УДК 519.221 DOI: 10.25206/2307-5430-2023-10-64-68

ПРИМЕНЕНИЕ ИНСТРУМЕНТОВ PYTHON В КУРСЕ СТАТИСТИКИ

APPLYING OF PYTHON TOOLS IN A STATISTICS COURSE

А. А. Кузнецова
Московский государственный технический университет
им. Н. Э. Баумана, г. Москва, Россия

A. A. Kuznetsova
Bauman Moscow State Technical University, Moscow, Russia

Аннотация. Рассмотрены методические аспекты преподавания математической статистики в техническом вузе с использованием лабораторных работ. Приведены примеры задач, допускающих решения с использованием библиотек SciPy и NumPy языка Python. Обоснованы преимущества данного языка программирования перед другими компьютерными математическими системами. Рассмотрены задачи генерации данных из заданного распределения, построения доверительных интервалов, проверки гипотез, корреляционного анализа и некоторые другие.

Ключевые слова: математическая статистика; лабораторная работа; Python.

Abstract. The methodological aspects of teaching mathematical statistics in a technical university using laboratory work are considered. Examples of tasks that can be solved using the SciPy and NumPy libraries of the Python language are given. The advantages of this programming language over other computer mathematical systems are substantiated. The problems of generating data from a given distribution, constructing confidence intervals, testing hypotheses, correlation analysis, and some others are considered.

Keywords: mathematical statistics; laboratory work; Python.

Лабораторные работы в статистике являются важной компонентой обучения, так как позволяют применять различные методы статистического анализа на приближенных к реальным данных (большим данным, данным, содержащим шум, т.е. к данным, работа с которыми в рамках семинарских занятий затруднена из-за большого объема вычислений). Прибегая к помощи компьютерных математических систем (КМС) при проведении занятий по математической статистике, желательно использовать распространенные и свободно доступные программные продукты. Существует большое количество КМС, позволяющих проводить статистические вычисления: SPSS, SAS, R, Statistica, Statgraphyics, MS Excel и его аналоги (Libre Office, Open Office Calc). В частности, о возможностях применения MS Excel для проведения лабораторных работ по статистике см. [1], [4], [5].

Python является одним из наиболее популярных языков программирования, хорошо подходящим для решения статистических задач в силу ряда причин: простоты синтаксиса; большого количества библиотек и инструментов, которые могут быть использованы для решения задач и визуализации статистических данных; отсутствия требований к выполнению лицензионных соглашений при использовании; распространённости, благодаря которой студенты обычно уже знакомы с основами языка.

Настоящая статья посвящена описанию некоторых инструментов языка Python, которые могут быть использованы при проведении лабораторных работ по статистике. Мы рассмотрим задачи, в которых используются нормально распределённые данные.

NumPy и *SciPy* – это две наиболее мощные библиотеки для математических расчетов и вычислений. Мы рассмотрим в первую очередь возможности модуля *stats* библиотеки *SciPy*, а также некоторые инструменты библиотеки *NumPy*. Предварительно подключим нужные нам инструменты следующим образом:

```
import numpy as np
from scipy import stats
```

Сгенерировать нормально распределённые данные можно несколькими способами, например, с помощью функции *norm.rvs*

```
sample = stats.norm.rvs(loc=0, scale=1, size=1000),
```

где параметр *loc* отвечает за среднее значение, *scale* за среднеквадратическое отклонение, а *size* – за размер выборки.

Для генерации многомерного нормального распределения с заданным вектором средних значений и заданной матрицей ковариаций нам понадобится использовать функцию *multivariate_normal*:

```
mean = [0, 0]
cov = [[1, 0], [0, 1]]
data = stats.multivariate_normal(mean=mean, cov=cov).rvs(size=1000)
```

Предположим, мы хотим **проверить гипотезу о нормальности** распределения некоторой. Мы можем использовать функцию *normaltest* модуля *stats* библиотеки *SciPy,* предварительно сгенерировав выборку из нормального распределения другим способом - с помощью функции *random* библиотеки *numpy:*

```
sample = np.random.normal(size=100)
stat, p = stats.normaltest(sample)
if p > alpha:
 print('Дисперсии равны')
else:
 print('Дисперсии не равны')
```

Результатом вызова функции *normaltest* является кортеж статистики критерия и значения *p*, указывающей на вероятность того, что наблюдаемые данные могут быть получены из нормального распределения. Если p-значение больше выбранного уровня значимости *alpha*, то гипотеза о нормальности распределения выборки принимается, иначе – отвергается.

