Домашнее задание 1
по курсу "Прикладные методы математической статистики"
Вариант 1

от Татаринова Никиты Алексеевича, БПИ196

к 27.02.2021

Задача 1

Условие

Файл "Данные к задаче 1.ods"содержит сведения о продолжительности грудного вскармливания в неделях в выборке рожениц. Число после названия переменной соответствует номеру варианта (так, duration_6 - данные для шестого варианта). Ваша цель - оценить среднюю продолжительность вскармливания.

Решение

$\overline{X} = \frac{1}{n}\sum\limits_{i=1}^{20} X_i = 23,6$

$n \cdot S^2 = (n-1)\cdot\hat{\sigma}^2 = \sum\limits_{i=1}^{20}\left(X_i - \overline{X}\right)^2 = 1200,8$, то есть $S^2 = 60,04$ и $\hat{\sigma}^2 = 63,2$.

а)

Условие

Рассчитайте 90% доверительный интервал для средней продолжительности, считая распределение признака нормальным.

Решение

Необходимо рассчитать 90% доверительный интервал для средней продолжительности, считая распределение признака нормальным.

$$1-\alpha = 0,9 \quad \Leftrightarrow \quad \alpha = 0,1 \quad \Leftrightarrow \quad \frac{\alpha}{2} = 0,05 \quad \Leftrightarrow \quad 1-\frac{\alpha}{2} = 0,95$$

Так как случайные величины независимы, нормально распределены и дисперсия неизвестна, воспользуемся соответствующей формулой:

$$\overline{X} - t_{n-1,\frac{\alpha}{2}} \cdot \frac{\hat{\sigma}}{\sqrt{n}} < \mu < \overline{X} + t_{n-1,\frac{\alpha}{2}} \cdot \frac{\hat{\sigma}}{\sqrt{n}}$$

$$23,6 - 1,729 \cdot \sqrt{\frac{63,2}{20}} < \mu < 23,6 + 1,729 \cdot \sqrt{\frac{63,2}{20}}$$

$$20,526 < \mu < 26,674$$

б)

Условие

Постройте график "квантиль-квантиль" и попробуйте понять, соответствует ли распределение времени вскармливания нормальному закону.

Решение

порядок квантили, p	выборочная квантиль, $\hat{Q}(p)$	квантиль $\mathcal{N}(0,1)$, $\Phi^{-1}(p)$	теоретическая квантиль, $Q(p) = \overline{X} + \hat{\sigma} \cdot \Phi^{-1}(p)$
$^1/_{21}$	9	-1,668	10,337
$^2/_{21}$	11	-1,309	13,192
$^3/_{21}$	12	-1,068	15,113
$^4/_{21}$	12	-0,876	16,635
$^5/_{21}$	18	-0,712	17,936
$^6/_{21}$	18	-0,566	19,101
$^7/_{21}$	22	-0,431	20,176
$^8/_{21}$	22	-0,303	21,191
$^9/_{21}$	23	-0,180	22,169
$^{10}/_{21}$	25	-0,060	23,125
$^{11}/_{21}$	26	0,060	24,075
$^{12}/_{21}$	26	0,180	25,031
$^{13}/_{21}$	28	0,303	26,009
$^{14}/_{21}$	28	0,431	27,024
$^{15}/_{21}$	29	0,566	28,100
$^{16}/_{21}$	31	0,712	29,264
$^{17}/_{21}$	32	0,876	30,565
$^{18}/_{21}$	33	1,068	32,087
$^{19}/_{21}$	33	1,309	34,008
$^{20}/_{21}$	34	1,668	36,863

На основании таблицы построим график "квантиль-квантиль", на горизонтальной оси которого отложим значения теоретической квантили, а на вертикальной - значения выборочной квантили.

