Прикладной статистический анализ

Парная регрессия

Рассмотрим (ненастоящие) данные, описывающие количество прогулов и средний бал набора студентов за один из семестров:

```
students <- read.csv("students.csv")
plot(students)
```

На получившейся диаграме рассеяния отчетливо видна обратная зависимость между количеством пропусков и успеваемостью. Напомним, что мы можем вычислить долю *линейной* зависимости посчитав коэффициент корреляции $r_{x,y} = \frac{\overline{xy} - \overline{xy}}{s_x s_y}$, где $s_x = \sqrt{\overline{x^2} - \overline{x}^2}$, $s_y = \sqrt{\overline{y^2} - \overline{y}^2}$ - выборочные дисперсии.

Упражнение 1

Реализуйте собственные функции $myVar(x)$ и $myCor(x, y)$, вычисляющие выборочную дисперсии и коэффициент корреляции.

Упражнение 2

Используя написанные Вами функции посчитайте коэффициент корреляции для количества пропусков и среднего бала студентов. Изменятся ли эти величины, если средний бал всем понизить на единицу, а количество пропусков увеличить на 20%? Почему? Сравните еорреляцию, посчитанную Вами с ответом встроенной функции `cor()` .

Парная регрессия

Значение коэффициента корреляции в данных студентов указывает на наличие некоторой линейной зависимости между двумя представленными величинами: доля объяснённой дисперсии $R^2 = r^2_{x,y}$ равна примерно 0.45.

Давайте попробуем найти прямую $y = \beta_0 + \beta_1 x$, наилучшим образом описывающую наши данные. Напомним, что в случае парной регрессии для нахождения наиболее подходящих оценок для коэффициентов β_0, β_1 применяются формулы: $\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x}$, $\hat{\beta}_1 = r_{x,y} \frac{s_y}{s_x}$.

Упражнение 3

Напишите функцию $myLm(x, y)$, которая возвращает вектор, состоящий из двух элементов с именами `b0` и `b1`, равных $\hat{\beta}_0$ и $\hat{\beta}_1$ соответственно. Нарисуйте значения датафрейма `students` и прямую с посчитанными Вами коэффициентами на одном графике.

Упражнение 4

Линейная регрессия применима не ко всем возможным данным и её не стоит применять вслепую: всегда необходимо сначала удостовериться в применимости линейной модели. Первым шагом может служить визуализация ваших данных на графике.

Квартетом Энскомба называют известные 4-е набора данных, которые разительно отличаются друг от друга внешне, но имеют почти идентичные регрессионные характеристики (выборочное среднее, выборочная дисперсия, корреляция, регрессионная линия, коэффициент детерминации).

В R по умолчанию задана переменная `anscombe`, содержащая эти данные (столбцы x1 & y1, x2 & y2, x3 & y3, x4 & y4). нарисуйте эти наборы на графиках, посчитайте для каждого из них регрессионные характеристики и убедитесь в их близости.

Встроенные функции для нахождения коэффициентов линейной регрессии

Для нахождения коэффициентов линейной регрессии в R может быть использована функция `lm()` (от linear model). В качестве примера, применим её к первому набору данных из квартета Энскомба:

```
ansX <- anscombe[["x1"]]
ansY <- anscombe[["y1"]]
lm(ansY ~ ansX)
```

Мы не будем подробно разбирать запись `ansY ~ ansX` и скажем только, что она задаёт **формулу**, указывающую, что мы хотим предсказывать значения, относящиеся к `ansY` по `ansX`.

Вывод функции должен содержать два числа, помеченные как '(Intercept)' и `ansX`. *Intercept* обозначает свободный коэфициент регрессионной прямой (β_0), *ansX* же отмечен коэффициент, на который следует умножать значения регрессора (β_1). Если вы корректно выполнили предшествующие упражнения, то ваши коэффициенты должны совпасть с выводом `lm()` (с точностью до некоторой погрешности).

