

Программные системы статистического анализа

Тимофеева А.Ю.

к.э.н., доцент кафедры ТПИ

a.timofeeva@corp.nstu.ru

8 сентября, 2021

Балльно-рейтинговая система

Практические работы, в том числе	60 баллов
– Реализация на R	10 баллов
– Модельный пример	10 баллов
– Реальный пример	10 баллов
– Выводы, интерпретация	10 баллов
– Оформление отчета	10 баллов
– Сравнение с другими языками	10 баллов
Экзамен	40 баллов

- Практические работы выполняются в индивидуально.
Защищаются в устной форме.
- Экзамен в виде теста.

Справочная литература на русском языке

- ① Тимофеева А. Ю., Хайленко Е.А. Вероятностные основы методов и алгоритмов анализа данных. - Новосибирск : Изд-во НГТУ, 2020.
- ② Тимофеева А. Ю. Теория вероятностей и математическая статистика. - Новосибирск : Изд-во НГТУ, 2017. - Ч. 2.
- ③ Шипунов А. Б. и др. Наглядная статистика. Используем R! - М.: ДМК Пресс. - 2012.
- ④ Зарядов И. С. Введение в статистический пакет R: типы переменных, структуры данных, чтение и запись информации, графика. - М.: РУДН. - 2010.
- ⑤ Мастицкий С.Э., Шитиков В.К. (2014) Статистический анализ и визуализация данных с помощью R. - Электронная книга, адрес доступа: <http://r-analytics.blogspot.com>
- ⑥ Храмов Д.А. Сбор данных в интернете на языке R. - М.: ДМК Пресс, 2017.

Содержание курса

① Основы программирования в R

- Типы и структуры данных
- Чтение, обработка, запись информации
- Векторизация
- Функции
- Циклы и условные операторы

② Анализ взаимосвязей

③ Базовая и расширенная графика

④ Анализ текстовых данных

Лицензионные статистические пакеты

- * EViews
- * SAS
- * SPSS
- * Stata
- * STATISTICA
- * S-PLUS

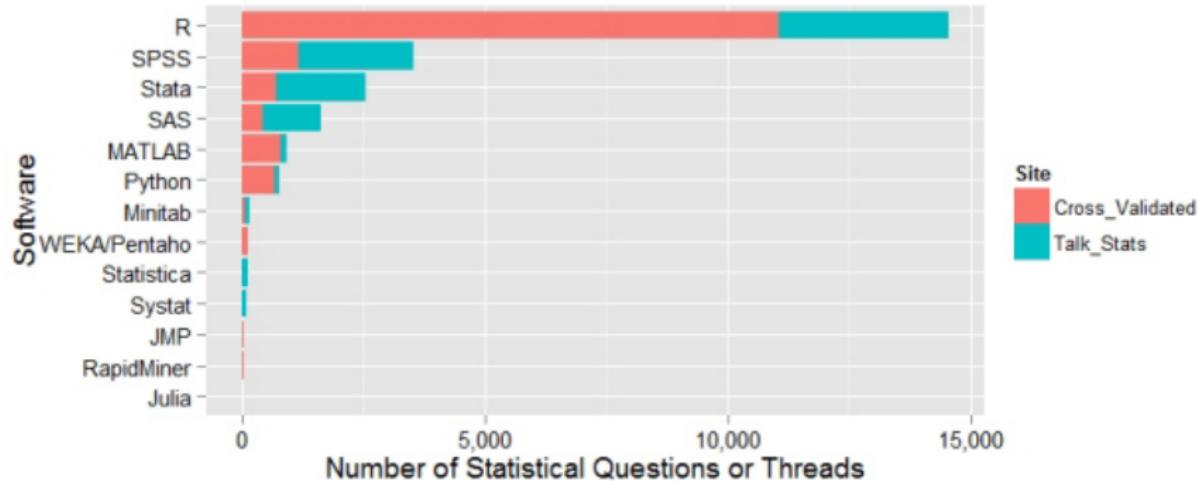
Свободное программное обеспечение

- * Gretl (Regression, Econometrics and Time-series Library)
- * R (programming language)

Активность обсуждения ПО для статистической обработки на специализированных форумах

<http://r4stats.com/blog/>

Posted on October 19, 2015 by Bob Muenchen



Графический интерфейс для R

- R Console (только командная строка)
- R Commander (полезен для образовательных целей)
- RStudio (Windows, Linux, Mac OS X, web-интерфейс)
- Rattle (R Analytical Tool To Learn Easily - среда для разглядывания данных)
- JGR (Java GUI для R)
- rpanel (пакет для создания простых графических приложений)

Преимущества среды R

Свободное ПО

Можно скачать с сайта r-project.org

Универсальность использования

- Обработка данных
- Математические вычисления
- Оптимизация
- Моделирование
- Графическое отображение данных
- Создание собственных пакетов и web-приложений

Преимущества среды R

Совместимость

- Считываемые форматы данных
- Форматы вывода
- Работа с \LaTeX
- Работа в интегрированных средах разработки

Широкие графические возможности

- Полная настройка параметров вывода в графическое устройство
- Трехмерная графика
- Анимация

Справочная информация

Доступны подробная справка, документация, справочная и научная литература, форумы, блоги, сообщества в социальных сетях

Недостатки R

Работа с командной строкой

Требуется знание основных команд и синтаксиса языка

Документация

Очень мало литературы на русском языке

Ввод данных

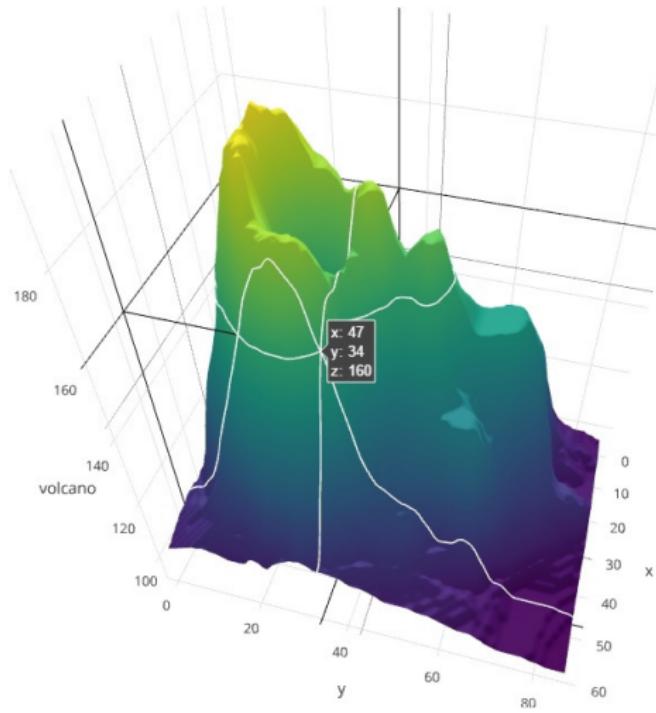
В отличие от статистических пакетов, отображающих таблицы данных, (SPSS, STATISTICA) в R неудобно непосредственно вводить данные, имеет смысл только считывать их. Поэтому данные опросов вводятся в другие программы (статистические пакеты, Excel)

Особенности языка R

- Интерпретатор
- Объектно-ориентированный
- Векторизованные функции
- Динамическая типизация
- Возможность создания пользовательских функций
- Наличие операторов цикла, условных операторов

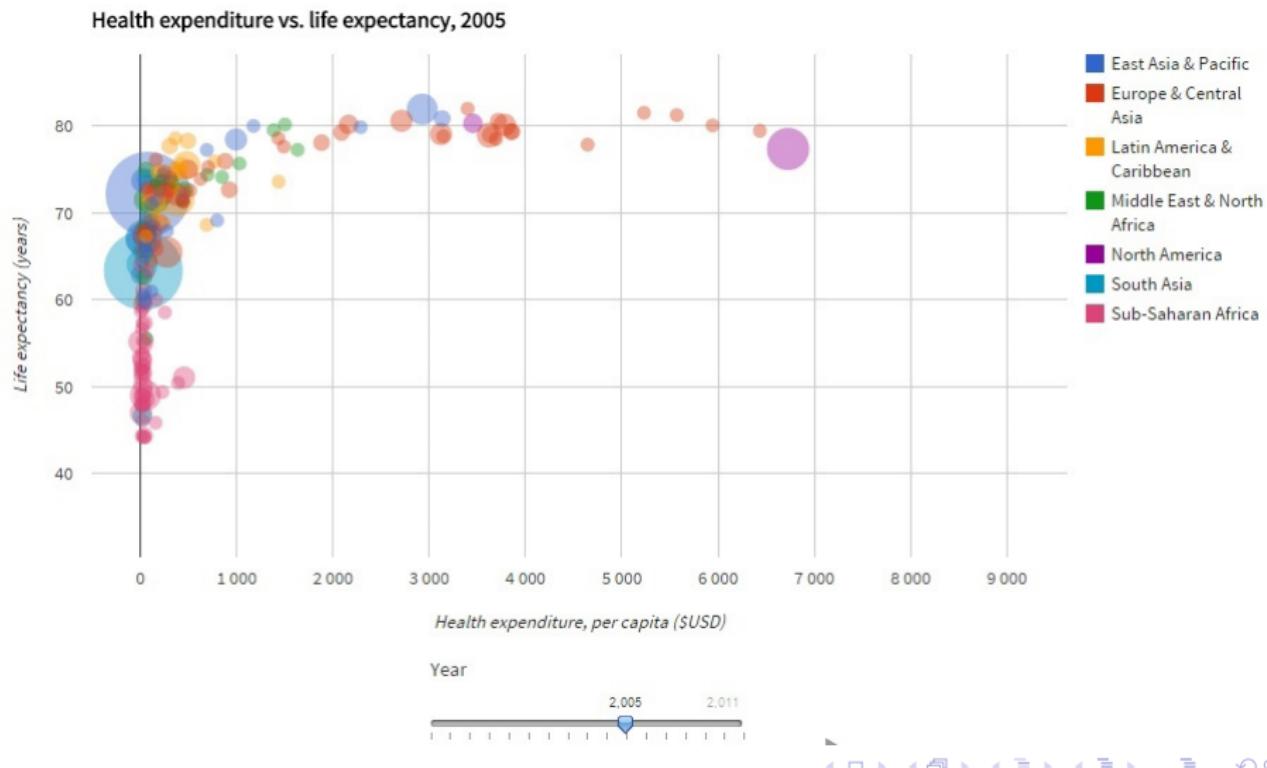
Примеры использования R

<https://plot.ly/r/>



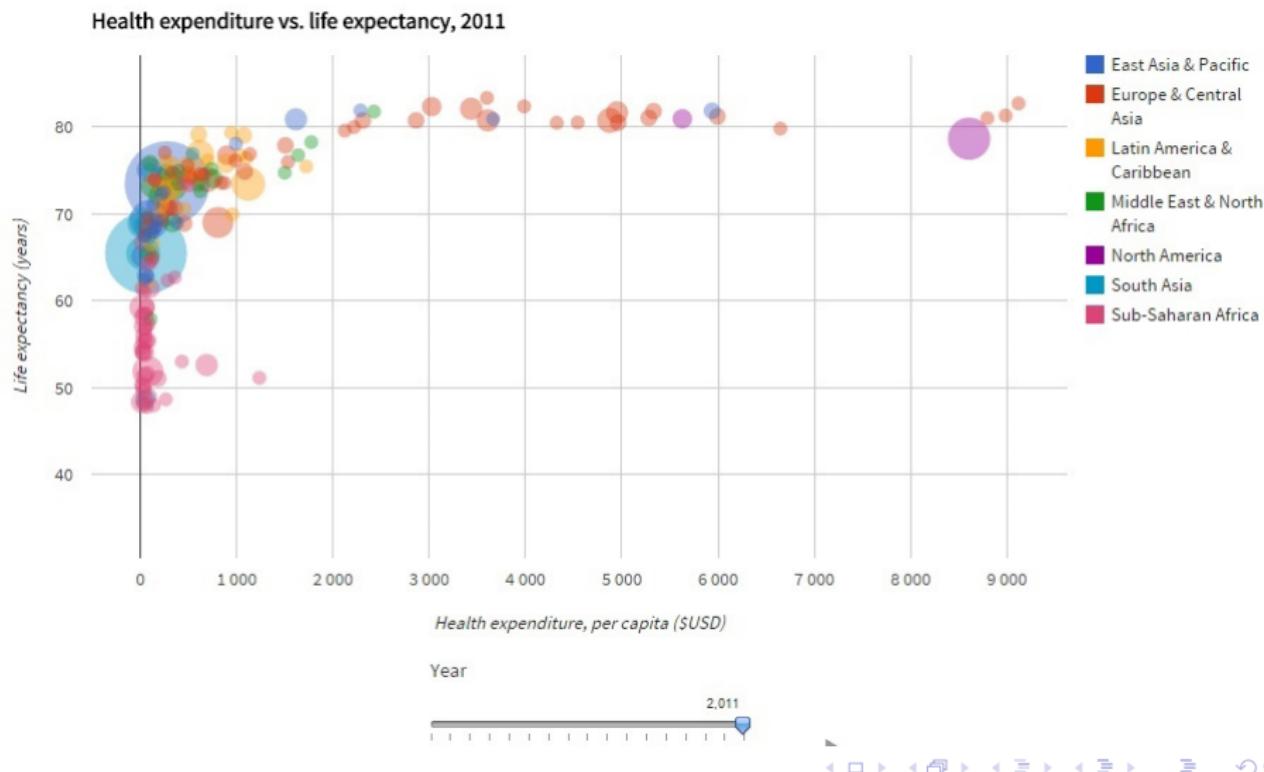
Примеры использования R

<http://shiny.rstudio.com>



Примеры использования R

<http://shiny.rstudio.com>



Примеры использования R

Анализ социальных сетей (R for Social Network Analysis)



Примеры использования R

tm Package for Text Mining

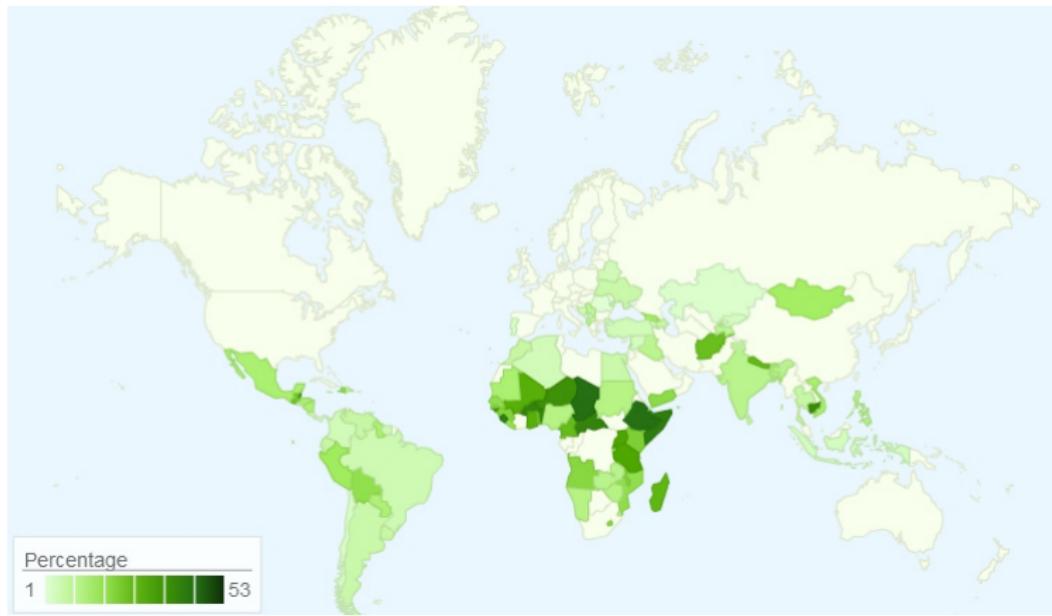
wordcloud Package for Word cloud



Примеры использования R

Интерактивные карты (геоинформационные сервисы)

<http://www.r-bloggers.com/opendata-r-google-easy-maps/>



Импорт данных

Базовые считываемые форматы

- .txt
- .csv

Внешние считываемые форматы с помощью library{foreign}

- .sav (SPSS)
- .dta (Stata)
- .dbf (database file)

Таблица данных

Таблица "Объект-свойство"(data frame) - список, каждый элемент которого имеет одну и ту же размерность, равную числу строк в таблице. Элементы списка могут быть разнотипными. К такой таблице можно обращаться как к двумерному массиву.

Типы переменных и значений

- Числовой (numeric, integer)
- Текстовые данные преобразуются в фактор с числом уровней, равным уникальным текстовым значениям, и автоматической кодировкой (проблем по считываем кириллического текста не возникало)
- Пропущенные значения (пустые ячейки) в числовых переменных заменяются на NA (not available), для текстовых может создаваться пустой уровень, поэтому лучше заполнять пропуски NA в исходном документе

Исходные данные в формате csv

1	A	B	C	D	E	F	G	H	I	J
1	latitude	longitude	rooms	square1	square2	square3	stage	stages	type	price
2	54,9878	83,0452		3	78	47	12	4	17 к	4200
3	54,9815	82,87		3	97	62	14	9	11 к	7500
4	55,0392	82,8954		2	72	36	15	10	10 к	4800
5	55,0332	82,9045		3	110	60	30	4	10 к	8900
6	55,0324	82,9348		2	103,4	48	24	11	19 к	6721
7	55,0546	82,9254		2	42	30	NA	6	12 к	4200
8	54,8516	83,0982		4	140	80	15	9	12 к	12250
9	55,0382	82,8952		2	45	29	6	5	5 к	3095
10	55,0356	82,9222	6+		260	NA	NA	16	17 к	41000
11	55,0188	82,946		1	42	20	12	10	16 к	4200
12	55,0186	82,9298		3	87	48	11	12	13 к	6000
13	54,9743	82,8854		2	43	28	6	1	5 к	2100
14	55,0168	82,935	6+		501	NA	NA	4	4 к	95000
15	55,0231	82,9245		3	174	161	10	2	21 к	15660
16	55,0177	82,9378		3	143	78	21	6	17 к	11000
17	55,0264	82,9355		2	67	34	12	3	9 к	5749
18	54,8621	82,965		1	39	19	6	1	9 к	1900

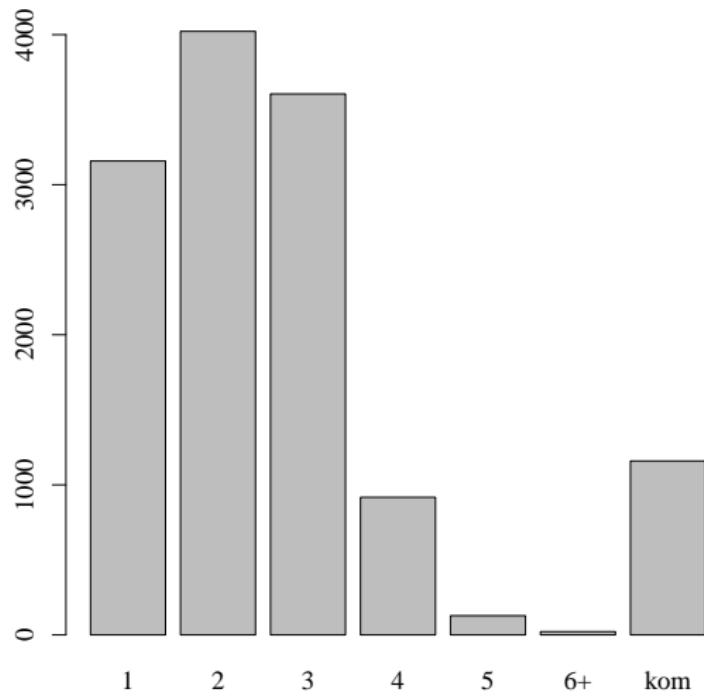
Считывание и просмотр данных в R

```
dd=read.csv2('Example_data.csv',stringsAsFactors=TRUE)
str(dd)

'data.frame': 13011 obs. of 10 variables:
 $ latitude : num 55 55 55 55 55 ...
 $ longitude: num 83 82.9 82.9 82.9 82.9 ...
 $ rooms     : Factor w/ 7 levels "1","2","3","4",...: 3 3 2 3 2 2 4 2 6 1 ...
 $ square1   : num 78 97 72 110 103 ...
 $ square2   : num 47 62 36 60 48 30 80 29 NA 20 ...
 $ square3   : num 12 14 15 30 24 NA 15 6 NA 12 ...
 $ stage     : int 4 9 10 4 11 6 9 5 16 10 ...
 $ stages    : int 17 11 10 10 19 12 12 5 17 16 ...
 $ type      : Factor w/ 15 levels "", "б", "бмк", "бт", ...: 7 7 7 7 7 7 7 7 7 7 ...
 $ price     : int 4200 7500 4800 8900 6721 4200 12250 3095 41000 4200 ...
```

Таблица частот

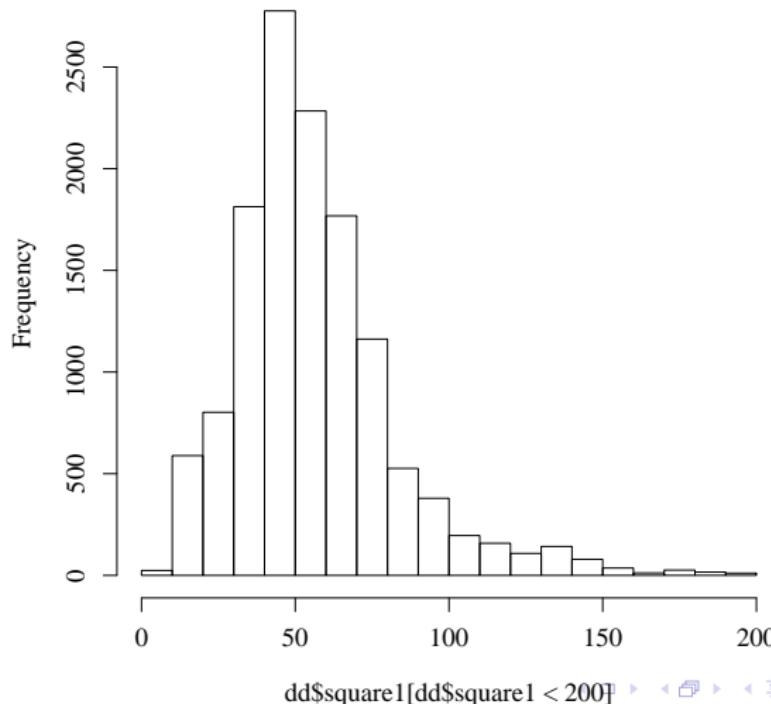
plot(dd\$rooms)



Гистограмма

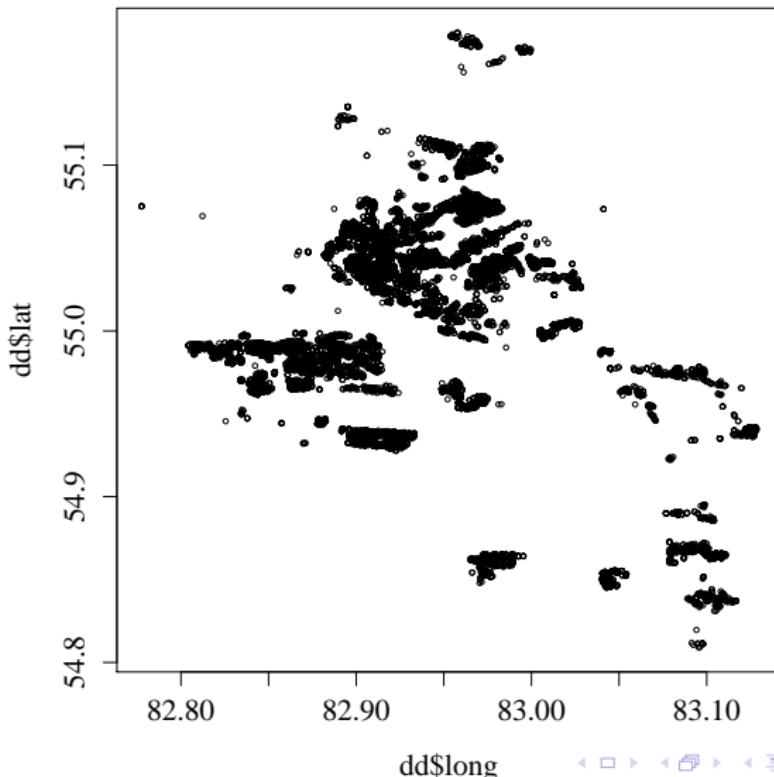
```
hist(dd$square1[dd$square1<200])
```

Histogram of dd\$square1[dd\$square1 < 200]



Корреляционное поле

```
plot(dd$long, dd$lat)
```

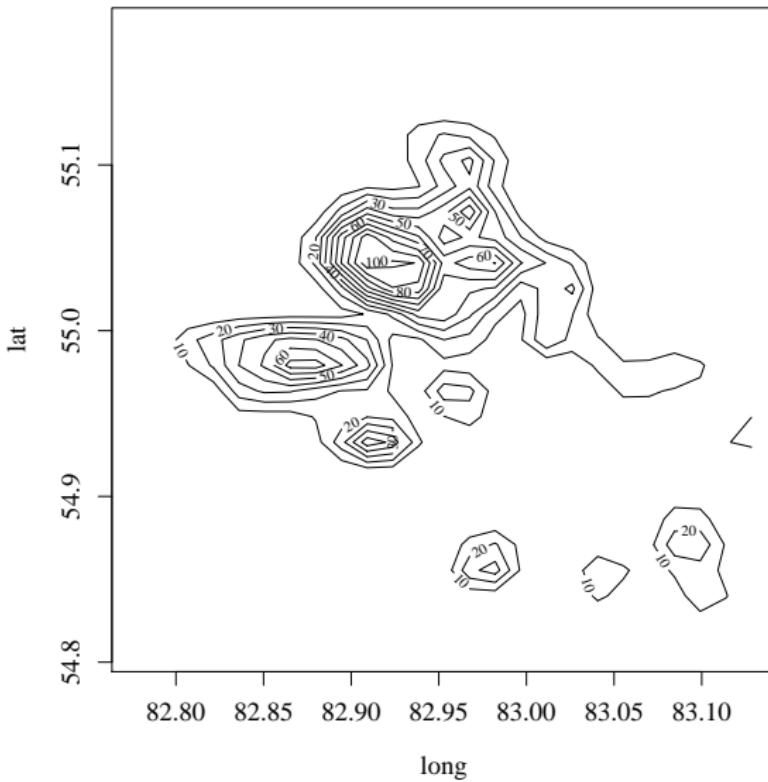


Оценка плотности распределения

```
> summary(dd$long)
   Min. 1st Qu. Median     Mean 3rd Qu.     Max.    NA's
82.78    82.90   82.94    82.94   82.97    83.13      4
> summary(dd$lat)
   Min. 1st Qu. Median     Mean 3rd Qu.     Max.    NA's
54.81    54.98   55.02    55.01   55.05    55.18      4
```

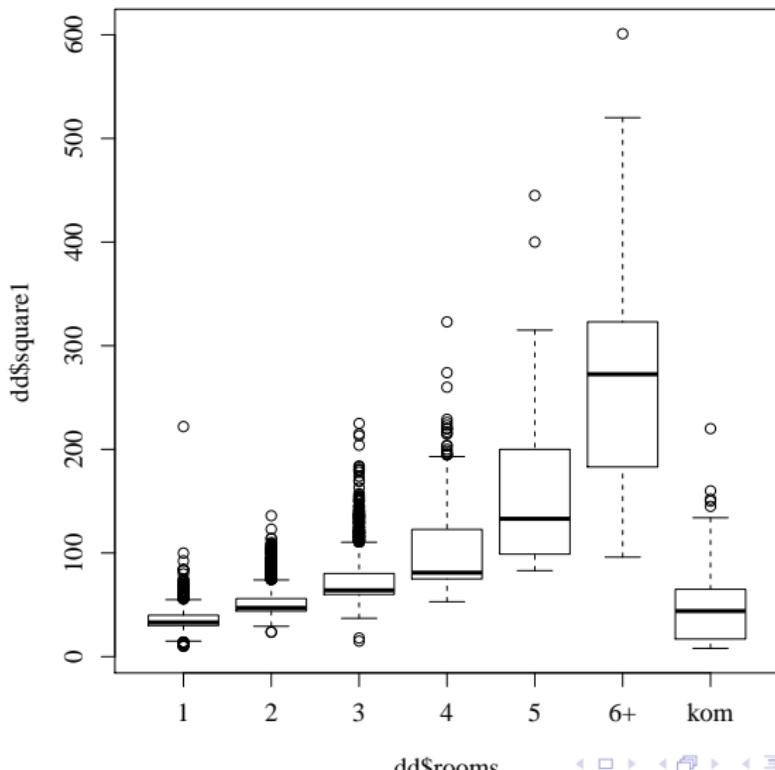
```
complete=complete.cases(dd[,1:2])
library(MASS)
f1=kde2d(dd$long[complete],dd$lat[complete])
contour(f1)
```

Оценка плотности распределения



Ящики с усами

plot(dd\$rooms, dd\$square1)



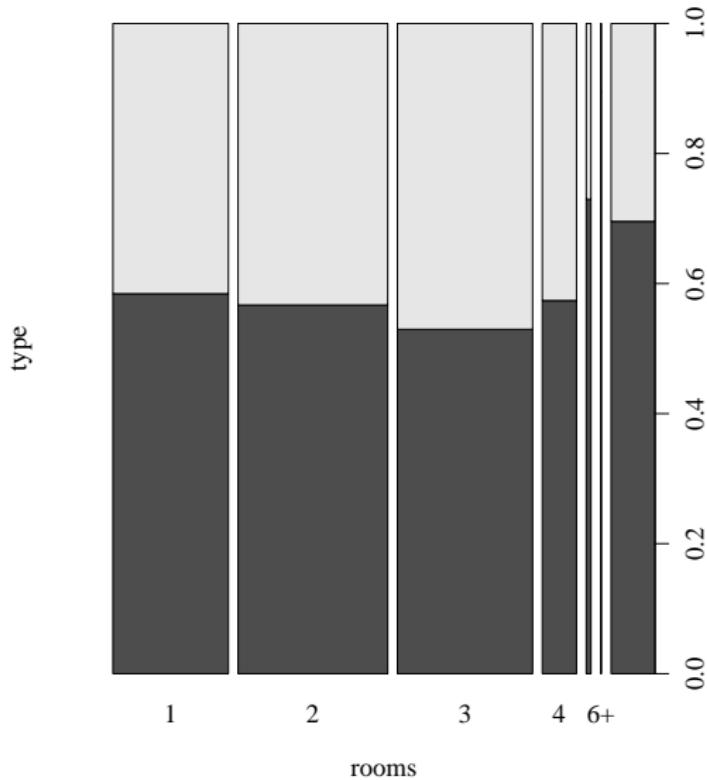
Кросс-табуляция

```
table(dd$type)
```

б	бмк	бт	гб	д	к	кмк	м	мжбкк	п	пб	пк	с	ш
26	13	10	2	26	7094	315	89	50	5244	3	26	15	80

```
tt=dd$type  
lev=levels(dd$type)  
type=factor(tt[tt==lev[6]|tt==lev[10]])  
rooms=dd$rooms[tt==lev[6]|tt==lev[10]]  
plot(rooms,type)
```

Кросс-табуляция



Кластеризация в R

Задача: выделить кластеры квартир по цене за кв. метр

1. Подготовка данных

```
price.m=dd$price/dd$square1  
summary(price.m)
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
	3.60	52.54	61.43	68.28	72.58	70830.00	31

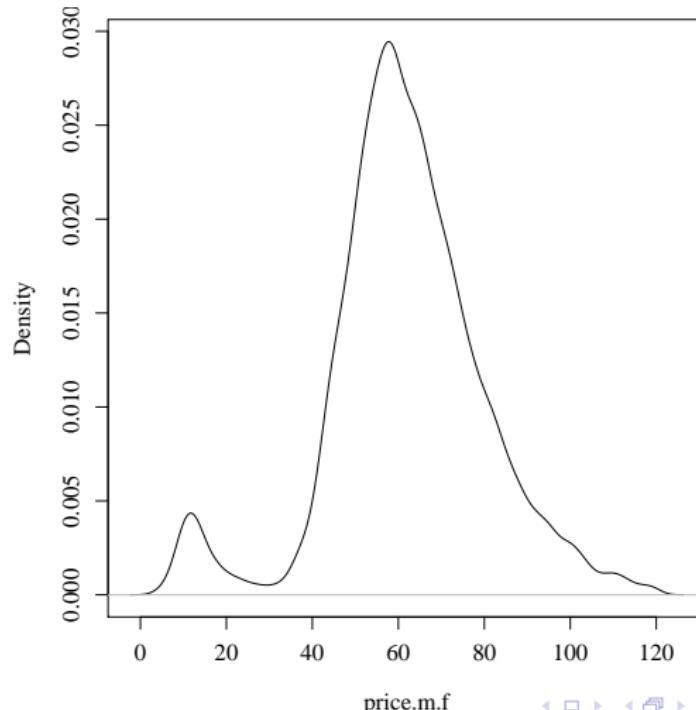
```
filt=!is.na(price.m)&price.m<quantile(price.m,0.99,na.rm=TRUE)  
price.m.f=price.m[filt]  
summary(price.m.f)
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
	3.60	52.46	61.29	62.02	72.16	120.40

Кластеризация в R

2. Визуальное представление (оценка функции плотности)

```
plot(density(price.m.f))
```



Кластеризация в R

3. Выделение кластеров методом k-средних

```
clust=kmeans(price.m.f,3)
```

```
str(clust)
```

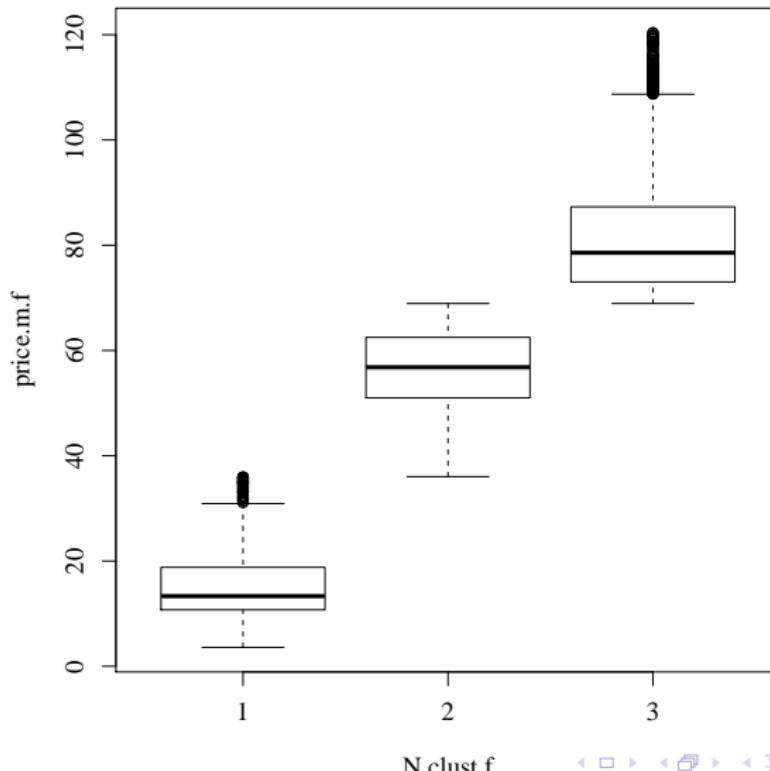
```
List of 9
$ cluster      : int [1:12850] 2 3 2 3 2 3 3 2 3 3 ...
$ centers      : num [1:3, 1] 15.8 56.3 81.6
..- attr(*, "dimnames")=List of 2
... .$. : chr [1:3] "1" "2" "3"
... .$. : NULL
$ totss        : num 4237107
$ withinss     : num [1:3] 36867 467365 472078
$ tot.withinss: num 976311
$ betweenss    : num 3260797
$ size         : int [1:3] 680 8162 4008
$ iter         : int 3
$ ifault       : int 0
- attr(*, "class")= chr "kmeans"
```

```
N.clust=clust$cluster
```

```
N.clust.f=as.factor(N.clust)
```

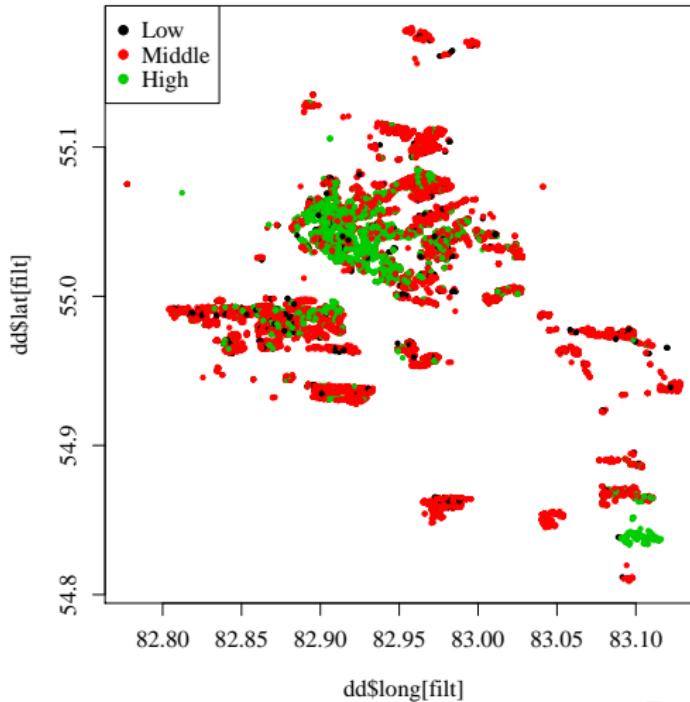
Кластеризация в R

`plot(N.clust.f,price.m.f)`



Кластеризация в R

```
plot(dd$long[filt],dd$lat[filt],col=N.clust,cex=0.5,pch=19)
legend('topleft',c('Low','Middle','High'),col=1:3,pch=19)
```



Практический пример

- ① С использованием статистической среды R провести первичную статистическую обработку и визуально представить исходные данные, сделать выводы о возможных взаимосвязях.
- ② Исходные данные содержатся в файле Example_data.csv. Представлена небольшая выборка объектов жилой недвижимости города Новосибирска по объявлениям о продаже на сайте ngs.ru.

Исходные данные

Таблица: Показатели и их обозначения

Обозначение	Показатель
latitude	Широта
longitude	Долгота
rooms	Количество комнат
square1	Общая площадь жилья
square2	Жилая площадь
square3	Площадь кухни
stage	Этаж
stages	Число этажей в доме
price	Цена продажи

Импорт данных

- Импорт данных в формате .csv осуществляется с помощью команды `read.csv2`. Для этого необходимо ввести команду:
- `dd=read.csv2('Folder/Example_data.csv', stringsAsFactors=TRUE)`
- Вместо `dd` может быть указано любое имя, по которому далее будете обращаться к объекту.
- Вместо `Folder` указывается путь к документу (через один прямой слэш или два обратных слэша).
- Команда `str(dd)` позволяет просмотреть тип переменных в таблице данных и их обозначения.

Графическое отображение одномерных данных

- Если фактор качественный, то графически отобразить таблицу частот можно с помощью команды
- `plot(dd$rooms)`
- Вместо `rooms` может быть указано любое название качественного фактора из таблицы данных.
- `hist(dd$square1)`
- Вместо `square1` может быть указано любое название количественного фактора из таблицы данных.

Взаимосвязь факторов типа "количественный-количественный"

- Корреляционное поле можно отобразить с помощью команды:
- `plot(dd$long, dd$lat)`
- Вместо `long` и `lat` могут быть названия двух любых количественных факторов.
- Иногда нагляднее будет выглядеть оценка плотности двумерного распределения, которую можно построить с помощью функции `kde2dMASS` (доступна в пакете `MASS`).
- Для этого сначала подключается пакет:
- `library(MASS)`
- Затем строятся оценки плотности, они записываются в отдельный объект:
- `f1=kde2d(dd$long,dd$lat)`

Функция kde2d{MASS}

- Функцию можно применять только к переменным, не содержащим пропущенные значения (NA). Для того чтобы удалить пропущенные значения из всех переменных одновременно можно использовать функцию complete.cases возвращающую логический вектор, в котором пропущенным наблюдением соответствует FALSE. А потом выбрать значения по этому условию.
- `complete=complete.cases(dd[,1:2])`
- `x=dd$long [complete]`
- `y=dd$lat [complete]`
- И применить функцию `kde2d{MASS}` к данным без пропущенных наблюдений:
- `f1=kde2d(x,y)`
- Чтобы отобразить результаты на графике в виде линий равного уровня используется команда:
- `contour(f1)`

Взаимосвязь факторов типа "качественный-количественный"

- Для отображения связи между качественным и количественным фактором используются кящики с усами, которые можно построить все той же функцией `plot`, где значением первого аргумента будет качественный признак, а второго - количественный.
- `plot(dd$rooms, dd$square1)`

Взаимосвязь факторов типа "качественный-качественный"

- Связь между двумя качественными признаками лучше всего интерпретировать путем визуального отображения таблицы сопряженности:
- `plot(dd$rooms, dd$type)`
- Чтобы отобразить на графике только нужные уровни фактора, можно использовать функцию `levels` для изменения уровней фактора:
 - `tt=dd$type`
 - `lev=levels(tt)`
 - `type=factor(tt[tt==lev[6]|tt==lev[10]])`
 - `rooms=dd$rooms[tt==lev[6]|tt==lev[10]]`

Кластеризация в R

- Осуществляется с помощью функции kmeans
- `clust=kmeans(price.m.f,3)`
- Вместо `price.m.f` может быть любой набор данных, по которому проводится кластеризация.

Программные системы статистического анализа

Тимофеева А.Ю.

к.э.н., доцент кафедры ТПИ

a.timofeeva@corp.nstu.ru

22 сентября, 2021

Функциональное программирование в R

- Создание функций
- Векторизованные функции, циклы
- Семейство функций `apply()`
- Оценка производительности
- Набор пакетов `tidyverse`
- Конвейер операций (пакет `magrittr`)
- Работа с тиблами

Элементы функционального программирования

В R можно создавать классы. Стандартные классы:

① S3

- нет формальной декларации класса
- функция может иметь разное поведение в зависимости от класса (*method dispatch*)
- такие функции называются *generic*

② S4

- строгое определение класса и его полей
- больше возможностей для реализации методов

③ Reference classes

Функции

Generic

Функция может себя по-разному вести в зависимости от того, чтобы приходит ей в качестве аргумента. Например, `print(x)`

- вызывается `print.data.frame(x)`, если `x` - `data.frame`;
- выводится `print.function(x)`, если `x` - функция и др.;
- если ни один из методов не подходит, то `print.default(x)`.

Чистые функции

- Детерминированные результат
(для работы со случайными числами `set.seed`)
- Без сторонних эффектов
(не использовать глобальное присвоение `<< -`)

Ленивые вычисления (lazy evaluations): параметры функций вызываются только тогда, когда они понадобятся, а не заранее.



Создание пользовательских функций

```
function ( arglist ) {body}
```

- Аргументы перечисляются через запятые по порядку или по имени, можно задавать значения по умолчанию, список аргументов переменной длины с помощью ... в конце
- При выполнении объекты - в первую очередь из окружения функции, далее из глобального окружения
- На выходе return() или возвращается последний результат вычислений, возвращает **только один объект**

```
LS_linear_system=function (X,y) {  
  solve(t(X) %*% X, t(X) %*% y)  
}  
LS_linear_system(matrix(1:10,5,2),runif(5))  
[1,] [1]  
[1,] -0.3086515  
[2,] [2]  
[2,] 0.1795257
```

Создание функций-оберток

На основе существующих, но с некоторыми доработками, например, с фиксированными параметрами или заданными выходными значениями.

На основе `plot()` создадим функцию, которая отображает график линией, а не точками.

```
plot_lines=function(x, y, ...){  
  plot(x, y, type="l", ...)  
}
```

На основе `kmeans()` создадим функцию, которая возвращает только вектор номеров кластеров.

```
kmeans_clust=function(x, centers, ...){  
  kmeans(x, centers, ...)$cluster  
}
```

Диагностические сообщения

Для вывода диагностических сообщений используются функции:

- message - выводит сообщение
- warning - выводит сообщение об ошибке, не останавливая выполнения команд
- stop - выводит сообщение об ошибке и останавливает выполнение команд

```
LS_linear_system=function (X, y) {  
  if(!is.matrix(X)) stop("X must be a matrix")  
  if(!is.vector(y)) stop("y must be a vector")  
  if(nrow(X) != length(y)) {  
    stop("X, y must have the same number of rows")}  
  if (det(t(X) %*% X)==0) stop("Collinearity problem")  
  solve(t(X) %*% X, t(X) %*% y)  
}
```

Функции как объекты первого порядка

- С функциями можно делать практически все то же самое, что и с другими объектами в R.
- Можно создать список из функций.
- Можно создавать функции внутри функций.
- Можно передать функцию в качестве аргумента в другие функции.

```
library(MASS)
LS_coef=function (X,y,FUN=lm) {
  FUN(y ~ X+0)$coef
}
set.seed(1553)
LS_coef(matrix(1:10,5,2),runif(5),rlm)
```

X1	X2
0.10234861	0.03898758

Циклы в R

```
v=NULL  
for (i in 1:10^7){  
v=v+i  
}
```

Проверим время выполнения.

```
v=NULL  
system.time(  
for (i in 1:10^7){  
v=v+i  
})
```

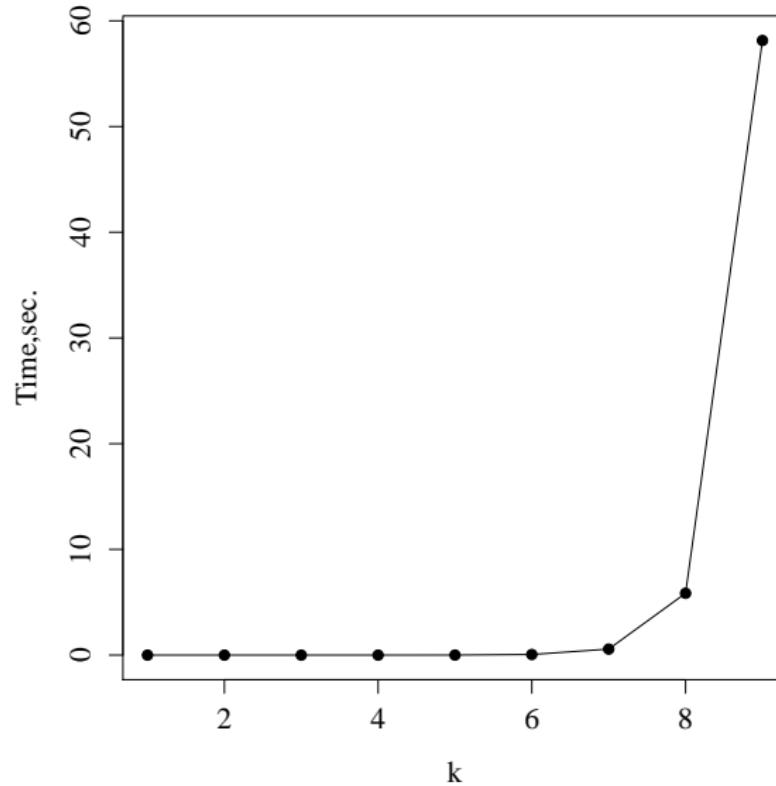
Время выполнения 0.575 секунд.

