7776878  
## 3 1.42783680 1.4337810 -0.9662225 -1.1808485  
## # A tibble: 3 x 9  
## cluster meanCosts sdCosts meanDoctors sdDoctors meanGDP sdGDP meanDeaths  
## <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1 -0.774 0.543 0.0341 0.419 -0.972 0.183 0.810  
## 2 2 0.778 0.596 -0.309 1.12 0.874 0.426 -0.854  
## 3 3 -1.18 0.0923 1.43 0.790 -0.966 0.226 1.43   
## # ... with 1 more variable: sdDeaths <dbl>



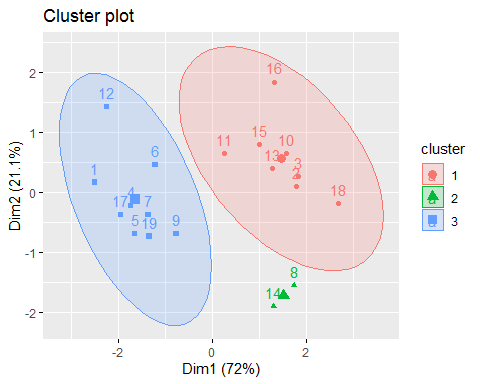
Теперь попробуем визуально оценить качество кластеризации через *главные компоненты* (дополнительно делается *дисперсионный анализ* по каждой переменной):

for (i in 2:5) getimage2(i)

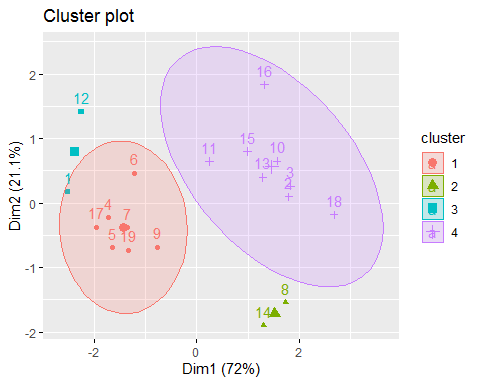
## Дисперсионный анализ для каждой переменной, 2 кластеров   
## Затраты:   
## Df Sum Sq Mean Sq F value Pr(>F)   
## cluster 1 12.768 12.768 41.49 6.09e-06 \*\*\*  
## Residuals 17 5.232 0.308   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Смерти:   
## Df Sum Sq Mean Sq F value Pr(>F)   
## cluster 1 15.381 15.381 99.83 1.57e-08 \*\*\*  
## Residuals 17 2.619 0.154   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Врачи:   
## Df Sum Sq Mean Sq F value Pr(>F)  
## cluster 1 2.021 2.0214 2.151 0.161  
## Residuals 17 15.979 0.9399   
## ВВП:   
## Df Sum Sq Mean Sq F value Pr(>F)   
## cluster 1 16.113 16.113 145.1 9.48e-10 \*\*\*  
## Residuals 17 1.887 0.111   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1



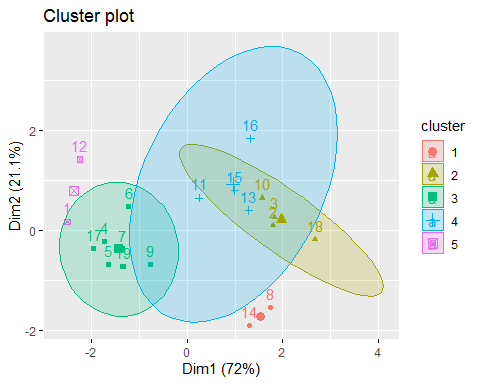
## Дисперсионный анализ для каждой переменной, 3 кластеров   
## Затраты:   
## Df Sum Sq Mean Sq F value Pr(>F)   
## cluster 2 13.195 6.598 21.97 2.58e-05 \*\*\*  
## Residuals 16 4.805 0.300   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Смерти:   
## Df Sum Sq Mean Sq F value Pr(>F)   
## cluster 2 15.497 7.749 49.53 1.4e-07 \*\*\*  
## Residuals 16 2.503 0.156   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Врачи:   
## Df Sum Sq Mean Sq F value Pr(>F)   
## cluster 2 9.809 4.904 9.58 0.00184 \*\*  
## Residuals 16 8.191 0.512   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## ВВП:   
## Df Sum Sq Mean Sq F value Pr(>F)   
## cluster 2 16.189 8.095 71.52 1.05e-08 \*\*\*  
## Residuals 16 1.811 0.113   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1



## Дисперсионный анализ для каждой переменной, 4 кластеров   
## Затраты:   
## Df Sum Sq Mean Sq F value Pr(>F)   
## cluster 3 13.453 4.484 14.79 9.42e-05 \*\*\*  
## Residuals 15 4.547 0.303   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Смерти:   
## Df Sum Sq Mean Sq F value Pr(>F)   
## cluster 3 16.103 5.368 42.44 1.45e-07 \*\*\*  
## Residuals 15 1.897 0.126   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Врачи:   
## Df Sum Sq Mean Sq F value Pr(>F)   
## cluster 3 12.831 4.277 12.41 0.000242 \*\*\*  
## Residuals 15 5.169 0.345   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## ВВП:   
## Df Sum Sq Mean Sq F value Pr(>F)   
## cluster 3 16.189 5.396 44.7 1.02e-07 \*\*\*  
## Residuals 15 1.811 0.121   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

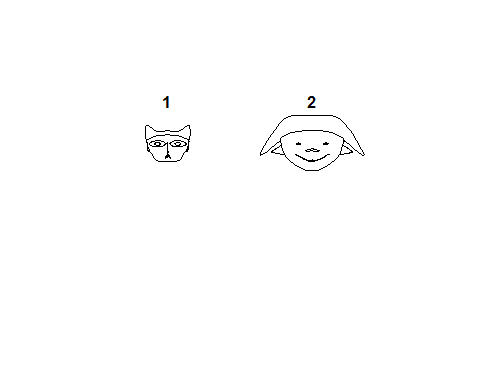


## Дисперсионный анализ для каждой переменной, 5 кластеров   
## Затраты:   
## Df Sum Sq Mean Sq F value Pr(>F)   
## cluster 4 15.085 3.771 18.11 2.01e-05 \*\*\*  
## Residuals 14 2.915 0.208   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Смерти:   
## Df Sum Sq Mean Sq F value Pr(>F)   
## cluster 4 16.114 4.029 29.91 1.01e-06 \*\*\*  
## Residuals 14 1.886 0.135   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Врачи:   
## Df Sum Sq Mean Sq F value Pr(>F)   
## cluster 4 14.707 3.677 15.63 4.6e-05 \*\*\*  
## Residuals 14 3.293 0.235   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## ВВП:   
## Df Sum Sq Mean Sq F value Pr(>F)   
## cluster 4 16.666 4.166 43.71 9.2e-08 \*\*\*  
## Residuals 14 1.334 0.095   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1



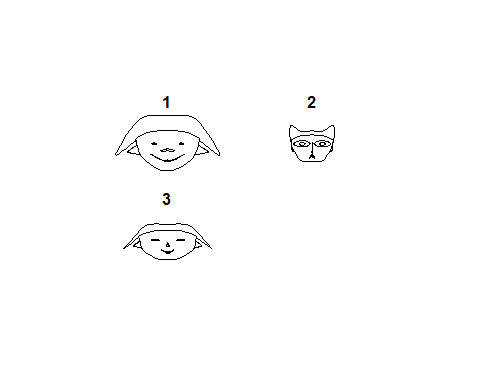
Ещё один способ оценивать качество разбиения на группы — **лица Чернова**. Идея состоит в том, чтобы для каждой группы найти какую-то характеристику каждой переменной (например, среднее), а затем на основе вектора таких характеристик постоить лица каждой группы: *чем лица более похожи, тем группы более близки*. Сделаем так **для двух кластеров**:

# функция делает анализ dataset по методу k-means с k кластерами, затем  
# добавляет результаты к датасету  
getbykmeans = function(dataset, k) {  
 fit = kmeans(dataset, k) #строится модель  
 # Добавляем кластер к фрейму данных  
 library(dplyr)  
 newdata = as\_data\_frame(dataset) %>% mutate(cluster = factor(fit$cluster))  
}  
# функция считает средние и рисует лица  
getfaces = function(k) {  
 # создаем матрицу средних  
 means = getbykmeans(data[, 2:5], k) %>% group\_by(cluster) %>% summarise\_all(funs(mean))  
   
 library(TeachingDemos)  
 faces(means[, 2:5]) #рисуем лица  
}  
getfaces(2)



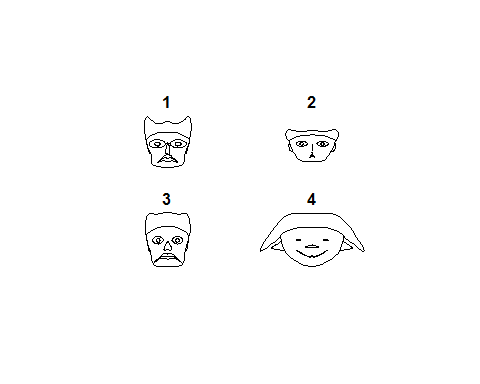
Видно, что лица сильно различаются. **Для трёх кластеров**

getfaces(3)



два лица уже похожи. **Для четырёх кластеров** имеется две пары очень похожих лиц (то есть данные разбиваются на два кластера, а дальнейшее разбиение уже излишнее – происходит поиск различий там, где их нет):

getfaces(4)



Отсюда **вывод**: достаточно было использовать только два кластера.

### Задание 3

Загрузим **данные**:

datacrude = data.frame(read\_excel("Приложение 1.xlsx"))  
data = datacrude[, -c(1)]  
data = data[, -c(1, 2, 16, 17)]  
data %>% tbl\_df()

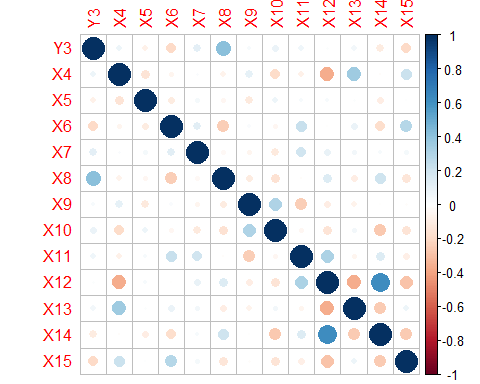
## # A tibble: 53 x 13  
## Y3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 13.3 0.89 0.34 1.73 31 0.28 0.89 0.14 1.12e5 166. 9.89e3  
## 2 10.2 0.93 0.33 0.99 0.15 0.25 1.8 0.3 3.76e4 186. 2.21e4  
## 3 13.7 1.33 0.17 1.73 0.14 0.47 1.53 0.31 4.52e4 220. 1.08e4  
## 4 12.8 0.68 0.32 0.47 0.18 1.53 0.6 0.18 7.67e4 169. 1.03e4  
## 5 10.6 0.89 0.36 1.73 0.31 0.21 1.39 0.37 7.36e3 39.9 5.53e4  
## 6 9.12 1.53 0.33 1.33 0.17 0.13 1.24 0.19 8.45e4 40.4 4.53e4  
## 7 26.0 1.12 0.15 0.97 0.26 0.38 1.77 0.41 1.14e5 103. 1.27e4  
## 8 23.4 0.99 0.32 1.82 0.290 0.38 0.09 0.36 7.80e3 37.0 5.76e4  
## 9 14.7 1.65 0.31 0.68 0.26 0.2 0.52 0.41 8.45e4 45.9 1.18e5  
## 10 10.0 0.56 0.15 1.8 0.28 0.35 0.8 2.06 3.59e4 40.1 6.44e4  
## # ... with 43 more rows, and 2 more variables: X14 <dbl>, X15 <dbl>

**Визуализация корреляционной матрицы** заданного набора переменных:

library(corrplot)  
(matr = cor(data))

