

Домашка домашка

Привет! Это грозное и ужасное домашнее задание по матстату, сделанное почти в последний момент. Делала Гарнова Вероника, группа БЭК 212.

Facts: люблю людей, горы и ежейков

Задача 1

Однажды в Самарканде турист заказывал Яндекс-такси. На десятом заказе впервые приехал таксист, который уже раньше приезжал к туристу. Для упрощения предположим, что все n таксистов Самарканда всегда на работе и приезжают равновероятно.

а) [5] Постройте график функции правдоподобия как функции от общего количества такси n . Найдите оценку числа n методом максимального правдоподобия.

Решение

Если таксист приезжает повторно — это успех! Используем биномиальное распределение, где всего у нас n таксистов, и мы считаем количество заказов до приезда первого знакомого водителя

$$f = C_{1,n} * 1/n * (1 - 1/n)^{(n-1)} = n * 1/n * (1 - 1/n)^{(n-1)} = (1 - 1/n)^{(n-1)} - \text{распределение на } n \text{ таксистах}$$

Найдем функцию правдоподобия как произведение для n элементов

$$L = \prod f(n) = ((1 - 1/n)^{(n-1)})^n = ((n-1)/n)^{(n*(n-1))}$$

Построим на основе этого график

```
import numpy as np
import matplotlib.pyplot as plt

def taxi(n):
    return ((n - 1)/n)**(n*(n - 1))

n_val = np.arange(1, 21)
taxi_val = [taxi(n) for n in n_val]

plt.plot(n_val, taxi_val)
plt.xlabel('Количество таксистов в Самарканде')
plt.ylabel('вероятность появления знакомого таксиста')
plt.title('Функция правдоподобия для приезжающих знакомых таксистов в Самарканде')
plt.show()
```

Конечно, логично, что максимальная вероятность встретить знакомого таксиста — когда он один в городе. Но очевидно, что это не то, что мы ищем. Не придумала, что делать дальше и как фиксировать

Задача 3 (есть)

Иноагент Иннокентий по 20 наблюдениям строит 95%-й доверительный интервал для математического ожидания несколькими способами: классический асимптотический нормальный интервал, с помощью наивного бутстрэпа, с помощью бутстрэпа t-статистики.

а) [15] Для каждого способа с помощью 10000 симуляций оцените вероятность того, что номинально 95%-й доверительный интервал фактически покрывает математическое ожидание, если наблюдения распределены экспоненциально с интенсивностью 1.

Решение:

- Кол-во наблюдений: $n = 20$
- интенсивность: $scale = 1$
- Уровень доверия: $alpha = 0.05$
- Кол-во симуляций: $m = 10\,000$

Как решаем:

1. Генерируем выборки
2. Находим мат ожидания и дисперсии в выборках
3. Строим доверительные интервалы
4. Считаем вероятность покрытия мат ожидания для каждого варианта

Что мы финально ищем: вероятность = кол-во случаев, когда интервал накрыл мат ожидания / 10 000

```
import numpy as np
from scipy.stats import norm, t

n = 20
alpha = 0.05
m = 10000
scale = 1
np.random.seed(29)

base = np.zeros((m, 2))
for i in range(m):
    x = np.random.exponential(scale=1, size=n)
    x_mean = np.mean(x)
    s = np.std(x, ddof=1)
    z_alpha = norm.ppf(1 - alpha/2)
```

```

        base[i, 0] = x_mean - z_alpha * s / np.sqrt(n)
        base[i, 1] = x_mean + z_alpha * s / np.sqrt(n)
base_cover = np.logical_and(scale >= base[:, 0], scale <= base[:, 1])
base_prob = np.mean(base_cover)

# Наивный бутстрэп
naivnii = np.zeros((m, 2))
for i in range(m):
    x = np.random.exponential(scale=1, size=n)
    y = np.random.choice(x, size=n, replace=True)
    x_mean = np.mean(y)
    naivnii[i, 0] = np.quantile(y, alpha/2)
    naivnii[i, 1] = np.quantile(y, 1 - alpha/2)
naivnii_cover = np.logical_and(scale >= naivnii[:, 0], scale <=
naivnii[:, 1])
naivnii_prob = np.mean(naivnii_cover)

# Бутстрэп t-статистики
t_stat = np.zeros((m, 2))
for i in range(m):
    x = np.random.exponential(scale=1, size=n)
    y = np.random.choice(x, size=n, replace=True)
    z = (np.mean(y) - np.mean(x)) / (np.std(y, ddof=1) / np.sqrt(n))
    t_alpha = t.ppf(1 - alpha/2, df=n-1)
    t_stat[i, 0] = np.mean(y) - t_alpha * np.std(y, ddof=1) /
np.sqrt(n)
    t_stat[i, 1] = np.mean(y) + t_alpha * np.std(y, ddof=1) /
np.sqrt(n)
t_cover = np.logical_and(scale >= t_stat[:, 0], scale <= t_stat[:, 1])
t_prob = np.mean(t_cover)

print("Классический асимптотический нормальный интервал:", base_prob)
print("Наивный бутстрэп:", naivnii_prob)
print("Бутстрэп t-статистики:", t_prob)