Задачи **проверки гипотез о параметрах распределений** также можно решать, используя известные формулы (см. [2], [3]), предполагающие вычисления выборочных характеристик и квантилей, либо автоматически.

В частности, проверить гипотезу о равенстве среднего значения нормальной выборки *sample* наперед заданному значению a_0 (то есть гипотезы $H_0 : a = a_0$ против альтернативы $H_1 : a \neq a_0$, см. [3, с. 176], формула (4.21)) можно с помощью метода *ttest_1samp,* в котором реализуется известный критерий Стьюдента, следующим образом:

```
results = stats.ttest_1samp (sample, popmean)
if results.pvalue > alpha:
 print("Среднее равно ожидаемому значению")
else:
 print("Среднее не равно ожидаемому значению")
```

Здесь *popmean* – ожидаемое среднее значение. Результатом применения критерия является кортеж, состоящий из *t*-статистики критерия и *results.pvalue* - так называемого *p-значения* (вероятности получить итоговую статистику при условии, что верна основная гипотеза). Заметим, что метод *ttest_1samp* не содержит уровень значимости критерия в качестве параметра. Для принятия решения мы должны сравнить значение *results.pvalue* с уровнем значимости *alpha* (в первом случае принимает гипотезу H_0, во втором – H_1):

Чтобы проверить гипотезу о равенстве средних двух выборок *sample1 и sample2 (*т.е. гипотезы $H_0 : a_1 = a_2$ против альтернативы $H_1 : a_1 \neq a_2$, см. [3, с. 176], формула (4.24)), мы сгенерируем 2 выборки из нормального распределения и применим метод *ttest_ind:*

```
sample1 = np.random.normal(loc=5, scale=2, size=100)
sample2 = np.random.normal(loc=10, scale=4, size=100)
results = stats.ttest_ind (sample1, sample2, equal_var=True/False)
```

Соответствующий метод получает на вход массивы выборок, а также необязательный параметр *equal_var*, задающий, считать ли равными дисперсии выборок; по умолчанию он примет значение *True*. Проверка гипотез проводится так же, как и в предыдущем случае, т.е. с помощью сравнения p-значения *results.pvalue* с заданным уровнем значимости *alpha.*

Для проверки гипотезы о равенстве дисперсий двух выборок *sample1 и sample2* из нормального распределения *(*т.е. гипотезы $H_0 : \sigma_1^2 = \sigma_2^2$ против альтернативы $H_1 : \sigma_1^2 \neq \sigma_2^2$, [2, с. 290]) можно использовать функцию *stats.levene* следующим образом:

```
stat, p = stats.levene(sample1, sample2)
if p > alpha:
 print('Дисперсии равны')
else:
 print('Дисперсии не равны')
```

В данном примере функция *levene* принимает две выборки из нормального распределения *sample1* и *sample2* и возвращает значение статистики и p-значение. Отметим, что в отличие от предыдущих примеров, результат вызова функции сразу записывается в 2 перемен-

ные, а не в кортеж. *Если p-значение* больше выбранного уровня значимости *alpha*, то гипотеза о равенстве дисперсий принимается, иначе - отвергается.

В Python есть специальный метод для проверки гипотезы о равенстве средних двух зависимых (парных) выборок (см. [2, с. 314]) в Python, применим метод *scipy.stats.ttest_rel(sample1, sample2).* Методу достаточно передать только два параметра с массивами данных, при этом размеры массивов должны быть одинаковы. Принятие решения осуществляется аналогично с помощью условного оператора.

Вычисление **доверительных интервалов** для параметров основных распределений можно провести двумя способами: непосредственно с использованием функции *interval* модуля *stats*, либо с использованием выборочных характеристик (выборочного среднего, смещенной или исправленной дисперсии), а также квантилей нужных распределений. Все инструменты для вычислений последних также содержатся в модуле *stats* библиотеки *SciPy,* кроме того, можно использовать библиотеку для научных вычислений *NumPy*.