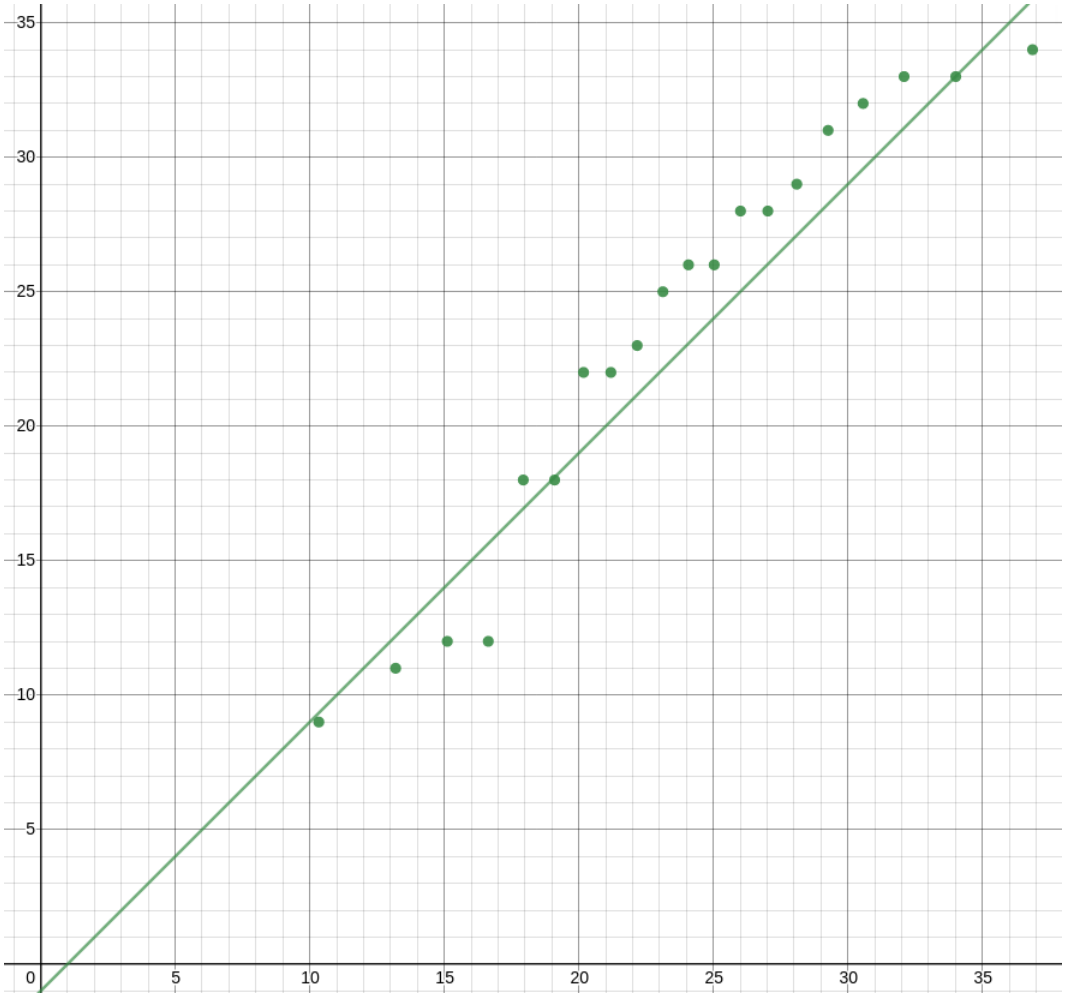

Рис 1.б.1: График "квантиль-квантиль"

Так как расположение наблюдаемых значений достаточно похоже на график линейной функции, можно считать, что теоретическое распределение хорошо описывает элементы выборки.

Таким образом, можно считать, что распределение времени вскармливания соответствует нормальному закону.

в)

Условие

Прочитайте ниже описание бутстрапа и рассчитайте этим методом 90% доверительный интервал для средней продолжительности вскармливания, сгенерировав 1000 перевыборок.

Решение

Для создания 1000 перевыборок используем язык программирования C++ (исходный код в файле "IntervalBoundaries.cpp", скриншоты кода в приложении). После создания перевыборок считаем их средние значения и сортируем эти значения. В таком случае, значение $0,05 \cdot 1000 = 50$ элемента будет нижней границей 90% доверительного интервала, а значение $0,95 \cdot 1000 + 1 = 951$ элемента будет его верхней границей (в языке C++ индексы начинаются с 0, поэтому нумерация смещена на (-1)).

В зависимости от перевыборок, эти границы будут разными. Один из результатов выполнения программы:

$$20,6 < \mu < 26,55$$

г)

Условие

При выполнении предыдущего пункта вы получите 1000 средних значений продолжительности вскармливания в перевыборках. Постройте гистограмму для этих значений. Похоже ли распределение среднего в перевыборках на нормальное?

Решение

По формуле Стерджесса количество интервалов равно $n = 1 + \lfloor log_2 N \rfloor = 1 + \lfloor log_2 1000 \rfloor = 10$.

