Лабораторная работа 4, ТВМС

Бочарников Андрей, M3238
Ковешников Глеб, M3238
Шишкин Алексей, M3238

15 апреля 2020 г.

Формулировка

Для случайной величины, распределенной по нормальному закону с параметрами (a,σ^2), выполнить следующие действия:

1. Задать параметры распределения $X \sim N(a, \sigma^2)$.
2. Построить выборку генеральной совокупности X.
3. Построить график гистограммы.
4. Проверить гипотезу о виде распределения по критерию хи-квадрат.
Аналогично для $X \sim U(a, b)$ - равномерно распределенной на $[a, b]$ случайной величины.

Входные данные

- Размер выборки для построения гистограммы: $n = 10^6$
- Размер выборки для проверки критерия χ^2: $n = 10^4$
- Параметры нормального распределения: $\sigma = 1, \mu = 1$
- Параметры равномерного распределения: $a = 20, b = 80$
- $\alpha = 0.05$
- Количество тестов – 10^3

Программа 1

Нормальное распределение. График гистограммы - график 1 в приложении после вывода.
В первой части программы строится гистрограмма с использованием функции hist и выводится на экран вместе с графиком функции распределения.
Во второй части производится три запуска тестов на проверку гипотез:
1. Выборка генерируется с помощью нормального распределения; нижняя граница на количество элементов попавших в интервал отстутствует; проверяется гипотеза о нормальном распределении; оценивается вероятность ошибки первого рода.
2. Выборка генерируется с помощью нормального распределения; соседние интервалы объединяются, чтобы количество элементов попавших на каждый интервал было не меньше 6; проверяется гипотеза о нормальном распределении; оценивается вероятность ошибки первого рода.
3. Выборка генерируется с помощью нормального распределения; нижняя граница на количество элементов попавших в интервал отстутствует; проверяется гипотеза о равномерном распределении; оценивается вероятность ошибки второго рода.

