Эконометрика с R

Артамонов Н.В.

24 марта 2019 г.

Оглавление

Глава 1

Линейная регрессия в R

R – это объектно-ориентированный язык программирования, ориентированный на статистическую обработку и визуализацию данных. Официальный сайт языка R: www.r-project.org

Для работы с R необходимо установить

1. интерпретатор языка R («командная строка»)
2. графическую оболочку (упрощает работу со скриптами).

Есть несколько графических оболочек. Наиболее распространенной и функциональной является оболочка RStudio (www.rstudio.com).

При работе с R следует придерживаться следующего:

- каждая команда занимает одну или несколько строк кода;
- если в одной строке нужно разместить несколько команд, то их следует разделять точкой с запятой;
- в именах переменных допускаются буквы, числа, знак подчёркивания, точка.

1.1 Типы данных в R

В R реализована поддержка следующих типов данных/классов

- **Числовые**: целочисленные integer, действительные double, комплексные complex;

- **логические** logical (значения FALSE и TRUE);
- **символьные** character (задаются в двойных или одинарных кавычках).

В R выделены следующие особые объекты:

- `NA`[1] – «отсутствие значения»;
- `NULL` – «ничто» или «пустое значение»;
- `Inf` – «бесконечность»;
- `NaN`[2] – «не число».

1.2 Основные операторы

R является языком с динамической типизацией, т.е. переменные не обязательно объявлять заранее и любой переменной можно присвоить любое значение. Для присвоения значения переменной в R используются следующие операторы: `=, <-, ->`

Пример. Следующие три следующие команды равносильны и присваивают переменной `a` значение 3:

```
a = 3
a <- 3
3 -> a
```

Пример. Следующие команды

```
x = 2; x <- TRUE; "abc" -> x
```

последовательно присваивают переменной `x` числовое, логическое и символьное значение. В итоге значение переменной равно `"abc"`

В R реализованы также следующие операции:

`+, -, *, /, ^`	арифметические операции
`%*%`	умножение матриц
`==, <, >, !=, <=, >=`	сравнение
`!, &, |`	логические операции (отрицание, И, ИЛИ)

[1] `NA` = Not Available
[2] `NaN` = Not A Number

1.3 Вектора, матрицы, дата фреймы

Рассмотрим работу в R со следующими классами данных:

1. векторы (vector)
2. списки (list)
3. матрицы (matrix)
4. факторы (factor)
5. дата фреймы (data frame)

Вектор представляет из себя одномерный массив одного примитивного типа (числовой, строковый и др.). Для явного задания вектора нужно использовать функцию `c()`.

Пример. Команды

```
a <- c(2.3, 2, 4, 5, 4.1)
b <- c("x", "Hello, world!", "abc")
```

создают числовой и строковый векторы длины 5 и 3 соответственно.

Чтобы получить элемент вектора, нужно в квадратных скобках указать его номер

Пример. Для векторов из предыдущего примера `a[1]` равно `2.3`, a`[3]` равно 4, `b[1]` равно `"x"`, `b[3]` равно `"abc"`.

При создании вектора его элементам можно присвоить имена

Пример. Команда

```
c <- c(a=1, b=2.4, c=-1)
```

создает числовой вектор длины три, элементы которого имеют имена `a`, `b` и `c`.

Значения элементов вектора можно получить как по номеру, так и по их имени. Так `c[1]=c["a"]=1`.

Отметим, что для команды

```
d.1 <- 1:5; d.2 <- 3:6
```

создают вектора

$$d.1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \end{pmatrix} \quad d.2 = \begin{pmatrix} 3 & 4 & 5 & 6 \end{pmatrix}$$

Список также можно рассматривать как одномерный массив. Но, в отличие от вектора, элементы списка могут быть разных примитивных типов. Для явного создания списка нужно использовать функцию `list()`

Пример. Команда

```
mylist1 <- list(-1, "abc", TRUE)
```

создает список, состоящий из числа, строки и логического значения `TRUE`. Тогда `mylist1[1]` равно `-1`, `mylist1[2]` равно `"abc"`.

При создании списка его элементам можно присвоить отдельные имена

Пример. Команда

```
mylist2 <- list(x = FALSE, y = "abc", z = 1.5)
```

создает список, элементы которого имеют имена `x`, `y` и `z`.

Чтобы получить значение элемента нужно после имени списка указать имя элемента через знак `$` или указать имя элемента в кавычках в квадратных скобках.

Пример. Для списка из предыдущего примера `mylist2$x` равно `FALSE`; `mylist2$z` равно `1.5`; `mylist2['y']` равно `"abc"`.