Функция `lm()` не просто выводит текст на экран, а возвращает некоторый объект, описывающий полученную модель. Этот объект мы можем использовать для доступа к её числовым характеристикам, например, самим коэффициентам:

```
fit <- lm(ansY ~ ansX)
fit$coefficients
```

или для вывода более полной информации:

```
summary(fit)
```

Вывод `summary()` для линейной модели

Подробно разберём вывод функции `summary()` для линейной регрессии, чтобы приведенные числа не казались нам "магическими".

```
Residuals:
     Min       1Q   Median       3Q      Max 
-1.92127 -0.45577 -0.04136  0.70941  1.83882 
```

Здесь представлены несколько показательных значений остатков $\hat{\varepsilon}$ для нашей модели. Напомним, что остатками называют расхождение между реальными данными и предсказаниемнашей модели:
$\hat{\varepsilon_i} = y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i)$. Так как регрессионная прямая будет проходить прямо через все точки данных (x_i, y_i) только в вырожденном случае, - наличие существенных остатков ожидаемо нормально.

Приведенные пять значений демонстрируют их величину с помощью минимального и максимального остатков, а также остатков, соответствующим на квантилям 0.25, 0.5 (медиана) и 0.75.

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)   3.0001     1.1247   2.667  0.02573 *
ansX          0.5001     0.1179   4.241  0.00217 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Здесь приводятся данные по каждому из коэффициентов регрессии. Первое число (*Estimate*) представляет собой просто полученную точечную оценку для коэффициентов, котоую мы научились получать и сами.

Второе число (*Std. Error*) соответствует стандартной ошибке нашей оценки. Чтобы понять, что здесь имеется ввиду, вспомним матричную форму нашей регрессионной модели:

$$\vec{y} = X\vec{\beta} + \vec{\varepsilon},$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1k} \\ 1 & x_{21} & x_{22} & \dots & x_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{nk} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

Здесь $\vec{y}$ и $\vec{\varepsilon}$ - вектора размера n, равного количеству наблюдений (у нас сейчас `length(ansY)`), $\vec{\beta}$ - вектор длины $p = k + 1$ (у нас 2), а матрица X имеет размерность nxp. Также известно, что $\vec{\varepsilon}$ - это вектор независимы нормальных случайных величин с нулевым средним и одниаковой дисперсией. Т.е. вектор имеет многомерное нормальное распределение $N(0, \sigma^2 I)$, где I - единичная матрица. Векто ор $\vec{y}$ же тогда имеет распределение $N(X\vec{\beta}, \sigma I)$.

В этих обозначениях оценку методом наименьших квадратов можно выразить как $\vec{\beta} = (X^T X)^{-1} X^T \vec{y}$. Но так как матрица ковариации $\vec{y}$ равна $\sigma^2 I$, то матрица ковариации вектора $\vec{\beta}$ будет равна

$$((X^T X)^{-1} X^T)\sigma^2 I((X^T X)^{-1} X^T)^T = \sigma^2 (X^T X)^{-1} X^T X (X^T X)^{-1} = \sigma^2 (X^T X)^{-1} =$$

$$\begin{bmatrix} \mathbb{D}\beta_0 & cov(\beta_0, \beta_1) & \dots & cov(\beta_0, \beta_k) \\ cov(\beta_1, \beta_0) & \mathbb{D}\beta_1 & \dots & cov(\beta_1, \beta_k) \\ \vdots & \vdots & \ddots & \vdots \\ cov(\beta_k, \beta_0) & cov(\beta_k, \beta_1) & \dots & \mathbb{D}\beta_k \end{bmatrix}$$

Отсюда и берутся оценки дисперсии, указанные в *Std. Error*. И мы можем вычислить их самостоятельно, найдя $(X^T X)^{-1}$ и умножив их на оценку $\hat{\sigma}^2 = \frac{1}{n-p} \hat{\vec{\varepsilon}}^T \hat{\vec{\varepsilon}} = \frac{1}{n-p} (\vec{y} - X\hat{\vec{\beta}})^T (\vec{y} - X\hat{\vec{\beta}})$:

```
n <- length(ansY)
y <- matrix(ansY, ncol = 1)
# Add column of 1s to the `ansX` data for \beta_0
X <- matrix(c(rep(1, n), ansX), ncol = 2)
beta <- matrix(fit$coefficients, ncol = 1)
p <- length(beta)
varianceEstimate <- sum((y - X %*% beta)^2) / (n - p)
covMatrix <- varianceEstimate * solve(t(X) %*% X)
```

