МИНИСТЕРСТВО ОБРАЗОВАНИЯ, НАУКИ И МОЛОДЕЖИ РЕСПУБЛИКИ КРЫМ

РЕСПУБЛИКАНСКОЕ ВЫСШЕЕ УЧЕБНОЕ ЗАВЕДЕНИЕ «КРЫМСКИЙ ИНЖЕНЕРНО-ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Прикладной информатики

«УТВЕРЖДАЮ»
Заведующий кафедрой
_____(Сейдаметова З.С.) «___»_____2024года

Методические указания и задания для выполнения лабораторных работ

по дисциплине

Б1.В.ДВ.06.01 «Обработка и анализ больших данных (Big Data)»

Направление подготовки
09.03.03 Прикладная информатика

Факультет экономики, менеджмента и информационных технологий

Симферополь 2024

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Цель лабораторного практикума освоить основные методы и приобрести навыки анализа данных в активно развивающейся среде программирования - язык R.

R—свободная программная среда вычислений **с открытым исходным кодом**. Это реализация языка S с дополнительными моделями, разработанными в языке S-Plus. R доступен в соответствии с лицензией GNU [ ].

Среда программирования развиваться добавлением пакетов. R-пакет сообществом разработчиков и представляет собой коллекцию наборов данных, функций языка R, документации и динамически загружаемых элементов, написанных на языке C или Fortran. C помощью этих пакетов аналитики обмениваться вычислительными методами, методами визуализации данных, результатами исследований друг с другом.

R **обладает обширной и непрерывно расширяющейся библиотекой пакетов**, которая содержит большое количество методов и примеров решений самых различных задач.

Некоторые пакеты представляют инструменты для обработки и анализа данных различных областей и направлений прикладных и академических исследований, другие являются новейшими разработками и имеют ограниченную область применения. R-пакеты предоставляют широкие возможности для разработки новых методов анализа в области статистики, эконометрики, биологии, химии, медицины, географии и других прикладных областей. Они позволяют предварительно и детально отработать методики и инструментарий, и затем реализовать их в коммерческих или промышленных программных продуктах.

R — это скриптовый язык. Скрипт или программный сценарий, любая исполняемая процедура, которая запускается автоматически или с помощью команды пользователя.

Скрипты используются не только в программировании, область их применения шире. Скрипт может быть использован для повторяемых и сложных операций.

Для обработки неупорядоченных данных требуются специальные языки программирования. Такие программные продукты как SAS и SPSS имеют специальные скриптовые языки для решения отдельных задач. R создан изначально как язык

программирования обработки и анализа статистических данных, и поэтому является более подходящим средством для этой цели.

Язык R предоставляет удобный набор инструментов в области анализа больших массивов данных. Он **интегрирован в ряд коммерческих пакетов**, таких как IBM SPSS и InfoSphere, а также Mathematica.

R интегрирован в системы публикации документов. Это позволяет встраивать статистические результаты и графику из среды R в тексты единым для публикаций способом.

Язык R имеет интуитивно понятный синтаксис как универсальный инструмент, разработанный специально для работы с данными. R также включает в себя графические возможности визуализации данных.

Язык R может работать с различным операционными системами. В данном практикуме представлена среда R в операционной системе Windows.

Для выполнения лабораторных работ необходимо установить язык R. Возможны варианты классической установки представленные в https://cran.rstudio.com/bin/windows/base/. Кроме того, в варианте PRO https://mran.revolutionanalytics.com/download/. Можно выбрать графический интерфейс RStudio: https://www.rstudio.com/products/rstudio/download/ — набор интегрированных инструментов, разработанных, чтобы наиболее продуктивно использовать R.

RStudio включает в себя консоль, редактор с подсветкой синтаксиса, который поддерживает как прямое исполнение кода, так и инструменты для построения графиков, сохранение истории команд, отладку и управление рабочим пространством.

При установке R, чтобы избежать возможных конфликтов, рекомендуется имя пользователя писать латиницей.

Во время установки R и RStudio рекомендуется отключить антивирусную программу.

При сохранении с расширение файлов *.R, рекомендуется не использовать названия файлов или папок, содержащих кириллицу или пробелы.

Для работы в R среде необходимо стабильное Интернет-соединение, т.к. может потребоваться онлайн-загрузка дополнительных пакетов для обработки данных.

В случае возникновении вопросов, на которые нет ответов в стандартной справке R, можно обратится к следующим ресурсам:

- http://rseek.org/ — Google, модифицированный под поиск по источникам, связанным с R;
- http://stats.stackexchange.com/ — статистические методы и их реализации в R;
- http://stackoverflow.com/ — программирование в R;
- http://vk.com/rstatistics — группа в «ВКонтакте»
- http://r-analytics.blogspot.ru/ — масштабный проект на русском языке

ЛАБОРАТОРНАЯ РАБОТА №1.

Тема: Основы работы с R. Некоторые методы статистического анализа данных.

Цель работы

- Ознакомиться с интерфейсом редактора R или RSudio,
- Приобрести навыки работы в режиме консоли и написания скриптов
- Приобрести навыки поиска и подключения внешних пакетов.
- Ознакомиться с основными методами статистической обработки данных.

Задание

1. Загрузить данные для указанного варианта в переменную вектор.
2. Получить справочную информацию по указанным данным, ознакомиться с их содержимым.
3. Проверить, есть ли среди данных пропуски.
4. Создать новую переменную-вектор, в которой будут 1, если значение в исходном векторе больше среднего, и –1, если значение переменной меньше среднего, и 0, если значение равно среднему.
5. Вывести описательную статистику.
6. Построить графики абсолютных частот и плотности распределения.

Указания к выполнению работы

Консольный режим

После запуска R включается консольный режим работы (Рисунок 1). Любая команда, написанная пользователем, будет сразу выполнена R по нажатию Enter.

Рассмотрим основные выражения в R: числа, строки и логические переменные.

Например, консоль среды R можно использовать как калькулятор:

```
> 1 + 1
[1] 2
> 6 * 7
[1] 42
```

```
sqrt(16)
[1] 4
```

Результат выполнения сразу появляется в консоли. Строки печатаются в кавычках: двойных или одинарных:

```
>"Hello world!"
[1] "Hello world!"
'Hello world!'
[1] 'Hello world!'
>3 <4
```

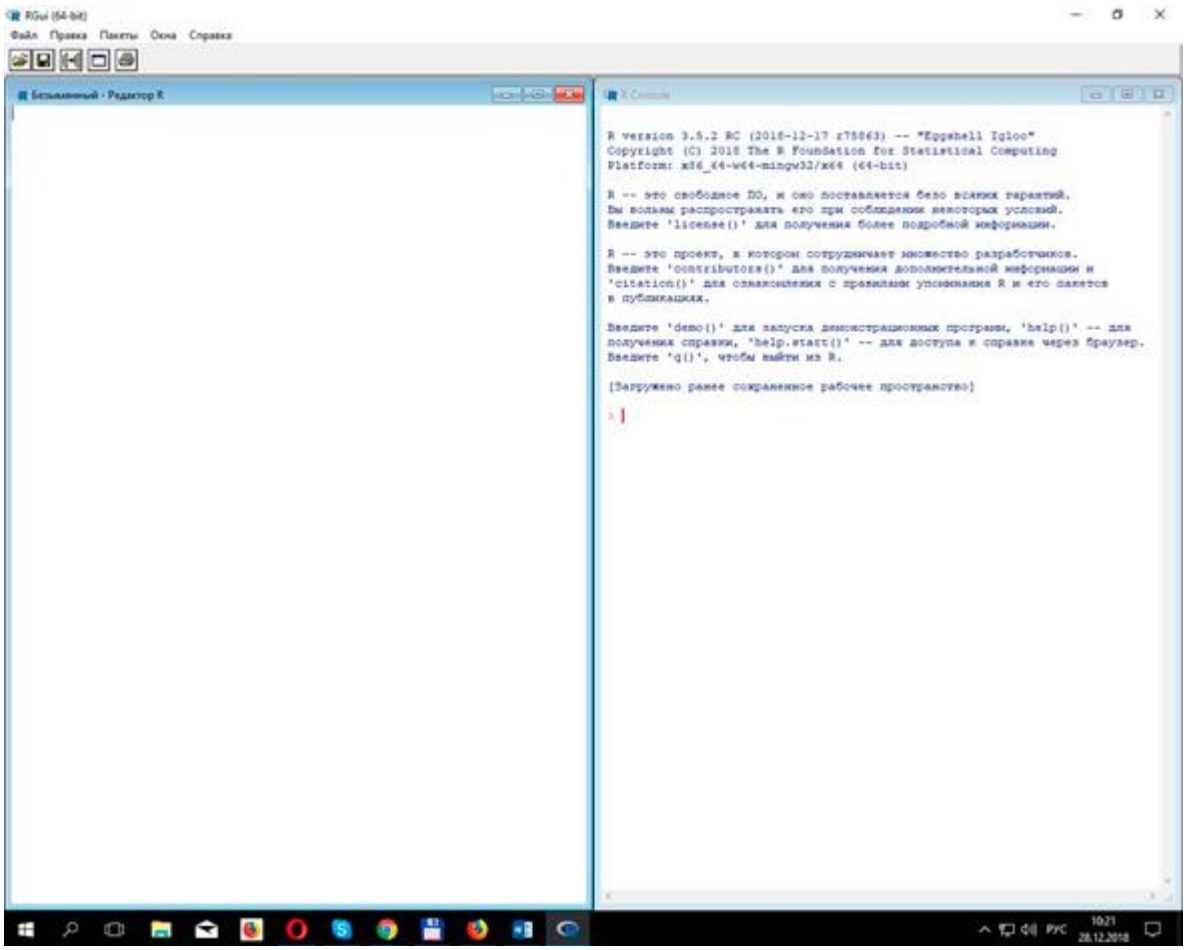


Рисунок 1. Консольный режим в R

Логические выражения возвращают TRUE или FALSE:

```
4
[1] TRUE
```

Чтобы сравнить два выражения, используется двойной знак равенства:

```
>2 + 2 == 5
[1] FALSE
```

Значения можно сохранять в переменную. Сохраним 39 в переменную x:

```
>x <- 39
```

И в обратную сторону:

```
> 5 -> X
```

Можно распечатать значение переменной в любое время, просто набрав ее имя в консоли. Напечатаем текущее значение x:

```
>x
[1] 42
```

Можно повторно назначить любое значение переменной в любое время.

R чувствителен к регистру: переменные x и X— это разные:

```
>X
X
[1] 5
```

Чтобы вызвать функцию, нужно обратиться к ней по имени, указав в скобках нужные аргументы. Например, функция суммы:

```
> sum(1, 3, 5)
[1] 9
```

Получить помощь о функции можно командой help(functionname) или ?functionname.

Вектор можно задать с помощью функции *c* ()(англ. - **C**ombine):

```
>y <- c(-3, 2, NA, 5)
>y
[1] -3  2  NA  5
```

NA — это пропущенное наблюдение (англ. Not Available). Его не следует путать с NaN (Not a Number — «не число», неопределенность):

```
> 0/0
[1] NaN
```

Функция sum дает сумму элементов вектора y:

```
>sum(y)
[1] NA
```

Необязательным аргументом функции sum является na.rm (англ. Remove NA), равный по умолчанию FALSE.

Если указать значение «истина», то функция суммы будет складывать все элементы вектора, исключая пропущенные:

```
>sum(y, na.rm = TRUE)
[1] 4
```

Последовательность чисел можно задать двумя способами: start:end либо функцией seq():

```
>5:9
[1] 5 6 7 8 9
>seq(5,9) [1] 5 6 7 8 9
>seq(10,50, by = 10) [1] 10 20 30 40 50
```

Обращаться к элементам вектора можно, используя квадратные скобки:

```
>     sentence <- c('mack', 'the', 'knife')
>     sentence[3]
[1] "knife"
>     sentence[c(1,3)]
[1]   "mack"     "knife"
```

Либо можно задать элементам вектора имена:

```
>          ranks <- 1:3
>          names(ranks) <- c("first", "second", "third")
>          ranks
first second          third 1 2               3
>          ranks["first"] first
1
```

Скрипты

Скрипты в R позволяют писать не по одной команде в консоли, а сразу целый набор команд и затем запускать либо построчно, либо сразу весь блок кода на исполнение.

Чтобы создать скрипт, следует выбрать Фай л –> Новый скрипт. Откроется новая область, в которой можно писать команды. Комментарии, которые не выполняет R обозначаются знаком #.

Нажатие Enter не приводит к исполнению команды в скрипте. Происходит переход на новую строку.

Чтобы выполнить команду в режиме скрипта, следует поставить курсор на нужную строку и нажать Ctrl+R. Если команда занимает более одной строки, то необходимо ставить знак + в конце каждой строки. Если код выполняется построчно, то в каждой строке требуется нажимать Ctrl+R. Для исполнения блока команд, его нужно выделить нажать Ctrl+R.

В R обычно работают с наборами данных. Такой набор данных называется data.frame и представляет собой таблицу, в которой каждый столбец — это некоторая переменная, а каждая строка — это одно наблюдение.

Создадим в режиме скрипта data.frame. Пусть имеются наблюдения за ростом и весом некоторых людей. Зададим два вектора:

```
rost <- c(160, 175, 155, 190, NA)
ves <- c(NA, 70, 48, 85, 60)
```

И объединим их в набор данных, который поместим в переменную df, а затем выведем на экран:

```
df <- data.frame(rost, ves) df
```

В консоли получим следующую таблицу:

```
> df
   rost  ves
1  160   NA
2  175   70
3  155   48
4  190   85
5   NA   60
```

Обращаться к конкретным наблюдениям df можно, используя квадратные скобки:

```
df[3,1]
[1] 155
```

Обращаться к переменным можно, используя знак $ или указывая столбец с пропуском номера строки:

```
df$rost
[1] 160 175 155 190  NA
```

или

```
df[ ,1]
[1] 160 175 155 190  NA
```

Обращаться к наблюдениям можно, указывая конкретную строку и пропуская номер столбца:

```
df[ ,1]
[1] 160 175 155 190  NA
```

Основные описательные статистики (среднее, стандартное отклонение и медиану) можно получить с помощью функций mean, sd и median:

```
mean(df$rost, na.rm = T)
[1] 170
sd(df$rost, na.rm = T)
[1] 15.81139
median(df$rost, na.rm = T)
[1] 167.5
```

Сохранить скрипт можно, нажав Файл –> Сохранить.

Установка пакетов

В R существует базовый набор пакетов (библиотек) содержащих базовые наборы функций. Этот пакет “base” загружается вместе с установкой R.

Для работы с другими пакетами необходимо их установить и подключить в сеансе работы. Установку достаточно выполнить один раз, а подключать в каждом сеансе работы.

Установка пакета осуществляется с помощью функции install.packages(). Связанные с ним пакеты автоматически устанавливаются.

Установленный пакет подключается функцией library ().

Например, следующие пакеты содержат:

psych — функции для расчета описательных статитик;

dplyr — функции для работы с data.frame;

ggplot2 — функции для построения графиков, диаграмм, карт и т. д.

Инсталляцию нескольких пакетов можно провести с помощью функции install.packages(c("psych","dplyr","ggplot2"))

Прямым сообщением об ошибке установки является только слово Error, появляющееся в консоли. Все остальные сообщения Warning являются просто предупреждениями о чем-либо.

Для выполнения данной лабораторной работы необходимо инсталлировать и подключить следующие пакеты:

library("psych") - описательные статистики

library("lmtest") - тестирование гипотез в линейных моделях

library("ggplot2") - графики

library("dplyr") - манипуляции с данными

library("MASS") - подгонка распределений

В базовом пакете datasets представлено несколько таблиц данных, которые широко используются в примерах работы основных методов анализа инструментами R. Наиболее популярны такие таблицы как iris и cars. Таблица iris содержит исходные данные по анализу цветов ирисов, которые использовались для иллюстрации решения задачи классификации Р. Фишером разработанного им метода дискриминантного анализа. В таблице cars представлен набора данных по автомобилям США.

Если просто набрать cars, то в окне консоли будет напечатана вся таблица. Команда help(cars) переводит к полной документации о данной таблице.

В этой таблице данных представлено 50 наблюдений и две переменных (скорость, миль/час и длина тормозного пути в футах).

```
head(cars, 10)
   speed dist
1      4    2
2      4   10
3      7    4
4      7   22
5      8   16
6      9   10
7     10   18
8     10   26
9     10   34
10    11   17
```

Поместим в переменную d встроенный в R набор данных по автомобилям: d <- cars. Тогда d приобретает тип данных data.frame .

Следующей командой можно посмотреть на этот набор данных, в результате чего будут перечислены все переменные и типы данных:

glimpse(d) - функция из пакета dplyr

Результат выполнения команды появится в консоли:

```
d <- cars
glimpse(d)
Observations: 50
Variables: 2
$ speed <dbl> 4, 4, 7, 7, 8, 9, 10, 10, 10, 11, 11, 12, 12, 12, 12, 13, 13,...
$ dist  <dbl> 2, 10, 4, 22, 16, 10, 18, 26, 34, 17, 28, 14, 20, 24, 28, 26,...
```

Переменные speed и dist имеют тип данных dbl (double) и содержат по 50 наблюдений. Для других типов данных используются следующие сокращения: chr (character/string), int (integer), fctr (factor), tims (time), lgl (logical).

Выше были представлены первые 10 наблюдений набора данных. Следующая команда позволяет представить последние шесть наблюдений:

```
> tail(d)
   speed dist
45    23   54
46    24   70
47    24   92
48    24   93
49    24  120
50    25   85
```

С помощью функции describe(d) из пакета psych можно получить таблицу с описательными статистиками: среднее, мода, медиана, стандартное отклонение, минимум/максимум, асимметрия, эксцесс и т. д.:

```
describe(d) - функция из пакета psych
describe(d)
      vars  n  mean    sd median trimmed   mad min max range  skew kurtosis
speed    1 50 15.40  5.29     15   15.47  5.93   4  25    21 -0.11    -0.67
dist     2 50 42.98 25.77     36   40.88 23.72   2 120   118  0.76     0.12
         se
speed 0.75
dist  3.64
```

Набор функции загруженных пакетов позволяет построить гистограмму абсолютных частот для переменной dist (длины тормозного пути).

```
qplot(data=d, dist, xlab="Длина тормозного пути (футы)",
ylab="Число автомобилей ",main="Данные по автомобилям
1920x")
```

В функции qplot, задается источник данных d (аргумент data), переменная для построения графика (dist), подписи оси (параметры функции xlab и ylab) и название графика (параметр main). Результат выполнения функции представлен на Рисунке 2.

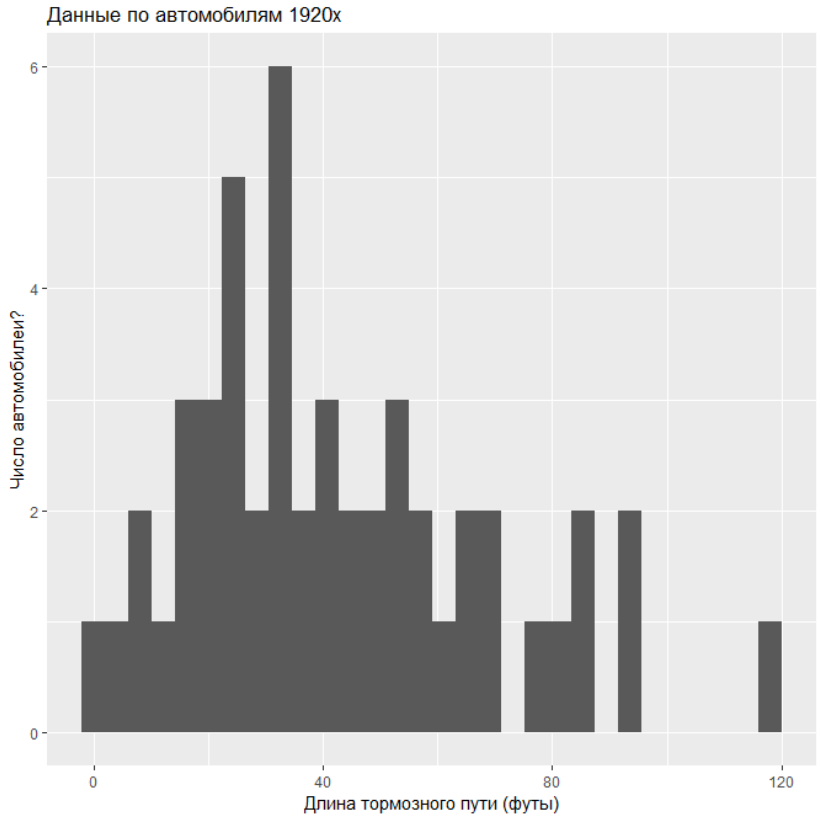


Рисунок 2. Гистограмма абсолютных частот для переменной d

Аналогично построим гистограмму плотности распределения - hist(d$dist, probability = TRUE, col="grey"):

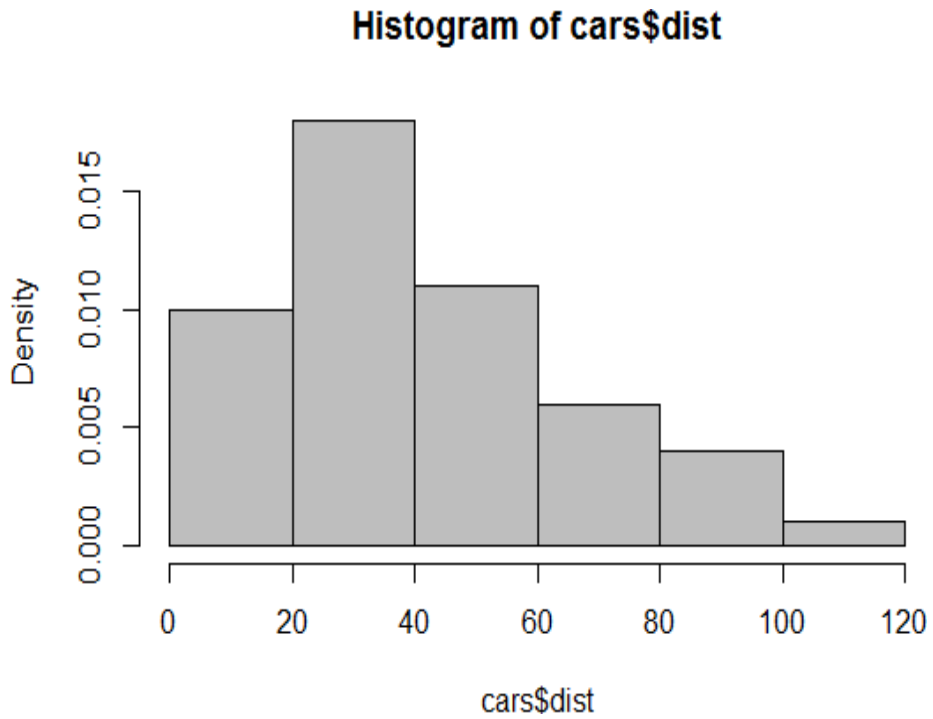


Рисунок 3. Гистограмма плотности распределения для переменной dist

Контрольные вопросы

1. Как найти среднее выборочное значение и стандартное отклонение?
2. Как проверить наличие пропусков в исходных данных?
3. Каким образом функция qplot разбивает выборку на интервалы?

Задания

№ варианта	**1.**	**2.**	**3.**	**4.**	**5.**	**6.**	**7.**	**8.**	**9.**	**10.**
Набор данных	CO2		ChickWeight		iris	Orange	airquality	faithful		
Имя переменной (вектора)	conc	uptake	weight	Time	Sepal.Length	circumference	Wind	Temp	eruptions	waiting

ЛАБОРАТОРНАЯ РАБОТА №2.

Тема: Линейная регрессия.

Цель работы

Изучить приемы исследования корреляционной зависимости, построения парной и множественной линейной регрессии.

Задание

1. Загрузить набор данных варианта, ознакомиться с его содержимым.
2. Построить график корреляционного поля для каждого фактора.
3. Построить уравнение парной линейной регрессии для каждого фактора.
4. Проверить значимость каждого из полученных уравнений регрессии. Показать уравнения регрессии с заданным в варианте доверительным интервалом на графиках.
5. Построить прогнозы по каждому из уравнений парной регрессии для заданных в варианте значений факторов.
6. Построить уравнение множественной линейной регрессии и получить корреляционную матрицу.
7. Построить прогноз по уравнению множественной регрессии для заданных в варианте значений факторов.

Указания к выполнению работы

Для выполнения данной лабораторной работы необходимо инсталлировать и подключить следующие пакеты:

library("psych") - описательные статистики
library("lmtest") - тестирование гипотез в линейных моделях
library("ggplot2") - графики
library("dplyr") - манипуляции с данными
library("MASS") - подгонка распределений
library("stats") – базовый пакет множественной линейной регрессии.

Парная линейная регрессия

Построение парной линейной регрессии рассмотрим на встроенном наборе данных cars.

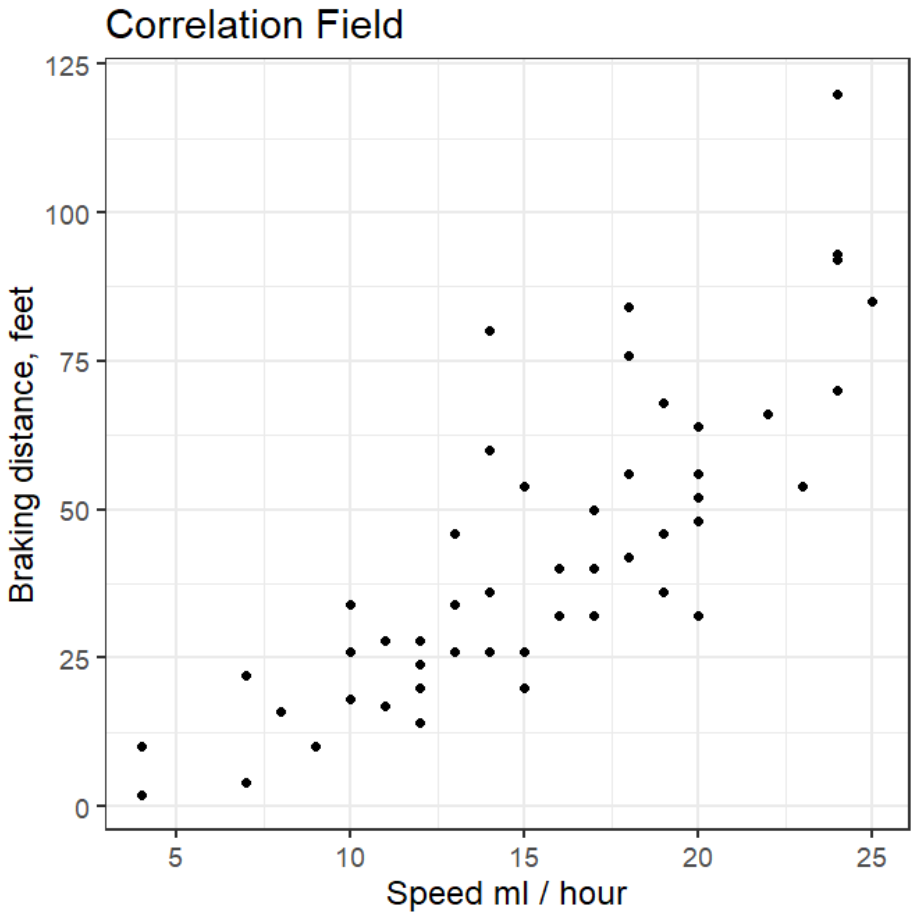
```
d <- cars
```

Проанализируем зависимость длины тормозного пути (dist) от скорости (speed). Визуализируем данные с помощью графика зависимости длины тормозного пути от скорости автомобиля:

```
qplot(data=d, speed, dist)
```

Для этого воспользуемся функциией ggplot (Рисунок 4).

```
> ggplot() +
geom_point(aes(x=d$speed, y=d$dist), size = 2) +
theme_bw(base_size = 18) + xlab("Speed ml / hour")+
ylab("Braking distance, feet") +
```




```
labs(title = "Correlation Field")
```

Рисунок 4. График зависимости длины тормозного пути dist от скорости автомобиля speed

Оценим модель линейной регрессии длины тормозного пути на скорость автомобиля. Для этого командой lm поместим в переменную model модель линейной регрессии, указав dist в качестве зависимой переменной, и через значок ~ переменную speed в качестве регрессора:

```
model <- lm(data=d, dist~speed)
```

Результат выполнения в консоли:

```
> model
Call:
lm(formula = dist ~ speed, data = d)
oefficients:
(Intercept)       speed
   -17.579       3.932
```

Intercept — это константа в уравнении регрессии, speed — коэффициент регрессии.