Матрица представляет собой двумерный массив одного примитивного типа. В R матрица рассматривается как вектор (одномерный) с дополнительными атрибутами: размером матрицы (число строк и/или число столбцов). Создать матрицу можно функцией

```
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,
        dimnanes = NULL)
```

с аргументами

`data`	вектор
`nrow`	число столбцов
`ncol`	число строк
`byrow`	если `FALSE`, то матрица заполняется по столбцам, иначе – по строкам
`dimnames`	список длины 2, содержащий имена строк и столбцов

Пример. Рассмотрим скрипт

```
v <- 1:6
m1 <- matrix(v, nrow = 2)
m2 <- matrix(data = v, ncol = 3)
m3 <- matrix(v, nrow = 2, byrow = TRUE)
```

Он создает вектор

$$v = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix}$$

и матрицы

$$m1 = m2 = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix} \quad m3 = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

Чтобы получить элемент матрицы в строке с номером i и столбце с номером j после имени матрицы нужно указать `[i,j]`. Если после имени матрицы указать `[i,]` или `[,j]`, то получим целиком строку или столбец матрицы соотвественно.

Пример. Для матриц из предыдущего примера `m1[1,2]` равно 3; `m3[2,3]` равно 6; `m1[,2]` равно (3, 4); а `m3[1,]` равно (4, 5, 6).

Факторы рассматриваются как вектор и используются для представления качественных данных. Факторная переменная представляет собой вектор, элементы которого принадлежат конечному множеству «качественных значений». Реализована работа как с упорядоченными, так и с неупорядоченными факторами.

Дата фрейм или таблица данных представляет собой двумерный массив для работы со статистическими данными разной природа: как количественными, так и качественными (факторными признаками). Это отличает дата фреймы от матрицы, которые объединяет значения одного примитивного типа.

Дата фрейм можно рассматривать как таблицу, в которой

- каждый столбец имеет свое имя, которое можно рассматривать как имя фактора/переменной
- столбцы – статистические данные по соответствующей переменной
- в каждом столбце могут значения только одного примитивного типа (числовые, факторные)

- в разных столбцах допускаются значения разных примитивных типов

Создать дата фрейм явно можно функцией `data.frame()`. Для создания факторной переменной следует использовать функцию `as.factor()`

Пример. Рассмотрим таблицу со статистическими данными

price	age	color
10000	3	white
12000	2	red
9300	5	white
15000	1	red
9700	3	white
11000	2	black

по продажам подержанных автомобилей. Переменные price (цена продажи) и age (возраст) являются количественными, а переменная color является качественной (факторной). Создадим дата фрейм

```
cardata <- data.frame(
        price = c(10000, 12000, 9300, 15000, 9700, 11000),
        age = c(3, 2, 5, 1, 3, 2),
        color = factor(c("white", "red", "white",
                "red", "white", "black")) )
```

Чтобы получить значения отдельной переменной из дата фрейма нужно после его имени указать имя переменной через знак `$`. Также как и для матриц значения дата фрейма можно получить по номеру строки/столбца или имени столбца

Пример. Для дата фрейма из предыдущего примера `cardata$age` равно (3, 2, 5, 1, 3, 2); `cardata[1,2]` равно 3; `cardata[1,]` равно (10000, 3, white); `cardata[,1]` и `cardata[,'price']` равно (10000, 12000, 9300, 15000, 9700, 11000)

При импортировании статистических данных из внешних источников (файл, базы данных и др) соотвествующие функции, как правило, автоматически создают дата фреймы.

1.4 Функции

Функции бывают либо стандартными[3] (например, функция `cor`, вычисляющая корреляцию, функция `lm`, осуществляющая оценку линейной модели методом наименьших квадратов), либо подключаемые из внешних пакетов (см раздел 1.7).

Функции имеют аргументы двух типов – обязательные и необязательные. Обязательные должны быть заданы, необязательные могут быть заданы, а могут – и нет. В этом случае им как правило приписывается значение по умолчанию. Порядок аргументов не важен. Причем в большинстве случаев можно писать подставляемый в функцию объект, не указывая, каким из аргументов функции он соответствует.

Например коэффициент корреляции между двумя переменными `x` и `y` можно вычислить с помощью такой команды

```
cor(x, y)
```

Можно указать необязательный аргумент `method`, говорящий, какой именно коэффициент корреляции будет вычислен: Пирсона, Спирмена, Кендалла. По умолчанию вычисляется коэффициент Пирсона, коэффициент Спирмена вычисляется так

```
cor(x, y, method = "spearman")
```

Как правило в результате выполнения функции создается сложный объект (часто список). Его компоненты перечисляются в описании к функции. Например стандартная команда `cor.test`, осуществляющая тест коэффициента корреляции на значимость, создает объект (список)

```
a <- cor.text(x,y)
```

который содержит компоненты `a$statistic`, где содержится значение тестовой статистики, `a$p.value`, где содержится P-значение теста.

1.5 Функция `summary()`

Эта функция может быть применена ко многим объектам[4] и выводит «итоги» для объекта. Результаты выполнения зависят от класса объекта:

[3] входят в первоначальную установку R

[4] Такие функции называются **generic functions**

- Для вектора `summary()` вычисляет описательные статистики (min/max, среднее, медиана, квартили)
- Для матриц `summary()` вычисляет описательные статистики по столбцам (min/max, среднее, медиана, квартили)
- Для дата фреймов для каждого столбца (переменной)
  - для количественной переменной вычисляются описательные статистики (min/max, среднее, медиана, квартили)
  - для факторной переменной вычисляются количество наблюдений в выборке для каждого «значения» качественного признака.
- Для оцененной модели регрессии выводится протокол оценивания, который включает основные показатели: оценки коэффициентов, показатель «качества подгонки» R^2, основные тестовые статистики и др.