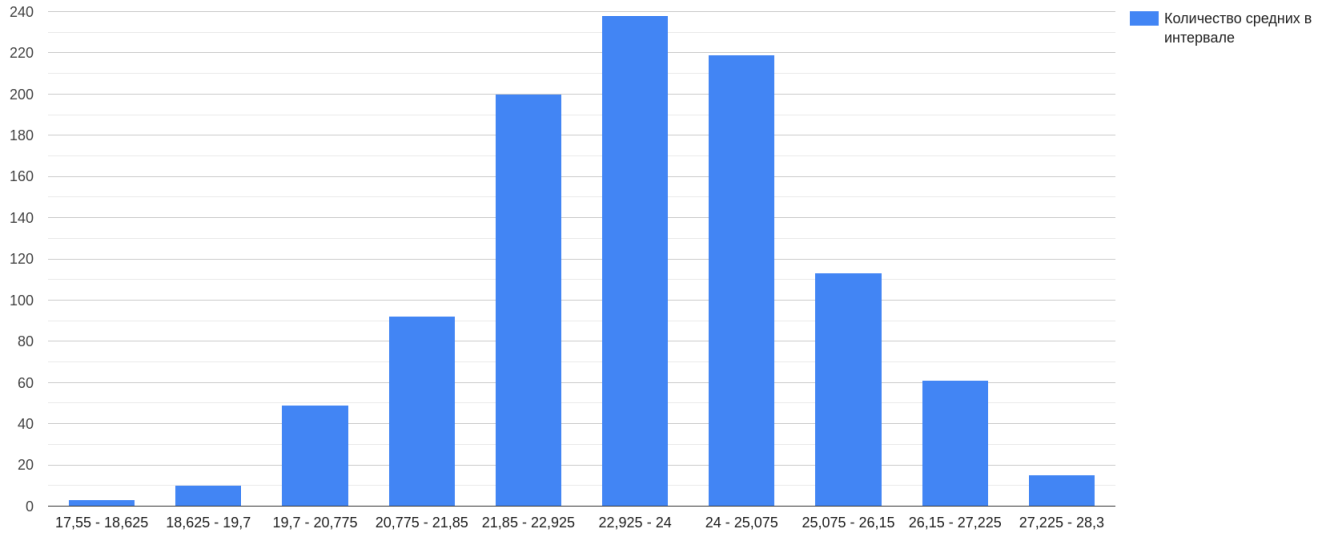

Рис 1.г.1: Гистограмма количеств средних значений перевыборок по интервалам

На гистограмме наглядно видно, что распределение среднего в перевыборках похоже на нормальное.

Приложение

Код программы

```cpp
#include <iostream>
#include <fstream>
#include <random>

const unsigned int bootstrap_length = 1000;

template<typename T>
int sgn(T val) {
    return (T(0) < val) - (val < T(0));
}

int cmp(const void *a, const void *b) {
    return sgn( val: *(double *) a - *(double *) b);
}

///Вычисляет выборочное среднее выборки sample
///размера size.
double sample_mean(const double *const sample,
                   const unsigned int size) {
    double result = 0;
    for (unsigned int i = 0; i < size; i++) {
        result += sample[i];
    }
    return result / static_cast<double>(size);
}

///Вычисляет дисперсию выборки sample размера size.
double sample_variance(const double *const sample,
                       const unsigned int size) {
    double sample_mean_ = sample_mean(sample, size);
    double result = 0;
    for (unsigned int i = 0; i < size; i++) {
        result += (sample[i] - sample_mean_) *
                  (sample[i] - sample_mean_);
    }
    return result / static_cast<double>(size);
}
```

1

```cpp
///Вычисляет скорректированную дисперсию выборки sample
///размера size.
double biased_sample_variance(const double *const sample,
                              const unsigned int size) {
    double sample_mean_ = sample_mean(sample, size);
    double result = 0;
    for (unsigned int i = 0; i < size; i++) {
        result += (sample[i] - sample_mean_) *
                  (sample[i] - sample_mean_);
    }
    return result / static_cast<double>(size - 1);
}

///Получает выборку из файла path. Формат файла:
///первый элемент - размер выборки (сохраняется
///в size); далее идут элементы выборки в соответствующем
///количестве. Корректность файла не проверяется.
///Возвращает ссылку на выборку.
double *get_sample(const char *const path,
                   unsigned int &size) {
    //Файл, откуда получаем элементы выборки.
    std::ifstream fin;
    fin.open(path);
    //Размер выборки.
    fin >> size;
    auto *sample = new double[size];
    for (unsigned int i = 0; i < size; i++) {
        fin >> sample[i];
    }
    fin.close();
    return sample;
}
```

2

```cpp
///Сортирует выборку sample размера size и выводит в файл
///path основную информацию об этой выборке.
void print_sample_data(const char *const path,
                       double *const sample,
                       const unsigned int size) {
    //Файл, в который выводим информацию о выборке.
    std::ofstream fout;
    fout.open(path);
    //Сортируем выборку.
    qsort(sample, size, sizeof(double), cmp);

    fout << "Отсортированная выборка:\n";
    for (unsigned int i = 0; i < size; i++) {
        fout << sample[i] << " ";
    }
    fout << "\nВыборочное среднее: " << sample_mean(sample, size) << "\n";
    fout << "Выборочная дисперсия: " << sample_variance(sample, size) << "\n";
    fout << "Скорректированная выборочная дисперсия: " <<
         biased_sample_variance(sample, size) << "\n";
    fout.close();
}
```