Таким образом, уравнение регрессии имеет вид:

$$dist^m_i = -17.579 + 3.9324\ speed$$

Можно вывести значения вектора ошибок модели — разницу между реальной длиной тормозного пути dist и полученной по модели $dist^m_i$. Выведем первые 10 значений этого вектора с точностью до двух цифр после запятой:

```
model$residuals[1:10]
      1           2           3           4           5           6           7           8
3.849460 11.849460 -5.947766 12.052234  2.119825 -7.812584 -3.744993  4.255007
      9          10
12.255007 -8.677401
```

Более полный набор расчетов по модели можно получить командой summary:

```
summary(model)
Call:
lm(formula = dist ~ speed, data = d)
Residuals:
    Min      1Q  Median      3Q     Max
-29.069  -9.525  -2.272   9.215  43.201
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791     6.7584  -2.601   0.0123 *
speed         3.9324     0.4155   9.464 1.49e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared:  0.6511,    Adjusted R-squared:  0.6438
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

Кроме коэффициентов регрессии, R выводит:

- стандартные ошибки коэффициентов (Std. Error);
- наблюдаемые значения t-критерия при проверке значимости коэффициентов регрессии (t value);
- P-значения для коэффициентов регрессии (P-value).

Звездочками или точками в столбце справа R показывает значимость или незначимость коэффициентов: *** — значимы на уровне значимости менее 0.001; ** — значимы на уровне значимости 0.001; * — значимы на уровне значимости 0.01; — значимы на уровне

значимости 0.05 и т. д. Эти обозначения приведены в разделе Signif. codes.

Коэффициент детерминации (Multiple R-squared) равен 0.6511; скорректированный коэффициент детерминации (Adjusted R-squared) равен 0.6438. Наблюдаемое значения F-критерия проверки значимости уравнения в целом и P-значение:

F-statistic: 89.57 on 1 and 48 DF, p-value: 1.49e-12

Таким образом, уравнение регрессии получилось значимым.

Добавим на график линию регрессии с 95% доверительными интервалами (Рисунок 5):

```
qplot(data = d, speed, dist) +
stat_smooth(method="lm", level = 0.95) +
theme_bw(base_size = 18)
```

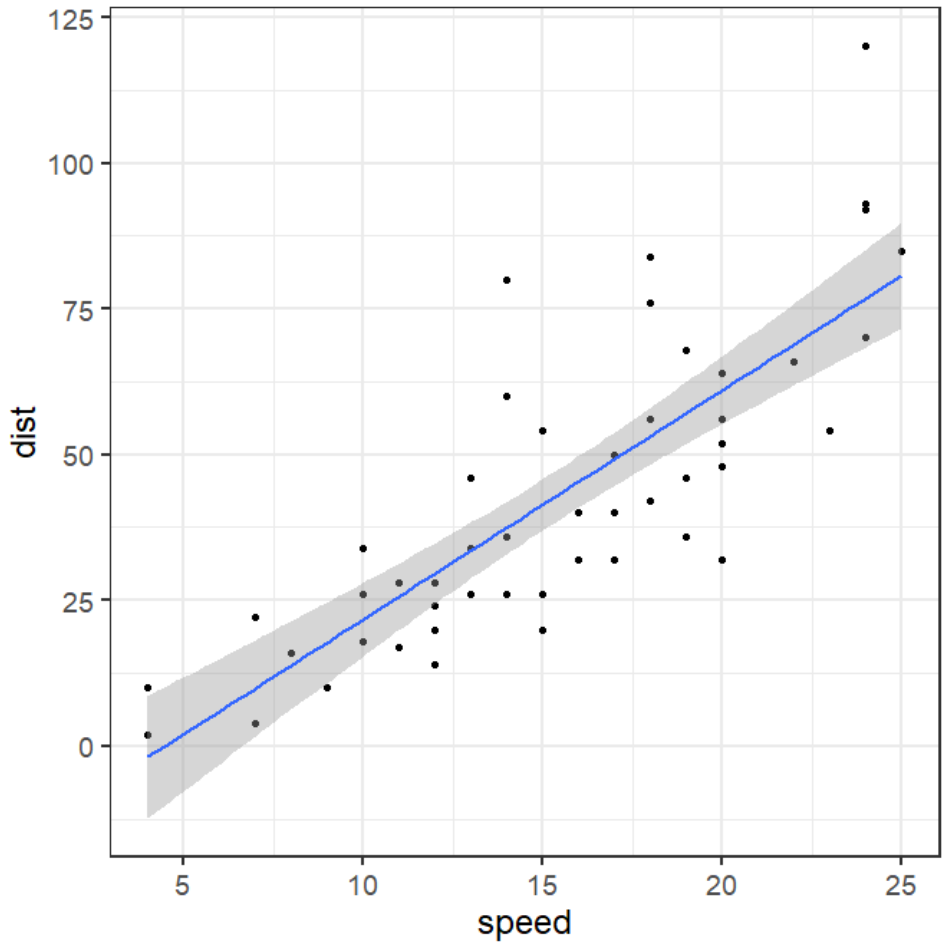


Рисунок 5. График линейной регрессии с доверительными интервалами

Рассчитаем 95% доверительные интервалы для параметров линейной регрессии с помощью функции базового пакета stats:

```
confint(model, level = 0.95)
```

```
              2.5 %    97.5 %
```

```
(Intercept) -31.167850 -3.990340
speed         3.096964  4.767853
```

Необъясненная сумма квадратов отклонений:

```
RSS <- deviance(model)
RSS
[1] 11353.52
```

Рассчитаем полную сумму квадратов, воспользовавшись известными функциями sum и mean:

```
TSS <- sum((y-mean(y))^2)
> TSS
[1] 6238439
```

Для построения прогноза по полученной модели, следует задать значения регрессора и поместить их в структуру data.frame. набора данных:

```
nd<-data.frame(speed=c(40,60))
> nd
  speed
1    40
2    60
```

Определим прогноз функцией predict:

```
predict(model,nd)
      1        2
139.7173 218.3654
```

Множественная линейная регрессия

Рассмотрим встроенный набор данных по социально-экономическим показателям в 47 провинциях Швейцарии в 1888 г.

```
t <- swiss - встроенный набор данных по Швейцарии
```

Этот набор данных содержит 6 переменных по 47 наблюдений, каждая из которых измеряется в процентах (help(swiss)):

Fertility — рождаемость;

Agriculture — % мужчин, занятых в сельском хозяйстве;

Examination — % призывников, получивших высшую оценку на экзамене в армии;

Education — % призывников, имеющих образование помимо начального;

Catholic — % католиков среди населения;

Infant.Mortality — % детей, умерших до года.

Выведем этот набор данных:

```
t <- swiss
```

```
> glimpse(t)
Observations: 47
Variables: 6
$ Fertility        <dbl> 80.2, 83.1, 92.5, 85.8, 76.9, 76.1, 83.8, 92.4, 82...
$ Agriculture      <dbl> 17.0, 45.1, 39.7, 36.5, 43.5, 35.3, 70.2, 67.8, 53...
$ Examination      <int> 15, 6, 5, 12, 17, 9, 16, 14, 12, 16, 14, 21, 14, 1...
$ Education        <int> 12, 9, 5, 7, 15, 7, 7, 8, 7, 13, 6, 12, 7, 12, 5, ...
$ Catholic         <dbl> 9.96, 84.84, 93.40, 33.77, 5.16, 90.57, 92.85, 97....
$ Infant.Mortality <dbl> 22.2, 22.2, 20.2, 20.3, 20.6, 26.6, 23.6, 24.9, 21...
```

Встроенный пакет graphics содержит функцию pairs, позволяющую получить все возможные диаграммы рассеяния на одном графике, а также выполняет их сглаживание с помощью опции pairs(swiss, panel = panel.smooth). Результатом будет график, показанный на Рисунке 6.

Функция cor позволяет вычислить корреляцию между двумя выборками, и вычисляет корреляционную матрицу для всех переменных из набора данных:

```
cor(swiss)
                 Fertility Agriculture Examination Education  Catholic
Fertility    1.0000000  0.35307918 -0.6458827  -0.66378886 0.4636847
Agriculture  0.3530792  1.00000000 -0.6865422 -0.63952252  0.4010951
Examination -0.6458827 -0.68654221  1.0000000  0.69841530 -0.5727418
Education   -0.6637889 -0.63952252  0.6984153  1.00000000 -0.1538589
Catholic     0.4636847  0.40109505 -0.5727418 -0.15385892  1.0000000
Infant.M     0.4165560 -0.06085861 -0.1140216 -0.09932185 0.1754959
             Infant.Mortality
Fertility          0.41655603
Agriculture       -0.06085861
Examination       -0.11402160
Education         -0.09932185
Catholic           0.17549591
Infant.Mortality   1.00000000
```

Корреляционную матрицу можно получить и в другом виде, например, с помощью функции sjp.corr из пакета sjPlot (Рисунок 7):

```
library("sjPlot")
sjp.corr(t) - функция из пакета sjPlot
```

Есть функция, которая позволяет получить корреляционную матрицу, диаграммы рассеяния и сглаженные распределения одновременно (Рисунок 8):

```
library("GGally")
ggpairs(t) – Ggally функция из пакета
```

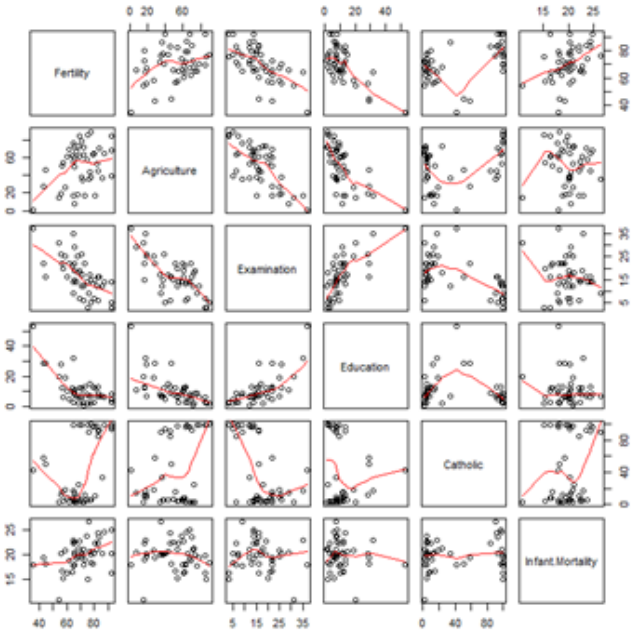


Рисунок 6. Диаграммы рассеяния, полученные с помощью функции pairs

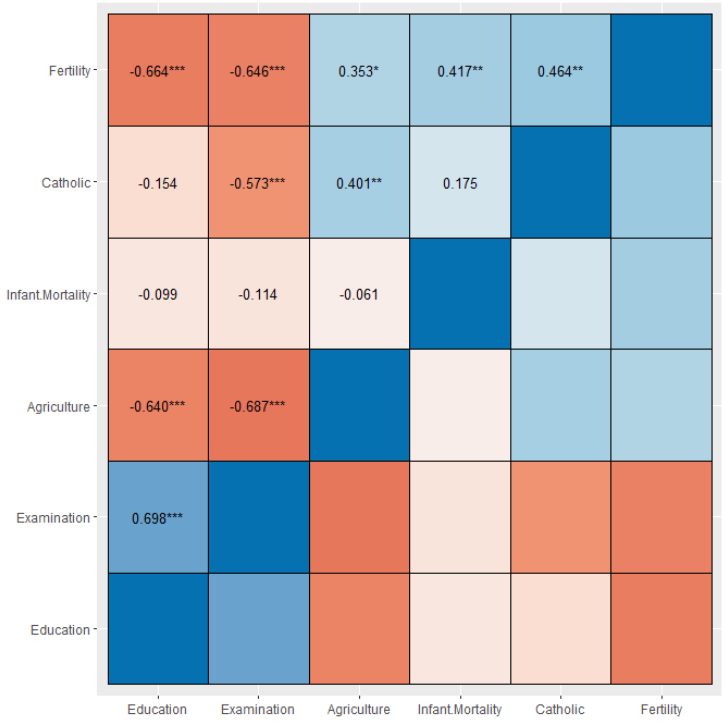


Рисунок 7. Корреляционная матрица, полученная с помощью функции sjp.corr

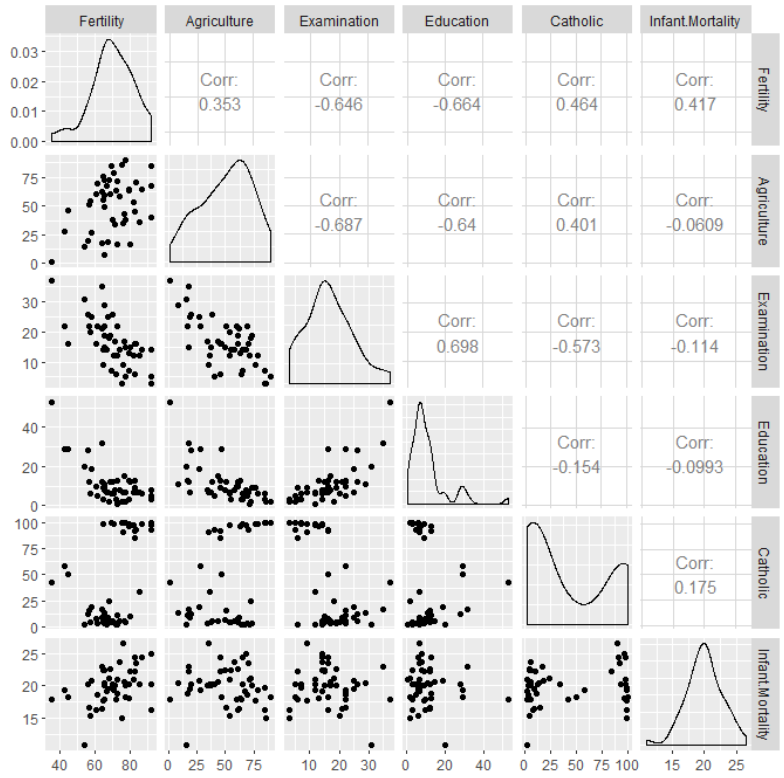


Рисунок 8. Корреляционная матрица, диаграммы рассеяния и сглаженные распределения, полученные с помощью функции ggpairs

Для оценки регрессии рождаемости с остальными переменными, используют функцию lm, а регрессоры перечислить через знак «плюс»: model2 <- lm(data=t, Fertility~Agriculture+Education+Catholic)

В данном случае регрессорами стали % занятых в с/х; % католического населения и % имеющих образование выше начального.

Получить оценки коэффициентов уравнения регрессии, а также проверить основные гипотезы можно с помощью функция summary: summary(model2)

```
Call:
lm(formula = Fertility ~ Agriculture + Education + Catholic,
    data = t)
Residuals:
    Min      1Q  Median      3Q     Max
-15.178  -6.548   1.379   5.822  14.840
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 86.22502    4.73472  18.211  < 2e-16 ***
Agriculture -0.20304    0.07115  -2.854  0.00662 **
Education   -1.07215    0.15580  -6.881 1.91e-08 ***
Catholic     0.14520    0.03015   4.817 1.84e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Residual standard error: 7.728 on 43 degrees of freedom
Multiple R-squared: 0.6423, Adjusted R-squared: 0.6173
F-statistic: 25.73 on 3 and 43 DF, p-value: 1.089e-09

Построим прогноз по аналогии с парной линейной регрессией. Отличие заключается лишь в том, что в наборе данных необходимо указать значения каждого фактора. Для этого создадим новый набор данных: nd2 <- data.frame(Agriculture=0.5,Catholic=0.5, Education=20)
predict(model2,nd2)
1
64.75316

Построение прогноза по нескольким точкам выполняется с помощью векторов значений. Для этого создаем новый набор данных: nd2<-data.frame(Agriculture = c(0.5,0.8), Catholic=c(0.5,0.65), Education=c(20, 25))
predict(model2,nd2)
1 2
64.75316 59.35330

Контрольные вопросы

1. Что такое Intercept в парной линейной регрессии?
2. Как проверить значимость уравнения регрессии в целом?
3. Как узнать разброс значений коэффициентов регрессии?

Задания

№ варианта	**1.**	**2.**	**3.**	**4.**	**5.**
Набор данных	airquality	state.x77	Cars93*	Cars93*	Cars93*
y	Ozone	Life Exp***	Price	Min.Price	Max.Price
Факторы и их значения для прогноза	Solar.R=350 Wind = 8,3 Temp = 80	Illiteracy = 1,0 Murder = 12 Income = 4000	Horsepower = 200 RPM = 5200 Passengers = 4	Horsepower = 210 RPM = 5500 Passengers = 4	Horsepower = 220 RPM = 6000 Passengers= 6
№ варианта	**6.**	**7.**	**8.**	**9.**	**10.**
Набор данных	stackloss	longley	longley	LifeCycleSavings	Anscombe**
y	stack.loss	GNP	Employed	sr	education
Факторы и их значения для прогноза	Air.Flow= 55 Water.Temp= 20 Acid.Conc = 89	Unemployed = 221 Armed.Forces= 180 Population= 125	GNP.deflator=102 Armed.Forces= 170 Population = 110	pop15 = 35,5 pop75 = 1,5 dpi = 2500 ddpi = 2,15	income= 3200 young= 347,8 urban = 425

ЛАБОРАТОРНАЯ РАБОТА №3.

Тема: Временные ряды

Цель работы: Изучить базовые способы анализа, моделирования и прогнозирования временных рядов.

Задание

1. Загрузить данные из временного ряда для своего варианта за последние полгода.
2. Построить график временного ряда с графиками автокорреляционной и частной автокорреляционной функций.
3. Подобрать вручную порядок ARMA-модели. Для подбора использовать визуальный анализ графика.
4. Подобрать ARMA-модель автоматически.
5. Построить прогноз на указанное в варианте число шагов. Показать прогноз на графике.

Указания к выполнению работы

Анализ временных рядов

Для выполнения работы необходимо получить следующие пакеты:

library("lubridate") - работа с датами
library("zoo") - работа с временными рядами
library("xts") - дополнительные функции для работы с временными рядами
library("dplyr") - работа с наборами данных
library("ggplot2") - графики
library("forecast") - прогнозы
library("lmtest") - тестирование гипотез в линейных моделях
library("quantmod") - загрузка данных с различных источников

Сгенерируем тестовые выборки с помощью функции arima.sim из базового пакета stats. Выполним симуляцию AR(1)-процесса в 100 наблюдений по модели: $y_t = 0.165y_{t-1} + \varepsilon_t$ и поместим в переменную y.

```
y <- arima.sim(n=100, list(ar=0.165))
```

В этой функции необходимо указать объем выборки, а в параметре list коэффициенты модели.

Можно построить полученный ряд на графике (рис. 17):

```
plot(y)
```

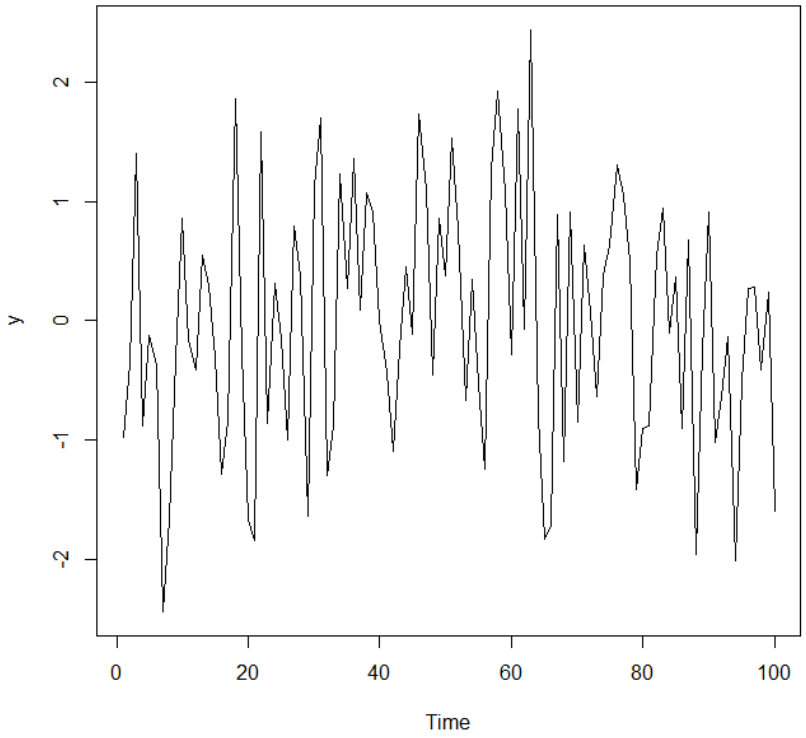


Рис. 7. График временного ряда

Графики автокорреляционной и частной автокорреляционной функций для процесса y вызываются командами соответственно: Acf(y) и Pacf(y). Все три графика командой: tsdisplay(y). Результат выполнения команды показан на рисунке 18.

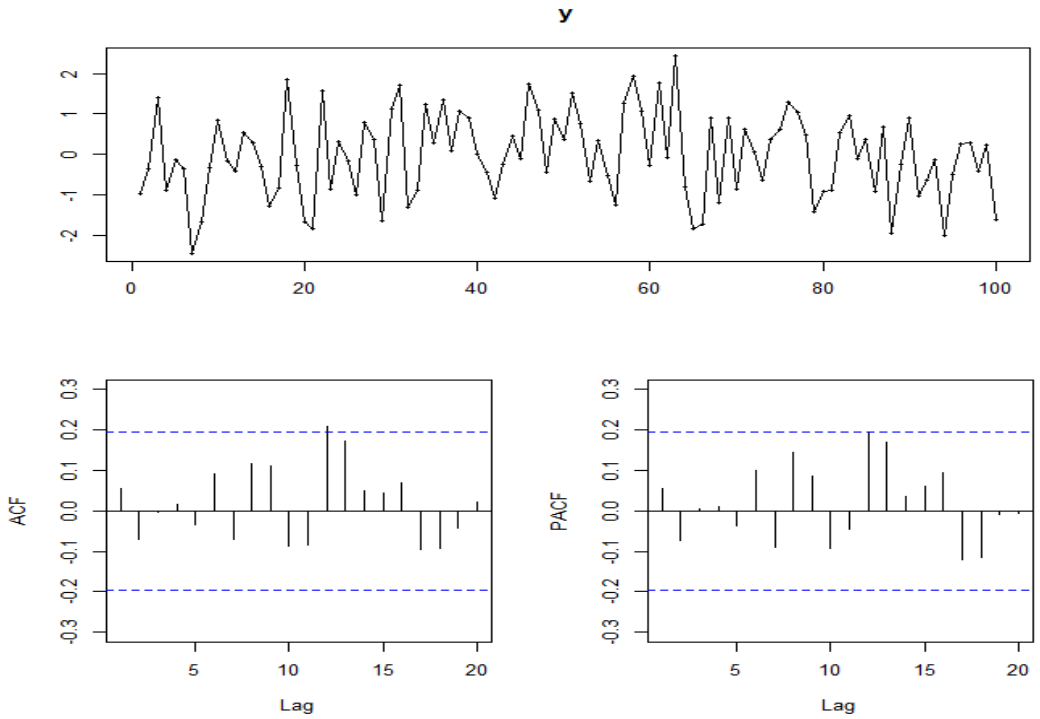


Рисунок 10. Графики временного ряда, автокорреляционной и частной автокорреляционной функций

В анализе временных рядов достаточно широкое распространение получили авторегрессионные модели. В R языке программно реализованы множество моделей авторегрессии. Рассмотрим основные из них модели.

AR - авторегрессионная модель временных рядов, в которой значения временного ряда в данный момент линейно зависят от предыдущих значений этого же ряда. Авторегрессионный процесс порядка p AR(p).

ARMA (модель авторегрессии) — скользящего среднего одна из математических моделей, использующихся для анализа и прогнозирования стационарных временных рядов в статистике. Модель ARMA обобщает две более простые модели временных рядов — модель авторегрессии (AR) и модель скользящего среднего (MA).

ARIMA (модель Бокса — Дженкинса) — интегрированная модель авторегрессии — скользящего среднего — модель и методология анализа временных рядов. Является расширением моделей ARMA для нестационарных временных рядов, которые можно сделать стационарными взятием разностей некоторого порядка от исходного временного ряда (так называемые интегрированные или разностно-стационарные временные ряды). Модель ARIMA(p, d, q) означает, что разности временного ряда порядка d подчиняются модели ARMA(p, q).

Методология ARIMA (Бокса — Дженкинса) к временным рядам заключается в том, что в первую очередь оценивается стационарность ряда. Различными тестами выявляются наличие единичных корней и порядок интегрированности временного ряда (обычно ограничиваются первым или вторым порядком). Далее при необходимости (если порядок интегрированности больше нуля) ряд преобразуется взятием разности соответствующего порядка и уже для преобразованной модели строится некоторая ARMA-модель, поскольку предполагается, что полученный процесс является стационарным, в отличие от исходного нестационарного процесса (разностно-стационарного или интегрированного процесса порядка d.

Рассмотрим примеры симуляции основных моделей авторегрессии процессов: AR, MA, ARMA, ARIMA.

Процесс MA(1) $y_t = \varepsilon_t - 0.8\varepsilon_{t-1}$

```
arima.sim(n=100, list(ma=-0.8))
```

Процесс ARMA(1,1) $y_t = 0.5y_{t-1} + \varepsilon_t - 0.8\varepsilon_{t-1}$

```
arima.sim(n=100, list(ma=-0.8, ar=0.5))
```

Процесс ARMA(2,2) $y_t = 0.9y_{t-1} - 0.5y_{t-2} + \varepsilon_t - 0.2\varepsilon_{t-1} + 0.3\varepsilon_{t-2}$

```
arima.sim(n = 100, list(ar = c(0.9, -0.5), ma = c(-0.2, 0.3))
```

Процесс случайного блуждания: $y_t = y_{t-1} + \varepsilon_t$

```
arima.sim(n=100, list(order=c(0,1,0)))
```

Белый шум: $y_t = \varepsilon_t$;

```
arima.sim(n=100, list(order=c(0,0,0)))
```

Подберем различные ARIMA-модели под реальные данные. Для этого выберем из представленных в пакетах данные, например, временного ряда, содержащий 98 ежегодных наблюдений за уровнями воды в озере Гурон с 1875 по 1972 гг. (рис. 19).

```
head(LakeHuron, 11)
Time Series:
Start = 1875
End = 1885
Frequency = 1
 [1] 580.38 581.86 580.97 580.80 579.79 580.39 580.42 580.82 581.40 581.32
[11] 581.44
```

Оценим ARMA(2,1) модель с помощью функции Arima из пакета forecast:

mod<-Arima(y, order=c(2,0,1))

Результат оценивания:

summary(mod)

```
Series: y
ARIMA(2,0,1) with non-zero mean
Coefficients:
         ar1      ar2      ma1     mean
      0.2555  0.2171  -0.2525  -0.0345
s.e.  0.3003  0.1015   0.2993   0.1351
sigma^2 estimated as 0.9638:  log likelihood=-138.07
AIC=286.13   AIC c=286.77   BIC=299.16
Training set error measures:
                    ME      RMSE       MAE      MPE     MAPE      MASE
Training set 0.002868029 0.9619142 0.7718891 102.4087 113.4457 0.6963022
                    ACF1
Training set -0.003086129
```

В результате получим модель:

$y_t = y_{t-1} + \varepsilon t$

В строке s.e. представлены стандартные ошибки коэффициентов. Оценка дисперсии $\hat{\sigma}^2 = 0.4749$. Строкой ниже перечислены значения критерия Акаике (AIC=216.48), скорректированного критерия Акаике (AICc=217.13) и критерия Шварца (BIC=229.4).