```

Классический асимптотический нормальный интервал: 0.9086

Наивный бутстрэп: 0.9951

Бутстрэп t-статистики: 0.798

Предварительно можем отметить, что ближе всего к 95% оценка через наивный бутстрэп, дальше всего — через бутстрэп t-статистики

б) [5] Пересчитайте вероятности накрытия, если наблюдения имеют распределение Стьюдента с тремя степенями свободы.

```

n = 20
alpha = 0.05
m = 10000

```

```

np.random.seed(29)

```

```
sims = np.random.standard_t(3, size=(m, n)) #генерируем выборки / sims
- simulations
```

```
base = np.zeros((m, 2))
for i in range(m):
    x = np.random.standard_t(3, size=(m, n))
    x_mean = np.mean(x)
    s = np.std(x, ddof=1)
    z_alpha = t.ppf(1 - alpha/2, df=n-1)
    base[i, 0] = x_mean - z_alpha * s / np.sqrt(n)
    base[i, 1] = x_mean + z_alpha * s / np.sqrt(n)
base_cover = np.logical_and(0 >= base[:, 0], 0 <= base[:, 1])
base_prob = np.mean(base_cover)
```

```
# Наивный бутстрэп
```

```
naivnii = np.zeros((m, 2))
for i in range(m):
    x = np.random.standard_t(3, size=(m, n))
    y = np.array([np.random.choice(x[i], size=n, replace=True) for i
in range(m)])
    x_mean = np.mean(y)
    naivnii[i, 0] = np.quantile(y, alpha/2)
    naivnii[i, 1] = np.quantile(y, 1 - alpha/2)
naivnii_cover = np.logical_and(0 >= naivnii[:, 0], 0 <= naivnii[:, 1])
naivnii_prob = np.mean(naivnii_cover)
```

```
# Бутстрэп t-статистики
```

```
t_stat = np.zeros((m, 2))
for i in range(m):
    x = np.random.standard_t(3, size=(m, n))
    y = np.random.choice(x, size=n, replace=True)
    z = (np.mean(y) - np.mean(x)) / (np.std(y, ddof=1) / np.sqrt(n))
    t_alpha = t.ppf(1 - alpha/2, df=n-1)
    t_stat[i, 0] = np.mean(y) - t_alpha * np.std(y, ddof=1) /
np.sqrt(n)
    t_stat[i, 1] = np.mean(y) + t_alpha * np.std(y, ddof=1) /
np.sqrt(n)
t_cover = np.logical_and(0 >= t_stat[:, 0], 0 <= t_stat[:, 1])
t_prob = np.mean(t_cover)
```

```
print("Классический асимптотический нормальный интервал:", base_prob)
print("Наивный бутстрэп:", naivnii_prob)
print("Бутстрэп t-статистики:", t_prob)
```

```
-----
-----
KeyboardInterrupt                                Traceback (most recent call
last)
<ipython-input-72-67017026a9fb> in <cell line: 22>()
```

```

22 for i in range(m):
23     x = np.random.standard_t(3, size=(m, n))
--> 24     y = np.array([np.random.choice(x[i], size=n, replace=True)
for i in range(m)])
25     x_mean = np.mean(y)
26     naïvnii[i, 0] = np.quantile(y, alpha/2)

<ipython-input-72-67017026a9fb> in <listcomp>(.0)
22 for i in range(m):
23     x = np.random.standard_t(3, size=(m, n))
--> 24     y = np.array([np.random.choice(x[i], size=n, replace=True)
for i in range(m)])
25     x_mean = np.mean(y)
26     naïvnii[i, 0] = np.quantile(y, alpha/2)

```

KeyboardInterrupt:

в) [5] Какой способ оказался лучше?