Эта матрица (`covMatrix`) ещё не содержит требуетмые числа, ведь в выводе `summary()` приводятся стандартные ошибки, т.е. корни из дисперсий:

```
sqrt(diag(covMatrix))
```

Эти числа уже должны совпадать со значениями *Std. Error*, посчитанными с помощью `summary()` .

Значение *t value* есть *Estimate* / *Std. Error* и будет иметь распределение Стьюдента. *Pr(>|t|)* - основанное на этом значении p-value в тесте Стьюдента. Эти значения указывают насколько значимы для предсказания соответствующие коэффициенты. Указанные напротив них звездочки являются просто визуальным индикатором достигнутой значимости, а их значения описаны в разделе *Signif. codes*.

```
Residual standard error: 1.237 on 9 degrees of freedom
Multiple R-squared:  0.6665,    Adjusted R-squared:  0.6295
```

Residual standard error есть ни что иное, как оценка среднеквадратичного отклонения ошибок дисперсии, оторое можно получить из уже посчитанной нами дисперсии:

```
sqrt(varianceEstimate)
# 9 degrees of freedom
n - p
```

Multiple R-squared в нашем случае это просто R^2, - квадрат корреляции данных:

```
cor(ansX, ansY)^2
```

Более подробно об этом значении (и об *Adjusted R-squared*) мы поговорим, когда будем обсуждать множественную регрессию.

F-статистика

Из лекций нам известно, что F-статистика для выбора между гипотезами

$$H_o : C\vec{\beta} = \vec{d},$$

$$H_a : C\vec{\beta} \neq \vec{d};$$

определяется как

$$\frac{\Delta SSE}{SSE} = \frac{SSE_{H_o} - SSE}{SSE} \geqslant 0,$$

где $\hat{\vec{\varepsilon}}^T \hat{\vec{\varepsilon}}$, а

$$SSE_{H_o} = (C\hat{\vec{\beta}} - d)^T \left(C(X^T X)^{-1} C^T\right)^{-1} (C\hat{\vec{\beta}} - d).$$

Значение *F-statistic* в выводе соответствует гипотезе $\beta_1 = \ldots = \beta_k = 0$, в нашем случае просто: $\beta = 1$.

Расчитаем эту статистику для проверки гипотезы $\beta_1 = 0$. Её матричный вид:

$$\begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad d = 0;$$

Чтобы расчитать F-статистику сначала зададим входящие в её определение данные C и d в R:

```
C <- matrix(c(0, 1), ncol = 2)
d <- 0
```

тогда F-статистику можно расчитать с помощью:

```
side <- C %*% beta - d
D.SSE <- t(side) %*% solve(C %*% solve(t(X) %*% X) %*% t(C)) %*% side
SSE <- t(y - X %*% beta) %*% (y - X %*% beta)
(F.stat <- (D.SSE/SSE) * (n - p))
```

что совпадает с выводом R.

Более того, в нашем простом случае эта гипотеза соответствует проверке значимости параметра β_1 и в этом случае *F-статистика* есть квадрат статистики Стьюдента для β_1, которую мы уже рассматривали ранее:

```
sqrt(F.stat)
```

Множественная регрессия

Для множественной регрессии нам понадобится другой набор данных, содержащий в себе больше переменных.