Значение критерия Акаике можно вывести командой:

AIC(mod)

```
[1] 286.1336
```

Командой fitted(mod) можно получить модельные значения временного ряда:

```
head(fitted(mod))
Time Series:
Start = 1
End = 6
Frequency = 1
[1] -0.01768787 -0.05682477  0.08159288 -0.29484919 -0.31361364  0.08158120
```

Реальные и модельные значения можно представить на одном графике (Рисунок 11) с помощью функции:

matplot(cbind(y, fitted(mod)), type='l')

Функция matplot представляет значения столбцов матрицы, поэтому аргумент должен задать столбцы. Команда cbind соединяет в матрицу столбец *y* и столбец значений, рассчитанных по модели. type='l' указывает, что тип графика — линия.

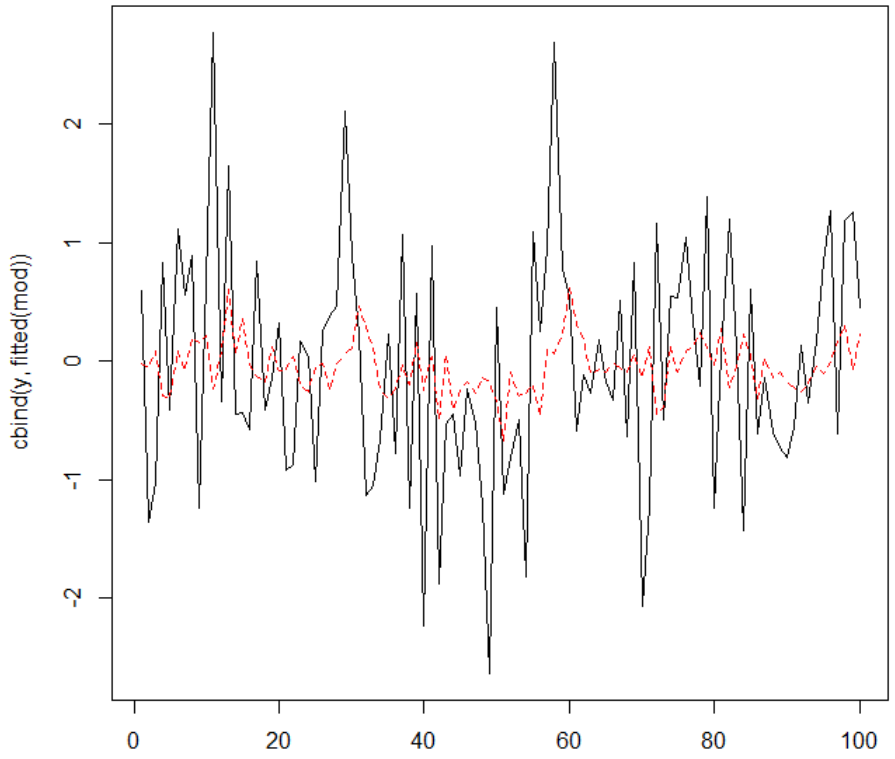


Рисунок 11. График модельных и реальных значений временного ряда

Построим прогноз вперед на 7 шагов:

prognoz<- forecast(mod, h=7)

Выведем полученные значения. Результат включает 80% и 95% доверительный интервал для прогноза:

> prognoz

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
101	0.313181523	-0.9449811	1.571344	-1.611012	2.237375
102	0.160509704	-1.0976587	1.418678	-1.763693	2.084712
103	0.090805043	-1.1968877	1.378498	-1.878551	2.060161
104	0.039844628	-1.2497973	1.329487	-1.932492	2.012182
105	0.011688513	-1.2802883	1.303665	-1.964220	1.987597

106 -0.006570872 -1.2990279 1.285886 -1.983213 1.970072
107 -0.017349965 -1.3100656 1.275366 -1.994388 1.959688

Построим график прогноза (Рисунок 12) с помощью функции: plot(prognoz)

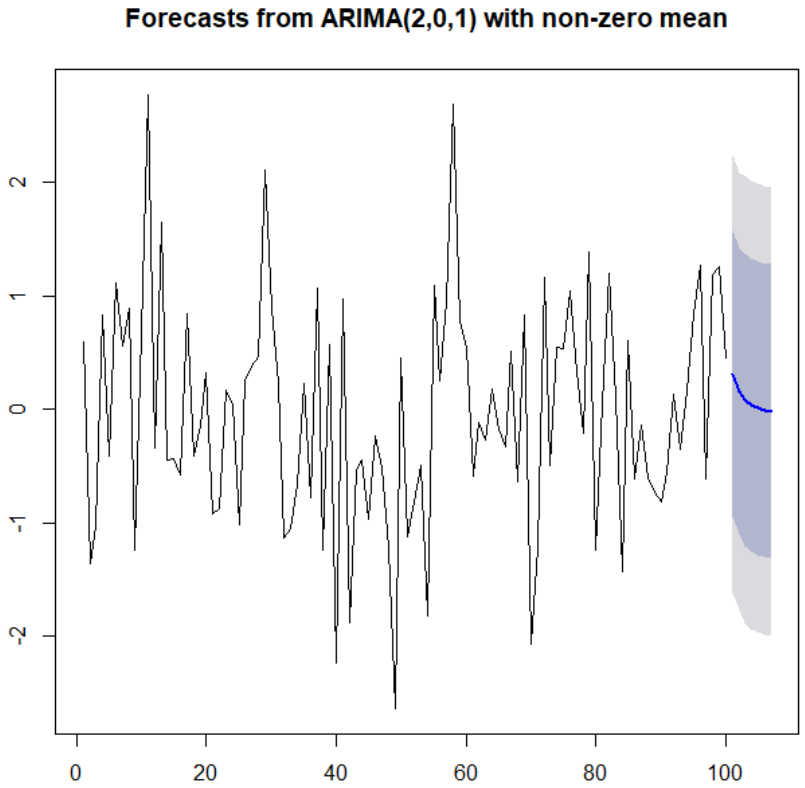


Рисунок 12. График прогноза по модели ARIMA

R предоставляет возможность автоматически подбирать ARIMA-модель по штрафному критерию (минимальное значение критерия Акаике):

```
mod_a<- auto.arima(y)
summary(mod_a)
Series: y
ARIMA(2,0,0) with zero mean
Coefficients:
         ar1       ar2
      0.7518  -0.3743
s.e.  0.1165   0.1173
sigma^2 estimated as 0.3745:  log likelihood=-57.77
AIC=121.54   AICc=121.95   BIC=127.97
Training set error measures:
                    ME     RMSE      MAE      MPE     MAPE      MASE
```

Training set 0.01501697 0.6022078 0.4115077 20.34583 124.5448 0.7750357
ACF1
Training set -0.04503634

Таким образом, получим модель:

$y_t = y_{t-2} + \varepsilon_t + 0.7518\,\varepsilon_{t-1} - 0.3743t$

Построим график процесса и модели:

> plot(forecast(mod_a,h=20))

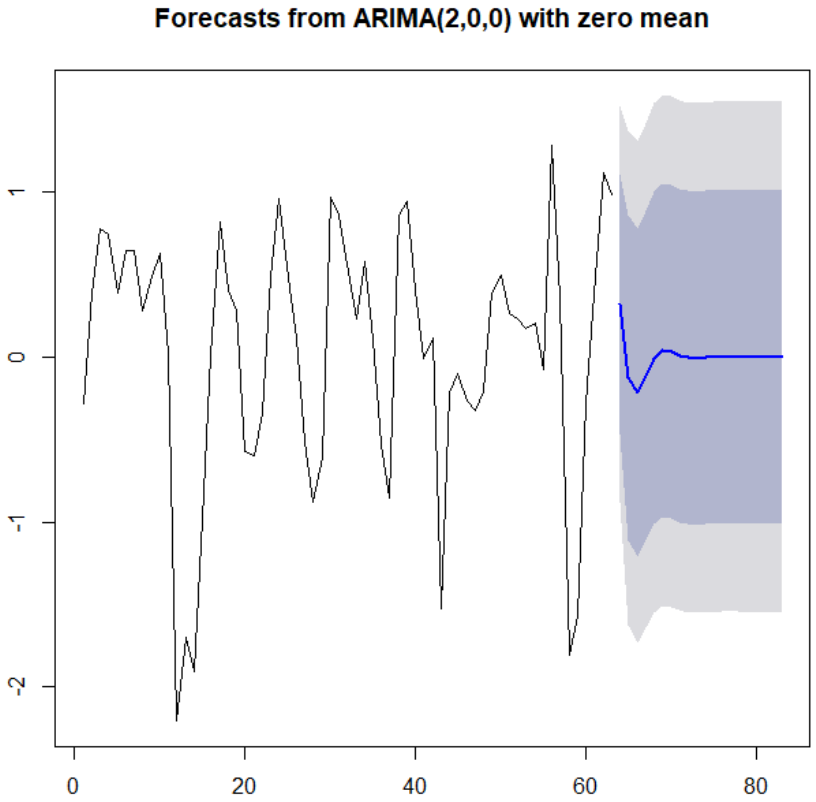


Рисунок 13. График процесса и модели ARIMA(2, 0, 0)

Загрузка данных из различных источников в R

Для корректного перевода дат на английский язык на русский в Windows, необходимо сначала выполнить следующую команду:

Sys.setlocale("LC_TIME","C")
[1] "C"
> head(C)
1 function (object, contr, how.many, ...)
2 {
3 if (!nlevels(object))

```
4       stop("object not interpretable as a factor")
5    if (!missing(contr) && is.name(Xcontr <- substitute(contr)))
6       contr <- switch(as.character(Xcontr), poly = "contr.poly",
```

Пакет quantmod позволяет загружать данные из нескольких источников, а именно:

- Yahoo! Finance (OHLC data) — http://finance.yahoo.com/;
- Federal Reserve Bank of St. Louis FRED® (11,000 экономических временных рядов) — http://research.stlouisfed.org/fred2/;
- Google Finance (OHLC data) — http://finance.google.com/;
- Oanda, The Currency Site (FX and Metals) — http:www.oanda.com/.

Для загрузки данных используется одна и та же функция getSymbols. Например, о стоимости акций Гугл с finance.google.com за период с 1 января 2018г. по 1 декабря 2018г.:

```
getSymbols("GOOG", src="yahoo", from="2018-01-01", to="2018-12-01")
```

GOOG — это краткое название акций компании на бирже (ти́кер). Теперь эти данные загружены в переменную с таким же названием GOOG. Если не указывать начальную или конечную дату, то будут загружены все доступные данные. Можно посмотреть шесть первых значений и шесть последних значений командами head и tail соответственно.

```
head(GOOG)
```

	GOOG.Open	GOOG.High	GOOG.Low	GOOG.Close	GOOG.Volume	GOOG.Adjusted
2018-01-02	1048.34	1066.94	1045.230	1065.00	1237600	1065.00
2018-01-03	1064.31	1086.29	1063.210	1082.48	1430200	1082.48
2018-01-04	1088.00	1093.57	1084.002	1086.40	1004600	1086.40
2018-01-05	1094.00	1104.25	1092.000	1102.23	1279100	1102.23
2018-01-08	1102.23	1111.27	1101.620	1106.94	1047600	1106.94
2018-01-09	1109.40	1110.57	1101.231	1106.26	902500	1106.26

```
tail(GOOG)
```

	GOOG.Open	GOOG.High	GOOG.Low	GOOG.Close	GOOG.Volume	GOOG.Adjusted
2018-11-23	1030.00	1037.590	1022.399	1023.88	691500	1023.88
2018-11-26	1038.35	1049.310	1033.910	1048.62	1942800	1048.62
2018-11-27	1041.00	1057.580	1038.490	1044.41	1803200	1044.41
2018-11-28	1048.76	1086.840	1035.760	1086.23	2475400	1086.23
2018-11-29	1076.08	1094.245	1076.000	1088.30	1468900	1088.30
2018-11-30	1089.07	1095.570	1077.880	1094.43	2580200	1094.43

На Рисунке 14 представлен график процесса изменения цены акций:

barChart(GOOG, theme = "white")

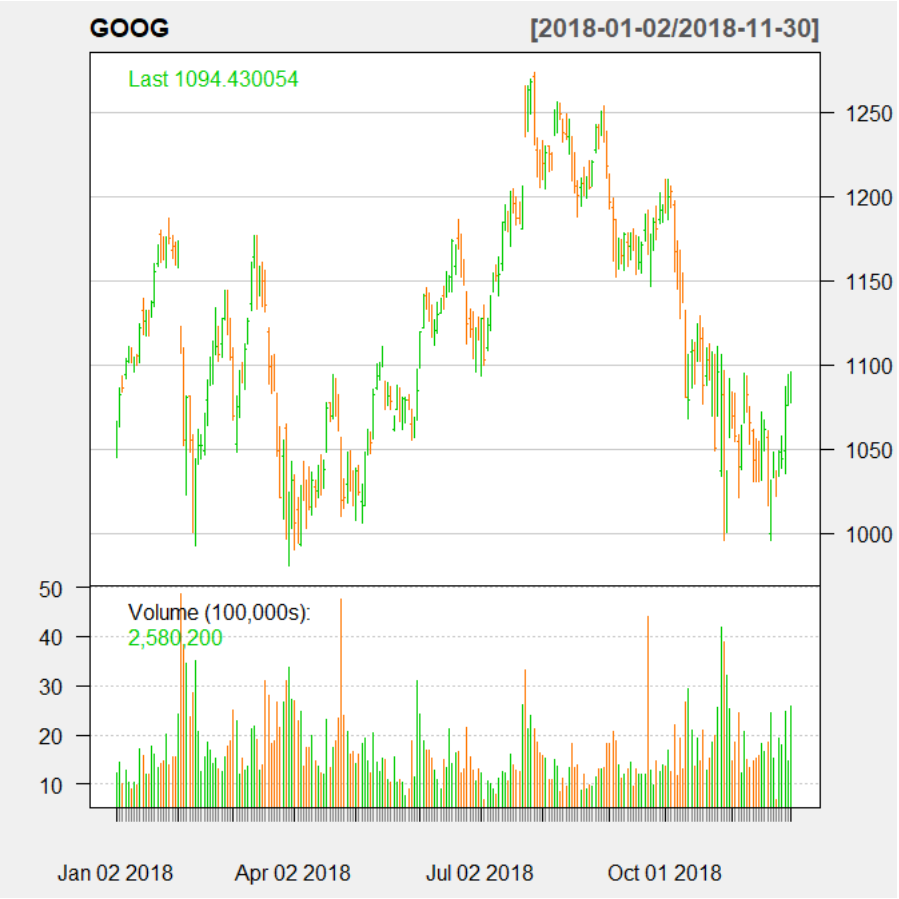


Рисунок 14. График стоимости акций Гугл GOOG

Также можно посмотреть на цену закрытия с помощью функции tsdisplay (рис. 11):

tsdisplay(GOOG$GOOG.Close)

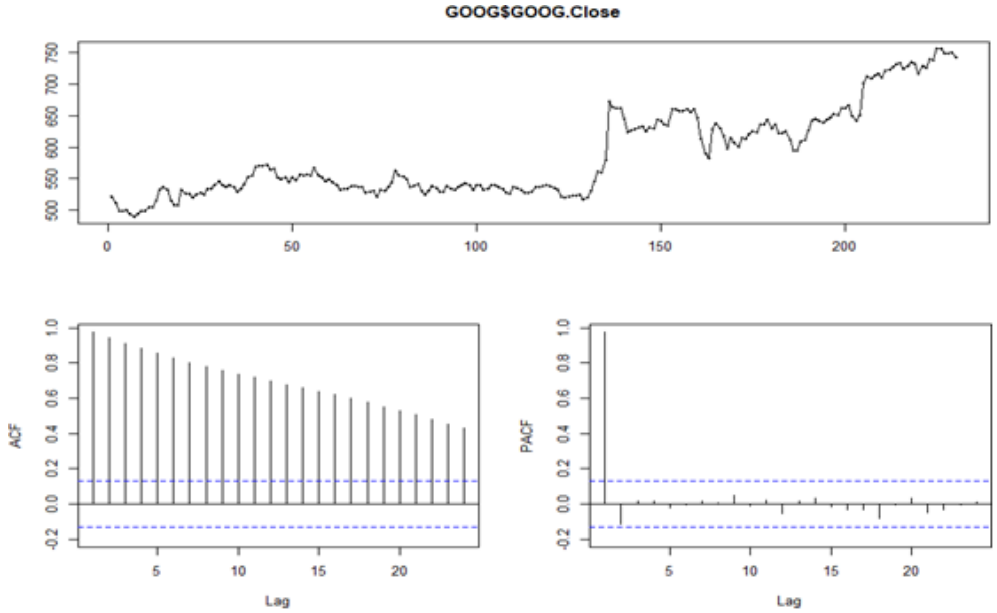


Рис. 15 График цены закрытия акций Гугл GOOG

Контрольные вопросы

1. В чем отличие автокорреляционной и частной автокорреляционной функций?
2. Что такое случайное блуждание и белый шум?
3. Как можно сравнить модели временного ряда по точности?

Варианты заданий

Изучить примеры:

1. Индекс потребительских цен для всех городских потребителей: все товары:

```
getSymbols("CPIAUCNS", src="FRED")
head(CPIAUCNS)
tail(CPIAUCNS)
barChart(CPIAUCNS, theme = "white")
```

2. Данные о стоимости акций Apple с finance.yahoo.com
   getSymbols("AAPL", src = "yahoo", from="2018-01-01", to="2018-12-01")
   head(AAPL)
   tail(AAPL)
   barChart(AAPL, theme = "white")
3. Японские свечи — вид интервального графика и технический индикатор, применяемый для отображения изменений биржевых котировок акций:
   candleChart(AAPL, multi.col=TRUE, theme="white")
4. Провести сранительный анализ изменения цен на акции Гугл, Аппл и Майкрософт:
   getSymbols(c("AAPL", "MSFT", "GOOG"), src = "yahoo", from="2018-01-01", to="2018-12-01")
   head(MSFT)
   tail(MSFT)
   barChart(MSFT, theme = "white")

Для более детальной информации можно обратиться на сайт разработчиков пакета [7].

Задания

№ варианта	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
Компания	Yandex	Twitter	Facebook	ВКонтакте	Amazon	EBay	Microsoft	Сбербанк	Вымпелком	МТС
Тикер	YNDX	TWTR	FB	VK	AMZN	EBAY	MSFT	SBNC	VIP	MBT
Глубина прогноза	10	15	20	25	30	25	20	15	10	20

ЛАБОРАТОРНАЯ РАБОТА №4.

Тема: Генетические алгоритмы

Цель работы: Научиться применять генетические алгоритмы (ГА) для поиска оценок параметров нелинейных моделей.

Задание

Генетические алгоритмы

1. Выполнить генерацию значений x и y.
2. Построить график зависимости y от x.
3. Задать функцию зависимости y от x в R и соответствующую ей функцию МНК.
4. Определить минимальные и максимальные значения параметров.
5. Запустить поиск решения с помощью ГА. Размер популяции, предельное число итераций и условие останова алгоритма заданы в варианте.
6. Построить график значений fitness-функции и график с исходными данными и полученной моделью.
7. Повторить пункты 5 и 6 еще 2 раза и сравнить полученные результаты.
8. Выбрать наилучший из трех результатов по значению коэффициента детерминации.

Рассмотрим решение задачи нелинейного МНК с помощью генетического алгоритма на встроенном наборе данных *trees*.

Для работы необходимо подключить следующие пакеты [10]:

library("GA") - генетические алгоритмы

library("dplyr") - работа с наборами данных

library("ggplot2") - графики

library("spuRs") - содержит набор данных trees

Получим описание набора данных по деревьям trees из пакета spuRs командой (рис. 25): help("trees", package = "spuRs")

```
head(trees, 5)
    ID   Age  Vol
1 1.1.1  9.67   5
2 1.1.1 19.67  38
3 1.1.1 29.67 123
4 1.1.1 39.67 263
5 1.1.1 49.67 400
```

В этом наборе данных содержится 1200 наблюдений и три переменных (ID дерева, который включает номер местоположения, площадки и номер дерева; возраст дерева; объем в куб. дм). Будем рассматривать зависимость объема дерева (переменная Vol) от возраста (переменная Age).

Активируем набор данных командой:

```
data("trees", package = "spuRs")
```

Посмотрим на этот набор данных:

```
glimpse(trees)
Observations: 1,200
Variables: 3
$ ID  <fct> 1.1.1, 1.1.1, 1.1.1, 1.1.1, 1.1.1, 1.1.1, 1.1.1, 1.1.1, 1.1.1, ...
$ Age <dbl> 9.67, 19.67, 29.67, 39.67, 49.67, 59.67, 69.67, 79.67, 89.67, 9...
$ Vol <dbl> 5, 38, 123, 263, 400, 555, 688, 820, 928, 1023, 1104, 1156, 121...
```

Выберем для дальнейшей работы только данные для деревьев с определенным местоположением, например, ID = 1.3.111. Поместим в переменную tree выбранную часть исходной выборки:

```
tree<- trees[trees$ID == "1.3.11", 2:3]
> head(tree)
     Age     Vol
153  2.44   2.2
154 12.44  20.0
155 22.44  93.0
156 32.44  262.0
157 42.44  476.0
158 52.44  705.0
```

Команда trees[] выбирает нужные столбцы из набора данных, таким образом, что остаются только второй и третий столбцы, но так, чтобы в первом столбце значение параметра ID было равно 1.3.11.

Посмотрим теперь на результат отбора tree:

```
head(tree)
     Age   Vol
153  2.44   2.2
154 12.44  20.0
155 22.44  93.0
156 32.44 262.0
157 42.44 476.0
158 52.44 705.0
glimpse(tree)
Observations: 12
Variables: 2
$ Age <dbl> 2.44, 12.44, 22.44, 32.44, 42.44, 52.44, 62.44, 72.44, 82.44, 9...
```

$ Vol <dbl> 2.2, 20.0, 93.0, 262.0, 476.0, 705.0, 967.0, 1203.0, 1409.0, 16...

В результате отбора получен набор данных из 12 наблюдений с двумя переменными. Опишем зависимость объема дерева (y) от его возраста (x) с помощью логистической функции Ричардса:

$$y = a\left(1 - e^{-bx}\right)^c$$

Далее применим метод наименьших квадратов нелинейными параметрами модели:

$$a, b, c = \arg\min_{a,b,c} \sum_{i=1}^{12} \left(Vol_i - a\left(1 - e^{-b \cdot Age_i}\right)^c \right)^2$$

Минимизируем сумму квадратов ошибок численно - с помощью генетического алгоритма. Генетический алгоритм реализован в R функцией ga из пакета GA.

Функция ga максимизирует функцию fitness, которую следует задать. Поскольку стоит задача минимизации, то воспользуемся следующим соотношением:

$$\arg\max_{\Theta}(\Theta) = \arg\min_{\Theta}(-\Theta)$$

Для этого зададим в R функцию Ричардса - она будет использована для функции fitness, указанной в аргументах функции ga. Пусть вектор параметров функции обозначен theta:

```
theta<-c('a','b','c')
theta
[1] "a" "b" "c"
```

Тогда a = theta[1], b = theta[2], c = theta[3]. Задаем функцию richards:

```
richards<- function(x, theta)
theta[1] * (1 - exp(-theta[2] * x))^theta[3]
```

В первой строке указан аргумент функции x и вектор параметров theta. Через пробел в следующей строке описывается сама функция Ричардса, порядок параметров a, b, c в векторе theta должен сохраняться.

Задаем функцию fit суммы квадратов ошибок и после пробела поставим перед функцией суммы квадратов ошибок знак минус:

```
fit<- function(theta, x, y) -sum((y - richards(x, theta))^2)
```

В функции ga задаются следующие параметры:

fit — нужная нам функция для аргумента fitness, которую максимизирует генетический алгоритм;

type выберем real-valued, поскольку возраст деревьев и объем являются действительными числами;
x и y — это соответственно возраст дерева (tree$Age) и объем дерева (tree$Vol) — аргументы функции fitness;
min — это вектор минимальных значений параметров a, b, c функции Ричардса;
max — вектор максимальных значений параметров a, b, c;
crossover — функция в R, выполняющая кроссовер, т. е. функция которая образует потомков, объединив часть генетической информации от родителей;
popsize — размер популяции;
maxiter — максимальное число итераций, после которого работа генетического алгоритма прекращается;
run — число последовательных поколений без какого-либо улучшения в значении fitness-функции перед остановом алгоритма и т. д.

Запишем результат выполнения функции ga в переменную myGA, задав значения описанным аргументам:

```
myGA<- ga(type = "real-valued", fitness = fit,x = tree$Age, y = tree$Vol,
lower = c(3000, 0, 2), upper = c(4000, 1, 4),
popSize = 500, crossover = gareal_blxCrossover,
maxiter = 5000, run = 200, names = c("a", "b", "c"))
```

Функция при работе будет выводить в консоль значения fit на каждой итерации алгоритма:

```
GA | iter = 1 | Mean = -72419457.3 | Best =   -291334.8
GA | iter = 2 | Mean = -63473178.46 | Best =    -49558.01
GA | iter = 3 | Mean = -54600630.33 | Best =    -49558.01
GA | iter = 4 | Mean = -49167737.25 | Best =    -19542.81
GA | iter = 5 | Mean = -40984882.20 | Best =    -13161.85
GA | iter = 6 | Mean = -32543947.79 | Best =    -12542.47
GA | iter = 7 | Mean = -25654182.201 | Best =     -7127.087
```

Посмотрим на краткие результаты подбора функции Ричардса на исходные данные:

```
summary(myGA)
-- Genetic Algorithm ------------------
GA settings:
Type                = real-valued
Population size     = 500
Number of generations = 5000
Elitism             = 25
```

Crossover probability = 0.8
Mutation probability = 0.1
Search domain =
a b c
lower 3000 0 2
upper 4000 1 4
GA results:
Iterations = 1801
Fitness function value = -2773.837
Solution =
a b c
[1,] 3592.47 0.0154404 2.783237

В результате получено следующее уравнение регрессии:

$$\hat{y} = 3589.788\left(1 - e^{-0.015x}\right)^{2.786}$$

Команда plot(myGA) строит график изменения значений функции fitness на всех 1801 итераций алгоритма (Рисунок 16).