пункт б не удалось домучить, но наивно предполагаю, что лучше всего наивный бутстрап

Задача 4

Проверьте гипотезу о том, что ожидаемые результаты экзамена по теории вероятностей тех, у кого фамилия начинается с гласной буквы и с согласной буквы, равны. В качестве альтернативной гипотезы возьмите гипотезу о неравенстве.

а) [5] Используйте тест Уэлча.

```

import pandas as pd
from scipy.stats import ttest_ind

df = pd.read_csv('/content/резь экзаменов - Лист1.csv')
df.dropna(subset=['Баллы'], inplace=True) # удаление строк с
пропущенными значениями в столбце 'Баллы'
alpha = 0.05
glas_scores = []
sogl_scores = []
for i in range(len(df)):
    if df.loc[i, 'Фамилия'][0] in ['А', 'Е', 'Ё', 'И', 'О', 'У', 'Ы',
'Э', 'Ю', 'Я']:
        glas_scores.append(df.loc[i, 'Баллы'])
    else:
        sogl_scores.append(df.loc[i, 'Баллы'])

t_stat, p_value = ttest_ind(glas_scores, sogl_scores, equal_var=False)

```

```
print("p-value:", p_value)
print(alpha - p_value)

p-value: 0.3909901940797269
-0.3409901940797269
```

Так как уровень значимости меньше p-value, то не отвергаем нулевую гипотезу, а значит результаты для двух групп примерно равны.

б) [5] Используйте наивный бутстрэп.

```
n = 10000
means_glas = []
means_sogl = []

for i in range(n):
    bootstrapped_glas = np.random.choice(glas_scores,
size=len(glas_scores), replace=True)
    bootstrapped_sogl = np.random.choice(sogl_scores,
size=len(sogl_scores), replace=True)

    means_glas.append(np.mean(bootstrapped_glas))
    means_sogl.append(np.mean(bootstrapped_sogl))

means_diff = np.array(means_glas) - np.array(means_sogl)

p_value = np.sum(means_diff >= (np.mean(glas_scores) -
np.mean(sogl_scores))) / n

print("p-value:", p_value)
print(alpha - p_value)

p-value: 0.4972
-0.4472
```

Так как разница снова отрицательная, не отвергаем нулевую гипотезу

в) [5] Используйте бутстрэп t-статистики.

```
t_statistic = (np.mean(glas_scores) - np.mean(sogl_scores)) /
np.sqrt((np.var(glas_scores) / len(glas_scores)) +
(np.var(sogl_scores) / len(sogl_scores)))

p_value = 2 * (1 - ttest_ind(glas_scores, sogl_scores).pvalue)

print("p-value:", p_value)
print(alpha - p_value)

p-value: 1.2768640717018125
-1.2268640717018124
```

аналогично, p-value сильно больше уровня доверия, значит не отвергаем нулевую гипотезу

г) [5] Используйте перестановочный тест.

```
diff_mean = np.mean(glas_scores) - np.mean(sogl_scores)
merged = glas_scores + sogl_scores

diffs = []
for i in range(n):
    change = np.random.permutation(n)
    glas_change = change[:len(glas_scores)]
    sogl_change = change[len(glas_scores):]
    diff = np.mean(glas_change) - np.mean(sogl_change)
    diffs.append(diff)

p_val = (np.sum(np.abs(diffs) >= np.abs(diff_mean)) + 1) / (n + 1)
print("p-value:", p_val)
print(alpha - p_value)

p-value: 0.9984001599840016
-1.2268640717018124
```

И снова получаем отрицательный результат. В 4 различных случаях подтвердилось, что мы не отвергаем нулевую гипотезу и считаем, что результаты студентов с фамилиями с согласной или гласной буквы равны

Задача 5

Составьте таблицу сопряжённости, поделив студентов писавших экзамен на четыре группы по двум признакам: набрал ли больше медианы или нет, на согласную или гласную букву начинается фамилия.