Циклы работают медленно.

Циклы в R

```
loop_time=NULL
for (k in 1:9){
  loop_time[k]=system.time(
    {v=NULL
      for (i in 1:10^k) v=v+i}
    )[3]
}
```

Циклы в R



Векторизованные функции

```
system.time({v=sum(1:10^k)})
```

Время выполнения 0 секунд вне зависимости от k.

Предпочтительнее использовать векторизованные функции.

Пример векторизованных функций

- Статистические: mean, sd, var
- Математические: diff, sqrt
- Генерация случайных чисел: replicate
- Для обработки массивов: apply
- Внешнее произведение: outer
- Для создания векторизованных функций: Vectorize
- Вызов функции на списке аргументов: do.call

Векторизованные функции: replicate

- Можно использовать для моделирования
- Повторять выражение (expr - второй аргумент) заданное количество раз (n - первый аргумент)
- Окончательный результат регулируется с помощью параметра simplify: TRUE - массив, FALSE - список

```
set.seed(2020)
replicate(4, rnorm(5))
```

	[,1]	[,2]	[,3]	[,4]
[1,]	0.3769721	0.7205735	-0.8531228	1.80004312
[2,]	0.3015484	0.9391210	0.9092592	1.70399588
[3,]	-1.0980232	-0.2293777	1.1963730	-3.03876461
[4,]	-1.1304059	1.7591313	-0.3715839	-2.28897495
[5,]	-2.7965343	0.1173668	-0.1232602	0.05830349

Векторизованные функции: семейство apply

- `apply` применяет заданную функцию к строкам или столбцам массива
- `lapply`, `sapply` применяют заданную функцию к элементам списка, столбцам структуры данных
- `mapply` - многомерная версия `apply`

```
set.seed(2020)
norm_samples=replicate(10, rnorm(500,5,3))
mean_samples=apply(norm_samples,2,mean)
sd_samples=apply(norm_samples,2,sd)
mean_samples
[1] 4.836259 5.003674 5.128221 4.882071 4.735781
sd_samples
[1] 3.193859 3.026544 3.041978 2.992568 2.975400
```

Анонимные функции

Анонимные функции

Функции, которые будут использоваться один раз и не имеют названия.

Например, посчитаем количество пропусков в каждом столбце таблицы данных:

```
dd=read.csv2("Example_data.csv")
sapply(dd, function(x) sum(is.na(x)))
  latitude longitude      rooms    square1    square2
            0           0          0           0          32
square3      stage     stages      type      price
            42           0          0           0           0
```

Векторизованные функции: outer

- Применяется к векторам или массивам
- Вычисляет заданную функцию, по умолчанию FUN="*"
- %o% -бинарный оператор outer(x, y, "*")

```
outer(1:3, letters[1:3], "paste", sep=". ")
      [,1]   [,2]   [,3]
[1,] "1.a" "1.b" "1.c"
[2,] "2.a" "2.b" "2.c"
[3,] "3.a" "3.b" "3.c"
```

Векторизованные функции: Vectorize

- Позволяет векторизовать функцию по заданному аргументу
- Исходная функция в качестве заданного аргумента принимает скаляр
- По умолчанию упрощает результат (SIMPLIFY = TRUE)

```
kmeans_clust=function(x, centers){  
kmeans(x, centers)$cluster  
}
```

```
kmeans_vect=Vectorize(kmeans_clust , "centers")
```

```
kmeans_results=kmeans_vect(2:5 ,x=norm_samples)  
str(kmeans_results)  
int [1:500, 1:4] 1 2 1 1 1 2 2 2 2 2 ...
```

Векторизованные функции: do.call

- Выполняет вызов функции по имени или выражению
- Можно передать список аргументов

```
variables=expand.grid(letters[1:2], 1:3, c("+", "-"))
```

```
str(variables)
'data.frame':   12 obs. of  3 variables:
 $ Var1: Factor w/ 2 levels "a","b": 1 2 1 2 1 2 1 2 1
 $ Var2: int  1 1 2 2 3 3 1 1 2 2 ...
 $ Var3: Factor w/ 2 levels "+","-": 1 1 1 1 1 1 1 2 2 2
```

```
do.call("paste", c(variables, sep = ""))
[1] "a1+" "b1+" "a2+" "b2+" "a3+" "b3+" "a1-" "b1-" ..
```

Оценка времени выполнения кода

Время выполнения фрагмента кода можно измерить как разницу между системным временем в начале и в конце фрагмента кода.

- `Sys.time` и `system.time`
- `library(tictoc)`

При каждом запуске время выполнения кода принимает случайное значение.

Специализированные библиотеки позволяют получить осредненную оценку времени выполнения.

- `library(rbenchmark)`
- `library(microbenchmark)`

Оценка времени выполнения кода: Sys.time

Сравним время выполнения векторизованной функции и цикла.

```
set.seed(2021)
start_time = Sys.time()
kmeans_results=kmeans_vect(2:5,x=norm_samples)
end_time = Sys.time()
end_time - start_time
Time difference of 0.005933285 secs
```

```
set.seed(2021)
start_time = Sys.time()
for (k in 2:5){
  kmeans_results[,k-1]=kmeans(norm_samples, k)$cluster
}
end_time = Sys.time()
end_time - start_time
Time difference of 0.007445097 secs
```

Оценка времени выполнения кода: system.time

Сравним время выполнения векторизованной функции и цикла.

```
set.seed(2021)
system.time({ kmeans_vect(2:5, x=norm_samples) })
  user  system  elapsed
0.001   0.000   0.002

set.seed(2021)
system.time({ for (k in 2:5){
  kmeans_results[, k-1]=kmeans(norm_samples, k)$cluster
} })
  user  system  elapsed
0.003   0.000   0.004
```

Оценка времени выполнения кода: system.time

- "Elapsed CPU time общее время, которое затребовалось на выполнение кода.
- "User CPU time процессорное время, затраченное текущим процессом (т. е. текущим сеансом R)
- "System CPU time процессорное время, затраченное ядром (операционной системой) от имени текущего процесса (открытие файлов, выполнение ввода, вывода и др.).

Оценка времени выполнения кода: tictoc

```
tic("total")
tic("data_generation")
X = matrix(rnorm(50000*1000), 50000, 1000)
b = sample(1000)
y = runif(1) + X %*% b + rnorm(50000)
toc()
#data_generation: 2.639 sec elapsed
tic("model_fitting")
model = lm(y ~ X)
toc()
#model_fitting: 39.833 sec elapsed
toc()
#total: 45.366 sec elapsed
```

Оценка времени выполнения кода: rbenchmark

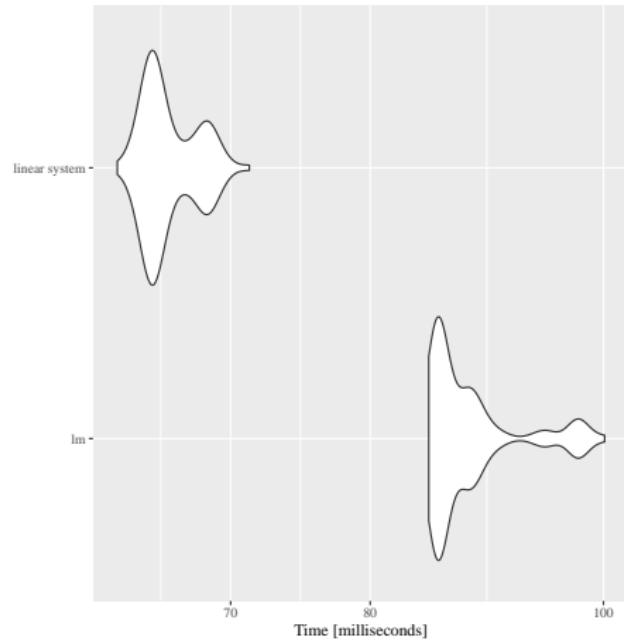
```
benchmark("lm" = {  
    X = matrix(rnorm(1000), 100, 10)  
    y = X %*% sample(10) + rnorm(100)  
    b = lm(y ~ X + 0)$coef  
},  
"linear_system" = {  
    X = matrix(rnorm(1000), 100, 10)  
    y = X %*% sample(10) + rnorm(100)  
    b = solve(t(X) %*% X, t(X) %*% y)  
},  
replications = 1000,  
columns = c("test", "elapsed",  
          "relative", "user.self"))  
#      test elapsed relative user.self  
#2 linear_system 0.088     1.000     0.087  
#1 lm            0.517     5.875     0.517
```

Оценка времени выполнения кода: microbenchmark

```
set.seed(2017)
n = 10000
p = 100
X = matrix(rnorm(n*p), n, p)
y = X %*% rnorm(p) + rnorm(100)
mbm = microbenchmark("lm" = {b = lm(y ~ X + 0)$coef},
                      "linear_system" = {
                        b = solve(t(X) %*% X, t(X) %*% y)
                      })
mbm
Unit: milliseconds
      expr        min         iq       mean      median
      lm 84.61135 85.31712 87.87545 85.94311
linear_system 62.81170 64.75893 65.98472 65.25156
```

Оценка времени выполнения кода: microbenchmark

```
library(ggplot2)
autoplot(mbm)
```



Программные системы статистического анализа

Тимофеева А.Ю.

к.э.н., доцент кафедры ТПИ

a.timofeeva@corp.nstu.ru

6 октября, 2021

Пакеты ядра tidyverse

Tidyverse - это набор пакетов R, созданных для науки о данных.
Все пакеты имеют общую философию дизайна, грамматику и структуры данных.

- ggplot2 - для визуализации
- tibble - для работы с тиблами, продвинутый вариант датафрейма
- tidyverse - для формата tidy data
- readr - для чтения файлов в R
- purrr - замена семейства функций apply()
- dplyr - для преобразования данных
- stringr - для работы со строковыми переменными
- forcats - для работы с переменными-факторами

Дополнительные пакеты tidyverse

- vroom - для быстрой загрузки табличных данных
- readxl - для чтения .xls и .xlsx
- jsonlite - для работы с JSON
- xml - для работы с XML
- DBI - для работы с базами данных
- rvest - для веб-скреппинга
- lubridate - для работы с датами и временем
- tidytext - для работы с текстами и корпусами
- glue - для продвинутого объединения строк
- magrittr - с несколькими вариантами pipe оператора

Загрузка данных с помощью readr

```
library(readr)
dd=read_csv2("Example_data.csv")
```

Parsed with column specification :

```
cols(
  latitude = col_double(),
  longitude = col_double(),
  rooms = col_character(),
  square1 = col_double(),
  square2 = col_double(),
  square3 = col_double(),
  stage = col_integer(),
  stages = col_integer(),
  type = col_character(),
  price = col_integer()
)
```

Загрузка данных с помощью readr

При загрузке данных с помощью readr, получаем tibble,
а не data.frame.

```
str(dd)
#tibble [ , 10] [500 x 10]

class(dd)
#[1] "tbl_df"     "tbl"        "data.frame"
```

Тиббл (tibble) - это “усовершенствованный” data.frame.

Почти все, что работает с data.frame, работает и с тибблами.

Однако у тибблов есть свои дополнительные преимущества.

Например, более аккуратный вывод в консоль.

Тиббл

```
dd
# A tibble: 500 X 10
  latitude longitude rooms square1 square2 square3
  <dbl>     <dbl>   <chr>    <dbl>    <dbl>    <dbl>
1 55.0      82.9    2        57       35       9
2 55.0      83.0    2        48       27       8
3 55.0      82.9    3        58       38       8
4 55.1      83.0    1        37.7    19.2     9
5 55.1      82.9    3        86       54      12
6 55.0      82.9    3        90       54      12
7 55.0      82.8    2        44       28       6
8 55.0      82.8    3        60       38      8.6
9 55.1      82.9    3        86       51      14
10 55.0     83.0    3        58       41       6
# ... with 490 more rows
```

Тиблы

Функции различных пакетов tidyverse сами конвертируют в тиблы при необходимости.

Если же нужно это сделать самостоятельно, то можно это сделать так:

```
dd_df = as.data.frame(dd)
class(dd_df)
# [1] "data.frame"
as_tibble(dd_df)
```

Можно создавать тиблы вручную с помощью функции `tibble()`, которая работает аналогично функции `data.frame()`.

Конвейер операций magrittr

Оператор `%>%` называется “пайпом” (pipe), т.е. “трубой.”

Буквально означает “применить функцию”.

Он позволяет:

- записывать последовательность действий слева направо ($x \%>\% f \%>\% h$), а не изнутри наружу $h(f(x))$.
- легко добавлять в конвейер новое действие.
- избавиться от использования вложенных функций.

Пайп не дает какой-то дополнительной функциональности или дополнительной скорости работы.

Он создан исключительно для читабельности и комфорта.

Конвейер операций magrittr: примеры

Вместо

```
sum(sqrt(abs(sin(1:22))))
```

можно записать

```
1:22 %>%
  sin() %>%
  abs() %>%
  sqrt() %>%
  sum()
```

Если результат выполнения функции используется не первым аргументом следующей функции, то его позиция указывается с помощью .

```
"AAA" %>%
  c("___", ., "___")
## [1] "___"           "AAA" "___"
```

Главные пакеты tidyverse: dplyr и tidyr

dplyr — это самая основа всего tidyverse.

Этот пакет предоставляет основные функции для манипуляции с тиблами.

Пакет tidyr дополняет dplyr, предоставляя полезные функции для тайдификации тиблов.

Тайдификация (“аккуратизация”) данных означает приведение табличных данных к такому формату, в котором:

- Каждая переменная имеет собственный столбец
- Каждый наблюдение имеет собственную строку
- Каждое значение имеет свою собственную ячейку

Большинство функций dplyr и tidyr работают с целым тиблом сразу, принимая его в качестве первого аргумента и возвращая измененный тибл. Это позволяет превратить весь код в последовательный набор применяемых функций, соединенных пайпами.

Работа с колонками тиббла: tidyselect

dplyr::select() позволяет выбирать колонки по номеру или имени (кавычки не нужны).

```
dd %>%  
  select(1,5)
```

```
dd %>%  
  select(rooms, stage, type, price)
```

Оператор : для выбора нескольких соседних колонок

```
dd %>%  
  select(square1:stages)
```

Используя ! можно вырезать ненужные колонки.

```
dd %>%  
  select(!square1:square3)
```

Работа с колонками тиббла: tidyselect

Другие известные логические операторы (& и |) тоже работают в tidyselect.

В дополнение к логическим операторам и : в tidyselect есть набор вспомогательных функций, работающих исключительно в контексте выбора колонок с помощью tidyselect.

Вспомогательная функция позволяет обратиться к последней колонке тиббла:

```
dd %>%  
  select(stage:last_col())
```

А функция everything() позволяет выбрать все колонки. Она не будет дублировать выбранные колонки, поэтому ее можно использовать для перестановки колонок в тиббле:

```
dd %>%  
  select(price, rooms, everything())
```

Работа с колонками тиббла: relocate

Для перестановки колонок удобнее использовать специальную функцию `relocate()`.

```
dd %>%  
  relocate(price, .after = rooms)
```

```
dd %>%  
  relocate(price, where(is.numeric), .after = rooms)
```

Можно даже выбирать колонки по паттернам в названиях.

Например, можно найти колонки с одинаковым префиксом:

```
dd %>%  
  select(starts_with("square"))
```

Работа с колонками тиббла: where

Можно выбирать по содержимому колонок с помощью `where()`.
Это напоминает применение `sapply()` на датафрейме для индексирования колонок: в качестве аргумента для `where` принимается функция, которая применяется для каждой из колонок, после чего выбираются только те колонки, для которых было получено TRUE.

```
dd %>%  
  select(where(is.numeric))
```

Как выбрать все колонки без NA?

```
dd %>%  
  select(where(function(x) !any(is.na(x))))
```

Работа с колонками тиббла: rename, pull

Переименование колонок

```
dd %>%  
  select(x = latitude, y = longitude)
```

```
dd %>%  
  rename(x = latitude, y = longitude)
```

pull() делает то же самое, что и индексирование с помощью \$, т.е. извлекает из тиббла вектор с выбранным названием.

```
dd %>%  
  pull(price) %>%  
  mean()
```

В отличие от базового R, tidyverse **нигде не сокращает имплицитно результат вычислений до вектора**, поэтому функция pull() - это основной способ извлечения колонки из тиббла как вектора.

Работа со строками тibble: slice, filter

Функция `dplyr::slice()` выбирает строки по их числовому индексу.

```
dd %>%  
  slice(1:3)
```

Функция `dplyr::filter()` позволяет выбирать строки по условию.

Причем для условий нужно использовать не векторы из тibble, а
название колонок (без кавычек) как будто бы они были переменными
в окружении.

```
dd %>%  
  filter.rooms > 2) %>%  
  select.rooms) %>%  
  table()
```

	3	4	5	6+	ком
	135	36	5	4	47

Работа со строками тibble: семейство функций slice()

У функции slice() есть множество родственников, которые объединяют функционал обычного slice() и filter().

Например, можно выбрать заданное количество строк, содержащих наибольшие или наименьшие значения по колонке:

```
dd %>%  
  slice_max(price, n = 3)
```

```
dd %>%  
  slice_min(price, n = 3)
```

Можно выбирать заданное количество случайных строк:

```
dd %>%  
  slice_sample(n = 3)
```

```
dd %>%  
  slice_sample(prop = .01)
```

Работа со строками тibble: удаление строк с NA

С помощью пакета `tidyverse` можно выбрать только строки без пропущенных значений.

```
dd %>%  
  drop_na()
```

Можно выбрать колонки, наличие NA в которых будет приводить к удалению соответствующих строк (не затрагивая другие строчки, в которых есть NA в остальных столбцах).

```
dd %>%  
  drop_na(square3)
```

Для выбора колонок здесь можно использовать `tidyselect`.

Работа со строками тibble: сортировка строк

Функция `dplyr::arrange()` сортирует строки от меньшего к большему (или по алфавиту - для текстовых значений) по выбранной колонке.

```
dd %>%  
  arrange(price)
```

Для сортировки в обратном порядке есть функция `desc()`.

```
dd %>%  
  arrange(desc(price))
```

Можно сортировать по нескольким колонкам сразу.

```
dd %>%  
  arrange(type, desc(price))
```

Работа со строками тибла: создание колонок

Функция `dplyr::mutate()` позволяет создавать новые колонки в тибле.

```
dd %>%  
  mutate(price.m2 = price/square1)
```

`dplyr::transmute()` - это аналог `mutate()`, который не только создает новые колонки, но и сразу же выкидывает все старые.

Допустимые операции:

- Векторизованные операции (длина новой колонки должна равняться числу строк тибла)
- Операции, которые возвращают одно значение (значение будет одинаковым на всю колонку, т.е. будет работать правило циклического повторения)

Работа со строками тибла: создание колонок

Правило циклического повторения не будет работать в остальных случаях (ошибка, если длина вектора не равна 1 или числу строк тибла).

Циклическое повторение кратных друг другу векторов - частый источник ошибок.

При необходимости разработчики dplyr рекомендуют использовать функцию rep().

```
dd %>%
```

```
mutate(one_and_two = rep(1:2, length.out = nrow(.)))
```

Агрегация данных в тибле

Агрегация по группам может использоваться для усреднения данных при различных условиях.

Агрегация в dplyr состоит из двух этапов:

- группировки

`group_by()`

- сводки итогов

`summarise()`

Агрегация данных в тибле: summarise

Функция `summarise()` работает очень похоже на `mutate()`, но в `summarise()` используются функции, которые возвращают вектор длиной 1 (`mean()`, `sd()`) или больше (`range`). Можно создавать несколько колонок через запятую (это работает и для `mutate()`).

```
dd %>%
```

```
  mutate(price.m2 = price/square1) %>%
  summarise(min(price.m2),
            max(price.m2))
```

```
# A tibble: 1 x 2
  `min(price.m2)` `max(price.m2)`
  <dbl>           <dbl>
1      5.94          200
```

Агрегация данных в тибле: summarise

Если внутри `summarise()` используются функции, возвращающие вектор из нескольких значений, то создается тибл такой же длины, как и получившийся вектор.

```
dd %>%  
  mutate(price.m2 = price/square1) %>%  
  summarise(price.m2.range = range(price.m2))
```

```
# A tibble: 2 x 1  
  price.m2.range  
  <dbl>  
1      5.94  
2     200
```

Агрегация данных в тибле: summarise

В dplyr есть дополнительные функции агрегации для более удобного индексирования в стиле tidyverse.

Например, функции dplyr::nth(), dplyr::first() и dplyr::last() позволяют извлекать значения из вектора по индексу.

```
dd %>%
```

```
  mutate(price.m2 = price/square1) %>%
  arrange(price.m2) %>%
  summarise(first = first(price.m2),
            tenth = nth(price.m2, 10),
            last = last(price.m2))
```

```
# A tibble: 1 x 3
  first   tenth  last
  <dbl>   <dbl> <dbl>
1 5.94    11.4  200
```

Агрегация данных в тибле: группировка

```
dplyr::group_by()
```

функция для группировки данных в тибле по дискретной переменной для дальнейшей агрегации с помощью summarise().

После применения появятся атрибут groups.

```
dd %>%  
  group_by(type) %>%
```

```
# A tibble: 500 x 10  
# Groups:   type [8]
```

Агрегация данных в тибле: группировка

Если после группировки применить функцию `summarise()`, то получим тибл со значением для каждой из групп.

```
dd %>%
```

```
group_by(type) %>%  
  summarise(min(price), max(price))
```

	type	'min(price)'	'max(price)'
	<chr>	<int>	<int>
1	б	6350	6350
2	д	1500	1500
3	к	475	35000
4	кмк	2500	8800
5	м	2750	3200
6	п	640	6550
7	пк	2150	2200
8	ш	1400	7000

Подсчет количества строк

Функции dplyr::n(), dplyr::count().

dd %>%

```
group_by(type) %>%  
summarise(n = n())
```

	type	n
	<chr>	<int>
1	б	1
2	д	1
3	к	272
4	кмк	12
5	м	4
6	п	206
7	пк	2
8	ш	2

Подсчет количества строк

Функция `n()` вместе с группировкой внутри `filter()` позволяет удобным образом “отрезать” от тибла редкие группы.

```
dd %>%
```

```
group_by(type) %>%  
filter(n() > 20) %>%  
summarise(min(price),max(price))
```

	type	'min(price)'	'max(price)'
	<chr>	<int>	<int>
1	к	475	35000
2	п	640	6550

или наоборот, выделить только маленькие группы.

```
dd %>%
```

```
group_by(type) %>%  
filter(n()==1)
```

Таблица частот

Для создания таблицы частот вместо совместного применения группировки и агрегации можно использовать функцию `count()`.

```
dd %>%  
  count(type)
```

Таблицу частот удобно сразу проранжировать.

```
dd %>%  
  count(type, sort = TRUE)
```

Функция `count()`, несмотря на свою простоту, является одной из наиболее используемых в `tidyverse`.

Уникальные значения

dplyr::distinct() - это более быстрый аналог unique(), позволяющий извлекать уникальные значения для одной или нескольких колонок.

```
dd %>%  
  distinct (rooms , type )
```

	rooms	type
	<chr>	<chr>
1	2	к
2	3	п
3	1	п
4	3	к
5	2	п
6	3	кмк
7	1	к
8	2	д
...

Создание колонок с группировкой

Иногда нужно агрегировать данные, но при этом сохранить исходную структуру тибла.

Например, нужно посчитать средние значения по группе для последующего сравнения с индивидуальными значениями.

```
dd %>%
```

```
group_by(rooms) %>%
  mutate(price_mean = mean(price)) %>%
  select(rooms, price, price_mean)
```

	rooms	price	price_mean
	<chr>	<int>	<dbl>
1	2	5200	3202.
2	2	3600	3202.
3	3	3630	4491.
4	1	2350	2456.
5	3	5700	4491.

Преобразование нескольких колонок

Пусть мы хотим посчитать среднюю цену и площадь жилья, группируя по числу комнат.

Можно посчитать это внутри одного summarise(), использую запятую:

```
dd %>%
```

```
group_by(rooms) %>%
  summarise(price_mean = mean(price),
            square1_mean = mean(square1))
```

	rooms	price_mean	square1_mean
	<chr>	<dbl>	<dbl>
1	1	2456.	35.9
2	2	3202.	49.8
3	3	4491.	69.8
4	4	5659.	92.4
5	5	8090	119

Преобразование нескольких колонок: dplyr::across()

Если таких колонок будет много, то это уже станет неудобным. Поэтому в dplyr есть функция для операций над несколькими колонками сразу: `dplyr::across()`. Она схожа с функциями семейства `apply()` и использует `tidyselect` для выбора колонок.

Конструкции с функцией `across()` можно разбить на три части.

- ❶ Выбор колонок с помощью `tidyselect`.
- ❷ Применение функции `across()`. Первый аргумент `.col` — колонки, выбранные с помощью `tidyselect`, по умолчанию `everything()`. Второй аргумент `.fns` — функция или список функций, которые будут применены к выбранным колонкам. Если функции требуют дополнительных аргументов, то они перечисляются внутри `across()`.
- ❸ Использование `summarise()` или другой функции dplyr. В качестве аргумента функции используется результат работы функции `across()`.

Преобразование нескольких колонок: dplyr::across()

```
dd %>%  
  group_by(rooms) %>%  
  summarise(across(square1:square3, mean, na.rm=TRUE))
```

	rooms	square1	square2	square3
	<chr>	<dbl>	<dbl>	<dbl>
1	1	35.9	19.0	8.28
2	2	49.8	30.1	8.07
3	3	69.8	45.0	9.77
4	4	92.4	60.1	14.9
5	5	119	72.5	12.2
6	6+	307.	162.	34.6
7	ком	44.4	16.6	8.77

Преобразование нескольких колонок: dplyr::across()

Использование tidyselect внутри across() открывает большие возможности.

```
dd %>%
  select(!square1:square3) %>%
  group_by(rooms) %>%
  summarise(across(where(is.numeric),
                  mean, na.rm = TRUE),
            across(where(is.character),
                  function(x){mean(nchar(x))}))
```

	rooms	latitude	longitude	stage	stages	price	type
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	1	55.0	82.9	5.13	8.88	2456.	1.09
2	2	55.0	82.9	4.36	7.61	3202.	1.06
3	3	55.0	83.0	4.69	8.6	4491.	1.03
4	4	55.0	82.9	3.81	9.5	5659.	1.06

Преобразование нескольких колонок: dplyr::across()

Можно использовать список функций.

```
dd %>%  
  group_by(rooms) %>%  
  summarise(across(square1:square3,  
    list(min = function(x) min(x, na.rm = TRUE),  
        mean = function(x) mean(x, na.rm = TRUE),  
        max = function(x) max(x, na.rm = TRUE),  
        na_n = function(x, ...) sum(is.na(x))))  
  )  
  )
```

Массовые операции с колонками с помощью mutate():

```
dd %>%  
  mutate(across(where(is.character), as.factor))
```

Объединение нескольких тибллов

Для структурно схожих тибллов:

- По строкам (тибллы должны иметь одинаковые колонки)

`bind_rows()`

- По столбцам (тибллы должны иметь одинаковые строки)

`bind_cols()`

Для реляционных данных: семейство функций

`*_join()`

`left_join()`

`right_join()`

`full_join()`

`inner_join()`

`semi_join()`

`anti_join()`

Широкий и длинный формат данных

Принцип tidy data предполагает, что каждая строчка содержит в себе одно измерение, а каждая колонка - одну характеристику.

Повторные измерения можно хранить как:

- одну колонку для каждого измерения (**широкий формат**),
- две колонки (**длинный формат**): одна колонка - для идентификатора измерения, другая колонка - для записи самого измерения.

Перевод из одного формата в другой с помощью пакета `tidyR`:

- из широкого в длинный формат

`pivot_longer()`

- из длинного в широкий формат

`pivot_wider()`

Программные системы статистического анализа

Тимофеева А.Ю.

к.э.н., доцент кафедры ТПИ

a.timofeeva@corp.nstu.ru

20 октября, 2021

Управление памятью в R

- Размер объекта
- Использование памяти и сборка мусора
- Модификация на месте
- Профилирование памяти

Размер объекта

```
object.size()
```

сообщает о пространстве, выделенном для объекта
(оценка памяти, которая используется для хранения объекта в R).

```
object.size(1:10)
```

```
# 96 bytes
```

```
object.size(mean)
```

```
# 1184 bytes
```

```
df=read.csv2("Example_data_utf8.csv")
```

```
object.size(df)
```

```
# 33656 bytes
```

```
library(readr)
```

```
tb=read_csv2("Example_data_utf8.csv")
```

```
object.size(tb)
```

```
# 42664 bytes
```

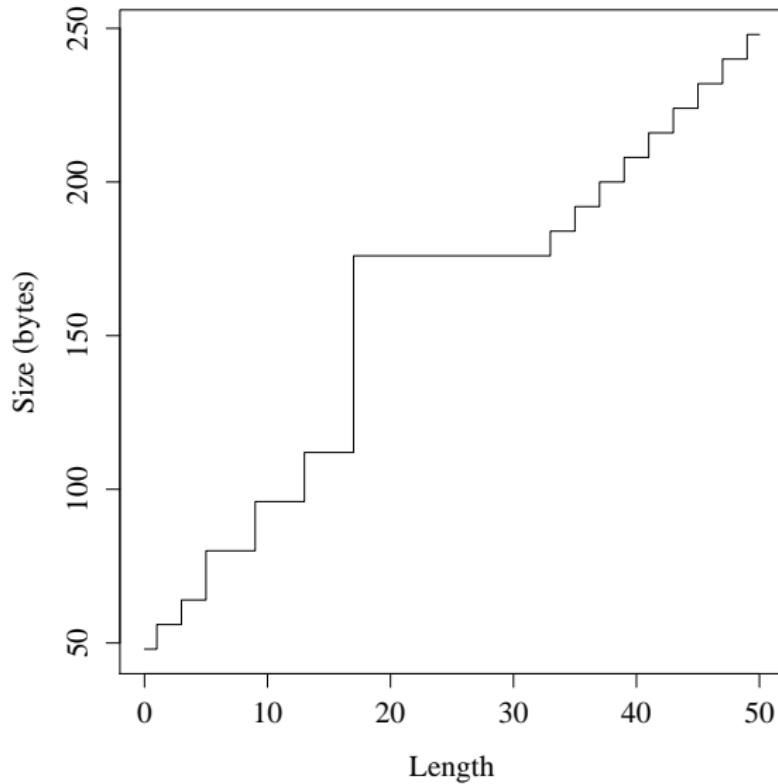
Выделение памяти: целочисленные векторы

- Размер пустого вектора будет равен нулю?
- Растет ли использование памяти пропорционально длине вектора?

Вычислим и отобразим использование памяти целочисленными векторами длиной от 0 до 50 элементов.

```
sizes = sapply(0:50, function(n)
                object.size(seq_len(n)))
plot(0:50, sizes, xlab = "Length",
     ylab = "Size (bytes)", type = "s")
```

Выделение памяти: целочисленные векторы



Выделение памяти: целочисленные векторы

Каждый вектор длины 0 занимает 48 байт памяти:

```
object.size(numeric())
# 48 B
object.size(logical())
# 48 B
object.size(character())
# 48 B
object.size(list())
# 48 B
```

Выделение памяти: пустой вектор

При хранении каждого объекта в R выделяется память для:

- метаданных объекта (базовый тип, информация, используемую для отладки и управления памятью),
- двух указателей: один на следующий объект в памяти и один на предыдущий объект (двусвязный список, позволяющий внутреннему коду R перебирать каждый объект в памяти),
- указателя на атрибуты.

У векторов есть дополнительные компоненты:

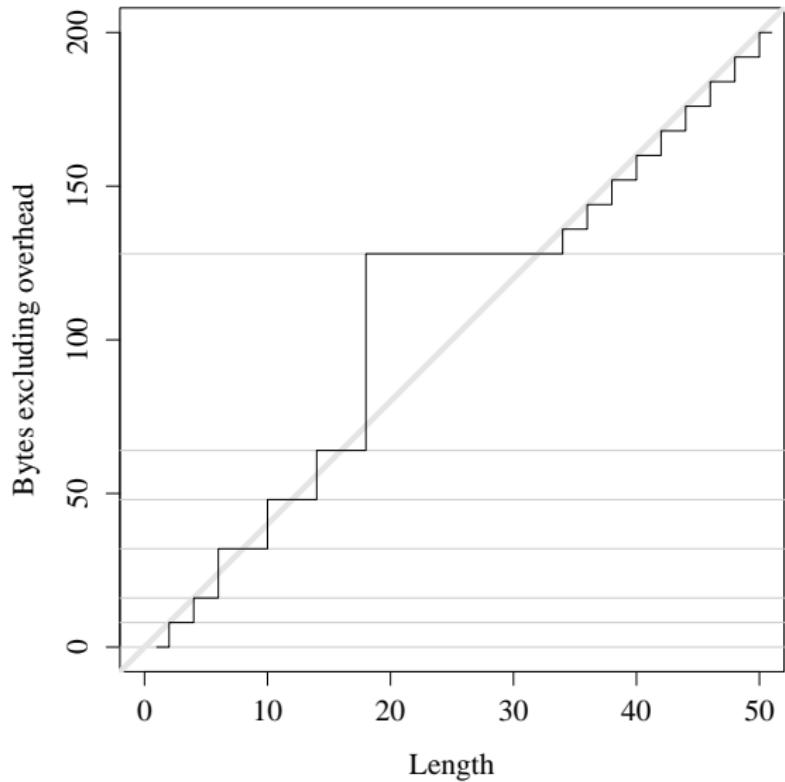
- длина вектора,
- данные: пустой вектор имеет 0 В данных, числовые векторы занимают 8 В для каждого элемента, целочисленные векторы 4 В и комплексные векторы 16 В.

Выделение памяти: выделение блоков памяти

Почему нерегулярно увеличивается объем памяти?

- Запрос памяти у операционной системы - относительно дорогая операция.
- Необходимость запрашивать память каждый раз, когда создается небольшой вектор, значительно замедлит работу R.
- Вместо этого R запрашивает большой блок памяти, а затем управляет этим блоком самостоятельно.
- Этот блок называется малым векторным пулом и используется для векторов длиной менее 128 В.
- Он выделяет только векторы длиной 8, 16, 32, 48, 64 или 128 В.
- Для объектов более 128 В память запрашивается в количестве, кратном 8 В.

Выделение блоков памяти



Размер объекта: библиотека pryr

```
pryr :: object_size ()
```

отличается тем, что учитывает память, разделяемую элементами в объекте, и память из окружений, связанных с объектом.

```
pryr :: compare_size ()
```

сравнивает результаты с object.size().

```
library(pryr)
compare_size(mean)
# base pryr
# 1184 1184
mmm = list(mean, mean, mean)
compare_size(mmm)
# base pryr
# 3632 1264
```

Размер объекта при создании копий

При создании копий объекта создаются указатели на существующие объекты, поэтому размер объекта не увеличивается пропорционально числу копий.

Ошибочно рассматривать размеры копий по отдельности.

Чтобы узнать, сколько места они занимают вместе:

```
object_size(mean, mmm)  
# 1,264 B
```

В этом случае объекты вместе занимают столько же места, что и `mmm` отдельно.

Но если общих компонентов нет, то можно сложить размеры отдельных компонентов, чтобы узнать общий размер.

Использование памяти

```
pryr :: mem_used ()
```

сообщает общий размер всех объектов в памяти.

Это число не согласуется с объемом памяти, сообщаемым операционной системой, по ряду причин:

- Включаются только объекты, созданные R,
но не сам интерпретатор R.
- И R, и операционная система ленивы: они не освобождают память
до тех пор, пока она действительно не понадобится.
R может удерживать память,
потому что ОС еще не запросила ее обратно.
- R считает память, занятую объектами, но могут быть пробелы
из-за удаленных объектов.

Изменение памяти во время выполнения кода

```
pryr :: mem_change()
```

сообщает, как изменяется память во время выполнения кода.

Положительные числа - увеличение объема памяти, используемой R, а отрицательные числа - уменьшение.

- Даже операции, которые ничего не делают, занимают немного памяти.
- R отслеживает историю команд.
- Поэтому можно игнорировать все, что меньше нескольких килобайт.

```
mem_change(NULL)  
# 736 B
```

Удаление объектов из окружения

Функция `rm()` удаляет объекты и освобождает часть памяти.

```
mem_used ()  
# 26 MB  
mem_change({x = runif (1e6)})  
# 8.02 MB  
mem_used ()  
# 34.3 MB  
mem_change(rm(x))  
# -8 MB  
mem_used ()  
# 26.3 MB
```

Удаление объектов из окружения

- Можно использовать функцию `rm()` для удаления промежуточных результатов расчета.
- Однако, когда расчеты производятся внутри функции, выводится только конечный результат, а все остальное все равно теряется, поэтому нет необходимости использовать функцию `rm()`.
- Лучше избегать использования промежуточных переменных для хранения результатов.
- Но если это необходимо и только в очень редких случаях, функцию `rm()` следует использовать.
- В более общих случаях помогает сборка мусора.

Освобождение памяти

- R использует сборку мусора (*garbage collection - GC*).
- GC автоматически освобождает память от объектов, которые больше не используется.
- Он делает это, отслеживая, сколько имен указывает на каждый объект, и когда нет имен, указывающих на объект, он удаляет этот объект.
- В R версии 4.0 введен ARC (автоматический подсчет ссылок), который позволяет лучше отслеживать, сколько объектов указывает на один и тот же адрес памяти.
- Как правило, не имеет смысла вызывать сборку мусора `gc()` самостоятельно.
- R автоматически запускает сборку мусора всякий раз, когда ему требуется больше места.

Сборка мусора

```
library(bench)

gc_mark = mark(small={x = runif(100); rm(x)},
               large={x = runif(1e6); rm(x)},
               check=FALSE, memory=TRUE)

# expression   'itr/sec' 'mem_alloc' 'gc/sec' 'n_itr'
#1 small       59510.     3.32KB      23.8    9996
#2 large        47.7      7.63MB     9.54     20
```

Для малого вектора `gc()` запускается один раз на 2500 итераций, для большого вектора один раз на 5 итераций. И в том, и в другом случае за все итерации выполнено 4 сборки мусора.

Утечки памяти

- GC заботится об освобождении памяти от объектов, которые больше не используются.
- Утечка памяти происходит, когда вы продолжаете указывать на объект, не осознавая этого.
- В R две основные причины утечек памяти (они захватывают окружение):
 - формулы,
 - замыкания.

```
f0 = function() {  
  x = runif(1e6)  
  10}  
  
mem_change(y <- f0())  
# 18.4 kB  
compare_size(y)  
# base pryr  
# 56 56
```

Утечки памяти

В функции `f0()` на `x` ссылаются только внутри функции, поэтому, когда функция завершает работу, возвращается память и чистое изменение памяти равно 0.

Функция `f1()` возвращает объекты, которые захватывают окружение, поэтому `x` не освобождается после завершения функции.

```
f1 = function() {  
  x1 = runif(1e6)  
  a ~ b}  
mem_change(y1 <- f1())  
# 8 MB  
compare_size(y1)  
#     base      pryr  
#     728 8000888
```

Модификация на месте

Что происходит с x в следующем коде?

```
x = 1:10  
x[5] = 10  
x  
# [1] 1 2 3 4 10 6 7 8 9 10
```

В зависимости от обстоятельств:

- R изменяет x на месте;
- R делает копию x в новое место, изменяет копию, а затем использует имя x, чтобы указать на новое место.

В примере x изменится на месте. Но если другая переменная также указывает на x, то R скопирует ее в новое место.

Чтобы изучить происходящее более подробно, нам потребуется информация о местоположении переменной в памяти и том, сколько имен указывают на это местоположение.

Подсчет ссылок

- В R версии 4.0 был введен ARC (автоматический подсчет ссылок).
- Ранее R отслеживал количество копий объекта с помощью NRC (именованного подсчета ссылок). Он имел только 3 варианта: 0, 1 и 2.
- ARC может увеличивать и уменьшать эти числа, чтобы лучше отслеживать, сколько объектов указывает на один и тот же адрес памяти.

`pryr :: address`

возвращает расположение объекта в памяти.

`pryr :: refs`

возвращает количество ссылок, указывающих на объект.

Подсчет ссылок

```
z = rnorm(1)
c(address(z), refs(z))
# [1] "0x5607a88f24a8" "1"

y = z
c(address(y), refs(y), refs(z))
# [1] "0x5607a88f24a8" "2"    "2"

v = runif(1)
w = v
refs(v)
# [1] 2
rm(w)
refs(v)
# [1] 1 # R 4.1.0 # [1] 2 # R 3.5.1
```

Подсчет ссылок

- Непримитивные функции, которые касаются объекта, могут увеличивать счетчик ссылок.
- Примитивные функции обычно этого не делают.

```
f = function(z) z
{z = runif(1); f(z); refs(z)}
# [1] 1 # R 4.1.0 # [1] 3 # R 3.5.1
```

```
{v = runif(10); sum(v); refs(v)}
# [1] 2 # R 4.1.0 # [1] 1 # R 3.5.1
```

```
f = function(z) 10
{z = runif(1); f(z); refs(z)}
# [1] 1 # R 4.1.0 # [1] 1 # R 3.5.1
```

Количество ссылок и модификация на месте

- Если `refs(x)` равен 1, модификация будет произведена на месте.
- Если `refs(x)` больше или равно 2, R сделает копию (это гарантирует, что другие указатели на объект не будут затронуты).

```
x = 1:10
y = x
c(address(x), address(y))
# [1] "0x555df57243f8" "0x555df57243f8"
```

```
x[5] = 10
c(address(x), address(y))
# [1] "0x555df5e63c38" "0x555df57243f8"
```

у продолжает указывать на одно и то же место,
расположение x в памяти изменяется.

Отслеживание создания копий

`tracemem()` печатает сообщение каждый раз при копировании отслеживаемого объекта.

```
z = rnorm(10)
tracemem(z)
# [1] "<0x565420881fa8>"
```



```
z[5] = runif(1)
```

```
y = z
z[5] = runif(1)
# tracemem[0x565420881fa8 -> 0x5654208823c8]:
```

Модификация на месте

Как правило, при условии, что объект не упоминается в другом месте, любая примитивная функция замены будет приводить к модификации на месте:

```
[[<-, <-, @<-, $<-, attr<-, attributes<-, class<-,  
dim<-, dimnames<-, names<-, levels<-
```

Эти примитивные функции написаны таким образом, что не увеличивают счетчик ссылок.

Лучше использовать `refs()` и `address()`, чтобы выяснить, когда объекты копируются.

Хотя определить, что копии создаются, несложно, но предотвратить такое поведение - сложная задача.

Циклы и копирование объектов

Циклы `for` в R имеют репутацию медленных.

Часто такая медлительность возникает из-за того, что в цикле изменяется копия объекта вместо того, чтобы производить модификацию на месте.

```
x = data.frame(matrix(runif(100 * 1e4), ncol = 100))
medians = sapply(x, median)

for(i in seq_along(medians)) {
  x[, i] <- x[, i] - medians[i]
}
```

Каждая итерация цикла копирует data frame!

Циклы и копирование объектов

Отследим адреса объекта и число ссылок на него на каждой итерации.

```
for(i in 1:5) {  
  x[, i] <- x[, i] - medians[i]  
  print(c(address(x), refs(x)))  
}  
# [1] "0x564a428571d0" "2"  
# [1] "0x564a42f21640" "2"  
# [1] "0x564a44ef7290" "2"  
# [1] "0x564a4508fd20" "2"  
# [1] "0x564a428b3790" "2"
```

В R 3.5.1 `refs(x)` будет равен 2, в R 4.1.0 он будет равен 1, но адреса объекта в памяти все равно будут меняться на каждой итерации.

На каждой итерации `x` перемещается в новое место потому, что `[<- Data.frame` не является примитивной функцией.