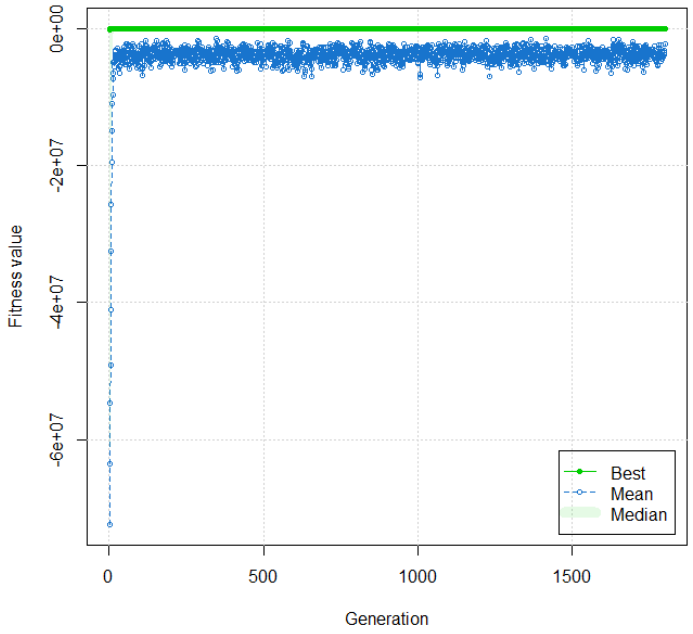


Рисунок 16 Изменение значений fitness-функции в зависимости от числа итераций

Модельные значения y можно получить командой:
richards(tree$Age, myGA@solution)
[1] 0.3711079 27.9843786 117.7207450 268.8958483 467.7303553
[6] 697.7710514 944.0183718 1194.3551448 1439.7523288 1673.9526216
[11] 1892.9743287 2094.5983888

Построим на графике исходную выборку и результат подбора функции Ричардса (Рисунок 17):

```
ggplot() +
geom_point(aes(x=tree$Age, y=tree$Vol))+
geom_line(aes(x=tree$Age, y=richards(tree$Age, myGA@solution)))
```

В данной команде указано:

ggplot() активирует построение графика с помощью функций пакета ggplot2;

geom_point(aes(x=tree$Age, y=tree$Vol)) — строит точками зависимость объема дерева от возраста по исходной выборке;

geom_line(aes(x=tree$Age, y=tree$Vol))) — строит линиейные модельные значения объема деревьев в зависимости от возраста, рассчитанные по функции Ричардса с параметрами, найденными генетическим алгоритмом.

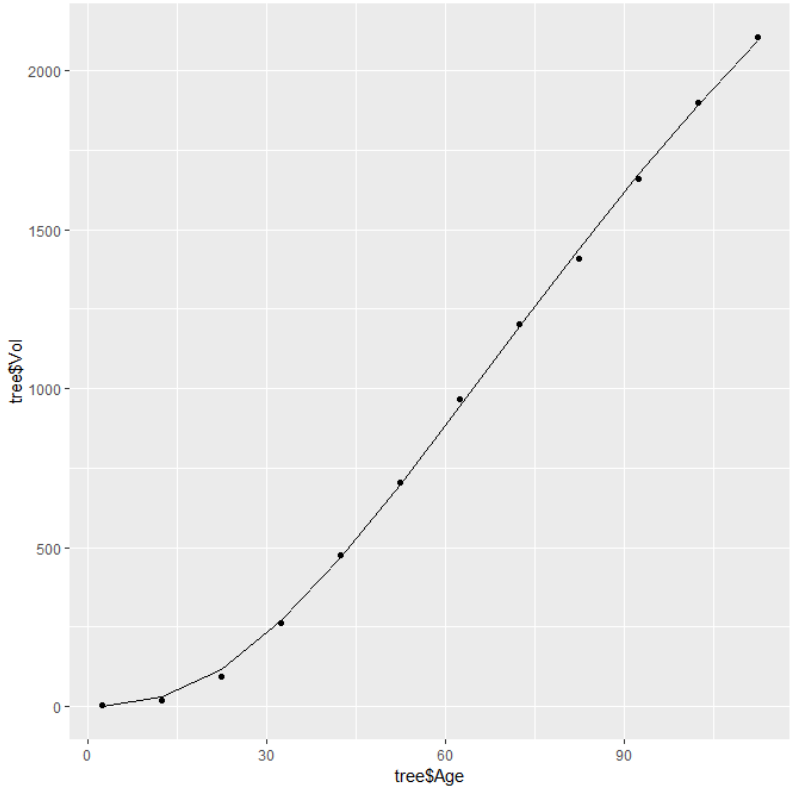


Рисунок 17. График исходных данных и полученной функции Ричардса

Для оценки точности модели рассчитаем ее коэффициент детерминации:

```
RSS <- sum((tree$Vol-richards(tree$Age, myGA@solution))^2)
TSS <- sum((tree$Vol-mean(tree$Vol))^2)
R2 <- 1-RSS/TSS
[1] 0.9995554
```

С помощью полученной модели построим прогноз объема дерева для возраста 100 и 200 лет:

```
Age_forecast <- c(100,200)
> richards(Age_forecast, myGA@solution)
[1] 1841.078 3154.943
```

Для построения графика прогноза в виде гладкой линии по требуется большое число точек, которые удобнее задать в виде последовательности от 100 до 200 с шагом 5:

```
Age_forecast <- seq(100,200, by = 5)
ggplot() +
geom_point(aes(x=tree$Age, y=tree$Vol)) +
geom_line(aes(x=tree$Age, y=richards(tree$Age, myGA@solution))) +
geom_line(aes(x=Age_forecast,y=richards(Age_forecast,
myGA@solution), color = "maroon")) +
theme_bw(base_size = 18)
```

Результат показан на рисунке 18.

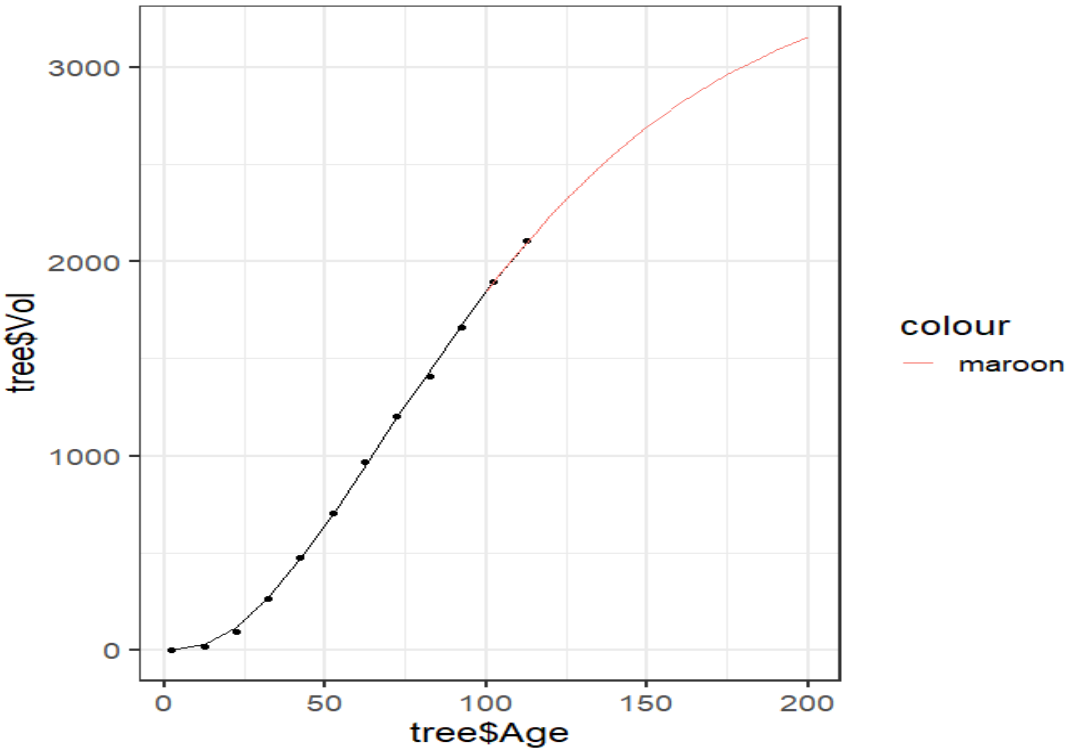


Рисунок 18. График исходных данных и полученной функции Ричардса с прогнозом от 100 до 200 лет

Генерация данных

Выполним генерацию значений *x* с нормальным законом распределения, математическим ожиданием 45 и

среднеквадратическим отклонением (СКО). Объем выборки зададим равным 100:

```
x <- rnorm(n = 100, mean = 45, sd = 7)
```

Для генерации *y* необходимо задать формулу зависимости, например, линейную, и наложить на нее случайную помеху ε:

```
a <- 20
b <- 5
y <- a + b*x
```

Генерацию ε также выполним по нормальному закону распре деления, с нулевым математическим ожиданием и единичной дисперсией. Однако, поскольку генераторы случайных чисел неидеальны, необходимо дополнительно выполнить нормировку и центрирование значений:

```
er <- rnorm(n = 100, mean = 0, sd = 1)
er <- (er- mean(er))/sd(er)
```

После этого можно наложить помеху на формулу зависимости с заданным СКО, равным 2:

```
Ser <- 5
y <- y + er*Ser
```

В результате получим зависимость, показанную на Рисунке 19.

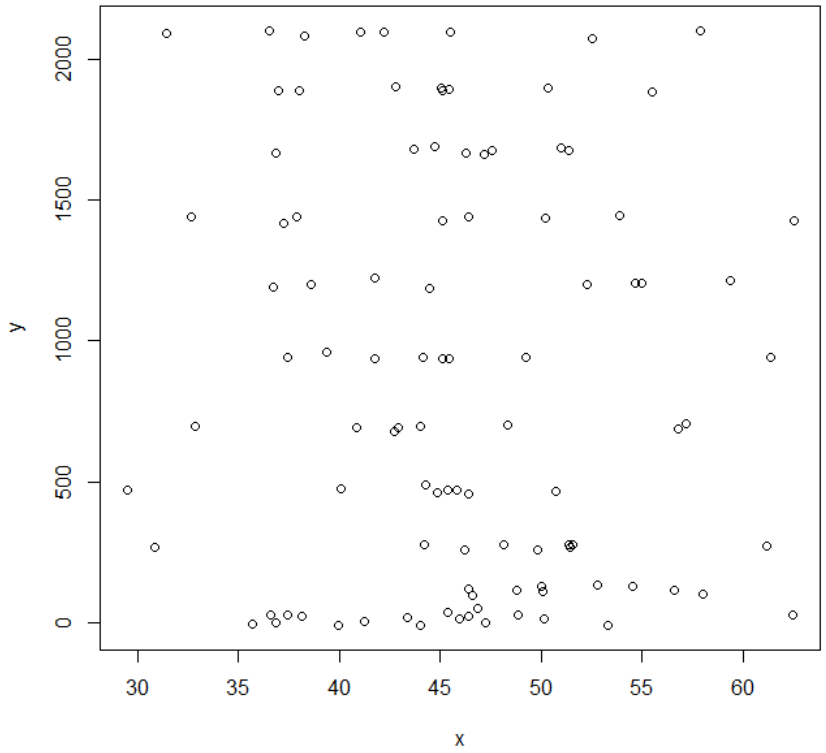


Рисунок 19. Сгенерированная зависимость *y* от *x*

Контрольные вопросы

1. Максимизирует или минимизирует генетический алгоритм в R fitness-функцию?
2. Как задать в R функцию
   $y = a(x - b)^3 + c$?
3. Каким образом в R можно сгенерировать выборку со стандартным нормальным законом распределения?

Задания

№ варианта	**1.**	**2.**	**3.**	**4.**	**5.**	**6.**	**7.**	**8.**	**9.**	**10.**
Среднее *x*	7	30	100	20	50	10	25	40	60	33
Ст. отклон. *x*	2	3	15	4	6	1	4	7	10	8
Объем выборки	50	100	150	70	50	30	60	90	110	80
Формула *y*	$y = a + b(x-c)^2 + \varepsilon$			$y = a + \frac{b}{x-c} + \varepsilon$		$y = a + e^{bx+c} + \varepsilon$		$y = \frac{a}{1+e^{-b(x-c)}} + \varepsilon$		
a	10	100	–5	10	150	20	1000	50	80	20
b	–2	5	–0.1	3	–10	0.8	–0.25	0.25	0.3	0.22
c	5	32	90	10	30	0.5	5	35	52	30
Ст. отклон. помехи	4	28	14	0.05	0.1	3000	0.25	3	6	1.5

ЛАБОРАТОРНАЯ РАБОТА № 5
Дискриминантный анализ в R

Дискриминантный анализ (ДА) - это метод многомерной классификации, который позволяет разделить объекты на две или более отдельные группы на основе измеренных характеристик этих объектов. Измеренные характеристики называют предикторами или независимыми переменными, тогда как классификационная группа – множество входящих в нее объектов.

В ДА используют данные, заранее известных классов, создают модель, которую далее можно использовать для классификации будущих наблюдений. Это полезный аналитического метод, в оценке и понимании взаимосвязи между независимыми переменными и дискретной зависимой переменной. Он отличается от регрессионного анализа тем, что зависимая переменная должна быть дискретной. А также отличается от кластерного анализа тем, что для создания модели необходимо заранее знать классы.

Обычно рассматривается два типа ДА:

1. Линейный дискриминантный анализ (LDA) предполагает, что ковариация независимых переменных одинакова для всех классов.
2. Квадратичный дискриминантный анализ (QDA) не предполагает равной ковариации между классами.

И LDA, и QDA требуют того, чтобы количество независимых переменных было меньше размера выборки, и оба они предполагают нормальность многомерного распределения среди независимых переменных. То есть независимые переменные соответствуют нормальному или гауссовскому распределению.

5.1. Данные и необходимые пакеты

Для демонстрации методов дискриминантного анализа используем два набора данных.

Первый набор данных находится в таблице mtcars. Эта таблица представлена в пакете datasets R. Данные были извлечены из американского журнала Motor Trend 1974 года и включают в себя показатель расхода топлива и 10 вариантов конструкции двигателя и трансмиссии 32 автомобилей (1973-74 модели).

Второй набор данных представлен таблицей измерений цветков ирисов (iris), который использовал Р. Фишера На этом наборе данных он в 1936 году продемонстрировал применение разработанного им

метода дискриминантного анализа. Набор данных стал классическим, и его часто используют в литературе для иллюстрации работы различных статистических алгоритмов.

Этот набор данных в разных вариантах доступен в программных пакетах R. Данные можно вывести консоль с помощью функции view(iris) просмотра данных в форме электронной таблицы. Набор данных iris включает измерения длины и ширины (в сантиметрах) чашелистиков и лепестков 150 цветков, которые можно разделить на три различных вида: ирис сетоза, ирис разноцветный и ирис вирджиния. Измерения длины и ширины - это независимые переменные, которые будут использоваться для прогнозирования вида цветка, дискретной зависимой переменной.

Линейный дискриминантный анализ – это метод классификации объектов на основании некоторых признаков (независимых переменных) и позволяет отнести некоторый объект к одной из двух или нескольких групп. Число групп определяется как число категорий зависимой переменной.

Дискриминантный анализ используется для анализа данных в том случае, когда зависимая переменная категориальная, а предикторы (независимые переменные) — интервальные. В результате дискриминантного анализа строится дискриминантная функция:

$$d = b_1x_1+b_2x_2+... + b_nx_n+a,$$

где x_1 и x_n – значения переменных, соответствующих рассматриваемым случаям, b_1-b_n и a – коэффициенты, которые получить с помощью применения процедуры дискриминантного анализа. Коэффициенты подбираются таким образом, чтобы по вычисленным значениям дискриминантной функции можно с максимальной точностью выполнить распределение по группам объектов классификации.

5.2. Процедура дискриминантного анализа

1. формулировка проблемы, определения целей, зависимой и независимых переменных. Выборку делят на две части. Анализируемую выборку используют для вычисления дискриминантной функции; проверочную – для проверки достоверности модели.

2. определение функции, включает выведение такой линейной комбинации предикторов (дискриминантных функций), чтобы группы максимально возможно различались между собой значениями предикторов.
3. определение статистической значимости. Включает проверку нулевой гипотезы о том, что в совокупности средние всех дискриминантных функций во всех группах равны между собой. Если нулевая гипотезу отклоняется, то можно перейти к интерпретации результата.
4. интерпретация дискриминантных весов или коэффициентов аналогична такой же стадии во множественном регрессионном анализе.
5. проверка достоверности включает вычисление классификационной матрицы. Дискриминантные веса, определенные с помощью анализируемой выборки, умножают на значения независимых переменных в проверочной выборке, чтобы получить дискриминантные показатели для объектов в этой выборке. Затем все объекты распределяют по группам, исходя из дискриминантных показателей и соответствующего правила принятия решения. Определяют процент верно классифицированных объектов и сравнивают его с процентом объектов, которые можно ожидать на основе классификации методом случайного выбора.

Для оценки коэффициентов существует два известных подхода. Прямой метод включает оценку дискриминантной функции при одновременном введении всех предикторов. Альтернативный ему пошаговый метод включает последовательное введение предсказанных переменных, исходя из их способности дискриминировать группы.

5.3. Предварительный анализ данных

Важный этап анализа данных – это предварительный их анализ, который заключается в выявлении характера связи между анализируемыми переменными. Обнаруженные на этом этапе закономерности определяют, например, выбор статистической

модели для описания данных, выбор модели данных, необходимость преобразования нелинейно связанных переменных, и т.д.

Характер связи между переменными проще всего выявить, визуализируя возможные зависимости используя соответствующие графические средства. Например, для анализа нескольких количественных переменных используются матричные диаграммы рассеяния или парные диаграммы рассеяния. В качестве примера рассмотрим данные представленные в таблице mtcars. Эта таблица представлена в пакете datasets R. Данные были извлечены из американского журнала Motor Trend 1974 года и включают в себя показатель расхода топлива и 10 вариантов конструкции двигателя и трансмиссии 32 автомобилей (1973-74 модели).

Таблица 5.1

mpg	Miles/(US) gallon	Мили/(американский) галлон
cyl	Number of cylinders	Количество цилиндров
disp	Displacement (cu.in.)	Рабочий объем в кубических дюймах (куб. дюйм)
hp	Gross horsepower	Мощность лошадиная сила
drat	Rear axle ratio	Передаточное отношение задней оси
wt	Weight (1000 lbs)	Вес (1000 фунтов)
qsec	1/4 mile time	Время прохождения 1/4 мили
vs	Engine (0 = V-shaped, 1 = straight)	Двигатель (0 = V-образный, 1 = прямой)
am	Transmission (0 = automatic, 1 = manual)	Коробка передач (0 = автоматическая, 1 = ручная)
gear	Number of forward gears	Количество передних передач

Функция str() представляет структуру таблицы данных mtcars

```
str(mtcars)
```

```
'data.frame': 32 obs. of 11 variables:
 $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num 6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num 160 160 108 258 360 ...
 $ hp : num 110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt : num 2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num 16.5 17 18.6 19.4 17 ...
 $ vs : num 0 0 1 1 0 1 0 1 1 1 ...
 $ am : num 1 1 1 0 0 0 0 0 0 0 ...
```

```
 $ gear: num 4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num 4 4 1 1 2 1 4 2 2 4 ...
```

Кроме того, можно просмотреть первые десять строк:

```
head(mtcars)
     mpg cyl disp hp drat wt qsec vs am gear carb
Mazda RX4   21.0 6 160 110 3.90 2.620 16.46 0 1 4 4
Mazda RX4 Wag  21.0 6 160 110 3.90 2.875 17.02 0 1 4 4
Datsun 710  22.8 4 108 93 3.85 2.320 18.61 1 1 4 1
Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1
Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2
Valiant   18.1 6 225 105 2.76 3.460 20.22 1 0 3 1
```

Матричные диаграммы рассеяния в R можно построить при помощи нескольких функций. Проще всего воспользоваться функцией pairs(), входящей в базовую версию R:

```
pairs(mtcars)
```

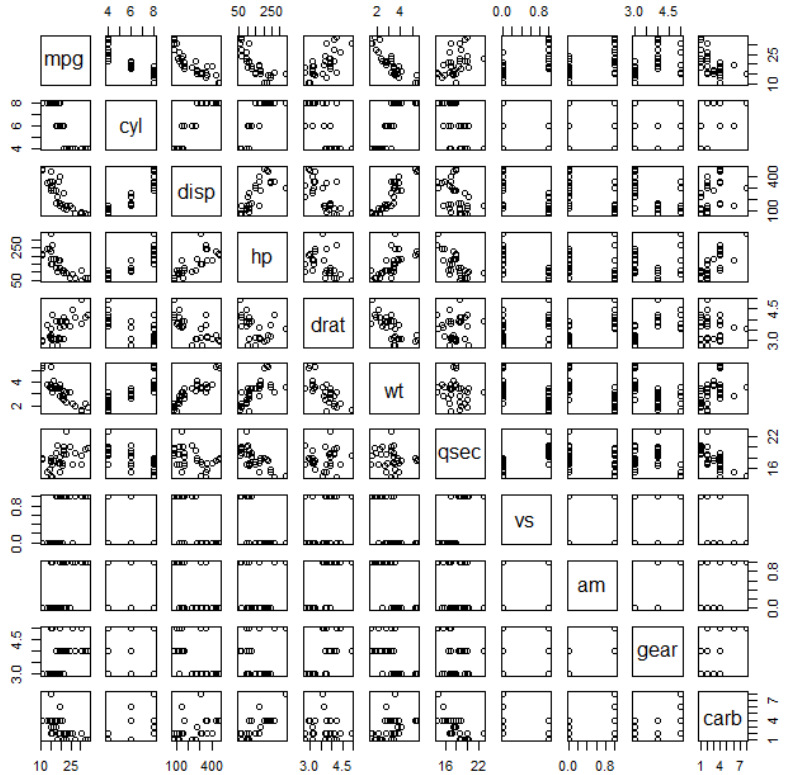


Рисунок 5.1. Матричная диаграмма рассеяния визуализация функцией pairs(mtcars)

Функция pairs() имеет ряд аргументов для тонкой настройки графика. Например, для облегчения интерпретации характера связи между анализируемыми переменными можно добавить

сглаживающую кривую к каждой диаграмме рассеяния (аргумент panel со значением panel.smooth):

```
pairs(mtcars, panel = panel.smooth)
```

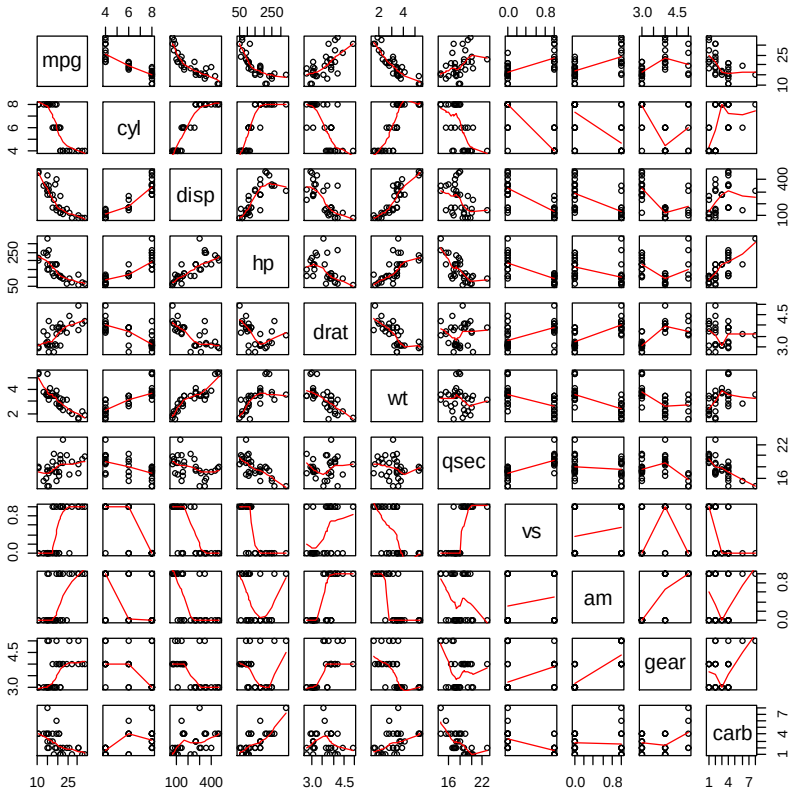


Рисунок 5.2. Матричная диаграмма рассеяния визуализация функцией pairs(mtcars) с добавлением сглаживающей кривой

Аргументы lower.panel и upper.panel позволяют назначить практически любую функцию для преобразования исходных данных и последующего отображения результатов работы этой функции на графике (ниже и выше центральной диагонали матрицы соответственно). Например, можно добавить к графику значения коэффициентов корреляции Спирмена:

```
# Функция для расчета коэффициентов корреляции
panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
{ usr <- par("usr"); on.exit(par(usr))
 par(usr = c(0, 1, 0, 1))
 r <- abs(cor(x, y, method = "spearman"))
 txt <- format(c(r, 0.123456789), digits=digits)[1]
 txt <- paste(prefix, txt, sep = "")
 # эта команда позволяет изменять размер шрифта в соответствии
 # со значением коэффициента корреляции:
 if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
 text(0.5, 0.5, txt, cex = cex.cor * r)}
# Строим график:
```

```
pairs(mtcars, lower.panel = panel.cor)
```

Рисунок 5.3. Матричная диаграмма рассеяния визуализация функцией pairs(mtcars) с добавлением значений коэффициентов корреляции Спирмена

Функция splom() пакета lattice также позволяет строить матричные диаграммы рассеяния:

```
library(lattice)
splom(mtcars)
```

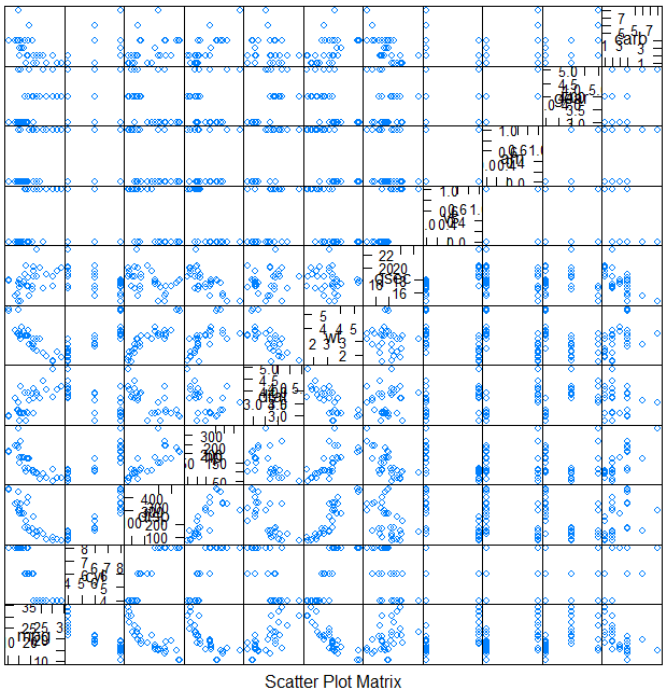


Рисунок 5.4. Матричная диаграммы рассеяния визуализированная функцией

splom(mtcars)

Следует представить функцию ggpairs() из пакета GGally, являющегося дополнением к пакету ggplot2. Она позволяет строить простые матричные диаграммы рассеяния

```
ggpairs(mtcars)
```

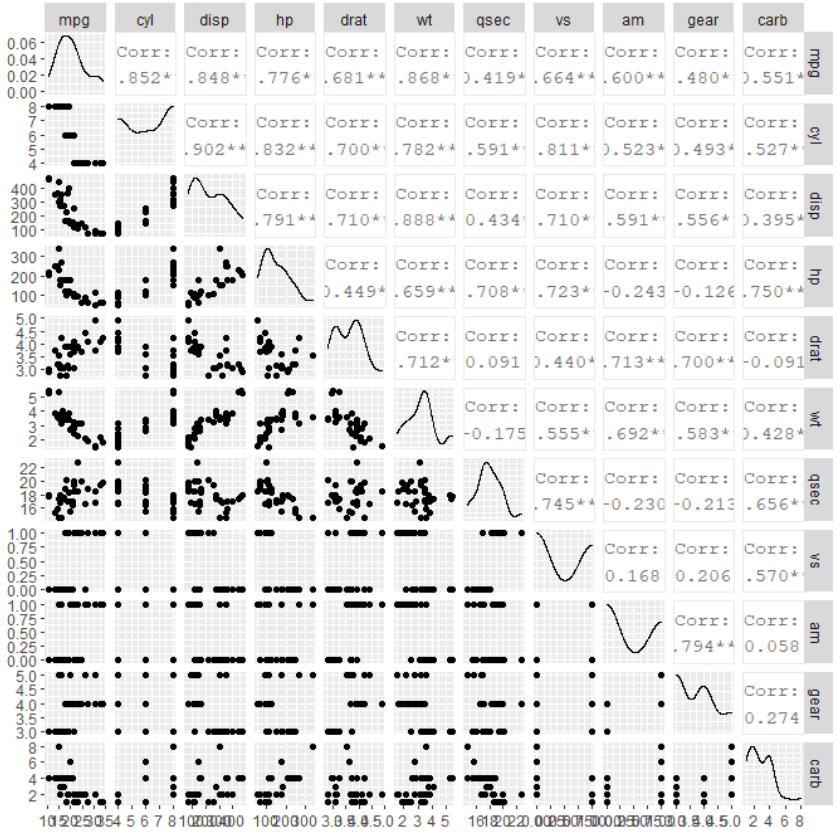


Рисунок 5.5. Матричная диаграмма рассеяния визуализированная функцией ggpairs()

Визуализация множества данных для последующего построения дискриминанты

```
library(graphics)
pairs(mtcars, main = "mtcars data", gap = 1/4)
coplot(mpg ~ disp | as.factor(cyl), data = mtcars,
  panel = panel.smooth, rows = 1)
## possibly more meaningful, e.g., for summary() or bivariate plots:
mtcars2 <- within (mtcars, {
 vs <- factor(vs, labels = c("V", "S"))
 am <- factor(am, labels = c("automatic", "manual"))
 cyl <- ordered(cyl)
 gear <- ordered(gear)
 carb <- ordered(carb)
})
summary(mtcars2)
```

Функция coplot() визуализирует анализируемые данные, разбиваются их на отдельные категории, в соответствии с уровнями какого-то фактора, для каждого из которых строится свой график (= панель) определенного типа. Графики объединяются на одном рисунке, что существенно облегчает выявление статистических закономерностей и структур в анализируемых данных. На Рисунке 3.6. показана зависимость переменной mpg и disp данных matcars по факторам переменной cyl – количество цилиндров.