3

```cpp
///В файл sample_path сохраняет bootstrap_length перевыборок длины size
///от выборки sample размера size. В файл mean_path сохраняются
///средние значения перевыборок. В файл mean_sorted_path сохраняются
///отсортированные средние значение перевыборок. В файл interval_path
///сохраняются границы 90% доверительного интервала.
///Возвращается массив средних значений перевыборок.
double *bootstrap(const char *const sample_path,
                  const char *const mean_path,
                  const char *const mean_sorted_path,
                  const char *const interval_path,
                  const double *const sample,
                  const unsigned int size) {
    //Генератор случайных чисел для составления перевыборок.
    std::random_device rd;
    std::mt19937 engine(rd());
    std::uniform_int_distribution<unsigned int>
            element_index( a: 0, b: size - 1);
    //Файл с перевыборками.
    std::ofstream sample_fout;
    sample_fout.open(sample_path);
    //Файл со средними значениями перевыборок.
    std::ofstream mean_fout;
    mean_fout.open(mean_path);
    //Файл с отсортированными средними значениями
    //перевыборок.
    std::ofstream mean_sorted_fout;
    mean_sorted_fout.open(mean_sorted_path);
    //Файл с границами 90% доверительного интервала.
    std::ofstream interval_fout;
    interval_fout.open(interval_path);
    //Массив средних значений перевыборок.
    auto *means = new double[bootstrap_length];
    //Создаём bootstrap_length новых перевыборок.
    for (unsigned int i = 0; i < bootstrap_length; i++) {
        means[i] = 0;
        //Записываем size элементов перевыборки в файл.
        for (unsigned int j = 0; j < size; j++) {
            sample_fout << sample[element_index( &: engine)] << " ";
            means[i] += sample[element_index( &: engine)];
        }
```

4

```cpp
        sample_fout << "\n";
        means[i] /= static_cast<double>(size);
        mean_fout << means[i] << "\n";
    }
    //Сортируем массив средних для вычисления 90% доверительного
    //интервала.
    qsort(means, bootstrap_length, sizeof(double), cmp);
    for (unsigned int i = 0; i < bootstrap_length; i++) {
        mean_sorted_fout << means[i] << "\n";
    }
    interval_fout << means[static_cast<unsigned long long>(
                                0.05 * bootstrap_length) - 1] << " "
                  << means[static_cast<unsigned long long>(
                          0.95 * bootstrap_length)];

    sample_fout.close();
    mean_fout.close();
    mean_sorted_fout.close();
    interval_fout.close();
    return means;
}
```

5

```cpp
int main() {
    //Из файла "sample.txt" получаем выборку sample размера
    //size, сортируем её и выводим основную информацию о ней.
    unsigned int size;
    double *sample = get_sample( path: "sample.txt", & size);
    print_sample_data( path: "sample_data.txt", sample, size);

    //Создаём bootstrap_length перевыборок (сохраняем в файл
    //"bootstrap.txt"), вычисляем для каждой среднее значение
    //(сохраняем в файл "bootstrap_mean.txt" и после сортировки
    //в файл "bootstrap_mean_sorted.txt") и получаем
    //эти средние значения в виде массива. Границы интервала
    //сохраняются в файл "interval_boundaries.txt".
    double *means = bootstrap( sample_path: "bootstrap.txt",
                               mean_path: "bootstrap_mean.txt",
                               mean_sorted_path: "bootstrap_mean_sorted.txt",
                               interval_path: "interval_boundaries.txt",
                              sample, size);


    //Очищаем память.
    delete[] sample;
    delete[] means;
    return 0;
}
```

6

Названия и содержимое файлов, полученных в результате работы программы

1) "sample.txt" - файл, задаваемый пользователем: первое число должно быть целым - размер выборки, - после чего должны следовать элементы выборки в размере их количества (первое число файла). При некорректном формате входного файла работа программы непредсказуема.

2) "sample_data.txt" - файл, в котором содержится основная информация об этой выборке, а именно: отсортированная выборка, её среднее, её дисперсия и её скорректированная дисперсия.

3) "bootstrap.txt" - файл, содержащий 1000 перевыборок введённой пользователем выборки.

4) "bootstrap_mean.txt" - файл, содержащий средние значения 1000-и перевыборок из файла "bootstrap.txt".

5) "bootstrap_mean_sorted.txt" - файл, содержащий отсортированные средние значения перевыборок из файла "bootstrap.txt".

6) "interval_boundaries.txt" - файл, содержащий границы 90% доверительного интервала для средней продолжительности вскармливания.