Пример. Для дата фрейма `cardata` из предыдущего примера получаем

```
> summary(cardata)
     price            age          color
 Min.   : 9300   Min.   :1.000   black:1
 1st Qu.: 9775   1st Qu.:2.000   red  :2
 Median :10500   Median :2.500   white:3
 Mean   :11167   Mean   :2.667
 3rd Qu.:11750   3rd Qu.:3.000
 Max.   :15000   Max.   :5.000
```

1.6 Основные распределения

Функции основных распределений, встроенных в R[5]:

[5]Вместо * нужно подставить q, d, p или r если нужно вычислить квантиль, плотность pdf. функция распределения cdf или сгенерировать случайную выборку соответственно

Функция R	распределение
`*norm`	нормальное (гауссово)
`*lnorm`	log-нормальное
`*beta`	бета
`*binom`	биномиальное
`*nbinom`	отрицательное биномиальное
`*cauchy`	Коши
`*chisq`	χ^2
`*exp`	экспоненциальное
`*f`	Фишера или F-распределение
`*gamma`	гамма
`*geom`	геометрическое
`*hyper`	гипергеометрическое
`*pois`	Пуассона
`*t`	Стьюдента или t-распределение
`*unif`	равномерное
`*weibull`	Вебулла

Рассмотрим как вычислять критические значения основных распределений. Пусть `a` – уровень значимости (обычно `a = 0.1, 0.05, 0.01`).

Распределение	команда в R для критических значений
$\mathcal{N}(0,1)$	`qnorm(p=1-a/2)`
χ^2_n	`qchisq(p=1-a, df=n)`
t_n	`qt(p=1-a/2, df=n)`
$F_{k,n}$	`qf(p=1-a, df1=k, df2=n)`

1.7 Работа с внешними пакетами

Базовый функционал языка R может быть существенно расширен за счёт внешних подключаемых пакетов, написанных сторонними разработчиками. Практически все (в том числе самые современные) статистические и эконометрические методы (и не только) реализованы в виде сторонних пакетов[6].

[6]При этом может случиться, что один и тот же метод может быть реализован в разных пакетах, созданных независимо разными разработчиками. Естественно, что реализации могут сильно различаться.

Внешние пакеты содержат функции, реализующие дополнительную функциональность, и иногда статистические наборы данных (датасеты) для иллюстрации функционала пакета.

Внешние пакеты хранятся во внешних репозиториях. Официальный репозиторий называется CRAN[7] (cran.r-project.org). По умолчанию пакеты загружаются из него. Также можно загружать пакеты из неофициальных репозиториев. Например, многие пакеты замещаются разработчиками на GitHub (github.com).

Чтобы воспользоваться внешним пакетом его необходимо сначала загрузить из репозитория и установить локально (это достаточно сделать один раз). Для этого нужно выполнить команду

```
install.packages("PackageName")
```

Чтобы воспользоваться всеми возможностями пакета в скрипте его сначала нужно загрузить в память с локального диска командой

```
library("PackageName")
```

или

```
require("PackageName")
```

Пакет предварительно должен быть установлен, иначе будет сообщение об ошибке. После этого в любом месте кода можно обращаться ко всем функциям и датасетам пакета.

Если по каким либо причинам нужно вызвать только одну функцию из пакета **без его полноценной загрузки** в память, то нужно писать

```
PackageName::FunctionName(...)
```

Такая ситуация, например, может возникать если есть конфликт имён.

1.8 Загрузка данных

R поддерживает широкий функционал по импорту статистических данных из файлов многих форматов (csv, MS Excel и проч.). Это обеспечивается как встроенным функционалом R, так и сторонними пакетам.

При импорте данных из файлов (таблиц данных) функции возвращают значение в формате дата фрейм.

[7] CRAN = Comprehensive R Archive Network

1.8.1 Загрузка встроенных в пакеты датасетов

Как говорилось ранее многие сторонние пакеты содержат встроенные наборы данных для демонстрации функционала пакета. Например, пакет по визуализации данных `ggplot2` содержит набор данных `diamonds` о почти 54000 бриллиантах.

Чтобы загрузить встроенные датасеты нужно воспользоваться функцией

```
data(dataset, package = "PackageName")
```

где `dataset` – имя датасета в пакете. Если пакет загружен командой `library` или `require`, то аргумент `package` можно пропуст (значение по умолчанию `NULL`)

1.8.2 Импорт данных из файла формата `csv`

Наиболее простой способ импорта статистических данных – использовать внешние данные в формате `csv`-файла[8]. Формат `csv` представляет собой таблицу, сохранённую в текстовом формате по строкам, в которой значения в каждой строке разделяются специальными символами (запятая, пробел, знак табуляции, точка с занятой). При создании файла `csv` (например, в MS Excel) отдельно указывается как представлять десятичные дроби: с десятичной точкой или с десятичной запятой.