3.1 Исходный код

```
pkg load statistics

clc;
clear all;
```

```
function res = test_Chi2_1(tests, n, m)
  res = 0;
  mu = 1;
  sigma = 1;
  alpha = 0.05;
  for t = 1 : tests
    X = sort(normrnd(mu, sigma, n, 1));

    l = min(X);
    r = max(X);

    delta = (r - l) / m;
    cnt_in_bucket = hist(X, m);
    walls = [];
    x_coords = [];
    for i = 1 : m
      walls(i, 1) = l + delta * (i - 1);
      walls(i, 2) = l + delta * i;
      x_coords(i) = (walls(i, 2) + walls(i, 1)) / 2;
    endfor

    #Выборочное среднее
    E = sum(x_coords .* cnt_in_bucket) / n;
    #Выборочная дисперсия
    D = sum((x_coords - E) .^ 2 .* cnt_in_bucket) / n;
    SQRT_D = sqrt(D);

    P = [];
    for i = 1 : m
      P(i) = normcdf(walls(i, 2), E, SQRT_D) - normcdf(walls(i, 1), E, SQRT_D);
    endfor

    hi2 = sum(((cnt_in_bucket - n .* P) .^ 2) ./ (n .* P));
    res = res + (hi2 >= chi2inv(1 - alpha, m - 1 - 2));
  endfor
  printf("Нормальное распределение проходит проверку гипотезы о нормальном 
     распределении\n");
  printf("Для alpha = %d, вероятность ошибки первого рода получается %d\n", alpha,
     res / tests);
endfunction

function res = test_Chi2_2(tests, n, m)
  res = 0;
  mu = 1;
  sigma = 1;
  alpha = 0.05;
  for t = 1 : tests
    X = sort(normrnd(mu, sigma, n, 1));

    l = min(X);
    r = max(X);

    delta = (r - l) / m;
    cnt_in_bucket = hist(X, m);
    x_coords = [];
    walls = [];
    for i = 1 : m
      cur_l = (i - 1) * delta + l;
      cur_r = cur_l + delta;
      walls(i, 1) = cur_l;
      walls(i, 2) = cur_r;
      x_coords(i) = (cur_r + cur_l) / 2;
```

```
    endfor

    fixed_cnt_in_bucket = [];
    fixed_walls = [];
    fixed_x_coords = [];
    fixed_m = 0;
    for i = 1 : m
      if (i == 1 || fixed_cnt_in_bucket(fixed_m) >= 6)
        fixed_m = fixed_m + 1;
        fixed_walls(fixed_m, 1) = walls(i, 1);
        fixed_cnt_in_bucket(fixed_m) = 0;
        fixed_x_coords(fixed_m) = 0;
      endif
      fixed_walls(fixed_m, 2) = walls(i, 2);
      fixed_cnt_in_bucket(fixed_m) = fixed_cnt_in_bucket(fixed_m) +
          cnt_in_bucket(i);
      fixed_x_coords(fixed_m) = fixed_x_coords(fixed_m) + cnt_in_bucket(i) *
          x_coords(i);
    endfor

    fixed_x_coords = fixed_x_coords ./ fixed_cnt_in_bucket;

    #Выборочное среднее
    E = sum(fixed_x_coords .* fixed_cnt_in_bucket) / n;
    #Выборочная дисперсия
    D = sum((fixed_x_coords - E) .^ 2 .* fixed_cnt_in_bucket) / n;
    SQRT_D = sqrt(D);

    P = [];
    for i = 1 : fixed_m
      P(i) = normcdf(fixed_walls(i, 2), E, SQRT_D) - normcdf(fixed_walls(i,
          1), E, SQRT_D);
    endfor

    hi2 = sum(((fixed_cnt_in_bucket - n .* P) .^ 2) ./ (n .* P));
    res = res + (hi2 >= chi2inv(1 - alpha, fixed_m - 1 - 2));
  endfor
  printf("Нормальное распределение проходит проверку гипотезы о нормальном 
      распределении\n");
  printf("Данные сгруппированы, чтобы выполнялось nj >= 6\n");
  printf("Для alpha = %d, вероятность ошибки первого рода получается %d\n", alpha,
      res / tests);
endfunction

function res = test_Chi2_3(tests, n, m)
  res = 0;
  mu = 1;
  sigma = 1;
  alpha = 0.05;
  for t = 1 : tests
    X = sort(normrnd(mu, sigma, n, 1));

    l = min(X);
    r = max(X);

    delta = (r - l) / m;
    cnt_in_bucket = hist(X, m);
    x_coords = [];
    walls = [];
    for i = 1 : m
      walls(i, 1) = l + delta * (i - 1);
      walls(i, 2) = l + delta * i;
      x_coords(i) = (walls(i, 1) + walls(i, 2)) / 2;
```

```
    endfor

    #Выборочное среднее
    E = sum(x_coords .* cnt_in_bucket) / n;
    #Выборочная дисперсия
    D = sum((x_coords - E) .^ 2 .* cnt_in_bucket) / n;
    SQRT_D = sqrt(D);

    P = [];
    for i = 1 : m
      P(i) = unifcdf(walls(i, 2), l, r) - unifcdf(walls(i, 1), l, r);
    endfor

    hi2 = sum(((cnt_in_bucket - n .* P) .^ 2) ./ (n .* P));
    res = res + (hi2 < chi2inv(1 - alpha, m - 1 - 2));
  endfor
  printf("Возьмём данные из нормального распределения и веротности из равномерного\n");
  printf("Тогда вероятность ошибки второго рода для alpha = %d, получается %d\n",
      alpha, res / tests);
endfunction

n = 10 ^ 6;
mu = 1;
sigma = 1;
m = 10 ^ 2;
tests = 10 ^ 3;

X = sort(normrnd(mu, sigma, n, 1));

l = min(X);
r = max(X);
delta = (r - l) / m;
x_coords = [];
y_coords = hist(X, m)' / (n * delta);

for i = 1 : m
  cur_l = (i - 1) * delta + l;
  cur_r = cur_l + delta;
  x_coords(i) = (cur_r + cur_l) / 2;
endfor

x_coords_for_normpdf = l:0.1:r;
stairs(x_coords, y_coords);
hold on;
plot(x_coords_for_normpdf, normpdf(x_coords_for_normpdf, mu, sigma));

printf("Размер выборки = %d\n", n);
printf("Выбранная длина интервалов = %d\n", delta);
printf("Количество интервалов = %d\n", m);

printf("\n");

# PART 2

test_Chi2_1(tests, 10 ^ 4, m);

printf("\n");

test_Chi2_2(tests, 10 ^ 4, m);

printf("\n");

test_Chi2_3(tests, 10 ^ 4, m);
```