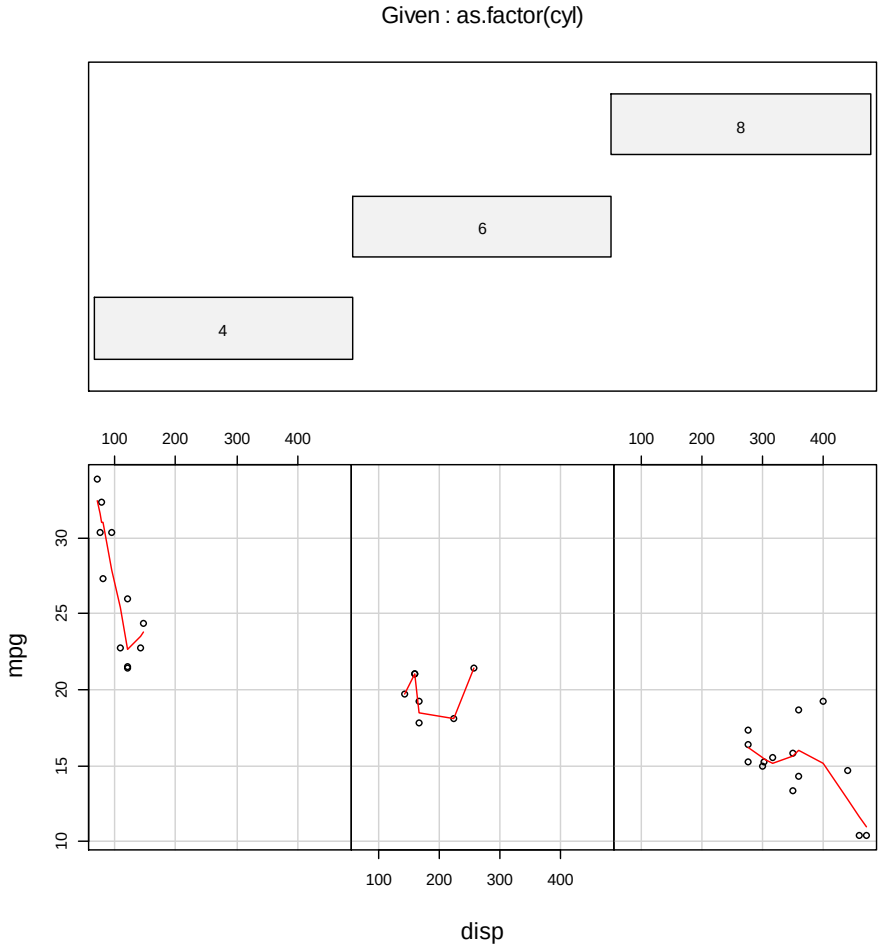


Рисунок 5.6 Зависимость переменной mpg и disp данных matcars по факторам переменной cyl – количество цилиндров.

```
summary(mtcars2)
      mpg        cyl         disp             hp             drat
 Min.   :10.40   4:11   Min.   : 71.1   Min.   : 52.0   Min.   :2.760
 1st Qu.:15.43   6: 7   1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080
 Median :19.20   8:14   Median :196.3   Median :123.0   Median :3.695
 Mean   :20.09          Mean   :230.7   Mean   :146.7   Mean   :3.597
 3rd Qu.:22.80          3rd Qu.:326.0   3rd Qu.:180.0   3rd Qu.:3.920
 Max.   :33.90          Max.   :472.0   Max.   :335.0   Max.   :4.930

       wt             qsec       vs              am     gear   carb
 Min.   :1.513   Min.   :14.50   V:18   automatic:19   3:15   1: 7
 1st Qu.:2.581   1st Qu.:16.89   S:14   manual   :13   4:12   2:10
 Median :3.325   Median :17.71                         5: 5   3: 3
```

```
 Mean   :3.217   Mean   :17.85                                  4:10
 3rd Qu.:3.610   3rd Qu.:18.90                                  6: 1
 Max.   :5.424   Max.   :22.90                                  8: 1
```

5.5. Дискриминантный анализ на примере данных iris

Шаг 1. Необходимые пакеты.

Необходимо загрузить следующие пакеты:

- tidyverse - обработка и визуализация данных
- MASS - функци LDA и QDA
- klaR - функции графика распределения

```
library(tidyverse)
library(MASS)
library(klaR)
set.seed(101)
sample_n(iris, 10)
##  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 56   5.7   2.8   4.5   1.3 versicolor
## 7   4.6   3.4   1.4   0.3  setosa
## 106   7.6   3.0   6.6   2.1 virginica
## 97   5.7   2.9   4.2   1.3 versicolor
## 37   5.5   3.5   1.3   0.2  setosa
## 44   5.0   3.5   1.6   0.6  setosa
## 85   5.4   3.0   4.5   1.5 versicolor
## 48   4.6   3.2   1.4   0.2  setosa
## 89   5.6   3.0   4.1   1.3 versicolor
## 77   6.8   2.8   4.8   1.4 versicolor
```

Шаг 2. Подготовка данных

После того, что набор данных доступен и пакеты загружены, необходимо сделать следующий шаг - подготовить данные для анализа. То есть разделить данные на два подмножества: набор для обучения и набор для тестирования. Обучающий набор используем для построения прогнозной модели, а затем проверим точность классификации на наборе для тестирования с тем, чтобы оценить точность модели. Разделим общий набор, используя 60% данных в качестве обучающего набора (train), а оставшиеся 40% - для набора тестирования (test).

```
training_sample <- sample(c(TRUE, FALSE), nrow(iris), replace = T, prob =
c(0.6,0.4))
train <- iris[training_sample, ]
test <- iris[!training_sample, ]
```

5.5.1. Линейный дискриминантный анализ (LDA)

LDA ищет линейные комбинации независимых переменных, которые позволяют наилучшим образом объяснить данные и

спрогнозировать принадлежность объекта к определенному классу. На основе результатов наблюдений рассчитываются количественные показатели дискриминатора для каждого класса. Вычисленные коэффициенты линейных комбинаций позволяют получить числовое значение для оценки принадлежности к классу с использованием следующего уравнения:

$$\delta_i(X) = -\frac{1}{2}\mu_i^T \Sigma^{-1} \mu_i + \mu_i^T \Sigma^{-1} X + \ln(\pi_i)$$

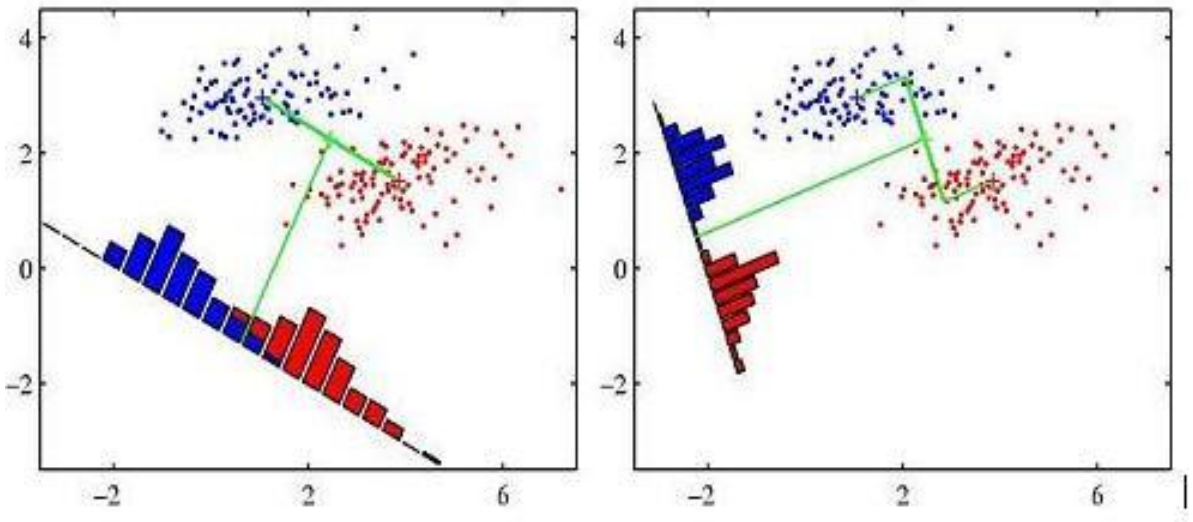
δi(X)=−12μiTΣ−1μi+μiTΣ−1X+ln(πi)

где

- δ_i дискриминантная оценка для класса i
- X матрица независимых переменных
- μ_i вектор, содержащий средние значения каждой переменной для класса i
- Σ ковариационная матрица переменных (предполагается, что она одинакова для всех классов)
- π_i априорная вероятность того, что наблюдение принадлежит классу i

Каждое наблюдение получает определенное числовое значение для каждого класса. Класс с наибольшей оценкой будет основанием классификации этого наблюдения.

На Рисунке 3.1. показано, как происходит классификация с использованием LDA. На левой части Рисунка 3.1.а показаны данные так, как если бы они были спроецированы на линейную комбинацию переменных или линейный дискриминант в одномерном пространстве. LDA находит линейные дискриминанты там, где наблюдается наибольшее различие между классами. Эта линия перпендикулярна гистограммам. На Рисунке 3.1.б линия разделяет пространство между двумя классами. Предполагается, что наблюдения на одной стороне линии будут относиться к одному классу, а наблюдения на другой стороне - к другому классу. Некоторые результаты классификации могут быть неправильными. Чем больше совпадений между классами, тем выше будет число ошибок.



а б

Рисунок 5.5.1. Классификация с использованием LDA

Используем набор обучающих данных для создания модели классификации. Функция lda() пакета MASS выполняет значительную часть работы по обработке данных. Представленный ниже код позволяет получить необходимые для анализа показатели.

```
lda.iris <- lda(Species ~ ., train)
lda.iris #show results

## Call:
## lda(Species ~ ., data = train)
##
## Prior probabilities of groups:
##  setosa versicolor virginica
## 0.3333333 0.2857143 0.3809524
##
## Group means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa   5.046429 3.403571  1.478571 0.2607143
## versicolor  5.933333 2.770833  4.295833 1.3125000
## virginica  6.603125 2.984375  5.509375 2.0437500
##
## Coefficients of linear discriminants:
##      LD1  LD2
## Sepal.Length 0.5081196 -0.925690
## Sepal.Width 1.9631006 -1.206187
## Petal.Length -2.1214689 1.868896
## Petal.Width -2.9806917 -3.850355
##
## Proportion of trace:
## LD1 LD2
## 0.9901 0.0099
```

Априорные вероятности групп показывают π_i, вероятность случайного выбора наблюдения из класса i из общего обучающего

набора. Поскольку в исходном наборе данных имеется по 50 наблюдений каждого вида (всего 150 наблюдений), тогда априорные вероятности должны быть близки к 33,3% для каждого класса.

Среднее значение группы показывает μ_i, среднее значение для каждой из независимых переменных для каждого класса i.

Коэффициенты линейных комбинаций - это коэффициенты для каждого дискриминанта. Например, первый линейный дискриминант (LD1) - это линейная комбинация:

LD1 = 0.51*Sepal.Length + 1.96*Sepal.Width – 2.12*Petal.Length – 2.98*Petal.Width

Показатель (Proportion of trace) описывают пропорцию межклассовой дисперсии, которая объясняется последовательными дискриминантными функциями. Как можно видеть, LD1 объясняет 99% дисперсии.

Визуализация группирования наблюдений с помощью базовой функции plot().

```
plot(lda.iris, col = as.integer(train$Species))
```

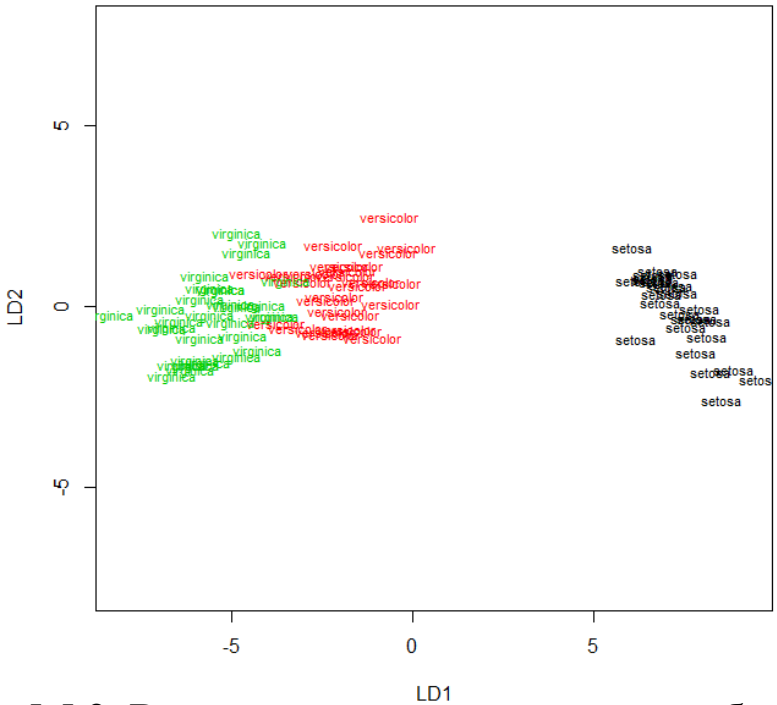


Рисунок 5.5.2. Визуализация группирования наблюдений.

Существует три отдельные группы, которые частично пересекаются между Virginica и Versicolor. Добавление кода dimen =

1 в построении графика приведет только к одному измерению (LD1). Это можно представить, как о проекцию данных на LD1 в форме гистограммы этих данных.

```
plot(lda.iris, dimen = 1, type = "b")
```

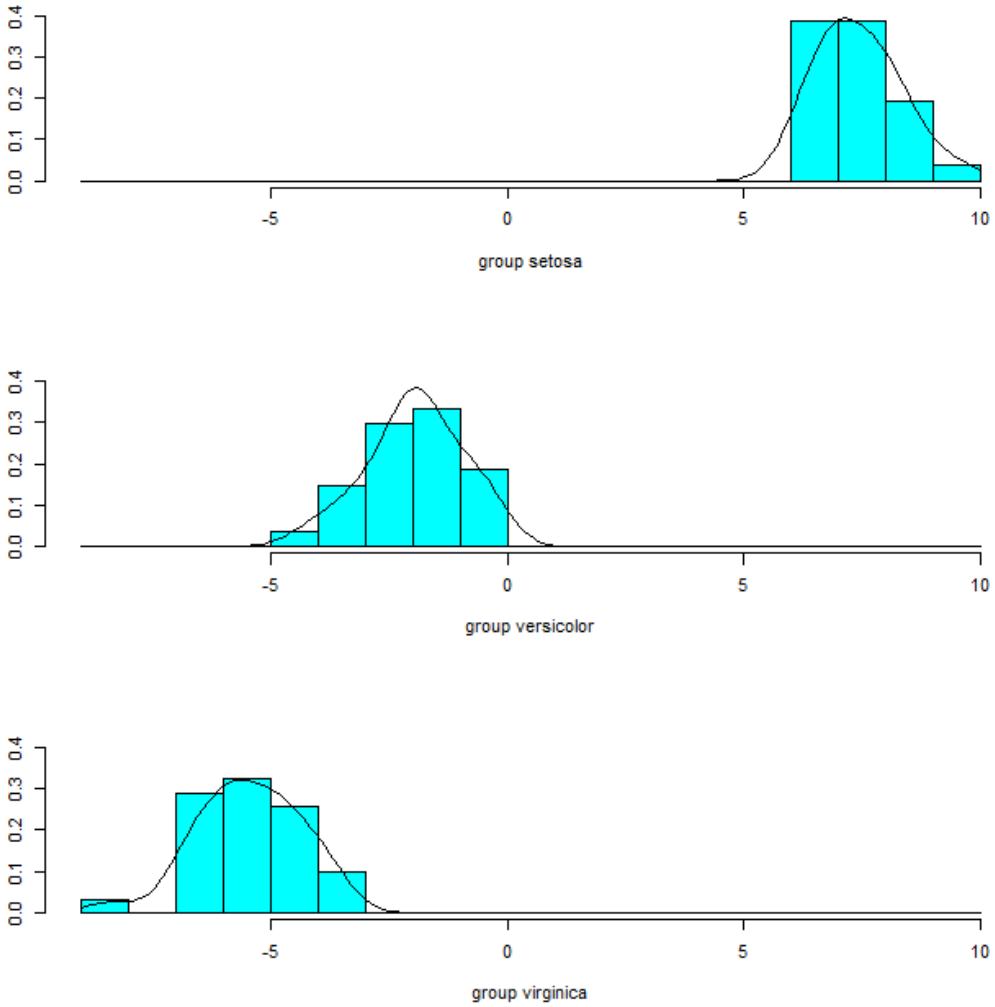


Рисунок 5.5.3. Графики разделения и области перекрытия между группами Setosa, Versicolor, Virginica.

На Рисунке 3.3. представлены графики иллюстрируют разделение между группами, а также области перекрытия, которые могут вносить ошибку при определении классов.

5.5.2. Графики разбиения LDA

Использование функции partimat() из пакета klaR предоставляет альтернативный способ построения линейных дискриминантных

функций. Функция partimat() выводит массив графиков для каждой комбинации двух переменных. Каждый график дает иное представление об одних и тех же данных. Окрашенные области выделяют каждую область классификации. Предполагается, что любое наблюдение, относящееся к определенной области, относится к определенному классу. Каждый график также включает явную частоту ошибок для этого представления данных.

```
partimat(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
data=train, method="lda")
```

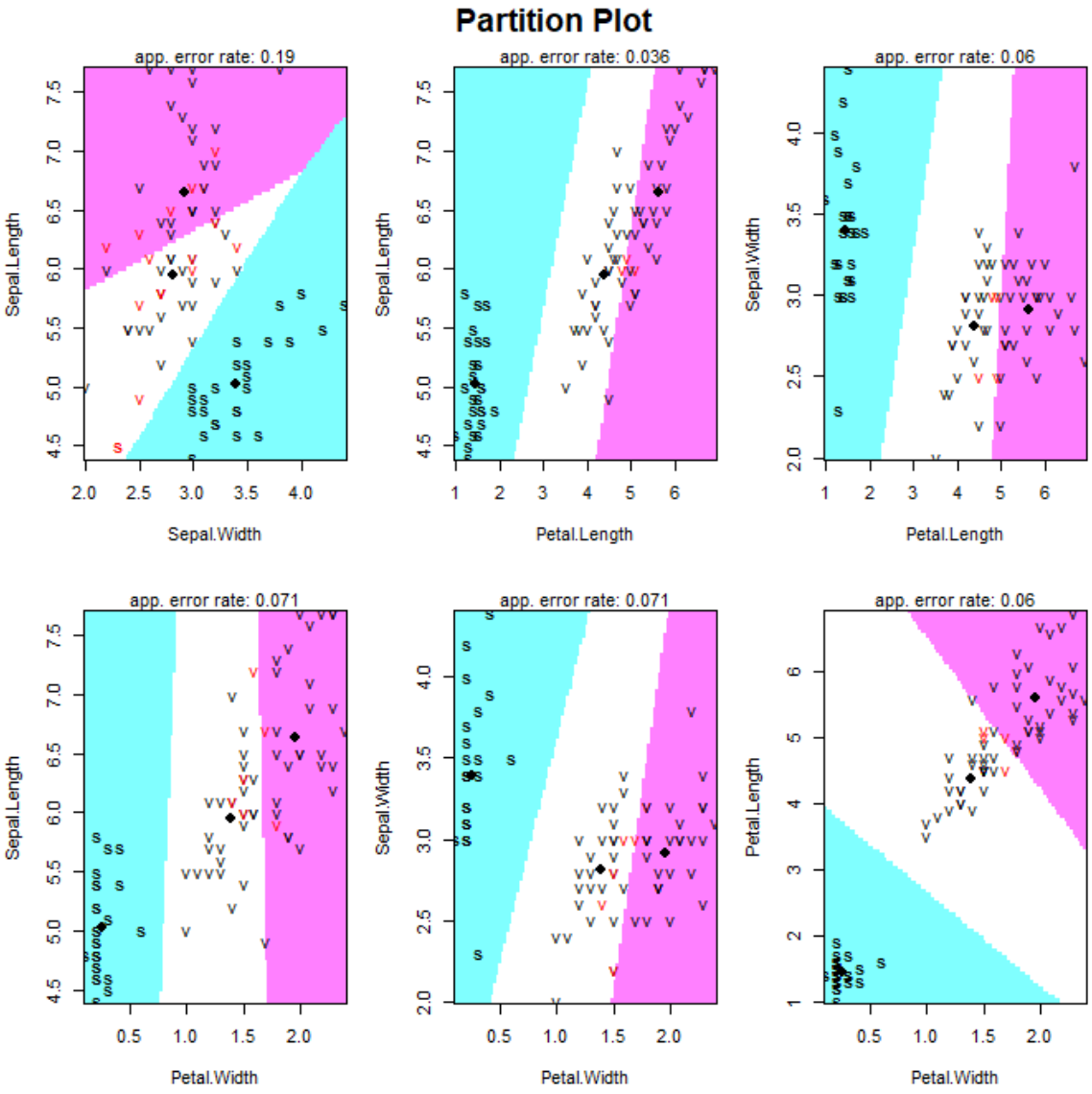


Рисунок 5.5.4. Массив графиков для каждой комбинации двух переменных.

5.5.3. Оценка точности классификации LDA

Оценим точность полученной модели классификации. Сначала запустим модель с обучающим набором, используемым для проверки

правильности соответствия модели данным с помощью функции predict(). Результатом является матрица с количеством наблюдений ошибочной классификации для классов, обозначенных в названиях строк и столбцов.

```
lda.train <- predict(lda.iris)
train$lda <- lda.train$class
table(train$lda,train$Species)
##
##     setosa versicolor virginica
## setosa     28    0    0
## versicolor  0    22    1
## virginica  0    2  31
```

Общее количество правильно предсказанных наблюдений представлено на диагонали. Таким образом, эта модель правильно соответствует обучающим данным почти для каждого наблюдения. Проверка обучающего набора не доказывает точность, но плохое соответствие обучающим данным может быть признаком того, что модель не является хорошей.

Запустим тестовый набор этой модели для того, чтобы определить ее точность.

```
lda.test <- predict(lda.iris,test)
test$lda <- lda.test$class
table(test$lda,test$Species)
##
##     setosa versicolor virginica
## setosa     22    0    0
## versicolor  0    25    0
## virginica  0    1  18
```

Основываясь на более ранних графиках, можно оценить ошибочную классификацию нескольких наблюдений iris versicolor и iris virginica. В целом модель очень хорошо работает с тестовым набором с точностью 98,5%.

5.5.4. Квадратичный дискриминантный анализ (QDA)

QDA, как и LDA, предполагает, что независимые переменные имеют нормальное распределение (многомерная нормальность). В отличие от LDA, QDA не предполагает равной ковариации между классами. QDA не находит линейных комбинаций независимых переменных, но находит квадратичную функцию независимых

переменных. Ниже приведено уравнение для расчета дискриминантных оценок:

$$\delta_i(X) = -1/2 * \ln(|\Sigma_i|) - 1/2 * (X-\mu)^T * \Sigma^{-1}_i * (X-\mu) + \ln(\pi_i)$$

где

- δ_i дискриминантная оценка для класса i
- X матрица независимых переменных
- μ вектор, содержащий средние значения каждой переменной
- Σ ковариационная матрица переменных (предполагается, что она одинакова для всех классов)
- π_i априорная вероятность того, что наблюдение принадлежит классу i

Подробное объяснение этого уравнения можно найти в []. Как и LDA, класс с наибольшим уровнем дискриминации определит предсказание конкретного наблюдения.

Используется функция qda из пакета MASS. Выходные данные очень похожи на выходные данные LDA с теми же априорными вероятностями и средними значениями класса, но нет линейных дискриминантов, рассчитанных с помощью QDA.

```
qda.iris <- qda(Species ~ Sepal.Length + Sepal.Width + Petal.Length +
Petal.Width, train)
qda.iris #show results
## Call:
## qda(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
##  data = train)
##
## Prior probabilities of groups:
##  setosa versicolor virginica
## 0.3333333 0.2857143 0.3809524
##
## Group means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa   5.046429 3.403571  1.478571 0.2607143
## versicolor  5.933333 2.770833  4.295833 1.3125000
## virginica  6.603125 2.984375  5.509375 2.0437500
```

5.5.5. Визуализации результатов дискриминанты QDA

Использование функции partimat() позволяет построить квадратичные дискриминантные функции. Единственное отличие кода от приведенного выше раздела LDA - это замена method="lda" на method="qda". Графики хорошо визуализируют различие между линейными функциями LDA, и квадратичными функциями QDA. Цветные области выделяют каждую область классификации. Предполагается, что любое наблюдение, относящееся к определенной области, относится и к определенному классу. Каждый график также включает явную частоту ошибок для этого представления данных.

```
partimat(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
data=train, method="qda")
```

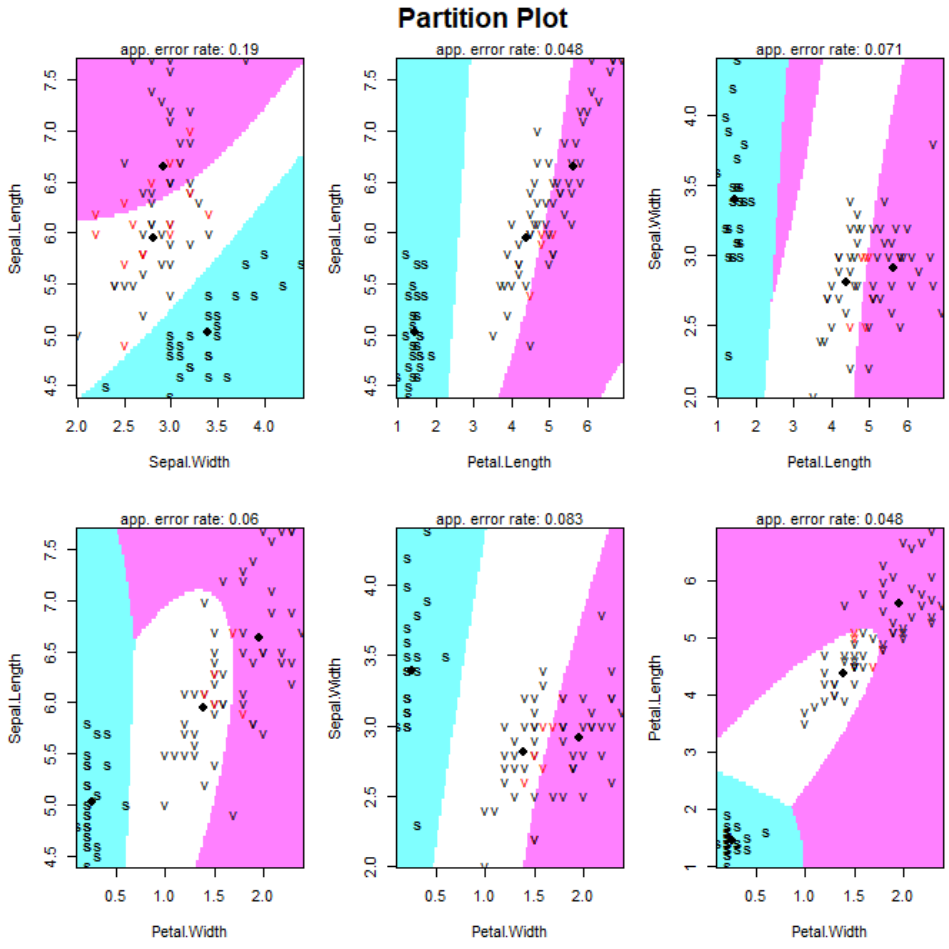


Рисунок 5.5.5. Визуализация квадратичных дискриминантных функций.