Для импорта данных из файла формата `csv` можно использовать встроенную функцию

```
read.csv(file, header = TRUE, sep = ",",
                dec = ".", quote = "\"")
```

с аргументами

`file`	имя файла или web-ссылка (строковая переменная)
`header`	если `TRUE`, то первая строка интерпретируется как название переменной
`sep`	разделитель между значениями в стоке: `";"` `","` `"\t"` `"␣"`
`dec`	разделитель десятичной дроби: `"."` `","`
`quote`	символ цитирования (строковые переменные)

[8] csv = Comma Separated Values

1.8.3 Импорт данных из файла формата MS Excel

В R нет встроенных функций для импорта данных из файлов формата MS Excel. Поэтому необходимо воспользоваться сторонними пакетами.

Вариант №1 Пакет `XLConnect` позволяет работать с файлами MS Excel. В этом пакете есть функция для чтения данных из файла

```
library("XLConnect")
readWorksheetFromFile(file, sheet, startRow = 0,
        startCol = 0, endRow = 0, endCol = 0,
        header = TRUE, ...)
```

с аргументами

`file`	имя файла или web-ссылка (строковая переменная)
`sheet`	имя или номер листа в файле
`startRow`	номер начальной строки
`startCol`	номер начальной колонки
`endRow`	номер последней строки
`endCol`	номер последней колонки
`header`	если `TRUE`, то первая строка интерпретируется как название переменной
`region`	диапазон на листе в формате 'A1:C5'

startRow, startCol, endRow, endCol по умолчанию равны 0. Это означает, что диапазон выбирается автоматически.

Вариант №2 Пакет `xlsx` также позволяет работать с файлами MS Excel. В нём, в частности, реализована функция

```
library("xlsx")
read.xlsx(file, sheetIndex, sheetName = NULL,
        rowIndex = NULL, startRow = NULL, endRow = NULL,
        colIndex=NULL, header=TRUE, ...)
```

Основные аргументы

`file`	имя файла или web-ссылка (строковая переменная)
`sheetIndex`	номер листа
`sheetName`	имя листа в файле (строковая переменная)
`rowIndex`	числовой вектор с указанием номеров строк
`startRow`	номер начальной строки
`endRow`	номер последней строки
`colIndex`	числовой вектор с указанием номеров строк
`header`	если `TRUE`, то первая строка интерпретируется как название переменной

1.9 Оценивание линейной регрессии

Оценивание линейной регрессии методом наименьших квадратов осуществляется с использованием функции[9]

```
lm(formula, data, subset, weights, ...)
```

Основные аргументы

`formula`	спецификация регрессии (объект класса formula)
`data`	дата фрейм, на котором оценивается модель
`subset`	(опционально) используется, если нужно оценить модель не по всему дата фрейму, а только по его части
`weights`	(опционально) вектор весов для метода WLS

В результате выполнения функции создаётся объект типа lm, который представляет собой список, содержащий информацию о подогнанной модели. Для вывода протокола оценивания можно использовать функцию `summary()`.

Спецификация регрессии (объект класса formula) указывается по правилу:

- Регрессия с константой

$$\begin{pmatrix}\text{Зависимая}\\ \text{переменная}\end{pmatrix} \sim 1 + \text{регрессор1} + \text{регрессор2} + \cdots$$

  или более кратко

$$\begin{pmatrix}\text{Зависимая}\\ \text{переменная}\end{pmatrix} \sim \text{регрессор1} + \text{регрессор2} + \cdots$$

[9] lm = Linear Model

- Регрессия без константы

$$\begin{pmatrix}\text{Зависимая}\\ \text{переменная}\end{pmatrix} \sim 0 + \text{регрессор1} + \text{регрессор2} + \cdots$$

Названия факторов указываются как в дата фрейме, на котором оценивается модель.

Нужно отметить следующие факты:

- Если зависимая переменная или регрессоры в модели берутся с логарифмами, то в спецификацию модели нужно вставить функцию `log()`. Например

```
log(y)~log(x1)+log(x2)+x3
```

- Если в число объясняющих переменных включён регрессор в степени k (обычно квадрат k=2), то в функции `lm` в спецификации нужно указать `I(...^k)`. Например

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

- Если в число объясняющих переменных включено произведение регрессоров, то в функции `lm` это произведение нужно включить в `I(...)`. Например

```
y~age+male+I(age*male)
```

Пример. Пусть дата фрейм `MyDFrame` содержит переменные $y, x1, x2, x3$ и оцениваемая регрессия выглядит так:

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + u_i,$$

Тогда оценивание осуществляется командой

```
mymodel <- lm(formula=y~x1+x2+x3, data=MyDFrame)
```

Вывод протокола оценивания

```
summary(mymodel)
```

Объект типа lm (список) ссодержит информацию об оцененной модели. В частности

`coefficients`	коэффициенты модели
`residuals`	остатки
`fitted`	предсказанные значения

Пример. Для модели из предыдущего примера получить коэффициенты, остатки и предсказанные значения можно командами

```
mymodel$coefficients
mymodel$residuals
mymodel$fitted
```

Альтернативно можно использовать функции

```
coef(object)
coefficients(object)
residuals(object)
resid(object)
fitted(object)
fitted.values(object)
```

Вычислить показатели информационных критериев Акаике и Байесовского (Шварца) можно с помощью функций

```
AIC(object, ..., k = 2)
BIC(object, ...)
```

Они позволяют получить показатели информационных критериев сразу для нескольких оценённых моделей. Параметр `k` – это коэффициент при штрафном слагаемом (по умолчанию равен 2 для стандартного критерия Акаике).