Профилирование памяти

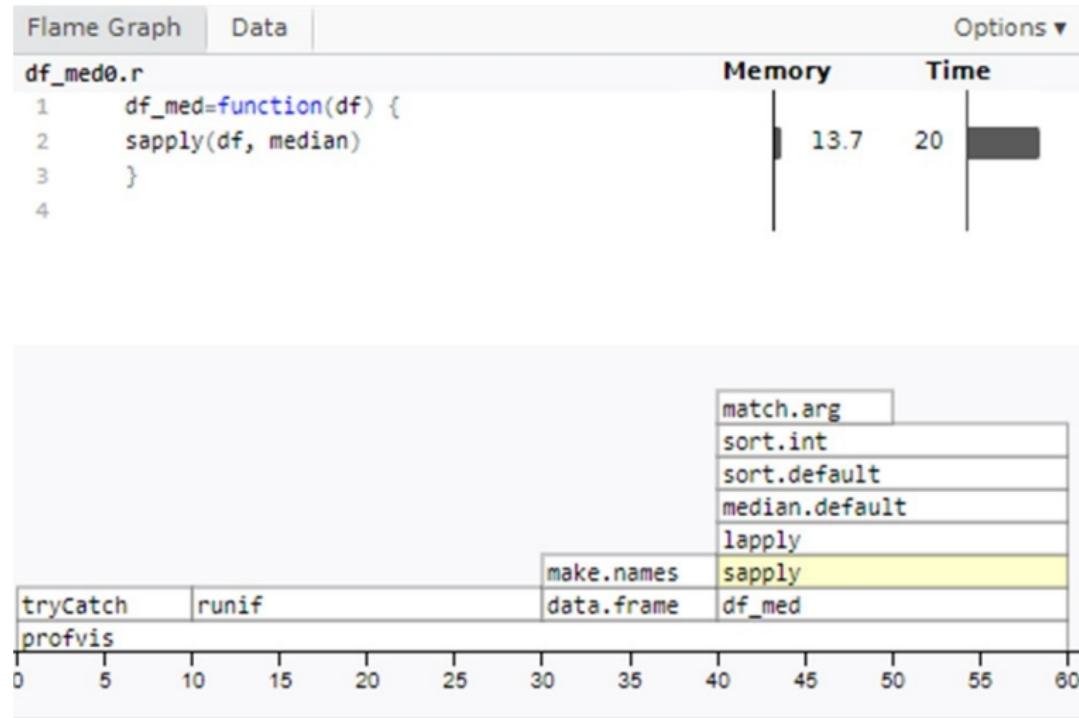
- Функцию `Rprof()` можно использовать для профилирования кода R как по скорости, так и по использованию памяти.
- `Rprof()` в определенные моменты (по умолчанию каждые 0.02 с) записывает в журнал, какие функции в настоящее время находятся в стеке.
- Пакет `profviz` построен на основе `Rprof()`, но предоставляет более полезную сводную информацию.

Профилирование памяти: пример

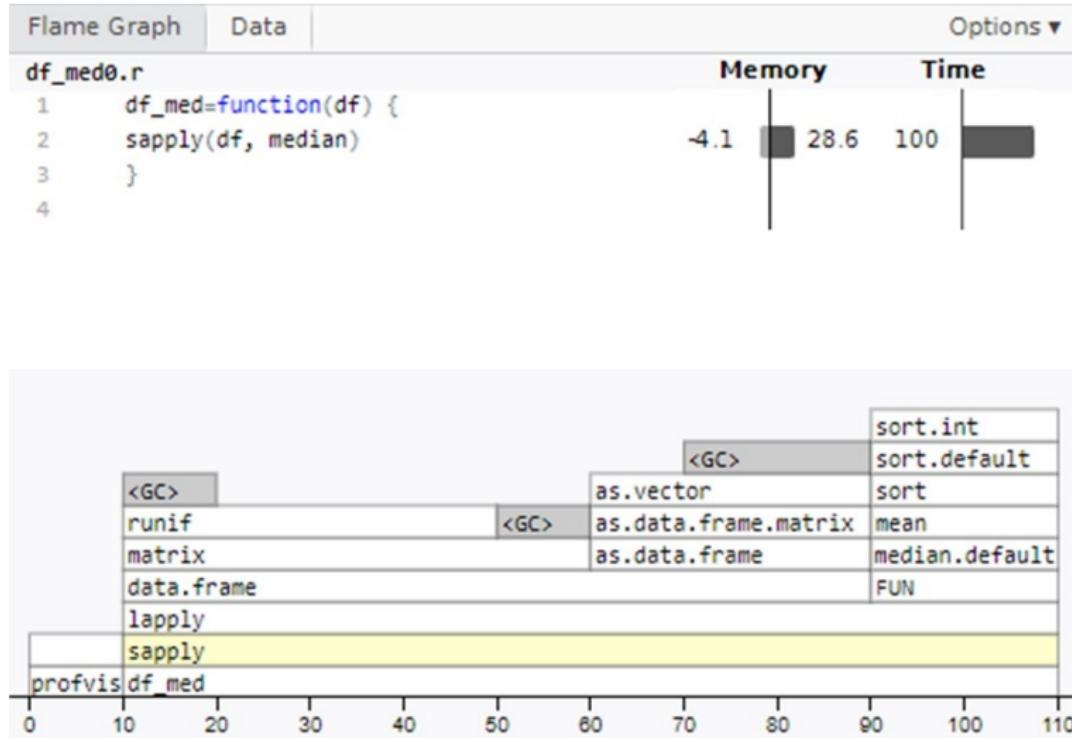
Файл df_med0.r содержит код функции, рассчитывающей медианы для всех столбцов data frame:

```
df_med=function(df) {  
  sapply(df, median)  
}  
  
library(profvis)  
source('df_med0.r')  
set.seed(1005)  
profvis( df_med (  
  data.frame(matrix(runif(100 * 1e4), ncol = 100))  
    )  
)
```

Профилирование памяти: пример R 4.1.0



Профилирование памяти: пример R 3.4.0



Профилирование памяти: пример

- Пример показывает, что функции `sapply()`, `median()`, `data.frame()` по-разному работают в R версиях 4.1.0 и 3.4.0.
- В версии 4.1.0 выделяется 13.7 Mb памяти.
- В версии 3.4.0 выделяется 28.6 Mb памяти и освобождается 4.1 Mb памяти.
- В версии 3.4.0 часто запускается сборщик мусора (`<GC>`).
- Вероятно, это связано с использованием функции `as.data.frame`, которая приводит к многократному созданию копий.

Визуализация данных с использованием ggplot2

Пакет `ggplot2` – современный стандарт для создания графиков в R.
Для этого пакета пишут массу расширений.

- Слои
- `geoms` - визуальные метки, отображающие точки данных.
- Система координат
- `Stat` для визуализации преобразований признака
- Фасетизация
- Визуализация комбинаций признаков

Слои

- Графики `ggplot2` многослойные, то есть строятся они поэтапно, по слоям.
- Сначала указывается датафрейм, с которым мы работаем, и интересующие нас показатели (первый слой).
- Затем указывается тип графика (второй слой).
- Затем настройки для подписей, легенды и прочее (остальные слои).
- Все слои добавляются через `+`
- Для любого графика указывается функция `aes`, сокращенно от `aesthetics`, в качестве аргументов которой задаются переменные интереса (которые хотим отобразить на графике), а также элементы оформления графика, которое непосредственно связано с переменными в датафрейме.

Визуальные метки, отображающие точки данных

Визуализация значений одной переменной.

- `geom_area` - закрашенные области
- `geom_density` - плотность распределения
- `geom_dotplot` - график частот в виде точек
- `geom_freqpoly` - многоугольник частот
- `geom_histogram` - гистограмма
- `geom_bar` - столбчатая диаграмма

Визуальные метки, отображающие точки данных

Визуализация значений двух переменных.

- `geom_point` - диаграмма рассеяния
- `geom_smooth` - непараметрическая оценка кривой
- `geom_text` - добавить названия объектов, метки классов
- `geom_density2d` - проекция двумерной функции плотности
- `geom_boxplot` - ящик с усами
- `geom_violin` - violin plot
- `geom_errorbar` - визуализация ошибок

Система координат

- coord_cartesian - декартова система координат
- coord_fixed - декартова система координат с фиксированным соотношением сторон между x и y
- coord_flip - перевернутая декартова система координат
- coord_polar - полярные координаты
- coord_trans - преобразованные декартовы координаты
- coord_map - картографические проекции из пакетов mapproj, mercator, azequalarea, lagrange и других

Stat для визуализации преобразований признака

Несколько слоев проще указать с помощью функции `stat_`, которая относится к статистическому преобразованию, а не к внешнему виду.

- `stat_ecdf` - эмпирическая функция распределения
- `stat_ellipse` - эллипс рассеяния для данных из нормального распределения
- `stat_unique` - удаляет дубликаты
- `stat_identity` - оставляет данные как они есть
- `stat_summary_bin` - групповые операции над значениями у для каждого уникального (интервала) значения x
- `after_stat` - отображение вычисленных переменных

Фасетизация

Достаточно мощным инструментом анализа данных является фасетизация, которая позволяет разбивать графики на основе какой-то переменной.

- `facet_wrap` - отдельные области графика для каждой категории одного или нескольких признаков
- `facet_grid (. ~ x)`
фасеты в столбцах по переменной x
- `facet_grid (y ~ .)`
фасеты в строках по переменной y
- `facet_grid (y ~ x)`
фасеты по сетке переменных x и y
- `stat_summary_bin` - групповые операции над значениями y для каждого уникального (интервала) значения x
- `after_stat` - отображение вычисленных переменных

Визуализация комбинаций признаков

Способы визуализации отношений между многими признаками:

- `geom_parallel_sets` - потоковая Диаграмма (Sankey diagram) визуализирует попарные отношения между переменными
- `UpSetR :: upset` - график UpSet потенциально может визуализировать все возможные комбинации и является хорошей альтернативой диаграмме Вена, с большим количеством переменных

Программные системы статистического анализа

Тимофеева Анастасия Юрьевна

3 ноября 2021

Структура лекции

- Описательная статистика
- Аномальные наблюдения
- Корреляционный анализ
- Регрессионный анализ
- Устойчивые методы
- Модели бинарного выбора
- Подбор структуры модели

Считаем данные

```
ngs=read.csv2("~/Данные/ngs.csv")
```

Выделим в данных факторы

sapply применяет заданную функцию ко всем столбцам таблицы данных

is.factor возвращает TRUE, если переменная - фактор, FALSE - иначе

Ind_fact=sapply(ngs,is.factor)

Описательная статистика

Стандартные средства (для каждой статистики использовать свою функция)

```
sapply(ngs[, !Ind_fact], mean, na.rm=TRUE)
```

```
##   latitude longitude  square1  square2  square3    stage
## 55.006430  82.944374 57.826892 34.438654  9.413700 4.772577
##   stages   price
## 8.479748 3816.819153
```

Описательная статистика

Стандартные средства (для каждой статистики использовать свою функция)

```
sapply(ngs[, !Ind_fact], sd, na.rm=TRUE)
```

```
## latitude longitude square1 square2 square3
## 6.374533e-02 6.576442e-02 3.099927e+01 1.934243e+01 5.479476e+00
## stage stages price
## 3.413925e+00 4.116010e+00 1.002552e+04
```

Описательная статистика

Стандартные средства (для каждой статистики использовать свою функция)

sapply(ngs[, Ind_fact], table)

```
## $rooms
##
##      1     2     3     4     5    6+ ком
## 3158 4022 3605  917  128   22 1159
##
## $type
##
##      б     бмк    бт    гб     д     к    кмк    м    мжбкк    п     пб     пк
##  26     13    10     2    26  7094   315    89     50   5244     3     26
##      с     ш
##     15     80
```

Пакет psych - вся статистика сразу!

```
# install.packages("psych") # если не установлен
library(psych)
describe(nsgs)
```

```
##      vars   n    mean     sd median trimmed    mad    min
## latitude  1 13007  55.01  0.06  55.02  55.01  0.05  54.81
## longitude 2 13007  82.94  0.07  82.94  82.94  0.05  82.78
## rooms*    3 13011   2.66  1.65   2.00  2.35  1.48  1.00
## square1   4 12980  57.83 31.00  54.00  53.95 19.27  8.00
## square2   5 12191  34.44 19.34  30.60  31.94 17.20  6.00
## square3   6 11574   9.41  5.48   8.00  8.39  2.97  1.00
## stage     7 13011   4.77  3.41   4.00  4.33  2.97  1.00
## stages    8 13011   8.48  4.12   9.00  7.91  5.93  1.00
## type*     9 12993   7.72  2.05   6.00  7.63  0.00  1.00
## price    10 13011 3816.82 10025.52 3000.00 3194.32 1186.08 350.00
##                  max    range skew kurtosis    se
## latitude    55.18  0.37 -0.70   0.58  0.00
## longitude  83.13  0.35  0.67   0.35  0.00
## rooms*     7.00   6.00  1.45   1.66  0.01
## square1   601.00 593.00  3.38  29.03  0.27
## square2  340.00 334.00  2.97  21.74  0.18
## square3  111.00 110.00  4.59  40.56  0.05
## stage     24.00  23.00  1.38   2.49  0.03
## stages    27.00  26.00  1.26   2.16  0.04
## type*    14.00  13.00  0.35  -1.16  0.02
## price   850000.00 849650.00 70.04 5489.41 87.89
```

Статистика в разрезе категориальной переменной

```
describeBy(ngs, ngs$rooms, skew=FALSE, ranges=FALSE)
```

```
## $ 1`  
##      vars   n   mean     sd    se  
## latitude  1 3156  55.01  0.06  0.00  
## longitude 2 3156  82.94  0.07  0.00  
## rooms*    3 3158   1.00  0.00  0.00  
## square1   4 3154  35.23  9.35  0.17  
## square2   5 2955  18.29  5.18  0.10  
## square3   6 2796   8.20  3.04  0.06  
## stage     7 3158   4.97  3.76  0.07  
## stages    8 3158   9.02  4.58  0.08  
## type*    9 3153   7.64  2.04  0.04  
## price    10 3158 2456.53 822.74 14.64  
##  
## $ 2`  
##      vars   n   mean     sd    se  
## latitude  1 4021  55.01  0.07  0.00  
## longitude 2 4021  82.94  0.06  0.00  
## rooms*    3 4022   2.00  0.00  0.00  
## square1   4 4015  51.62 11.91  0.19  
## square2   5 3849  31.37  6.39  0.10  
## square3   6 3796   8.50  3.83  0.06  
## stage     7 4022   4.58  3.41  0.05  
## stages    8 4022   8.20  4.14  0.07  
## type*    9 4017   7.76  2.06  0.03  
## price    10 4022 3399.78 1367.38 21.56  
##  
## $ 3`  
##      vars   n   mean     sd    se  
## latitude  1 3605  55.01  0.06  0.00  
## longitude 2 3605  82.95  0.06  0.00  
## rooms*    3 3605   3.00  0.00  0.00  
## square1   4 3600  73.65 21.26  0.35  
## square2   5 3431   46.64 11.94  0.20  
## square3   6 3390  10.46  6.34  0.11  
## stage     7 3605   4.99  3.22  0.05  
## stages    8 3605   8.84  3.93  0.07  
## type*    9 3603   7.88  2.04  0.03  
## price    10 3605 4796.19 2828.08 47.10  
##  
## $ 4`  
##      vars   n   mean     sd    se  
## latitude  1  916  55.01  0.06  0.00  
## longitude 2  916  82.94  0.06  0.00  
## rooms*    3  917   4.00  0.00  0.00  
## square1   4  916  98.91 35.99  1.19  
## square2   5  863  63.27 20.07  0.68  
## square3   6  842  12.71  9.01  0.31  
## stage     7  917   5.12  3.63  0.12  
## stages    8  917   9.32  3.52  0.12  
## type*    9  914   7.72  1.99  0.07  
## price    10 917 7510.36 22786.91 752.49  
##  
## $ 5`  
##      vars   n   mean     sd    se  
## latitude  1 128   55.00  0.06  0.01  
## longitude 2 128   82.94  0.06  0.01  
## rooms*    3 128   5.00  0.00  0.00  
## square1   4 128  155.86 68.90  6.09  
## square2   5 119  97.71 40.65  3.73  
## square3   6 110  16.12 10.56  1.01  
## stage     7 128   6.89  3.38  0.30  
## stages    8 128   9.38  2.87  0.25  
## type*    9 128   7.14  1.80  0.16  
## price    10 128 11786.84 8945.67 790.69  
##  
## $ 6+`  
##      vars   n   mean     sd    se  
## latitude  1 22   55.00  0.06  0.01  
## longitude 2 22   82.95  0.06  0.01  
## rooms*    3 22   6.00  0.00  0.00  
## square1   4 22  289.25 139.23 29.68  
## square2   5 17  171.34  86.08 20.88  
## square3   6 17  29.54  24.89  6.04  
## stage     7 22   6.45  4.50  0.96  
## stages    8 22   9.59  5.03  1.07  
## type*    9 22   6.27  1.70  0.36  
## price    10 22 28002.27 25929.07 5528.10  
##  
## $kom`  
##      vars   n   mean     sd    se  
## latitude  1 1159  55.00  0.06  0.00  
## longitude 2 1159  82.94  0.07  0.00  
## rooms*    3 1159   7.00  0.00  0.00  
## square1   4 1145  43.81 28.97  0.86  
## square2   5  957  16.60  6.45  0.21  
## square3   6  623   8.53  3.26  0.13  
## stage     7 1159   3.70  2.35  0.07  
## stages    8 1159   6.08  2.55  0.07  
## type*    9 1156   7.34  2.09  0.06  
## price    10 1159 1662.70 24941.62 732.63  
##  
## attr("call")  
## by.data.frame(data = x, INDICES = group, FUN = describe, type = type,  
##   skew = FALSE, ranges = FALSE)
```

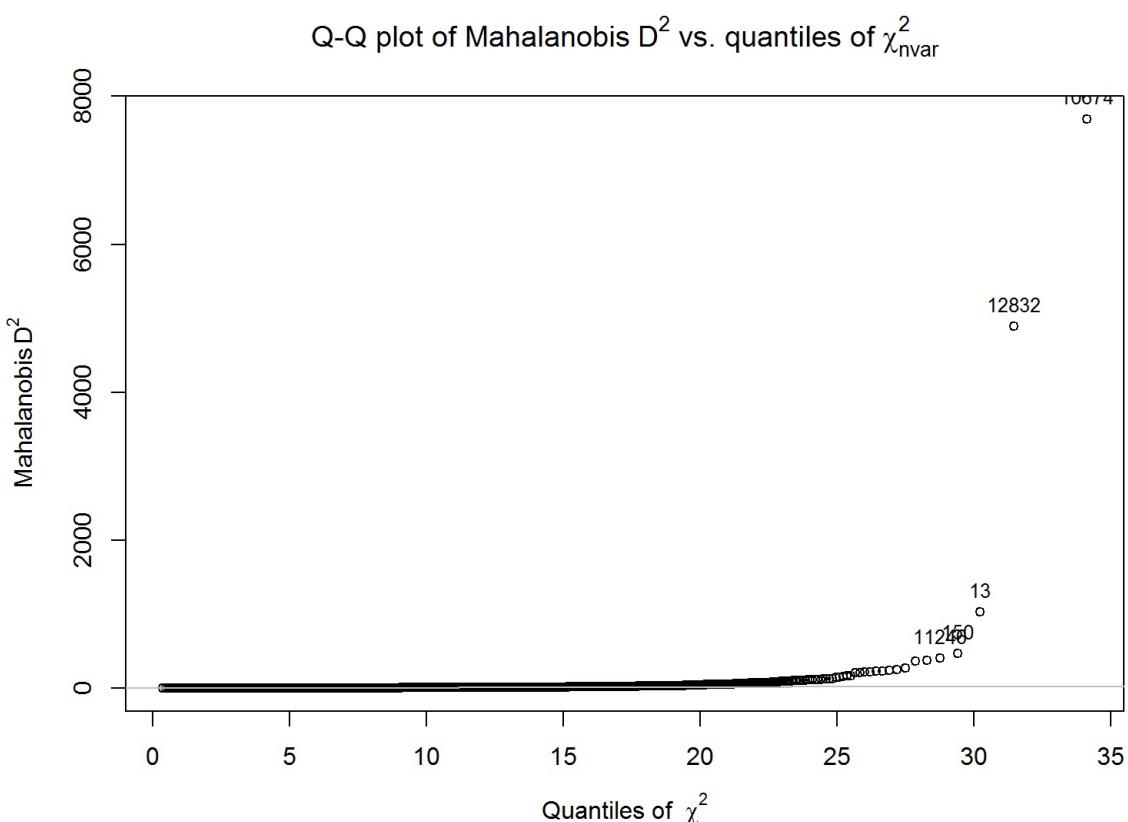
Стандартизация (нормирование и центрирование) переменных

```
ngs_st=scale(ngs[, !Ind_fact])
```

Поиск выбросов в многомерном массиве

Графическое отображение отклонения точек от многомерного нормального эллипсоида

```
out_obs=outlier(ngs[,!Ind_fact],cex=.8)
```



Гистограмма расстояний

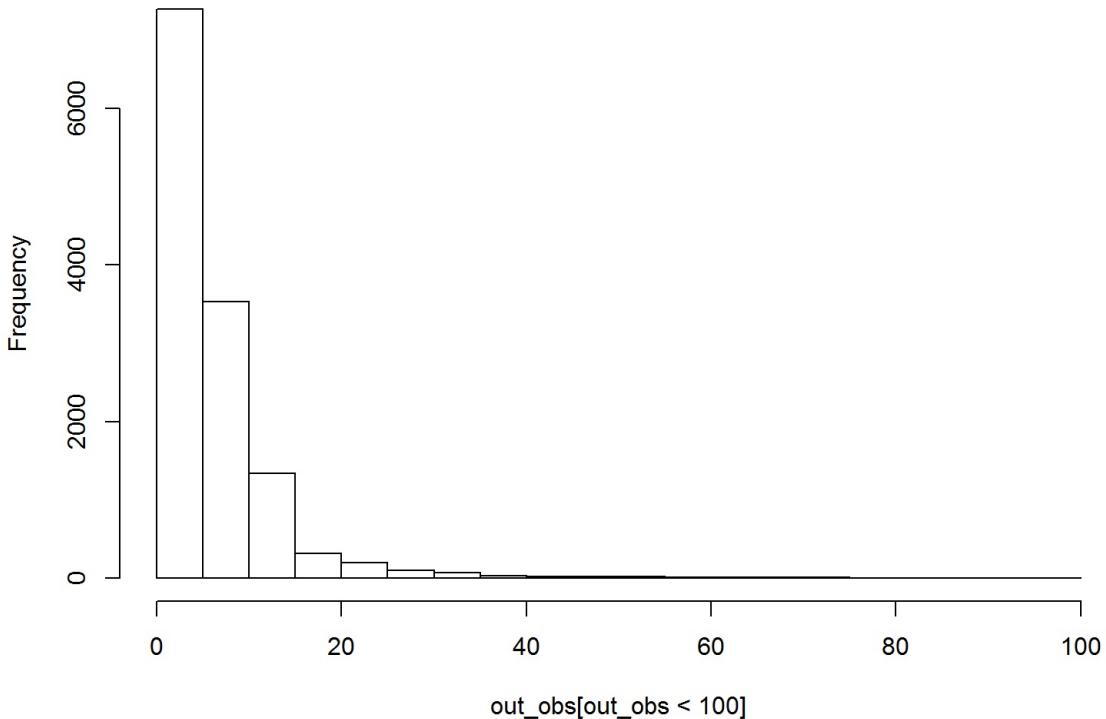
```
hh=hist(out_obs[out_obs<100],5*0:20)
```

```
## [1] 7273 3527 1339 315 196 103 69 34 22 20 9 7 11  
## [15] 12 3 4 2 3 4 69 34 22 20 9 7 11
```

```
hh$breaks
```

```
## [1] 0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80  
## [18] 85 90 95 100
```

Histogram of out_obs[out_obs < 100]



Создаем индикатор для неаномальных наблюдений

```
ind_no_out<=out_obs<=20
```

Удаляем лишние объекты

```
rm{out_obs}  
rm{hh}  
rm{ngs_st}
```

Корреляционный анализ (средствами пакета psych)

между количественными переменными

Коэффициент корреляции Пирсона

```
lowerCor(ngs[,!Ind_fact]) # исходный вариант
```

```
##      lattd lngtd squrl squr2 squr3 stage stags price
## latitude   1.00
## longitude -0.27  1.00
## square1   0.06 -0.01  1.00
## square2   0.05  0.00  0.88  1.00
## square3   0.06  0.00  0.65  0.46  1.00
## stage      0.08  0.00  0.20  0.14  0.26  1.00
## stages     0.12  0.03  0.24  0.18  0.35  0.60  1.00
## price      0.03 -0.01  0.31  0.78  0.65  0.06  0.08  1.00
```

Корреляционный анализ (средствами пакета psych)

между количественными переменными

Коэффициент корреляции Пирсона

```
lowerCor(ngs[ind_no_out,!Ind_fact]) # без выбросов
```

```
##      lattd lngtd squrl squr2 squr3 stage stags price
## latitude   1.00
## longitude -0.25  1.00
## square1   0.07 -0.02  1.00
## square2   0.06  0.00  0.87  1.00
## square3   0.09  0.00  0.64  0.43  1.00
## stage      0.07  0.01  0.18  0.12  0.29  1.00
## stages     0.11  0.04  0.27  0.17  0.45  0.58  1.00
## price      0.13 -0.04  0.78  0.75  0.63  0.24  0.36  1.00
```

Корреляционный анализ (средствами пакета psych)

между количественными переменными

Коэффициент корреляции Спирмена

```
lowerCor(ngs[ , !Ind_fact], method = "spearman")
```

Корреляционный анализ (средствами пакета psych)

между количественными переменными

Коэффициент корреляции Кендалла

```
lowerCor(ngs[ ,!Ind_fact], method="kendall")
```

Корреляционный анализ стандартными средствами

Менее приятный вид

```
cor(ngs[ind_no_out,!Ind_fact],method="spearman",use = "pairwise.c
```

```
##           latitude   longitude   square1   square2   square3
## latitude  1.000000000 -0.0006874851  0.07255940  0.06142538  0.09121618
## longitude -0.0006874851  1.000000000  0.01114777  0.01956774  0.05705990
## square1   0.0725593959  0.0111477724  1.00000000  0.83524353  0.56778992
## square2   0.0614253848  0.0195677420  0.83524353  1.00000000  0.33478340
## square3   0.0912161840  0.0570599039  0.56778992  0.33478340  1.00000000
## stage     0.0577290282  0.0171057908  0.16089257  0.11311641  0.30252814
## stages    0.0993481840  0.0782818018  0.28453573  0.17304394  0.58236619
## price     0.1954908745 -0.0209829387  0.74033966  0.76975304  0.56763098
##           stage   stages   price
## latitude  0.05772903  0.09934818  0.19549007
## longitude 0.01710579  0.07828180 -0.02098294
## square1   0.16089257  0.28453573  0.74033966
## square2   0.11311641  0.17304394  0.76975304
## square3   0.30252814  0.58236619  0.56763098
## stage     1.00000000  0.49093617  0.26314861
## stages    0.49093617  1.00000000  0.44193966
## price     0.26314861  0.44193966  1.00000000
```

Корреляционный анализ с категориальными переменными

Таблица сопряженности

```
cross_table=table(ngs[,Ind_fact][,1],ngs[,Ind_fact][,2])  
cross_table
```

```
##  
##       6   бмк    бт    гб    д     к   кмк    м   мжбкк    п    пб    пк    с  
## 1   10    7    4    1   10  1720   120   22   14 1223    1    5    3  
## 2    7    2    3    0     8 2163   91   26   19 1652    1   11    5  
## 3    6    3    1    1    2 1825   84   25   12 1621    1    3    5  
## 4    0    1    1    0     0 501   16   12    4 372    0    6    1  
## 5    0    0    0    0     0 89    3    1    1 33    0    1    0  
## 6+   1    0    0    0     0 17    1    1    0 2    0    0    0  
## ком   2    0    1    0     6 779    0    2    0 341    0    0    1  
##  
##      ш  
## 1   13  
## 2   29  
## 3   14  
## 4   0  
## 5   0  
## 6+  0  
## ком 24
```

Корреляционный анализ с категориальными переменными

Удалим уровни с большим числом нулей в таблице

```
mater_del=c(1:5,11:14)
cross_table1=cross_table[,-mater_del]
```

Корреляционный анализ с категориальными переменными

Полихорическая корреляция

polychoric(cross_table1)

```
## [1] "You seem to have a table, I will return just one correlation."
```

```
## $rho
## [1] -0.04101978
##
## $objective
## [1] 2.360105
##
## $tau.row
##      1      2      3      4      5      6+
## -0.6990487 0.1285059 0.9540540 1.2856604 1.3447517 1.3552310
##
## $tau.col
##      K      КМК      М      МЖБКК
## 0.1372043 0.1999220 0.2177465 0.2279914
```

Регрессионный анализ

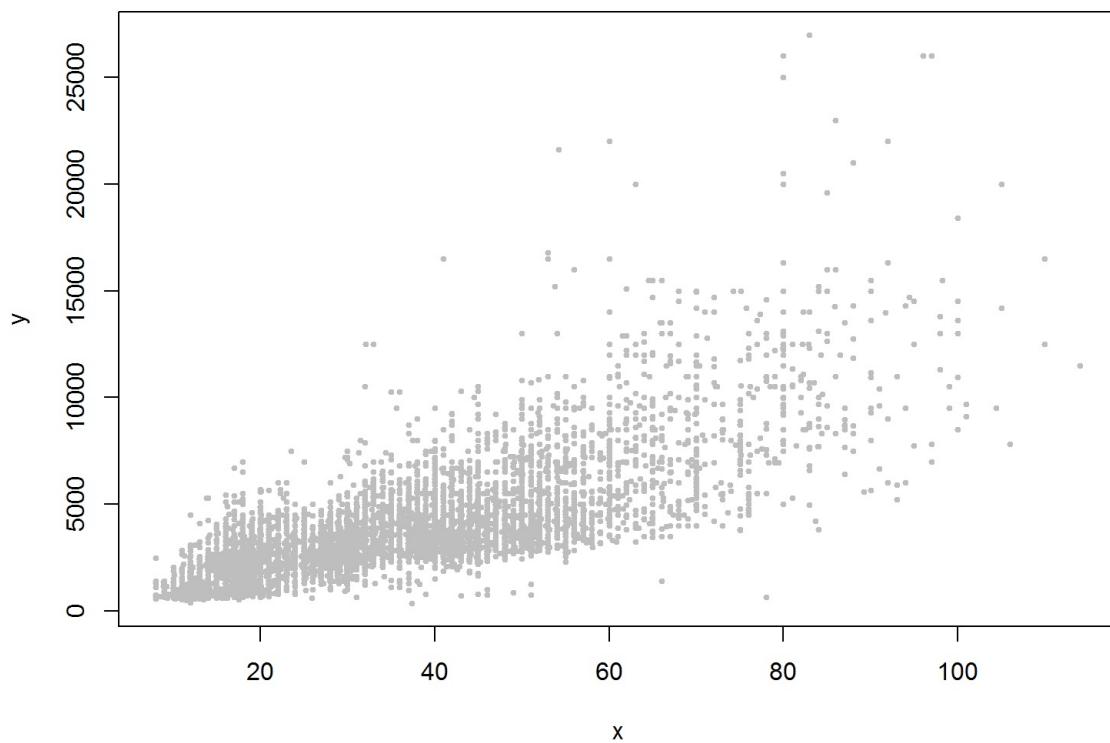
Построим модель, описывающую зависимость цены от жилой площади

Переменные

```
x=ngs$square2[ind_no_out]  
y=ngs$price[ind_no_out]
```

Корреляционное поле

```
plot(x,y,col="gray",pch=19,cex=0.5)
```



Стандартная функция множественной регрессии

Линейная зависимость

lm(y~x)

```
## 
## Call:
## lm(formula = y ~ x)
## 
## Coefficients:
## (Intercept)          x
## -2.633        103.574
```

Стандартная функция множественной регрессии

Линейная зависимость

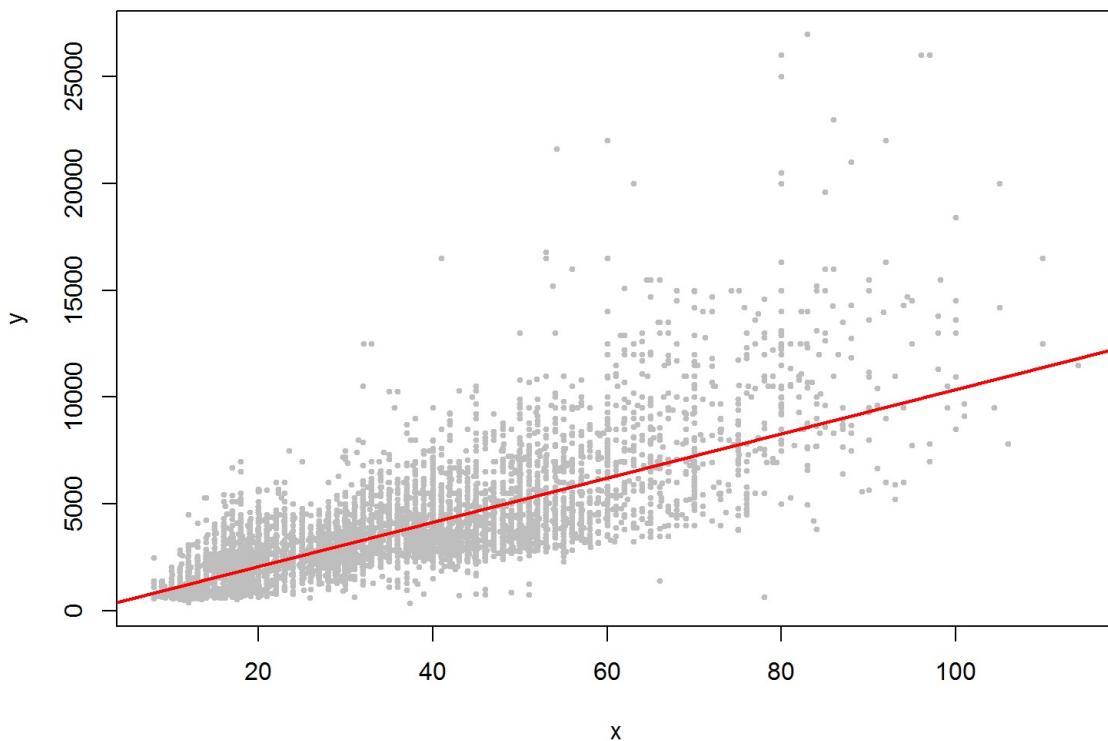
summary(lm(y~x))

```
## 
## Call:
## lm(formula = y ~ x)
## 
## Residuals:
##   Min     1Q Median     3Q    Max 
## -7426.1 -774.8 -133.1  531.3 18406.0 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -2.6330   30.6028 -0.086   0.931    
## x            103.5740   0.8368 123.768 <2e-16 ***  
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 1398 on 11718 degrees of freedom
## (734 observations deleted due to missingness)
## Multiple R-squared:  0.5666, Adjusted R-squared:  0.5665 
## F-statistic: 1.532e+04 on 1 and 11718 DF,  p-value: < 2.2e-16
```

Стандартная функция множественной регрессии

Линейная зависимость

```
plot(x, y, col="gray", pch=19, cex=0.5)
abline(lm(y~x)$coef, col="red", lwd=2)
```



Стандартная функция множественной регрессии

Квадратичная зависимость

lm(y~x+I(x^2))

```
## 
## Call:
## lm(formula = y ~ x + I(x^2))
## 
## Coefficients:
## (Intercept)          x      I(x^2)
## 1363.4121     22.7726     0.9818
```

функция I() возвращает аргументу математический смысл

Стандартная функция множественной регрессии

Квадратичная зависимость

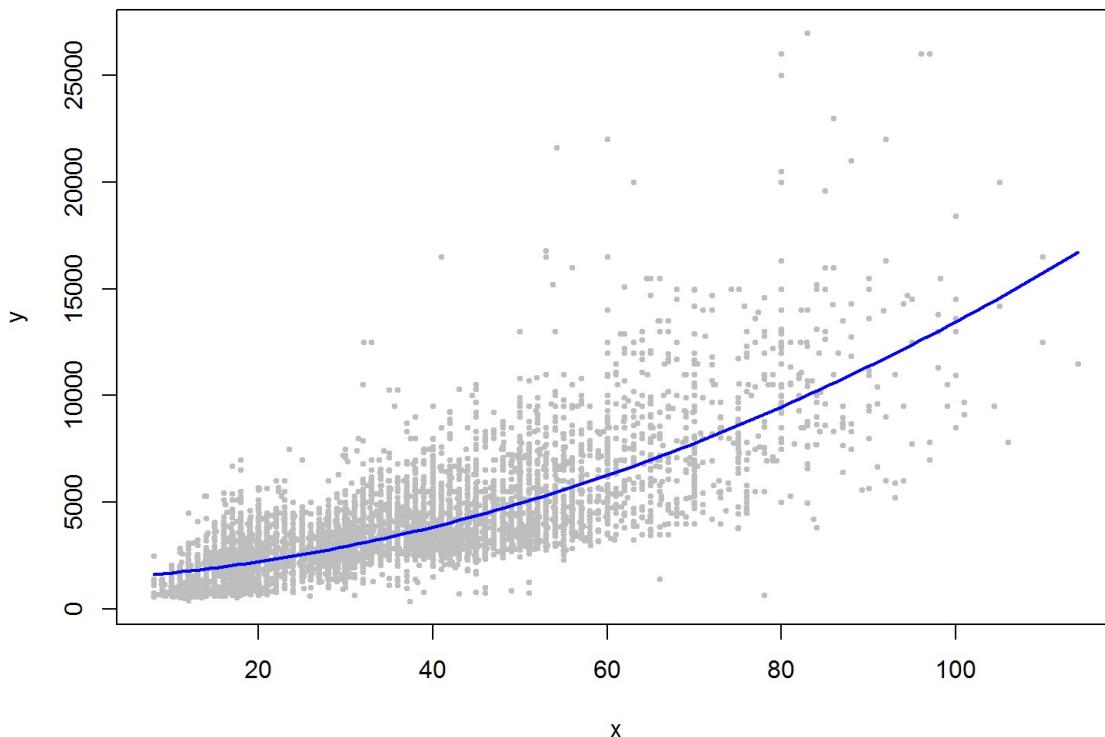
summary(lm(y~x+I(x^2)))

```
## 
## Call:
## lm(formula = y ~ x + I(x^2))
## 
## Residuals:
##   Min     1Q Median     3Q    Max 
## -8462.7 -747.4 -109.5  465.7 16983.1 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 1.363e+03 5.574e+01 24.461 < 2e-16 ***
## x           2.277e+01 2.909e+00  7.827 5.41e-15 ***
## I(x^2)      9.818e-01 3.396e-02 28.912 < 2e-16 ***  
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 1351 on 11717 degrees of freedom
## (734 observations deleted due to missingness)
## Multiple R-squared:  0.5954, Adjusted R-squared:  0.5954 
## F-statistic:  8623 on 2 and 11717 DF,  p-value: < 2.2e-16
```

Стандартная функция множественной регрессии

Квадратичная зависимость

```
plot(x,y,col="gray",pch=19,cex=0.5)
t2=lm(y~x+x^2)$coef #запишем коэффициенты в отдельный вектор
curve(t2[1]+t2[2]*x+t2[3]*x^2,col="blue",lwd=2,add=TRUE)
```



Стандартная функция множественной регрессии

Интервальный прогноз отклика

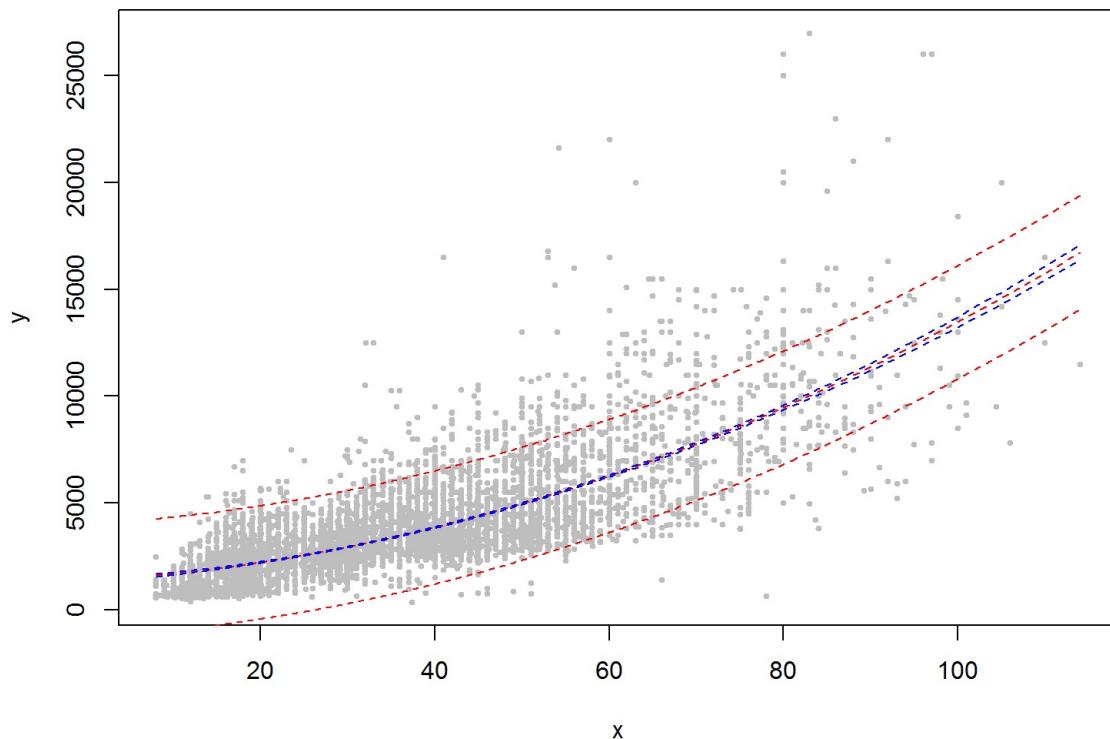
```
model=lm(y~x+I(x^2))
confidence_int=predict.lm(model,interval="confidence")
predict_int=predict.lm(model,interval="prediction")
```

```
## Warning in predict.lm(model, interval = "prediction"): predictions on current data refer to _future_ responses
```

Стандартная функция множественной регрессии

Интервальный прогноз отклика

```
plot(x,y,col="gray",pch=19,cex=0.5)
ind=order(model$modell$x)
for(i in 1:3){
  lines(model$modell$x[ind],confidence.int[ind,i],col="blue",lty=2)
  lines(model$modell$x[ind],predict_int[ind,i],col="red",lty=2)
}
```



Построение зависимости от категориальной переменной

Восстановим зависимость между ценой и числом комнат

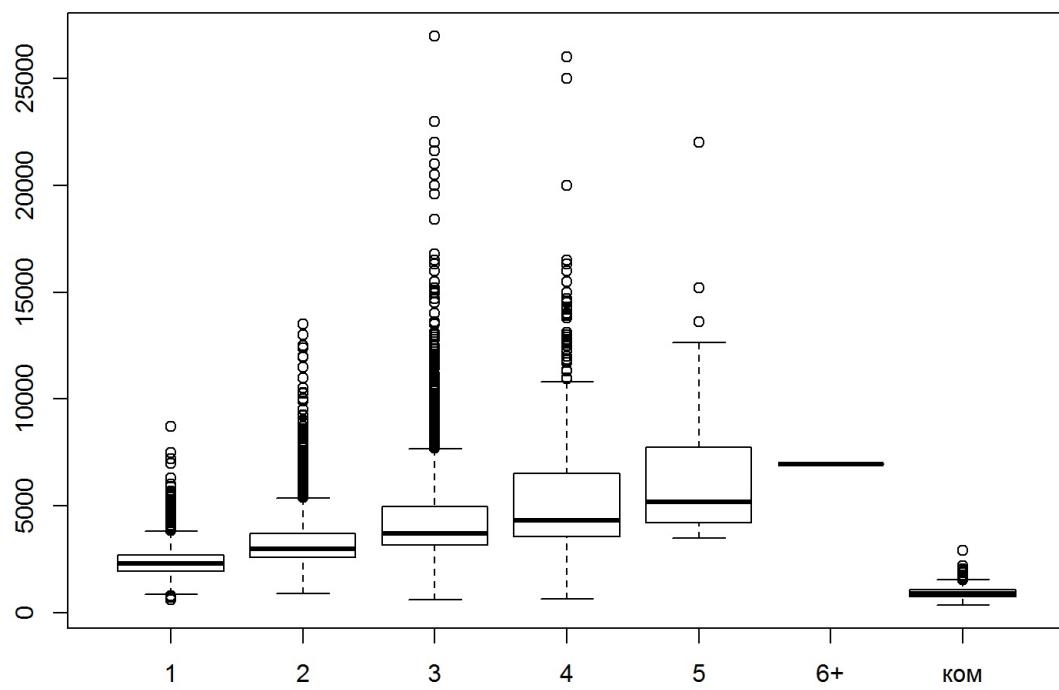
```
x=nqss$rooms{ind_no_out}
y=nqss$price{ind_no_out}
summary(lm(y~x))
```

```
## 
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##   Min     1Q   Median     3Q    Max 
## -4918.4 -909.1 -280.9  260.3 22440.9 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 2430.89    31.52  77.113 < 2e-16 ***
## x2          932.74    42.05  22.181 < 2e-16 ***
## x3         2128.21    43.45  48.975 < 2e-16 ***
## x4         3137.47    69.74  44.988 < 2e-16 ***
## x5         4026.73   211.87 19.066 < 2e-16 ***
## x6+        4519.11  1239.90   3.645 0.000269 ***  
## xxom       -1491.20   61.66 -24.183 < 2e-16 ***
## ---      
## Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 1753 on 12447 degrees of freedom
## Multiple R-squared:  0.3267, Adjusted R-squared:  0.3264 
## F-statistic: 1807 on 6 and 12447 DF, p-value: < 2.2e-16
```