3.2 Выходные данные

Размер выборки = 1000000
Выбранная длина интервалов = 0.0936386
Количество интервалов = 100

Нормальное распределение проходит проверку гипотезы о нормальном распределении
Для $\alpha = 0.05$, вероятность ошибки первого рода получается 0.31

Нормальное распределение проходит проверку гипотезы о нормальном распределении
Данные сгруппированы, чтобы выполнялось $n_j >= 6$
Для $\alpha = 0.05$, вероятность ошибки первого рода получается 0.063

Возьмём данные из нормального распределения и веротности из равномерного
Тогда вероятность ошибки второго рода для $\alpha = 0.05$, получается 0

Программа 2

Равномерное распределение. График гистограммы - график 2 после вывода.
В первой части программы строится гистрограмма с использованием функции hist и выводится на экран вместе с графиком функции распределения.
Во второй части производится два запуска тестов на проверку гипотез:
1. Выборка генерируется с помощью равномерного распределения; нижняя граница на количество элементов попавших в интервал отстутствует; проверяется гипотеза о равномерном распределении; оценивается вероятность ошибки первого рода.
2. Выборка генерируется с помощью равномерного распределения; нижняя граница на количество элементов попавших в интервал отстутствует; проверяется гипотеза о нормальном распределении; оценивается вероятность ошибки второго рода.

4.1 Исходный код

```
pkg load statistics

clc;
clear all;

function res = test_Chi2_1(tests, n, m)
  res = 0;
  a = 20;
  b = 80;
  alpha = 0.05;
  for t = 1 : tests
    X = sort(unifrnd(a, b, n, 1));

    l = min(X);
    r = max(X);

    delta = (r - l) / m;
    cnt_in_bucket = hist(X, m);
    walls = [];
    x_coords = [];
    for i = 1 : m
      walls(i, 1) = l + (i - 1) * delta;
      walls(i, 2) = l + i * delta;
      x_coords(i) = (walls(i, 2) + walls(i, 1)) / 2;
    endfor

    #Выборочное среднее
    E = sum(x_coords .* cnt_in_bucket) / n;
```

```
    #Выборочная дисперсия
    D = sum((x_coords - E) .^ 2 .* cnt_in_bucket) / n;
    SQRT_D = sqrt(D);

    P = [];
    for i = 1 : m
      P(i) = unifcdf(walls(i, 2), l, r) - unifcdf(walls(i, 1), l, r);
    endfor

    hi2 = sum(((cnt_in_bucket - n .* P) .^ 2) ./ (n .* P));
    res = res + (hi2 >= chi2inv(1 - alpha, m - 1 - 2));
  endfor
  printf("Равномерное распределение проходит проверку гипотезы о равномерном 
     распределении\n");
  printf("Для alpha = %d, вероятность ошибки первого рода получается %d\n", alpha,
     res / tests);
endfunction

function res = test_Chi2_2(tests, n, m)
  res = 0;
  a = 20;
  b = 80;
  alpha = 0.05;
  for t = 1 : tests
    X = sort(unifrnd(a, b, n, 1));

    l = min(X);
    r = max(X);

    delta = (r - l) / m;
    cnt_in_bucket = hist(X, m);
    walls = [];
    x_coords = [];
    for i = 1 : m
      walls(i, 1) = l + delta * (i - 1);
      walls(i, 2) = l + delta * i;
      x_coords(i) = (walls(i, 2) + walls(i, 1)) / 2;
    endfor

    #Выборочное среднее
    E = sum(x_coords .* cnt_in_bucket) / n;
    #Выборочная дисперсия
    D = sum((x_coords - E) .^ 2 .* cnt_in_bucket) / n;
    SQRT_D = sqrt(D);

    P = [];
    for i = 1 : m
      P(i) = normcdf(walls(i, 2), E, SQRT_D) - normcdf(walls(i, 1), E, SQRT_D);
    endfor

    hi2 = sum(((cnt_in_bucket - n .* P) .^ 2) ./ (n .* P));
    res = res + (hi2 < chi2inv(1 - alpha, m - 1 - 2));
  endfor
  printf("Возьмём данные из равномерного распределения и вероятности из 
     нормального\n");
  printf("Тогда для alpha = %d, вероятность ошибки второго рода получается - %d\n",
     alpha, res / tests);
endfunction

n = 10 ^ 6;
m = 10 ^ 2;
a = 20;
b = 80;
```

```
# PART 1

X = sort(unifrnd(a, b, 1, n));

l = min(X);
r = max(X);
delta = (r - l) / m;
y_coords = hist(X, m) / (n * delta);

x_coords = [];
for i = 1 : m
  cur_l = (i - 1) * delta + l;
  cur_r = cur_l + delta;
  x_coords(i) = (cur_r + cur_l) / 2;
endfor

real_y = 1 / (b - a);
stairs(x_coords, y_coords);
hold on;
plot([a b], [real_y real_y], "linewidth", 1);

printf("Размер выборки = %d\n", n);
printf("Выбранная длина интервалов = %d\n", delta);
printf("Количество интервалов = %d\n", m);

printf("\n");

# PART 2

test_Chi2_1(10 ^ 3, 10 ^ 4, m);

printf("\n");

test_Chi2_2(10 ^ 3, 10 ^ 4, m);
```