Для тестирования гипотез о коэффициентах необходимо оценить матрицу вариации (ковариационную матрицу) вектора оценок коэффициентов. OLS-оценку этой ковариационный матрицы

$$\widehat{\mathrm{Var}}(\hat{\boldsymbol{\beta}}_{OLS}) = s^2(\mathbf{X}'\mathbf{X})^{-1}$$

можно получить с помощью функции

```
vcov(object)
```

Для вычисления робастных[10] (HC и HAC) оценок ковариационной матрицы необходимо использовать сторонние пакеты. Рассмотрим два примера:

- В пакете `sandwich`[11] функции

  ```
  library("sandwich")
  vcovHC(x, type = c("HC3", "const", "HC", "HC0",
          "HC1", "HC2", "HC4", "HC4m", "HC5"), ...)
  vcovHAC(x, order.by = NULL, ...)
  ```

  вычисляют соотвественно HC- и HAC-оценки ковариационной матрицы вектора оценок коэффициентов.

- В пакете `car`[12] функция

  ```
  library("car")
  hccm(model, type = c("hc3", "hc0", "hc1", "hc2",
          "hc4"), ...)
  ```

  вычисляет HC-оценку матрицы вариации вектора оценок коэффициентов

Основные аргументы функций:

`x, model`	объект класса lm
`type`	тип HC-оценки
`order.by`	по какой переменной необходимо упорядочить данные (для HAC-оценки)

1.10 Прогнозирование по оценённой модели линейной регрессии

Прогнозирование для оценённой регрессии (объекта класса lm) осуществляется с помощью функции

[10] HC = Heteroskedasticity Consistent; HAC = Heteroskedasticity-Autocorrelation Consistent

[11] Оценки типа HC и HAC ковариационной матрицы вектора оценок коэффициентов часто называются сэндвич-оценками

[12] car = Companion to Applied Regression

```
predict(object, newdata,
        interval = c("none", "confidence", "prediction"),
        level=0.95, ...)
```

Основные аргументы

`object`	оцененная модель (объект класса lm)
`newdata`	значения регрессоров (в формате дата фрейма), для которых вычисляются прогнозы
`interval`	доверительный интервал для прогноза
`level`	доверительная вероятность (для интервалов)

Пример. Рассмотрим модель из предыдущего примера. Пусть необходимо построить прогноз для двух наборов регрессоров:

$$\mathbf{x}_1 = \begin{pmatrix} 2.3 & 9.3 & 5.7 \end{pmatrix}$$
$$\mathbf{x}_2 = \begin{pmatrix} 4.3 & 4.5 & 6.3 \end{pmatrix}$$

В R сначала необходимо создать дата фрейм с двумя наборами значений регрессоров (пусть `newdf`) и затем вычисляем (точечный) прогноз

```
newdf <- data.frame(x1 = c(2.3, 4.3), x2 = c(9.3, 4.5),
                          x3 = c(5.7, 6.3) )
mypredict <- predict(mymodel, newdata = newdf)
```

1.11 Тестирование гипотез о коэффициентах

1.11.1 t-тест для коэффициентов регрессии

Протокол оценивания линейной регрессии (объекта класса lm), выводимой с использованием функции `summary`, содержит в частности

- OLS-оценки коэффициентов
- стандартные ошибки коэффициентов
- t-статистики для значимости коэффициентов
- P-значения для t-статистик

Если необходимы только результаты t-теста для коэффициентов или доверительные интервалы, то можно использовать соотвествующие функции из пакета `lmtest`

```
library("lmtest")
coeftest(x, vcov. = NULL, ...)
coefci(x, parm = NULL, level = 0.95, vcov. = NULL, ...)
```

Основные аргументы

`x`	оцененная модель (объект класса lm)
`vcov.`	оценка матрицы вариации вектора оценок коэффициентов (по умолчанию OLS-оценка)
`parm`	можно явно указать для каких переменных выводить доверительные интервалы
`level`	доверительная вероятность (для интервалов)

Аргумент `vcov.` следует использовать, если необходимо вычислить робастные тестовые статистики (например, для регрессии с гетероскедастичностью или с серийной корреляцией).

Пример. Команда

```
library("lmtest")
coeftest(mymodel, vcov. = vcovHC(mymodel) )
```

выводит результаты t-теста для коэффициентов с использованием робастных тестовых статистик.