Построение зависимости от категориальной переменной

Наличие зависимости подтверждает и анализ ящиков с усами

`plot(x, y)`



Оценка эффектов нескольких категориальных переменных и их взаимодействий

```
x1=nqs$rooms[ind_no_out]
x2=nqs$type[ind_no_out]
y=nqs$price[ind_no_out]
summary(lm(y~(x1+x2)^2))
```

```
## 
## Call:
## lm(formula = y ~ (x1 + x2)^2)
## 
## Residuals:
##   Min     1Q Median     3Q    Max 
## -4784.8 -651.3 -123.7  381.8 21481.8 
## 
## Coefficients: (35 not defined because of singularities)
##               Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 2403.000   496.838   4.837 1.34e-06 ***
## x12         1604.000   774.265   2.072 0.03832 *  
## x13         3618.500   811.332   4.460 8.27e-06 *** 
## x14         3906.667   1814.194   2.153 0.03131 *  
## x15         2795.000   1721.096   1.624 0.10441  
## x16+        4796.061   1111.871   4.314 1.62e-05 *** 
## x1ком       -1645.500   1216.999  -1.352 0.17637  
## x2бмк       345.571    774.265   0.446  0.65537  
## x2бт       -923.000    929.498  -0.993  0.32072  
## x2гб      -533.000   1647.824  -0.323  0.74635  
## x2д        -665.000   702.634  -0.946  0.34394  
## x2к         198.268   498.310   0.398  0.69073  
## x2кмк      696.266   522.597   1.332  0.18278  
## x2м         161.850   608.499   0.266  0.79026  
## x2мбкк     497.455   686.480   0.725  0.46868  
## x2п        -249.061   498.865   0.499  0.61761  
## x2пб      -203.000   1647.824  -0.123  0.90196  
## x2пк       102.000   860.548   0.119  0.90565  
## x2с         1480.333   1034.250   1.431  0.15237  
## x2ш        -871.846   660.856  -1.319  0.18710  
## x12:x2бмк  2897.429   1478.636   1.960  0.05007 .  
## x13:x2бмк  -1313.071   1498.378  -0.876  0.38087  
## x14:x2бмк  -555.238   2472.330  -0.225  0.82231  
## x15:x2бмк  NA        NA        NA        NA      
## x16+:x2бмк NA        NA        NA        NA      
## x1ком:x2бмк NA        NA        NA        NA      
## x12:x2бт   -300.667   1428.087  -0.211  0.83325  
## x13:x2бт   -1698.500   1934.904  -0.878  0.38006  
## x14:x2бт   -436.667   2525.252  -0.173  0.86272  
## x15:x2бт   NA        NA        NA        NA      
## x16+:x2бт  NA        NA        NA        NA      
## x1ком:x2бт  2165.500   2136.979   1.013  0.31091  
## x12:x2гб   NA        NA        NA        NA      
## x13:x2гб   -2288.500   2365.420  -0.967  0.33332  
## x14:x2гб   NA        NA        NA        NA      
## x15:x2гб   NA        NA        NA        NA      
## x16+:x2гб  NA        NA        NA        NA      
## x1ком:x2гб  NA        NA        NA        NA      
## x12:x2д    -1394.500   1074.660  -1.298  0.19444  
## x13:x2д    -3481.500   1462.650  -2.380  0.01732 *  
## x14:x2д    NA        NA        NA        NA      
## x15:x2д    NA        NA        NA        NA      
## x16+:x2д   NA        NA        NA        NA      
## x1ком:x2д   755.333   1462.650   0.516  0.60557  
## x12:x2к    -531.561   775.960  -0.685  0.49333  
## x13:x2к    -701.523   813.131  -0.863  0.38830  
## x14:x2к    539.591   1816.350   0.297  0.76641  
## x15:x2к    2545.275   1741.885   1.461  0.14398  
## x16+:x2к   NA        NA        NA        NA      
## x1ком:x2к   -4.084   1218.998  -0.003  0.99733  
## x12:x2кмк  187.076   809.846   0.231  0.81732  
## x13:x2кмк  847.030   854.537   0.991  0.32160  
## x14:x2кмк  3621.401   1895.214   1.911  0.05605 .  
## x15:x2кмк  NA        NA        NA        NA      
## x16+:x2кмк NA        NA        NA        NA      
## x1ком:x2кмк NA        NA        NA        NA      
## x12:x2м    29.025   908.716   0.032  0.97452  
## x13:x2м    -932.524   942.873  -0.989  0.32267  
## x14:x2м    -980.608   1907.651  -0.514  0.60723  
## x15:x2м    2440.150   2356.708   1.035  0.30050  
## x16+:x2м   NA        NA        NA        NA      
## x1ком:x2м   -169.350   1684.858  -0.101  0.91994  
## x12:x2мбк  1733.045   989.027   1.752  0.07975 .  
## x13:x2мбк  -946.080   1091.433  -0.867  0.38606  
## x14:x2мбк  412.879   2082.915   0.198  0.84287  
## x15:x2мбк  NA        NA        NA        NA      
## x16+:x2мбк NA        NA        NA        NA      
## x1ком:x2мбк NA        NA        NA        NA      
## x12:x2н    -891.588   776.532  -1.148  0.25092  
## x13:x2н    -2336.359   813.514  -2.872  0.00409 ** 
## x14:x2н    -2145.736   1816.583  -1.181  0.23755  
## x15:x2н    -70.758   1743.270  -0.041  0.96762  
## x16+:x2н   NA        NA        NA        NA      
## x1ком:x2н   408.253   1220.876   0.334  0.73809  
## x12:x2н6   -1114.000   2352.964  -0.473  0.63590  
## x13:x2н6   -2318.500   2365.420  -0.980  0.32702  
## x14:x2н6   NA        NA        NA        NA      
## x15:x2н6   NA        NA        NA        NA      
## x16+:x2н6  NA        NA        NA        NA      
## x1ком:x2н6  NA        NA        NA        NA      
## x12:x2нк   -904.273   1147.863  -0.788  0.43084  
## x13:x2нк   -2476.833   1405.269  -1.763  0.07800 .  
## x14:x2нк   -2331.667   2048.514  -1.138  0.25505  
## x15:x2нк   NA        NA        NA        NA      
## x16+:x2нк  NA        NA        NA        NA      
## x1ком:x2нк  NA        NA        NA        NA      
## x12:x2с    -2239.333   1384.199  -1.618  0.18573  
## x13:x2с    -3151.833   1405.269  -2.243  0.02492 *  
## x14:x2с    NA        NA        NA        NA      
## x15:x2с    NA        NA        NA        NA      
## x16+:x2с   NA        NA        NA        NA      
## x1ком:x2с   -1037.833   2184.579  -0.475  0.63474  
## x12:x2ш   -597.912   935.141  -0.639  0.52259  
## x13:x2ш   166.775   1012.157   0.165  0.86913  
## x14:x2ш   NA        NA        NA        NA      
## x15:x2ш   NA        NA        NA        NA      
## x16+:x2ш   NA        NA        NA        NA      
## x1ком:x2ш   1025.868   1333.527   0.769  0.44174  
## ... 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1571 on 12374 degrees of freedom
##   (17 observations deleted due to missingness)
## Multiple R-squared:  0.4618, Adjusted R-squared:  0.4591
## F-statistic: 171.3 on 62 and 12374 DF,  p-value: < 2.2e-16
```

Оценка эффектов нескольких категориальных переменных и их взаимодействий

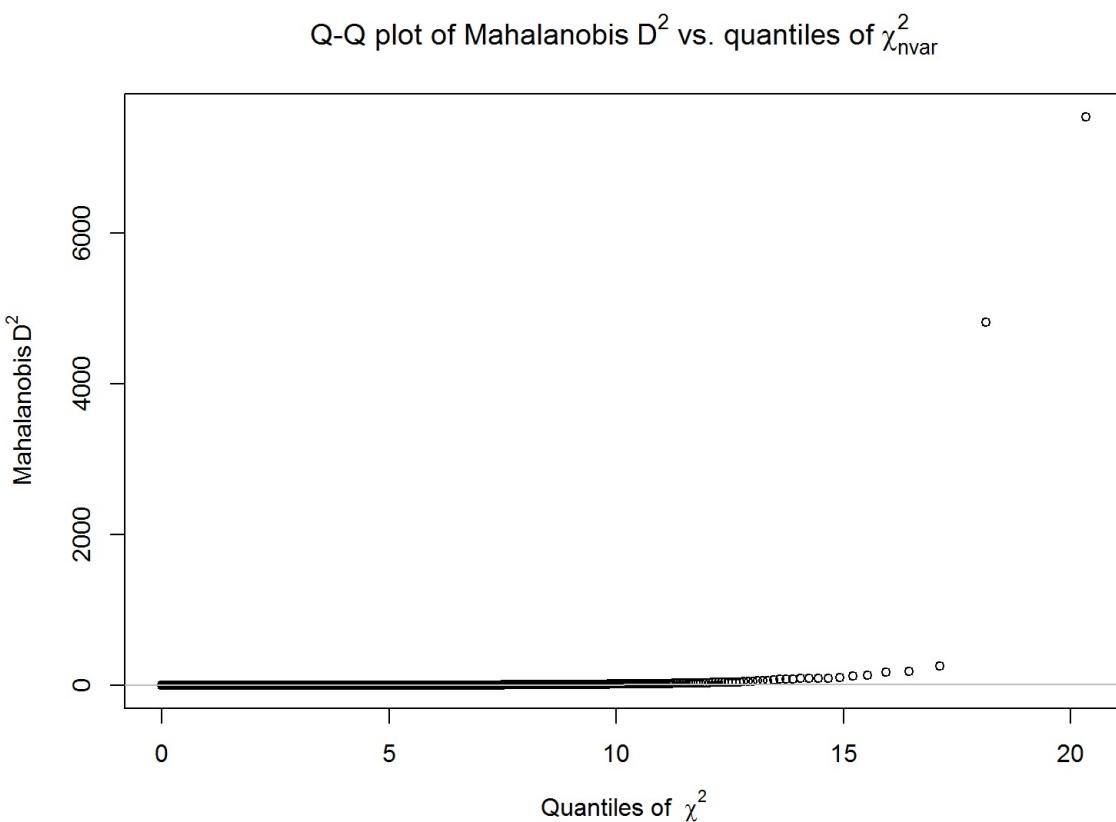
summary(lm(y~x1+x2+x1*x2)) # тот же результат x1*x2 - эффект взаимодействия

```
## 
## Call:
## lm(formula = y ~ x1 + x2 + x1 * x2)
## 
## Residuals:
##   Min     1Q Median     3Q    Max 
## -4784.8 -651.3 -123.7  381.8 21481.8 
## 
## Coefficients: (35 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 2403.000   496.838  4.837 1.34e-06 ***
## x12         1604.000   774.265  2.072 0.03832 *  
## x13         3618.500   811.332  4.460 8.27e-06 *** 
## x14         3906.667  1814.194  2.153 0.03131 *  
## x15         2795.000  1721.096  1.624 0.10441  
## x16+       4796.061  1111.871  4.314 1.62e-05 *** 
## x1ком      -1645.500  1216.999 -1.352 0.17637  
## x2бнк       345.571  774.265  0.446 0.65537  
## x2гт       -923.000  929.498 -0.993 0.32072  
## x2г6       -533.000  1647.824 -0.323 0.74635  
## x2д        -665.000   702.634 -0.946 0.34394  
## x2к        198.268   498.310  0.398 0.69073  
## x2кмк      696.266   522.597  1.332 0.18278  
## x2м        161.850   608.499  0.266 0.79026  
## x2мжбкк    497.455   686.480  0.725 0.46868  
## x2п        -249.061   498.865 -0.499 0.61761  
## x2пб       -203.000  1647.824 -0.123 0.90196  
## x2лк       182.000   860.548  0.119 0.90565  
## x2с        1488.333  1034.250  1.431 0.15237  
## x2ш       -871.846   660.856 -1.319 0.18710  
## x12:x2бнк  2897.429  1478.636  1.960 0.05007 .  
## x13:x2бнк -1313.071  1498.378 -0.876 0.38087  
## x14:x2бнк -555.238  2472.330 -0.225 0.82231  
## x15:x2бнк   NA     NA     NA     NA    
## x16+::x2бнк  NA     NA     NA     NA    
## x1ком::x2бнк  NA     NA     NA     NA    
## x12::x2гт   -300.667  1428.087 -0.211 0.83325  
## x13::x2гт  -1698.500  1934.904 -0.878 0.38086  
## x14::x2гт  -436.667  2525.252 -0.173 0.86272  
## x15::x2гт   NA     NA     NA     NA    
## x16+::x2гт   NA     NA     NA     NA    
## x1ком::x2гт  2165.500  2136.979  1.013 0.31091  
## x12::x2г6   NA     NA     NA     NA    
## x13::x2г6   -2288.500  2365.420 -0.967 0.33332  
## x14::x2г6   NA     NA     NA     NA    
## x15::x2г6   NA     NA     NA     NA    
## x16+::x2г6   NA     NA     NA     NA    
## x12::x2д   -1394.500  1074.660 -1.298 0.19444  
## x13::x2д   -3481.500  1462.650 -2.380 0.01732 *  
## x14::x2д   NA     NA     NA     NA    
## x15::x2д   NA     NA     NA     NA    
## x16+::x2д   NA     NA     NA     NA    
## x1ком::x2д   755.333  1462.650  0.516 0.60557  
## x12::x2к   -531.561  775.960 -0.685 0.49333  
## x13::x2к   -701.523  813.131 -0.863 0.38830  
## x14::x2к   539.591  1816.350  0.297 0.76641  
## x15::x2к   2545.275  1741.885  1.461 0.14398  
## x16+::x2к   NA     NA     NA     NA    
## x1ком::x2к   -4.084  1218.998 -0.003 0.99733  
## x12::x2кмк  187.076  809.846  0.231 0.81732  
## x13::x2кмк  847.030  854.537  0.991 0.32160  
## x14::x2кмк  3621.401  1895.214  1.911 0.05605 .  
## x15::x2кмк   NA     NA     NA     NA    
## x16+::x2кмк  NA     NA     NA     NA    
## x1ком::x2кмк  NA     NA     NA     NA    
## x12::x2м   29.025  908.716  0.032 0.97452  
## x13::x2м  -932.524  942.873 -0.989 0.32267  
## x14::x2м  -988.608  1907.651 -0.514 0.60723  
## x15::x2м  2440.150  2356.708  1.035 0.30050  
## x16+::x2м   NA     NA     NA     NA    
## x1ком::x2м   -169.350  1684.858 -0.101 0.91994  
## x12::x2мбкк 1733.045  989.027  1.752 0.07975 .  
## x13::x2мбкк  -946.080  1091.433 -0.867 0.38606  
## x14::x2мбкк  412.879  2082.915  0.198 0.84287  
## x15::x2мбкк   NA     NA     NA     NA    
## x16+::x2мбкк  NA     NA     NA     NA    
## x1ком::x2мбкк  NA     NA     NA     NA    
## x12::x2п   -891.588  776.532 -1.148 0.25092  
## x13::x2п  -2336.359  813.514 -2.872 0.00409 ** 
## x14::x2п  -2145.736  1816.583 -1.181 0.23755  
## x15::x2п  -70.758  1743.270 -0.041 0.96762  
## x16+::x2п   NA     NA     NA     NA    
## x1ком::x2п   408.253  1220.876  0.334 0.73809  
## x12::x2п6  -1114.000  2352.964 -0.473 0.63590  
## x13::x2п6  -2318.500  2365.420 -0.980 0.32702  
## x14::x2п6   NA     NA     NA     NA    
## x15::x2п6   NA     NA     NA     NA    
## x16+::x2п6   NA     NA     NA     NA    
## x1ком::x2п6   NA     NA     NA     NA    
## x12::x2пк  -904.273  1147.863 -0.788 0.43084  
## x13::x2пк  -2476.833  1405.269 -1.763 0.07800 .  
## x14::x2пк  -2331.667  2048.514 -1.138 0.25505  
## x15::x2пк   NA     NA     NA     NA    
## x16+::x2пк   NA     NA     NA     NA    
## x1ком::x2пк   NA     NA     NA     NA    
## x12::x2с   -2239.333  1384.199 -1.618 0.10573  
## x13::x2с  -3151.833  1405.269 -2.243 0.02492 *  
## x14::x2с   NA     NA     NA     NA    
## x15::x2с   NA     NA     NA     NA    
## x16+::x2с   NA     NA     NA     NA    
## x1ком::x2с   -1037.833  2184.579 -0.475 0.63474  
## x12::x2ш  -597.912  935.141 -0.639 0.52259  
## x13::x2ш  166.775  1012.157  0.165 0.86913  
## x14::x2ш   NA     NA     NA     NA    
## x15::x2ш   NA     NA     NA     NA    
## x16+::x2ш   NA     NA     NA     NA    
## x1ком::x2ш   1025.868  1333.527  0.769 0.44174  
## 
## --- 
## Signif. codes: 0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 1571 on 12374 degrees of freedom
## (17 observations deleted due to missingness)
## Multiple R-squared: 0.4618, Adjusted R-squared: 0.4591 
## F-statistic: 171.3 on 62 and 12374 DF, p-value: < 2.2e-16
```

Взвешенная регрессия

- Возьмем исходные данные (без удаления выбросов)
- Посчитаем для них расстояния
- Присвоим наблюдениям с меньшим расстоянием больший вес

```
x=nge$square2  
y=nge$price  
out_obs=outlier(cbind(x,y),cex=.8)
```



Оценки взвешенной линейной модели

```
summary(lm(y~x,weights = 1/out_obs))
```

```
## 
## Call:
## lm(formula = y ~ x, weights = 1/out_obs)
##
## Weighted Residuals:
##    Min      1Q  Median      3Q     Max 
## -8866.4 -1759.9 -359.7  707.6 21745.2 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 447.866   49.392   9.068 <2e-16 ***
## x            91.421    1.447  63.165 <2e-16 ***  
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2905 on 12189 degrees of freedom
##   (820 observations deleted due to missingness)
## Multiple R-squared:  0.2466, Adjusted R-squared:  0.2465 
## F-statistic: 3990 on 1 and 12189 DF,  p-value: < 2.2e-16
```

Оценки взвешенной квадратичной модели

```
summary(lm(y~x+I(x^2),weights = 1/out_obs))
```

```
## 
## Call:
## lm(formula = y ~ x + I(x^2), weights = 1/out_obs)
##
## Weighted Residuals:
##    Min      1Q  Median      3Q     Max 
## -8856.9 -1798.9 -403.8   659.9 21894.8 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 621.77793  99.28778  6.262 3.92e-10 ***
## x            81.20763   5.26117 15.435 < 2e-16 ***
## I(x^2)       0.14761   0.07311  2.019  0.0435 *  
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2905 on 12188 degrees of freedom
## (820 observations deleted due to missingness)
## Multiple R-squared:  0.2469, Adjusted R-squared:  0.2467 
## F-statistic: 1997 on 2 and 12188 DF,  p-value: < 2.2e-16
```

Сравним с оценками без взвешивания

summary(lm(y~x))

```
## 
## Call:
## lm(formula = y ~ x)
## 
## Residuals:
##   Min     1Q Median     3Q    Max 
## -21471  -875   -135    643  68012 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -524.7930   35.5549 -14.76 <2e-16 ***
## x            122.0523   0.9002 135.59 <2e-16 ***  
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 1922 on 12189 degrees of freedom
##   (820 observations deleted due to missingness)
## Multiple R-squared:  0.6013, Adjusted R-squared:  0.6013 
## F-statistic: 1.838e+04 on 1 and 12189 DF,  p-value: < 2.2e-16
```

Устойчивое оценивание

Устойчивая оценка ковариационной матрицы методом MCD

```
# install.packages("robustbase") # если не установлен  
library(robustbase)
```

```
##  
## Attaching package: 'robustbase'  
  
## The following object is masked from 'package:psych':  
##  
##   cushny
```

```
resMCD=covMcd(ngs[ , !Ind_fact])  
round(resMCD$cov,2)
```

```
##   latitude longitude square1 square2 square3 stage stages  
## latitude    0.00     0.00   0.06   0.02   0.01   0.01   0.01  
## longitude   0.00     0.00   0.05   0.02   0.01   0.01   0.02  
## square1     0.06     0.05 287.60 214.25 13.14  7.00 11.75  
## square2     0.02     0.02 214.25 176.86  4.32  1.96  2.62  
## square3     0.01     0.01 13.14   4.32  3.68  1.84  3.58  
## stage       0.01     0.01  7.00   1.96  1.84 10.16  5.12  
## stages      0.01     0.02 11.75   2.62  3.58  5.12 10.80  
## price       15.55   -0.11 12393.26 8781.33 831.42 678.26 1139.90  
##  
##   price  
## latitude    15.55  
## longitude   -0.11  
## square1    12393.26  
## square2    8781.33  
## square3    831.42  
## stage      678.26  
## stages     1139.90  
## price      895699.68
```

Устойчивое оценивание

Расчет корреляционной матрицы

```
n=dim(nqs[,'Ind_fact])[2]
cor_matr=matrix(nrow=n,ncol=n)
for_(i_in_1:n){
  for_(j_in_1:n){
    cor_matr[i,]=resMCD$cov[i,j]/sqrt(resMCD$cov[i,i]*resMCD$cov[j,j])
  }
}
round(cor_matr,2)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,] 1.00 0.19 0.05 0.03 0.10 0.04 0.07 0.24
## [2,] 0.19 1.00 0.04 0.02 0.10 0.04 0.08 0.00
## [3,] 0.05 0.04 1.00 0.95 0.40 0.13 0.21 0.77
## [4,] 0.03 0.02 0.95 1.00 0.17 0.05 0.06 0.70
## [5,] 0.10 0.10 0.40 0.17 1.00 0.30 0.57 0.46
## [6,] 0.04 0.04 0.13 0.05 0.30 1.00 0.49 0.22
## [7,] 0.07 0.08 0.21 0.06 0.57 0.49 1.00 0.37
## [8,] 0.24 0.00 0.77 0.70 0.46 0.22 0.37 1.00
```

Устойчивая регрессия методом LTS - функция lqs {MASS}

```
x=nls$square2  
y=ndss$price  
library(MASS)  
ltsreg(y~x)
```

```
## Call:  
## lqs(formula = y ~ x, method = "lts")  
##  
## Coefficients:  
## (Intercept)      x  
##   1407.35      45.38  
##  
## Scale estimates 763 734
```

```
lmsreg(y~x)
```

```
## Call:  
## lqs(formula = y ~ x, method = "lms")  
##  
## Coefficients:  
## (Intercept)      x  
##   1325          50  
##  
## Scale estimates 785.7 744.1
```

Устойчивая регрессия с помощью функции lmRob {robust}

```
#install.packages("robust")
library(robust)
```

```
## Loading required package: fit.models
## Loading required package: lattice
## Loading required package: rrcov
## Scalable Robust Estimators with High Breakdown Point (version 1.4-3)
```

```
summary(lmRob(y~x))
```

```
##
## Call:
## lmRob(formula = y ~ x)
##
## Residuals:
##    Min      1Q  Median      3Q     Max
## -11636.70   -487.01    71.74   721.74  79244.65
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 731.7057   20.7733  35.22 <2e-16 ***
## x           72.0946    0.6189 116.50 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 839.4 on 12189 degrees of freedom
## Multiple R-Squared: 0.336
##
## Test for Bias:
##              statistic p-value
## M-estimate    514.1     0
## LS-estimate   1583.1     0
## 820 observations deleted due to missingness
```

Устойчивая регрессия с помощью функции lmRob {robust}

Применима для качественных входных факторов

```
x=nqss$rooms  
y=nqss$price  
summary(lmRob(y~x))
```

```
##  
## Call:  
## lmRob(formula = y ~ x)  
##  
## Residuals:  
##      Min      1Q  Median      3Q     Max  
## -17966.67   -364.73    31.13   831.13  849070.92  
##  
## Coefficients:  
##             Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 2350.36    14.49  162.24 <2e-16 ***  
## x2          664.37    19.69  33.75 <2e-16 ***  
## x3         1218.51    20.89  58.34 <2e-16 ***  
## x4        1612.16    35.43  45.51 <2e-16 ***  
## x5        2196.81   158.39  13.87 <2e-16 ***  
## x6+       16316.31   552.94  29.51 <2e-16 ***  
## xxom      -1421.28    27.28 -52.09 <2e-16 ***  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Residual standard error: 797.6 on 13004 degrees of freedom  
## Multiple R-Squared:  0.7538  
##  
## Test for Bias:  
##                 statistic p-value  
## M-estimate     -639.1      1  
## LS-estimate     977.9      0
```

Устойчивая регрессия с помощью функции lmRob {robust}

Сравните с исходным вариантом (без удаления выбросов)

summary(lm(y~x))

```
## 
## Call:
## lm(formula = y ~ x)
## 
## Residuals:
##   Min     1Q Median     3Q    Max 
## -27302 -1046    -500   213 848337 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 2456.5    175.1  14.031 < 2e-16 ***
## x2          943.2    233.9   4.032 5.55e-05 ***
## x3         2339.7    239.8   9.757 < 2e-16 ***
## x4         5053.8    369.1  13.693 < 2e-16 ***
## x5         9329.5    887.1  10.517 < 2e-16 ***
## x6+        25545.7   2104.9  12.136 < 2e-16 ***
## xxom       -793.8    337.9  -2.349  0.0188 *  
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 9839 on 13004 degrees of freedom
## Multiple R-squared:  0.03738,   Adjusted R-squared:  0.03694 
## F-statistic: 84.17 on 6 and 13004 DF,  p-value: < 2.2e-16
```

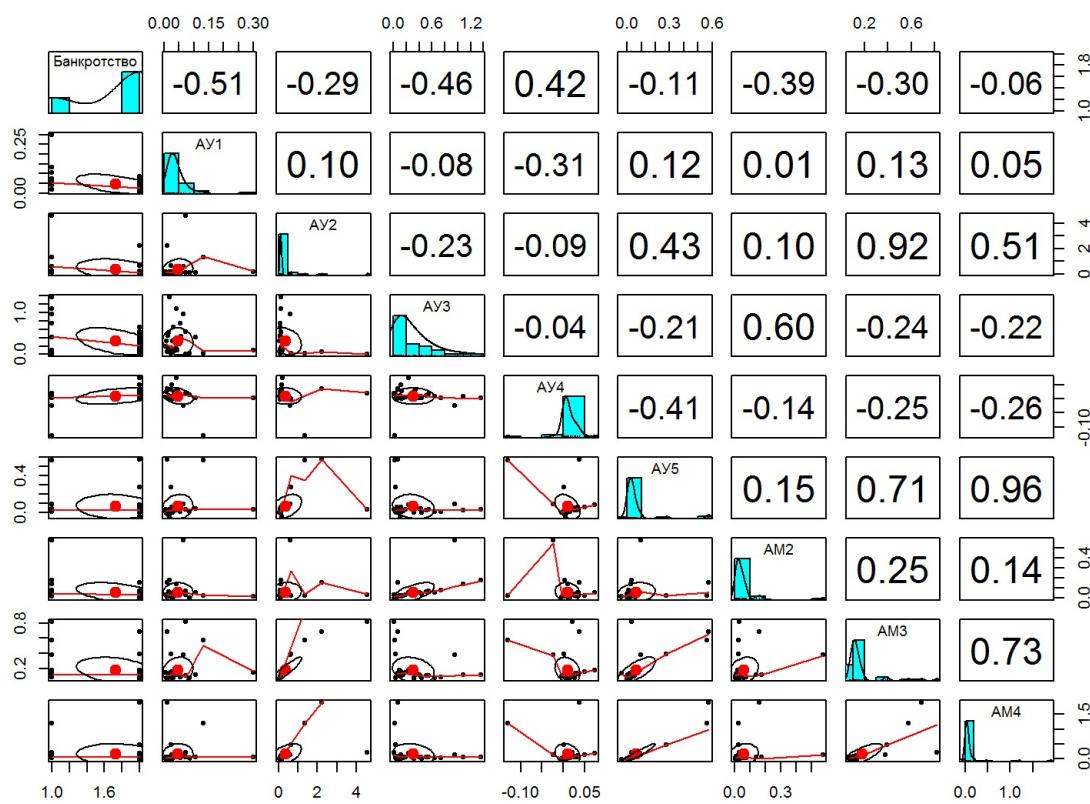
Два варианта в одной таблице

```
lmRobest=lmRob(y~x)$coef  
lm_est=lm(y~x)$coef  
round(cbind(lmRobest,lm_est),0)
```

```
##          lmRobest lm_est  
## (Intercept)    2350    2457  
## x2            664     943  
## x3           1219    2340  
## x4           1612    5054  
## x5           2197    9330  
## x6+          16316   25546  
## xkom         -1421    -794
```

Модели бинарного выбора

```
Data_for_logit=read.csv2("Data_for_logit.csv")
library(psych)
pairs.panels(Data_for_logit[,-1])
```



Построим модель зависимости банкротства от показателей АУ1, АУ3, АУ4

```
glm_for_all <- glm(Банкротство ~ АУ1+АУ2+АУ3+АУ4+АУ5+AM2, data=Dat)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(glm_for_all)
```

```
## 
## Call:
## glm(formula = Банкротство ~ АУ1 + АУ2 + АУ3 + АУ4 + АУ5 + AM2,
##      family = "binomial", data = Data_for_logit)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max 
## -2.58403  0.00000  0.03396  0.27887  0.68145 
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept) 7.4035   3.7043  1.999  0.0456 *  
## АУ1        -105.5169  72.0290 -1.465  0.1429    
## АУ2       -15.0822  13.6457 -1.105  0.2690    
## АУ3        -0.4871  11.1982 -0.043  0.9653    
## АУ4       285.7236 217.9526  1.311  0.1899    
## АУ5        47.3725  46.1220  1.027  0.3044    
## AM2       -49.6453  98.1206 -0.506  0.6129    
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 43.1810  on 36  degrees of freedom
## Residual deviance: 9.8419  on 30  degrees of freedom
## AIC: 23.842
##
## Number of Fisher Scoring iterations: 14
```

Подбор модели с помощью критерия AIC

```
step.aic <- step(glm_for_all)
```

```
## Start: AIC=23.84
## Банкротство ~ AY1 + AY2 + AY3 + AY4 + AY5 + AM2
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##          Df Deviance   AIC
## - AY5     1   11.142  23.142
## <none>    9.842  23.842
## - AY2     1   13.519  25.519
## - AY4     1   14.475  26.475
## - AY1     1   19.701  31.701
## - AM2     1   72.087  84.087
## - AY3     1  144.175 156.175
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
## Step: AIC=23.14
## Банкротство ~ AY1 + AY2 + AY3 + AY4 + AM2
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##          Df Deviance   AIC
## - AM2     1   11.291  21.291
## - AY3     1   11.310  21.310
## <none>    11.142  23.142
## - AY4     1   14.792  24.792
## - AY2     1   15.572  25.572
## - AY1     1   20.326  30.326
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
## Step: AIC=21.29
## Банкротство ~ AY1 + AY2 + AY3 + AY4
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##          Df Deviance   AIC
## <none>    11.291  21.291
## - AY4     1   14.793  22.793
## - AY2     1   17.637  25.637
## - AY1     1   20.505  28.505
## - AY3     1   21.571  29.571
```

Результат оценивания

```
glm_for_4 <- glm(Банкротство ~ АY1 + АY2 + АY3 + АY4, data=Data_fo
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(glm_for_4)
```

```
## 
## Call:
## glm(formula = Банкротство ~ АY1 + АY2 + АY3 + АY4, family = "binomial",
##      data = Data_for_logit)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.69177  0.00000  0.09623  0.24932  0.78909
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.932     2.903   2.043  0.0410 *
## АY1        -75.359    52.557  -1.434  0.1516
## АY2         -3.696     5.006  -0.738  0.4604
## АY3         -5.681    2.558  -2.221  0.0264 *
## АY4        233.575   171.865   1.359  0.1741
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 43.181  on 36  degrees of freedom
## Residual deviance: 11.291  on 32  degrees of freedom
## AIC: 21.291
##
## Number of Fisher Scoring iterations: 9
```

Устойчивое оценивания модели `glm` с помощью пакета `robust`

```
glm_robust <- glmRob(Банкротство ~ АҮ1 + АҮ2 + АҮ3 + АҮ4, data=Dat
```

```
##  
## Call: glmRob(formula = Банкротство ~ АҮ1 + АҮ2 + АҮ3 + АҮ4, family = "binomial",  
##                 data = Data_for_logit)  
## Deviance Residuals:  
##      Min       10      Median       30       Max  
## -2.606e+00 -2.107e-08  1.004e-01  2.737e-01  1.004e+00  
##  
## Coefficients:  
##             Estimate Std. Error z value Pr(>|z|)  
## (Intercept)  6.146   0.45235 13.59 4.790e-42  
## AҮ1        -78.107   0.06405 -1219.49 0.000e+00  
## AҮ2        -6.021   0.19869  -30.38 1.036e-201  
## AҮ3        -5.568   0.26044  -21.38 2.172e-101  
## AҮ4        249.535   0.03048 8187.90 0.000e+00  
##  
## (Dispersion Parameter for binomial family taken to be 1 )  
##  
## Null Deviance: 51.29 on 36 degrees of freedom  
##  
## Residual Deviance: 12.18213 on 32 degrees of freedom  
##  
## Number of Iterations: 37  
##  
## Correlation of Coefficients:  
##          (Intercept) AҮ1     AҮ2     AҮ3  
## AҮ1  0.31639  
## AҮ2  0.35668  -0.85889  
## AҮ3  0.84047  0.11536 -0.33913  
## AҮ4  0.11039  0.35451  0.07671  0.02981
```

Программные системы статистического анализа

Тимофеева Анастасия Юрьевна

17 ноября 2021

Структура лекции

- Графики для исследования связей между объектами
- Методы снижения размерности: метод главных компонент, метод многомерного шкалирования
- Методы классификации без учителя: кластерный анализ, иерархический кластерный анализ, нечеткая классификация
- Методы классификации с учителем: деревья классификации, нейронные сети, алгоритм “Случайный лес”
- Разведочный и иерархический факторный анализ
- Анализ ассоциативных правил

Визуализация матрицы расстояний между объектами

Используем функцию `heatmap{rgl}`

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

```
library(rgl)
```

Визуализация матрицы расстояний между объектами

На примере данных о банках

Считаем данные и построим матрицу расстояний

```
Data_for_logit=read.csv2("Data_for_logit.csv")
distMatrix=as.matrix(dist(Data_for_logit[,-(1:2)]))
```

Визуализация матрицы расстояний между объектами

По рисунку можно понять, какие банки сильно отличаются от других по характеристикам, а какие ближе всего друг к другу

```
heatmap(distMatrix, symm = T, col = terrain.colors(20))
```

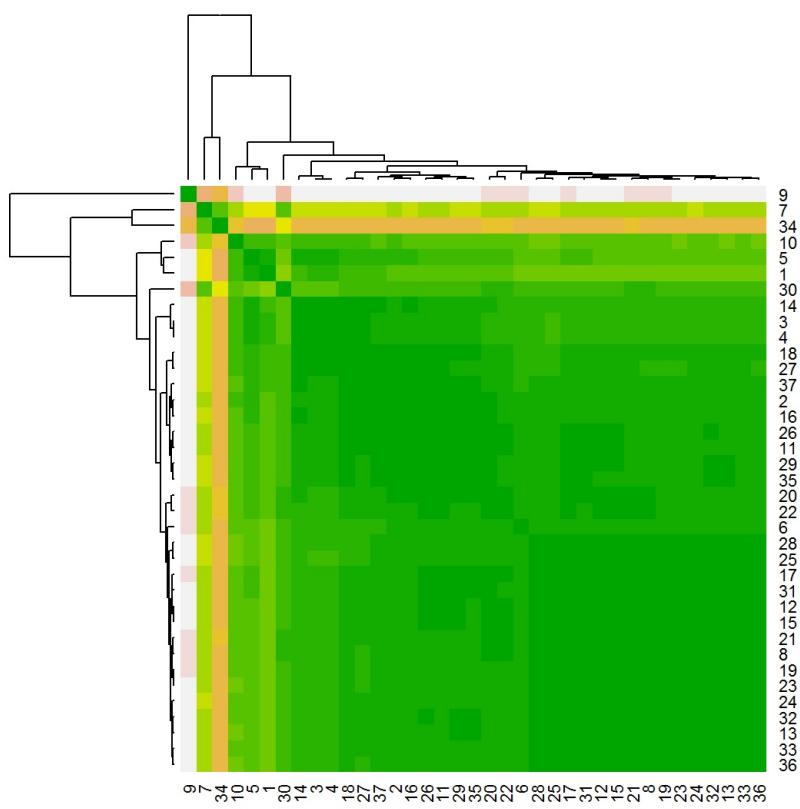


График параллельных координат

Используем функцию `parcoord{MASS}`

Подключим библиотеку

```
library(MASS)
```

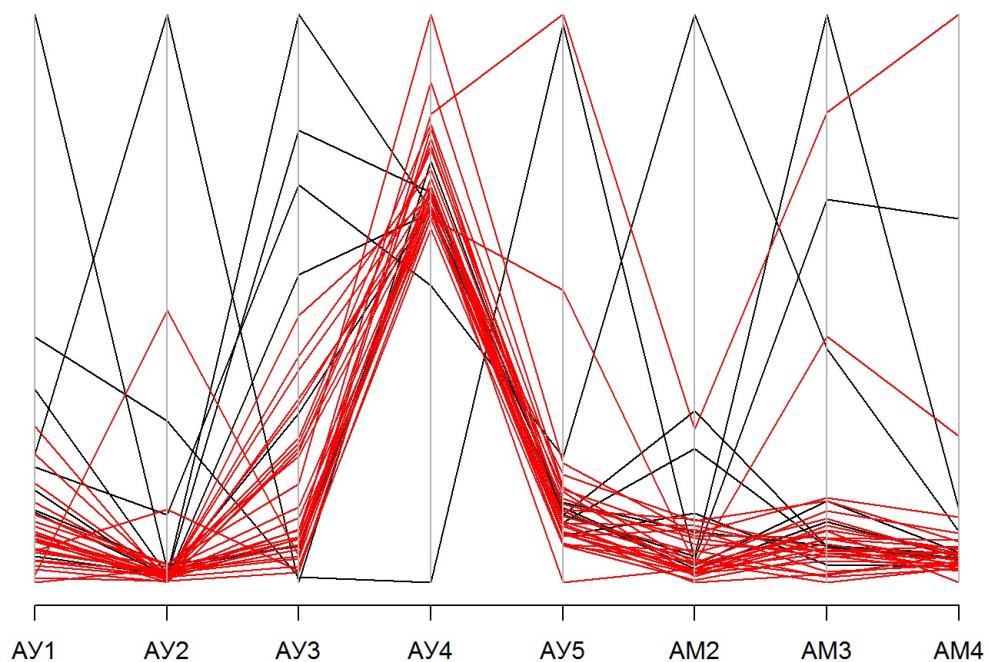
Создадим числовой вектор: 1 - банкрот, 2 - небанкрот

```
for_col=as.numeric(Data_for_logit$Банкротство)
```

Это будет соответствовать цветам: 1 - черный, 2 - красный

Наглядное отображение отличий между банками-банкротами и небанкротами

```
parcoord(Data_for_logit[,-(1:2)], col=for_col)
```



Построения статистических графиков для каждого класса

Используем функцию `qplot{ggplot2}`

С помощью параметра `facets` выделяем классы

Подключим библиотеку

```
library(ggplot2)
```

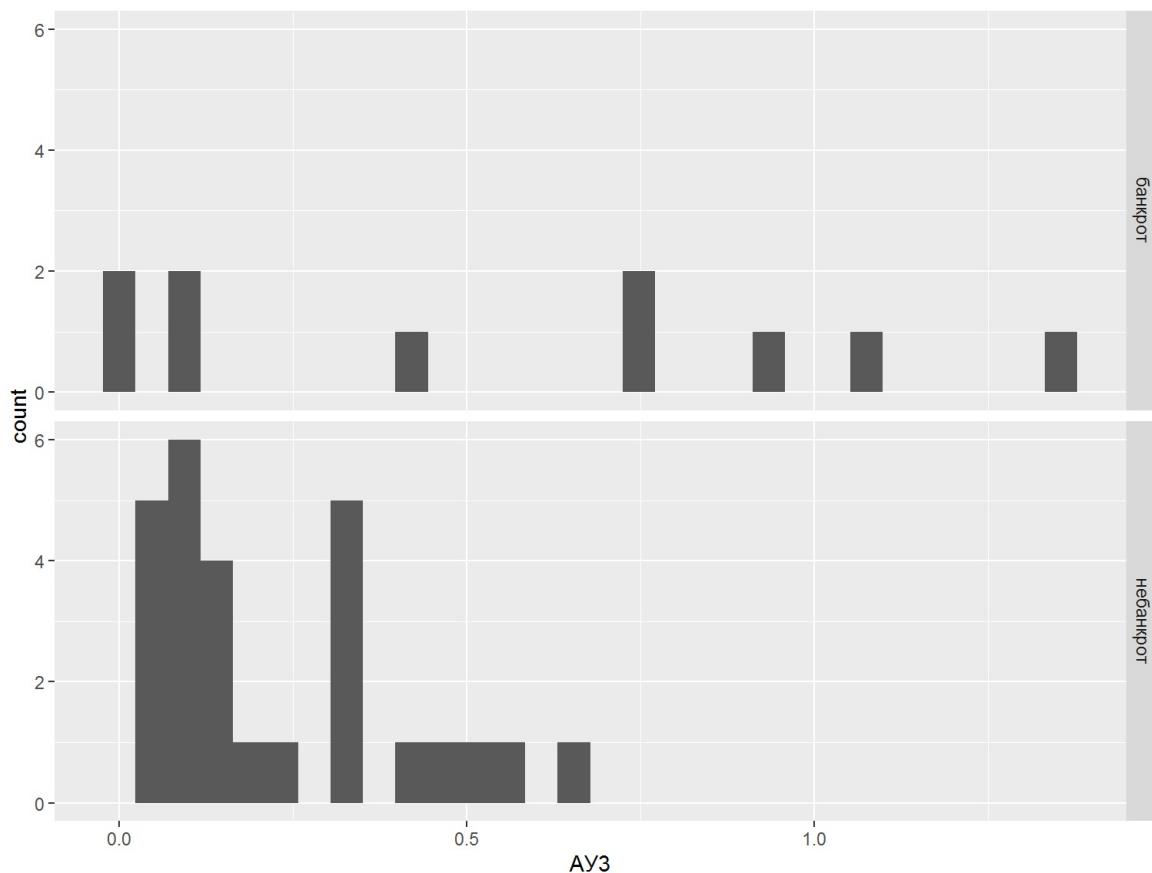
```
qplot(AY3,data=Data_for_logit, facets=Банкротство ~.)
```

Сравним банки-банкроты и небанкроты по показателю АУЗ

По результатам оценивания модели бинарного выбора АУЗ увеличивал вероятность банкротства.