## Y3 X4 X5 X6 X7  
## Y3 1.00000000 0.07650448 -0.072024390 -0.19034971 0.11487125  
## X4 0.07650448 1.00000000 -0.143222378 -0.05165605 0.02087309  
## X5 -0.07202439 -0.14322238 1.000000000 -0.10105745 0.04470227  
## X6 -0.19034971 -0.05165605 -0.101057451 1.00000000 0.12261894  
## X7 0.11487125 0.02087309 0.044702272 0.12261894 1.00000000  
## X8 0.41247593 -0.06787551 -0.041525168 -0.24074470 -0.03683463  
## X9 0.03646200 0.10789499 -0.113522027 0.02500297 -0.05390234  
## X10 0.08137700 -0.18976437 0.070132267 -0.04524674 -0.11608259  
## X11 0.06540456 -0.07079232 0.033116213 0.22954802 0.19025176  
## X12 0.01947580 -0.36101268 0.032584704 -0.00409325 0.09312382  
## X13 0.05655101 0.35261229 0.017754637 0.08272441 0.05377388  
## X14 -0.10504111 -0.01039568 -0.090105051 -0.17096373 0.03176305  
## X15 -0.19945635 0.21542253 0.008607521 0.28241807 0.04977926  
## X8 X9 X10 X11 X12  
## Y3 0.41247593 0.0364619968 0.08137700 0.06540456 0.01947580  
## X4 -0.06787551 0.1078949930 -0.18976437 -0.07079232 -0.36101268  
## X5 -0.04152517 -0.1135220265 0.07013227 0.03311621 0.03258470  
## X6 -0.24074470 0.0250029745 -0.04524674 0.22954802 -0.00409325  
## X7 -0.03683463 -0.0539023369 -0.11608259 0.19025176 0.09312382  
## X8 1.00000000 -0.1007706318 -0.15649952 -0.01102561 0.14129715  
## X9 -0.10077063 1.0000000000 0.30517775 -0.24507910 -0.09717025  
## X10 -0.15649952 0.3051777477 1.00000000 -0.04511818 -0.14281619  
## X11 -0.01102561 -0.2450790952 -0.04511818 1.00000000 0.31083103  
## X12 0.14129715 -0.0971702535 -0.14281619 0.31083103 1.00000000  
## X13 -0.09511029 -0.0677385171 0.05003536 -0.05178378 -0.36987532  
## X14 0.20164538 -0.0005369078 -0.26948437 0.14258917 0.62254065  
## X15 -0.13295559 -0.0159931107 -0.14937007 -0.08199636 -0.28801774  
## X13 X14 X15  
## Y3 0.05655101 -0.1050411148 -0.199456352  
## X4 0.35261229 -0.0103956837 0.215422531  
## X5 0.01775464 -0.0901050514 0.008607521  
## X6 0.08272441 -0.1709637327 0.282418070  
## X7 0.05377388 0.0317630498 0.049779257  
## X8 -0.09511029 0.2016453817 -0.132955585  
## X9 -0.06773852 -0.0005369078 -0.015993111  
## X10 0.05003536 -0.2694843746 -0.149370073  
## X11 -0.05178378 0.1425891667 -0.081996358  
## X12 -0.36987532 0.6225406492 -0.288017745  
## X13 1.00000000 -0.2539722434 0.073307149  
## X14 -0.25397224 1.0000000000 -0.254812455  
## X15 0.07330715 -0.2548124551 1.000000000

corrplot(matr)



В наборе данных в целом нет значительных корреляций, поэтому сжать его до трёх-четырёх главных компонент без сильной потери информации не получится.

**Результаты аналаза методом главных компонент без вращения**:

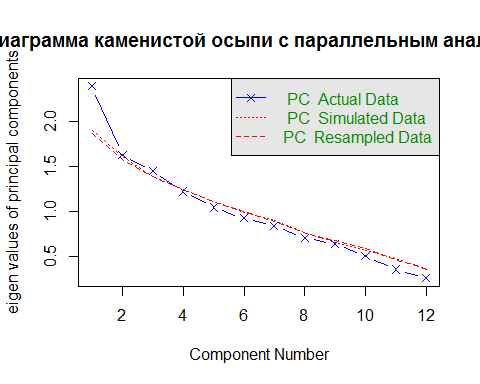
library(psych)  
principal(data[, -1], nfactors = 8, rotate = "none") #Создание модели

## Principal Components Analysis  
## Call: principal(r = data[, -1], nfactors = 8, rotate = "none")  
## Standardized loadings (pattern matrix) based upon correlation matrix  
## PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8 h2 u2 com  
## X4 -0.45 0.21 -0.60 0.18 0.31 0.12 -0.23 0.18 0.84 0.1588 3.7  
## X5 0.03 -0.03 0.29 -0.66 -0.02 0.52 -0.27 0.30 0.96 0.0374 3.3  
## X6 -0.20 0.55 0.45 0.37 -0.15 -0.19 0.03 0.17 0.77 0.2290 3.8  
## X7 0.08 0.49 0.12 0.00 0.38 0.50 0.54 -0.24 0.99 0.0061 4.4  
## X8 0.37 -0.14 -0.46 -0.23 -0.12 -0.16 0.56 0.46 0.99 0.0132 4.7  
## X9 -0.21 -0.47 0.10 0.62 0.16 0.34 0.08 0.30 0.90 0.0986 3.6  
## X10 -0.23 -0.54 0.56 0.01 0.30 -0.11 0.10 0.17 0.80 0.1992 3.3  
## X11 0.36 0.53 0.32 -0.04 0.31 -0.28 -0.07 0.32 0.80 0.2047 4.8  
## X12 0.85 0.10 0.14 0.15 0.04 0.07 -0.14 0.05 0.80 0.1977 1.2  
## X13 -0.54 0.19 -0.19 -0.24 0.51 -0.19 -0.06 0.08 0.73 0.2682 3.3  
## X14 0.73 0.03 -0.32 0.28 0.15 0.16 -0.25 0.08 0.84 0.1649 2.3  
## X15 -0.47 0.42 -0.04 0.12 -0.53 0.24 0.01 0.23 0.81 0.1936 3.9  
##   
## PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8  
## SS loadings 2.39 1.63 1.45 1.23 1.05 0.93 0.84 0.71  
## Proportion Var 0.20 0.14 0.12 0.10 0.09 0.08 0.07 0.06  
## Cumulative Var 0.20 0.34 0.46 0.56 0.65 0.72 0.79 0.85  
## Proportion Explained 0.23 0.16 0.14 0.12 0.10 0.09 0.08 0.07  
## Cumulative Proportion 0.23 0.39 0.54 0.66 0.76 0.85 0.93 1.00  
##   
## Mean item complexity = 3.5  
## Test of the hypothesis that 8 components are sufficient.  
##   
## The root mean square of the residuals (RMSR) is 0.06   
## with the empirical chi square 27.65 with prob < NA   
##   
## Fit based upon off diagonal values = 0.87

Здесь *сперва выведена матрица корреляций*, затем **SS loadings** — это собственные значения каждой компоненты, **Proportion Var** — доля дисперсий, объясняемых каждой компонентой, **Cumulative Var** — кумулятивная доля (первая компонента объясняет 20%, первые две вместе — 34%, первые три — 46% и т. д.), дальше то же самое, только в пропорции. *По принципу Кайзера* следует выделить только 5 компонент, хотя, судя по объяснённым дисперсиям, и восьми может быть недостаточно.

Построим **диаграмму каменистой осыпи**:

fa.parallel(data[, -1], fa = "pc", show.legend = T, main = "Диаграмма каменистой осыпи с параллельным анализом")



Здесь, как и ожидалось, не наблюдается резкого убывания собственных значений, поэтому нельзя сказать, сколько конкретно главных компонент следовало бы выделить. Возмём 6 факторов и **проведём анализ с вращением осей**:

# варимакс с нормализацией  
(vm = principal(apply(data[, -1], 2, scale), nfactors = 6, rotate = "varimax"))

## Principal Components Analysis  
## Call: principal(r = apply(data[, -1], 2, scale), nfactors = 6, rotate = "varimax")  
## Standardized loadings (pattern matrix) based upon correlation matrix  
## RC1 RC5 RC3 RC2 RC4 RC6 h2 u2 com  
## X4 0.64 -0.02 -0.27 -0.28 0.32 0.30 0.75 0.25 2.9  
## X5 -0.04 -0.06 0.05 -0.02 -0.87 0.17 0.80 0.20 1.1  
## X6 -0.06 0.75 0.06 0.33 0.23 0.08 0.74 0.26 1.6  
## X7 0.05 0.13 -0.06 0.16 -0.18 0.75 0.65 0.35 1.3  
## X8 -0.10 -0.49 -0.39 0.02 0.03 -0.24 0.46 0.54 2.5  
## X9 -0.10 0.06 0.51 -0.61 0.32 0.24 0.81 0.19 2.9  
## X10 0.05 -0.03 0.85 -0.04 -0.10 -0.16 0.76 0.24 1.1  
## X11 -0.10 0.07 0.02 0.78 0.08 0.25 0.69 0.31 1.3  
## X12 -0.71 -0.27 -0.08 0.33 0.10 0.29 0.78 0.22 2.2  
## X13 0.82 -0.03 0.07 0.16 0.01 0.09 0.72 0.28 1.1  
## X14 -0.48 -0.46 -0.27 0.04 0.32 0.39 0.77 0.23 4.4  
## X15 0.13 0.71 -0.39 -0.27 -0.10 -0.01 0.75 0.25 2.0  
##   
## RC1 RC5 RC3 RC2 RC4 RC6  
## SS loadings 1.88 1.63 1.45 1.41 1.18 1.14  
## Proportion Var 0.16 0.14 0.12 0.12 0.10 0.09  
## Cumulative Var 0.16 0.29 0.41 0.53 0.63 0.72  
## Proportion Explained 0.22 0.19 0.17 0.16 0.14 0.13  
## Cumulative Proportion 0.22 0.40 0.57 0.73 0.87 1.00  
##   
## Mean item complexity = 2  
## Test of the hypothesis that 6 components are sufficient.  
##   
## The root mean square of the residuals (RMSR) is 0.09   
## with the empirical chi square 56.3 with prob < 6.9e-09   
##   
## Fit based upon off diagonal values = 0.74

По результатам анализа видно, что *первая компонента сильно коррелирует с переменной Х13 (среднегодовой фонд заработной платы), пятая — с Х6 (удельный вес покупных изделий) и Х15 (оборачиваемость нормированных оборотных средств), третья — с Х10 (фондоотдача) и Х9 (удельный вес потерь от брака), вторая — с Х11 (среднегодовая численность ППП), четвёртая — с Х5 (удельный вес рабочих в составе ППП), шестая — с Х7 (коэффициент сменности оборудования)*.

Также можно увидеть **матрицу весовых коэффициентов**:

round(unclass(vm$weights), 2) #делается округление до второго знака, чтобы не занимала много места

## RC1 RC5 RC3 RC2 RC4 RC6  
## X4 0.37 -0.11 -0.16 -0.16 0.24 0.30  
## X5 -0.04 -0.05 0.00 -0.11 -0.76 0.23  
## X6 -0.10 0.50 0.05 0.24 0.22 -0.01  
## X7 0.06 0.05 0.01 0.00 -0.20 0.69  
## X8 0.01 -0.28 -0.27 0.02 0.01 -0.24  
## X9 -0.14 0.05 0.34 -0.47 0.23 0.32  
## X10 0.04 -0.07 0.59 0.06 -0.03 -0.07  
## X11 0.06 0.03 0.10 0.57 0.11 0.11  
## X12 -0.32 -0.07 0.00 0.12 0.06 0.19  
## X13 0.54 -0.17 0.09 0.25 0.05 0.10  
## X14 -0.18 -0.22 -0.13 -0.08 0.22 0.31  
## X15 -0.10 0.48 -0.34 -0.26 -0.11 -0.02

а *корреляционная матрица для главных компонент показывает их ортогональность*:

cor(vm$scores)

## RC1 RC5 RC3 RC2 RC4  
## RC1 1.000000e+00 -8.012483e-16 -2.760045e-16 4.618482e-16 -1.510586e-17  
## RC5 -8.012483e-16 1.000000e+00 2.955139e-16 -4.884503e-16 9.632302e-16  
## RC3 -2.760045e-16 2.955139e-16 1.000000e+00 3.283986e-17 -2.461076e-16  
## RC2 4.618482e-16 -4.884503e-16 3.283986e-17 1.000000e+00 1.488276e-16  
## RC4 -1.510586e-17 9.632302e-16 -2.461076e-16 1.488276e-16 1.000000e+00  
## RC6 -1.196198e-16 6.003697e-16 -2.182999e-16 1.132043e-16 -1.495914e-15  
## RC6  
## RC1 -1.196198e-16  
## RC5 6.003697e-16  
## RC3 -2.182999e-16  
## RC2 1.132043e-16  
## RC4 -1.495914e-15  
## RC6 1.000000e+00

### Задание 4

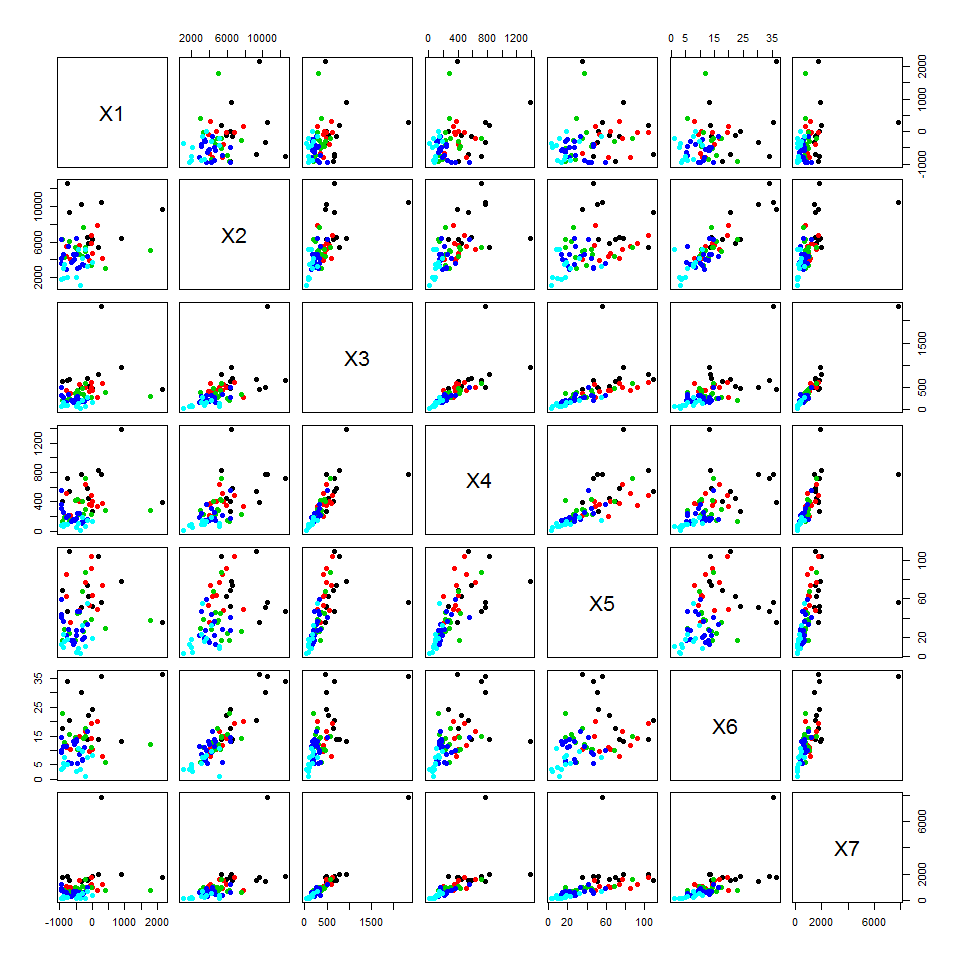
Загрузим **данные**:

data = data.frame(read\_excel("Приложение 2.xlsx"))  
data$CLASS = factor(data$CLASS)  
data %>% tbl\_df()