Непосредственное вычисление доверительного интервала для параметра среднего нормального распределения в случае неизвестной дисперсии можно провести по известным формулам [2, с. 217]:

alpha = 0.95

t_critical = stats.t.ppf(q=(1+alpha)/2, df=len(sample)-1)

loc = sample.mean()

scale = sample.sem()

*radius = t_critical * scale*

lower_bound = loc - radius

upper_bound = loc + radius

Здесь: *alpha* - уровень доверия, *df* (от англ. «degrees of freedom») — число степеней свободы, равное n−1, n – объем выборки; для выборки *sample* вычисляется как *len(sample)-1, loc* – выборочное среднее; для выборки *sample* вычисляется как *sample.mean(), scale*– так называется оценка стандартной ошибки, равная $\sqrt{s^2/n}$, где s^2 – исправленная выборочная дисперсия, для выборки *sample* вычисляется как *sample.sem().*

Вычисления можно сократить и упростить, если использовать метод *interval* модуля распределения Стьюдента *stats.t* (используемые переменные повторяют предыдущий пример):

confidence_interval = statst.t.interval(alpha=0.95, df=len(sample)-1, loc=sample.mean(), scale = sample.sem()).

Результат вызова метода, т.е. кортеж значений *confidence_interval* составляют доверительный интервал.

Для нахождения **коэффициента корреляции** с помощью библиотеки SciPy можно использовать функцию *pearsonr*. Эта функция принимает два аргумента: два массива данных, которые нужно проанализировать:

sample1 = [1, 2, 3, 4, 5]

sample2 = [5, 4, 3, 2, 1]

corr, p_value = stats.pearsonr(sample1, sample2)

В данном случае значение *corr=1* (данные линейно зависимы). Полученное же значение *p_value* можно использовать для проверки гипотезы о равенстве нулю коэффициента корреляции (т.е. гипотезы $H_0 : \rho = 0$ против альтернативы $H_1 : \rho \neq 0$, [2, с. 327]). Если *p_value* меньше уровня значимости *alpha*, то это означает, что есть статистически значимая корреляция между двумя переменными и можно отвергнуть нулевую гипотезу. Если *p_value* больше уровня значимости, то нет оснований отвергать нулевую гипотезу и можно сделать вывод, что корреляция между двумя переменными статистически незначима.

При работе с многомерными нормальными данными для вычисления матрицы ковариаций можно использовать функцию *numpy.cov*. Эта функция принимает на вход массив данных и возвращает матрицу ковариаций.

В настоящей работе приведены примеры только некоторых из инструментов, доступных в Python для решения задач математической статистики. В зависимости от ваших конкретных потребностей, можно рассмотреть и другие инструменты, которые лучше всего подходят для предлагаемой задачи, в частности, библиотеку Pandas, работающую с данными, представленные непосредственно в виде таблиц; библиотеки Matplotlib и Seaborn, содержащие широкий перечень инструментов визуализации данных, и даже, возможно, библиотеку для машинного обучения Scikit-learn.

Библиографический список

1. Ветров Л.Г., Кузнецова А.А., Сунчалина А.Л. Прикладная статистика. М.: Изд-во МГТУ им. Н.Э. Баумана, 2017. 52 с.

2. Гмурман В. Е. Теория вероятностей и математическая статистика. 12-е изд. М.: Юрайт, 2017. 479 с.

3. Горяинов В. Б., Павлов И. В., Цветкова Г. М. Математическая статистика. М.: Изд-во МГТУ им. Н.Э. Баумана, 2001. 424 с.

4. Кузнецова А.А. Лабораторные работы в курсе статистики // Актуальные проблемы преподавания математики в техническом вузе. 2018. Т. 6. С. 159-164.

5. Ветров Л.Г., Сунчалина А.Л. Лабораторные работы в курсе математической статистики // Инженерный журнал: наука и инновации. 2013. Вып. 5. С. 1 – 13.

Сведения об авторе:

Анна Александровна Кузнецова, кандидат физико-математических наук, доцент

E-mail: kuznetsova.a.a@bmstu.ru, kuznetsova.a.a@bk.ru; SPIN-code: 2012-1660, ORCID: 0000-0002-0512-8498.