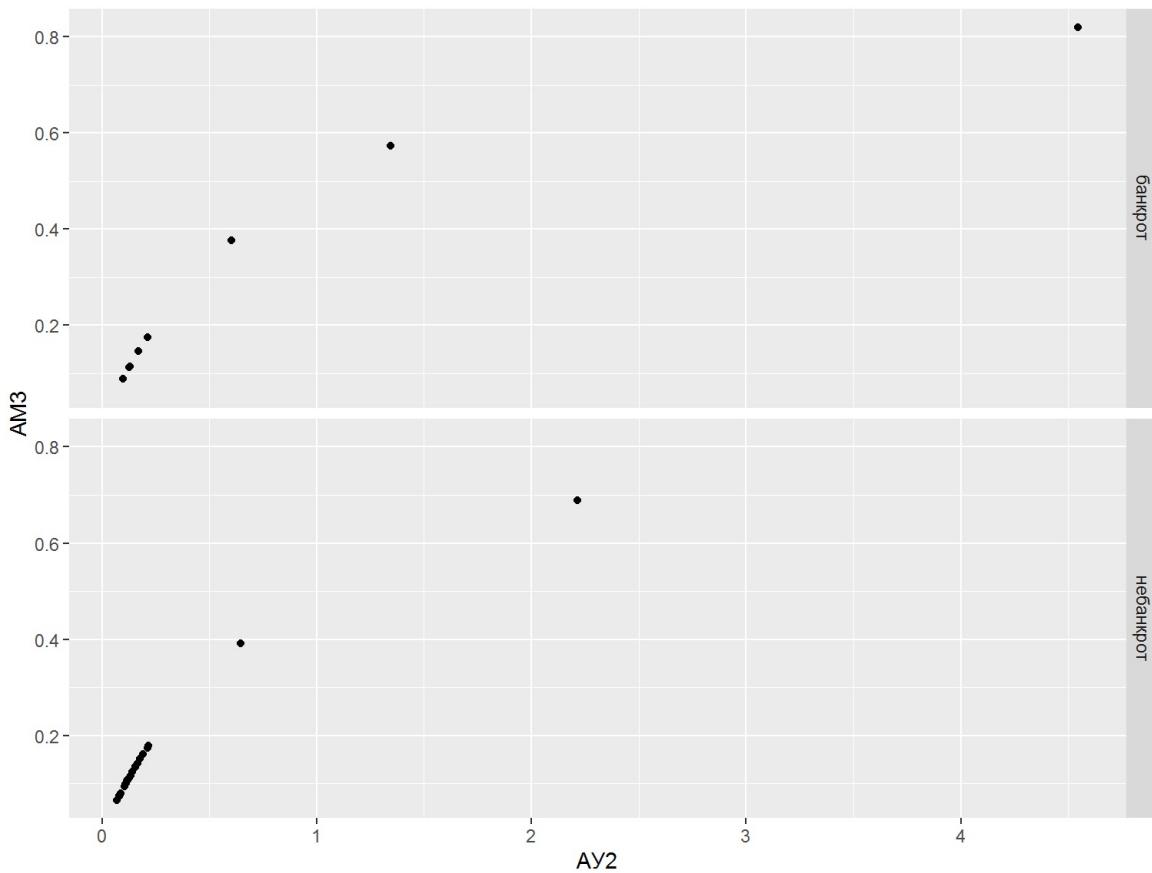
```
qplot(AUZ, data=Data_for_logit, facets=Банкротство ~.)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Сравним банки-банкроты и небанкроты по показателю АУ2 и АМЗ

```
qplot(AU2, AM3, data=Data_for_logit, facets=Банкротство ~.)
```



Снижение размерности - метод главных компонент

Перед использованием метода нужно стандартизировать переменные.

Снизим размерность данных о банках.

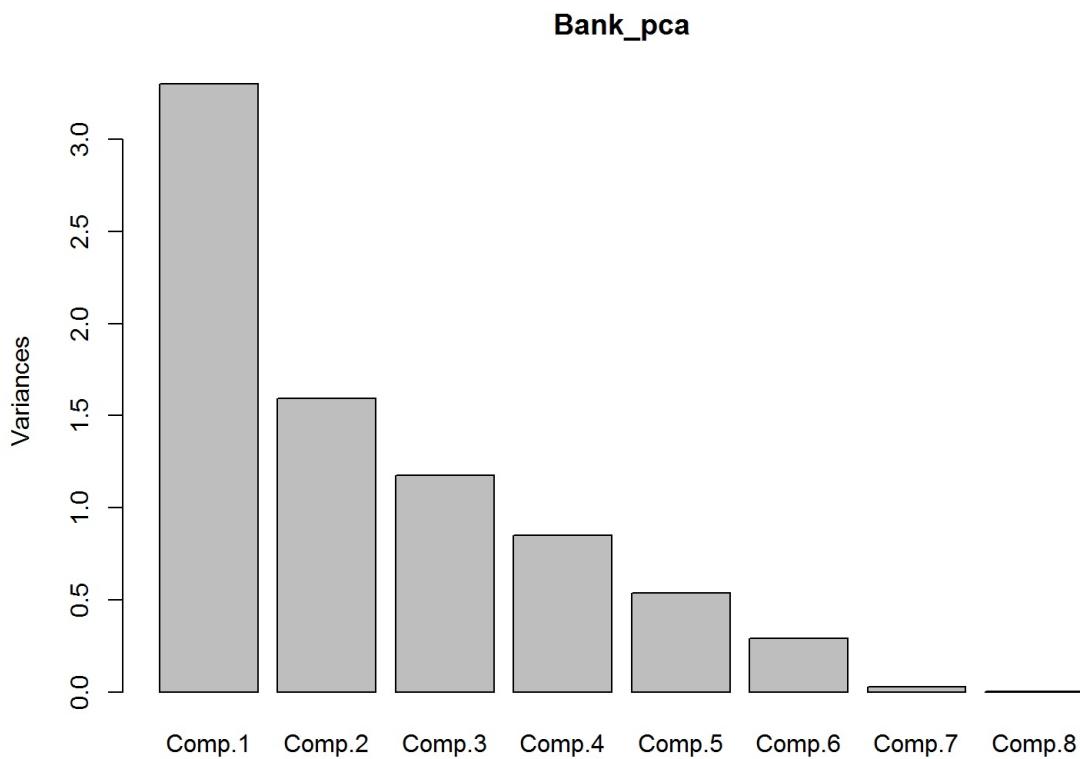
```
stand_data=scale(Data_for_logit[,-(1:2)])
```

Применим метод главных компонент к стандартизованным данным

```
Bank_pca=princomp(stand_data)
```

График долей объясненной дисперсии

```
plot(Bank_pca)
```



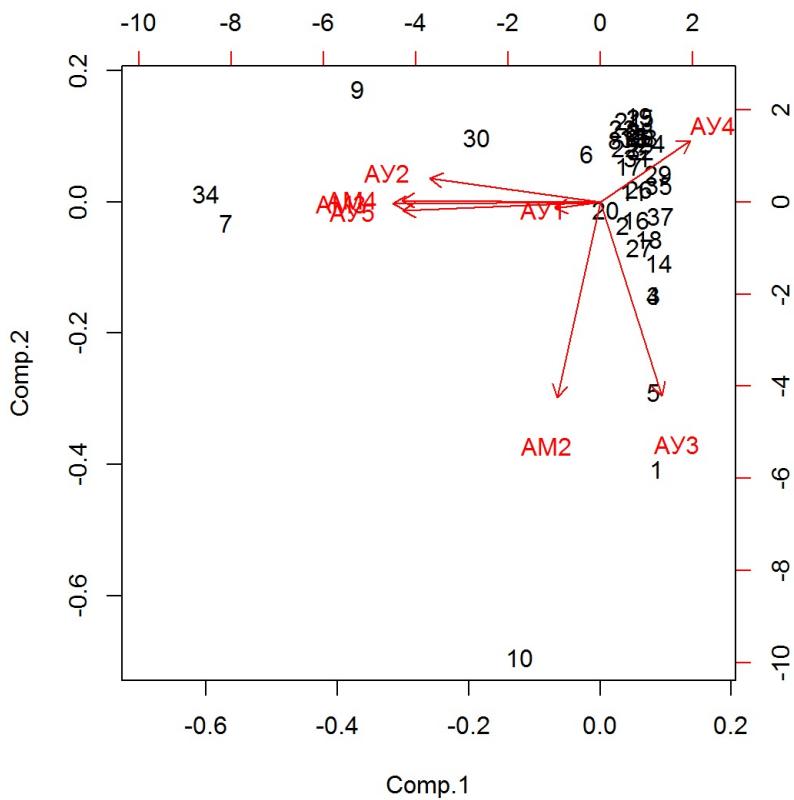
Нагрузки по факторам

Loadings(Bank_pca)

```
##  
## Loadings:  
##   Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8  
## AY1 -0.111    0.706 -0.516 -0.466  
## AY2 -0.418   -0.252 -0.527  0.326  0.222  0.279  0.493  
## AY3  0.152 -0.684                           0.698  
## AY4  0.222  0.216 -0.605 -0.284 -0.663   -0.124  
## AY5 -0.485                           0.408 -0.273   -0.519  0.494  
## AM2 -0.105 -0.691 -0.160 -0.126 -0.128 -0.653  0.109  0.127  
## AM3 -0.508                           -0.170 -0.274  0.159   -0.454 -0.636  
## AM4 -0.486                           0.342 -0.347  0.146  0.642 -0.291  
##  
##           Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8  
## SS loadings  1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000  
## Proportion Var 0.125  0.125  0.125  0.125  0.125  0.125  0.125  0.125  
## Cumulative Var 0.125  0.250  0.375  0.500  0.625  0.750  0.875  1.000
```

График в пространстве первых двух компонент

biplot(Bank_pca)



Оцененные значения компонент

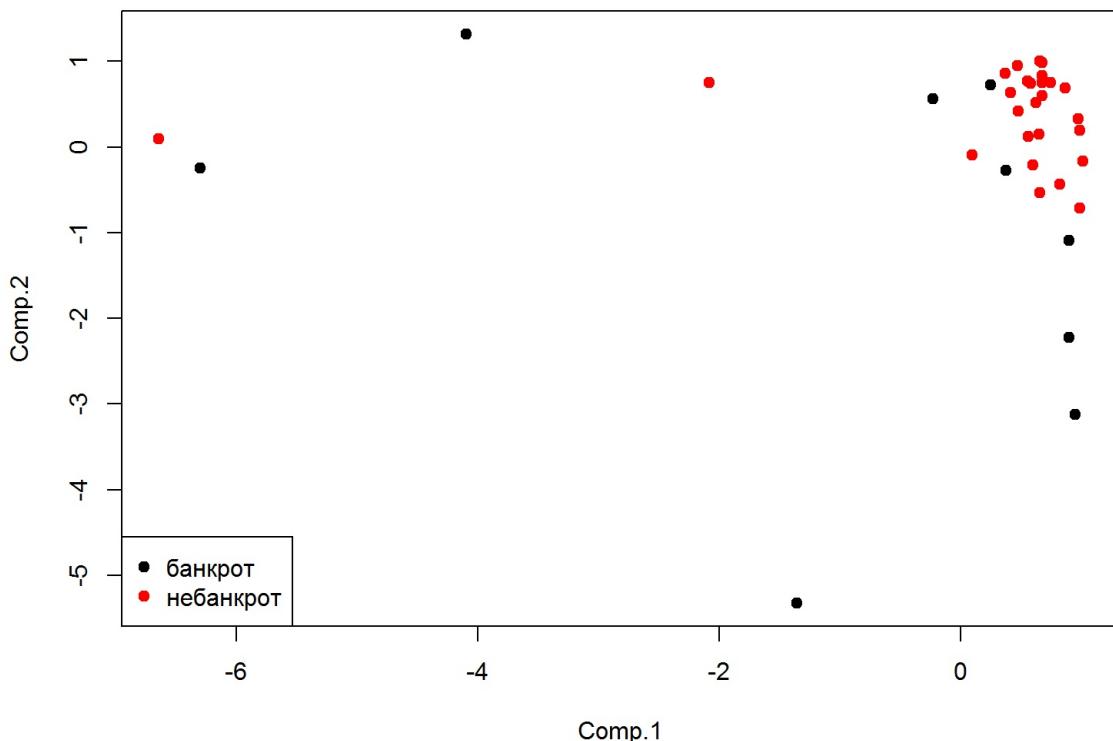
```
comp_pr=predict(Bank_pca)
head(comp_pr)
```

```
##           Comp.1      Comp.2      Comp.3      Comp.4      Comp.5      Comp.6
## [1,] 0.9508681 -3.1245301 -0.5726378 0.17133122 0.19475229 1.30603177
## [2,] 0.3790891 -0.2774234 1.0805738 -0.43435501 -0.42715512 0.19873173
## [3,] 0.9010845 -1.0899108 0.2921199 -0.03219398 0.11025201 0.67308024
## [4,] 0.9010845 -1.0899108 0.2921199 -0.03219398 0.11025201 0.67308024
## [5,] 0.9026801 -2.2287811 -0.2523176 -0.08087846 -0.07148988 0.99374644
## [6,] -0.2268072 0.5614558 3.9049743 -2.45097437 -2.05305350 -0.00239899
##           Comp.7      Comp.8
## [1,] -0.08893263 -0.04978963
## [2,] 0.02033542 0.04195509
## [3,] 0.06853503 -0.03069599
## [4,] 0.06853503 -0.03069599
## [5,] -0.04789615 -0.05610615
## [6,] 0.07703120 -0.08299792
```

Сравним банкроты и небанкроты

в пространстве двух главных компонент

```
plot(comp_pr[,1:2], pch=19, col=for_col)
legend("bottomleft", c("банкрот", "небанкрот"), pch=19, col=1:2)
```



Снижение размерности - метод многомерного шкалирования

Метод использует матрицу расстояний между объектами

Рассчитаем манхэттенское расстояние между объектами

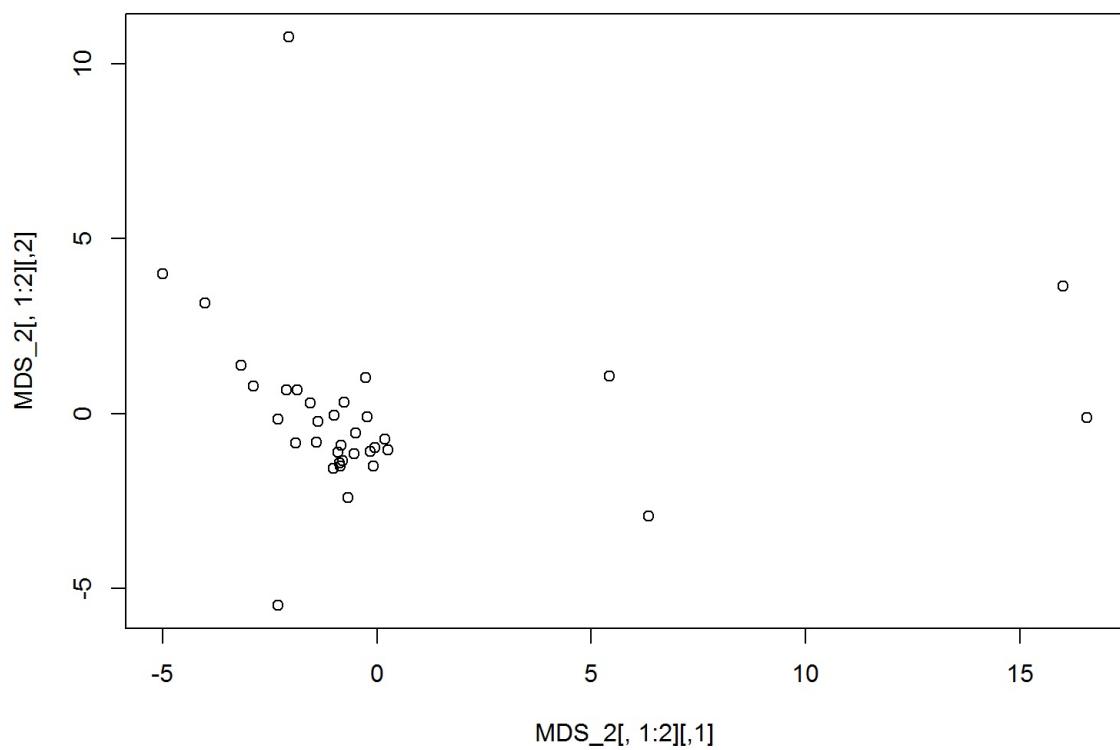
```
mdist=dist(stand_data,method="manhattan")
```

С помощью функции cmdscale вычисляются координаты точек в пространстве заданной размерности.

```
MDS_2<-cmdscale(mdist,2)
```

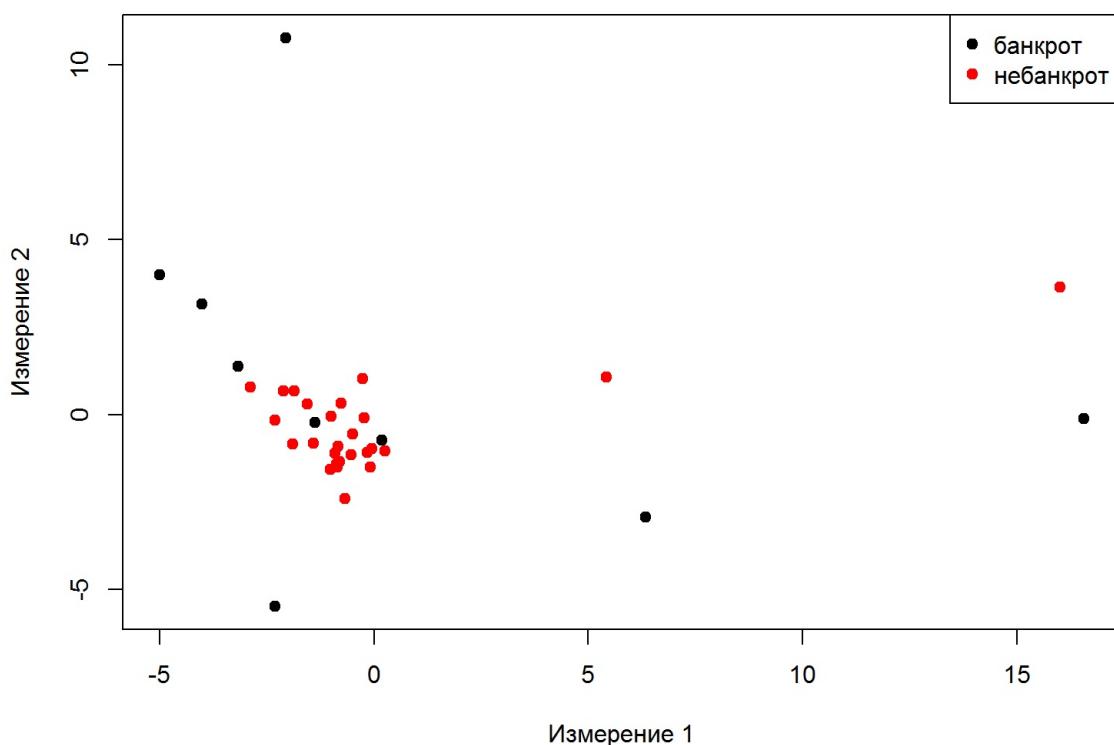
Изображение результатов в двумерном пространстве

```
plot(MDS_2[,1:2])
```



Сравним банкроты и небанкроты

```
plot(MDS_2[, 1:2], pch=19, col=for.col,  
xlab="Измерение 1", ylab="Измерение 2")  
legend("topright", c("банкрот", "небанкрот"), pch=19, col=1:2)
```



Кластерный анализ

Метод k-средних

```
Bank_kmeans=kmeans(stand_data,2)
```

```
## [1] 3 34
```

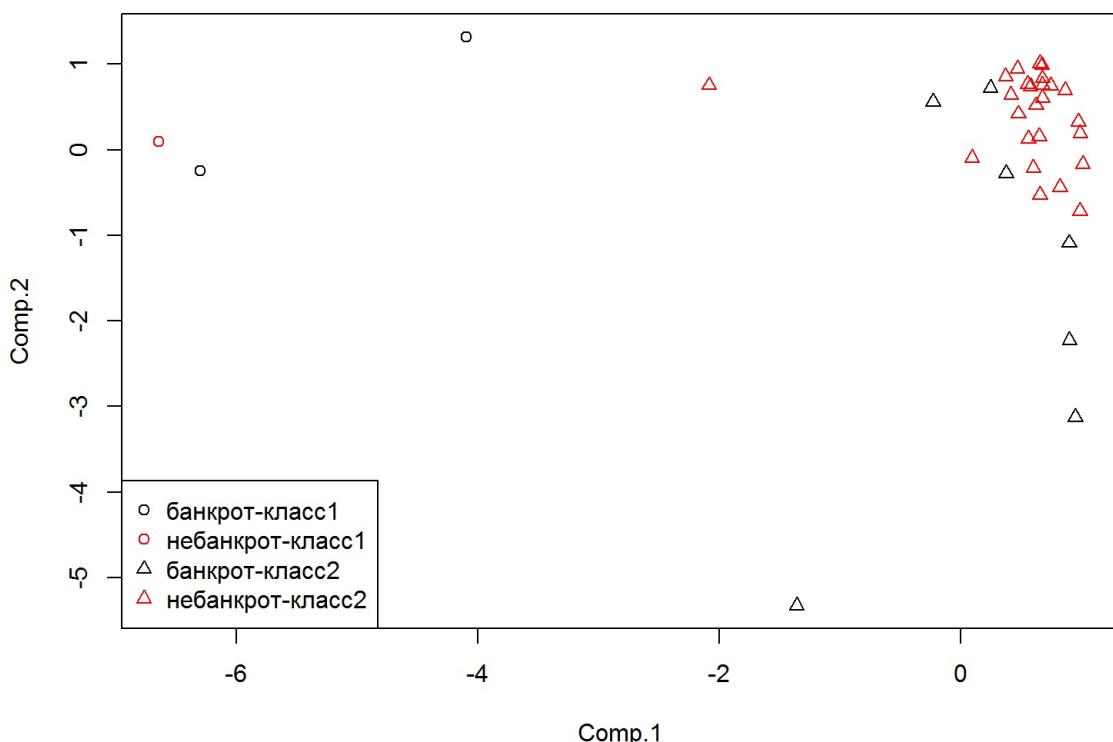
Удалось ли классифицировать банки с точки зрения вероятности банкротства?

```
table(Bank_kmeans$cluster, Data_for_logit$Банкротство)
```

```
##  
##      банкрот небанкрот  
## 1      2      1  
## 2     8     26
```

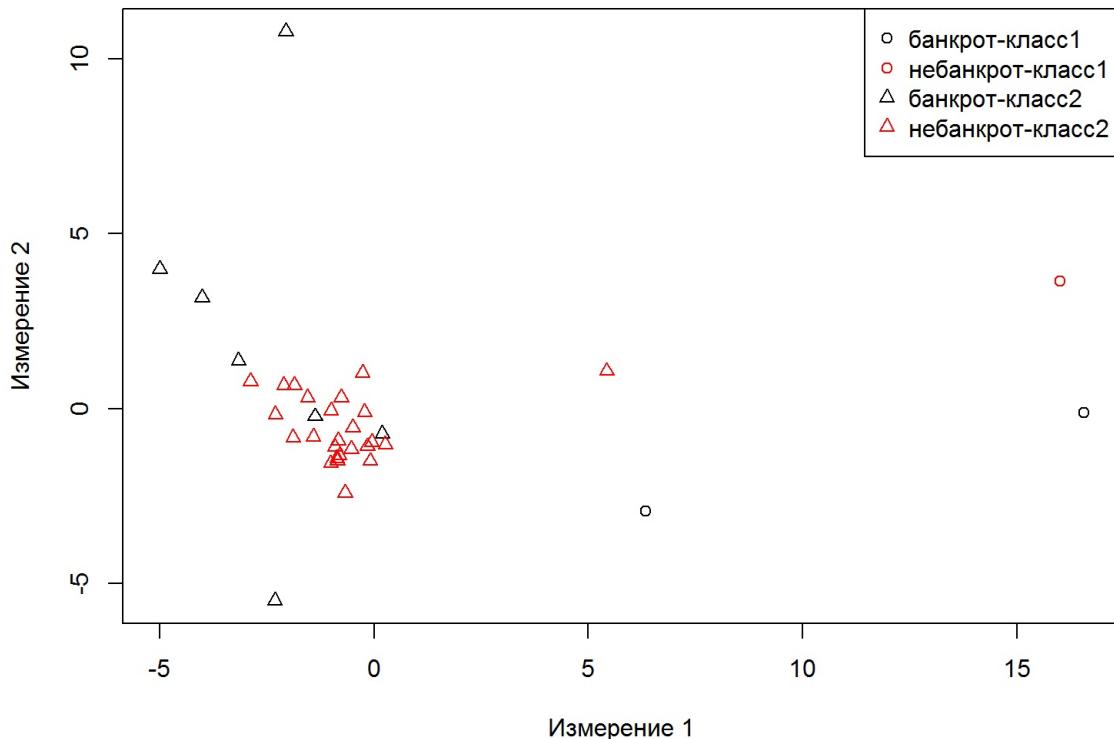
Отобразим результат в пространстве меньшей размерности (PCA)

```
plot(comp.pca[,1:2], pch=Bank_kmeans$cluster, col=for.col)
legend("bottomleft", c("банкрот-класс1", "небанкрот-класс1",
                      "банкрот-класс2", "небанкрот-класс2"),
       pch=rep(1:2, each=2), col=1:2)
```



Отобразим результат в пространстве меньшей размерности (MDS)

```
plot(MDS_2[, 1:2], pch=Bank_kmeans$cluster, col=for_col,
     xlab="Измерение 1", ylab="Измерение 2")
legend("topright", c("банкрот-класс1", "небанкрот-класс1",
                     "банкрот-класс2", "небанкрот-класс2"),
       pch=rep(1:2, each=2), col=1:2)
```

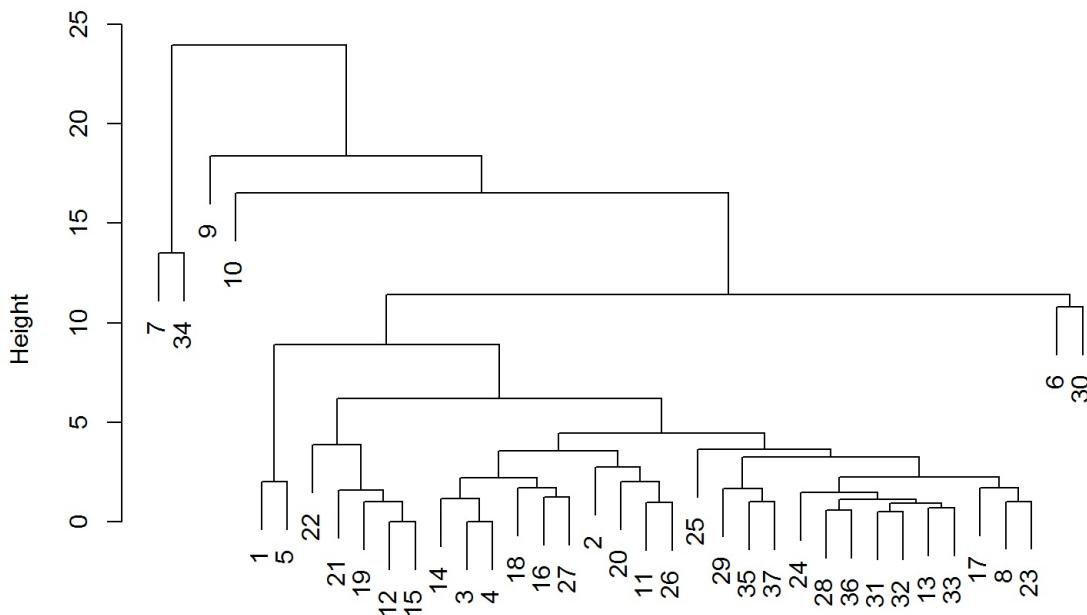


Иерархический кластерный анализ

Построение дендрограммы

```
Bank hc=hclust(mdist)  
plot(Bank_hc)
```

Cluster Dendrogram



mdist
hclust (*, "complete")

С использованием функции `pvclust{pvclust}`

```
install.packages("pvclust")
```

```
library(pvclust)
```

Классификация показателей деятельности банков

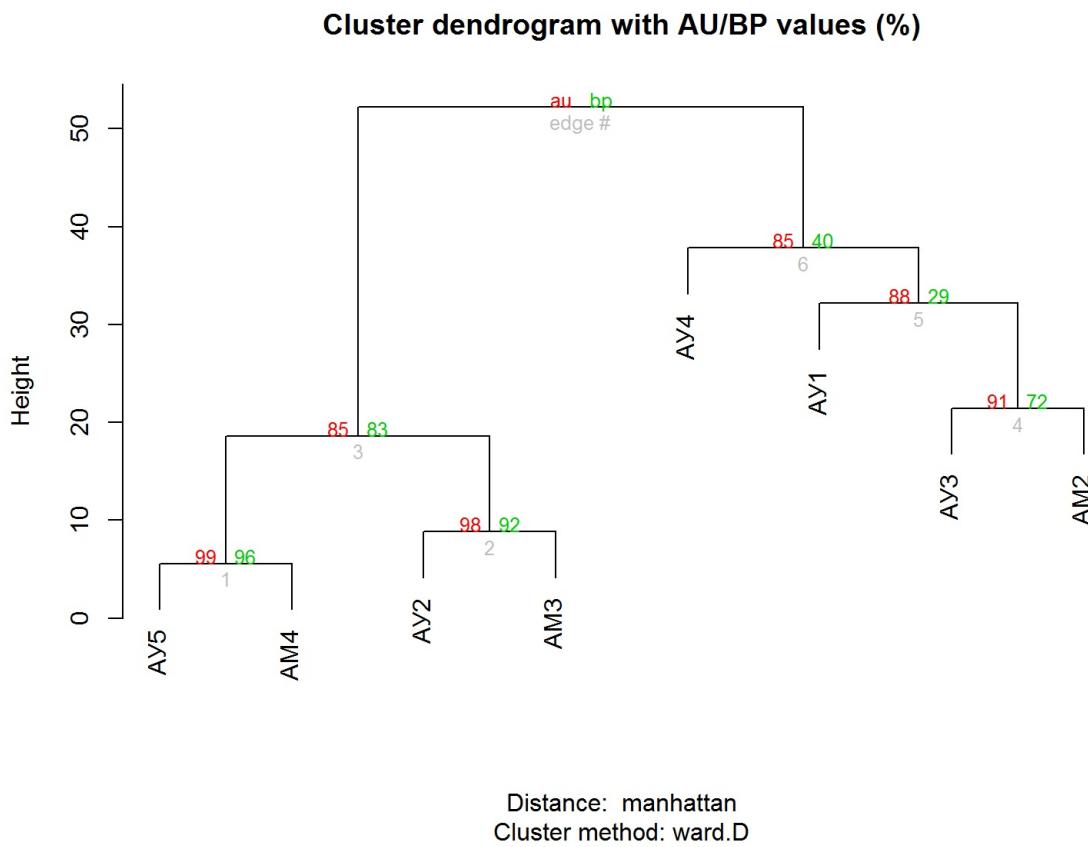
```
Bank_ind_pv=pvclust(stand_data, method.dist="manhattan",
method.hclust="ward", nboot=100)
```

```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

```
## Bootstrap (r = 0.49)... Done.
## Bootstrap (r = 0.59)... Done.
## Bootstrap (r = 0.68)... Done.
## Bootstrap (r = 0.78)... Done.
## Bootstrap (r = 0.89)... Done.
## Bootstrap (r = 1.0)... Done.
## Bootstrap (r = 1.08)... Done.
## Bootstrap (r = 1.19)... Done.
## Bootstrap (r = 1.3)... Done.
## Bootstrap (r = 1.38)... Done.
```

Классификация показателей деятельности банков

```
plot(Bank_ind_pv)
```



Классификация банков

```
Bank_pv=pvclust(t(stand_data), method.dist="manhattan",
method.hclust="ward", nboot=100)
```

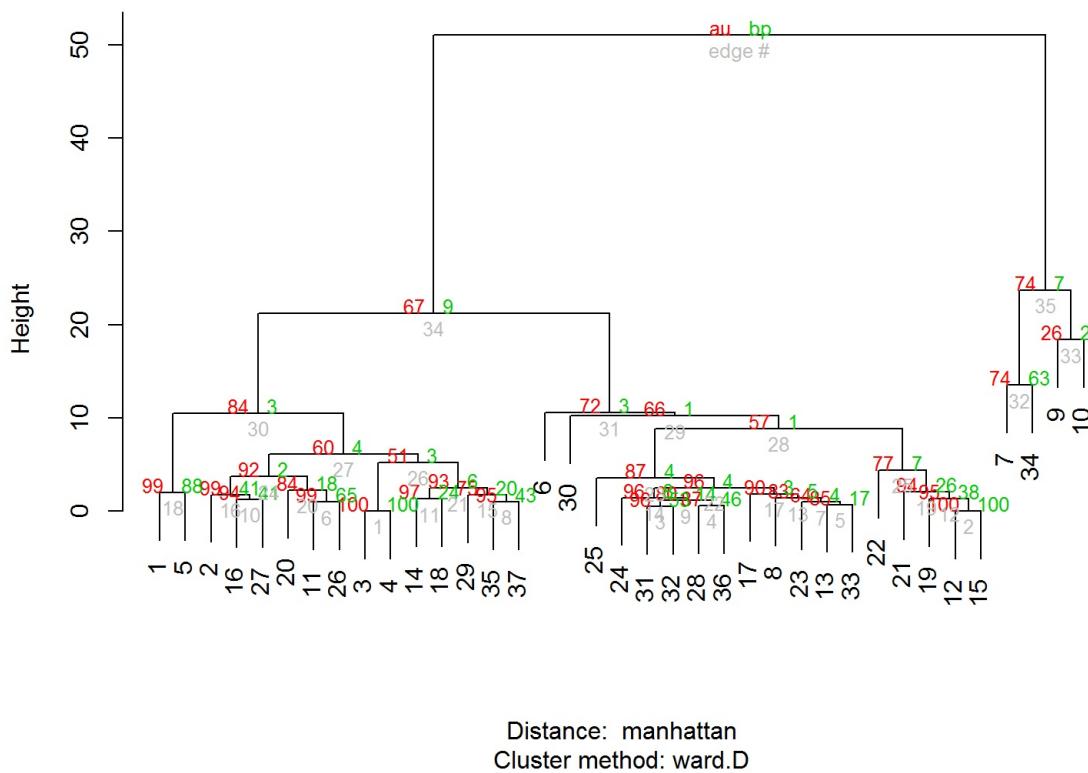
```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

```
## Bootstrap (r = 0.5)... Done.
## Bootstrap (r = 0.5)... Done.
## Bootstrap (r = 0.62)... Done.
## Bootstrap (r = 0.75)... Done.
## Bootstrap (r = 0.88)... Done.
## Bootstrap (r = 1.0)... Done.
## Bootstrap (r = 1.0)... Done.
## Bootstrap (r = 1.12)... Done.
## Bootstrap (r = 1.25)... Done.
## Bootstrap (r = 1.38)... Done.
```

Классификация банков

plot(Bank_pv)

Cluster dendrogram with AU/BP values (%)



Нечеткие методы кластеризации

Воспользуемся функцией fanny{cluster}

```
library(cluster)
```

```
Bank_fuzzy=fanny(Data_for_logit[,-(1:2)], 2)
```

Принадлежности к I и II классу и номер кластера

```
cbind(Bank_fuzzy$membership,Bank_fuzzy$clustering)[1:10,]
```

```
##      [,1]     [,2]     [,3]
## [1,] 0.6329984 0.3670016 1
## [2,] 0.5654406 0.4345594 1
## [3,] 0.7235737 0.2764263 1
## [4,] 0.7235737 0.2764263 1
## [5,] 0.6683103 0.3316897 1
## [6,] 0.3507995 0.6492805 2
## [7,] 0.5357116 0.4642884 1
## [8,] 0.1839394 0.8160606 2
## [9,] 0.5245687 0.4754313 1
## [10,] 0.6143938 0.3856062 1
```

Кластеры в двумерном пространстве

```
plot(Bank_fuzzy, which=1, main="")
```

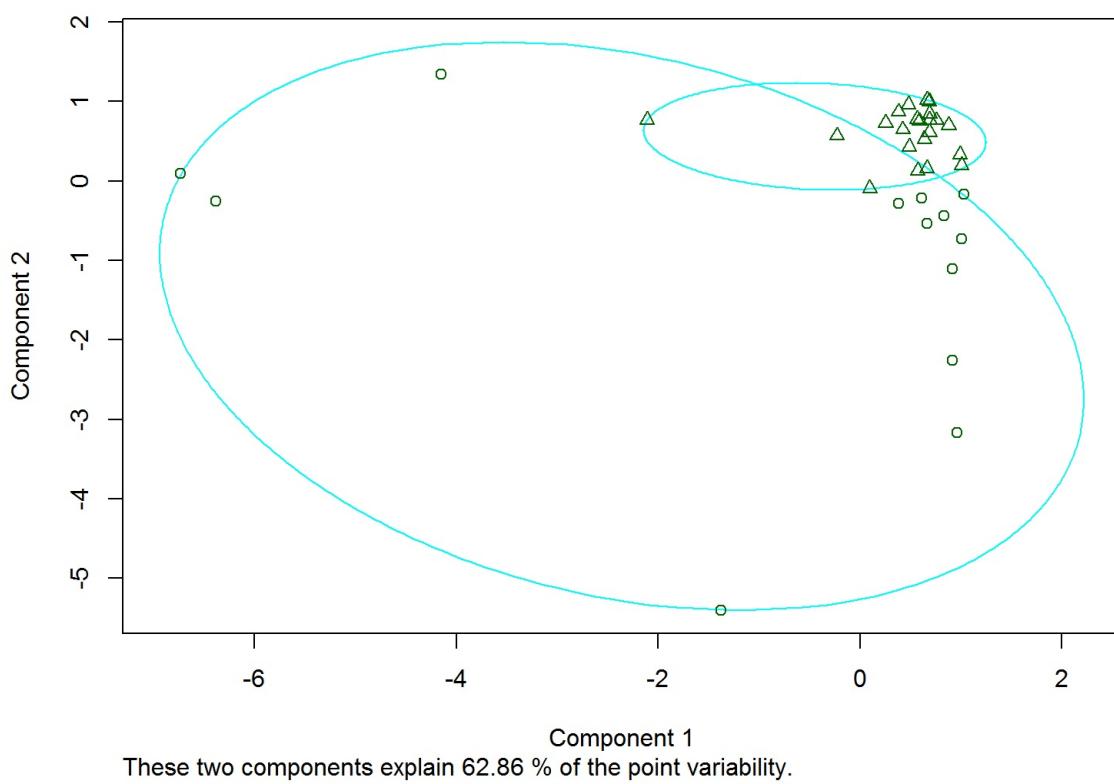
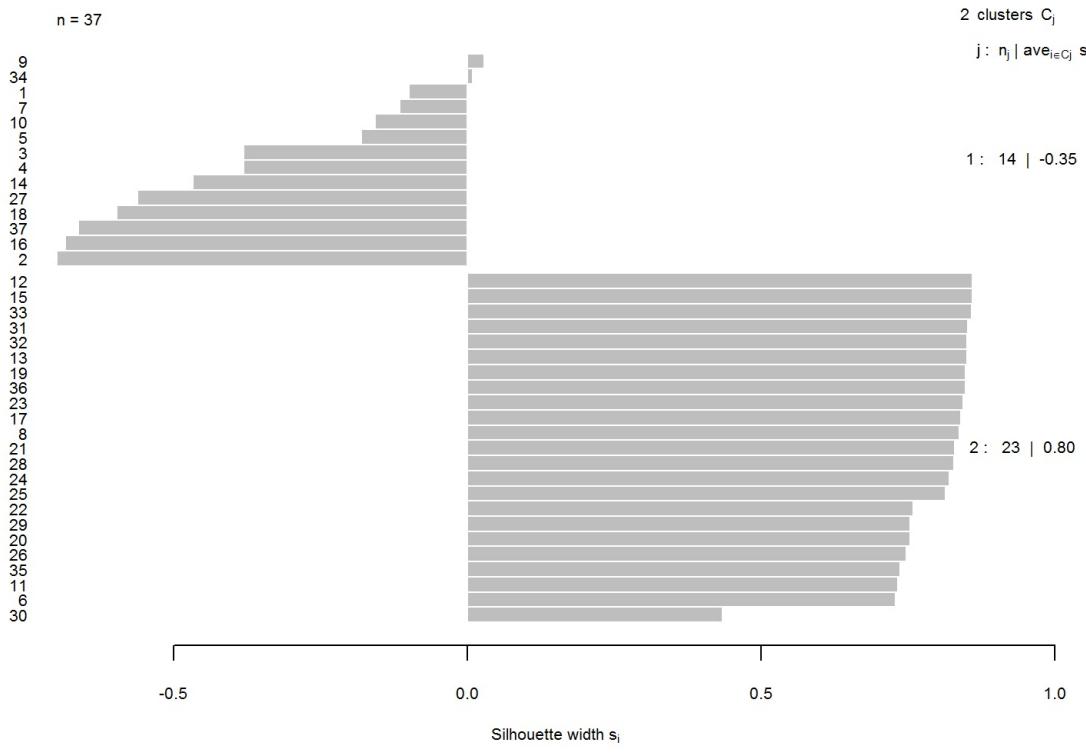


График силуэтов

Хорошо классифицированные объекты - ближе к 1.

Плохо классифицированные объекты - ближе к 0.

```
| opar=par(no.readonly = TRUE)
| par(ps=8) # меняем ширину 8pt
| plot(Bank_fuzzy, which=2, main="")
```



```
| par(opar)
```

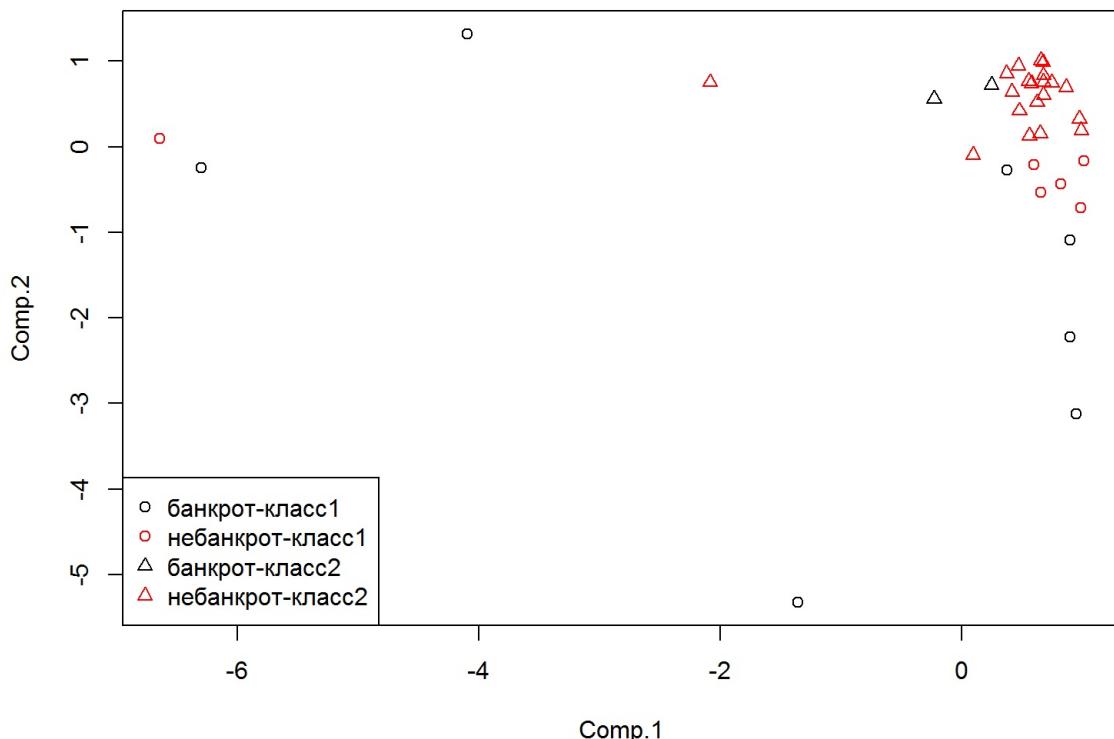
Удалось ли классифицировать банки с точки зрения вероятности банкротства?

```
table(Bank_fuzzy$clustering, Data_for_logit$Банкротство)
```

```
##  
##      банкрот небанкрот  
## 1      8      6  
## 2      2     21
```

Отобразим результат в пространстве главных компонент

```
plot(comp.prl[,1:2], pch=Bank.fuzzy$clustering, col=for.col),
legend("bottomleft", c("банкрот-класс1", "небанкрот-класс1",
"банкрот-класс2", "небанкрот-класс2"),
pch=rep(1:2, each=2), col=1:2)
```



Классификация с учителем

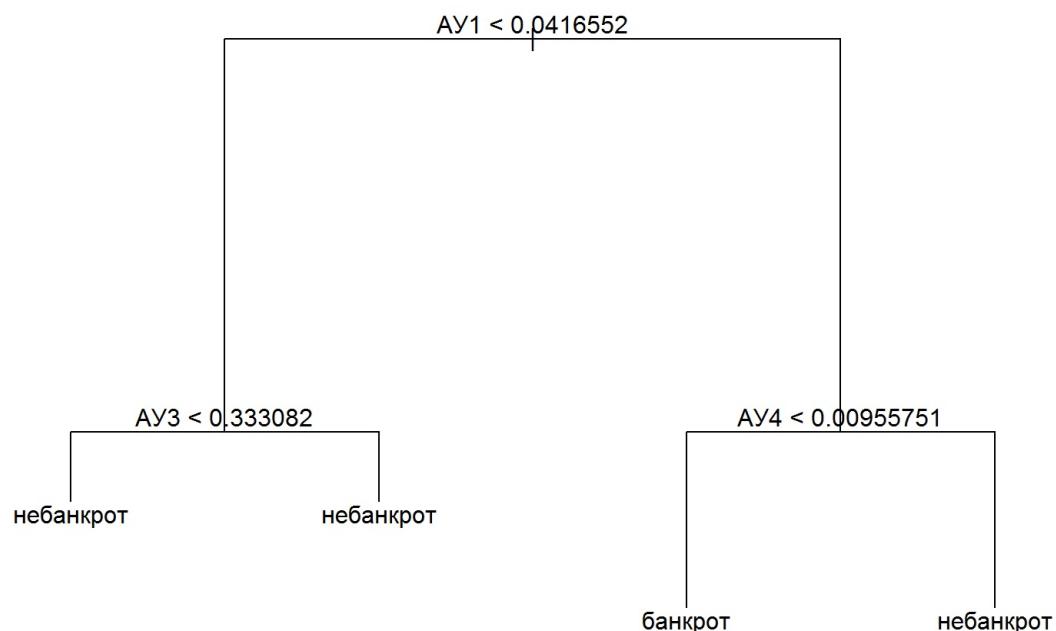
Деревья классификации

```
install.packages("tree")
```

```
library(tree)
```

Дерево классификации банков

```
Bank_tree=tree(Data_for_logit$Банкротство ~ .,
plot(Bank_tree)
text(Bank_tree)
```



Модели множественного выбора

Оценивание с помощью нейронных сетей - функция multinom{nnet}

```
install.packages("nnet")
```

```
library(nnet)
```

Моделирование статуса занятости

Воспользуемся индивидуальными данными Росстата

Считываем данных в формате .sav.