Функция predict() работает точно так же, как и раньше, и может быть использована для создания матрицы ошибок обучающих данных.

```
qda.train <- predict(qda.iris)
train$qda <- qda.train$class
table(train$qda,train$Species)
##
##    setosa versicolor virginica
## setosa   28   0   0
## versicolor  0   23   1
## virginica  0   1  31
```

Эта модель очень хорошо соответствует обучающим данным. Хотя, проверка обучающего набора не определяет точность классификации, но плохое соответствие может быть признаком того, что модель не является хорошей. Сгенерируем матрицу ошибок с тестовыми данными.

```
qda.test <- predict(qda.iris,test)
test$qda <- qda.test$class
table(test$qda,test$Species)
##
##    setosa versicolor virginica
## setosa   22   0   0
## versicolor  0   24   0
## virginica  0   2  18
```

Модель правильно предсказала класс наблюдений в 97% случаев. Это немного менее точно, чем модель LDA, но это не всегда так.

Вопросы для самоконтроля

1. Как задаются данные для выполнения дискриминантного анализа в R?
2. Как загрузить пакет для дискриминантного анализа и как он называется?
3. Как получить коэффициенты дискриминантного уравнения?
4. Как получить информацию о числе корректно классифицированных случаев?
5. Какая переменная меньше всего влияет на LD1? Подсказка: какая переменная имеет наименьший коэффициент LD1?
6. Попробуйте удалить эту переменную и перезапустить модель. Подсказка: попробуйте заменить ~ . в строке кода lda.iris <- lda(Species ~ ., train) что-то вроде ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width
7. Какова точность этой новой модели?
8. Продолжайте удалять переменные, пока не останется только одна. Насколько точен LDA с одной переменной?

9. Используя метод дискриминантного анализа, сделайте прогноз значений переменной отклика для нескольких объектов обучающей выборки.
10. Оцените качество прогноза.

Задания

Используйте данные из таблиц для проведения линейного и квадратичного дискриминантного анализа:

№ варианта	**1.**	**2.**	**3.**	**4.**	**5.**	**6.**	**7.**	**8.**	**9.**	**10.**
Набор данных	CO2		ChickWeight		iris	Orange	airquality	faithful		
Имя переменной (вектора)	conc	uptake	weight	Time	Sepal.Length	circumference	Wind	Temp	eruptions	waiting

ЛАБОРАТОРНАЯ РАБОТА № 6
Кластерный анализ. Кластеризация k-средних с использованием R

6.1. Кластерный анализ.

Кластерный анализ используется для «интеллектуальной» группировки, классификации результатов исследований. Кластерный анализ (КА) решает задачу разбиения полученной выборки объектов (ситуаций) на подмножества кластеры, таким образом, чтобы каждый кластер состоял из сходных объектов, а объекты разных кластеров существенно отличались. Задача кластеризации относится к статистической обработке. Поскольку КА —это многомерная статистическая процедура, которая выполняет сбор данных, содержащих информацию о выборке объектов, и затем упорядочивающая объекты в сравнительно однородные группы (кластеры). Кластер - группа элементов, характеризуемых общим свойством, главная цель кластерного анализа - нахождение групп схожих объектов в выборке. Спектр применений кластерного анализа очень широк: археология, медицина, психология, социология, химия, биология и так далее.

Кластерный анализ выполняет следующие основные задачи:

- Разработка типологии или классификации.
- Исследование полезных концептуальных схем группирования объектов.
- Порождение гипотез на основе исследования данных.
- Проверка гипотез или исследования для определения, действительно ли типы (группы), выделенные тем или иным способом, присутствуют в имеющихся данных.

Независимо от предмета исследования применение кластерного анализа предполагает следующие этапы:

- Отбор выборки для кластеризации.
- Определение множества переменных, по которым будут оцениваться объекты в выборке.
- Вычисление значений той или иной меры сходства между объектами. Применение метода кластерного анализа для создания

групп сходных объектов. Проверка достоверности результатов кластерного решения.

Кластеризация K-средних - это хорошо известный метод обучения без учителя. Как следует из названия, он формирует кластеры «K» по данным, используя среднее значение данных. Алгоритмы без учителя - это класс алгоритмов, к которым следует подходить осторожно. Использование неправильного алгоритма приведет к совершенно неудачным результатам, и все усилия пойдут насмарку. В отличие от алгоритмов контролируемого обучения, где все еще можно обойтись сохранением некоторых частей в виде неизвестного черного ящика, важно использовать технику, когда, начиная с допущений, сделанных для реализации процесса, методы оптимизации и реализации.

6.2. Гипотезы и процесс

Гипотезы помогают упростить проблему, а затем упрощенная проблема может быть точно решена. Чтобы разделить набор данных на кластеры, необходимо определить критерии кластера и на их основе выбирать метод. Метод кластеризации K-средних основан на двух допущениях относительно структуры кластеров:

- кластеры имеют центроидную форму в пространстве признаков
- кластеры имеют близкие размеры.

Предположение о центроидности помогает разделить кластеры, когда алгоритм работает с данными и формирует кластеры. Если это предположение нарушается, образующиеся кластеры могут не соответствовать ожиданиям. С другой стороны, предположение о размере кластеров помогает определить границы кластера. Это предположение помогает вычислить количество точек данных, которые должен иметь каждый кластер. Это предположение также дает некоторое преимущество. Кластеры в методе K-средних определяются как среднее значение всех точек данных в кластере. Исходя из этого предположения, можно начинать с центров кластеров с какого угодно первоначального расположения центров. Выбор начальных точек кластеров в любом месте не влияет на сходимость

алгоритма к тем же конечным кластерам, поскольку центры будут находиться как можно дальше друг от друга.

Алгоритм работает следующим образом. На первом шаге - назначаются начальные кластеры. Можно указать любые K кластеров или позволить алгоритму назначать их случайным образом. Алгоритм работает так, что на каждой итерации все точки данных назначаются одному из кластеров на основе ближайшего расстояния от центра. После того, как все точки назначены одному из кластеров, центры кластеров обновляются. Новые центры определяются на основе среднего значения центроидов всех точек в кластере. Это повторяется, итерация за итерацией, пока не происходит никаких изменений в назначении кластера любой из точек данных. Есть много вариантов вычислений центров кластеров. Например, следует решить, как определяется расстояние каждой точки данных по отношению к центру кластера. Обычно используется евклидово расстояние. Алгоритм прост и понятен, но использовать, как оказывается не так просто, как может показаться.

6.3. Алгоритм k - средних

Рассмотрим пример работы алгоритма K-средних. R включает набор данных о времени ожидания между извержениями и продолжительностью извержения гейзера Old Faithful в Йеллоустонском национальном парке, представленные в таблице «faithful» пакета datasets. Набор данных состоит из 272 наблюдений 2 функций - eruptions (извержение), waiting (ожидание).

```
#Просмотр набора данных faithful
> head(faithful)
 eruptions waiting
1  3.600  79
2  1.800  54
3  3.333  74
4  2.283  62
5  4.533  85
6  2.883  55
```

```
> plot(faithful)
```

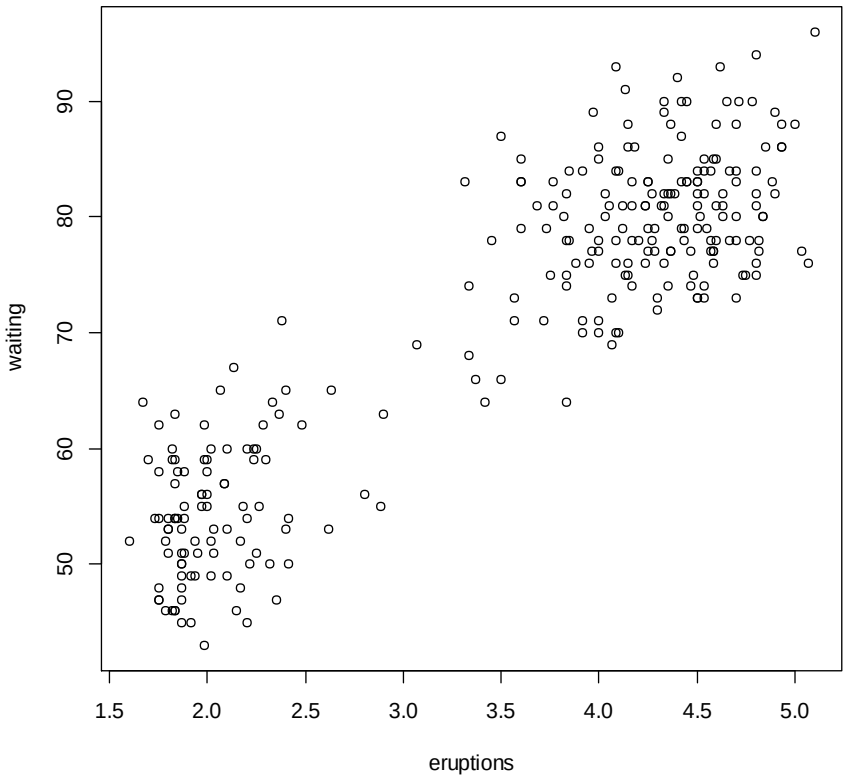


Рисунок 6.1. Данные времени ожидания между извержениями и продолжительностью извержения гейзера Old Faithful

Глядя на набор данных, можем заметить два кластера. Воспользуемся функцией kmeans() в R для формирования кластеров. Посмотрим, как алгоритм кластеризации K-средних работает с данными.

```
#Укажем 2 центра
k_clust_start=kmeans(faithful, centers=2)

#Напечатаем данные используя клатеры
plot(faithful, col=k_clust_start$cluster,pch=2)
```

Так это достаточно небольшой набор данных, кластеры формируются практически сразу. Для того, чтобы увидеть кластеры, их центры или размеры используют переменная k_clust_start, которая содержит информацию об обоих центрах и размере кластеров.

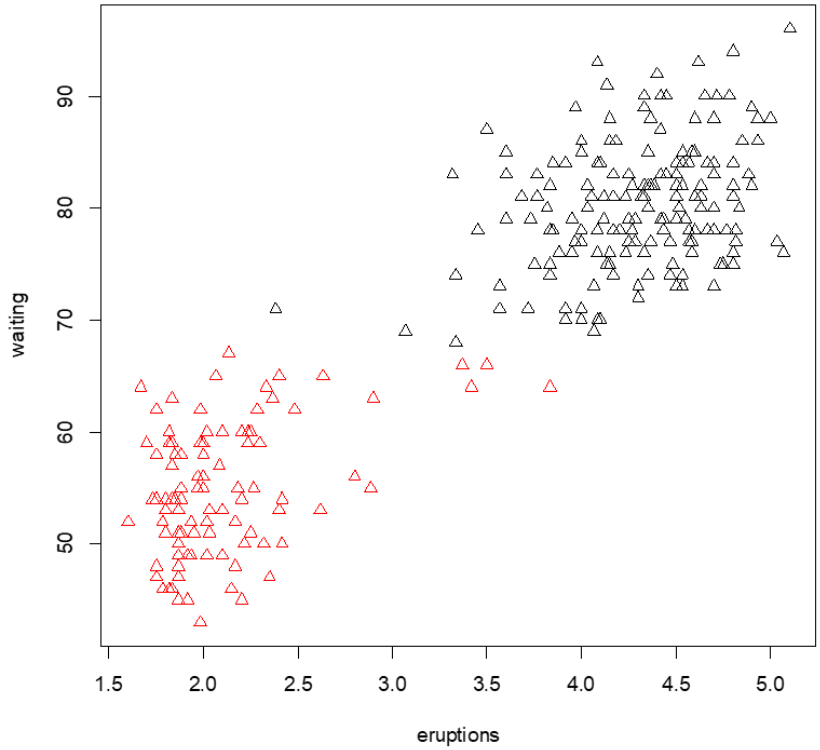


Рисунок 6.2. Распределение по кластерам данных времени ожидания между извержениями и продолжительностью извержения гейзера Old Faithful

Результат распределения по кластерам данных времени ожидания между извержениями и продолжительностью извержения гейзера Old Faithful представлены на Рисунке 2.

```
> # Используйте центры, чтобы найти центры кластеров
> k_clust_start$centers
 eruptions waiting
1 4.29793 80.28488
2 2.09433 54.75000
>
> #Используйте размер, чтобы найти размеры кластера
> k_clust_start$size
[1] 172 100
```

Первый кластер состоит из 172 членов и сосредоточен на значении извержений 4,29793 и значении ожидания 80,28488. Второй кластер состоит из 100 членов со значением извержений 2,09433 и значением ожидания 54,75. Таким образом, извержения обычно происходят либо в течение ~ 2 минут, либо ~ 4,3 минуты. При более длительных извержениях время ожидания также увеличивается.

6.4. Детальный анализ работы алгоритма кластеризации

Пусть в некотором наборе данных есть кластеры, которые можно четко идентифицировать, но не как k-средние. То есть, набор данных, который не удовлетворяет предположениям о распределении по центроидам. Например, набор данных, который представляет собой два концентрических круга.

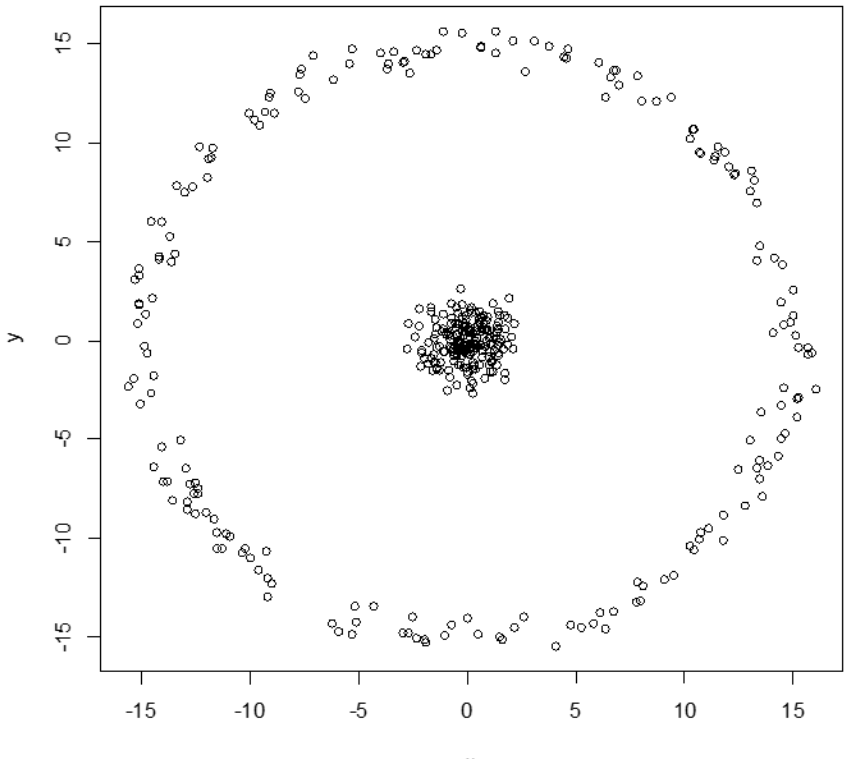


Рисунок 6.3. Набор данных представленных кластерами из концентрических кругов

Сгенерируем такой набор данных.

```
> #Код сгенерирует разные, но похожие графики
> library(plyr)
> library(dplyr)
> #Генерация случайных данных, которые будут первым кластером
> clust1 = data_frame(x = rnorm(200), y = rnorm(200))
> #Создайте второй кластер, который будет "окружать" первый кластер
> clust2 =data_frame(r = rnorm(200, 15, .5), theta = runif(200, 0, 2 * pi),
+     x = r * cos(theta), y = r * sin(theta)) %>%
+ dplyr::select(x, y)
> #Объедините данные
> dataset_cir= rbind(clust1, clust2)
> #Визоализируем данные
> plot(dataset_cir)
```

Таким образом, есть два кластера - один посередине, а другой вокруг первого. Это опровергает предположение о том, что кластеры имеют сферическую форму. Внутренние данные имеют сферическую форму, а внешний круг - нет. Тем не менее, рассмотрим, как k-means работает с этими данными.

```
> #Соответствие модели k-средних
> k_clust_spher1=kmeans(dataset_cir, centers=2)
> #Визуализация данных и кластеров
> plot(dataset_cir, col= c("black"),pch=c(2,15), cex=1.3)
```

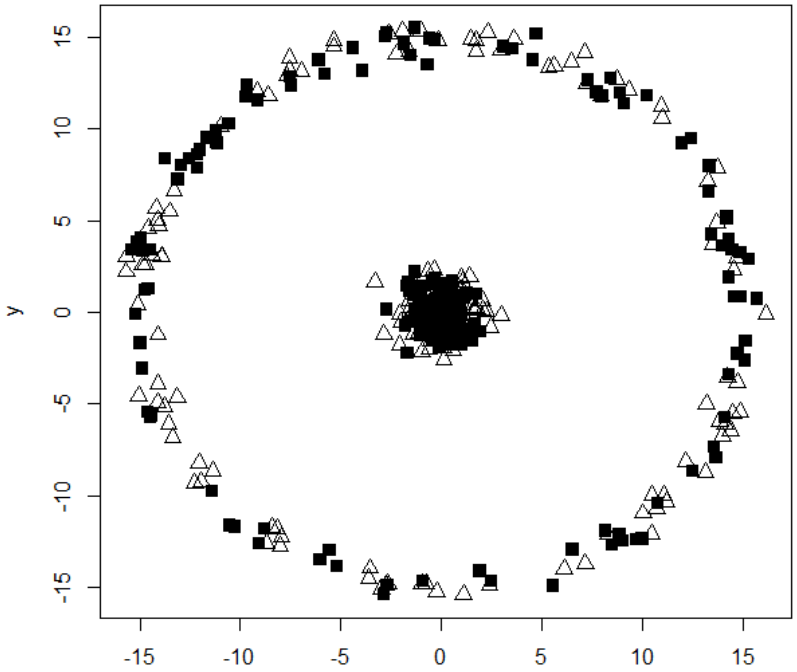


Рисунок 6.4. Результат кластеризации с помощью алгоритма k - средних кластеров представленных концентрическими кругами.

Таким образом есть 2 кластера, но алгоритм k-среднее не работает. Самый простой способ в этом случае - преобразовать данные в полярный формат. Преобразуем их и визуализируем их.

```
> # Использование функции для преобразования
> cart2pol=function(x,y){
+ #Это r
+ newx=sqrt(x^2 + y^2)
+ #Это тета
+ newy=atan(y/x)
+ x_y=cbind(newx,newy)
+ return(x_y)
+ }
> dataset_cir2=cart2pol(dataset_cir$x,dataset_cir$y)
> plot(dataset_cir2)
```

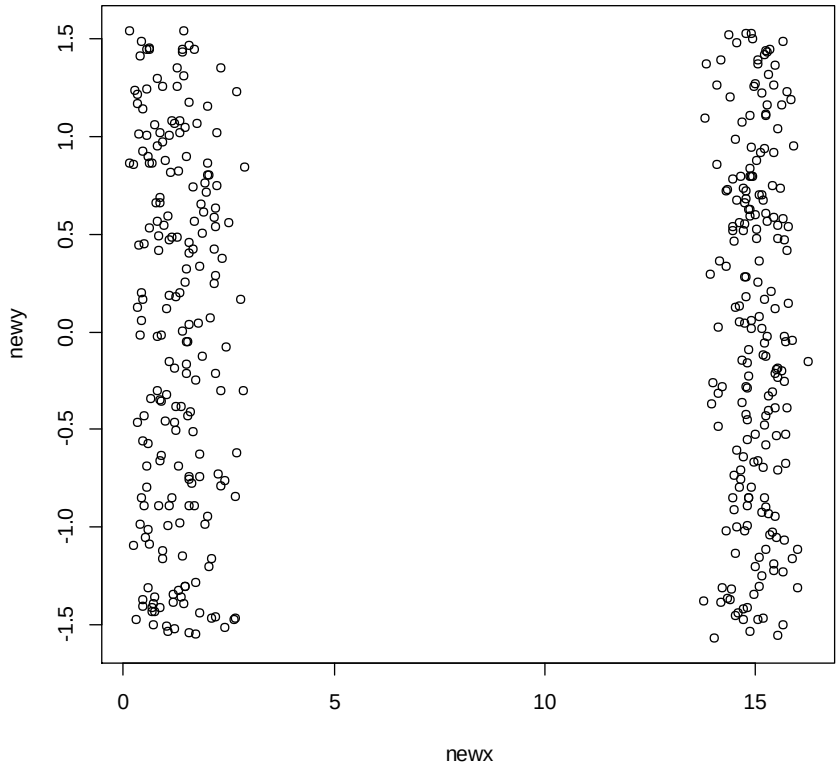


Рисунок 6.5. Визуализация преобразованных исходных круговых данных.

Запускаем модель k-средних для этих данных.

```
> k_clust_spher2=kmeans(dataset_cir2, centers=2)
> #Постройте данные и кластеры
> plot(dataset_cir2, col=k_clust_spher2$cluster,pch=2)
```

Алгоритм k-средних работает хорошо и правильно преобразует данные.

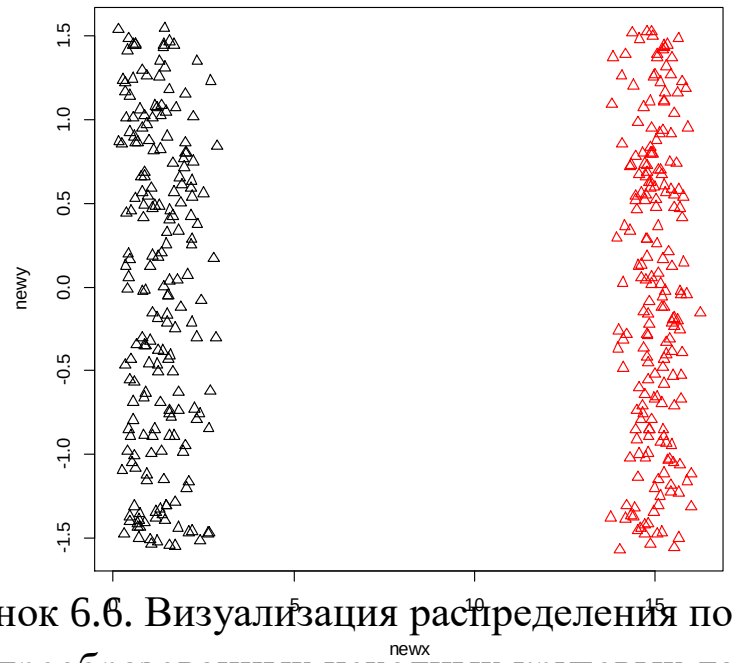


Рисунок 6.6. Визуализация распределения по кластерам преобразованных исходных круговых данных.

Теперь посмотрим на данные, в которых кластеры имеют разный размер. Одинаковый размер не означает, что кластеры должны быть в точности равными. Это просто означает, что ни в одном кластере не должно быть «слишком мало» участников.

```
> #Сделать первый кластер из 1000 случайных значений
> clust1 = data_frame(x = rnorm(1000), y = rnorm(1000))
> #Сохраните 10 значений вместе, чтобы получился второй кластер
>
clust2=data_frame(x=c(5,5.1,5.2,5.3,5.4,5,5.1,5.2,5.3,5.4),y=c(5,5,5,5,5,4.9,4.
9,4.9,4.9,4.9))
> #Объедините данные
> dataset_uneven=rbind(clust1,clust2)
> plot(dataset_uneven)
```

В этом случае выделяются два чистых кластера, но они не удовлетворяют аналогичным требованиям к условиям алгоритма k-средних.

```
> k_clust_spher3=kmeans(dataset_uneven, centers=2)
> plot(dataset_uneven, col=k_clust_spher3$cluster,pch=2)
```

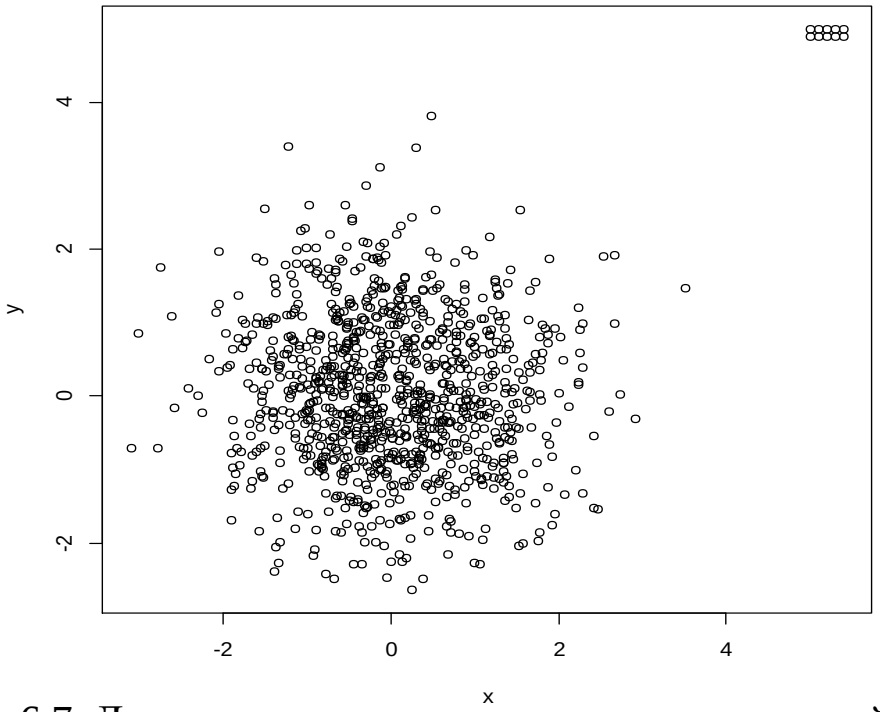


Рисунок 6.7. Данные, в которых кластеры имеют разный размер

Алгоритм k-means минимизирует меж кластерные и внутри кластерные расстояния и создавать «плотные» кластеры. В этом процессе он неправильно назначает некоторые точки данных в первом кластере второму кластеру. Это делает кластеризацию неточной.