## # A tibble: 65 x 8  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <fct>  
## 1 2174 9658 466 386 35 36.4 1756 1   
## 2 274 10477 2321 767 56 35.6 7884 1   
## 3 -146 6567 713 581 74 13.8 1501 1   
## 4 -338 10282 499 764 51 30.2 1466 1   
## 5 -716 9316 677 533 109 20.5 1486 1   
## 6 893. 6425 944 1390 78 13.2 1936 1   
## 7 191 5367 786 819 104 13.7 2011 1   
## 8 0 6342 486 261 52 24.1 1841 1   
## 9 -107 5868 531 450 63 22.3 1608 1   
## 10 -903 6330 636 401 69 17.6 1768 1   
## # ... with 55 more rows

**Визуализация**:

pairs(data[, 1:7], col = data$CLASS, pch = 16)

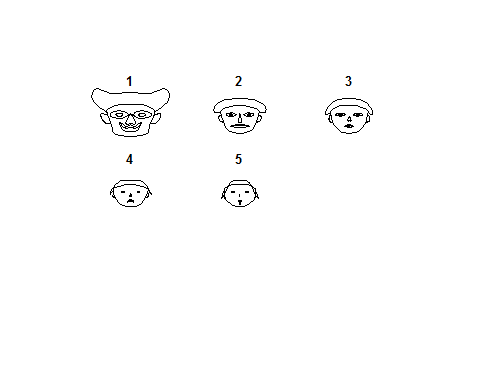


Напишем несколько вспомогательных функций:

library(MASS)  
# лица Чернова  
showfaces = function() {  
 newdata = as\_data\_frame(data) %>% group\_by(CLASS) %>% summarise\_all(funs(mean))  
 print(faces(newdata[, 2:8])) #рисуем лица  
}  
# визуализация кластеров через главные компоненты  
showimage = function() {  
 library(factoextra)  
 print(fviz\_cluster(list(data = data[, 1:7], cluster = data[, 8]), ellipse.type = "norm"))  
}  
  
# матрицы, обратные матрицам ковариации для каждого класса  
covinv = function(df) {  
 res = list()  
 for (i in 1:length(levels(df$CLASS))) res[[i]] = tryCatch({  
 df[df$CLASS == i, 1:7] %>% as.matrix() %>% cov() %>% solve  
 }, error = function(r) NA)  
 # res[[i]]=df[df$CLASS==i,1:7] %>% as.matrix() %>% cov() %>% solve  
 res  
}  
  
# расстояния Махаланобиса от элемента до каждого из классов  
distance = function(means, covs, elem) {  
 res = c()  
 for (i in 1:nrow(means)) {  
 vec = (means[i, ] - elem)  
 res[i] = (vec %\*% covs[[i]]) %\*% vec  
 }  
   
 return(sqrt(res))  
}  
  
# поиск номера элемента в датафрейме  
find.number = function(df, elem) {  
 sm = 0  
 i = 0  
 len = length(elem)  
 while (sm != len) {  
 i = i + 1  
 v = ifelse(df[i, ] == elem, T, F)  
 sm = sum(v)  
 }  
 return(i)  
}

Лица Чернова показывают, что в двух случаях между парами групп сильной разницы нет:

showfaces()



Проведём обучение через *линейный дискриминантный анализ*:

ldadat <- lda(CLASS ~ ., data, method = "moment")

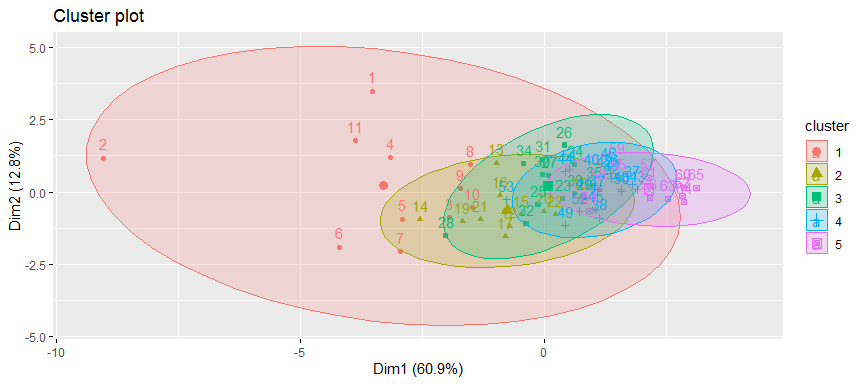
**Результаты обучения**:

# Функция вывода результатов классификации  
Out\_CTab <- function(model, group) {  
 cat("Таблица неточностей \"Факт/Прогноз\" по обучающей выборке: \n")  
 classified <- predict(model)$class  
 t1 <- table(group, classified)  
 print(t1)  
 # Точность классификации и расстояние Махалонобиса  
 Err\_S <- mean(group != classified)  
 mahDist <- dist(model$means %\*% model$scaling)  
 cat("Точность классификации:", 1 - Err\_S[1], "\n")  
 cat("Расстояния Махалонобиса:\n")  
 print(mahDist)  
   
 # Таблица 'Факт/Прогноз' и ошибка при скользящем контроле  
 t2 <- table(group, LDA.cv <- update(model, CV = T)$class)  
 Err\_CV <- mean(group != LDA.cv)  
 cat("Ошибка при скользящем контроле:", Err\_CV[1], "\n")  
 Err\_S.MahD <- c(Err\_S, mahDist)  
 Err\_CV.N <- c(Err\_CV, length(group))  
 cbind(t1, Err\_S.MahD, t2, Err\_CV.N)  
   
 cat("Результаты многомерного дисперсионного анализа: \n")  
 ldam <- manova(as.matrix(data[, 1:7]) ~ data$CLASS)  
 print(summary(ldam, test = "Wilks"))  
}  
Out\_CTab(ldadat, data$CLASS)

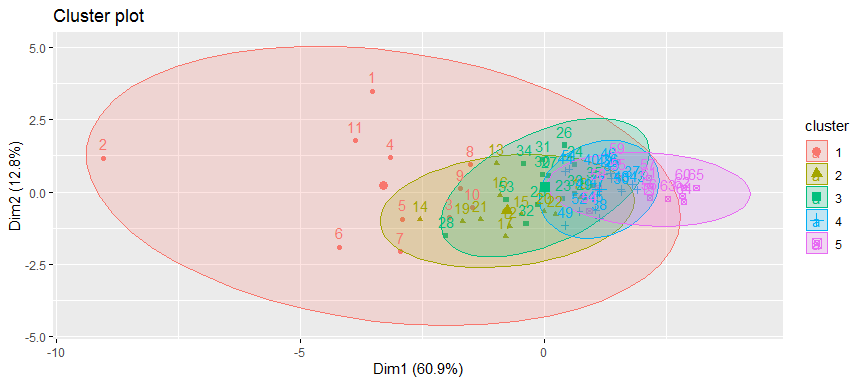
## Таблица неточностей "Факт/Прогноз" по обучающей выборке:   
## classified  
## group 1 2 3 4 5  
## 1 9 2 0 0 0  
## 2 0 8 2 1 0  
## 3 0 2 7 4 0  
## 4 0 0 2 15 1  
## 5 0 0 0 3 9  
## Точность классификации: 0.7384615   
## Расстояния Махалонобиса:  
## 1 2 3 4  
## 2 3.016556   
## 3 3.680769 1.726019   
## 4 4.766641 2.584622 1.213160   
## 5 6.461644 4.081787 2.854497 1.855913  
## Ошибка при скользящем контроле: 0.4615385   
## Результаты многомерного дисперсионного анализа:   
## Df Wilks approx F num Df den Df Pr(>F)   
## data$CLASS 4 0.12674 5.4173 28 196.12 2.994e-13 \*\*\*  
## Residuals 60   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

*Точность классификации* достаточно низкая (74%), вдобавок *лямбда Уилкса* (**0.126**) сильно отличается от нуля. Откорректируем классификацию, чтобы добиться стопроцентной точности:

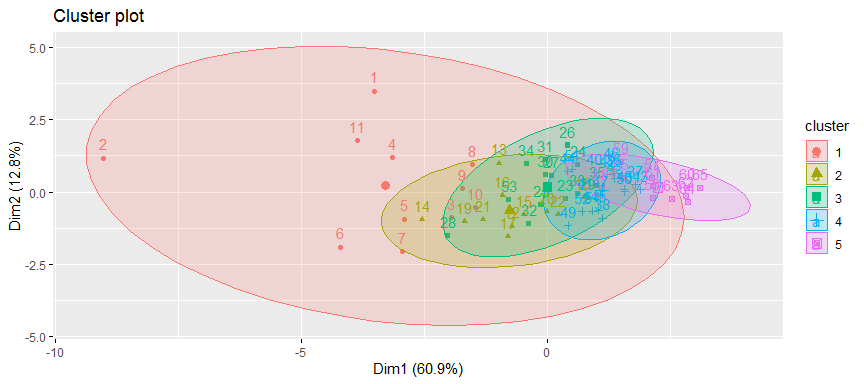
acc = 0 #точность  
repeat {  
 showimage()  
   
 ldadat <- lda(CLASS ~ ., data, method = "moment")  
 means = ldadat$means  
 cov.mat = covinv(data)  
   
 # для всех неправильно найденных найти расстояния до кластеров, отнесённых  
 # экспертами  
 prclass = predict(ldadat, data[, 1:7])$class  
 st = data[data$CLASS != prclass, ]  
 acc = 1 - nrow(st)/nrow(data)  
 cat("Точность классификации:", acc, "\n")  
 if (near(acc, 1))   
 break  
   
 if (nrow(st) == 1) {  
 number.of.max.distance = 1  
 } else {  
 distances = c()  
 for (i in 1:nrow(st)) {  
 cls = as.numeric(st[i, 8])  
 vec = (means[cls, ] - as.numeric(st[i, 1:7]))  
 if (is.na(cov.mat[[cls]])) {  
 distances[i] = NA  
 } else {  
 distances[i] = (vec %\*% cov.mat[[cls]]) %\*% vec  
 }  
 }  
 distances = sqrt(distances)  
 # номер элемента (в таблице неверно отнесённых) с максимальным расстоянием  
 # для своего кластера  
 number.of.max.distance = which.max(distances)  
 }  
   
 tt = st[number.of.max.distance, ] #сам элемент  
 cat("Неправильно отнесённый элемент с максимальным расстоянием (", max(distances,   
 na.rm = T), ")\n")  
 # номер того же элемента, но в исходном фрейме  
 number.of.max.distance.new = find.number(data, tt)  
 print(data[number.of.max.distance.new, ])  
 # сделать замену на кластер с минимальным расстоянием  
 data[number.of.max.distance.new, 8] = predict(ldadat, tt[, 1:7])$class #levels(data$CLASS)[which.min(distance(ldadat$means,cov.mat,as.numeric(tt[,-8])))]   
 cat("Заменяется на\n")  
 print(data[number.of.max.distance.new, ])  
}



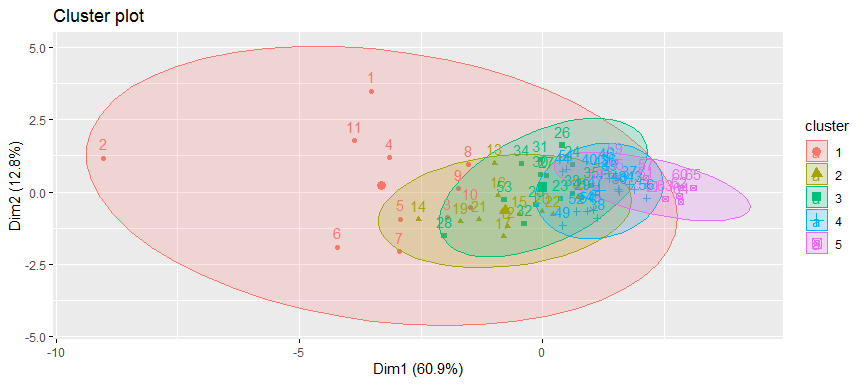
## Точность классификации: 0.7384615   
## Неправильно отнесённый элемент с максимальным расстоянием ( 3.93429 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 53 -934 6322 510 548 41 14.7 1187 4  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 53 -934 6322 510 548 41 14.7 1187 3