```
library(foreign)
FL=read.spss("C:/Users/MMNSK1/Documents/Данные Росстата/файлы в s
to.data.frame = TRUE)
```

Разбиение на обучающую и контрольную выборки

В качестве обучающей выборки возьмем данные о жителях Москвы

```
msk=levels(Fl$Ter)[35]  
Fl_msk=Fl[Fl$Ter==msk,]
```

В качестве контрольной выборки возьмем данные о жителях Санкт-Петербурга

```
spb=levels(Fl$Ter)[31]  
Fl_spb=Fl[Fl$Ter==spb,]
```

Удалим остальные данные

```
rm(Fl)
```

Уровни фактора, описывающего статус занятости

levels(FL_Msk\$R1v8)

```
## [1] "Не определено"  
## [2] "Работающий по найму (по письменному договору или устной дог)"  
## [3] "В отпуске по беременности и родам, по уходу за ребенком до 1"  
## [4] "В отпуске по уходу за ребенком от 1,5 до 3-х лет"  
## [5] "Работающий на собственном предприятии, в собственном деле"  
## [6] "Не работающий и ищущий работу (безработный)"  
## [7] "На пенсии (неработающий пенсионер)"  
## [8] "Учащийся, студент"  
## [9] "Занимающийся домашним хозяйством, уходом за детьми или други"  
## [10] "Временно или длительно нетрудоспособный"  
## [11] "Не работающий и не ищущий работу по другим причинам"
```

Объединим уровни: [2], [5] - "работают"

[1], [6], [10], [11] - "не работают"

[3], [4], [9] - "сидят дома"

[7] - "на пенсии"

[8] - "учатся"

Объединение уровней фактора

```
new_lev=vector(length=11)
new_lev[c(1,5,10,11)]="не работают"
new_lev[c(2,4,10,11)]="работают"
new_lev[c(3,4,9)]="сидят дома"
new_lev[7]="на пенсии"
new_lev[8]="учатся"
levels(FL_Msk$R1v8)=new_lev
levels(FL_Spb$R1v8)=new_lev
```

Проверим результат

```
summary(FL_Msk$R1v8)
```

	## не работают	работают	сидят дома	на пенсии	учатся
##	263	2234	47	404	196

Удалим в переменной “пол” лишний уровень

```
summary(FL_Msk$Pol) # пол
```

```
##   0    1    2  
##   0 1331 1813
```

```
levels(FL_Spb$Pol) = levels(FL_Msk$Pol)[c(2:2:3)]
```

Характеристика возраста

summary(FL_Msk\$R1v2)

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
##	0.00	30.00	46.00	43.79	57.00	94.00

Уровни фактора, описывающего семейное положение

levels(FL_Msk\$R1v5)

```
## [1] "Не определено"  
## [2] "Состоит в зарегистрированном браке"  
## [3] "Состоит в незарегистрированном браке"  
## [4] "Вдовец/вдова"  
## [5] "Разведен(а)"  
## [6] "Разошелся(ась)"  
## [7] "Никогда не состоял(а) в браке"
```

Объединим уровни

[4], [5], [6] - "после брака"

[2], [3] - "в браке"

[1], [7] - "до брака"

```
new_lev=vector(length=7)
new_lev[c(1,7)]="до брака"
new_lev[c(2,3)]="в браке"
new_lev[c(4,5,6)]="после брака"
levels(FL_Msk$R1v5)=new_lev
```

Проверим результат

```
summary(FL_Msk$R1v5)
```

	до брака	в браке	после брака
##	914	1649	581

Удалим в переменной “уровень образования” лишний уровень

```
summary(FL_Msk$R1v7)
```

```
##      Не определено Не имеет основного общего  
##           234                  26  
##      Основное общее          Среднее общее  
##            87                  332  
##  Среднее профессиональное Высшее профессиональное  
##          961                 1504
```

```
levels(FL_Msk$R1v7)[1:2] = levels(FL_Spb$R1v7)[2]
```

Оценивание модели множественного выбора

С помощью функции `multinom{nnet}`

```
multinom_test=multinom(R1v8 ~ Pol + R1v2 + R1v5 + R1v7, FL_Msk,
```

```
## # weights: 50 (36 variable)
## initial value 5060.072797
## iter 10 value 1992.738703
## iter 20 value 1201.806702
## iter 30 value 1026.869673
## iter 40 value 990.241521
## iter 50 value 987.602388
## iter 60 value 986.981571
## iter 70 value 986.819519
## final value 986.809856
## converged
```

Результаты оценивания

summary(multinom_test)

```
## Call:  
## multinom(formula = R1v8 ~ Pol + R1v2 + R1v5 + R1v7, data = FL_Msk,  
##           maxit = 1000)  
##  
## Coefficients:  
## (Intercept)      Pol2      R1v2 R1v5в браке R1v5после брака  
## работают     -13.873166  0.7360949  0.12643688  -1.053477   -2.134979  
## сидят дома   -17.736754 12.4379504  0.04293307  5.041464    2.178247  
## на пенсии     -23.162544  1.7612751  0.37118647  -1.711368   -2.895378  
## учатся        -2.112839  0.5475439  -0.06832526  -9.334054   -9.271709  
##  
## R1v7Основное общее R1v7Среднее общее  
## работают          10.280378   14.221316  
## сидят дома        1.429843   1.216445  
## на пенсии         2.449150   7.619243  
## учатся            5.818659   8.115837  
##  
## R1v7Среднее профессиональное R1v7Высшее профессиональное  
## работают          14.245591   14.0630307  
## сидят дома        0.7668079   0.9488132  
## на пенсии         7.4468787   6.8596191  
## учатся            5.6796055   3.8525867  
##  
## Std. Errors:  
## (Intercept)      Pol2      R1v2 R1v5в браке R1v5после брака  
## работают     0.5611872  0.3373689  0.02223281 0.519773774   0.669393886  
## сидят дома   0.5466992  0.5476199  0.02704769 1.735595025  1.829544354  
## на пенсии     0.9293507  0.3889699  0.02542963 0.662083151   0.791094849  
## учатся        0.3968953  0.3335168  0.03367195 0.004407404   0.002763955  
##  
## R1v7Основное общее R1v7Среднее общее  
## работают          0.7000710   0.6153977  
## сидят дома        1.7849986   1.6600373  
## на пенсии         0.9263376   0.6919411  
## учатся            0.6822857   0.9429541  
##  
## R1v7Среднее профессиональное R1v7Высшее профессиональное  
## работают          0.4381319   0.3358005  
## сидят дома        1.4701848   1.4130316  
## на пенсии         0.5314448   0.4523525  
## учатся            0.8661918   0.8458844  
##  
## Residual Deviance: 1973.62  
## AIC: 2045.62
```

Предсказанные значения вероятностей

```
pp=fitted(multinom.test)
head(data.frame(round(pp,2),FL_Msk$R1v8))
```

```
##      не.работают работают сидят.дома на.пенсии учатся FL_Msk.R1v8
## 52542    0.01    0.97    0.00    0.02    0.00   работают
## 52543    0.00    0.12    0.00    0.00    0.87   учатся
## 52544    0.00    0.57    0.00    0.42    0.00   работают
## 52545    0.00    0.86    0.00    0.14    0.00   на пенсии
## 52546    0.03    0.95    0.02    0.00    0.00   работают
## 52547    0.01    0.12    0.00    0.00    0.87   учатся
```

Точность предсказания на обучающей выборке

```
predict_Msk=predict(multipom_test, FL_Msk, type = "class")  
table(predict_Msk,FL_Msk$RIV8)
```

```
##  
## predict_Msk  не работают работают сидят дома на пенсии учатся  
##  не работают      234      0      1      0     21  
##  работают        24    2120     43    139     18  
##  сидят дома       0      1      2      0      0  
##  на пенсии        1     88      1    265      0  
##  учатся           4     25      0      0    157
```

Точность предсказания на контрольной выборке

```
predict_Spb=predict(multinom_test, FL_Spb, type = "class")
table(predict_Spb,FL_Spb$RIV8)
```

```
## predict_Spb  не работают работают сидят дома на пенсии учатся
##  не работают      285       1       0       0       5
##  работают          7     1415      39      61       7
##  сидят дома        0       0       0       0       0
##  на пенсии          0      57       0     150       0
##  учатся            1      10       0       0      89
```

Классификация с помощью алгоритма “Случайный лес”

```
install.packages("randomForest")
library(randomForest)
```

В основе метода лежит построение большого числа деревьев классификации

```
set.seed(1239)
Status_rf=randomForest(R1v8 ~ Pol + R1v2 + R1v5 + R1v7, FL_Msk)
```

Точность предсказания на контрольной выборке

Смоделируем статус занятости жителей СПб и сравним предсказанные и фактические значения.

```
predict_Spb_rF = predict(Status_rf, FL_Spb)
```

```
## predict_Spb_rF не работают работают сидят дома на пенсии учатся
##   не работают    205      3      0      0      0
##   работают        7    1404     39     73     10
##   сидят дома      0      0      0      0      0
##   на пенсии        0     73      0    138      0
##   учатся           1      3      0      0     91
```

Факторный анализ

Цель - выявить скрытые факторы, объясняющие наблюдаемые корреляции между признаками

Пример: социальный статус жителя Москвы

```
library(psych)
```

Создадим таблицу данных с характеристиками жителей Москвы

```
data_FA=FL_Msk[,c("R1v8","Pol","R1v2","R1v5","R1v7")]
```

Статус занятости имеет неупорядоченные уровни.

Для построения корреляционной матрицы уровни должны быть упорядочены.

Оставим два варианта: работают или нет.

```
levels(data_FA$R1v8)[-2]="не работают"
```

Построим матрицу полихорических корреляций

С помощью функции polychor{polycor}

```
library(polycor)
```

```
polycor_matr=diag(5)
for(i in 1:4){
  for(j in (i+1):5){
    polycor_matr[i,j]=polycor_matr[j,i]=
    polychor(data_FA[,i],data_FA[,j])
  }
}
rownames(polycor_matr)=c("Работа", "Пол", "Возраст",
colnames(polycor_matr)=c("Работа", "Пол", "Возраст",
lowerMat(polycor_matr)
```

```
##      Работа Пол Возра Брак Ораз-
## Работа 1.00
## Пол -0.07 1.00
## Возраст 0.09 0.18 1.00
## Брак 0.26 0.27 0.71 1.00
## Образ-е 0.68 0.11 0.35 0.35 1.00
```

Факторный анализ с ортогональным вращением

```
FA_varimax=fa(polycor_matr, nfactors=2  
               fm="gls", rotate="varimax", n.iter=1)  
FA_varimax$loadings
```

```
##  
## Loadings:  
##          GLS1   GLS2  
## Работа      0.877  
## Пол        0.242  
## Возраст    0.834  0.143  
## Брак       0.829  0.256  
## Образ-е    0.246  0.797  
##  
##          GLS1   GLS2  
## SS loadings  1.502 1.499  
## Proportion Var 0.300 0.298  
## Cumulative Var 0.300 0.598
```

Факторный анализ с косоугольным вращением

```
FA_oblimin=fa(polyccor_matr, nfactors=2  
fm="gls", rotate="oblimin", n.iter=1)
```

```
## Loading required namespace: GPArotation
```

```
FA_oblimin$loadings
```

```
##  
## Loadings:  
##          GLS1    GLS2  
## Работа     0.902  
## Пол       0.254  
## Возраст   0.861  
## Брак      0.845  
## Образ-е   0.183  0.759  
##  
##          GLS1    GLS2  
## SS loadings 1.562 1.400  
## Proportion Var 0.312 0.280  
## Cumulative Var 0.312 0.592
```

Иерархический факторный анализ

Используем пакет `sem` для структурного моделирования

```
install.packages("sem")
```

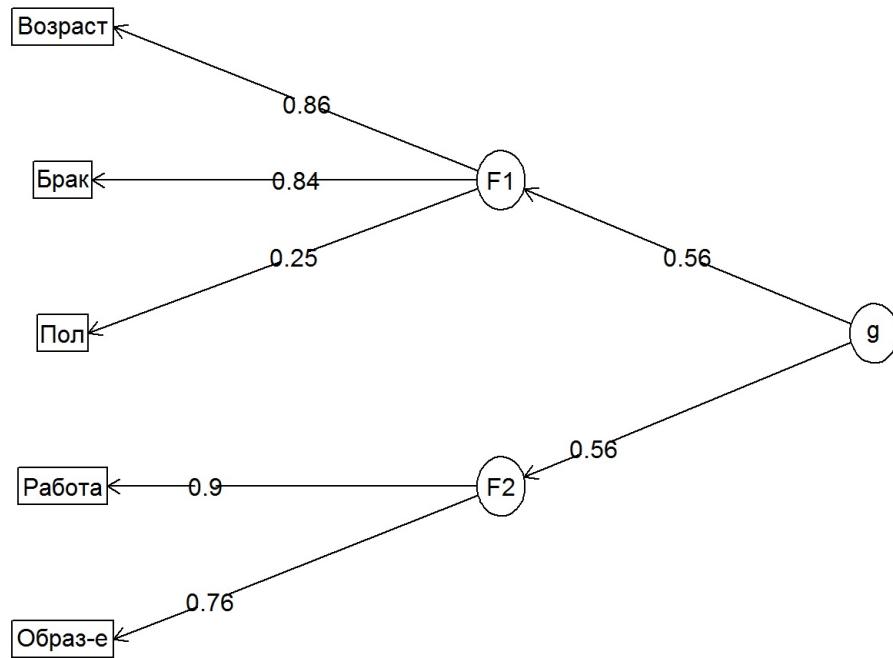
```
library(sem)
```

Графическое представление

```
hierarchFA<-omega(polykor.matr, nfactors=2, fm="gls", n.iter=1, digits=3, sl=FALSE)
```

```
## Three factors are required for identification -- general factor loadings set to be equal.  
## Proceed with caution.  
## Think about redoing the analysis with alternative values of the 'option' setting.
```

Omega



Анализ ассоциативных правил

Используем пакет arules

```
install.packages("arules")
```

```
library(arules)
```

```
## Loading required package: Matrix
```

```
##
```

```
## Attaching package: 'arules'
```

```
## The following objects are masked from 'package:base':
```

```
##
```

```
##     abbreviate, write
```

Категоризация непрерывной переменной “возраст”

```
data_FA$R1v2=cut(data_FA$R1v2,c(0,14,21,30,40,50,60,100),  
summary(data_FA$R1v2)
```

```
##   [0,14] (14,21] (21,30] (30,40] (40,50] (50,60] (60,100]  
##     234    164    436    447    622    648    593
```

Отображение всех обнаруженных правил

```
rules.all=apriori(data_FA)
```

```
inspect(rules.all)
```

```
##      lhs                      rhs          support confidence      lift
## [1] {R1v2=(21,30)} => {R1v8=работают} 0.1192748 0.8660917 1.210442
## [2] {R1v2=(30,40)} => {R1v8=работают} 0.1342239 0.9440716 1.328631
## [3] {R1v5=после брака} => {Pol=2}        0.1583966 0.8571429 1.486408
## [4] {R1v2=(40,50)} => {R1v8=работают} 0.1914758 0.9678457 1.362089
## [5] {R1v2=(50,60)} => {R1v8=работают} 0.1882952 0.9135802 1.285719
## [6] {R1v7=Высшее профессиональное} => {R1v8=работают} 0.4125318 0.8623670 1.213645
## [7] {R1v5=в браке}           => {R1v8=работают} 0.4395674 0.8380837 1.179470
## [8] {R1v8=работают},
##     R1v5=после брака}       => {Pol=2}        0.1071883 0.8467337 1.468357
## [9] {R1v2=(40,50),
##     R1v7=Высшее профессиональное} => {R1v8=работают} 0.1094148 0.9662921 1.359903
## [10] {R1v2=(40,50),
##      R1v5=в браке}           => {R1v8=работают} 0.1361323 0.9683258 1.362765
## [11] {Pol=2,
##      R1v2=(40,50)}          => {R1v8=работают} 0.1141858 0.9547872 1.343761
## [12] {R1v2=(50,60),
##      R1v5=в браке}           => {R1v8=работают} 0.1370865 0.9150743 1.287822
## [13] {Pol=2,
##      R1v2=(50,60)}          => {R1v8=работают} 0.1020992 0.8699187 1.224272
## [14] {Pol=1,
##      R1v7=Среднее профессиональное} => {R1v8=работают} 0.1071883 0.8575064 1.206804
## [15] {R1v5=в браке,
##      R1v7=Среднее профессиональное} => {R1v8=работают} 0.1574427 0.8263773 1.162995
## [16] {Pol=1,
##      R1v7=Высшее профессиональное} => {R1v8=работают} 0.1708015 0.9040404 1.272293
## [17] {Pol=1,
##      R1v5=в браке}
## [18] {R1v5=в браке,
##      R1v7=Высшее профессиональное} => {R1v8=работают} 0.2302799 0.8786408 1.236547
## [19] {Pol=2,
##      R1v7=Высшее профессиональное} => {R1v8=работают} 0.2449109 0.8651685 1.217587
## [20] {Pol=1,
##      R1v5=в браке,
##      R1v7=Высшее профессиональное} => {R1v8=работают} 0.2417303 0.8351648 1.175362
## [21] {Pol=2,
##      R1v5=в браке,
##      R1v7=Высшее профессиональное} => {R1v8=работают} 0.1221374 0.9014085 1.268589
```

Отображение правил с следствием “работают”/“не работают”

```
rules.work=apriori(data_FA, control = list(verbose=F),  
parameter = list(minlen=2, supp=0.05, conf=0.8),  
appearance = list(rhs=c("R1v8=работают",  
"R1v8=не работают"), default=[lhs]))
```

Округление значений и сортировка правил

quality(rules.work)=round{quality(rules.work), digits=3)
rules.work.sorted=sort(rules.work, by=lift)
inspect(rules.work.sorted)

```
##   lhs                                rhs          support confidence lift
## [1] {R1v2=[0,14]}                  => {R1v8=не работают}  0.074    1.000 3.455
## [2] {R1v7=Не имеет основного общего} => {R1v8=не работают}  0.083    1.000 3.455
## [3] {R1v2=[0,14],                   R1v7=Не имеет основного общего} => {R1v8=не работают}  0.074    1.000 3.455
## [4] {R1v2=[0,14],                   R1v5=до брака}           => {R1v8=не работают}  0.074    1.000 3.455
## [5] {R1v5=до брака,                 R1v7=Не имеет основного общего} => {R1v8=не работают}  0.081    1.000 3.455
## [6] {R1v2=[0,14],                   R1v5=до брака,               R1v7=Не имеет основного общего} => {R1v8=не работают}  0.074    1.000 3.455
## [7] {R1v2=[14,21],                  R1v5=до брака}           => {R1v8=не работают}  0.050    0.969 3.349
## [8] {R1v2=[14,21]}                  => {R1v8=не работают}  0.050    0.963 3.329
## [9] {Pol=1,                         R1v2=[40,50],             R1v5=в браке}           => {R1v8=работают}    0.066    0.995 1.401
## [10] {Pol=1,                         R1v2=[40,50]}           => {R1v8=работают}    0.077    0.988 1.390
## [11] {Pol=1,                         R1v2=[30,40]}           => {R1v8=работают}    0.059    0.984 1.385
## [12] {R1v2=[40,50],                  R1v5=браке,               R1v7=Среднее профессиональное} => {R1v8=работают}    0.051    0.975 1.373
## [13] {Pol=1,                         R1v2=[50,60],             R1v5=браке}           => {R1v8=работают}    0.078    0.972 1.368
## [14] {Pol=1,                         R1v2=[50,60]}           => {R1v8=работают}    0.086    0.971 1.367
## [15] {R1v2=[40,50],                  R1v7=Среднее профессиональное} => {R1v8=работают}    0.070    0.969 1.364
## [16] {R1v2=[40,50],                  R1v5=в браке}           => {R1v8=работают}    0.136    0.968 1.363
## [17] {R1v2=[40,50]}                  => {R1v8=работают}    0.191    0.968 1.362
## [18] {R1v2=[40,50],                  R1v7=Высшее профессиональное} => {R1v8=работают}    0.109    0.966 1.360
## [19] {R1v2=[40,50],                  R1v5=браке,               R1v7=Высшее профессиональное} => {R1v8=работают}    0.076    0.964 1.356
## [20] {R1v2=[21,30],                  R1v5=до брака,               R1v7=Высшее профессиональное} => {R1v8=работают}    0.060    0.959 1.350
## [21] {Pol=2,                         R1v2=[40,50],             R1v7=Высшее профессиональное} => {R1v8=работают}    0.067    0.959 1.350
## [22] {Pol=2,                         R1v2=[40,50]}           => {R1v8=работают}    0.114    0.955 1.344
## [23] {R1v2=[50,60],                  R1v5=в браке,               R1v7=Высшее профессиональное} => {R1v8=работают}    0.066    0.950 1.337
## [24] {R1v2=[30,40]}                  => {R1v8=работают}    0.134    0.944 1.329
## [25] {Pol=2,                         R1v2=[40,50],             R1v5=в браке}           => {R1v8=работают}    0.070    0.944 1.329
## [26] {Pol=2,                         R1v5=до брака,               R1v7=Высшее профессиональное} => {R1v8=работают}    0.059    0.944 1.329
## [27] {R1v2=[50,60],                  R1v7=Высшее профессиональное} => {R1v8=работают}    0.095    0.943 1.327
## [28] {R1v5=до брака,                 R1v7=Высшее профессиональное} => {R1v8=работают}    0.100    0.940 1.323
## [29] {R1v2=[30,40],                  R1v7=Высшее профессиональное} => {R1v8=работают}    0.087    0.938 1.320
## [30] {R1v2=[21,30],                  R1v7=Высшее профессиональное} => {R1v8=работают}    0.086    0.934 1.314
## [31] {R1v2=[30,40],                  R1v5=в браке}           => {R1v8=работают}    0.084    0.920 1.294
## [32] {Pol=2,                         R1v2=[50,60],             R1v7=Высшее профессиональное} => {R1v8=работают}    0.056    0.917 1.290
## [33] {Pol=2,                         R1v2=[30,40]}           => {R1v8=работают}    0.075    0.915 1.288
## [34] {R1v2=[50,60],                  R1v5=в браке}           => {R1v8=работают}    0.137    0.915 1.288
## [35] {R1v2=[50,60]}                  => {R1v8=работают}    0.188    0.914 1.286
## [36] {R1v2=[30,40],                  R1v5=в браке,               R1v7=Высшее профессиональное} => {R1v8=работают}    0.054    0.909 1.279
## [37] {Pol=1,                         R1v7=Высшее профессиональное} => {R1v8=работают}    0.171    0.904 1.272
## [38] {Pol=2,                         R1v2=[30,40],             R1v7=Высшее профессиональное} => {R1v8=работают}    0.050    0.903 1.271
## [39] {Pol=1,                         R1v5=в браке,               R1v7=Высшее профессиональное} => {R1v8=работают}    0.122    0.901 1.269
## [40] {R1v2=[50,60],                  R1v7=Среднее профессиональное} => {R1v8=работают}    0.080    0.900 1.266
## [41] {R1v2=[50,60],                  R1v5=в браке,               R1v7=Среднее профессиональное} => {R1v8=работают}    0.061    0.897 1.262
## [42] {Pol=1,                         R1v5=в браке,               R1v7=Среднее профессиональное} => {R1v8=работают}    0.084    0.880 1.239
## [43] {Pol=1,                         R1v5=в браке}           => {R1v8=работают}    0.230    0.879 1.237
## [44] {Pol=1,                         R1v2=[21,30]}           => {R1v8=работают}    0.052    0.878 1.235
## [45] {Pol=2,                         R1v2=[50,60]}           => {R1v8=работают}    0.102    0.870 1.224
## [46] {R1v5=в браке,                 R1v7=Высшее профессиональное} => {R1v8=работают}    0.245    0.865 1.218
## [47] {R1v7=Высшее профессиональное} => {R1v8=работают}    0.413    0.862 1.214
## [48] {R1v2=[21,30]}                  => {R1v8=работают}    0.119    0.860 1.210
## [49] {Pol=1,                         R1v7=Среднее профессиональное} => {R1v8=работают}    0.107    0.858 1.207
## [50] {Pol=2,                         R1v2=[50,60],             R1v5=в браке}           => {R1v8=работают}    0.059    0.849 1.194
## [51] {Pol=2,                         R1v2=[21,30]}           => {R1v8=работают}    0.067    0.847 1.192
## [52] {R1v2=[21,30],                  R1v5=до брака}           => {R1v8=работают}    0.084    0.844 1.188
## [53] {R1v5=в браке}                  => {R1v8=работают}    0.440    0.838 1.179
## [54] {Pol=2,                         R1v7=Высшее профессиональное} => {R1v8=работают}    0.242    0.835 1.175
## [55] {Pol=2,                         R1v5=в браке,               R1v7=Высшее профессиональное} => {R1v8=работают}    0.123    0.832 1.171
## [56] {R1v5=в браке,                 R1v7=Среднее профессиональное} => {R1v8=работают}    0.157    0.826 1.163
```

Программные системы статистического анализа

Тимофеева Анастасия Юрьевна

1 декабря 2021

Структура лекции

- Визуализация временных рядов
- Декомпозиция временного ряда
- Построение трендовых моделей
- Построение моделей с трендом и сезонностью
- Анализ остатков: коррелограмма
- Модель Хольта-Уинтерса
- Модель ARIMA
- Автоподбор модели временного ряда
- Использование R Markdown для создания отчетов

Анализ временных рядов

Считываем данные

```
pp=read.csv2("Средние потребительские цены.csv")
str(pp)
```

```
## 'data.frame': 95 obs. of 6 variables:
## $ Дата : Factor w/ 95 levels "авр.08","авг.09",..: 81 49 9 41 33 25 1 73 65 57 ...
## $ Фарш.мясной: num 141 143 144 146 149 ...
## $ Яйца : num 37.2 36.6 38.6 36.8 33 ...
## $ Хлеб.ржаной: num 18.6 19.5 20.3 20.9 21.4 ...
## $ Картофель : num 15.8 16.5 17.3 18.6 21.4 ...
## $ Бананы : num 36.9 47 48.2 36.6 35 ...
```

```
pp$Дата
```

```
[1] фев.08 мар.08 апр.08 май.08 июн.08 июл.08 авг.08 сентябрь.08 окт.08 ноябрь.08
[11] дек.08 январь.09 февр.09 мар.09 апр.09 май.09 июнь.09 июль.09 август.09 сентябрь.09
[21] октябрь.09 ноябрь.09 декабрь.09 январь.10 февр.10 март.10 апр.10 май.10 июнь.10 июль.10
[31] август.10 сентябрь.10 октябрь.10 ноябрь.10 декабрь.10 январь.11 февр.11 март.11 апр.11 май.11
[41] июнь.11 июль.11 август.11 сентябрь.11 октябрь.11 ноябрь.11 декабрь.11 январь.12 февр.12 март.12
[51] апрель.12 май.12 июнь.12 июль.12 август.12 сентябрь.12 октябрь.12 ноябрь.12 декабрь.12 январь.13
[61] февраль.13 март.13 апрель.13 май.13 июнь.13 июль.13 август.13 сентябрь.13 октябрь.13 ноябрь.13
[71] декабрь.13 январь.14 февраль.14 март.14 апрель.14 май.14 июнь.14 июль.14 август.14 сентябрь.14
[81] октябрь.14 ноябрь.14 декабрь.14 январь.15 февраль.15 март.15 апрель.15 май.15 июнь.15 июль.15
[91] август.15 сентябрь.15 октябрь.15 ноябрь.15 декабрь.15
# 95 Levels: февр.08 мар.08 апр.08 май.08 июн.08 июл.08 авг.08 сентябрь.08 окт.08 ноябрь.08
```

Создание одного временного ряда

```
ts_pp1=ts(pp1[,2], start=c(2008, 2), end=c(2015, 12),
          frequency=12)
```

```
##      Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct
## 2008 141.49 142.80 143.74 146.44 148.86 150.86 153.87 160.03 164.51
## 2009 173.20 175.89 178.30 179.43 180.82 181.45 181.94 182.39 183.26 183.89
## 2010 182.86 183.19 183.32 183.89 184.92 186.03 185.77 186.27 187.40 188.82
## 2011 197.23 200.91 202.82 203.53 205.31 206.73 207.58 209.07 211.69 213.93
## 2012 223.17 223.83 224.34 225.42 226.45 227.46 228.97 230.64 231.53 234.24
## 2013 235.10 234.86 234.88 234.33 234.84 234.94 235.20 234.85 237.33 238.68
## 2014 243.15 243.93 245.92 249.08 254.78 259.63 261.20 264.33 269.31 272.80
## 2015 287.09 295.59 300.43 302.01 302.43 302.20 302.44 303.81 306.07 306.95
##      Nov   Dec
## 2008 168.02 170.00
## 2009 183.95 184.36
## 2010 190.73 192.73
## 2011 217.58 219.72
## 2012 235.59 236.76
## 2013 239.98 240.76
## 2014 274.14 278.71
## 2015 307.04 307.54
```

Создание нескольких временных рядов

```
ts_pp=sapply(pp[,-1], ts, start=c(2008, 2), end=c(2015, 12),  
frequency=12, simplify =FALSE)  
ts_pp$Фарш.мясной
```

```
##      Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct  
## 2008 141.49 142.80 143.74 146.44 148.86 150.86 153.87 160.03 164.51  
## 2009 173.20 175.89 178.30 179.43 180.82 181.45 181.94 182.39 183.26 183.89  
## 2010 182.86 183.19 183.32 183.89 184.92 186.03 185.77 186.27 187.40 188.82  
## 2011 197.23 200.91 202.82 203.53 205.31 206.73 207.58 209.07 211.69 213.93  
## 2012 223.17 223.83 224.34 225.42 226.45 227.46 228.97 230.64 231.53 234.24  
## 2013 235.10 234.86 234.88 234.33 234.84 234.94 235.20 234.85 237.33 238.68  
## 2014 243.15 243.93 245.92 249.08 254.78 259.63 261.20 264.33 269.31 272.80  
## 2015 287.09 295.59 300.43 302.01 302.43 302.20 302.44 303.81 306.07 306.95  
##      Nov   Dec  
## 2008 168.02 170.00  
## 2009 183.95 184.36  
## 2010 190.73 192.73  
## 2011 217.58 219.72  
## 2012 235.59 236.76  
## 2013 239.98 240.76  
## 2014 274.14 278.71  
## 2015 307.04 307.54
```

Извлечение данных за указанный период

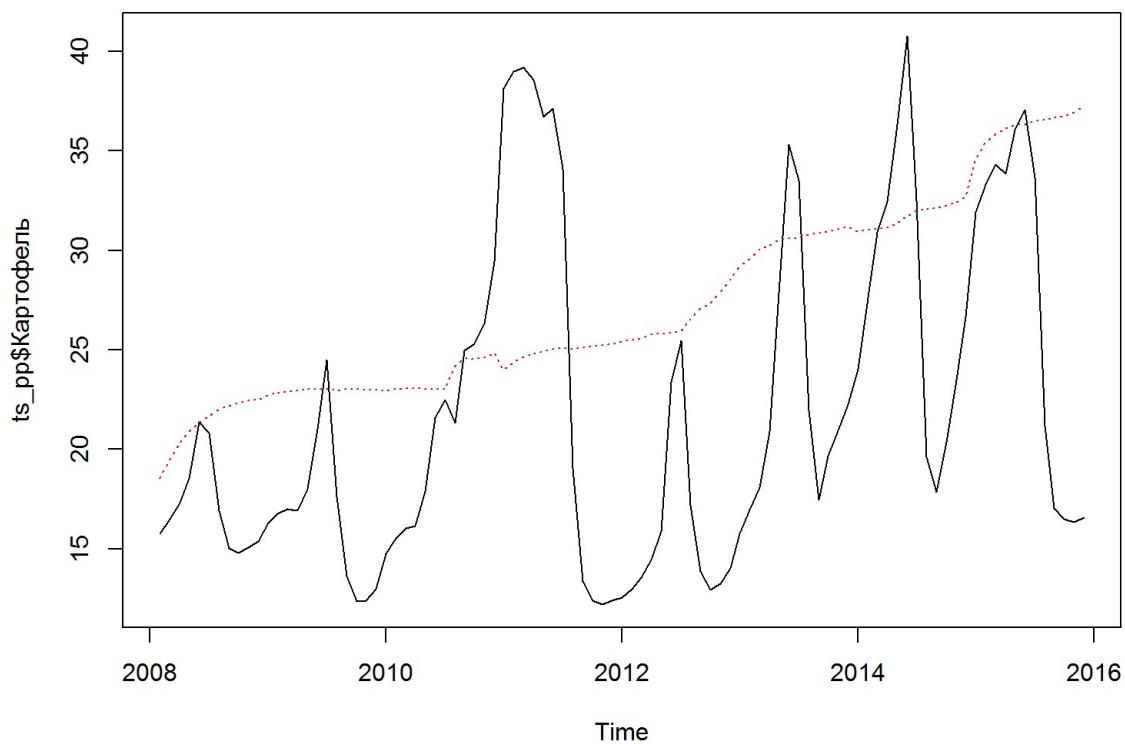
Извлечем данные только за 2014 год

```
Фарш2014=window(ts_pp$Фарш·мясной, start=c(2014, 1),  
Фарш2014
```

```
##          Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct  
## 2014 243.15 243.93 245.92 249.08 254.78 259.63 261.20 264.33 269.31 272.80  
##          Nov   Dec  
## 2014 274.14 278.71
```

График временного ряда

```
plot(ts_pp$Картофель)
lines(ts_pp$Хлеб.ржаной, col="red", lty=3)
```



Декомпозиция временного ряда

```
ts_decomp=stl(ts_pp$Картофель, s.window="period")  
ts_decomp$time.series
```

```
##      seasonal   trend  remainder  
## Feb 2008  0.5186306 14.95438  0.29698836  
## Mar 2008  1.4135992 15.30081 -0.20440510  
## Apr 2008  1.9823182 15.64723 -0.31954906  
## May 2008  4.0403574 15.94524 -1.43559316  
## Jun 2008  7.8108981 16.24324 -2.66413886  
## Jul 2008  6.4276004 16.47720 -2.08479698  
## Aug 2008 -2.6581990 16.71115  2.86704653  
## Sep 2008 -5.4149627 16.91568  3.52928612  
## Oct 2008 -5.3404763 17.12020  3.04027565  
## Nov 2008 -4.6784314 17.21565  2.55278032  
## Dec 2008 -3.5076362 17.31110  1.59653474  
## Jan 2009 -0.5937009 17.22118 -0.32748196  
## Feb 2009  0.5186306 17.13126 -0.85989491  
## Mar 2009  1.4135992 16.99973 -1.41333055  
## Apr 2009  1.9823182 16.86820 -1.91051668  
## May 2009  4.0403574 16.77533 -2.81568904  
## Jun 2009  7.8108981 16.68246 -3.47336298  
## Jul 2009  6.4276004 16.63860  1.44379528  
## Aug 2009 -2.6581990 16.59474  3.68345517  
## Sep 2009 -5.4149627 16.56088  2.48408416  
## Oct 2009 -5.3404763 16.52701  1.19346310  
## Nov 2009 -4.6784314 16.41160  0.67682969  
## Dec 2009 -3.5076362 16.29619  0.21144603  
## Jan 2010 -0.5937009 16.30028 -0.94658350  
## Feb 2010  0.5186306 16.30438 -1.29300928  
## Mar 2010  1.4135992 16.88253 -2.27613000  
## Apr 2010  1.9823182 17.46668 -3.30306122  
## May 2010  4.0403574 18.74938 -4.99974180  
## Jun 2010  7.8108981 20.03809 -6.24898396  
## Jul 2010  6.4276004 21.89254 -5.81013963  
## Aug 2010 -2.6581990 23.74699  0.24120635  
## Sep 2010 -5.4149627 25.83068  4.55428706  
## Oct 2010 -5.3404763 27.91436  2.74611771  
## Nov 2010 -4.6784314 29.60764  1.37079434  
## Dec 2010 -3.5076362 31.30092  1.64672070  
## Jan 2011 -0.5937009 31.93958  6.81412053  
## Feb 2011  0.5186306 32.57825  5.90312410  
## Mar 2011  1.4135992 31.95915  5.81725235  
## Apr 2011  1.9823182 31.34005  5.23763011  
## May 2011  4.0403574 29.85103  2.82861337  
## Jun 2011  7.8108981 28.36201  0.97709504  
## Jul 2011  6.4276004 26.33284  1.33956425  
## Aug 2011 -2.6581990 24.30366 -2.60546491  
## Sep 2011 -5.4149627 22.19777 -3.40281033  
## Oct 2011 -5.3404763 28.89188 -2.34140581  
## Nov 2011 -4.6784314 18.49225 -1.60381874  
## Dec 2011 -3.5076362 16.89262 -0.97498193  
## Jan 2012 -0.5937009 16.14221 -2.98850838  
## Feb 2012  0.5186306 15.39180 -2.92043107  
## Mar 2012  1.4135992 15.32957 -3.17317377  
## Apr 2012  1.9823182 15.26735 -2.80966696  
## May 2012  4.0403574 15.51599 -3.65634386  
## Jun 2012  7.8108981 15.76462 -0.20552234  
## Jul 2012  6.4276004 16.12870  2.90369992  
## Aug 2012 -2.6581990 16.49278  3.34542381  
## Sep 2012 -5.4149627 16.93466  2.37030481  
## Oct 2012 -5.3404763 17.37654  0.91393574  
## Nov 2012 -4.6784314 17.95231 -0.01386225  
## Dec 2012 -3.5076362 18.52809 -1.01045050  
## Jan 2013 -0.5937009 19.14489 -2.78118940  
## Feb 2013  0.5186306 19.76169 -3.32032454  
## Mar 2013  1.4135992 20.38120 -3.71479467  
## Apr 2013  1.9823182 21.00070 -2.07301527  
## May 2013  4.0403574 21.71385  2.09579592  
## Jun 2013  7.8108981 22.42700  5.10210552  
## Jul 2013  6.4276004 23.21428  3.86812091  
## Aug 2013 -2.6581990 24.00156  0.59663794  
## Sep 2013 -5.4149627 24.76018 -1.87521357  
## Oct 2013 -5.3404763 25.51879 -0.50831513  
## Nov 2013 -4.6784314 26.11250 -0.50467120  
## Dec 2013 -3.5076362 26.70621 -0.93857752  
## Jan 2014 -0.5937009 26.99761 -2.38390814  
## Feb 2014  0.5186306 27.28900 -0.17763501  
## Mar 2014  1.4135992 27.39422  2.09218020  
## Apr 2014  1.9823182 27.49944  2.98824492  
## May 2014  4.0403574 27.62644  4.67320554  
## Jun 2014  7.8108981 27.75344  5.22566457  
## Jul 2014  6.4276004 28.01245 -1.93005078  
## Aug 2014 -2.6581990 28.27146 -5.98326451  
## Sep 2014 -5.4149627 28.50736 -5.22239456  
## Oct 2014 -5.3404763 28.74325 -3.05277467  
## Nov 2014 -4.6784314 28.94024 -0.76180707  
## Dec 2014 -3.5076362 29.13723  1.02041827  
## Jan 2015 -0.5937009 29.30081  3.20288698  
## Feb 2015  0.5186306 29.46440  3.32696744  
## Mar 2015  1.4135992 29.06945  3.84695360  
## Apr 2015  1.9823182 28.67449  3.18318926  
## May 2015  4.0403574 27.84950  4.18014431  
## Jun 2015  7.8108981 27.02450  2.21459777  
## Jul 2015  6.4276004 26.13461  1.10759400  
## Aug 2015 -2.6581990 25.24511 -1.39698013  
## Sep 2015 -5.4149627 24.25334 -1.77837731  
## Oct 2015 -5.3404763 23.26157 -1.42109655  
## Nov 2015 -4.6784314 22.17731 -1.15887686  
## Dec 2015 -3.5076362 21.09304 -1.00540743
```

График декомпозиции временного ряда

```
plot(ts_decomp)
```

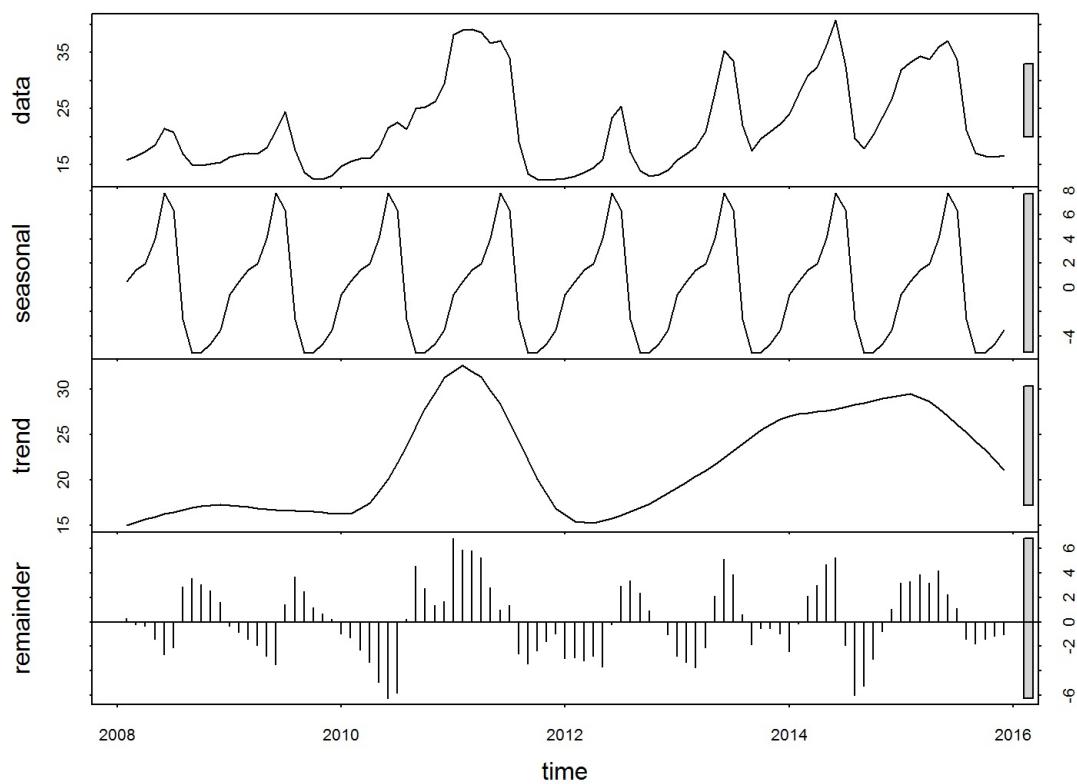


График декомпозиции временного ряда

```
plot(ts_pp$Картофель)
lines(ts_decomp$time.series[,2], lwd=2)
lines(ts_decomp$time.series[,1]+ts_decomp$time.series[,2], lty=2, c
```

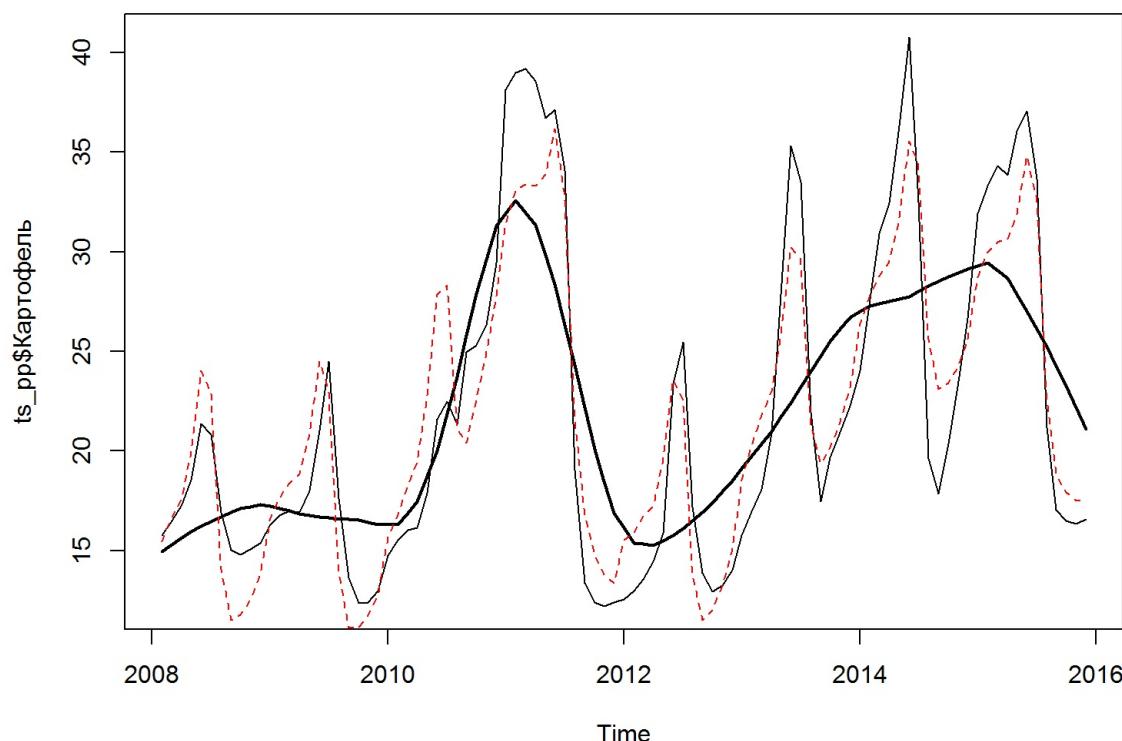


График с группировкой по месяцам

```
monthplot(ts_pp$Картофель)
```

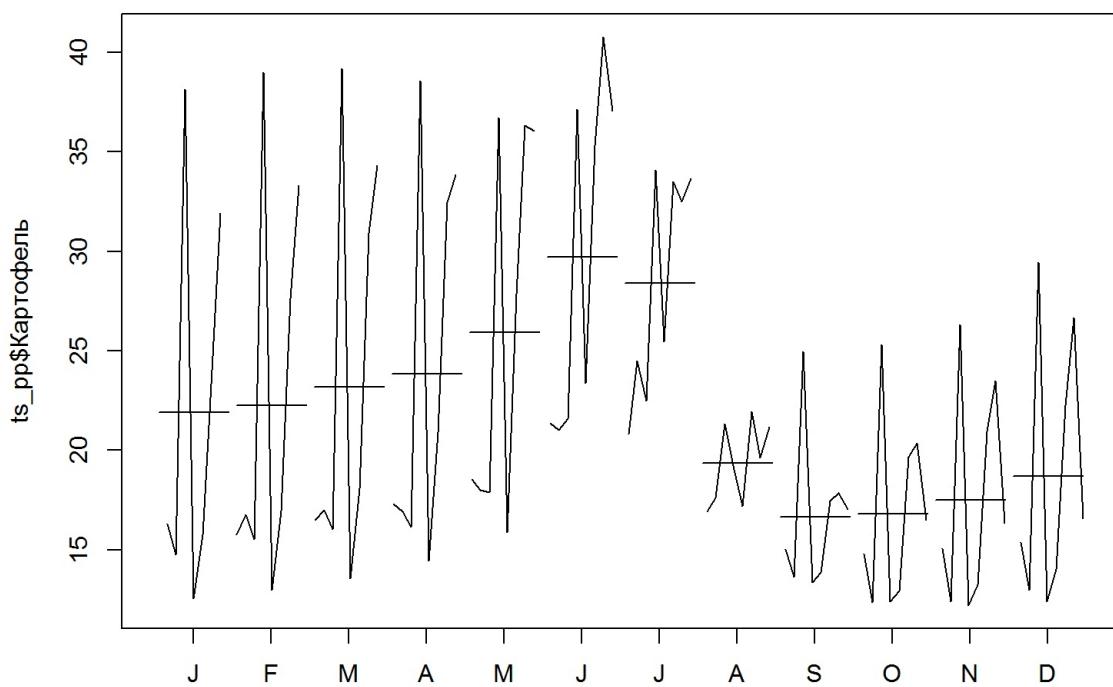
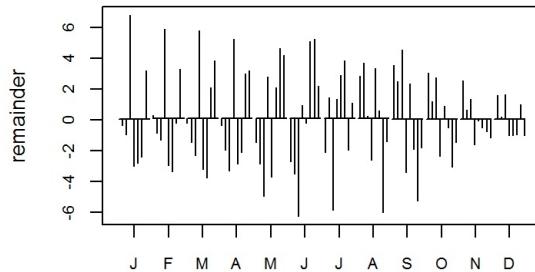
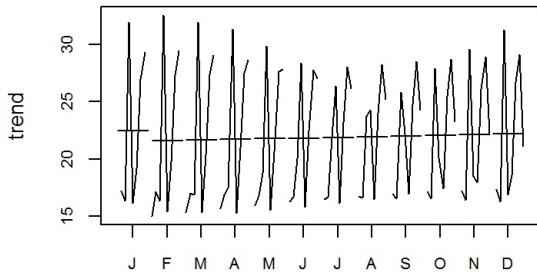
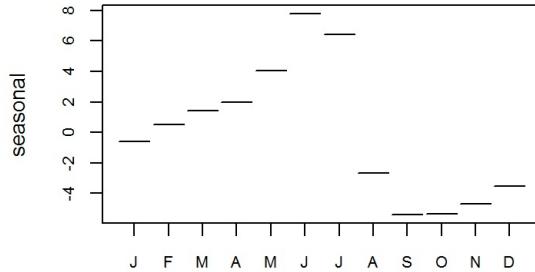
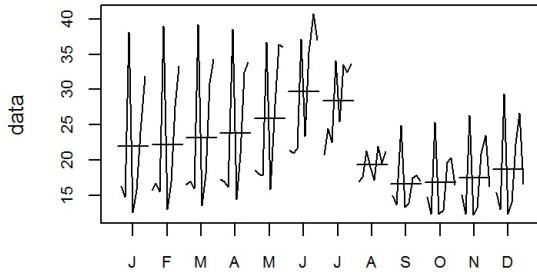


График с группировкой по месяцам