1.11.2 F-тест для коэффициентов регрессии

1. Для тестирования совместной значимости для нескольких коэффициентов (F-тест) можно использовать функцию[13]

```
anova(object, ...)
```

Аргументы функции `anova` – объекты класса lm. Функция вычисляет тестовую F-статистику, соответствующие степени свободы F-распределения и P-значение. Важно отметить, что F-статистика в данном случае вычисляется по OLS-оценке матрица вариации оценок коэффициентов.

Рассмотрим два случая:

[13] anova = ANalysis Of VAriations

- если указать в аргументе только один объект, то тестируется значимость каждого коэффициента;
- если тестируется совместная значимость нескольких коэффициентов, то нужно указать два объекта класса lm: регрессию с ограничениями («короткую») и без ограничений («длинную»).

Пример. Пусть оценена регрессия

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \beta_4 x_{4i} + \beta_5 x_{5i} + u_i.$$

Для тестирования гипотезы

$$H_0 : \beta_4 = \beta_5 = 0$$

можно использовать F-тест. Он осуществляет так

```
mymodel.short <- lm(y~x1+x2+x3, data = MyDFrame)
mymodel.long <- lm(y~x1+x2+x3+x4+x5, data = MyDFrame)
anova(mymodel.short, mymodel.long)
```

2. Основной недостаток функции `anova` – это невозможность использовать робастный вариант F-теста, например, для линейной регрессии с гетероскедастичностью.

Если в этом есть необходимость, то следует использовать функцию

```
library("lmtest")
waldtest(object, ..., vcov = NULL, test = c("Chisq", "F"), ...)
```

которая выполняет (асимптотический) F-тест или тест Вальда. Основные аргументы

`object`	оцененная модель (объект класса lm) или объект класса formula (в этом случае в аргументе нужно указать еще и дата фрейм `data=...`)
`vcov`	оценка матрицы вариации вектора оценок коэффициентов (по умолчанию OLS-оценка)
`test`	какую тестовую статистику вычислять: асимптотическую χ^2 или точную F

Рассмотрим два случая:

- если указать в аргументе только один объект, то тестируется значимость регрессии «в целом» (в отличие от `anova`);
- если тестируется совместная значимость нескольких коэффициентов, то нужно указать два объекта класса lm: регрессию с ограничениями («короткую») и без ограничений («длинную»).

Пример. Следующий код аналогичен предыдущему и тестирует совместную значимость влияния факторов x4 и x5

```
library("lmtest")
mymodel.short <- lm(y~x1+x2+x3, data = MyDFrame)
mymodel.long <- lm(y~x1+x2+x3+x4+x5, data = MyDFrame)
waldtest(mymodel.short, mymodel.long, test = "F")
```

3. Для тестирования произвольных линейных ограничений на коэффициенты необходимо использовать пакет `car` и функцию

```
library("car")
linearHypothesis(model, hypothesis.matrix,
        test = c("F", "Chisq"), vcov. = NULL,
        white.adjust = c(FALSE, TRUE, "hc3",
        "hc0", "hc1", "hc2", "hc4"), ...)
```

Основные аргументы

`model`	оцененная модель (объект класса lm)
`hypothesis.matrix`	матрица или вектор линейных ограничений на коэффициенты
`vcov.`	оценка матрицы вариации вектора оценок коэффициентов (по умолчанию OLS-оценка)
`white.adjust`	альтернатива vcov.
`test`	какую тестовую статистику вычислять: асимптотическую χ^2 или точную F

Разберем на примере

Пример. Пусть оценена регрессия

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \beta_4 x_{4i} + \beta_5 x_{5i} + u_i.$$

и необходимо тестировать гипотезу

$$H_0 : \ \beta_3 + 0.5\beta_4 = 0, \ \beta_5 = 0$$

Тогда код на языке R выглядит так

```
library("car")
mymodel <- lm(y~x1+x2+x3+x4+x5, data = MyDFrame)
linearHypothesis(mymodel, c("x3+0.5x4=0","x5=0"),
        test = "F")
```

Для тестирования гипотезы

$$H_0 : \ \beta_1 + \beta_2 + \beta_3 + \beta_4 + \beta_5 = 1$$

код на языке R выглядит так

```
library("car")
mymodel <- lm(y~x1+x2+x3+x4+x5, data = MyDFrame)
linearHypothesis(mymodel, c("x1+x2+x3+x4+x5=1"), test = "F")
```

Пример. Если необходимо использовать робастную тестовую статистику, то будет так

```
library("car")
library("sandwich")
mymodel <- lm(y~x1+x2+x3+x4+x5, data = MyDFrame)
linearHypothesis(mymodel, c("x3+0.5x4=0","x5=0"),
        test = "F", vcov. = vcovHC(mymodel) )
```

или равносильно

```
library("car")
mymodel <- lm(y~x1+x2+x3+x4+x5, data = MyDFrame)
linearHypothesis(mymodel, c("x3+0.5x4=0","x5=0"),
        test = "F", white.adjust = "hc3")
```

1.12 Тесты на адекватность, гетероскедастичность, автокорреляцию

Мультиколлинеарность Для проверки модели на мультиколлинеарность и вычисления показателей VIF необходим пакет `car` и функция

```
library("car")
vif(mod, ...)
```

где `mod` – объект класса lm.

