Университет ИТМО
Факультет программной инженерии и компьютерной техники

Теория Вероятности. Практическая работа №5.

Группа: P32131
Студент: Смирнов Виктор Игоревич
Вариант: 16

2023

1 Исходный код

1.1 Модуль для вычисления основных статистических показателей

```
'''
Statistics functions.
'''

from typing import (
    Collection,
    Iterable,
    List,
    NamedTuple,
    Callable,
)
from numbers import Number
from functools import reduce
from math import sqrt, log, ceil, factorial, exp

Probability = Number

Function = Callable[[Number], Number]


class Interval(NamedTuple):
    begin: Number
    end: Number

    @property
    def length(self) -> Number:
        return self.end - self.begin

    @property
    def middle(self) -> Number:
        return self.begin + self.length / 2

    def __str__(self) -> str:
        return f'({self.begin}, {self.end})'


class Bin(NamedTuple):
    interval: Interval
    count: int


class Histogram:
    def __init__(self, bins: List[Bin]):
        self.bin_list = bins

    @property
    def bins(self) -> List[Number]:
        return [self.bin_list[0].interval.begin] + \
            list(map(lambda b: b.interval.end, self.bin_list))

    @property
    def counts(self) -> List[int]:
        return list(map(lambda b: b.count, self.bin_list))


def max(numbers: Collection[Number]) -> Number:
    return reduce(
        lambda a, b: a if a > b else b,
        numbers,
        float('-inf'),
    )


def min(numbers: Collection[Number]) -> Number:
    return reduce(
        lambda a, b: a if a < b else b,
        numbers,
        float('inf'),
    )
```

```


def scope(numbers: Collection[Number]) -> Interval:
    return Interval(min(numbers), max(numbers))


def amplitude(numbers: Collection[Number]) -> Number:
    return scope(numbers).length


def mean(numbers: Collection[Number]) -> Number:
    return reduce(
        lambda a, b: a + b,
        numbers,
        0,
    ) / len(numbers)


def variance(numbers: Collection[Number], fixed=False) -> Number:
    m = mean(numbers)
    return reduce(
        lambda a, b: a + (b - m) ** 2,
        numbers,
        0,
    ) / (len(numbers) - (1 if fixed else 0))


def std(numbers: Collection[Number], fixed=False) -> Number:
    return sqrt(variance(numbers, fixed))


def distinct(numbers: Collection[Number]) -> Collection[Number]:
    return set(numbers)


def empirical_distribution_function(numbers: Collection[Number]) -> Function:
    return lambda t: len(list(filter(lambda n: n <= t, numbers))) / len(numbers)


def partition(scope: Interval, n: int) -> Iterable[Interval]:
    step = scope.length / n
    current = scope.begin + step
    while current < scope.end:
        yield Interval(current - step, current)
        current += step


def tabulate(scope: Interval, n: int) -> Iterable[Number]:
    return map(lambda interval: interval.middle, partition(scope, n))


def sturges_step(numbers: Collection[Number]) -> Number:
    return (max(numbers) - min(numbers)) / (1 + log(len(numbers)))


def histogram(numbers: Collection[Number], step) -> Histogram:
    def generate():
        F = empirical_distribution_function(numbers)
        scope = Interval(min(numbers) - step / 2, max(numbers))
        bins = ceil(scope.length / step)
        for interval in partition(scope, bins):
            count = (F(interval.end) - F(interval.begin)) * len(numbers)
            yield Bin(interval, count)
    return Histogram(list(generate()))
```

Листинг 1: Модуль для вычисления основных статистических показателей

1.2 Модуль для отрисовки графиков

```
'''
Plotting functions.
'''

```

```
import matplotlib.pyplot as plt
from typing import NamedTuple, Collection, Callable, List
from numbers import Number
from stati import Histogram

Function = Callable[[Number], Number]


class Point(NamedTuple):
    x: Number
    y: Number


class Plot:
    def __init__(self, title: str = ""):
        self.figure, self.axes = plt.subplots()
        self.axes.set_title(title)

    def points(self, points: Collection[Point]):
        self.axes.plot(
            list(map(lambda p: p.x, points)),
            list(map(lambda p: p.y, points))
        )

    def function(self, x: Collection[Number], f: Function):
        self.points(list(map(lambda x: Point(x, f(x)), x)))

    def histogram(self, histogram: Histogram):
        bins = histogram.bins
        counts = histogram.counts
        self.axes.hist(bins[:-1], bins, weights=counts)

    def show(self):
        plt.ion()
        plt.show()
```