```
op=par(mfrow = c(2, 2))
monthplot(ts$ Картофель, ylab = "data", cex.axis = 0.8)
monthplot(ts$ Картофель, choice = "seasonal", cex.axis = 0.8)
monthplot(ts$ Картофель, choice = "trend", cex.axis = 0.8)
monthplot(ts$ Картофель, choice = "remainder", type = "h", cex.axis = 0.8)
```



```
par(op)
```

Использование пакета forecast

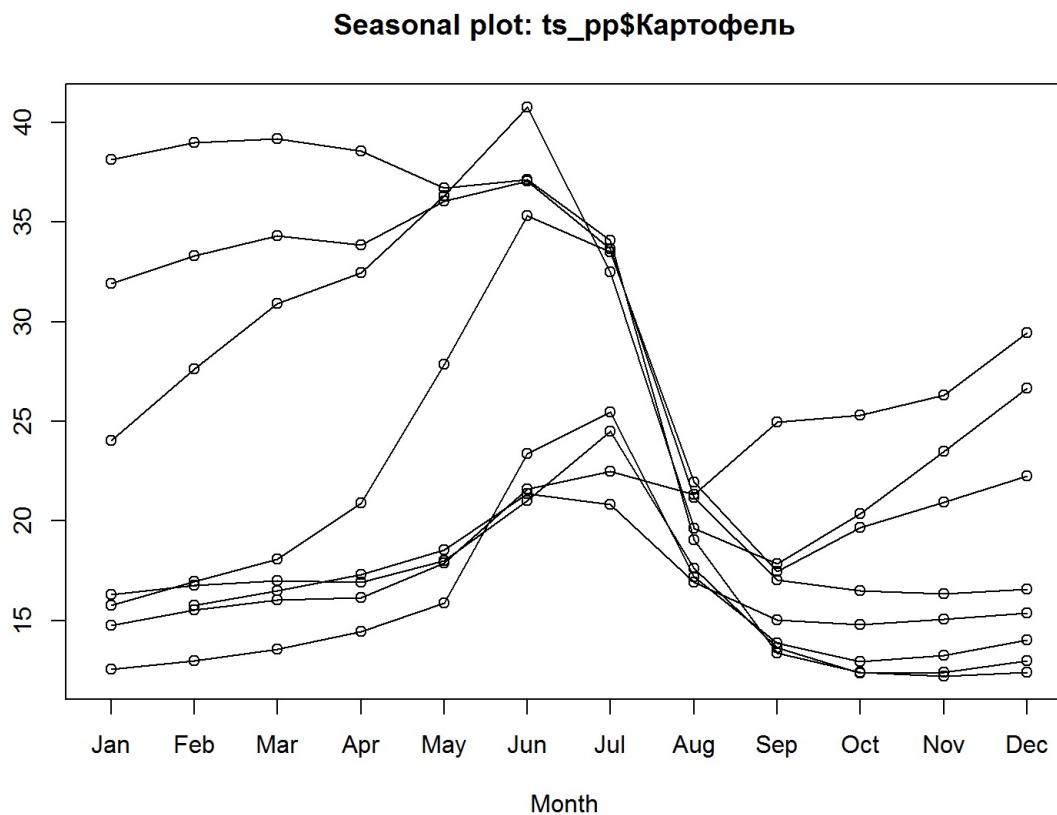
```
#install.packages("forecast")
library(forecast)
```

```
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##       as.Date, as.Date.numeric
##
## Loading required package: timeDate
##
## Warning in as.POSIXlt.POSIXct(Sys.time()): unable to identify current timezone ' ;B09A:>5 ':
## please set environment variable 'TZ'
##
## This is forecast 7.3
```

График с группировкой по месяцам

Данные за один год соединены одной линией

```
seasonplot(ts_pp$Картофель)
```



Линейная фильтрация временного ряда

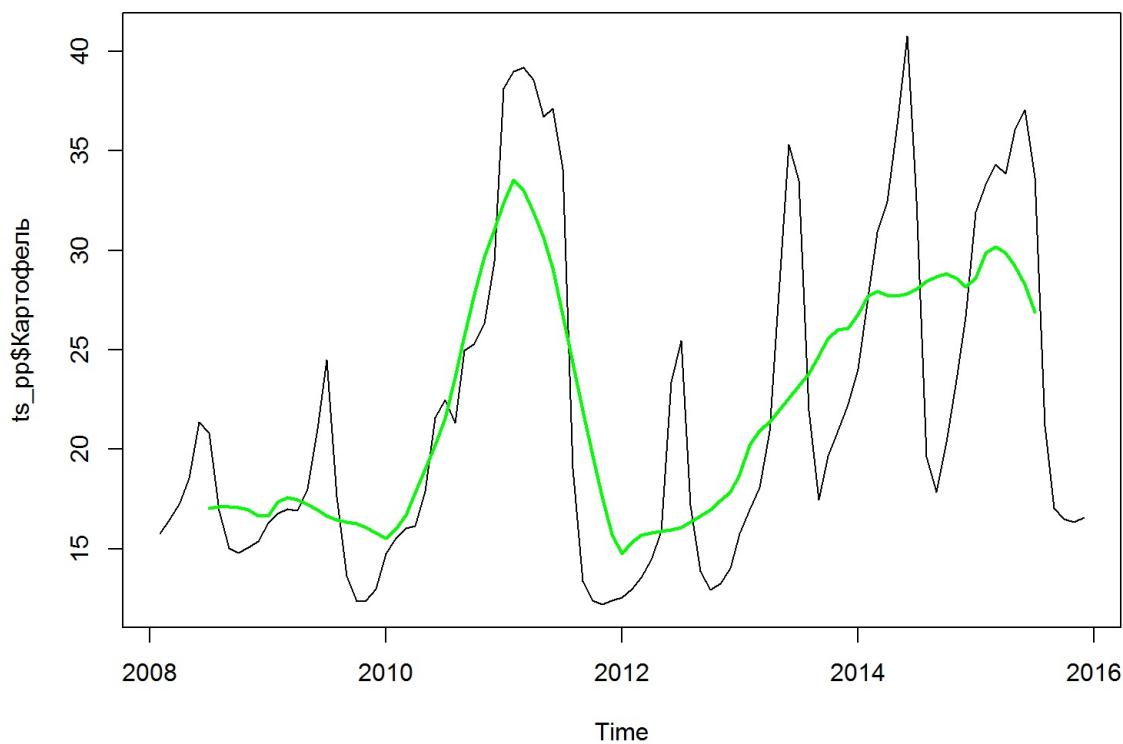
Скользящее среднее

filter задает веса (период - 11 месяцев с одинаковыми весами)

```
MA=filter(ts_pp$Картофель, filter=rep(1/11,11))
```

График скользящего среднего

```
plot(ts_pp$Картофель)  
lines(MA,col="green",lwd=2)
```

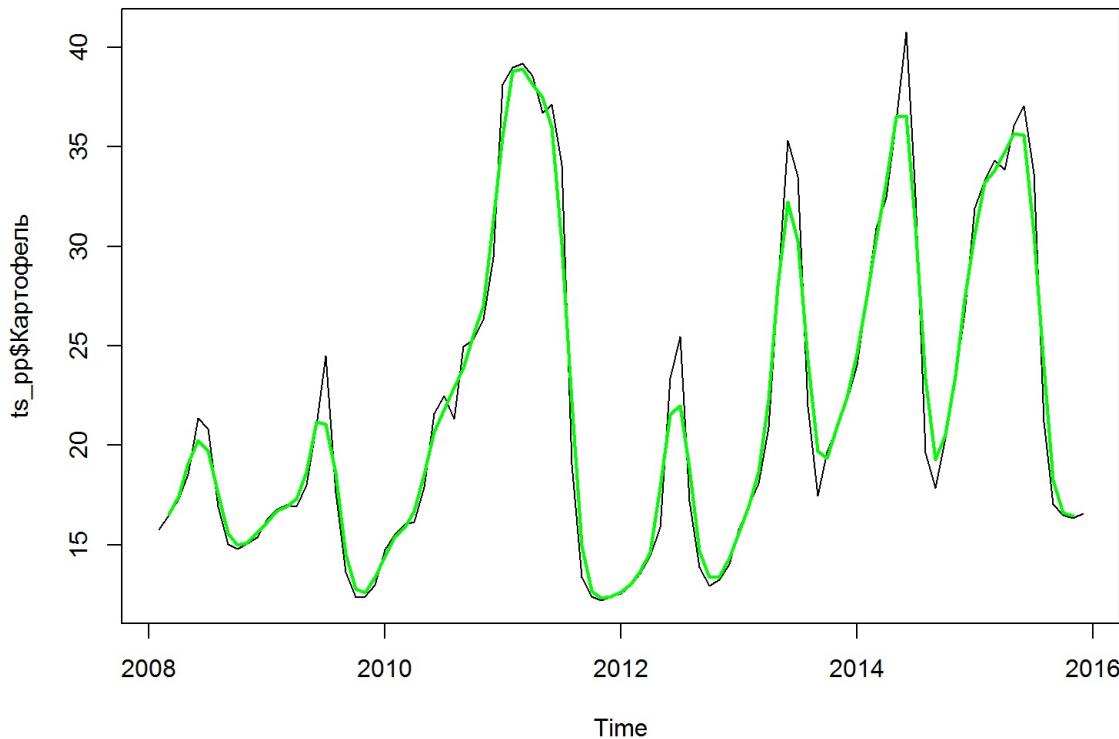


Скользящее среднее

Уменьшим период до 3 месяцев

Наблюдаемые колебания полностью воспроизводятся

```
MA=filter(ts_pp$Картофель, filter=rep(1/3,3))
plot(ts_pp$Картофель)
lines(MA,col="green",lwd=2)
```



Построение регрессионной модели тренда

Значения регрессора в трендовой модели

```
ts_pp$tr=time(ts_pp$Фарш_мясной)
ts_pp$tr=ts_pp$tr-ts_pp$tr[1] # приводим начало к нулевой отметке
```

```
##          Jan      Feb      Mar      Apr      May      Jun
## 2008 0.00000000 0.08333333 0.16666667 0.25000000 0.33333333
## 2009 0.91666667 1.00000000 1.08333333 1.16666667 1.25000000 1.33333333
## 2010 1.91666667 2.00000000 2.08333333 2.16666667 2.25000000 2.33333333
## 2011 2.91666667 3.00000000 3.08333333 3.16666667 3.25000000 3.33333333
## 2012 3.91666667 4.00000000 4.08333333 4.16666667 4.25000000 4.33333333
## 2013 4.91666667 5.00000000 5.08333333 5.16666667 5.25000000 5.33333333
## 2014 5.91666667 6.00000000 6.08333333 6.16666667 6.25000000 6.33333333
## 2015 6.91666667 7.00000000 7.08333333 7.16666667 7.25000000 7.33333333
##          Jul      Aug      Sep      Oct      Nov      Dec
## 2008 0.41666667 0.50000000 0.58333333 0.66666667 0.75000000 0.83333333
## 2009 1.41666667 1.50000000 1.58333333 1.66666667 1.75000000 1.83333333
## 2010 2.41666667 2.50000000 2.58333333 2.66666667 2.75000000 2.83333333
## 2011 3.41666667 3.50000000 3.58333333 3.66666667 3.75000000 3.83333333
## 2012 4.41666667 4.50000000 4.58333333 4.66666667 4.75000000 4.83333333
## 2013 5.41666667 5.50000000 5.58333333 5.66666667 5.75000000 5.83333333
## 2014 6.41666667 6.50000000 6.58333333 6.66666667 6.75000000 6.83333333
## 2015 7.41666667 7.50000000 7.58333333 7.66666667 7.75000000 7.83333333
```

Линейная модель

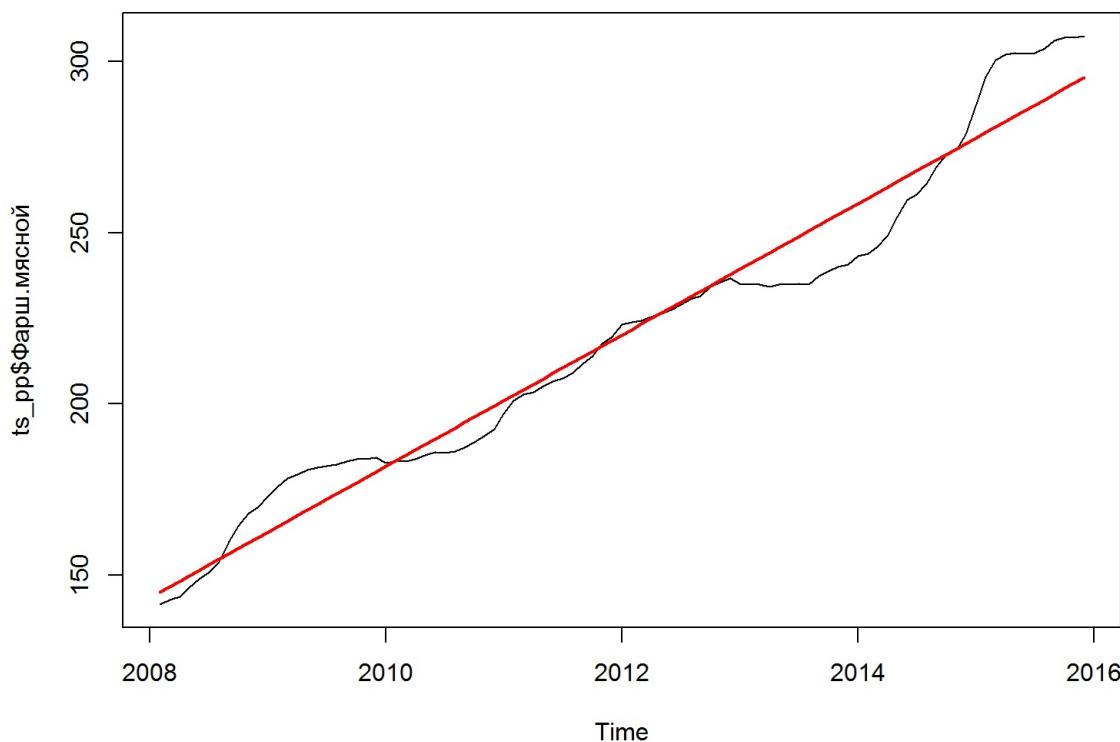
Аргумент na.action=NULL позволяет сохранить свойства временного ряда

```
tr_mod1=lm(Фарш.мясной~tr, data=ts_pp, na.action=NULL)
summary(tr_mod1)
```

```
## 
## Call:
## lm(formula = Фарш.мясной ~ tr, data = ts_pp, na.action = NULL)
## 
## Residuals:
##   Min     1Q Median     3Q    Max 
## -16.179 -5.506 -1.377  6.745 19.553 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 145.0901   1.8668  77.72 <2e-16 ***
## tr          19.1699   0.4117  46.56 <2e-16 ***  
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 9.17 on 93 degrees of freedom
## Multiple R-squared:  0.9589, Adjusted R-squared:  0.9584 
## F-statistic: 2168 on 1 and 93 DF, p-value: < 2.2e-16
```

График линейного тренда

```
plot(ts_pp$Фарш_мясной)
lines(tr_mod1$fit, col=2, lwd=2)
```



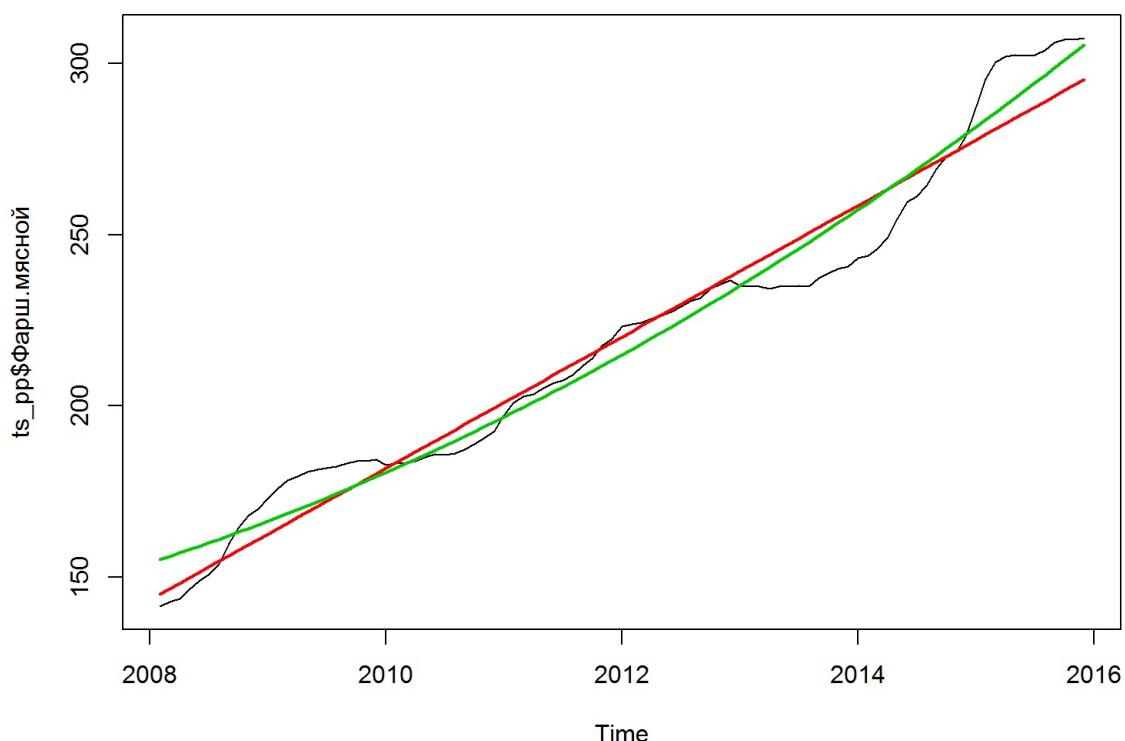
Квадратичная модель

```
tr_mod2=lm(Фарш.мясной~tr+I(tr^2),  
           data=ts_pp, na.action=NULL)
```

```
##  
## Call:  
## lm(formula = Фарш.мясной ~ tr + I(tr^2), data = ts_pp, na.action = NULL)  
##  
## Residuals:  
##      Min      1Q  Median      3Q     Max  
## -15.296 -5.007  2.157  5.822 14.743  
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 155.2192   2.3816 65.174 < 2e-16 ***  
## tr          11.3279   1.4052  8.061 2.69e-12 ***  
## I(tr^2)     1.0011   0.1736  5.767 1.07e-07 ***  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Residual standard error: 7.901 on 92 degrees of freedom  
## Multiple R-squared:  0.9698, Adjusted R-squared:  0.9691  
## F-statistic: 1477 on 2 and 92 DF,  p-value: < 2.2e-16
```

График квадратичного тренда

```
plot(ts_pp$Фарш.мясной)
lines(tr_mod1$fit, col=3, lwd=2)
lines(tr_mod2$fit, col=3, lwd=2)
```



Полином третьей степени

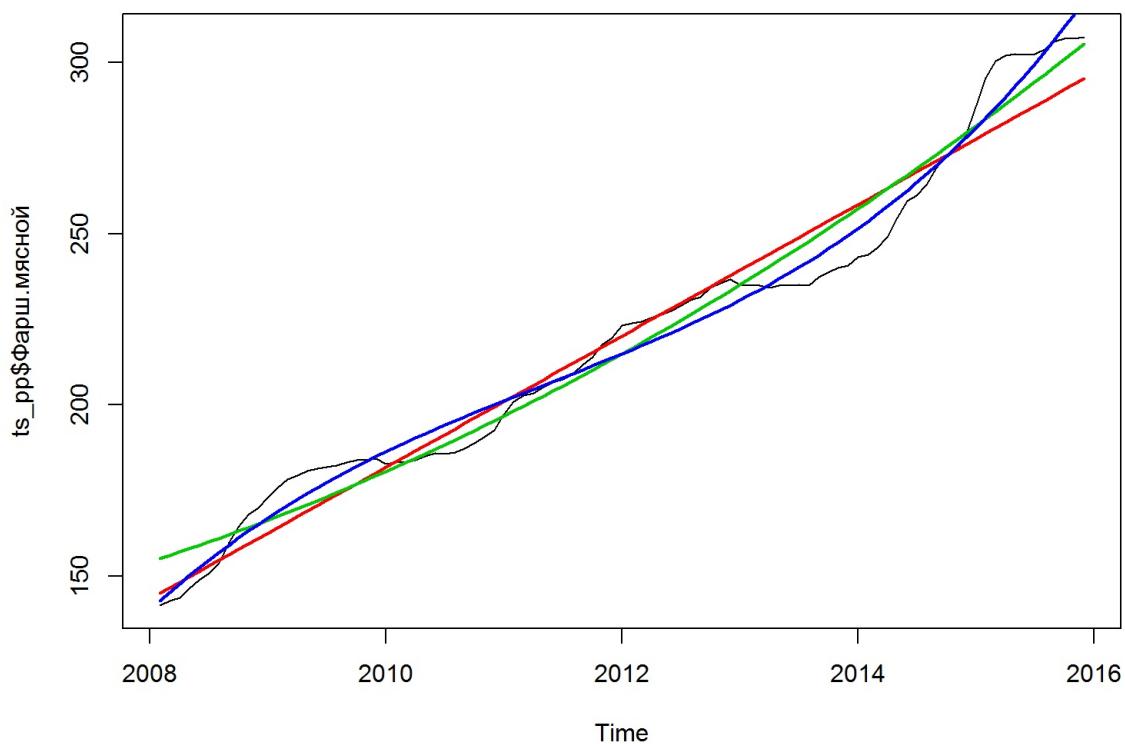
Не самая лучшая идея!

```
tr_mod3=lm(Фарш_мясной~tr+I(tr^2)+I(tr^3),
           data=ts_pp, na.action=NULL)
summary(tr_mod3)
```

```
## 
## Call:
## lm(formula = Фарш_мясной ~ tr + I(tr^2) + I(tr^3), data = ts_pp,
##      na.action = NULL)
##
## Residuals:
##   Min     1Q Median     3Q    Max 
## -10.2881 -4.2998 -0.4914  5.7492 13.6772 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 142.77415   2.39788 59.542 < 2e-16 ***
## tr          30.91263   2.66508 11.599 < 2e-16 ***
## I(tr^2)     -5.28252   0.79297 -6.662 2.02e-09 ***
## I(tr^3)      0.53478   0.06652  8.039 3.18e-12 ***
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.075 on 91 degrees of freedom
## Multiple R-squared:  0.9823, Adjusted R-squared:  0.9818 
## F-statistic: 1687 on 3 and 91 DF, p-value: < 2.2e-16
```

График полиномиального тренда

```
plot(ts_pp$Фарш_мясной)
lines(tr_mod1$fit, col=2, lwd=2)
lines(tr_mod2$fit, col=3, lwd=2)
lines(tr_mod3$fit, col=4, lwd=2)
```



Прогноз на будущее

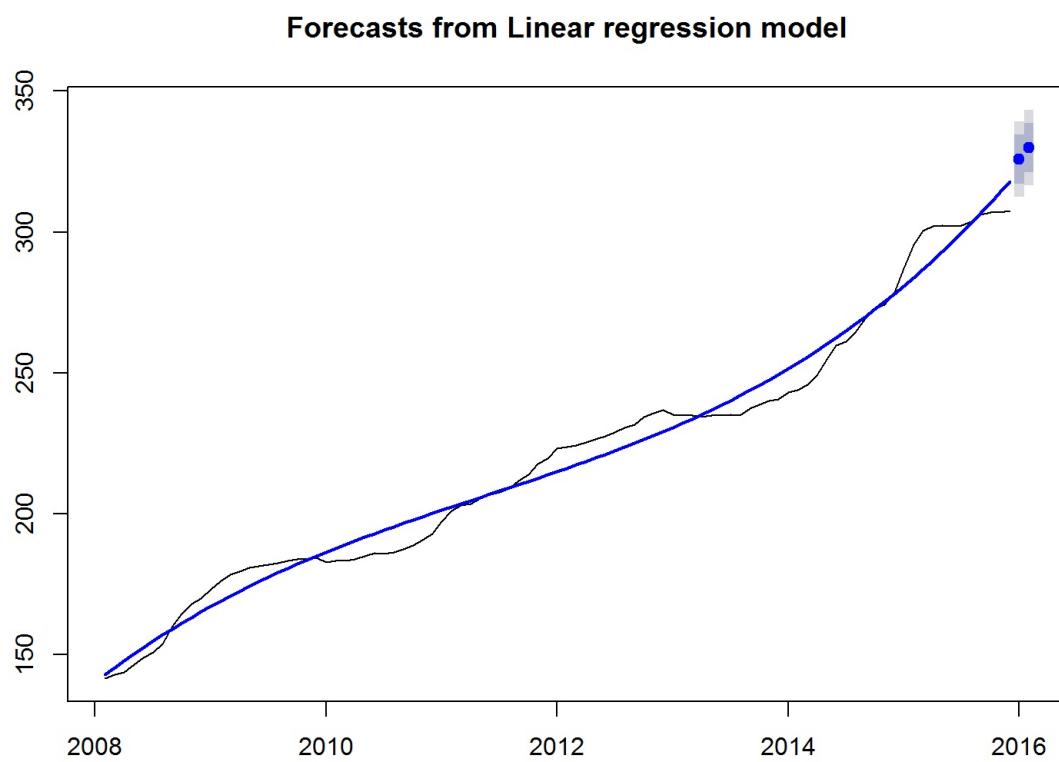
На два месяца вперед

В новых данных указываем янв. и февр. 2016 (8-й год)

```
tr_pr=data.frame(tr=c(8,8+1/12))
fcast1=forecast(tr_mod1, newdata=tr_pr)
fcast2=forecast(tr_mod2, newdata=tr_pr)
fcast3=forecast(tr_mod3, newdata=tr_pr)
```

Результат для полиномиальной модели

```
plot(fcast3)
lines(tr_mod3$fit, col=4, lwd=2)
```



Оценка сезонных эффектов одновременно с трендом

Создадим фактор сезонности

```
ts_pp$month=factor(cycle(ts_pp$Яйца))  
summary(ts_pp$month)
```

```
## 1 2 3 4 5 6 7 8 9 10 11 12  
## 7 8 8 8 8 8 8 8 8 8 8 8
```

Оценка модели с линейным трендом и сезонностью

```
tr_seas_mod=lm(Яйца~tr+month, data=ts_pp, na.action=NULL)
summary(tr_seas_mod)
```

```
## 
## Call:
## lm(formula = Яйца ~ tr + month, data = ts_pp, na.action = NULL)
## 
## Residuals:
##   Min     1Q Median     3Q    Max 
## -5.823 -3.349 -1.359  3.362 10.172 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 32.9347   1.8454 17.847 < 2e-16 ***
## tr          3.5054   0.1998 17.548 < 2e-16 ***  
## month2      -0.6449   2.2901 -0.282 0.778941  
## month3      -0.2771   2.2895 -0.121 0.903977  
## month4       0.4458   2.2891  0.195 0.846066  
## month5      -4.4238   2.2888 -1.933 0.056714 .  
## month6      -7.6734   2.2886 -3.353 0.001212 ** 
## month7      -8.3268   2.2886 -3.638 0.000478 *** 
## month8      -8.3764   2.2886 -3.660 0.000445 *** 
## month9      -5.8310   2.2888 -2.548 0.012713 *  
## month10     -3.3581   2.2891 -1.467 0.146200  
## month11     -1.4728   2.2895 -0.643 0.521853  
## month12      0.5601   2.2901  0.245 0.807388  
## --- 
## Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 4.422 on 82 degrees of freedom
## Multiple R-squared:  0.8162, Adjusted R-squared:  0.7893 
## F-statistic: 30.34 on 12 and 82 DF,  p-value: < 2.2e-16
```

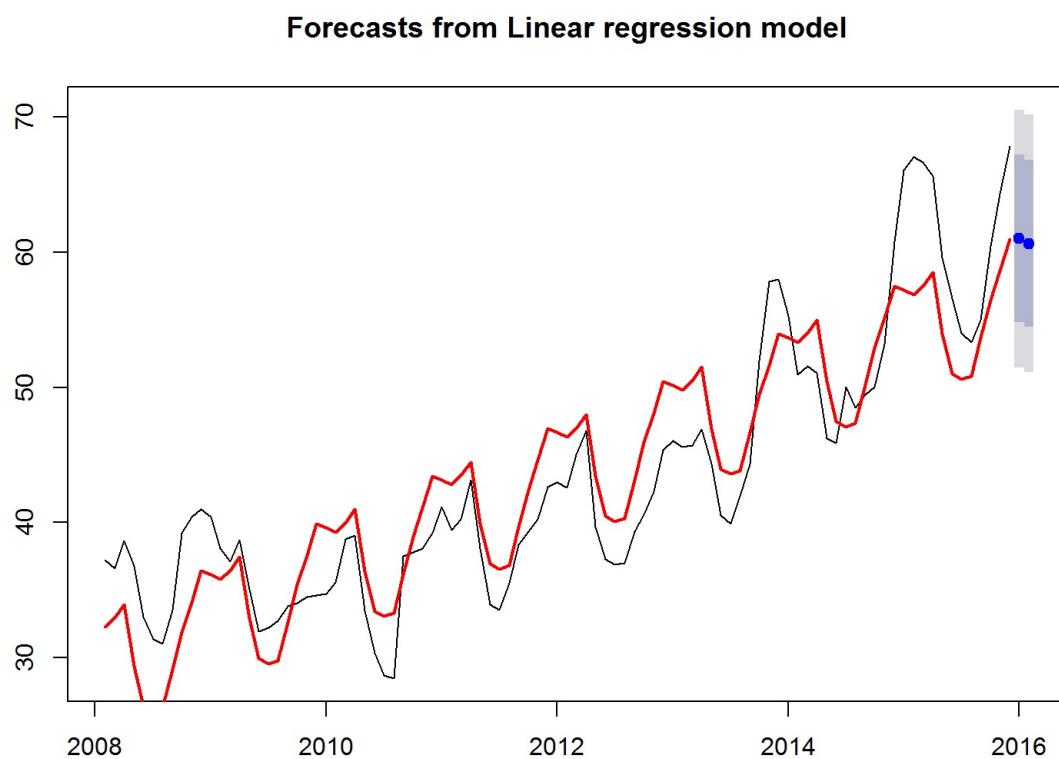
Прогноз на два шага вперед

```
tr_seas_pr=data.frame(tr=c(8,8+1/12),  
month=factor(c(1,2)),levels=levels(ts_pp$month)))  
fcast4=forecast(tr_seas_mod,newdata=tr_seas_pr)
```

```
##          Point Forecast    Lo 80     Hi 80    Lo 95     Hi 95  
## Jan 2016   60.97808 54.78039 67.17576 51.43509 70.52106  
## Feb 2016   60.62525 54.45135 66.79916 51.11888 70.13162
```

Прогноз на два шага вперед

```
plot(fcast4)
lines(tr_seas_mod$fit,col=2,lwd=2)
```



Оценка модели с квадратичным трендом и сезонностью

```
tr2_seas_mod=lm(Яйца~tr+I(tr^2)+month,
                 data=ts_pp, na.action=NULL)
summary(tr2_seas_mod)
```

```
## 
## Call:
## lm(formula = Яйца ~ tr + I(tr^2) + month, data = ts_pp, na.action = NULL)
##
## Residuals:
##   Min     1Q   Median     3Q    Max 
## -4.9134 -1.7409 -0.0944  1.3822  7.5865 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 40.95606   1.29319 31.670 < 2e-16 ***
## tr          -2.03534   0.47429 -4.291 4.88e-05 ***
## I(tr^2)      0.70733   0.05857 12.076 < 2e-16 *** 
## month2      -1.65191   1.37945 -1.198 0.234600    
## month3      -1.23982   1.37891 -0.899 0.371248    
## month4      -0.48255   1.37849 -0.358 0.727203    
## month5      -5.32761   1.37820 -3.866 0.000223 ***  
## month6      -8.56249   1.37802 -6.214 2.12e-08 *** 
## month7      -9.21095   1.37796 -6.684 2.72e-09 *** 
## month8      -9.26548   1.37802 -6.724 2.29e-09 *** 
## month9      -6.73483   1.37628 -4.887 5.09e-06 *** 
## month10     -4.28651   1.37849 -3.110 0.002586 **  
## month11     -2.43551   1.37891 -1.766 0.081119 .  
## month12     -0.44684   1.37945 -0.324 0.746828    
## ...
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.659 on 81 degrees of freedom
## Multiple R-squared:  0.9344, Adjusted R-squared:  0.9238 
## F-statistic: 88.68 on 13 and 81 DF, p-value: < 2.2e-16
```

Прогноз на два шага вперед

```
fcast5=forecast(tr2_seas_mod, newdata=tr_seas_pr)
```

```
##           Point Forecast    Lo 80     Hi 80    Lo 95     Hi 95
## Jan 2016   69.94251 66.09429 73.79074 64.01665 75.86838
## Feb 2016   69.06901 65.24819 72.88984 63.18534 74.95269
```

Прогноз на два шага вперед

```
plot(fcast5)
lines(tr2_seas_mod$fit, col=3, lwd=2)
```

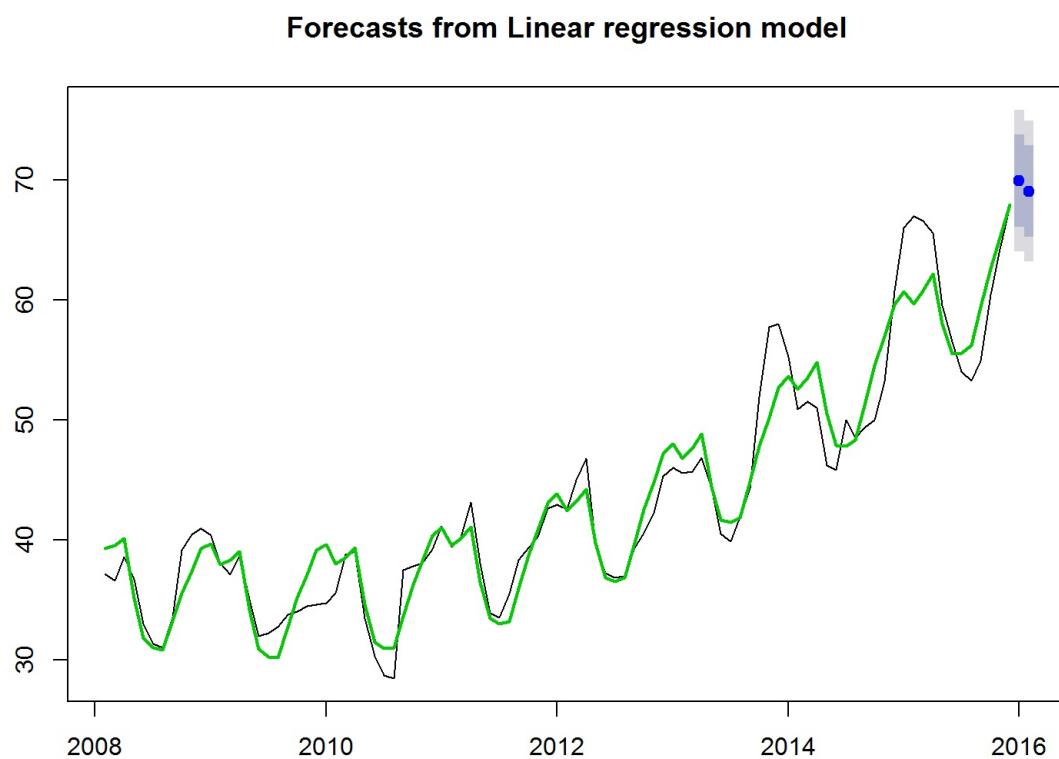


График остатков

Для модели цены яиц с квадратичным трендом

```
plot(tr2_seas_mod$resid)
```

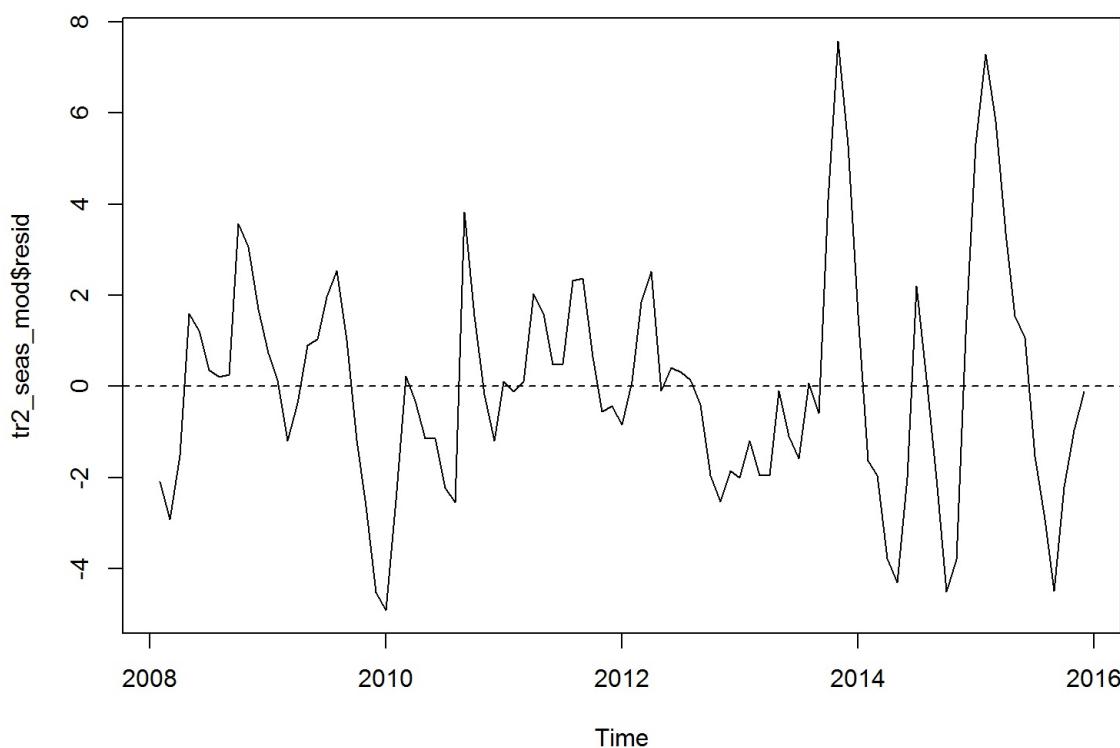
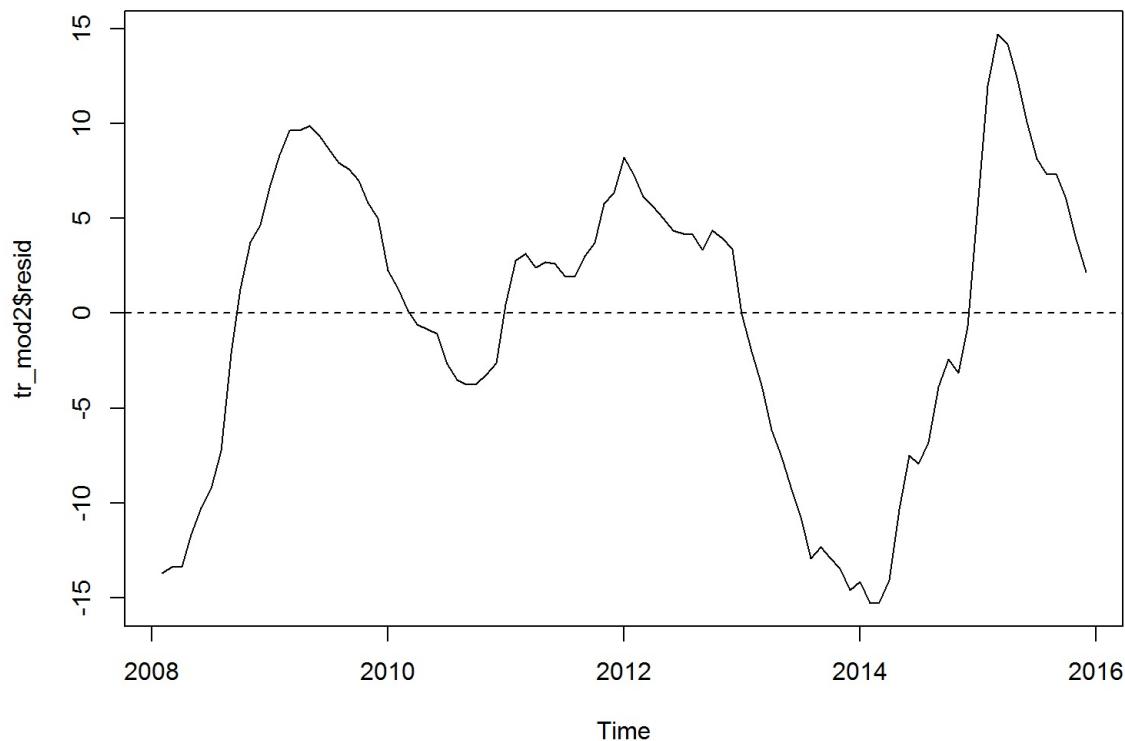


График остатков

Для модели цены фарша с квадратичным трендом

```
plot(tr_mod2$resid)
abline(h=0, lty=2)
```



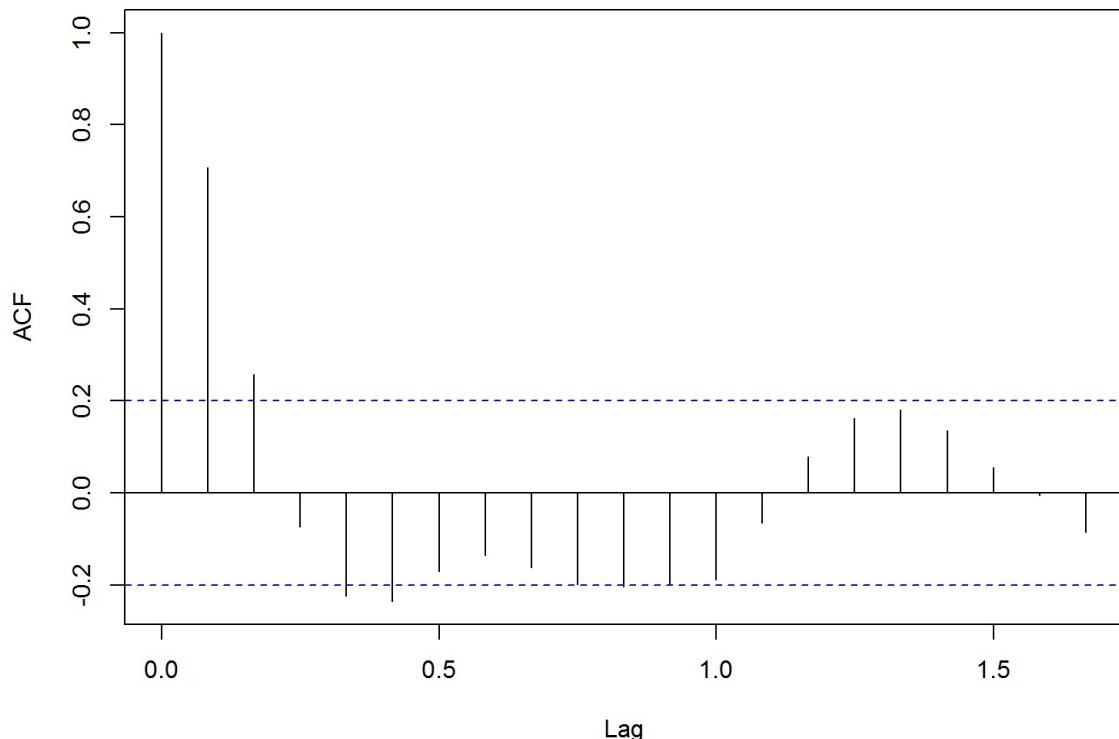
Анализ автокорреляции остатков

Автокорреляционная функция строится с помощью acf()

Показывает корреляцию между элементами временного ряда с заданным лагом

```
acf(tr2_seas_mod$resid, 20)
```

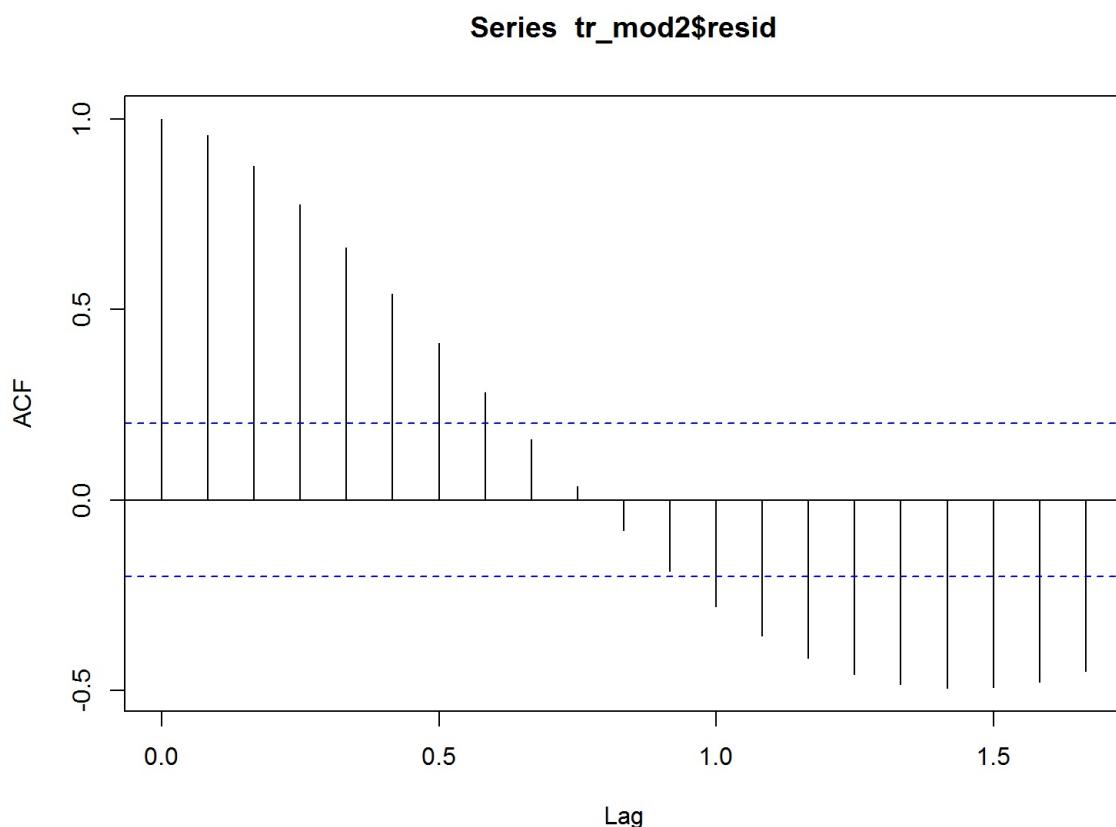
Series tr2_seas_mod\$resid



Анализ автокорреляции остатков

Показывает корреляцию между элементами временного ряда с заданным лагом