Выбор спецификации Для RESET-теста на «правильную спецификацию» (теста Рамсея) необходим пакет `lmtest` и функция

```
library("lmtest")
resettest(formula, power = 2:3, data, ...)
```

Основные аргументы

`formula`	спецификация модели (объект класса formula) или оцененная модель (объект класса lm)
`power`	набор степеней $\hat{y}_i^s$, добавляемых в уравнение
`data`	дата фрейм, на котором оценивается модель (если модель задается спецификацией)

Гетероскедастичность Рассмотрим основные тесты на гетероскедастичность, реализованные в пакете `lmtest`

Тест Бреуша-Пагана реализован в функции

```
library("lmtest")
bptest(formula, varformula = NULL, studentize = TRUE, data)
```

Основные аргументы

`formula`	спецификация модели (объект класса formula) или оцененная модель (объект класса lm)
`varformula`	можно явно указать спецификацию вспомогательной регрессии ($=\sim \cdots$). По умолчанию: на все регрессоры модели
`studentize`	если TRUE, то будет использован робастный (стьюдентизированный) вариант теста
`data`	дата фрейм, на котором оценивается модель (если модель задается спецификацией)

Для реализации теста Уайта необходимо в параметре varformula явно указывать зависимость от регрессоров.

Тест Голдфелда-Квандта реализован в функции

```
library("lmtest")
gqtest(formula, point = 0.5 , fraction = 0,
        alternative = c("greater", "two.sided", "less"),
        order.by = NULL, data)
```

Основные аргументы

`formula`	спецификация модели (объект класса formula) или оцененная модель (объект класса lm)
`point`	если меньше 1, то указывает в каком отношении разбивать выборку на две части. Иначе (если >1) индекс наблюдения, которые разбивает выборку на две части
`fraction`	если меньше 1, то указывает какую долю центральных наблюдения отбросить. Иначе число число отбрасываемых наблюдения (если >1)
`order.by`	по какой переменной упорядочить данные
`data`	дата фрейм, на котором оценивается модель (если модель задается спецификацией)

Серийная корреляция LM-тест на автокорреляцию (тест Бреуша-Годфри) реализован в пакете `lmtest` с использованием функции

```
library("lmtest")
bgtest(formula, order = 1, order.by = NULL,
        type = c("Chisq", "F"), data, ...)
```

Основные аргументы

`formula`	спецификация модели (объект класса formula) или оцененная модель (объект класса lm)
`order`	порядок автокорреляции
`order.by`	по какой переменной упорядочить данные
`type`	тип тестовой статистики
`data`	дата фрейм, на котором оценивается модель (если модель задается спецификацией)

Тест Дарбина-Уотсона реализован в нескольких пакетах. Рассмотрим два примера,

Первый пример теста

```
library("lmtest")
dwtest(formula, order.by = NULL,
        alternative = c("greater", "two.sided", "less"),
        data, ...)
```

Основные аргументы

`formula`	спецификация модели (объект класса formula) или оцененная модель (объект класса lm)
`order.by`	по какой переменной упорядочить данные
`alternative`	альтернатива нулевой гипотезе
`data`	дата фрейм, на котором оценивается модель (если модель задается спецификацией)

Стоит обратить внимание на тот факт, что по умолчанию берется односторонняя альтернатива $H_1 : \rho > 0$.

Второй пример

```
library("car")
durbinWatsonTest(model,
        alternative = c("two.sided", "positive", "negative"))
```

Основные аргументы

`model`	оцененная модель (объект класса lm)
`alternative`	альтернатива нулевой гипотезе

В данной реализации альтернатива по умолчанию двусторонняя.

1.13 Пример: зарплатное уравнение

Используем пакет `wooldridge`, который содержит данные для упражнений из учебника J. Wooldridge «Introductory Econometrics: A Modern Approach». Возьмём набор данных `card`, в котором есть, в частности, переменные

wage	почасовая оплата
age	возраст
south	бинарная, =1 если живет на юге
married	бинарная, =1 если женат

1. Рассмотрим модель регрессии

$$\ln(\text{wage}) = \beta_0 + \beta_1 \text{age} + \beta_2 \text{age}^2 + \beta_3 \text{south} + \beta_4 \text{married} + u$$

Загрузим необходимые пакеты (для тестирования модели) и оценим регрессию

```
library("wooldridge")
library("lmtest")
library("sandwich")
library("car)
data(card, package="wooldridge")
log_wage_eq<-lm(formula=log(wage)~age+I(age^2)+south+married,
        data=card)
```

Далее фиксируем уровень значимость 1%. Результаты t-теста для коэффициентов

```
> coeftest(log_wage_eq)

t test of coefficients:

               Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)  3.69654563  0.69060523   5.3526 9.323e-08 ***
age          0.15916673  0.04834351   3.2924  0.001005 **
I(age^2)    -0.00216737  0.00083838  -2.5852  0.009780 **
south       -0.25139969  0.01494638 -16.8201 < 2.2e-16 ***
married     -0.03242516  0.00364954  -8.8847 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Все коэффициента значимы. Тестируем значимость регрессии «в целом»