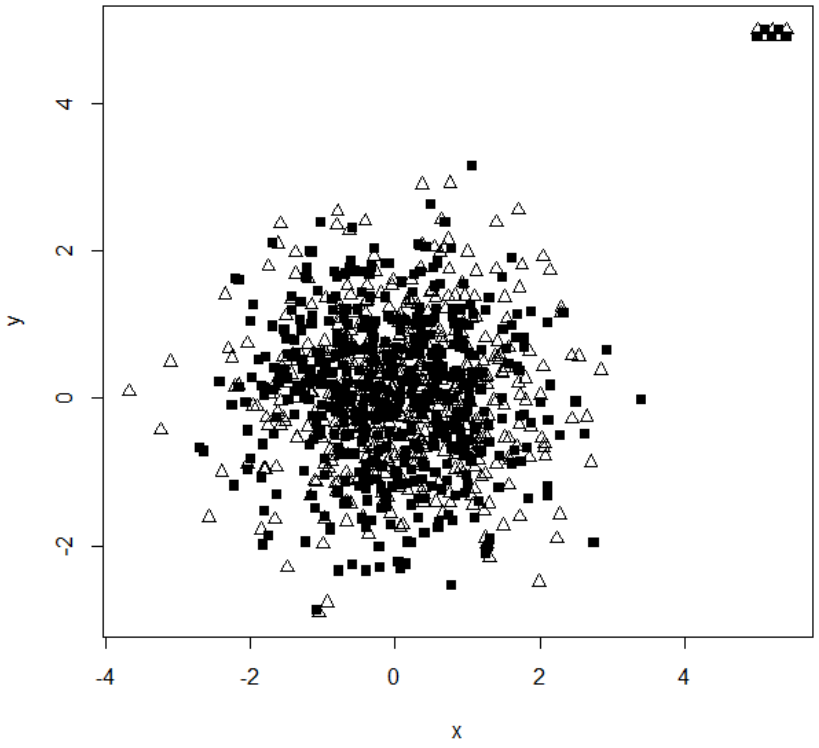


Рисунок 6.8. Визуализация результата алгоритма k-средних данных с разными размерами кластеров.

6.5. Определение значение k из исходных данных

Рассмотренные наборы данных просты, и кластеры легко идентифицировать. Сложные наборы данных не дают такой возможности. Метод «локтя» популярен для поиска подходящего значения «K» для кластеризации k-средних и является эвристикой, используемой при определении количества кластеров в наборе данных. Он состоит в построении графика объясненной вариации в зависимости от количества кластеров и выборе точки перегиба кривой или «локтя» для определения количества кластеров. Основан на том наблюдении, что увеличение количества кластеров позволяет уменьшить сумму дисперсии внутри каждого кластера. Большее количество кластеров позволяет извлекать более похожих друг на друга более мелких групп.

Для определения точки перегиба кривой используют суммы внутрикластерных отклонений относительно количества кластеров. Используется расчет SSE - ошибки суммы квадратов, то есть сумма квадратов разностей между предсказанными точками данных $\hat{y}_i$ и наблюдаемыми точками данных y_i.

$$SSE = \Sigma(\hat{y}_i - y_i)^2$$

Используя этот график, можно выбрать «k», который показывает резкое изменение SSE, создавая «эффект локтя».

Рассмотрим иллюстрацию на примере набора данных об ирисах (iris), изучив распределение по ширине и длине лепестка

```
> #Создадим вектор для хранения sse
> sse=vector('numeric')
> for(i in 2:15){
+ #Функция k-means в R имеет функцию внутри, которая сохраняет sse для каждой группы кластеров
+ sse[i-1]=sum(kmeans(iris[,3:4],centers=i)$withinss)
+ }
> #Преобразование sse в кадр данных и сохранение соответствующего значения k
> sse=as.data.frame(sse)
> sse$k=seq.int(2,15)
> #Создание сюжета. Этот сюжет также известен как screeplot
> plot(sse$k,sse$sse,type="b")
```

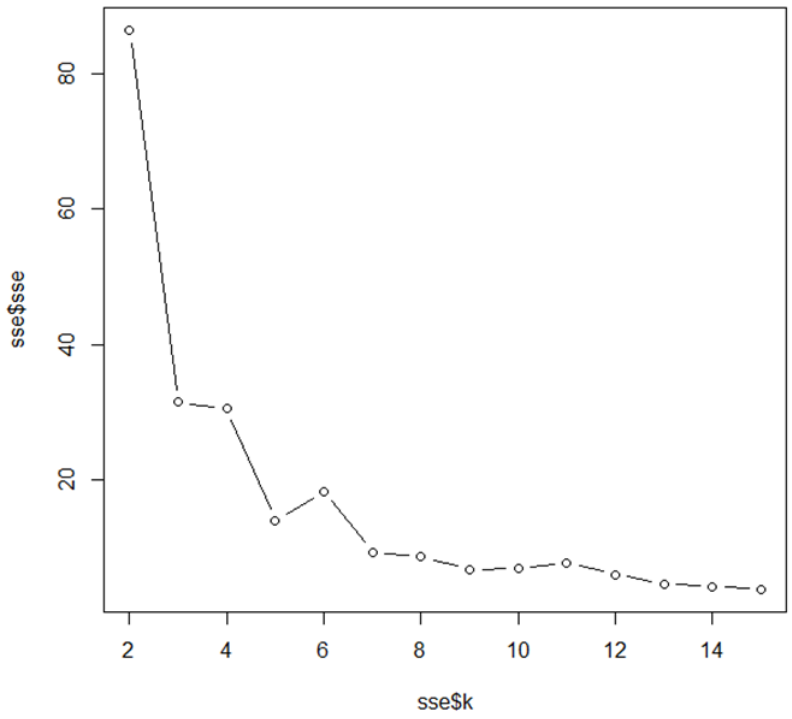


Рисунок 4.9. Первая точка перегиба k =3

На Рисунке 6.9 первая точка перегиба образуется при k = 3. Этот метод предполагает, что у нас должно быть 3 кластера для этих данных.

Алгоритм кластеризации k-средних - это один из первых и самых основных методов кластеризации без учителя. Этот метод для многих данных не только эффективен, но требует важность представления и понимания данных при обучении без учителя. Если любое из предположений о данных нарушается, то кластеры могут не сформироваться должным образом. Точно так же при определении значения количества кластеров «k» при использование неподходящих или произвольных значений может привести к неправильным кластерам. В каком-то смысле, кластеризация k-средних также зависит от определения правильного значения k для точной кластеризации данных. Поэтому, прежде чем приступить к реализации алгоритма, нужно четко понимать все аспекты алгоритма и его допущения, а не рассматривать его как черный ящик. В данном разделе показаны различные недостатки неправильной кластеризации данных в простых наборах данных, которые не удовлетворяют предположениям алгоритма.

Вопросы для самопроверки

1. Основные прикладные задачи классификации многомерных наблюдений.
2. Классификация объектов с учителем и без учителя.
3. Кластерный анализ и его особенности как метода классификации.
4. История возникновения и развития кластерного анализа.
5. Основные этапы кластерного анализа.
6. Расстояния между объектами и меры близости объектов в кластерном анализе.
7. Классификация методов кластерного анализа по различным параметрам.
8. Параллельные кластер-процедуры кластеризации.

9. Графическое отображение результатов кластеризации.
10. Метод К-средних.
11. 16. Сравнительный анализ методов иерархического и неиерархического кластерного анализа
12. 17. Представление результатов кластеризации.
13. Интерпретация результатов кластерного анализа.

Задание

№	Наименование	Содержание
1	beavers	Температура тела бобров Серия из двух
2	cars	Скорость и тормозной путь автомобилей
3	chickwts	Вес цыплят в зависимости от типа корма
4	co2	Концентрация co2 в атмосфере Мауна Лоа
5	crimtab	Наборы данных 3000 преступниках
6	discoveries	Ежегодное количество важных открытий
7	esoph	Данные о курении, употребления алкоголя и рака пищевода
8	euro	Курсы пересчета евро в евро
9	eurodist	расстояния между европейскими городами и между городами США
10	pressure	Давление паров ртути в зависимости от температуры
11	quakes	Локализация землетрясений у берегов Фиджи
12	rivers	Длины основных рек Северной Америки
13	rock	Встречаемость горных пород в образцах нефтяных пород
14	sleep	Данные о сне студентов
15	state	США Факты и цифры
16	sunspot.month	Ежемесячные данные о солнечных пятнах за месяц, с 1749 года по "настоящее"
17	sunspot.year	Ежегодные данные о солнечных пятнах, 1700-1988
18	sunspots	Ежемесячное количество солнечных пятен, 1749-1983
19	swiss	Швейцарские данные о рождаемости и социально-экономические показатели Швейцарии (1888)
20	treering	Ежегодные данные о годовых кольцев деревьев, -6000-1979
21	trees	Диаметр, высота и объем деревьев черной вишни
22	uspop	Перепись населения США
23	volcano	Топографическая информация о вулкане
24	women	Средний рост и вес американских женщин

ЛАБОРАТОРНАЯ РАБОТА № 7
Метод анализа главных компонент.
7.1. Метод анализ главных компонент на платформе R

Метод главных компонент (Principal component analysis) — это технология многомерного статистического анализа, используемая для *сокращения размерности* пространства *признаков* с минимальной потерей полезной информации.

С математической точки зрения метод главных компонент представляет собой ортогональное линейное преобразование, которое отображает данные из исходного пространство признаков в новое пространство меньшей размерности. Первая ось новой системы координат строится таким образом, чтобы дисперсия данных вдоль неё была бы максимальна. Вторая ось строится ортогонально первой так, чтобы дисперсия данных вдоль неё, была бы максимальной их оставшихся возможных и т.д. Первая ось называется первой главной компонентой, вторая — второй и т.д.

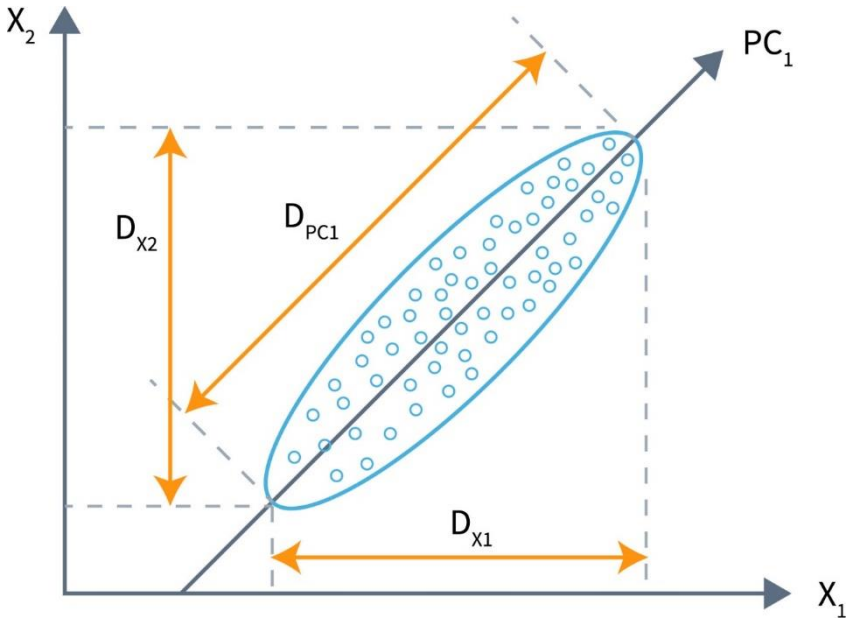


Рисунок 7.1 Снижение размерности исходного 2-мерного пространства с помощью метода главных компонент до 1-мерного.

На Рисунке 7.1 показано снижение размерности исходного 2-мерного пространства (X_1, X_2) с помощью метода главных компонент до 1-мерного. Первая главная компонента PC_1 ориентирована вдоль направления наибольшей вытянутости эллипсоида рассеяния точек объектов исходного набора данных в пространстве признаков, т.е. с ней связана наибольшая дисперсия.

На рисунке, также, несложно увидеть, что проекция дисперсии данных на ось первой главной компоненты DPC_1, больше, чем её проекции на исходные оси DX_1 и DX_2, но меньше их суммы. Т.е. с помощью первой главной компоненты выразить всю дисперсию данных не удается. Поэтому строят вторую, третью и т.д. главные компоненты, пока они суммарно не отразят всю дисперсию.

Таким образом, смысл метода главных компонент заключается в том, чтобы с каждой главной компонентой была связана определённая доля общей дисперсии исходного набора данных (её называют нагрузкой). В свою очередь, дисперсия, являющаяся мерой изменчивости данных, может отражать уровень их информативности.

То есть, вдоль некоторых осей исходного пространства признаков изменчивость может быть большой, вдоль других — малой, а вдоль третьих вообще отсутствовать. Предполагается, что чем меньше дисперсия данных вдоль оси, тем менее значим вклад переменной, связанной с данной осью и, следовательно, исключив эту ось из пространства (т.е. переменную из модели), можно уменьшить размерность задачи, почти не проиграв в информативности данных.

Следовательно, задача метода главных компонент заключается в том, чтобы построить новое пространство признаков меньшей размерности, дисперсия между осями которой будет перераспределена так, чтобы максимизировать дисперсию по каждой из них. Метод реализуется как последовательность следующих шагов:

1. Вычисляется общая дисперсия в исходном пространстве признаков. Вычисление общей дисперсии не сводится к простому суммированию дисперсий по каждой переменной, так как они, зачастую не являются независимыми. Поэтому суммируют взаимные дисперсии переменных из ковариационной матрицы.
2. Вычисляются собственные векторы и собственные значения ковариационной матрицы, определяющие направления главных компонент и величины связанных с ними дисперсий.
3. Снижают размерность следующим образом. Диагональные элементы ковариационной матрицы показывают дисперсию по исходной системе координат (переменных), а её собственные значения — по новой. Разделив дисперсию, связанную с каждой главной компонентой на сумму дисперсий по всем компонентам, получают долю дисперсии, связанную с каждой компонентой. После

этого отбрасывается столько главных компонент так, чтобы доля оставшихся составляла 80-90%.

Следует отметить, что такой директивный подход к выбору числа компонент не всегда даёт хорошие результаты. Это связано с тем, что часть дисперсии обусловлена шумами данных, а не информативностью компонент. Тогда, задав порог, например, 80% может оказаться, что в них только 60% дисперсии связаны с информативностью, а 20% с шумом. Поэтому на практике часто используют различные специальные критерии для определения числа компонент.

Основными ограничениями метода главных компонент являются:

- невозможность смысловой интерпретации компонент, поскольку они «вбирают» в себя дисперсию от нескольких исходных переменных;
- метод может работать только с непрерывными данными.

Метод главных компонент широко используется для снижения размерности входных данных на этапе их предварительного анализа.

Метод иногда рассматривают как часть более общего подхода к снижению размерности данных — факторного анализа.

Анализ главных компонент на примере данных iris.

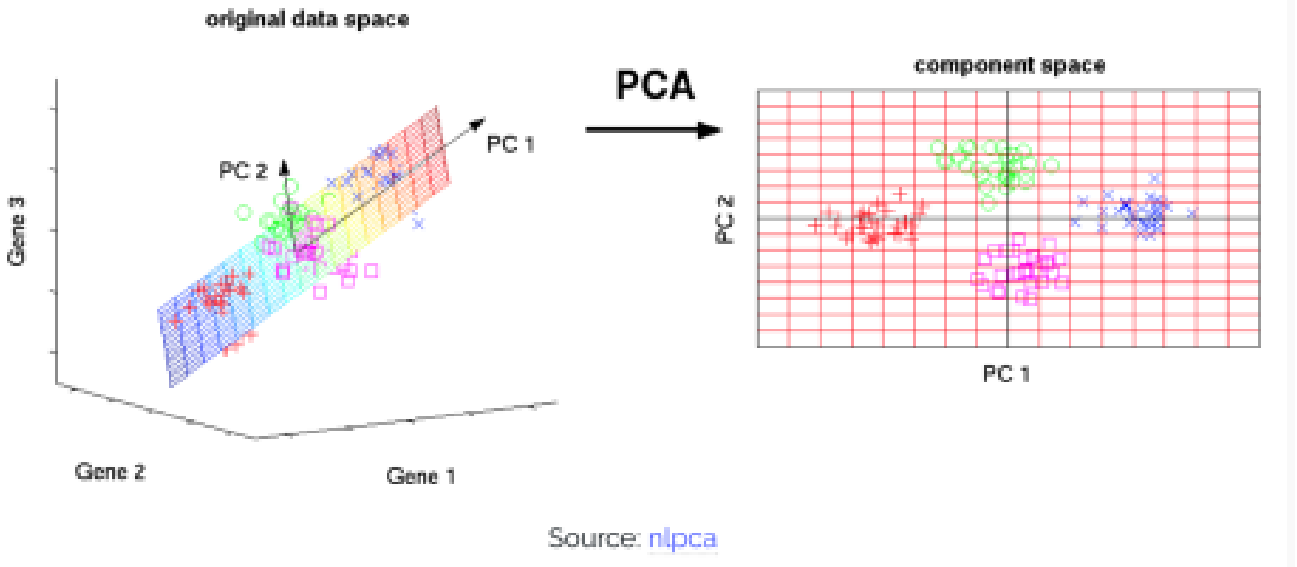


Рисунок 7.2. Сокращения размерности пространства признаков с минимальной потерей полезной информации.

7.2. Преобразование данных большой размерности в данные низкой размерности

Исходные данные iris представлены в пакете dataset языка R. Введем исходные данные:

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

Исключим столбец 5, содержащий название видов ирисов.

```
> #PCA with base R function
> iris1<-iris[,-5] # I selected all columns without 5
> head(iris1)
 Sepal.Length Sepal.Width Petal.Length Petal.Width
1   5.1   3.5   1.4   0.2
2   4.9   3.0   1.4   0.2
3   4.7   3.2   1.3   0.2
4   4.6   3.1   1.5   0.2
5   5.0   3.6   1.4   0.2
6   5.4   3.9   1.7   0.4
```

Функция princomp() пакета base выполняет анализ главных компонент на заданной числовой матрице данных и возвращает результаты в виде объекта класса princomp().

```
basePCA<-princomp(iris1)
basePCA
```

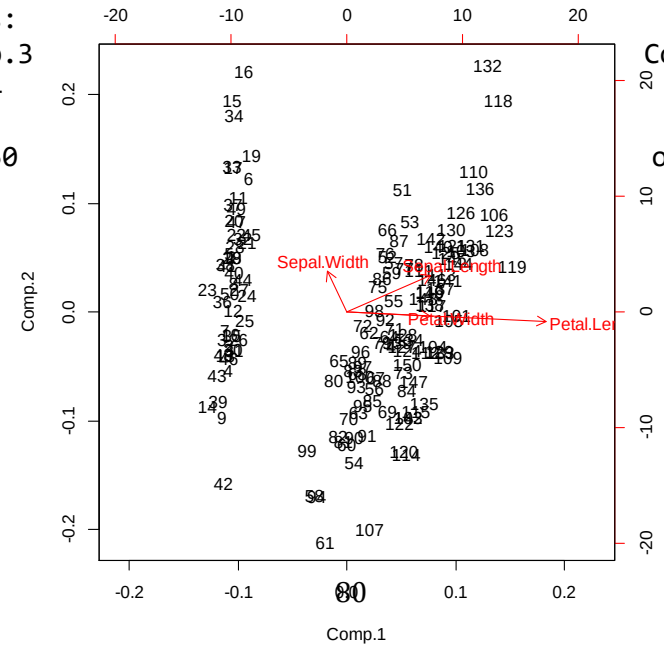
При печати basePCA получим следующие компоненты (Рисунок 5.3):

```
Call:
princomp(x = iris1)

Standard deviations:
 Comp.1 Comp.2 Comp.3
2.0494032 0.4909714
```

```
 4 variables and 150
```

```
biplot(basePCA)
```




```
Comp.4
 0.2787259 0.1538707
```

```
 observations.
```

Рисунок 7.3. Компоненты исходных данных

Используя библиотеку vegan, представим более информативный график.

```
library(vegan)
vrda<-rda(iris1)
biplot(vrda)
biplot(vrda, display = c("sites", "species"), type = c('text','points'),
+ main='Biplot for Principal Components with vegan library(iris data)')
```

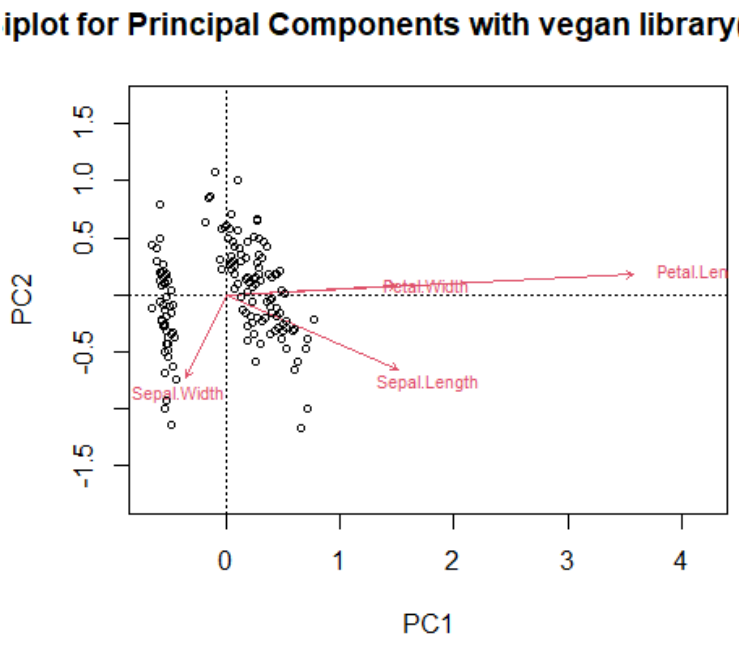


Рисунок 7.4. График повышенной информативности о компонентах с пакетом vegan (PC1 и PC2)

Используя PCA из пакета FactoMineR распечатаем график корреляции:

```
library(FactoMineR)
pca<-PCA(iris[,-1],graph=FALSE)
var <- get_pca_var(pca)
```

```
var

Principal Component Analysis Results for variables
 ===================================================
 Name  Description
1 "$coord" "Coordinates for the variables"
2 "$cor"  "Correlations between variables and dimensions"
3 "$cos2" "Cos2 for the variables"
4 "$contrib" "contributions of the variables"

corrplot(var$contrib, is.corr=FALSE)
```

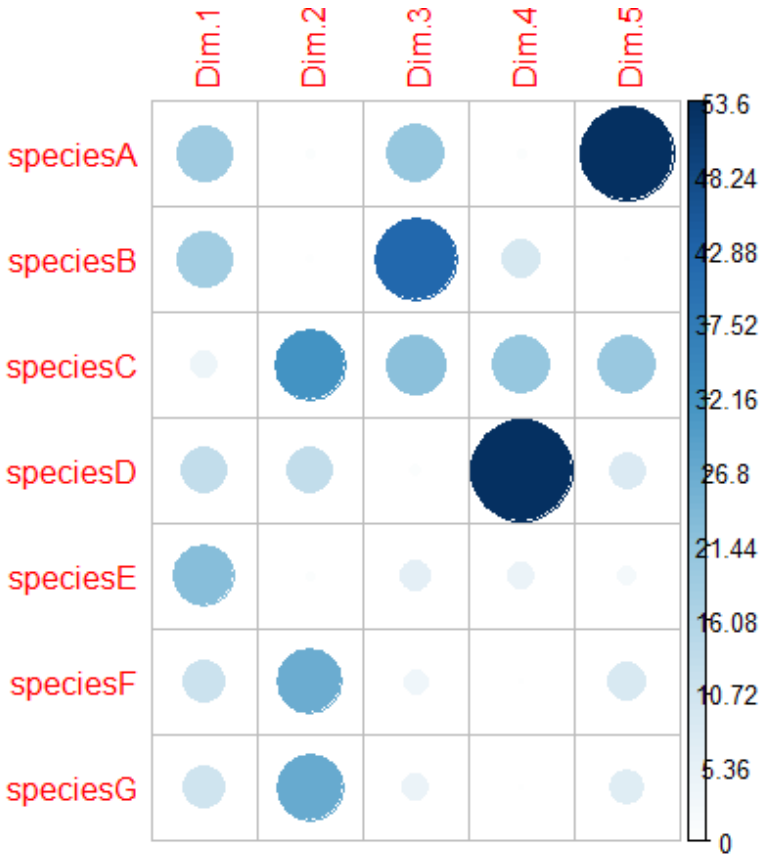


Рисунок 7.5. Визуализация корреляции исходных

7.3 Анализ собственных значений матрицы корреляций

Собственные значения указывают на количество вариаций, содержащиеся в главных компонентах. Собственные значения большие для первых PC и маленькие для последующих PC.

#Собственные значения

```
eig.val <- get_eigenvalue(pca)
eig.val
  eigenvalue variance.percent cumulative.variance.percent
Dim.1 3.964859352  56.6408479     56.64085
Dim.2 2.048397556  29.2628222     85.90367
Dim.3 0.599987080  8.5712440     94.47491
Dim.4 0.377301937  5.3900277     99.86494
```

```
Dim.5 0.009454076  0.1350582     100.00000
> #-----------

rda(X = iris1)
Call: rda(X = iris1)

    Inertia Rank
Total   4.573
Unconstrained 4.573 4
Inertia is variance

Eigenvalues for unconstrained axes:
 PC1 PC2 PC3 PC4
4.228 0.243 0.078 0.024
```

7.4 Интерпретация графиков PCA

Метод главных компонент позволяет получить разнообразные числовые данные и их визуализацию интерпретации результатов. Для этого необходимо:

- Понимать какая дисперсия объясняется каждым из компонентов.
- Понимать какие исходные переменные связаны с основными компонентами. (*Собственные значения измеряют величину вариации, объясняемую каждым главным компонентом (ГК), и будут наибольшими для первого ГК и меньшими для последующих ГК.В нашем случает около 57,85747 % отклонения объясняется этим первым собственным значением*).
- Объяснять кумулятивный процент, который получается путем сложения последовательных долей объясненной вариации для получения промежуточной суммы. (*Например, 73% плюс 23% равно 96% и так далее. Следовательно, около 85,00533% вариации объясняется первыми двумя собственными значениями вместе)*.