```
acf(tr_mod2$resid, 20)
```

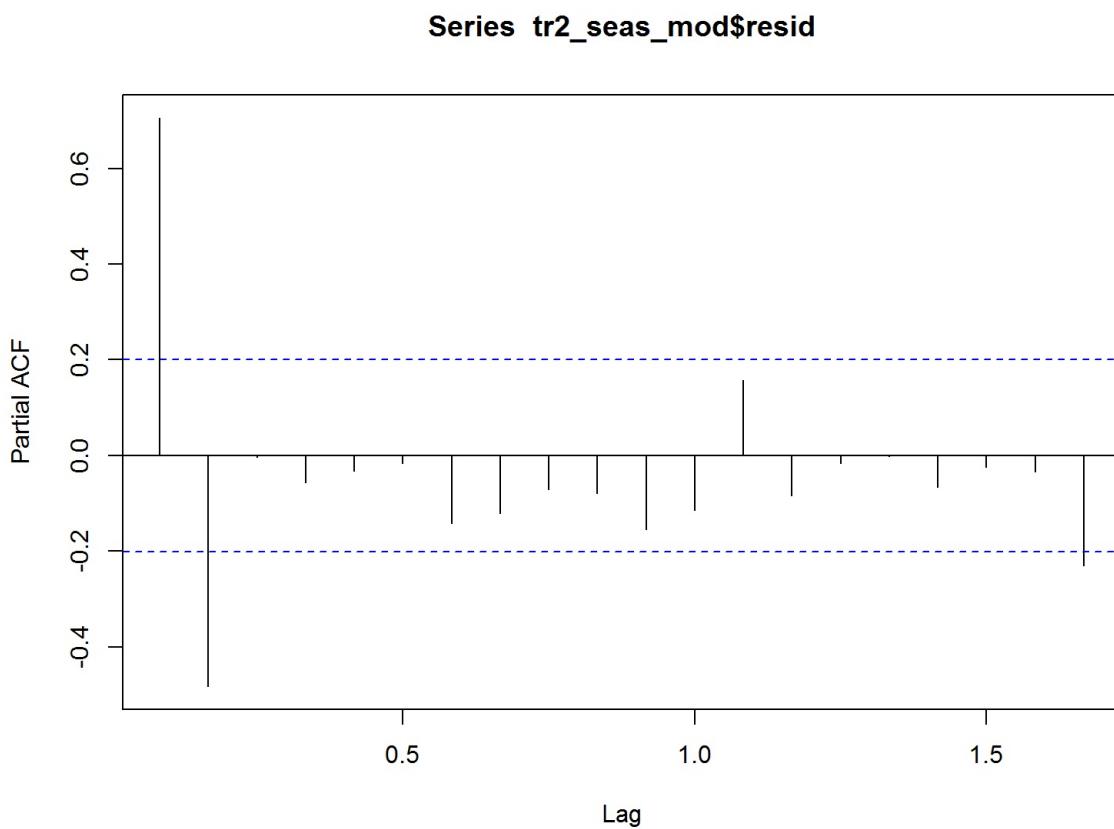


Частная автокорреляционная функция

- Строится с помощью pacf()
- Показывает частную корреляцию между элементами временного ряда с заданным лагом при исключении влияния промежуточных элементов

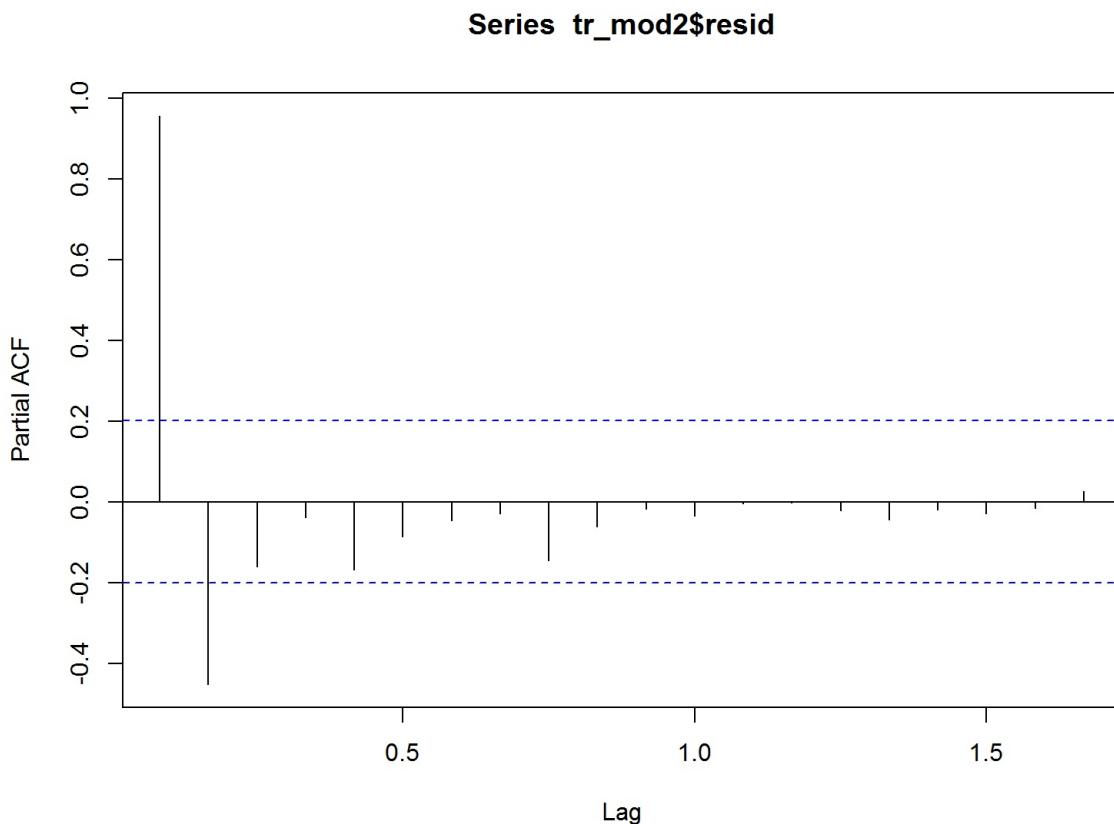
Частная автокорреляционная функция

```
pacf(tr2_seas_mod$resid,20)
```



Частная автокорреляционная функция

```
pacf(tr_mod2$resid, 20)
```



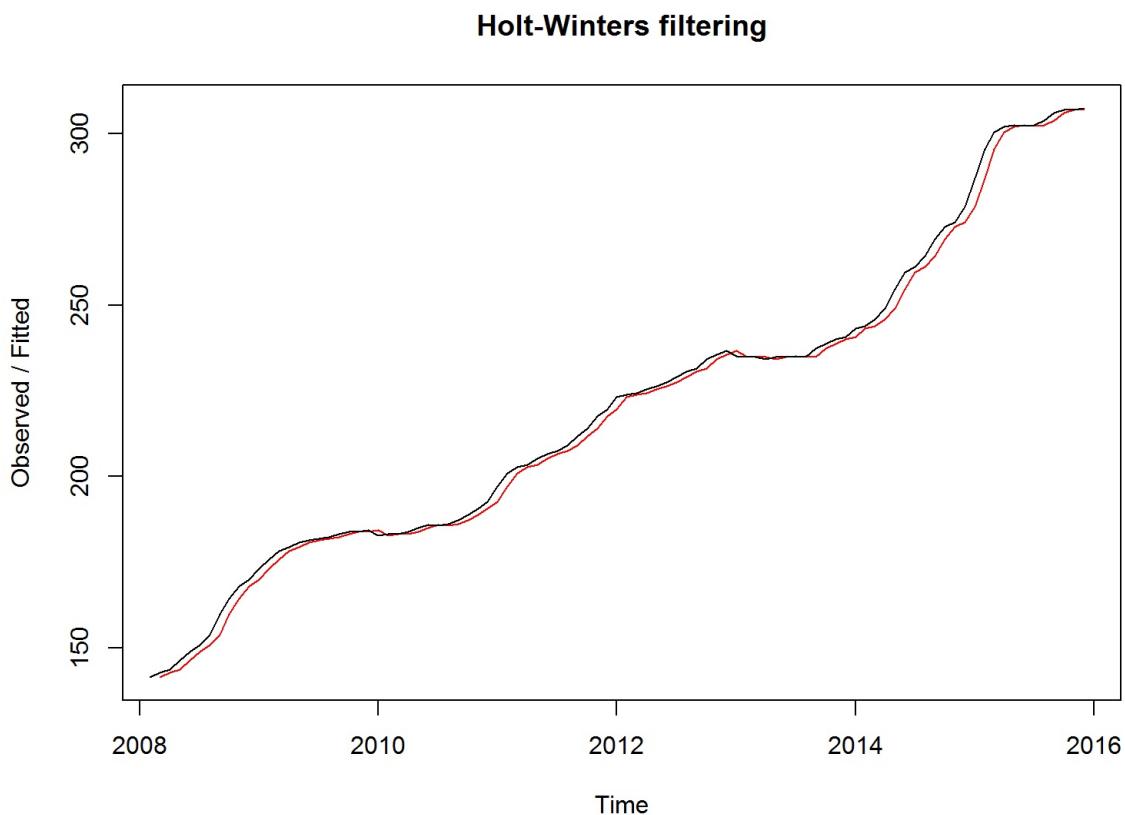
Модель Хольта-Уинтерса

- Включает параметры alpha, beta, gamma
- Для неизвестных параметров, включенных в модель, подбираются оптимальные значения путем минимизации остаточной суммы квадратов

Простое экспоненциальное сглаживание

Включается только параметр alpha

```
fit_EMA=HoltWinters(ts_pp$Фарш.мясной, beta=FALSE, gamma=FALSE)
plot(fit_EMA)
```

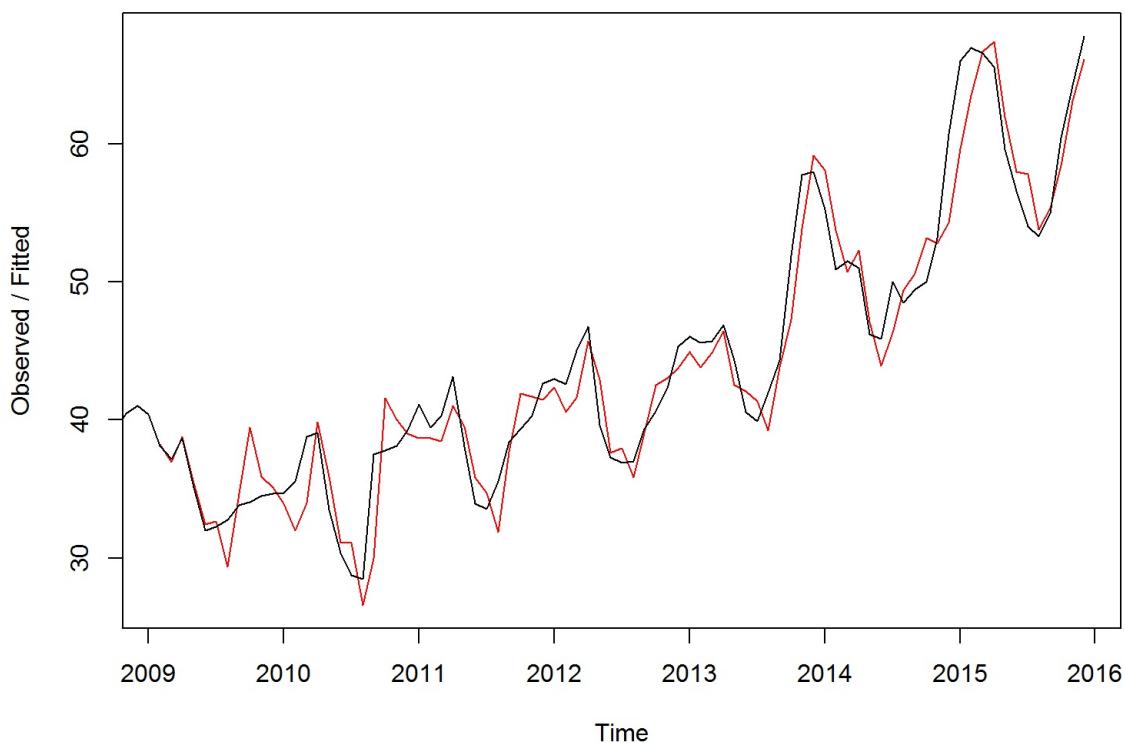


Экспоненциальное сглаживание с трендом и сезонностью

Включаются параметры alpha, beta, gamma

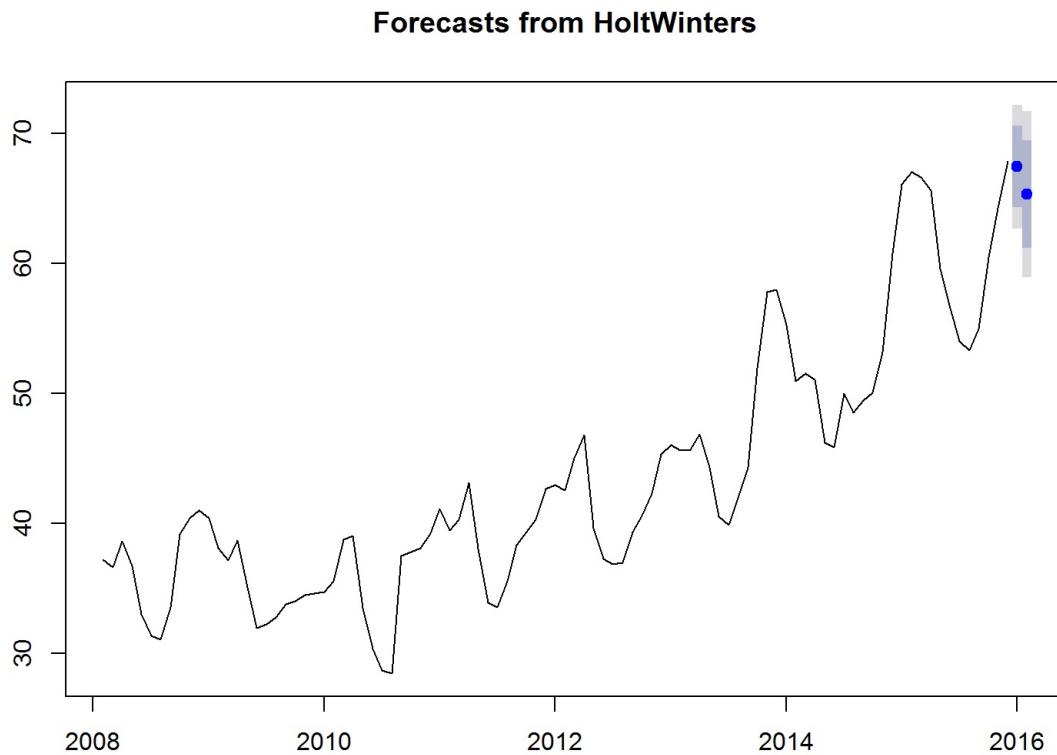
```
fit_HW<-HoltWinters(ts_pp$Яйца)  
plot(fit_HW)
```

Holt-Winters filtering



Прогноз будущего с помощью модели Хольта-Уинтерса

```
plot(forecast(fit_HW, 2))
```



```
forecast(fit_HW, 2)
```

```
##           Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2016      67.43032 64.31896 70.54168 62.67190 72.18874
## Feb 2016      65.30333 61.14100 69.46566 58.93759 71.66907
```

Модель ARIMA(p,d,q)

1. p - порядок авторегрессии
2. d - порядок интегрирования (разности)
3. q - порядок скользящего среднего

```
fit_ARIMA=arima(ts_яйца, order=c(2, 2, 0))  
accuracy(fit_ARIMA)
```

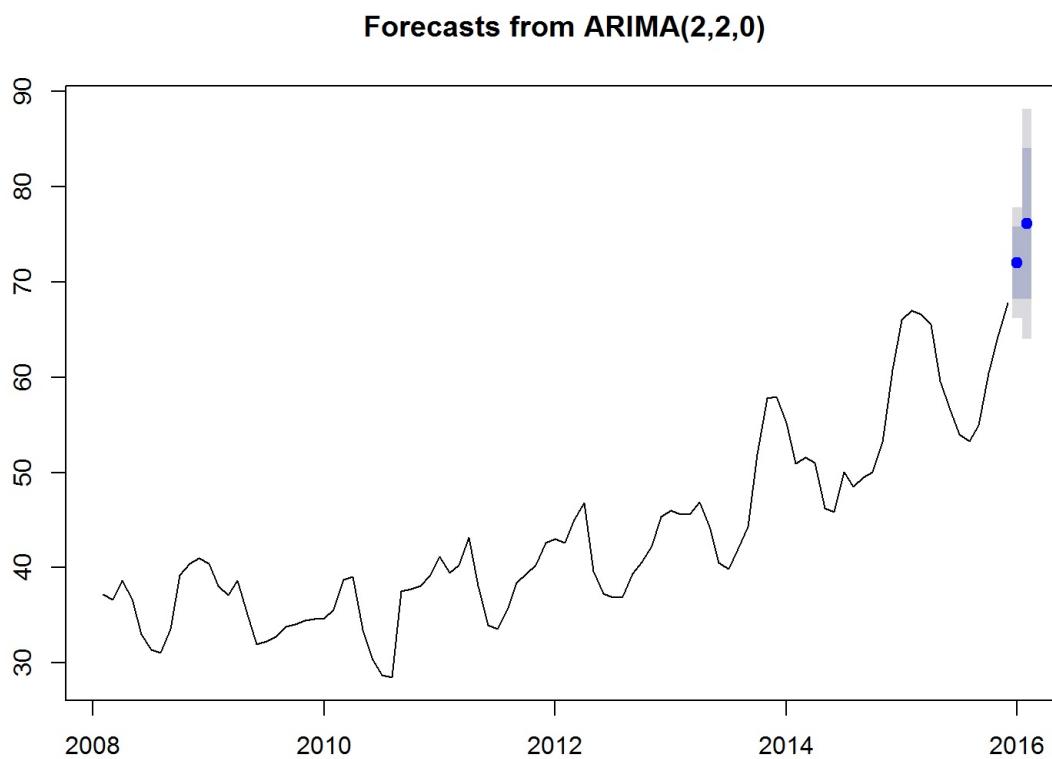
```
##           ME      RMSE      MAE      MPE      MAPE      MASE  
## Training set 0.07142346 2.936986 2.2592 0.2506404 5.282291 1.044229  
##  
##          ACF1  
## Training set -0.006884145
```

Прогноз с помощью модели ARIMA

```
forecast(fit_ARIMA, 2)
```

```
##           Point Forecast    Lo 80     Hi 80     Lo 95     Hi 95
## Jan 2016      72.00116 68.19702 75.80531 66.18322 77.81911
## Feb 2016      76.12983 68.23217 84.02748 64.05140 88.20825
```

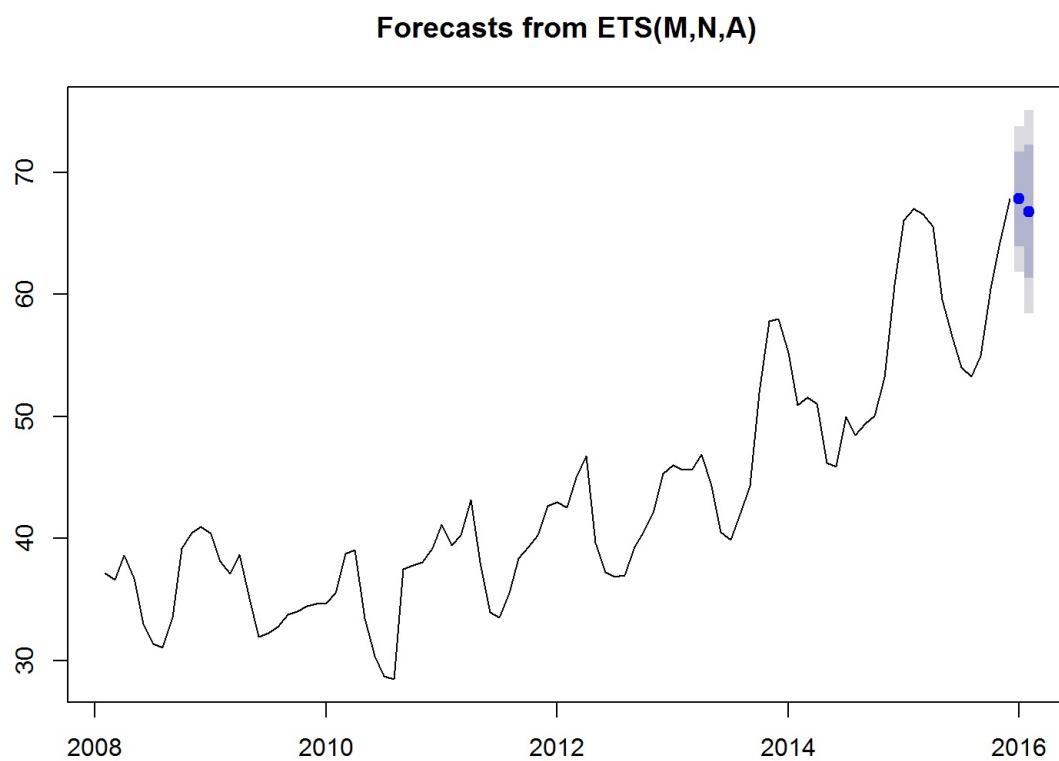
```
plot(forecast(fit_ARIMA, 2))
```



Автоподбор модели временного ряда

С помощью функции `ets{forecast}` подбирает: тип ошибки, тип тренда, тип сезонности

```
fit_auto_ets=ets(ts_pp$Яйца)
plot(forecast(fit_auto_ets,2))
```

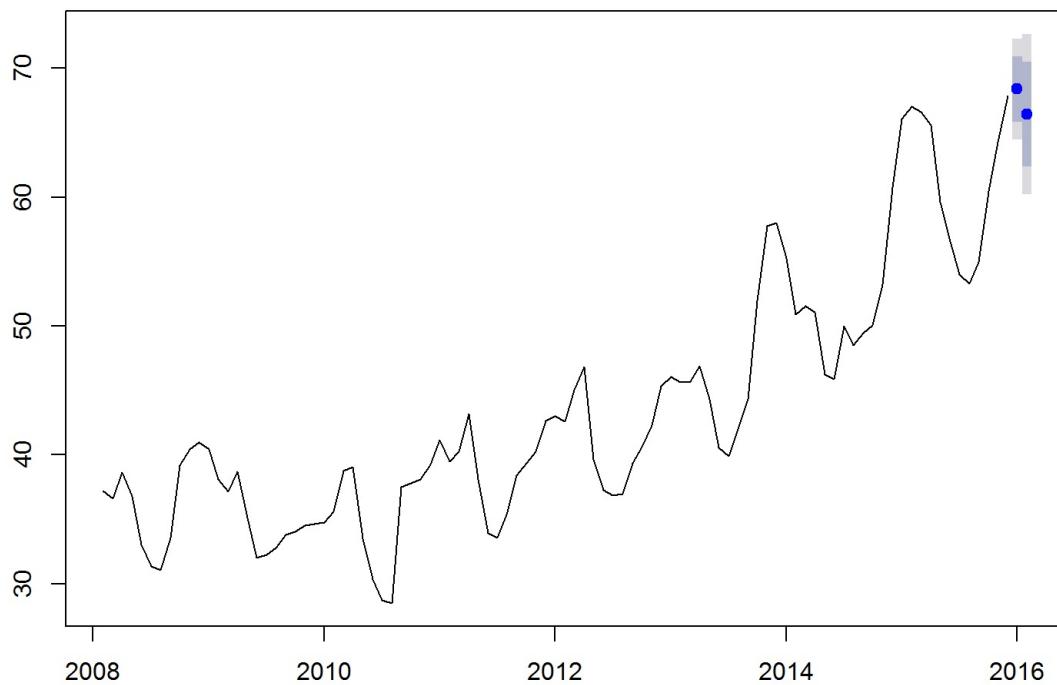


Автоподбор модели ARIMA

С помощью функции auto.arima{forecast} подбирает параметры p, d, q

```
fit_auto_ARIMA=auto_arima(ts_рп$Яйца)
plot(forecast(fit_auto_ARIMA,2))
```

Forecasts from ARIMA(2,1,1)(0,1,1)[12]



Создание отчета с помощью R Markdown

1. Скачать и подключить библиотеки
 - “markdown”
 - “rmarkdown”
 - “knitr”
2. В меню File-> New File-> R Markdown...
3. В появившемся окне слева выбирать Document и в списке - Word
4. Сохранить файл
5. В результате появляется файл с расширением .Rmd
6. Создать отчет
7. Нажать в окне файла сверху Knit Word
8. В результате появляется файл Word

Оформление отчетов в R Markdown

- Знаки с решетками # (##, ###) означают заголовки.
- Чтобы начать новый абзац, нужно сделать два абзацных отступа.
- R-код и результаты выполнения кода включаются в отчет с помощью: {r }

Параметры, регулирующие вывод R-кода и результатов

Аргументы указываются в выражении {`г`} через запятые

1. `include=FALSE` - код только выполняется, в отчет не включаются ни сам код, ни результаты (например, при подключении библиотеки)
2. `echo=TRUE` - включить код в отчет
3. `eval=FALSE` - код не выполняется
4. `eval=FALSE, echo=TRUE` - код не выполняется, но включается в отчет
5. `warning=FALSE` - чтобы не выводились сообщения-предупреждения

Программные системы статистического анализа

Тимофеева А.Ю.

к.э.н., доцент кафедры ТПИ

a.timofeeva@corp.nstu.ru

15 декабря, 2021

Структура лекции

- Работа со строками
 - Обработка
 - Расстояние между строками
 - Регулярные выражения
- Веб-скрапинг
 - Загрузка и обработка HTML-, XML-файлов
 - Поиск элементов разметки
 - Отправка запроса через форму
 - Функции навигации
 - Работа с кодировками

Работа со строками в R

Для работы со строками можно использовать:

- базовые функции R:

`grep()`, `nchar()`, `paste()`, `sprintf()`,
`substr()`, `strsplit()`, `regex()`, `gregexpr()`

- пакет `stringr` (часть tidyverse):

названия функций начинаются на `str_`

- пакет `stringi` (не входит в tidyverse):

названия функций начинаются на `stri_`

Базовые функции для работы со строками

- grep() – поиск по шаблону

```
grep("b", c("abc", "bda", "cca a", "abd"), value=FALSE)
# 1 2 4
```

- nchar() – число символов

```
x = "mdfhhbrgjhglioeseuwkodx"
nchar(x)
# 22
```

- sprintf() – команды форматирования, стилизованные под C

```
sprintf("%s scored %.2f percent", "Student", 72.3)
# "Student scored 72.30 percent"
```

Базовые функции для работы со строками

- `paste()` – объединение строк

```
paste ("x", 1:3, sep = "_")
# "x_1" "x_2" "x_3"
paste (c("x", "y", "z"), 1:3, collapse = ", ")
# "x, y, z"
```

- `substr()` – извлечение подстрок по номеру элементов

```
x = "Student scored 72.30 percent"
substr(x, 1, 7)
# "Student"
```

Базовые функции для работы со строками

- `strsplit()` – разбиение строки на слова

```
x = "Splitting sentence into words"  
strsplit(x, " ")  
[[1]]  
[1] "Splitting" "sentence" "into" "words"
```

- `regexpr()`, `gregexpr()` – регулярные выражения

```
x = "Student scored 72.30 percent"  
gregexpr("\d", x)  
[[1]]  
[1] 16 17 19 20  
attr(, "match.length")
```

Встроенные строки

- letters

```
# "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
```

- LETTERS

```
# "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"
```

- month.name

```
# "January" "February" "March" "April"
```

Случайные строки

- Создание случайных строк

```
library(stringr)
set.seed(2337)
stri_rand_strings(n = 3, length = 2*3:5)
# "9zUdIo"      "s8c2WjPs"      "Z4bbDfYaoQ"
```

- Перемешивает символы внутри строки

```
stri_rand_shuffle("Student scored 72.30 percent")
# "t7 u.dpt srt0eeode2 crncen3S"
```

- Псевдослучайный текст

```
stri_rand_lipsum(nparagraphs = 2)
# [1] "Lorem ipsum dolor sit amet, mauris vel
# [2] "Pellentesque mattis mauris orci nisl vitae
```

Соединение и разделение строк

- Соединение строк

```
| = str_c (LETTERS, rev(letters), sep = "_")  
|  
# "A_z" "B_y" "C_x" "D_w" "E_v" "F_u" "G_t" "H_s"
```

- Разделение строк в виде списка

```
str_split(month.name, "r")  
# [[1]]  
# [1] "Janua" "y"  
#  
# [[2]]  
# [1] "Feb" "ua" "y"
```

Соединение и разделение строк

— Разделение строк

```
library(tidyverse)
tibble(up=LETTERS,down=rev(letters)) %>%
  mutate(all=str_c(up, down, sep = " _")) %>%
  separate(col=all, into=c("col1", "col2"), sep="_")
# A tibble: 26 x 4
  up     down   col1   col2
  <chr> <chr> <chr> <chr>
1 A      z      A      z
2 B      y      B      y
3 C      x      C      x
4 D      w      D      w
5 E      v      E      v
6 F      u      F      u
```

Количество символов

- Подсчет

```
tt = tibble(mn = month.name)
tt %>%
  mutate(n_charactars = str_count(mn))
# A tibble: 12 x 2
  mn      n_charactars
  <chr>     <int>
1 January      7
2 February     8
3 March        5
4 April        5
5 May          3
6 June         4
7 July         4
```

Количество символов

— Подгонка

```
tt %>%
```

```
  mutate(mn_new = str_trunc(mn, 6))  
# A tibble: 12 x 2  
  mn      mn_new  
  <chr>   <chr>  
1 January Jan ...  
2 February Feb ...
```

```
tt %>%
```

```
  mutate(mn_new=str_trunc(mn,6 , side ="center"))  
# A tibble: 12 x 2  
  mn      mn_new  
  <chr>   <chr>  
1 January Ja ... y  
2 February Fe ... y
```

Количество символов

- Подгонка

```
tt %>%  
  mutate(mn_new = str_trunc(mn, 3, ellipsis = ""))  
# A tibble: 12 x 2  
  mn      mn_new  
  <chr>   <chr>  
1 January Jan  
2 February Feb
```

- “Раздуть” строку

```
tt %>% mutate(mn_new = str_pad(mn, 10))  
# A tibble: 12 x 2  
  mn      mn_new  
  <chr>   <chr>  
1 January "January"  
2 February "February"
```

Сортировка

- Базовая функция `sort()`

```
set.seed(38)
(s=sample(letters, 10))
# [1] "k"  "a"  "d"  "i"  "q"  "p"  "z"  "w"  "y"  "o"
sort(s)
# [1] "a"  "d"  "i"  "k"  "o"  "p"  "q"  "w"  "y"  "z"
```

- Функция `str_sort()` пакета `stringr`

```
str_sort(month.name)
# "April"      "August"     "December"
# "February"   "January"    "July"       "June"
# "March"       "May"        "November"   "October"
# "September"
```

Сортировка: сравнение алгоритмов

Алгоритмы сортировки функции sort()

- auto: radix или shell
- shell: сортировка Шелла
- quick: метод быстрой сортировки
- radix: поразрядная сортировка

Метод radix обычно превосходит другие методы, особенно для символьных векторов и небольших целых чисел.

Метод по умолчанию – auto.

Метод auto выбирает radix для коротких (меньше 2^{31} элемента) числовых векторов, целочисленных , логических векторы и факторов; в противном случае выбирается сортировка Шелла.

Но для строк большой размерности она может быть менее эффективна. Сравним с другими алгоритмами сортировки.

Сортировка: сравнение алгоритмов

Сформируем большой вектор из букв.

```
set.seed(2347)
huge = sample(letters, 1e7, replace = TRUE)
```

Замерим время сортировки с помощью пакета `bench`.

```
mm = mark (
  "radix" = sort(huge, method = "radix"),
  "shell" = sort(huge, method = "shell"),
  "quick" = sort(huge, method = "quick")
)
```

mm

	expression	min	median	'itr/sec'	mem_alloc
1	radix	240.51ms	240.8ms	4.12	114MB
2	shell	5.69s	5.69s	0.176	114MB
3	quick	5.68s	5.68s	0.176	114MB

Поиск подстроки

Функция `str_detect()` возвращает TRUE/FALSE при наличии/отсутствии подстроки

```
tibble(mn = month.name) %>%
  mutate(has_r = str_detect(mn, "r"))
# A tibble: 12 x 2
  mn      has_r
  <chr>   <lgl>
1 January  TRUE
2 February TRUE
3 March   TRUE
4 April   TRUE
5 May     FALSE
6 June   FALSE
7 July   FALSE
8 August FALSE
```

Поиск подстроки

Функция `str_which()` возвращает номер элемента, содержащего подстроку

```
tibble(mn = month.name) %>%
  slice(str_which(mn, "r"))
# A tibble: 8 x 1
  mn
  <chr>
1 January
2 February
3 March
4 April
5 September
6 October
7 November
8 December
```

Количество вхождений подстроки

Функция `str_count()` возвращает количество вхождений подстроки

```
tibble(mn = month.name) %>%
  mutate(has_r = str_count(mn, "r"))
# A tibble: 12 x 2
  mn      has_r
  <chr>    <int>
1 January     1
2 February    2
3 March       1
4 April       1
5 May         0
6 June        0
7 July        0
8 August      0
```

Изменение регистра строки

Прописные буквы

```
example = "Student scored 72.30 percent"  
str_to_upper(example)  
# "STUDENT SCORED 72.30 PERCENT"
```

Строчные буквы

```
str_to_lower(example)  
# "student scored 72.30 percent"
```

Каждое слово с прописной буквы

```
str_to_title(example)  
# "Student Scored 72.30 Percent"
```

Выделение подстроки

По индексам функцией str_sub()

```
tibble(mn = month.name) %>%
  mutate(mutate = str_sub(mn, start = 1, end = 2))
# A tibble: 12 x 2
  mn      mutate
  <chr>   <chr>
1 January  Ja
2 February Fe
3 March   Ma
4 April   Ap
5 May     Ma
6 June    Ju
7 July    Ju
8 August  Au
```

Выделение подстроки

По подстроке функцией `str_extract()`
(по умолчанию возвращает первое вхождение)

```
tibble(mn = month.name) %>%
  mutate(mutate = str_extract(mn, "r"))
# A tibble: 12 x 2
  mn      mutate
  <chr>    <chr>
1 January   r
2 February  r
3 March     r
4 April     r
5 May       NA
6 June      NA
7 July      NA
8 August    NA
```

Выделение подстроки

По подстроке функцией `str_extract_all()`
(возвращает список всех вхождений подстрок,
соответствующих шаблону)

```
str_extract_all(month.name, "r")
[[1]]
[1] "r"
[[2]]
[1] "r" "r"
[[3]]
[1] "r"
[[4]]
[1] "r"
[[5]]
character(0)
```

Замена подстроки

Функция `str_replace()` заменяет одну подстроку в строке на другую
(делает замену для первого вхождения)

```
tibble(mn = month.name) %>%  
  mutate(mutate = str_replace(mn, "r", "R"))  
# A tibble: 12 x 2  
  mn      mutate  
  <chr>   <chr>  
1 January JanuaRy  
2 February FebRuuary  
3 March   MaRch  
4 April   ApRil  
5 May     May  
6 June    June  
7 July    July  
8 August  August
```

Замена подстроки

Функция `str_replace_all()` заменяет все вхождения подстроки

```
tibble(mn = month.name) %>%  
  mutate(mutate = str_replace_all(mn, "r", "R"))  
# A tibble: 12 x 2  
  mn      mutate  
  <chr>   <chr>  
1 January JanuaRy  
2 February FebRuaRy  
3 March   MaRch  
4 April   ApRil  
5 May     May  
6 June    June  
7 July    July  
8 August  August
```

Удаление подстроки

Функция `str_replace()` удаляет первое вхождение подстроки

```
tibble(month.name) %>%
  mutate(mutate = str_remove(month.name, "r"))
# A tibble: 12 x 2
  month.name  mutate
  <chr>       <chr>
1 January     Januay
2 February    Febuary
```

Функция `str_replace_all()` удаляет все вхождения подстроки

```
tibble(month.name) %>%
  mutate(mutate = str_remove_all(month.name, "r"))
# A tibble: 12 x 2
  month.name  mutate
  <chr>       <chr>
1 January     Januay
2 February    Febuay
```

Расстояния между строками

Пакет `stringdist` позволяет рассчитать расстояния между строками. Он использует следующие метрики:

- `osa` – ограниченное расстояние Дамерау-Левенштейна
- `lv` – расстояние Левенштейна
- `dl` – полное расстояние Дамерау-Левенштейна
- `hamming` – расстояние Хэмминга (строки должны иметь одинаковое число символов)
- `lcs` – наибольшее общее расстояние между подстроками
- `qgram` – расстояние по q -граммам (подпоследовательность q последовательных символов в строке)
- `cosine` – косинусное расстояние между профилями q -грамм
- `jaccard` – расстояние Жаккара между профилями q -грамм
- `jw` – расстояние Джаро, Джаро–Винклера

Расстояния между строками: примеры

По умолчанию функция `stringdist` использует ограниченное расстояние Дамерау-Левенштейна. Подсчитывается количество операций (удалений, вставок, замен), которые нужно сделать, чтобы перевести одну строку в другую.

```
library(stringdist)
stringdist("Student", c("Student", "Strudel",
                       "Stupen", "Studen ''", "Prudent"))
# 0 3 2 1 2

stringdist("Student", c("Student", "Strudel",
                       "Stupen", "Studen ''", "Prudent"),
           method = "jaccard", q = 2)
# 0.0000000 0.6666667 0.6250000 0.2857143 0.5000000
```

Регулярные выражения

Регулярное выражение (regex) – шаблон (образец), описывающий некоторое множество строк общей структуры.

Метасимволы в регулярном выражении имеют специальное значение и сами себе не соответствуют.

, *, ?, +, ^, \$, {}, [], |, (), \

Для обычных символов, совпадающих с метасимволами, используется экранирующий символ (**двойной бэкслэш**).

Квантификаторы ограничивают число вхождений образца в строку:

* – хотя бы 0 раз;

+ – хотя бы 1 раз;

? – не более 1 раза;

{*n*} – ровно *n* раз;

{*n*,*m*} – *n* и более раз;

{*n*,*m*} – от *n* до *m* раз (*n* < *m*).

Жадность квантификаторов

Для строки

```
s = "<em>Hello , world !</em>"
```

регулярное выражение вернет:

```
regexpr( "<.+>" , s )
[1] 1
attr( , "match.length" )
[1] 22
```

Всю строку из 22 символов (жадный квантификатор).

Если нужен только первый , то нужно использовать ленивый квантификатор.

```
regexpr( "<.+?>" , s )
[1] 1
attr( , "match.length" )
[1] 4
```

Жадность квантификаторов

Если нужны все вхождения шаблона, то лучше использовать функцию `gregexpr()` вместе с ленивым квантификатором.

```
gregexpr( "<.+?>" , s )
[[1]]
[1] 1 18
attr( , "match.length" )
[1] 4 5
```

Ленивое поведение определяется знаком ?, располагающимся после квантификатора:

? – ленивый аналог *;

+? – ленивый аналог +;

{n,}? – ленивый аналог {n,}.

Положение шаблона внутри строки, операторы

Группа регулярных выражений, описывающих положение шаблона внутри строки:

- ^ – шаблон, следующий за этим символом, расположен в начале строки;
- \$ – в конце строки;
- \b – в начале слова или строки;
- \B – шаблон не находится в начале слова или строки;

Операторы:

- . – любой единичный символ;
- [...] – один символ из списка;
- [^...] – любой символ, кроме тех, что содержатся в списке;
- (...) – группировка.

Классы символов

Два способа задать классы символов:

- `[[:...:]]` – стандарт POSIX;
- `\\\dots` – Unix.

Примеры:

- цифра от 0 до 9: `[[:digit:]]`, `\d`;
- строчная буква: `[[:lower:]]`, `[a-z]`;
- прописная буква: `[[:upper:]]`, `[A-Z]`;
- любая буква: `[[:alpha:]]`, `[A-Za-z]`;
- цифра или буква: `[[:alnum:]]`, `[0-9A-Za-z]`;
- пробел, табуляция, вертикальная табуляция, новая строка, возврат каретки: `[[:space:]]`, `\s`;
- символ пунктуации: `[[:punct:]]`.

Веб-скрапинг: пакеты

Для выполнения HTTP-запроса по заданному URL и получения веб-страницы подходят пакеты:

- RCurl;
- httr.

Для обработки HTML-разметки можно использовать пакеты:

- XML;
- xml2.

Пакеты RCurl и XML разработаны Д.Т. Лэнгом, httr и xml2 – Х. Уикэном. Он разработал также пакет **rvest**, который позволяет получать данные с помощью httr и обрабатывать их в xml2.

Получение и обработка HTML документа

Функция `read_html` пакета `rvest` возвращает дерево HTML-элементов. Увидеть структуру документа можно с помощью функции `html_structure`.

```
library(rvest)
hdoc = read_html ("sample.html")
html_structure(hdoc)
```

Функции `html_nodes`, `html_node` принимают на вход документ, построенный функцией `read_html`, и путь к нужному элементу, указанный с помощью CSS-селектора (`css`) или XPath (`xpath`). `html_nodes` возвращает список элементов, расположенных по заданному пути, `html_node` – только первый элемент.

```
html_nodes(hdoc, "#date")
html_nodes(hdoc, xpath = "//span[@id='date']")
```

Разбор элемента

Обрабатаем простой документ:

```
simple_doc = read_html ('<a href="http://google.com"  
rel="nofollow">  
Link to the <b>Google</b></a>' )
```

Извлечем элемент а:

```
link = html_nodes (simple_doc, xpath = "//a")
```

Разберем его с помощью функций rvest.

Функция	Возвращает
html_name	имя элемента
html_attrs	атрибуты элемента
html_attr	отдельный атрибут по его имени
html_text	текст внутри элемента
html_children	список дочерних элементов

Разбор элемента

Имя элемента

```
html_name (link)
# [1] "a"
```

Его атрибуты

```
html_attrs (link)
# [[1]]
#           href          rel
# "http://google.com"  "nofollow"
```

Обращение к отдельному атрибуту по его имени

```
html_attr (link ,  "href")
# [1] "http://google.com"
```

Работа с формами

Работа с формами в пакете `rvest` состоит из трех этапов:

- ① найти форму и обработать ее, то есть определить, какие параметры она передает и каковы текущие значения этих параметров;
- ② установить нужные значения параметров формы;
- ③ отправить форму, то есть выполнить HTTP-запрос с заданными параметрами.

Функция `html_form` отвечает за поиск и обработку форм: принимает на вход документ XML или элемент, полученный в результате работы `read_html`.

Работа с формами: пример

```
form = '<form name="myform" action="textinput"  
        method="GET">  
  
    Input text: <br>  
    <input type="text" name="inputbox" value=""><br>  
    <input type="button" name="btn1" value="Read">  
    <input type="button" name="btn2" value="Write">  
    </form>'  
  
hdoc = read_html(form)  
myform = html_form(hdoc)  
myform  
# [[1]]  
# <form> 'myform' (GET textinput)  
# <input text> 'inputbox':  
# <input button> 'btn1': Read  
# <input button> 'btn2': Write
```

Работа с формами: заполнение и отправка

Значения полей формы задаются функцией `set_values`.

Она принимает на вход имя формы, обработанной `html_form`, и пары параметр=значение.

```
filled_form = set_values (myform [[1]],
                           inputbox = "Hello")
```

`filled_form`

```
# <form> 'myform' (GET textinput)
# <input text> 'inputbox': Hello
# <input button> 'btn1': Read
# <input button> 'btn2': Write
```

Функция `set_values` возвращает заполненную форму (`filled_form`), которая отправляется на сервер функцией `submit_form`.

```
submit_form (session, filled_form)
```

Работа с формами: сессии

В пакете `rvest` существуют функция `html_session`, принимающая на вход URL и возвращающая объект класса `session`, то есть сессию. Этот объект используется в функциях, реализующих HTTP-запросы, в частности, в `submit_form`.

Он указывает серверу, что речь о конкретном сеансе работы со скриптом-клиентом.

Аутентификация на форуме

```
session = html_session ("http://login.rutracker.org/
                           forum/login.php")
login_form = html_form (session)[[2]]
filled_form=set_values(login_form,
                       "login_username" = "mylogin",
                       "login_password" = "mypassword")
submit_form (session, filled_form)
```

Функции навигации

Функции, позволяющие перейти на новую веб-страницу:

```
jump_to (session , url)  
follow_link (session , i)
```

Функция `jump_to` переходит по заданному URL.

Функция `jump_to` – по условию `i`, которое может быть:

- целым числом: переход по i -й ссылке, найденной на странице;
- строкой: переход по первой ссылке, содержащей эту строку (с учетом регистра).

Функция `back` возвращается на шаг назад в списке помещенных страниц, а история посещений за сеанс возвращается функцией `session_history`.

Работа с кодировками

В пакете `rvest` существуют две функции:

- `guess_encoding` – пытается определить, в какой кодировке записан текст;
- `repair_encoding` – исправляет кодировку.

```
url = 'https://en.wikipedia.org/wiki/Character_encoding'
example = read_html(url)
text = example %>%
  html_nodes(xpath='//*[@id="mw-content-text"]') %>%
  html_text()
text %>% guess_encoding()
#      encoding language confidence
#1      UTF-8          en        1.00
#2 windows-1252       en        0.69
#3 windows-1250       ro        0.22
#4 windows-1254       tr        0.15
#5      UTF-16BE        -         0.10
```