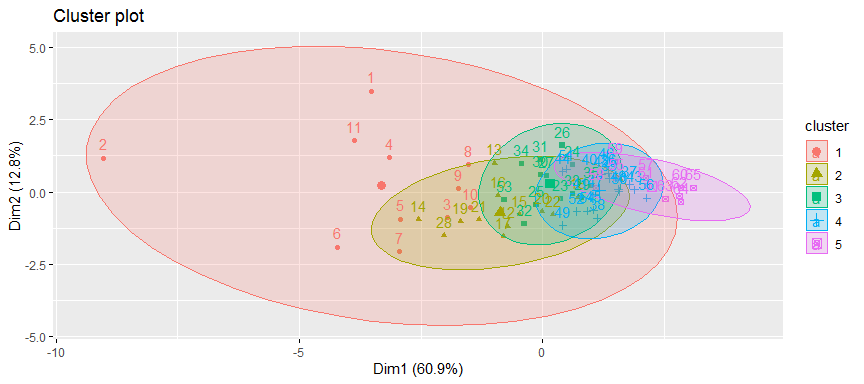
## Точность классификации: 0.7846154   
## Неправильно отнесённый элемент с максимальным расстоянием ( 3.016765 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 54 -161.6 3196 288 149 55 7.6 684 5  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 54 -161.6 3196 288 149 55 7.6 684 4



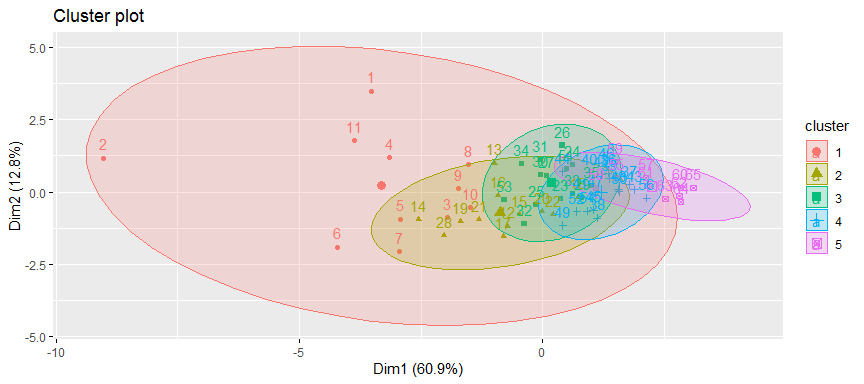
## Точность классификации: 0.7846154   
## Неправильно отнесённый элемент с максимальным расстоянием ( 2.970361 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 56 -879 3058 169 86 23 5.6 307 5  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 56 -879 3058 169 86 23 5.6 307 4



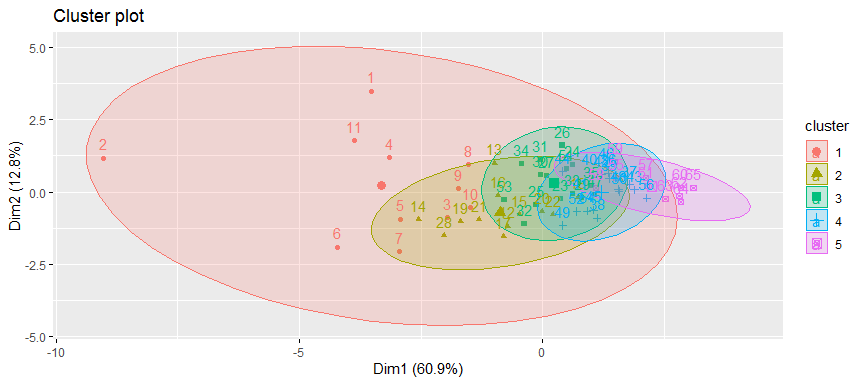
## Точность классификации: 0.8   
## Неправильно отнесённый элемент с максимальным расстоянием ( 2.904711 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 28 -205 5357 583 716 87 14.8 1606.6 3  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 28 -205 5357 583 716 87 14.8 1606.6 2



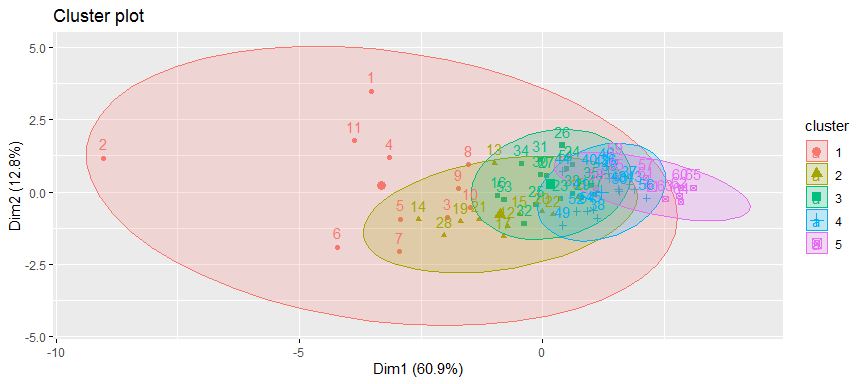
## Точность классификации: 0.8153846   
## Неправильно отнесённый элемент с максимальным расстоянием ( 3.172017 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 51 -500 6368 288 169 27 13.3 601 4  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 51 -500 6368 288 169 27 13.3 601 3



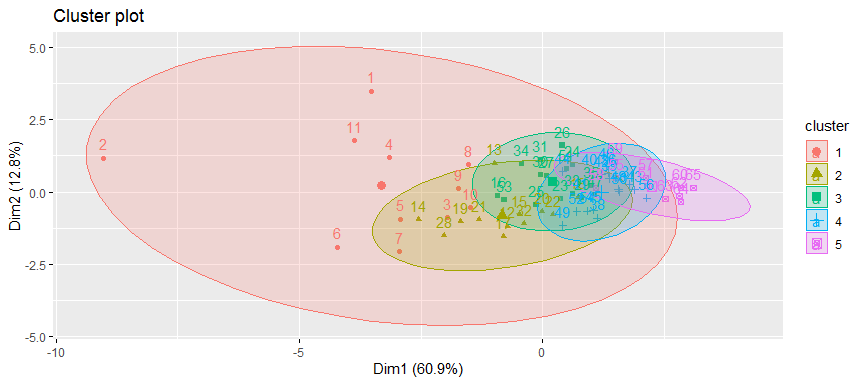
## Точность классификации: 0.8153846   
## Неправильно отнесённый элемент с максимальным расстоянием ( 3.208787 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 47 -728.3 5448 348 215 28 5.7 367 4  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 47 -728.3 5448 348 215 28 5.7 367 3



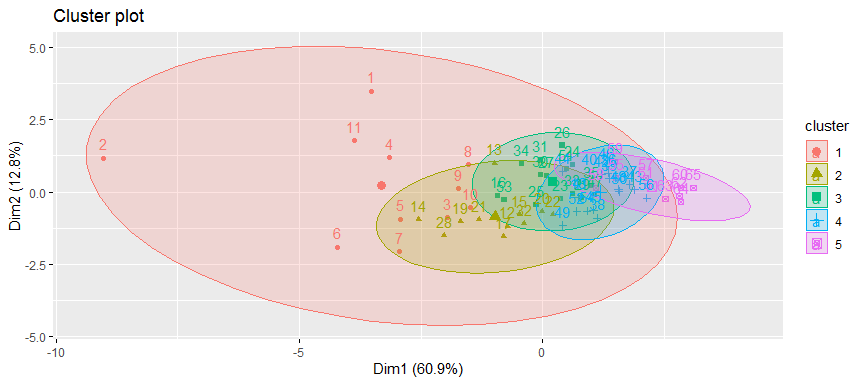
## Точность классификации: 0.8153846   
## Неправильно отнесённый элемент с максимальным расстоянием ( 2.955268 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 16 -380 5564 565 400 48 14.9 1517 2  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 16 -380 5564 565 400 48 14.9 1517 3



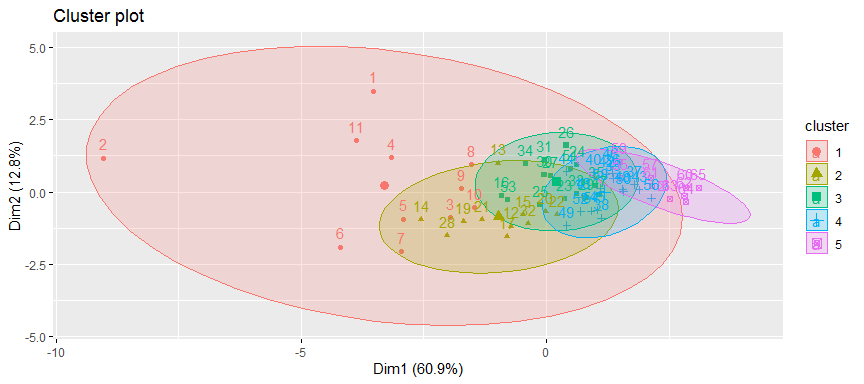
## Точность классификации: 0.8461538   
## Неправильно отнесённый элемент с максимальным расстоянием ( 2.838619 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 32 -314 4394 471 396 68 9.9 1065 3  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 32 -314 4394 471 396 68 9.9 1065 2



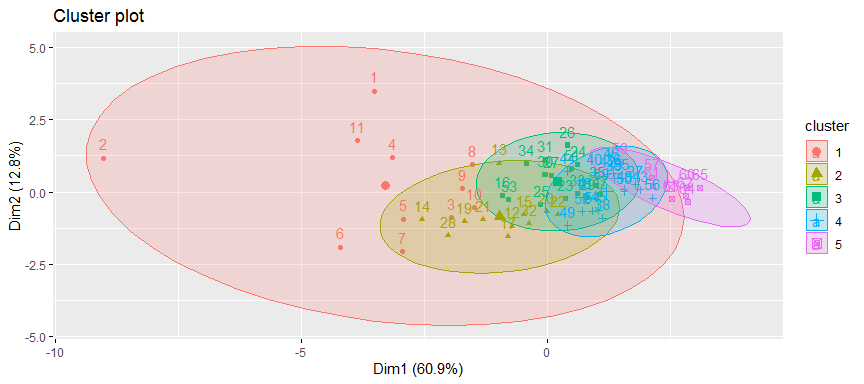
## Точность классификации: 0.8461538   
## Неправильно отнесённый элемент с максимальным расстоянием ( 2.488618 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 18 -666.8 3988 364 213 35 10.3 943 2  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 18 -666.8 3988 364 213 35 10.3 943 4



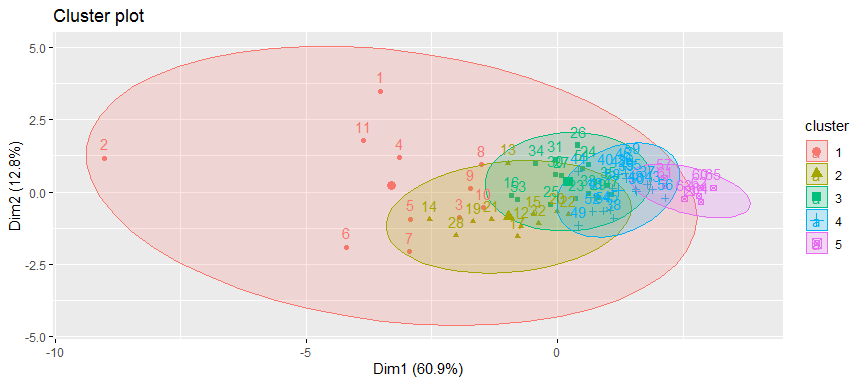
## Точность классификации: 0.8615385   
## Неправильно отнесённый элемент с максимальным расстоянием ( 2.434652 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 58 -310.7 4166 207 183 32 9.8 487 5  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 58 -310.7 4166 207 183 32 9.8 487 4



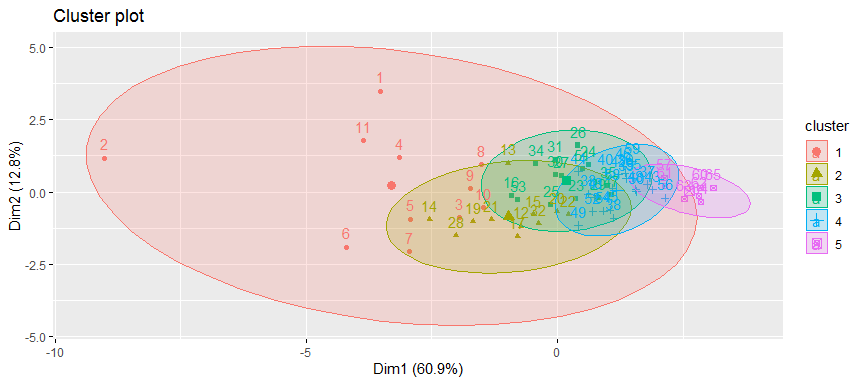
## Точность классификации: 0.8769231   
## Неправильно отнесённый элемент с максимальным расстоянием ( 2.629277 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 55 -4 3666 168 131 19 8.3 382 5  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 55 -4 3666 168 131 19 8.3 382 4