На рисунке 7.6, представлены только две оси (PC_1 и PC_2) необходимые для объяснения приблизительно 96% вариации данных, как это видно из значений совокупной доли.

```
fviz_pca_var(pca,select.var = list(contrib = 8), repel = TRUE)
```

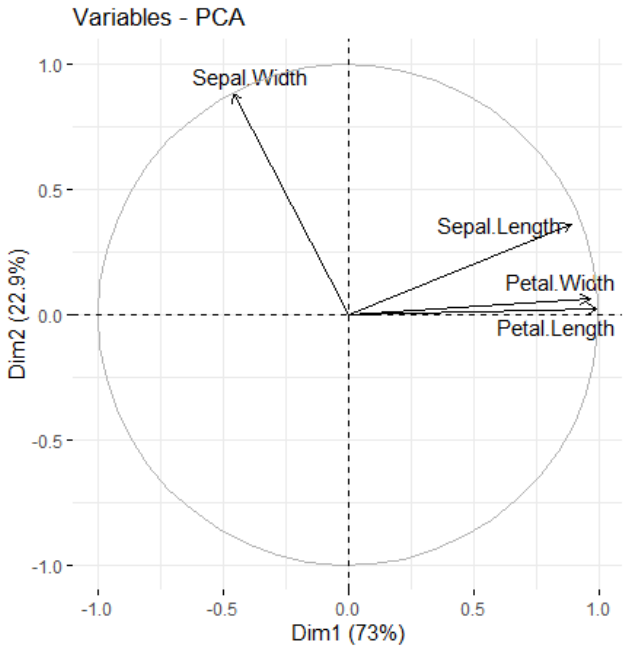


Рисунок 7.6. График переменных PCA

На Рисунке 7.7 показано, как с помощью функций представленных пакетов можно визуализировать распределение исходных данных на одной из плоскостей пары главных компонент.

```
p<-fviz_pca_biplot(basePCA,
 geom.ind = 'point', # show points only (nbut not 'text')
 col.ind = iris$Species, # color by groups
 palette = c('#00AFBB', '#E7B800', '#FC4E07'),
 addEllipses = TRUE, # Concentration ellipses
 legend.title = 'Species'
)

ggpubr::ggpar(p,
 title = 'Principal Component Analysis (iris data) with FactoMineR ',
 subtitle = 'simple biplot of individuals and variables',
 caption = 'Author: Hafez Ahmad',
 xlab = 'PC1', ylab = 'PC2',
 legend.title = 'Species', legend.position = 'top',
 ggtheme = theme_gray(), #palette = 'jco'
)
```

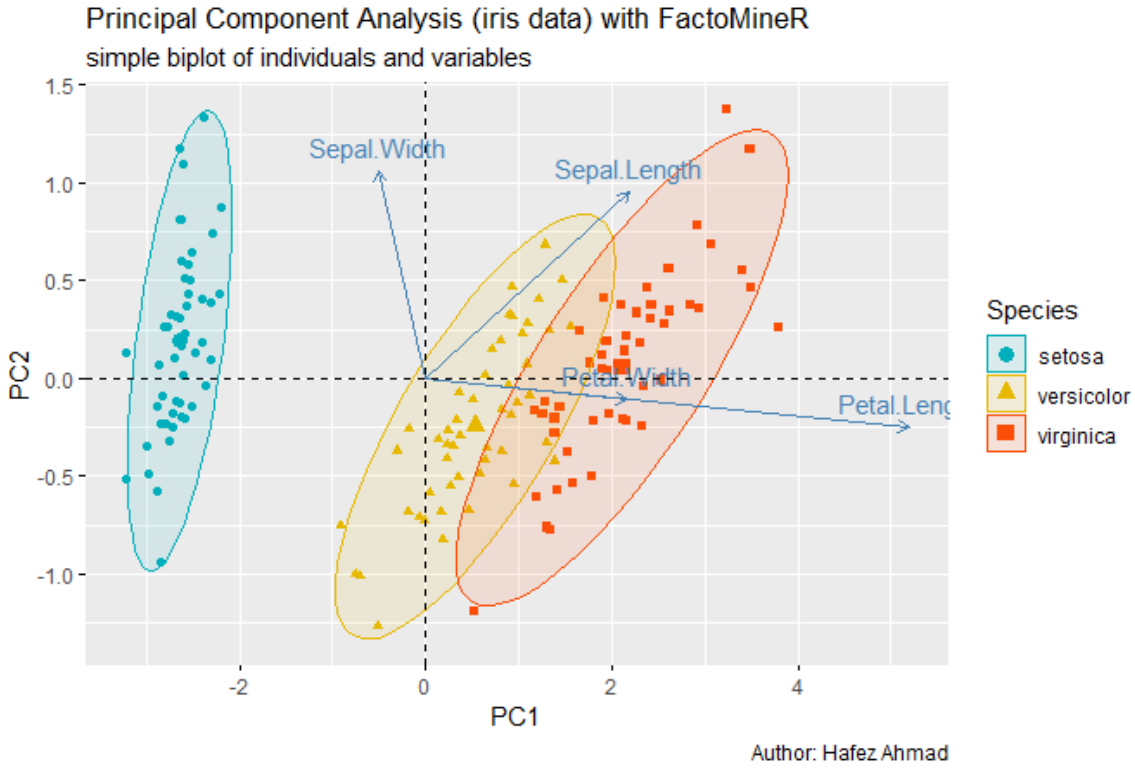


Рисунок 7.7. визуализация распределение исходных данных на одной из плоскостей пары главных компонент

Задания

Используйте следующие таблицы данных для анализа по методу главных компонент, содержащихся в них данных:

№ варианта	**1.**	**2.**	**3.**	**4.**	**5.**	**6.**	**7.**	**8.**	**9.**	**10.**
Набор данных	CO2		ChickWeight		iris	Orange	airquality	faithful		
Имя переменной (вектора)	conc	uptake	weight	Time	Sepal.Length	circumference	Wind	Temp	eruptions	waiting

ЛАБОРАТОРНАЯ РАБОТА № 8
Главные компоненты и факторный анализ.

Факторный анализ–это процедура, реализующая многомерный метод, изучения взаимосвязей между значениями переменных. С его большое число переменных, относящихся к имеющимся наблюдениям, сводят к меньшему количеству независимых величин, называемых факторами:

- в один фактор объединяются переменные, значительно коррелирующие между собой.
- переменные из разных факторов слабо коррелируют между собой.

Факторный анализ позволяет классифицировать наблюдаемые переменные, представленные в наблюдениях.

Фактор (Factor) – это латентная (скрытая) переменная, конструируемая таким образом, чтобы можно было объяснить получаемые корреляции между набором имеющихся переменных. Концепция факторного анализа заключается в «сжатии» информации. Предполагается, что известные переменные зависят от меньшего количества неизвестных переменных и случайной ошибки.

Факторный анализ рассматривается как более общий подход к снижению размерности данных в сравнении с методом главных компонен.

8.1. Главные компоненты

Функция princomp () производит анализ главных компонент без вращения.

```
# Анализ основных компонентов
# ввод необработанных данных и извлечение ПК
# из корреляционной матрицы
fit <- princomp(mydata, cor=TRUE)
> summary(fit) # print variance accounted for
Importance of components:
       Comp.1 Comp.2  Comp.3  Comp.4  Comp.5
Standard deviation  2.5706809 1.6280258 0.79195787 0.51922773 0.47270615
Proportion of Variance 0.6007637 0.2409516 0.05701793 0.02450886 0.02031374
Cumulative Proportion 0.6007637 0.8417153 0.89873322 0.92324208 0.94355581
       Comp.6  Comp.7  Comp.8  Comp.9  Comp.10
Standard deviation  0.45999578 0.36777981 0.35057301 0.277572792 0.228112781
Proportion of Variance 0.01923601 0.01229654 0.01117286 0.007004241 0.004730495
```

```
Cumulative Proportion 0.96279183 0.97508837 0.98626123 0.993265468 0.997995963
       Comp.11
Standard deviation  0.148473587
Proportion of Variance 0.002004037
Cumulative Proportion 1.000000000

loadings(fit) # pc loadings
Loadings:
  Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9 Comp.10
mpg 0.363   0.226   0.103 0.109 0.368 0.754 0.236 0.139
cyl -0.374   0.175    -0.169   0.231  -0.846
disp -0.368    -0.257 0.394 0.336 0.214   0.198
hp -0.330 0.249 -0.140   0.540    0.222 -0.576 0.248
drat 0.294 0.275 -0.161 -0.855  -0.244      -0.101
wt -0.346 -0.143 -0.342 -0.246   0.465    0.359
qsec 0.200 -0.463 -0.403  -0.165 0.330   0.232 -0.528 -0.271
vs 0.307 -0.232 -0.429 0.215 0.600 -0.194 -0.266   0.359 -0.159
am 0.235 0.429 0.206    0.571 -0.587    -0.178
gear 0.207 0.462 -0.290 0.265   0.244 0.605 -0.336  -0.214
carb -0.214 0.414 -0.529 0.127 -0.361 -0.184 -0.175 0.396 0.171
  Comp.11
mpg 0.125
cyl 0.141
disp -0.661
hp 0.256
drat
wt 0.567
qsec -0.181
vs
am
gear
carb -0.320

    Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
SS loadings  1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
Proportion Var 0.091 0.091 0.091 0.091 0.091 0.091 0.091 0.091 0.091
Cumulative Var 0.091 0.182 0.273 0.364 0.455 0.545 0.636 0.727 0.818
    Comp.10 Comp.11
SS loadings  1.000 1.000
Proportion Var 0.091 0.091
Cumulative Var 0.909 1.000

plot(fit,type="lines") # scree plot
```

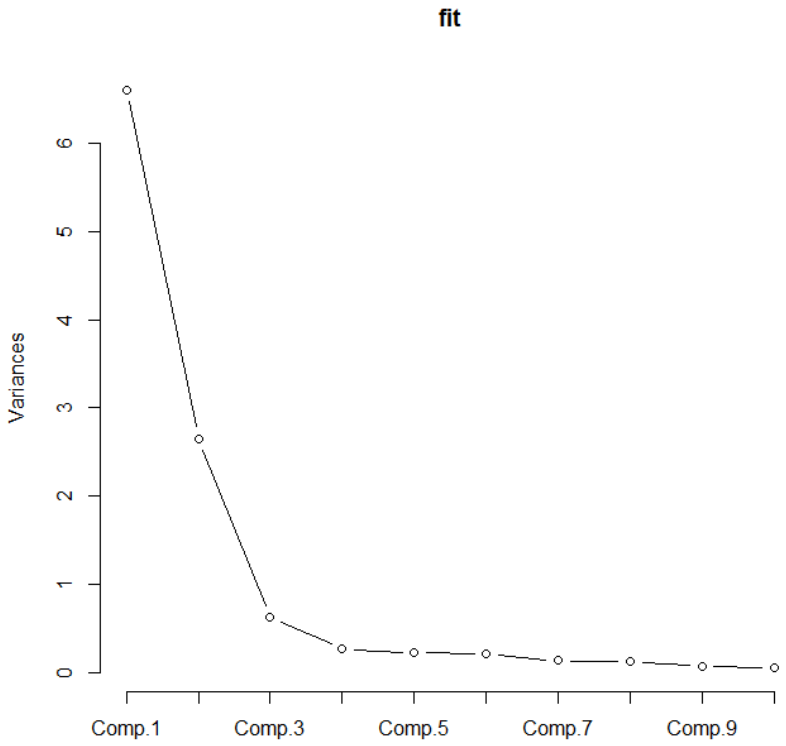


Рисунок 8.1 Основные компоненты

```
fit$scores # the principal components
biplot(fit)
```

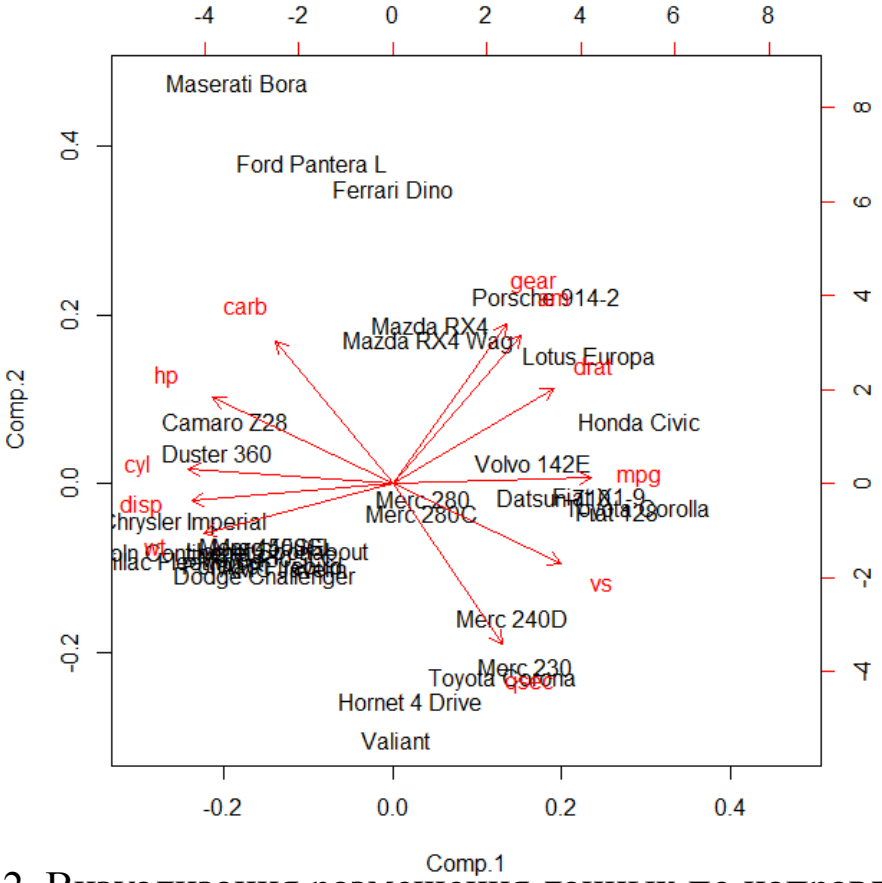


Рисунок 8.2. Визуализация размещения данных по направлениям основных компонент.

Используем значение аргумента cor = FALSE, чтобы основать главные компоненты на ковариационной матрице. Используйте параметр covmat = так, чтобы напрямую ввести корреляционную или ковариационную матрицу. При вводе ковариационной матрицы включите опцию n.obs = .

Функция factor.pa () из пакета psych может быть использована для извлечения и поворота основных компонентов.

```
# Varimax Rotated Principal Components
# retaining 5 components
library(psych)
fit <- principal(mydata, nfactors=5, rotate="varimax")
fit # print results
```

Аргумент mydata может быть матрицей необработанных данных или ковариационной матрицей. Обычно используется попарное удаление недостающих данных. Можно использовать варианты вращения получения нужной картины для интерпритации результатов "none", "varimax", "quatimax", "promax", "oblimin", "simpleimax" или "cluster"

8.2. Факторный анализ

Функция factanal () производит анализ факторов максимального правдоподобия.

```
# Факторный анализ Максимального правдоподобия
# ввод необработанных данных и извлечение 3 факторов,
# с вращением varimax
fit <- factanal(mydata, 3, rotation="varimax")
print(fit, digits=2, cutoff=.3, sort=TRUE)

Call:
factanal(x = mydata, factors = 3, rotation = "varimax")

Uniquenesses:
 mpg cyl disp hp drat wt qsec vs am gear carb
0.13 0.06 0.09 0.13 0.29 0.06 0.05 0.22 0.21 0.12 0.16

Loadings:
  Factor1 Factor2 Factor3
mpg 0.64 -0.48 -0.47
disp -0.72 0.54 0.32
drat 0.80
```

```
wt -0.78   0.52
am 0.88
gear 0.91
cyl -0.62 0.70
hp   0.72 0.51
qsec   -0.95
vs   -0.80
carb   0.56 0.72

    Factor1 Factor2 Factor3
SS loadings  4.38 3.52 1.58
Proportion Var 0.40 0.32 0.14
Cumulative Var 0.40 0.72 0.86

Test of the hypothesis that 3 factors are sufficient.
The chi square statistic is 30.53 on 25 degrees of freedom.
The p-value is 0.205


# печать factor 1 с factor 2
load <- fit$loadings[,1:2]
plot(load,type="n") # set up plot
text(load,labels=names(mydata),cex=.7) # add variable names
```

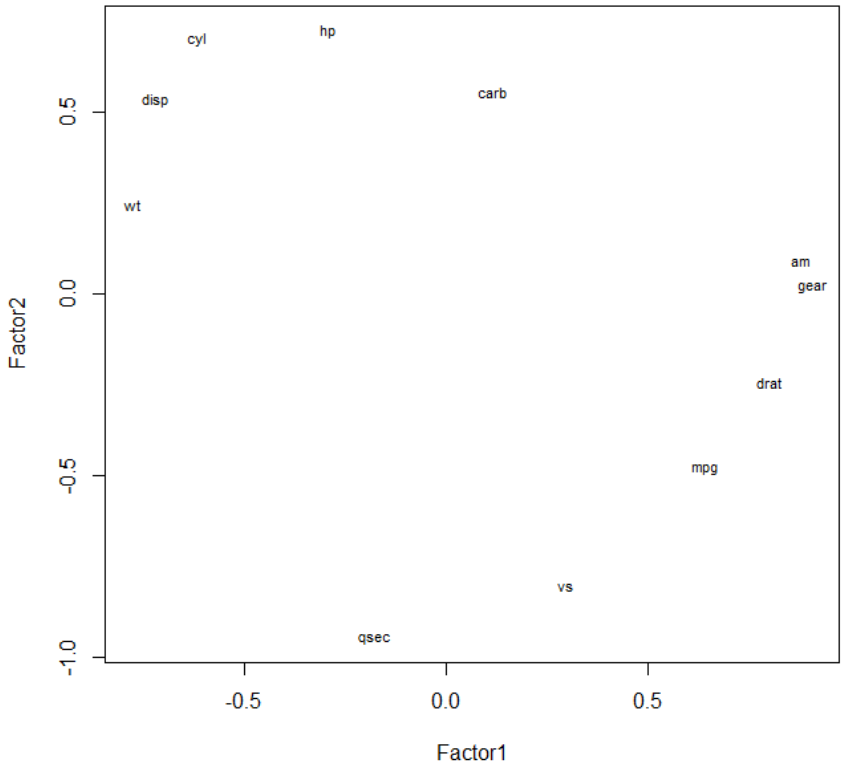


Рисунок 8.3 Визуализация распределения наблюдений в координатах factor 1 с factor 2

Параметры аргумента rotation= options include "varimax", "promax", and "none". = включают "varimax", "promax" и "none". Следует добавить опцию scores = "regression" или "Bartlett", чтобы получить факторные оценки. Используйте параметр covmat = , чтобы напрямую ввести корреляционную или ковариационную матрицу. При вводе ковариационной матрицы включите опцию n.obs = .

Функция factor.pa () в психологически пакете предлагает ряд функций факторного анализа, связанные, в том числе главной оси факторинга.

```
# Факторный анализ по осям главных компонент
library(psych)
fit <- fa(mydata, nfactors=3, rotate = 'varimax', fm = 'pa')
> fit # print results
Factor Analysis using method = pa
Call: fa(r = mydata, nfactors = 3, rotate = "varimax", fm = "pa")
Standardized loadings (pattern matrix) based upon correlation matrix
  PA2 PA3 PA1 h2 u2 com
mpg 0.67 -0.45 -0.48 0.88 0.120 2.7
cyl -0.63 0.70 0.28 0.96 0.042 2.3
disp -0.72 0.54 0.30 0.89 0.105 2.2
hp -0.31 0.68 0.55 0.87 0.132 2.3
drat 0.81 -0.24 -0.06 0.71 0.288 1.2
wt -0.79 0.25 0.48 0.91 0.092 1.9
qsec -0.18 -0.93 -0.21 0.94 0.060 1.2
vs 0.29 -0.80 -0.23 0.78 0.224 1.4
am 0.89 0.11 -0.08 0.82 0.184 1.0
gear 0.90 0.01 0.25 0.86 0.136 1.2
carb 0.09 0.51 0.78 0.87 0.132 1.8

      PA2 PA3 PA1
SS loadings   4.44 3.34 1.70
Proportion Var  0.40 0.30 0.15
Cumulative Var  0.40 0.71 0.86
Proportion Explained 0.47 0.35 0.18
Cumulative Proportion 0.47 0.82 1.00

Mean item complexity = 1.7
Test of the hypothesis that 3 factors are sufficient.

The degrees of freedom for the null model are 55 and the objective function was
15.4 with Chi Square of 408.01
The degrees of freedom for the model are 25 and the objective function was 1.3
```

```
The root mean square of the residuals (RMSR) is 0.02
The df corrected root mean square of the residuals is 0.02

The harmonic number of observations is 32 with the empirical chi square 0.85
with prob < 1
The total number of observations was 32 with Likelihood Chi Square = 31.8 with
prob < 0.16

Tucker Lewis Index of factoring reliability = 0.954
RMSEA index = 0.087 and the 90 % confidence intervals are 0 0.181
BIC = -54.84
Fit based upon off diagonal values = 1
Measures of factor score adequacy
             PA2 PA3 PA1
Correlation of (regression) scores with factors 0.98 0.97 0.93
Multiple R square of scores with factors   0.97 0.95 0.86
Minimum correlation of possible factor scores  0.93 0.89 0.71
```

Аргумент mydata может быть матрицей необработанных данных или ковариационной матрицей. Используется попарное удаление недостающих данных. Вращение может быть «варимакс» или «промакс».

8.3. Определение количества факторов для извлечения

Важнейшим решением в исследовательском факторном анализе является то, сколько факторов нужно извлечь. Пакет nFactors предлагает набор функций, помогающих принять это решение. Конечно, любое факторное решение должно быть интерпретируемым, чтобы быть полезным.

```
# Определите количество факторов для извлечения
library(nFactors)
ev <- eigen(cor(mydata)) # get eigenvalues
ap <- parallel(subject=nrow(mydata),var=ncol(mydata),
 rep=100,cent=.05)
nS <- nScree(x=ev$values, aparallel=ap$eigen$qevpea)
plotnScree(nS)
```

Пакет FactoMineR предлагает большое количество дополнительных функций для исследовательского факторного анализа. Это включает использование как количественных, так и качественных переменных, а также включение дополнительных

переменных и наблюдений. Вот пример типов графиков, которые вы можете создать с помощью этого пакета.

```
# PCA Variable Factor Map
library(FactoMineR)
result <- PCA(mydata) # graphs generated automatically
```

Пакет GPARotation предлагает множество вариантов вращения за пределами Varimax и PROMAX.

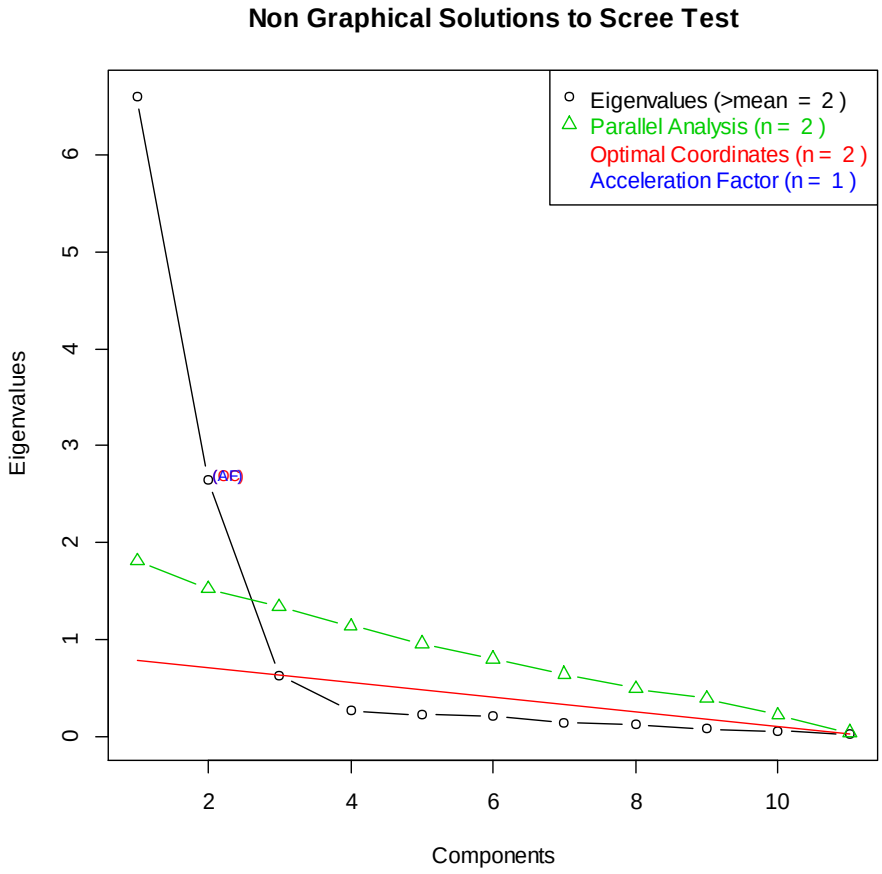


Рисунок 8.4. Визуадизация для качественного выделения знаимых факторов

Задания

Проведите факторный анализ с данными представленными в таблицаъ

№ варианта	**1.**	**2.**	**3.**	**4.**	**5.**
Набор данных	airquality	state.x77	Cars93*	Cars93*	Cars93*
y	Ozone	Life Exp***	Price	Min.Price	Max.Price
Факторы и их значения для прогноза	Solar.R=350 Wind = 8,3 Temp = 80	Illiteracy = 1,0 Murder = 12 Income = 4000	Horsepower = 200 RPM = 5200 Passengers = 4	Horsepower = 210 RPM = 5500 Passengers = 4	Horsepower = 220 RPM = 6000 Passengers= 6
№ варианта	**6.**	**7.**	**8.**	**9.**	**10.**
Набор данных	stackloss	longley	longley	LifeCycleSavings	Anscombe**
y	stack.loss	GNP	Employed	sr	education
Факторы и их значения для прогноза	Air.Flow= 55 Water.Temp= 20 Acid.Conc = 89	Unemployed = 221 Armed.Forces= 180 Population= 125	GNP.deflator=102 Armed.Forces= 170 Population = 110	pop15 = 35,5 pop75 = 1,5 dpi = 2500 ddpi = 2,15	income= 3200 young= 347,8 urban = 425

Требование к отчету.

Содержание отчета: отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выравнен по ширине.

Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчеты приводятся выводы о проделанной работе.

В отчет необходимо вставлять скриншоты выполненной работы и добавлять опи-сание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись, например, «Рисунок 1 – Создание подсистемы» и ссылку на него в тексте.

ЛИТЕРАТУРА

1. Айвазян С. А., Мхитарян В. С. Прикладная статистика и основы эконометрики. — М.: ЮНИТИ, 1998. — 650 с.
2. Анализ данных в R [Электронный ресурс] // Stepic.org. — Режим доступа: https://stepic.org/course/Анализ-данных-в-R-129.
3. Зорин А. В., Федоткин М. А. Введение в прикладной статистический анализ в пакете R: учебно-методическое пособие. — Нижний Новгород: ННГУ, 2010. — 50 с.
4. Мастицкий С. Э., Шитиков В. К. Статистический анализ и визуализация данных с помощью R. — М.: ДМК Пресс, 2015. — 496 с.
5. Семенычев В. К., Коробецкая А. А., Кожухова В. Н. Предложения эконометрического инструментария моделирования и прогнозирования эволюционных процессов: монография. — Самара: САГМУ, 2015. — 384 с.
6. Эконометрика [Электронный ресурс] // Coursera. — Режим доступа: https://www.coursera.org/course/econometrics.
7. Code School — Try R [Электронный ресурс]. — Режим доступа: http://tryr.codeschool.com/.
8. Coghlan A. A Little Book of R for Time Series [Электронный ресурс]. — 2015. — Режим доступа: https://media.readthedocs.org/pdf/a-little-book-of-r-for-time-series/latest/a-little-book-of-r-for-time-series.pdf.
9. ggplot2 Help [Электронный ресурс]. — Режим доступа: http://docs.ggplot2.org/current/index.html.
10. Package ‘GA’ [Электронный ресурс] // The Comprehensive R Archive Network. — Режим доступа: https://cran.r-project.org/web/packages/GA/GA.pdf.
11. Peng R. D. R Programming for Data Science [Электронный ресурс]. — Режим доступа: https://leanpub.com/rprogramming/.
12. quantmod: Quantitative Financial Modelling & Trading Framework for R [Электронный ресурс]. — Режим доступа: http://www.quantmod.com/.

Образец

МИНИСТЕРСТВО ОБРАЗОВАНИЯ, НАУКИ И МОЛОДЕЖИ РЕСПУБЛИКИ КРЫМ

РЕСПУБЛИКАНСКОЕ ВЫСШЕЕ УЧЕБНОЕ ЗАВЕДЕНИЕ

«КРЫМСКИЙ ИНЖЕНЕРНО-ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет экономики, менеджмента и информационных технологий

Кафедра прикладной информатики

Лабораторная работа

на тему

_____

по дисциплине

«Обработка и анализ больших данных (Big Data)»

Отчет по лабораторной работе

Исполнитель:

Студент(ка) группы _____

_____

Симферополь

_____