а) [5] Постройте 95% асимптотический интервал для отношения шансов хорошо написать экзамен («несогласных» к «согласным»). Проверьте гипотезу о том, что отношение шансов равно 1 и укажите P-значение.

```
df['Больше медианы'] = df['Баллы'] > df['Баллы'].median()
df['Начинается на гласную'] = df['Фамилия'].apply(lambda x: x[0] in
['А', 'Е', 'Ё', 'И', 'О', 'У', 'Ы', 'Э', 'Ю', 'Я'])

contingency_table = pd.crosstab(df['Больше медианы'], df['Начинается
на гласную']) # создание таблицы сопряженности
print(contingency_table)

odds_ratio = (contingency_table.iloc[1, 1]/contingency_table.iloc[0,
1])/(contingency_table.iloc[1, 0]/contingency_table.iloc[0, 0])
print("Отношение шансов хорошо написать экзамен (гласные к
согласным):", odds_ratio)
```

```
from scipy.stats import chi2_contingency
chi2, p_value, dof, expected = chi2_contingency(contingency_table)
print("P-value:", p_value)
```

```
se = np.sqrt(1/contingency_table.iloc[0, 0] +
1/contingency_table.iloc[1, 0] + 1/contingency_table.iloc[0, 1] +
1/contingency_table.iloc[1, 1])
lower_bound = np.exp(np.log(odds_ratio) - 1.96*se)
upper_bound = np.exp(np.log(odds_ratio) + 1.96*se)
print("95% асимптотический интервал для отношения шансов:",
lower_bound, "-", upper_bound)
print(alpha - p_value)
```

Начинается на гласную False True

Больше медианы

False	130	23
-------	-----	----

True	123	20
------	-----	----

Отношение шансов хорошо написать экзамен (гласные к согласным):

0.9190526687875574

P-value: 0.9280265277546991

95% асимптотический интервал для отношения шансов: 0.48075352433091567

- 1.756945638996774

-0.878026527754699

Разница отрицательная, значит не отвергаем нулевую гипотезу об отношении шансов, равном 1

[5] Постройте 95% асимптотический интервал для отношения вероятностей хорошо написать экзамен. Проверьте гипотезу о том, что отношение вероятностей равно 1 и укажите P-значение.

[5] Постройте 95% интервал для отношения шансов хорошо написать экзамен с помощью наивного бутстрэпа. Проверьте гипотезу о том, что отношение шансов равно 1 и укажите P-значение.

Задача 7

[10] С помощью chat gpt решите любую задачу из нашего курса теории вероятностей и статистики. Можно брать задачи из прошлых контрольных, лекций, семинаров и даже этого домашнего задания. В качестве ответа приведите полный диалог с chatgpt.

Простой диалог в виде двух реплик условия и ответа chatgpt даёт 6 баллов. Сложный диалог с наводками, указанием chatgpt на ошибки и их исправлением — 10 баллов.

Диалог с ChatGPT для попытки решения 1 задачи:

Me: Однажды в Самарканде турист заказывал Яндекс-такси. На десятом заказе впервые приехал таксист, который уже раньше приезжал к туристу. Для упрощения предположим, что все n таксистов Самарканда всегда на работе и приезжают равновероятно.

а) [5] Постройте график функции правдоподобия как функции от общего количества такси n . Найдите оценку числа n методом максимального правдоподобия.

Chat: Для решения данной задачи нужно использовать биномиальное распределение, так как мы ищем вероятность того, что при n попытках среди них найдется k успехов (в данном случае $k=1$).

Функция правдоподобия для биномиального распределения задается следующим образом:

$$L(n) = P(k=1;n) = n * (1/n) * (1-1/n)^{(n-1)}$$

где $P(k=1;n)$ - вероятность того, что на 10-м заказе приедет таксист, который уже раньше был у туриста, n - общее количество таксистов, k - количество успехов (в данном случае $k=1$).

Me: разве функция правдоподобия это не произведение функции плотности в разных точках?

Chat: Вы абсолютно правы, функция правдоподобия - это произведение функций плотности в разных точках. Я допустил ошибку в своем предыдущем ответе.

Для биномиального распределения функция правдоподобия будет выглядеть следующим образом:

$$L(n) = P(k=1;n)^{10} = (n * (1/n) * (1-1/n)^{(n-1)})^{10}$$

где $P(k=1;n)$ - вероятность того, что на 10-м заказе приедет таксист, который уже раньше был у туриста, n - общее количество таксистов, k - количество успехов (в данном случае $k=1$).

далее не пошло, потому что я идеологически запуталась, должно ли там быть 10 или n

Задача 8

[5] Укажите любой источник по теории вероятностей или статистике, который вам оказался полезен в течение года. Это может быть статья, видео, задача, всё что угодно. Объясните, с чем конкретно этот источник помог разобраться.

Лучше привести в пример внешний источник, не упомянутый на вики курса, но можно и внутренний.

Неожиданно приятным и полезным оказался [цикл видеосов Фила на YT](#)

Смотреть 10-05 — 10-12. Станет сильно понятнее про бутстрап, критерии Пирсона и Колмогорова, их применение в статистике и в прикладном варианте в python.