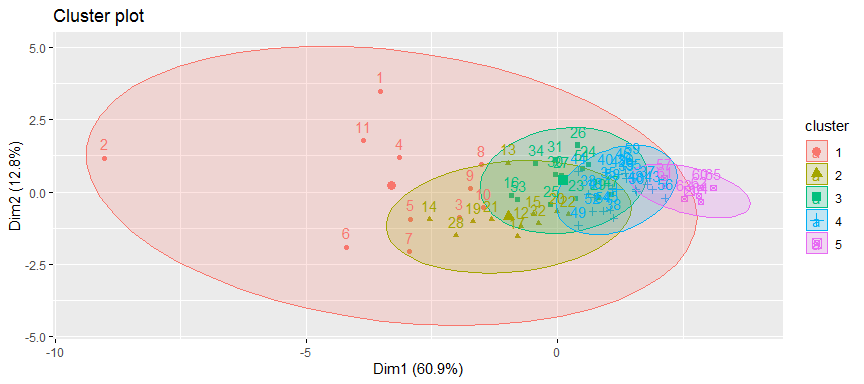
## Точность классификации: 0.8769231   
## Неправильно отнесённый элемент с максимальным расстоянием ( 2.474874 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 59 -437 5168 151 96 8 10.7 359 5  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 59 -437 5168 151 96 8 10.7 359 4



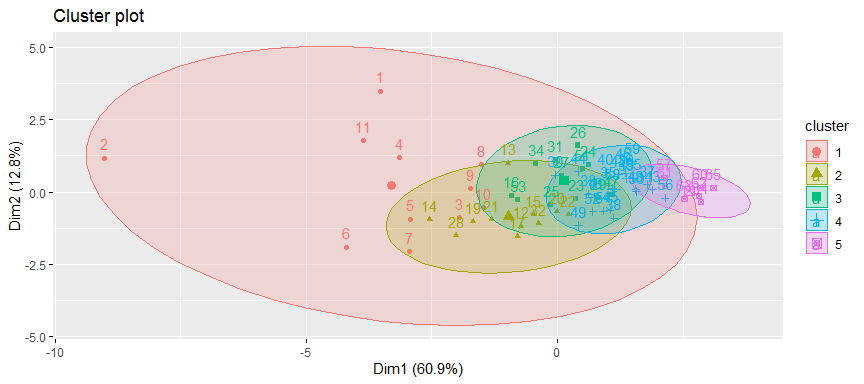
## Точность классификации: 0.8923077   
## Неправильно отнесённый элемент с максимальным расстоянием ( 2.333742 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 33 -27 3312 284 229 39 11.1 948 3  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 33 -27 3312 284 229 39 11.1 948 4



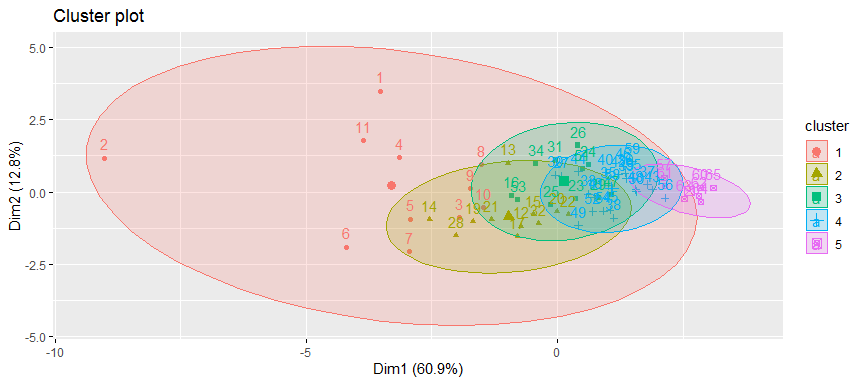
## Точность классификации: 0.8923077   
## Неправильно отнесённый элемент с максимальным расстоянием ( 2.664493 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 35 -842 4247 233 189 28 12.8 757 3  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 35 -842 4247 233 189 28 12.8 757 4



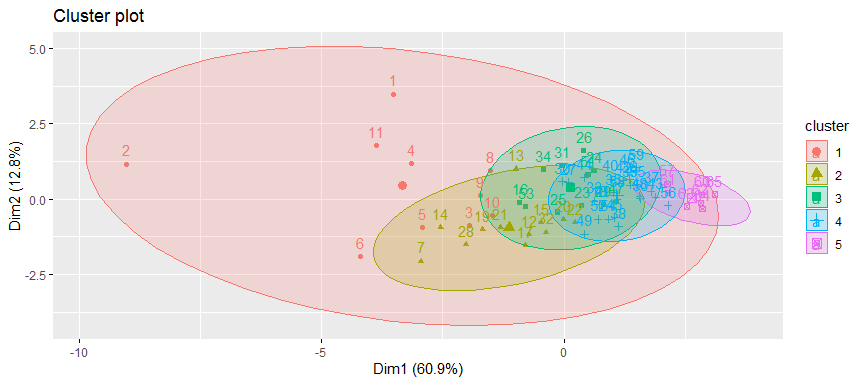
## Точность классификации: 0.9076923   
## Неправильно отнесённый элемент с максимальным расстоянием ( 2.489248 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 30 -205 4924 284 292 35 17.5 1010 3  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 30 -205 4924 284 292 35 17.5 1010 4



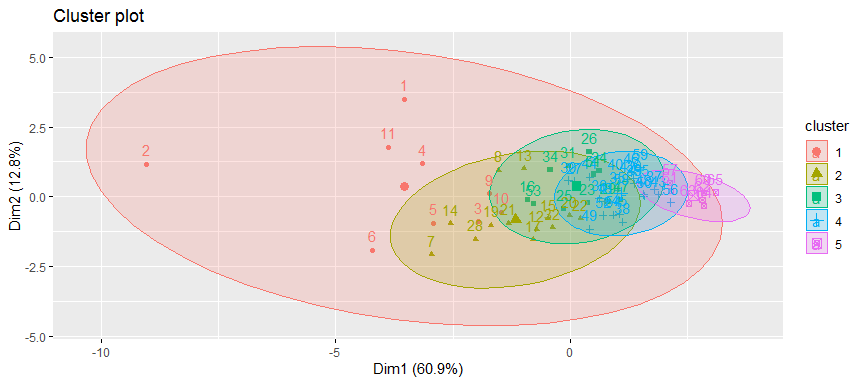
## Точность классификации: 0.9076923   
## Неправильно отнесённый элемент с максимальным расстоянием ( 2.894696 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 27 -514 5340 364 411 17 14.4 984 3  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 27 -514 5340 364 411 17 14.4 984 4



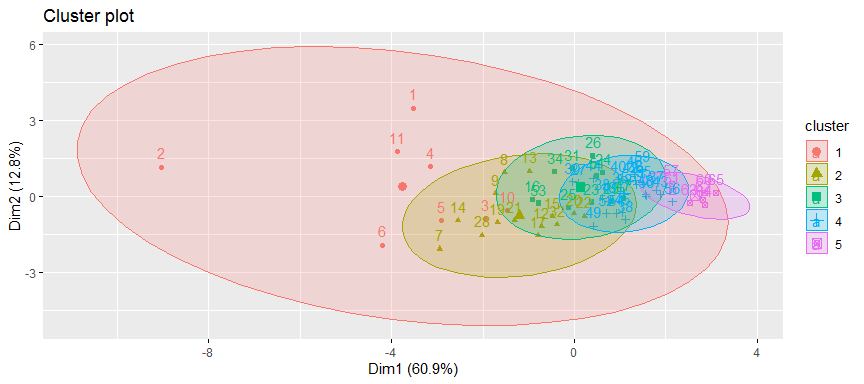
## Точность классификации: 0.9230769   
## Неправильно отнесённый элемент с максимальным расстоянием ( 2.275641 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 7 191 5367 786 819 104 13.7 2011 1  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 7 191 5367 786 819 104 13.7 2011 2



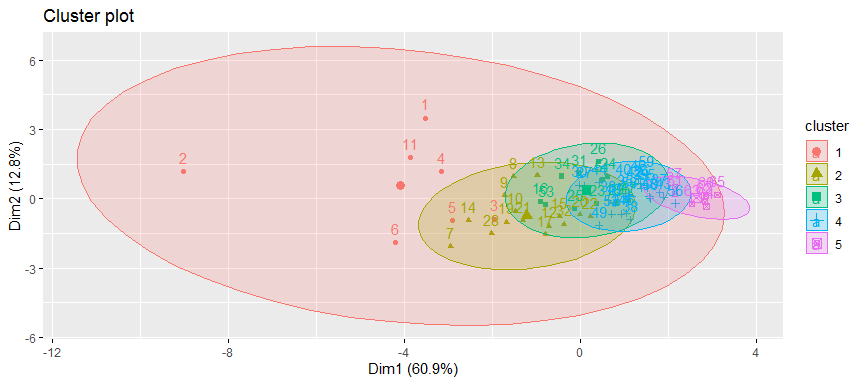
## Точность классификации: 0.9384615   
## Неправильно отнесённый элемент с максимальным расстоянием ( 2.203112 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 8 0 6342 486 261 52 24.1 1841 1  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 8 0 6342 486 261 52 24.1 1841 2



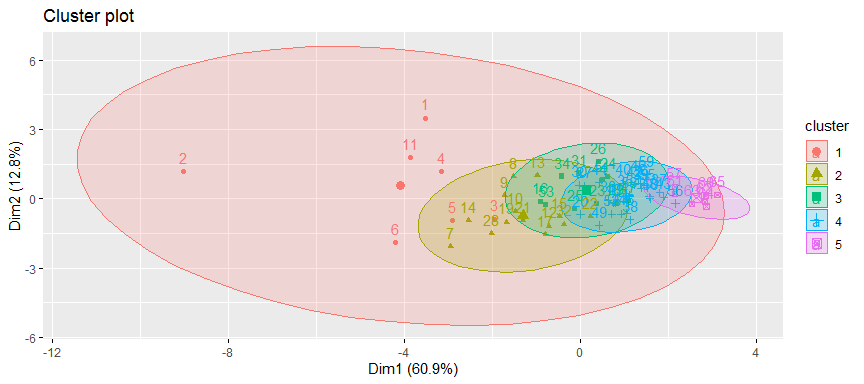
## Точность классификации: 0.9384615   
## Неправильно отнесённый элемент с максимальным расстоянием ( 2.607064 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 9 -107 5868 531 450 63 22.3 1608 1  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 9 -107 5868 531 450 63 22.3 1608 2



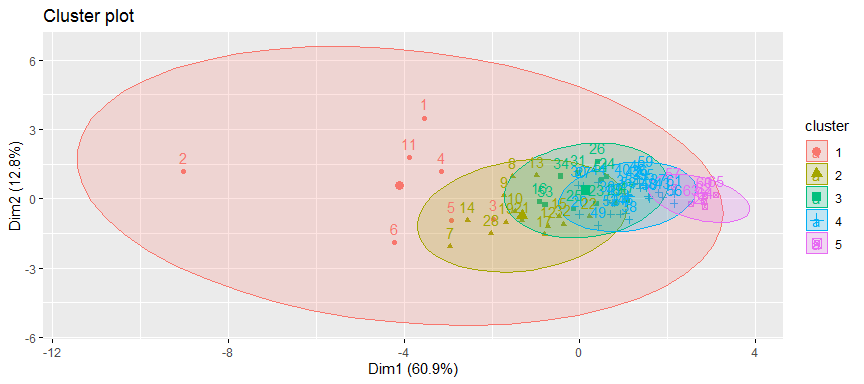
## Точность классификации: 0.9538462   
## Неправильно отнесённый элемент с максимальным расстоянием ( 2.474874 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 10 -903 6330 636 401 69 17.6 1768 1  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 10 -903 6330 636 401 69 17.6 1768 2



## Точность классификации: 0.9692308   
## Неправильно отнесённый элемент с максимальным расстоянием ( 2.046161 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 20 -94 3900 420 359 53 9.6 1034 2  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 20 -94 3900 420 359 53 9.6 1034 4



## Точность классификации: 0.9846154   
## Неправильно отнесённый элемент с максимальным расстоянием ( 2.046161 )  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 61 -855 3483 109 90 16 7.6 237 5  
## Заменяется на  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## 61 -855 3483 109 90 16 7.6 237 4



## Точность классификации: 1

**Результаты для полученной модели**:

Out\_CTab(ldadat, data$CLASS)

## Таблица неточностей "Факт/Прогноз" по обучающей выборке:   
## classified  
## group 1 2 3 4 5  
## 1 7 0 0 0 0  
## 2 0 14 0 0 0  
## 3 0 0 11 0 0  
## 4 0 0 0 27 0  
## 5 0 0 0 0 6  
## Точность классификации: 1   
## Расстояния Махалонобиса:  
## 1 2 3 4  
## 2 5.759607   
## 3 6.472220 3.248266   
## 4 9.288359 4.618247 3.208171   
## 5 12.795436 8.147295 6.595490 3.709843  
## Ошибка при скользящем контроле: 0.1384615   
## Результаты многомерного дисперсионного анализа:   
## Df Wilks approx F num Df den Df Pr(>F)   
## data$CLASS 4 0.028593 11.768 28 196.12 < 2.2e-16 \*\*\*  
## Residuals 60   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

cat("Групповые средние:\n")