Например рассмотрим данные, описывающие зависимость школьной успеваймости ребёнка *kid_score* от ряда параметров его/её матери:

Variable	Description
kid_score	успеваемость ребенка
mom_hs	Закончила ли старшую школу?
mom_iq	Её iq
mom_work	Работала ли в течении дошкольного возраста ребенка?
mom_age	Возраст, в котором она завела ребенка

```
cognitive <- read.csv("cognitive.csv")
```

Для того, чтобы заставить `lm()` предсказывать *kid_score* по нескольким переменным (например, *mom_iq* и *mom_age*), можно просто перечислить их через знак '+': `kid_score ~ mom_iq + mom_age` . Также можно просто использовать имена столбцов, если мпередать `lm()` наш датафрейм в качестве параметра *data*:

```
fit <- lm(kid_score ~ mom_iq + mom_age, data = cognitive)
summary(fit)
```

Здесь мы видим практически идентичный предыдущему вывод, только оценка теперь приведены для трёх коэффициентов.

Качественные переменные

Отдельно поговорим о переменных *mom_hs* и *mom_work*, которые мы не стали использовать при предсказании в предыдущий раз. Их отличие в том, что вместо обычного численного значения они принимают лишь конечный набор значений: *no* и *yes*. Чтобы применять регрессию к таким переменным они будут преобразованы в численные переменные следующим образом:

1. Одно из них (в нашем случае *no*) будет выбрано за базовое значение, соответствующее нулю;

2. Для каждого другого значения (*yes*) будет введена своя переменная, равная единице в записях таблицы, содержащих её значение.

Посмотрим как это будет выглядеть на нашем простом примере:

```
fit.alt <- lm(kid_score ~ mom_hs + mom_work, data = cognitive)
summary(fit.alt)
```

Как мы видим, вместо *mom_hs* и *mom_work*, функция `lm()` работала с переменными *mom_hsyes* и *mom_workyes*. Соответствующие им коэффициенты (`11.108` и `2.938`) стоит трактовать как средняя "надбавка" для успеваймости ребенка, если известно, что мать работала или имеет лучшее образование.

Т.е., *(Intercept)* соответствует средней успеваемости ребенка, если его мать не работала и не доучилась. Если же, например, известно, что она работала, но не доучилось, то к это следует добавить ещё балов: `75.653+11.108=86.761` .

Рассмотрим также какие у нас будут переменные, если у качественной переменной будет более двух значений на прмере связи iq близнецов (Сирил Берт использовал эти данные в своих исследованиях, но они уже давно считаются фальшивыми):

```
twins <- read.csv("twins.csv")
fit.twins <- lm(Foster ~ Biological + Social, data = twins)
summary(fit.twins)
```

Переменная *Social* здесь принимала 3 значения: *high*, *middle* и *low*. Как можно видеть из вывода, *high* было взято за базовое значение, а для остальных были введены свои переменные *Socialmiddle* и *Sociallow*. Они также могут принимать лишь значения `0` и `1` . При этом не более, чем одна из них может быть равна `1` (в таблице может быть указан лишь один класс), а еслиравенство всех нулю соответствует классу *high*.

Отбор переменных

Существует множество способов отбора переменных. Мы рассмотрим два основных: *forward selection* и *backwards elimination*. Оба отбора могут проводиться на основе разных характеристик, а мы, опяться же, остановимся на двух основных: *Adjusted R-squared* и p-значения.

Forward selection with Adjusted R-squared

Этот метод предписывает нам добавлять переменные по одной до тех пор, пока растет значение R^2 нашей модели (которое, как известно, должна описывать насколько хорошо наша

модель объясняет данные). Однако проблема в том, что R^2 от добавления новых переменных может только увеличиваться. Поэтому вместо него обычно рассматривают "поправленный" R^2 (*Adjusted R-squared*), в определение которого входит штраф на количество переменных:

$$1 - (1 - R^2)\frac{n - 1}{n - k - 1}$$

Рассмотрим вывод нашей линейной модели с использованием всех 4-ёх переменных и убедимся что приведённые значения обыного и "поправленного" R^2 совпадают:

```
fit.full <- lm(kid_score ~ mom_hs + mom_iq + mom_work + mom_age, data = cognitive)
summary(fit.full)
1 - (1 - 0.2171) * ((434 - 1)/(434 - 4 - 1))
```