Листинг 2: Модуль для отрисовки графиков

1.3 Скрипт для обработки данных

```
'''
Probability Theory Assignment 1.
Basics.
Variant 16.
Smirnov @vityaman Victor 2023.
(1, 16) - (1, 20), (2, 16) - (2, 20)
                                                                20        .

                        :                          ,
                                                    ,

                                                                 ,
                                                                                  ,

                                                                            .

                                                                     .


'''

from stati import *
from ploti import *

print('Probability Theory Assignment 1 by Smirnov Victor')

numbers = [
    -0.45, 0.52, -1.63, -0.42, -1.18,
    1.42,  0.66, -1.70,  0.17,  0.14,
    0.83, -0.48, -1.35,  0.31,  0.59,
```

```
    0.73,  0.00,  1.59,  0.17, -0.45
]

print('Input data:')
print(numbers)

numbers = sorted(numbers)
print('Sorted data:')
print(numbers)

distinct_numbers = sorted(distinct(numbers))
print('Distinct data:')
print(distinct_numbers)

print(f'Data size:          {len(numbers)}')
print(f'Distinct data size: {len(distinct_numbers)}')

print(f'Max:                {max(numbers)}')
print(f'Min:                {min(numbers)}')
print(f'Amplitude:          {amplitude(numbers)}')
print(f'Mean:               {mean(numbers)}')
print(f'Variance:           {variance(numbers)}')
print(f'Standart deviation: {std(numbers)}')

F = empirical_distribution_function(numbers)
plot = Plot('Empirical Distribution Function')
plot.function(tabulate(scope(numbers), 1000), F)
plot.show()

hist = histogram(numbers, sturges_step(numbers))
plot = Plot('Histogram')
plot.histogram(hist)
plot.points([Point(bin.interval.middle, bin.count) for bin in hist.bin_list])
plot.show()

input('Press any button to exit...')
```

Листинг 3: Скрипт для обработки данных

2 Результаты работы программы

2.1 Вывод в терминал

```
$ python3 lab5.py
Probability Theory Assignment 1 by Smirnov Victor
Input data:
[-0.45, 0.52, -1.63, -0.42, -1.18, 1.42, 0.66, -1.7, 0.17, 0.14, 0.83, -0.48, -1.35,
    0.31, 0.59, 0.73, 0.0, 1.59, 0.17, -0.45]
Sorted data:
[-1.7, -1.63, -1.35, -1.18, -0.48, -0.45, -0.45, -0.42, 0.0, 0.14, 0.17, 0.17, 0.31,
    0.52, 0.59, 0.66, 0.73, 0.83, 1.42, 1.59]
Distinct data:
[-1.7, -1.63, -1.35, -1.18, -0.48, -0.45, -0.42, 0.0, 0.14, 0.17, 0.31, 0.52, 0.59,
    0.66, 0.73, 0.83, 1.42, 1.59]
Data size:          20
Distinct data size: 18
Max:                1.59
Min:                -1.7
Amplitude:          3.29
Mean:               -0.026500000000000044
Variance:           0.82767275
Standart deviation: 0.9097652169653443
```

Листинг 4: Результаты вывода программы

2.2 Построенные графики

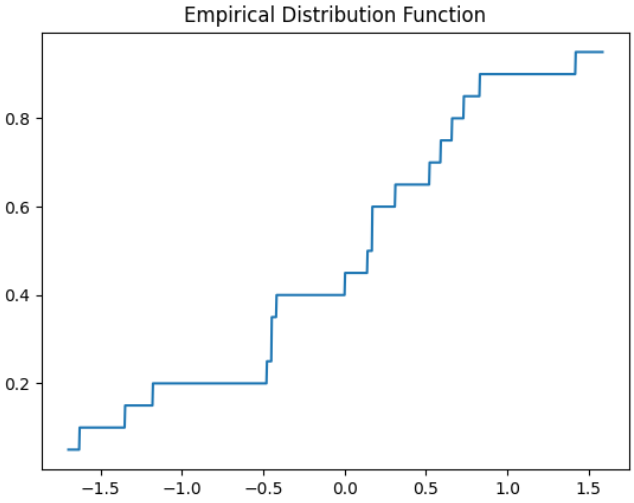

Рис. 1: График функции распределения

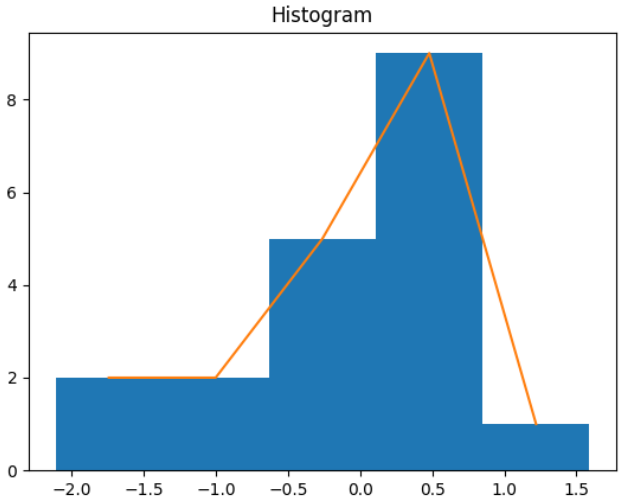

Рис. 2: График гистограммы распределения