## Групповые средние:

ldadat$means

## X1 X2 X3 X4 X5 X6 X7  
## 1 196.5429 9328.286 898.42857 733.4286 64.28571 26.228571 2547.8571  
## 2 -192.2286 5567.000 521.35714 452.2143 75.00000 15.278571 1384.9000  
## 3 -282.0909 5587.273 364.06364 276.5455 33.46364 12.709091 807.7273  
## 4 -490.5296 3906.000 241.04815 195.7037 29.53333 10.362963 641.8630  
## 5 -609.3667 2318.500 79.66667 55.2000 9.70000 3.283333 163.9333

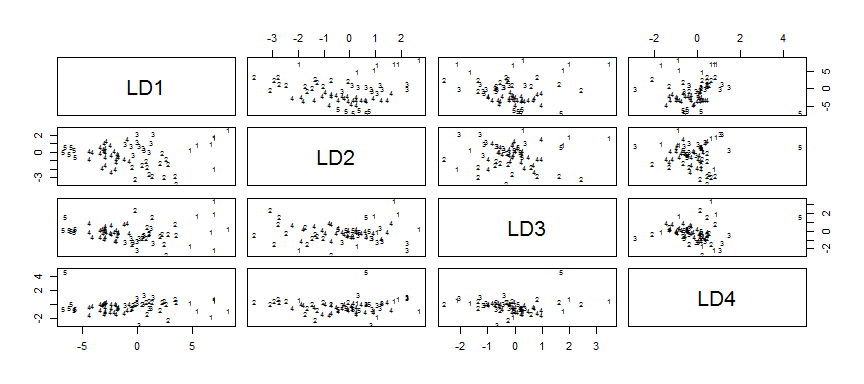
cat("Матрица дискриминантных функций:\n")

## Матрица дискриминантных функций:

ldadat$scaling

## LD1 LD2 LD3 LD4  
## X1 6.998429e-04 0.0001441873 -0.0001311974 0.0006733219  
## X2 4.517007e-05 0.0001983877 0.0003278794 0.0011001502  
## X3 2.380552e-02 0.0120910472 -0.0190093691 -0.0105312748  
## X4 -1.112724e-03 0.0005167446 0.0069277131 0.0008733516  
## X5 1.961759e-02 -0.0737548966 -0.0115067161 0.0135728516  
## X6 3.719980e-01 0.0382745899 -0.1952122703 -0.4280703443  
## X7 -6.905738e-03 -0.0033063245 0.0058332004 0.0034695279

plot(ldadat)



rf = lda(CLASS ~ ., data, method = "moment") #фиксируется модель для следующего задания

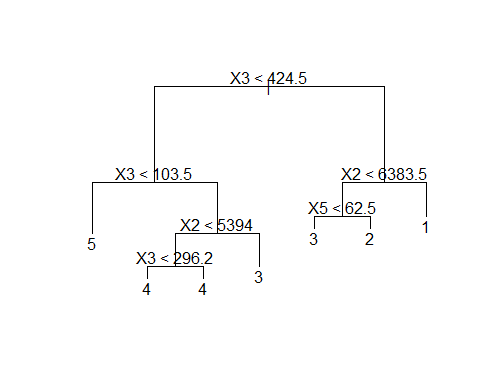
**Исправленная модель**:

data %>% tbl\_df()

## # A tibble: 65 x 8  
## X1 X2 X3 X4 X5 X6 X7 CLASS  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <fct>  
## 1 2174 9658 466 386 35 36.4 1756 1   
## 2 274 10477 2321 767 56 35.6 7884 1   
## 3 -146 6567 713 581 74 13.8 1501 1   
## 4 -338 10282 499 764 51 30.2 1466 1   
## 5 -716 9316 677 533 109 20.5 1486 1   
## 6 893. 6425 944 1390 78 13.2 1936 1   
## 7 191 5367 786 819 104 13.7 2011 2   
## 8 0 6342 486 261 52 24.1 1841 2   
## 9 -107 5868 531 450 63 22.3 1608 2   
## 10 -903 6330 636 401 69 17.6 1768 2   
## # ... with 55 more rows

Ошибка нулевая, все данные отнесены правильно. *Правило, по которому наблюдение относится в ту или иную группу* примерно сделующее (если попробовать иерархический анализ):

library(tree)  
datatree <- tree(data[, 8] ~ ., data[, -8])  
plot(datatree)  
text(datatree)



### Задание 5

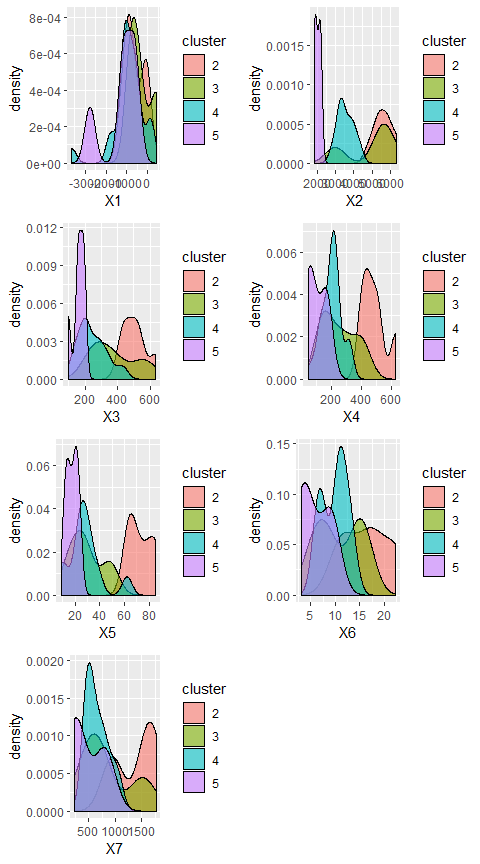
**Считаем тестовые данные и проведём их классификацию по уже построенной модели**:

data2 = data.frame(read\_excel("Приложение 3.xlsx"))  
data2 = apply(data2, 2, as.numeric) %>% as\_data\_frame()  
data2 = data2[31:80, ]  
  
cluster = predict(rf, data2)$class  
data2 = data.frame(cbind(data2, cluster))  
data2$cluster = factor(data2$cluster)  
  
data2 %>% tbl\_df()

## # A tibble: 50 x 8  
## X1 X2 X3 X4 X5 X6 X7 cluster  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <fct>   
## 1 -1080 3130 213 236 33 11.8 807 4   
## 2 403. 2969 382 274 29 5.7 728 3   
## 3 139 3264 340 316 28 6.8 711 4   
## 4 -444 3920 139 120 9 13.1 464 4   
## 5 -833 5563 271 148 15.7 8.7 426 3   
## 6 -380 5564 565 400 48 14.9 1517 3   
## 7 -790 5470 432 509 85 11.8 935 2   
## 8 -1205 3698 188. 156 21.6 10 507 4   
## 9 -751 3448 278 206 25 7.4 596 4   
## 10 -107 5868 531 450 63 22.3 1608 2   
## # ... with 40 more rows

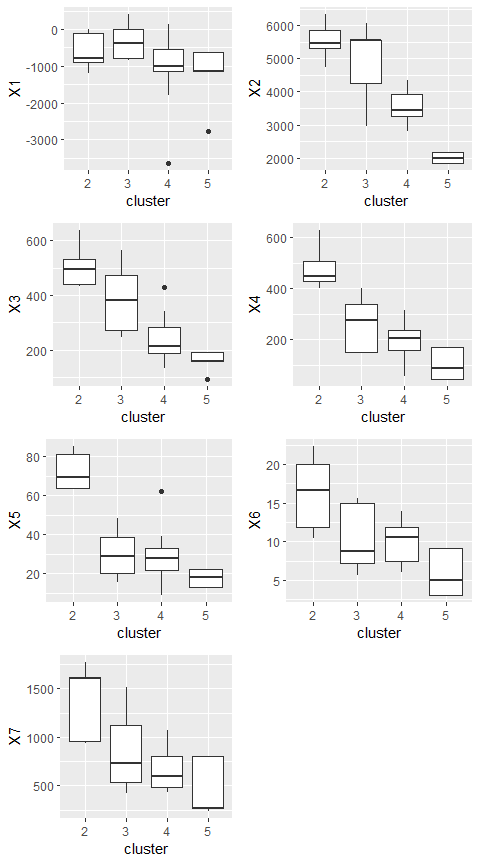
Построим **диаграммы ядерной плотности** для распределения каждой переменной X1-X7 по отнесенным группам, чтобы убедиться в правильности классификации:

library(ggplot2)  
library(ggpubr)  
  
ggarrange(ggplot(data2, aes(x = X1, fill = cluster)) + geom\_density(alpha = 0.6),   
 ggplot(data2, aes(x = X2, fill = cluster)) + geom\_density(alpha = 0.6),   
 ggplot(data2, aes(x = X3, fill = cluster)) + geom\_density(alpha = 0.6),   
 ggplot(data2, aes(x = X4, fill = cluster)) + geom\_density(alpha = 0.6),   
 ggplot(data2, aes(x = X5, fill = cluster)) + geom\_density(alpha = 0.6),   
 ggplot(data2, aes(x = X6, fill = cluster)) + geom\_density(alpha = 0.6),   
 ggplot(data2, aes(x = X7, fill = cluster)) + geom\_density(alpha = 0.6),   
 ncol = 2, nrow = 4)



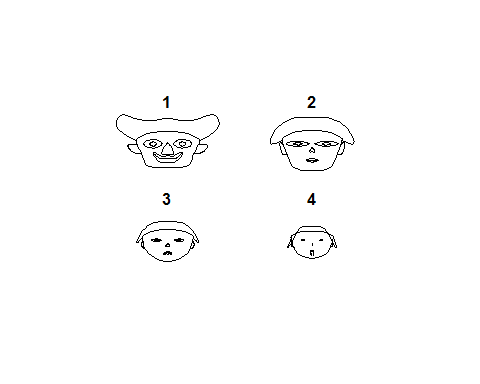
Другой вариант — **ящики с усами**:

ggarrange(ggplot(data2, aes(x = cluster, y = X1)) + geom\_boxplot(), ggplot(data2,   
 aes(x = cluster, y = X2)) + geom\_boxplot(), ggplot(data2, aes(x = cluster,   
 y = X3)) + geom\_boxplot(), ggplot(data2, aes(x = cluster, y = X4)) + geom\_boxplot(),   
 ggplot(data2, aes(x = cluster, y = X5)) + geom\_boxplot(), ggplot(data2,   
 aes(x = cluster, y = X6)) + geom\_boxplot(), ggplot(data2, aes(x = cluster,   
 y = X7)) + geom\_boxplot(), ncol = 2, nrow = 4)



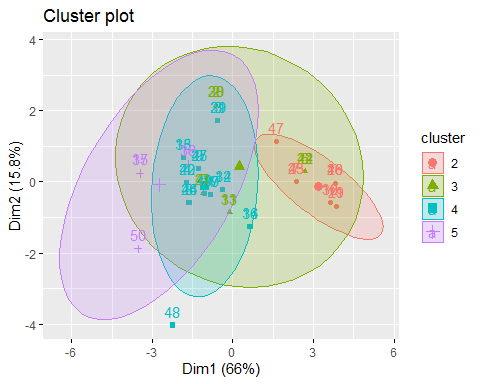
Также **лица Чернова**:

newdata = as\_data\_frame(data2) %>% group\_by(cluster) %>% summarise\_all(funs(mean))  
faces(newdata[, 2:8]) #рисуем лица



**Визуализация через главные компоненты**:

print(fviz\_cluster(list(data = data2[, 1:7], cluster = data2[, 8]), ellipse.type = "norm"))



В целом качество такое же, как в обучающей выборке.