Для, собственно, отбора переменных мы начнём с пробы моделей, содержащих по одной переменной:

```
summary(lm(kid_score ~ mom_hs, data = cognitive))
summary(lm(kid_score ~ mom_iq, data = cognitive))
summary(lm(kid_score ~ mom_work, data = cognitive))
summary(lm(kid_score ~ mom_age, data = cognitive))
```

Лучшее (наибольшее) значение *Adjusted R-squared* (`0.1991`) соответствует переменной *mom_iq*, поэтому мы сключим её в нашу модель. На втором шаге попробуем добавлять к ней по одной из оставшихся переменных:

```
summary(lm(kid_score ~ mom_iq + mom_hs, data = cognitive))
summary(lm(kid_score ~ mom_iq + mom_work, data = cognitive))
summary(lm(kid_score ~ mom_iq + mom_age, data = cognitive))
```

В этот раз наилучший результат (`0.2105`) показала *mom_hs*. Кроме того, это больше предыдучщего значения для только *mom_iq*: `0.2105 > 0.1991` . Поэтому на этом шаге мы отбираем переменные *mom_iq* и *mom_hs*. Теперь мы отобрали две переменных и у нас осталось ещё две, - на третьем шаге попробует добавлять их к нашей паре:

```
summary(lm(kid_score ~ mom_iq + mom_hs + mom_work, data = cognitive))
summary(lm(kid_score ~ mom_iq + mom_hs + mom_age, data = cognitive))
```

mom_work показала себя лучше всего и снова позволила улучшить значение *Adjusted R-squared* до `0.2109` . На четвертом шаге остается попробовать добавить последнюю переменную:

```
summary(lm(kid_score ~ mom_iq + mom_hs + mom_work + mom_age, data = cognitive))
```

Полученный результат в этотраз оказался меньше предыдущего: `0.2098 < 0.2109` , поэтому мы не станем включать `mom_age` и остановимся на *mom_iq*, *mom_hs* и *mom_work*.

Backwards elimination with p-values

Давайте теперь полностью поменяем подход: будем двигаться назад от самой полной модели и отбирать переменные по p-значению. Для этого нам нужно сначала решить, какой уровень значимости мы будем считать приемлимым для наших переменных. Это зависит от решаемой задачи, но мы остановимся на обычном требовании `< 0.05` .

Как и говорили выше, начнем с полной модели:

```
summary(lm(kid_score ~ mom_iq + mom_hs + mom_work + mom_age, data = cognitive))
```

Согласно выводу, переменные *mom_work* и *mom_age* не являются достаточно значимыми (*(Intercept)* является, но, заметим, что он всё равно никогда не рассматривается как самостоятельная переменная, которую можно выбросить). Но согласно алгоритму *backwards elimination* на каждом шаге выбрасывается лишь одна переменная, имеющая наибольшее p-значение. У нас это *mom_age*, - избавимся от неё и рассмотрим модель с тремя оставшимися переменными:

```
summary(lm(kid_score ~ mom_iq + mom_hs + mom_work, data = cognitive))
```

p-значение *mom_work* уменьшилось, но не достаточно, чтобы считать её значимой. Поэтому на втором шаге мы от неё избавляемся. На третьем шаге рассмотрим модель с оставшимися двумя переменными:

```
summary(lm(kid_score ~ mom_iq + mom_hs, data = cognitive))
```

Теперь все переменные являются значимыми! Это означает что мы остановимся на модели, содержащей только *mom_iq* и *mom_hs*.

Упражнение 5

Проведите *Forward selection with p-values* на тех же данных. В этом подходе на каждом шаге добавляется по (наиболее значимой) переменной до тех пор, пока включение новых не повлечет появления незначимых переменных.

Упражнение 6

Проведите *Backwards elimination with Adjusted R-squared* на тех же данных. В этом подходе начинают с полной модели и на каждом шаге выбрасывают по одной так, чтобы прирост

Adjusted R-squared был максимален. Останавливаются, когда удаление переменной не влечет роста *Adjusted R-squared*.