```
> waldtest(log_wage_eq)
Wald test

Model 1: log(wage) ~ age + I(age^2) + south + married
Model 2: log(wage) ~ 1
  Res.Df Df   F    Pr(>F)
1   2998
2   3002 -4 166 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

F-статистика равна 166, регрессия значима. Выполним тест Рамсея (с включением степеней предсказанных значений от 2 до 4)

```
> resettest(log_wage_eq, power=2:4)

RESET test

data:  log_wage_eq
RESET = 2.7821, df1 = 3, df2 = 2995, p-value = 0.03956
```

Гипотеза о спецификации не отвергается.

Осуществим проверку на гетероскедастичность с использованием теста Бреуша-Пагана:

```
> bptest(log_wage_eq)

studentized Breusch-Pagan test

data:  log_wage_eq
BP = 38.184, df = 4, p-value = 1.027e-07
```

Так как тест указывает на наличие гетероскедастичности, то следует использовать робастные тестовые статистики. Робастный t-тест для коэффициентов выглядит так

```
> hccm <- vcovHC(log_wage_eq)
> coeftest(log_wage_eq, vcov.=hccm)

t test of coefficients:

              Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)  3.69654563  0.69367532   5.3289 1.061e-07 ***
age          0.15916673  0.04868844   3.2691  0.001091 **
I(age^2)    -0.00216737  0.00084736  -2.5578  0.010583 *
south       -0.25139969  0.01527413 -16.4592 < 2.2e-16 ***
married     -0.03242516  0.00382663  -8.4736 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Как видно, влияние квадрата возраста стало незначимо. Робастный вариант теста на значимость регрессии

```
> waldtest(log_wage_eq, vcov=hccm)
```

```
Wald test

Model 1: log(wage) ~ age + I(age^2) + south + married
Model 2: log(wage) ~ 1
  Res.Df Df      F    Pr(>F)
1   2998
2   3002 -4 169.22 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Сделаем тест, выясняющий, есть ли значимое влияние возраста на зарплату.

```
> log_wage_eq_aux<-lm(formula=log(wage)~south+married, data=card)
> waldtest(log_wage_eq_aux, log_wage_eq, vcov=hccm)
Wald test

Model 1: log(wage) ~ south + married
Model 2: log(wage) ~ age + I(age^2) + south + married
  Res.Df Df      F    Pr(>F)
1   3000
2   2998  2 101.72 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Другой способ

```
> linearHypothesis(log_wage_eq, c("age=0", "I(age^2)=0"),
+                  test="F", white.adjust = "hc3")
Linear hypothesis test

Hypothesis:
age = 0
I(age^2) = 0

Model 1: restricted model
Model 2: log(wage) ~ age + I(age^2) + south + married

Note: Coefficient covariance matrix supplied.
```

```
  Res.Df Df      F    Pr(>F)
1   3000
2   2998  2 101.72 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Тестовая статистика равна 101.72. Влияние возраста значимо.

2. Рассмотрим теперь модель

$$\ln(\text{wage}) = \beta_0 + \beta_1 \text{age} + \beta_2 \text{south} + \beta_3 (\text{age} * \text{south}) + \beta_4 \text{married} + u$$

Оценим модель

```
log.wage.eq <- lm(log(wage)~age+south+I(age*south)+married,
                        data=card)
```

и потестируем модель на гетероскедастичность

```
> bgtest(log.wage.eq)

Breusch-Godfrey test for serial correlation of order up to 1

data:  log.wage.eq
LM test = 48.458, df = 1, p-value = 3.374e-12
```

Тест указывает на гетероскедастичность. Следовательно, нужно рассматривать робастный t-тест

```
> coeftest(log.wage.eq, vcov.=vcovHC(log.wage.eq))

t test of coefficients:

                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     5.2968268  0.0859488 61.6277  < 2e-16 ***
age             0.0405630  0.0029835 13.5959  < 2e-16 ***
south           0.1927689  0.1423088  1.3546  0.17565
married        -0.0328176  0.0037940 -8.6498  < 2e-16 ***
I(age * south) -0.0157908  0.0050955 -3.0990  0.00196 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Незначим только коэффициент β_2.

Потестируем значимость влияния географического фактора, т.е. гипотезу

$$H_0 : \beta_3 = \beta_3 = 0$$

используя робастный F-тест (тест Вальда)

```
> linearHypothesis(log.wage.eq, c("south=0", "I(age * south)=0"),
+                  test="F", white.adjust = "hc3")
Linear hypothesis test

Hypothesis:
south = 0
I(age * south) = 0

Model 1: restricted model
Model 2: log(wage) ~ age + south + I(age * south) + married

Note: Coefficient covariance matrix supplied.

  Res.Df Df      F    Pr(>F)
1   3000
2   2998  2 136.03 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Тестовая статистика равна 136.03, влияние фактора `south` значимо.