## Временные ряды

Используются две переменные:

p1 = nchar("Дмитрий") #число букв в имени  
p2 = nchar("Пасько") #число букв в фамилии

### Задание 1

Прочитаем и обработаем **данные**:

library(readxl)  
library(dplyr)  
tab = data.frame(t(read\_xlsx("Рожь18век.xlsx")))  
names(tab) = sapply(tab[1, ], as.character) #поставить правильные названия  
tab = tab[-1, ] #удалить строку с именами  
tab = data.frame(Year = sapply(rownames(tab), as.numeric), tab)  
tab = sapply(tab, as.numeric) #факторы перевести в числа  
tab %>% tbl\_df()

## # A tibble: 62 x 19  
## Year Северный1 Северный2 Восточный1 Восточный2 ЮгoВосточный1  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1707 39 31 45 32 20  
## 2 1708 44 38 50 39 22  
## 3 1709 51 40 41 37 23  
## 4 1710 52 42 39 39 24  
## 5 1711 57 46 51 41 NA  
## 6 1712 58 47 52 42 NA  
## 7 1713 56 48 54 43 NA  
## 8 1714 1 10 4 1 NA  
## 9 1715 8 1 2 4 NA  
## 10 1716 9 8 1 2 NA  
## # ... with 52 more rows, and 13 more variables: ЮгoВосточный2 <dbl>,  
## # Волжский1 <dbl>, Волжский2 <dbl>, ЦентральноЧерноземный1 <dbl>,  
## # ЦентральноЧерноземный2 <dbl>, ЦентральноПечериозельный1 <dbl>,  
## # ЦентральноПечериозельный2 <dbl>, Украинский1 <dbl>, Украинский2 <dbl>,  
## # ЗападнаяСибирь1 <dbl>, ЗападнаяСибирь2 <dbl>,  
## # ЕвропейскаяРоссия1 <dbl>, ЕвропейскаяРоссия2 <dbl>

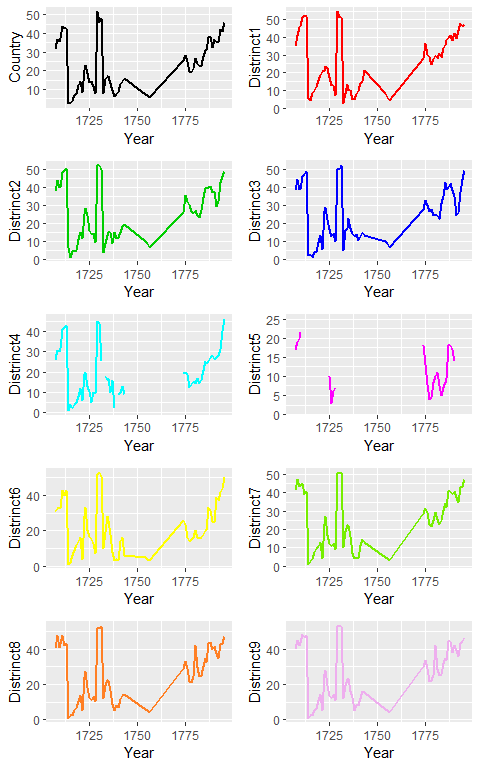
Создадим **фрейм со средними по годам для всей страны и её районов**:

tmptab = tab[, -1]  
means = list(rowMeans(tmptab, na.rm = T))  
for (i in 1:((ncol(tab) - 1)/2)) {  
 means[[i + 1]] = rowMeans(tmptab[, c(i, i + 1)], na.rm = T)  
}  
  
means = sapply(means, function(col) sapply(col, function(row) ifelse(is.nan(row),   
 NA, row))) #Заменить все NaN на NA  
means = data.frame(tab[, 1], means)  
names(means) = c("Year", "Country", "Distrinct1", "Distrinct2", "Distrinct3",   
 "Distrinct4", "Distrinct5", "Distrinct6", "Distrinct7", "Distrinct8", "Distrinct9")  
means %>% tbl\_df()

## # A tibble: 62 x 11  
## Year Country Distrinct1 Distrinct2 Distrinct3 Distrinct4 Distrinct5  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1707 32.1 35 38 38.5 26 17   
## 2 1708 36.8 41 44 44.5 30.5 19   
## 3 1709 36 45.5 40.5 39 30 20   
## 4 1710 37.5 47 40.5 39 31.5 21.5  
## 5 1711 43.6 51.5 48.5 46 41 NA   
## 6 1712 42.7 52.5 49.5 47 42 NA   
## 7 1713 42.7 52 51 48.5 43 NA   
## 8 1714 2.47 5.5 7 2.5 1 NA   
## 9 1715 2.67 4.5 1.5 3 4 NA   
## 10 1716 4.15 8.5 4.5 1.5 2 NA   
## # ... with 52 more rows, and 4 more variables: Distrinct6 <dbl>,  
## # Distrinct7 <dbl>, Distrinct8 <dbl>, Distrinct9 <dbl>

Визуализация полученных значений показывает, что *в целом цена ведёт себя одинаково по всем регионам*:

library(ggplot2)  
g1 = ggplot(means, aes(x = Year)) + geom\_line(aes(y = Country), size = 1)  
g2 = ggplot(means, aes(x = Year)) + geom\_line(aes(y = Distrinct1), size = 1,   
 col = 2)  
g3 = ggplot(means, aes(x = Year)) + geom\_line(aes(y = Distrinct2), size = 1,   
 col = 3)  
g4 = ggplot(means, aes(x = Year)) + geom\_line(aes(y = Distrinct3), size = 1,   
 col = 4)  
g5 = ggplot(means, aes(x = Year)) + geom\_line(aes(y = Distrinct4), size = 1,   
 col = 5)  
g6 = ggplot(means, aes(x = Year)) + geom\_line(aes(y = Distrinct5), size = 1,   
 col = 6)  
g7 = ggplot(means, aes(x = Year)) + geom\_line(aes(y = Distrinct6), size = 1,   
 col = 7)  
g8 = ggplot(means, aes(x = Year)) + geom\_line(aes(y = Distrinct7), size = 1,   
 col = "chartreuse2")  
g9 = ggplot(means, aes(x = Year)) + geom\_line(aes(y = Distrinct8), size = 1,   
 col = "chocolate1")  
g10 = ggplot(means, aes(x = Year)) + geom\_line(aes(y = Distrinct9), size = 1,   
 col = "plum2")  
library(ggpubr)  
ggarrange(g1, g2, g3, g4, g5, g6, g7, g8, g9, g10, nrow = 5, ncol = 2)



Зададим **новый временной ряд**:

price1 = c(40 + p1, 43 + p1, 40, 80, 74, 40 + p2, 55 + p2, 42 + p2, 42, 50,   
 40 + p2, 43, 43, 35 + p1, 40 + p1, 30, 36 + p1, 50, 30 + p1, 29, 45 + p1,   
 40, 42, 40, 36, 50, 30 + p1, 24 + p2, 25 + p2, 40, 32 + p1, 30, 20, 30,   
 25, 32 + p2)  
cat("Длина вектора:", length(price1), "\n")

## Длина вектора: 36

summary(price1) #минимальные характеристики

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 20.00 36.75 42.00 42.44 47.25 80.00

*Тест Стьюдента* для среднего значения (42.4444444) показывает **значимое отличие среднего ряда от нуля (p-value очень близкое к нулю)**:

(ts = t.test(price1, conf.level = 0.95)) #тест Стьюдента для среднего

##   
## One Sample t-test  
##   
## data: price1  
## t = 21.17, df = 35, p-value < 2.2e-16  
## alternative hypothesis: true mean is not equal to 0  
## 95 percent confidence interval:  
## 38.37422 46.51467  
## sample estimates:  
## mean of x   
## 42.44444

cat("Доверительный интервал: ", ts$conf.int, "\n")

## Доверительный интервал: 38.37422 46.51467

cat("Стандартная ошибка среднего: ", ts$stderr, "\n")

## Стандартная ошибка среднего: 2.004932

а **низкий коэффициент вариации говорит об однородности выборки**

vart = sd(price1)/mean(price1) \* 100  
cat("Коэффициент вариации равен", vart, "%\n")

## Коэффициент вариации равен 28.34197 %

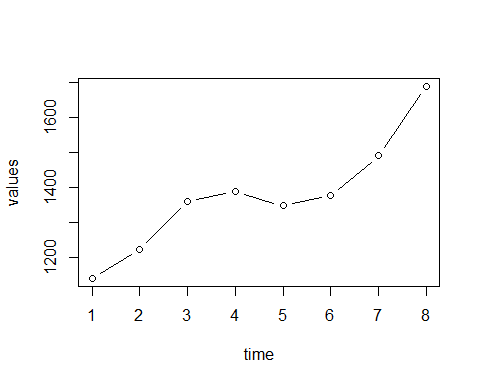
# так как коэффициент вариации < 30%, выборка достаточно однородная

**Замечание:** в проведённом тесте Стьюдента число степеней свободы было на 1 (а не на 2) меньше числа наблюдений, так как при тесте использовалась одна выборка. Можно считать, что исходная выборка сравнивается с выборкой из единственного элемента 0, поэтому число степеней свободы всего на 1 меньше числа наблюдений. Так же определяется число степеней свободы, например, [здесь](https://statpsy.ru/t-student/primer-t-test-single/).

### Задание 3

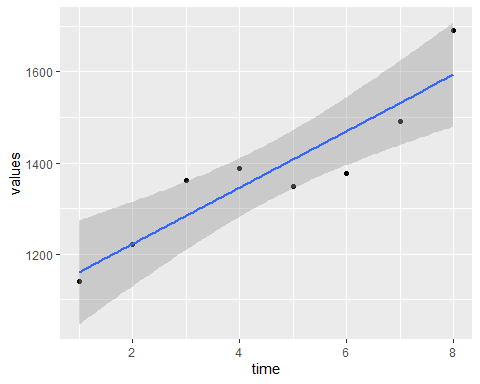
Создадим и визуализируем **временной ряд**:

library(ggplot2)  
  
yt = c(1133 + p1, 1222, 1354 + p1, 1389, 1342 + p2, 1377, 1491, 1684 + p2)  
data = data.frame(time = 1:length(yt), values = yt)  
plot(data, type = "b")



В целом, здесь наблюдается линейная составляющая. Построим линейную модель и **регрессионную прямую**:

fit = lm(values ~ time, data) #создание модели  
ggplot(data, aes(x = time, y = values)) + geom\_point() + geom\_smooth(method = lm)



Посмотрим **информацию о модели**:

summary(fit) #информация о модели

##   
## Call:  
## lm(formula = values ~ time, data = data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -93.14 -45.86 -10.46 51.20 96.00   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1098.57 56.30 19.513 1.17e-06 \*\*\*  
## time 61.93 11.15 5.555 0.00144 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 72.25 on 6 degrees of freedom  
## Multiple R-squared: 0.8372, Adjusted R-squared: 0.8101   
## F-statistic: 30.85 on 1 and 6 DF, p-value: 0.00144

confint(fit, level = 0.95) #доверительные интервалы

## 2.5 % 97.5 %  
## (Intercept) 960.81185 1236.33101  
## time 34.64811 89.20903

Самое важное здесь — модель описывает более 80% дисперсий (**Adjusted R-squared**), каждый её коэффициент (**Pr(>|t|)**) и она сама (**p-value** в самом низу, в строке с **F-statistic**) статистически значимы (поскольку p-value меньше 0.05). **Estimate** — это коэффициенты модели, значит сама регресионная прямая имеет вид

Коэффициент при показывает, на сколько в среднем увеличится при изменении на единицу. Дисперсионный анализ можно выполнить и отдельно с тем же результатом:

anova(fit)

## Analysis of Variance Table  
##   
## Response: values  
## Df Sum Sq Mean Sq F value Pr(>F)   
## time 1 161076 161076 30.854 0.00144 \*\*  
## Residuals 6 31323 5221   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Теперь выполним [прогноз для среднего и индивидульного значений](http://www.sthda.com/english/articles/40-regression-analysis/166-predict-in-r-model-predictions-and-confidence-intervals/) при :

# прогноз среднего  
predict(fit, data.frame(time = c(9)), se.fit = TRUE, interval = "confidence",   
 level = 0.95)$fit

## fit lwr upr  
## 1 1655.929 1518.169 1793.688

# прогноз индивидуального  
predict(fit, data.frame(time = c(9)), se.fit = TRUE, interval = "prediction",   
 level = 0.95)$fit

## fit lwr upr  
## 1 1655.929 1431.797 1880.06

Здесь первое число — само предсказанное значение, последующие — границы доверительного интервала.Итак, само предсказанное значение равно 1655.929, доверительный интервал для среднего: [1518.169,1793.688], для индивидуального: [1431.797,1880.06].

### Задание 4

Считаем **набор данных** и проведём некоторую обработку:

library(readxl)  
data = data.frame(read\_xlsx("РожьВсеГода.xlsx"))  
data[, -1] = apply(data[, -1], 2, as.numeric) #перевести в числа все строки  
y = t(data[, -1]) #транспонирование для удобства  
  
# получить массив лет  
ns = rownames(y)  
x = sapply(ns, function(s) as.numeric(substr(s, 2, nchar(s))))  
  
library(mice) #обработать пустые значения  
imp = mice(y, seed = 11)  
y = complete(imp, action = 1)  
  
df = data.frame(x = x, y = y[, 2]) #объединить данные в фрейм  
  
# print(df[sort(sample(1:nrow(df),13)),2,drop=FALSE]) #вывести 13 случайных  
# строк

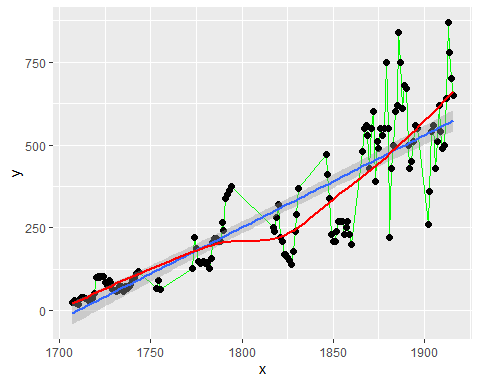
Получаем:

df %>% tbl\_df()

## # A tibble: 138 x 2  
## x y  
## <dbl> <dbl>  
## 1 1707 26  
## 2 1708 32  
## 3 1709 23  
## 4 1710 20  
## 5 1711 35  
## 6 1712 39  
## 7 1713 41  
## 8 1714 38  
## 9 1715 34  
## 10 1716 31  
## # ... with 128 more rows

При этом пропущенные значения были обработаны по алгоритму [MICE](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.169.5745&rep=rep1&type=pdf), **для дальнейшего исследования был взят второй район**. Визуализировав данные по нему

library(ggplot2)  
ggplot(df, aes(x = x, y = y)) + geom\_line(col = "green") + geom\_point(size = 2) +   
 geom\_smooth(method = lm) + geom\_smooth(se = F, col = "red")



приходим к выводу, что в целом поведение цен можно описать линейной моделью, однако для обычной линейной модели здесь явно не будет выполняться требование гомоскедастичности. Попробуем разные **преобразования данных** (качество модели определяется величиной Adjusted R-squared):

summary(lm(y ~ x, df))

##   
## Call:  
## lm(formula = y ~ x, data = df)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -274.50 -52.12 7.23 45.45 349.99   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -4754.7565 241.9835 -19.65 <2e-16 \*\*\*  
## x 2.7809 0.1333 20.87 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 105.9 on 136 degrees of freedom  
## Multiple R-squared: 0.762, Adjusted R-squared: 0.7603   
## F-statistic: 435.5 on 1 and 136 DF, p-value: < 2.2e-16

summary(lm(sqrt(y) ~ x, df))

##   
## Call:  
## lm(formula = sqrt(y) ~ x, data = df)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -7.370 -1.454 -0.163 1.628 6.901   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.444e+02 6.017e+00 -24.00 <2e-16 \*\*\*  
## x 8.829e-02 3.314e-03 26.65 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.633 on 136 degrees of freedom  
## Multiple R-squared: 0.8392, Adjusted R-squared: 0.8381   
## F-statistic: 710 on 1 and 136 DF, p-value: < 2.2e-16

summary(lm(log(y) ~ x, df))

##   
## Call:  
## lm(formula = log(y) ~ x, data = df)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.00146 -0.26602 -0.00776 0.26103 0.87028   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.762e+01 8.398e-01 -20.99 <2e-16 \*\*\*  
## x 1.264e-02 4.625e-04 27.34 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3675 on 136 degrees of freedom  
## Multiple R-squared: 0.8461, Adjusted R-squared: 0.8449   
## F-statistic: 747.6 on 1 and 136 DF, p-value: < 2.2e-16

summary(lm(log(y) ~ log(x), df))

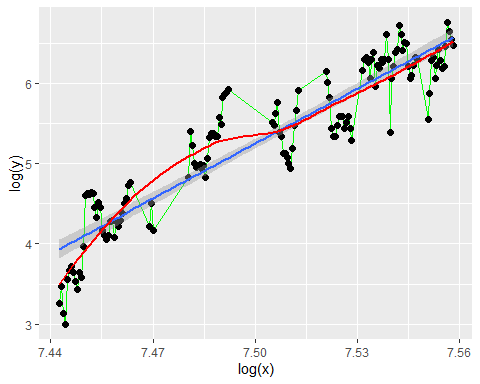
##   
## Call:  
## lm(formula = log(y) ~ log(x), data = df)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.97970 -0.26965 -0.00275 0.25610 0.85541   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -166.5906 6.2023 -26.86 <2e-16 \*\*\*  
## log(x) 22.9125 0.8266 27.72 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3633 on 136 degrees of freedom  
## Multiple R-squared: 0.8496, Adjusted R-squared: 0.8485   
## F-statistic: 768.3 on 1 and 136 DF, p-value: < 2.2e-16

Видно, что *модель с логарифмами описывает почти 85% дисперсий*, поэтому в дальнейшем будем использовать её. Эта модель имеет вид:

а иначе:

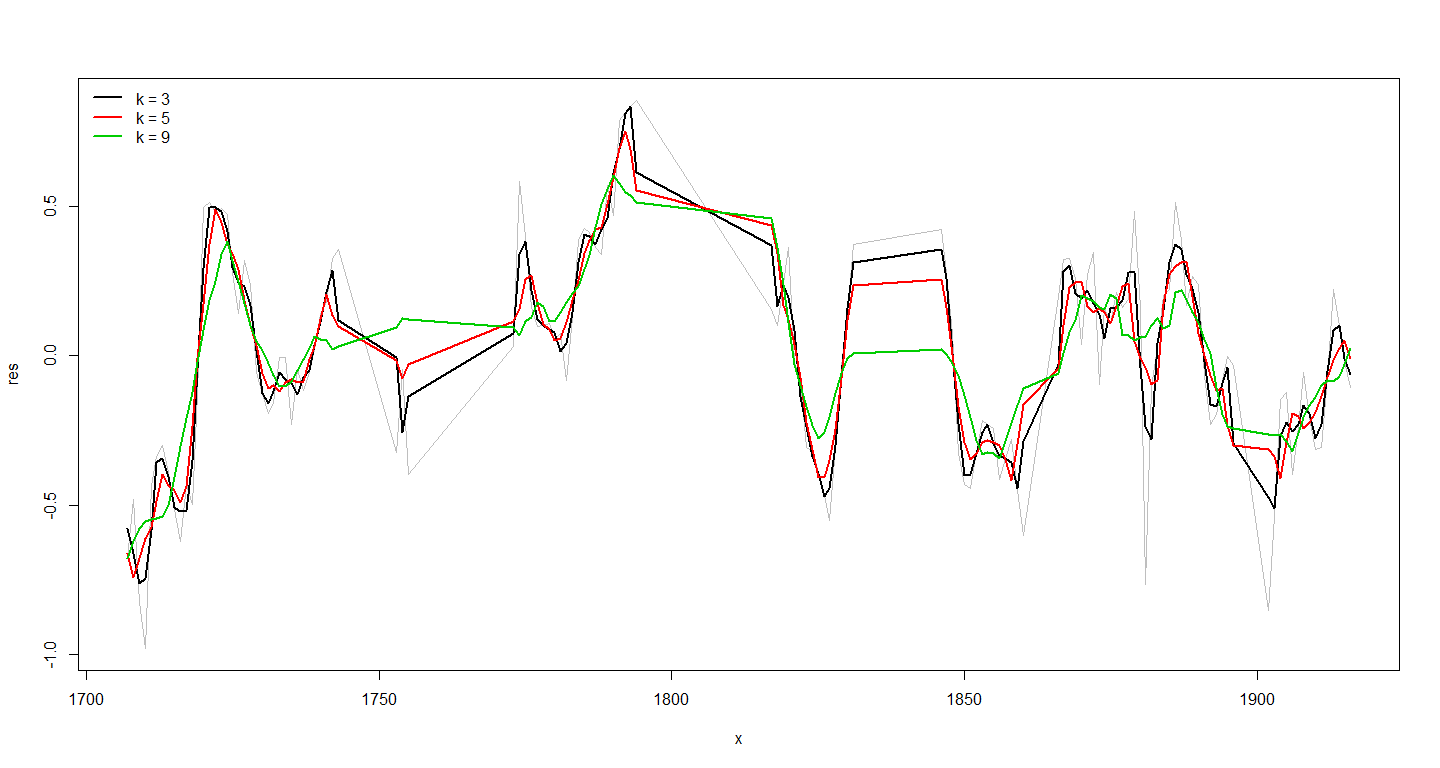
Построим тот же график:

ggplot(df, aes(x = log(x), y = log(y))) + geom\_line(col = "green") + geom\_point(size = 2) +   
 geom\_smooth(method = lm) + geom\_smooth(se = F, col = "red")



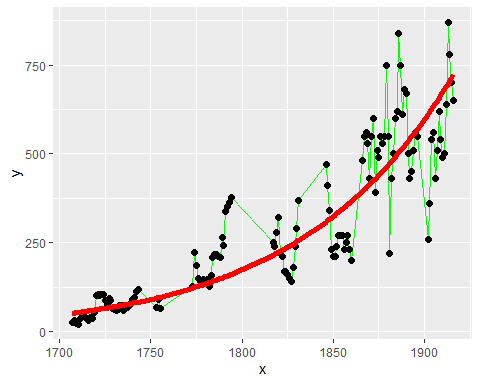
**Для остатков полученной модели проведём сглаживание методом k-средних**:

mt = lm(log(y) ~ log(x), df)  
res = mt$residuals  
# скользящее среднее  
library(caTools)  
k = c(3, 5, 9)  
plot(x, res, type = "l", col = "grey")  
for (i in 1:length(k)) {  
 lines(x, runmean(res, k[i]), col = i, lwd = 2)  
}  
  
legend("topleft", c(paste("k =", k)), col = 1:length(k), bty = "n", lwd = 2)



Построим график этой модели в исходной системе координат:

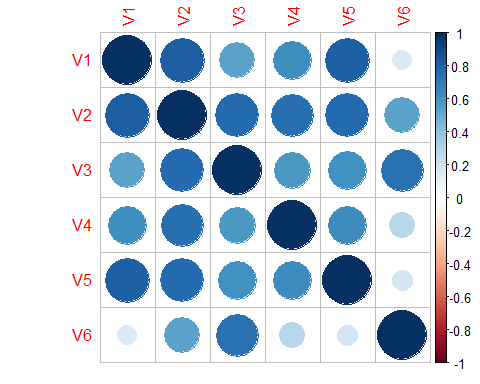
ggplot(df, aes(x = x, y = y)) + geom\_line(col = "green") + geom\_point(size = 2) +   
 geom\_line(aes(x = x, y = exp(-166.5906) \* x^22.9125), col = "red", size = 2)



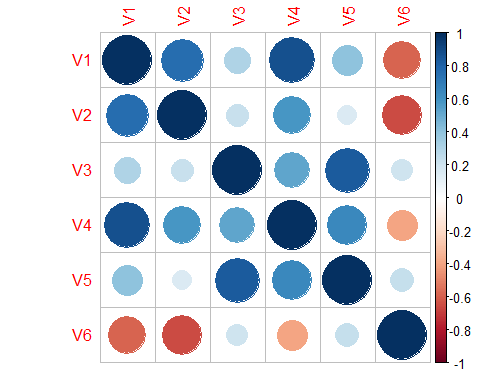
И последнее: **визуализируем корреляции между всеми исходными районами за последние 60 лет наблюдений по отрезкам в 20 лет**:

nn = 20  
  
for (i in seq(length(x) - 60, length(x) - nn, nn)) {  
 tmp = i:(i + nn - 1)  
 cat("Times:", x[tmp], "\n")  
 data = y[tmp, ]  
 cormatrix = cor(data)  
 cat("Матрица корреляций:\n")  
 print(cormatrix)  
 lower = abs(cormatrix[lower.tri(cormatrix)])  
 cat("Статистика по треугольнику корреляционной матрицы \n")  
 print(summary(lower[lower != 0]))  
 corrplot(cormatrix)  
}

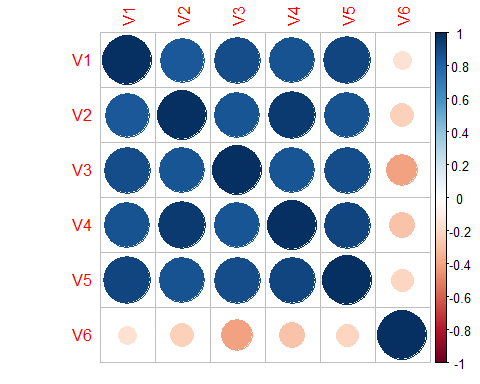
## Times: 1846 1847 1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860 1866 1867 1868 1869 1870   
## Матрица корреляций:  
## V1 V2 V3 V4 V5 V6  
## V1 1.0000000 0.8291448 0.5366430 0.6129233 0.8161189 0.1555005  
## V2 0.8291448 1.0000000 0.7717488 0.7558893 0.7731484 0.5312709  
## V3 0.5366430 0.7717488 1.0000000 0.5726800 0.6073105 0.7487677  
## V4 0.6129233 0.7558893 0.5726800 1.0000000 0.6249769 0.2797815  
## V5 0.8161189 0.7731484 0.6073105 0.6249769 1.0000000 0.1852402  
## V6 0.1555005 0.5312709 0.7487677 0.2797815 0.1852402 1.0000000  
## Статистика по треугольнику корреляционной матрицы   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1555 0.5340 0.6129 0.5867 0.7638 0.8291



## Times: 1871 1872 1873 1874 1875 1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889 1890   
## Матрица корреляций:  
## V1 V2 V3 V4 V5 V6  
## V1 1.0000000 0.7649144 0.3027825 0.8765815 0.4005356 -0.5869152  
## V2 0.7649144 1.0000000 0.2207233 0.5826532 0.1561994 -0.6556954  
## V3 0.3027825 0.2207233 1.0000000 0.5261564 0.8357040 0.2055036  
## V4 0.8765815 0.5826532 0.5261564 1.0000000 0.6427869 -0.3945789  
## V5 0.4005356 0.1561994 0.8357040 0.6427869 1.0000000 0.2367902  
## V6 -0.5869152 -0.6556954 0.2055036 -0.3945789 0.2367902 1.0000000  
## Статистика по треугольнику корреляционной матрицы   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1562 0.2698 0.5262 0.4926 0.6492 0.8766



## Times: 1891 1892 1893 1894 1895 1896 1902 1903 1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915   
## Матрица корреляций:  
## V1 V2 V3 V4 V5 V6  
## V1 1.0000000 0.8424524 0.8816869 0.8683768 0.9197496 -0.1501039  
## V2 0.8424524 1.0000000 0.8579754 0.9580594 0.8652964 -0.2309437  
## V3 0.8816869 0.8579754 1.0000000 0.8514249 0.8895560 -0.4079528  
## V4 0.8683768 0.9580594 0.8514249 1.0000000 0.9179652 -0.2816789  
## V5 0.9197496 0.8652964 0.8895560 0.9179652 1.0000000 -0.2186937  
## V6 -0.1501039 -0.2309437 -0.4079528 -0.2816789 -0.2186937 1.0000000  
## Статистика по треугольнику корреляционной матрицы   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.1501 0.3448 0.8580 0.6761 0.8856 0.9581

 Наглядно видно, что ближе к концу наблюдений корреляция между переменными возрастает. Проблемы с шестой переменной связаны с тем, что для неё было очень много пропущенных значений.