4.2 Выходные данные

Размер выборки = 1000000
Выбранная длина интервалов = 0.599999
Количество интервалов = 100

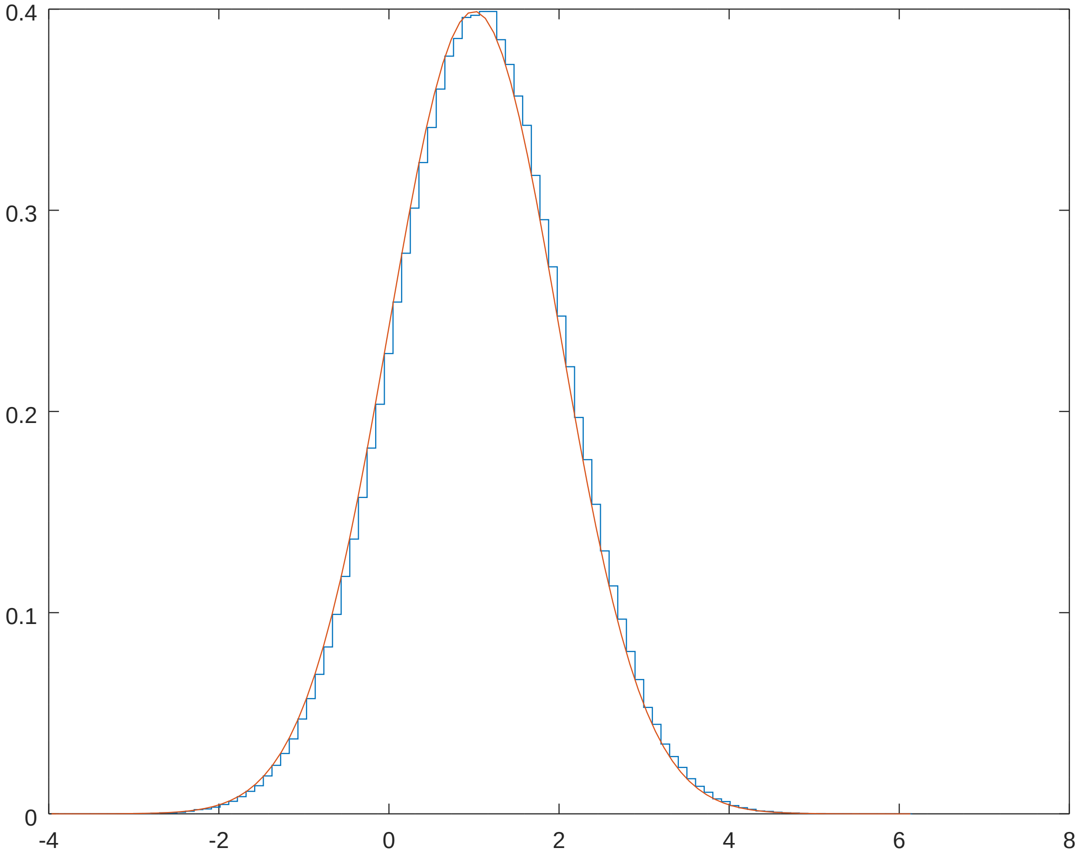
Равномерное распределение проходит проверку гипотезы о равномерном распределении
Для $\alpha = 0.05$, вероятность ошибки первого рода получается 0.071

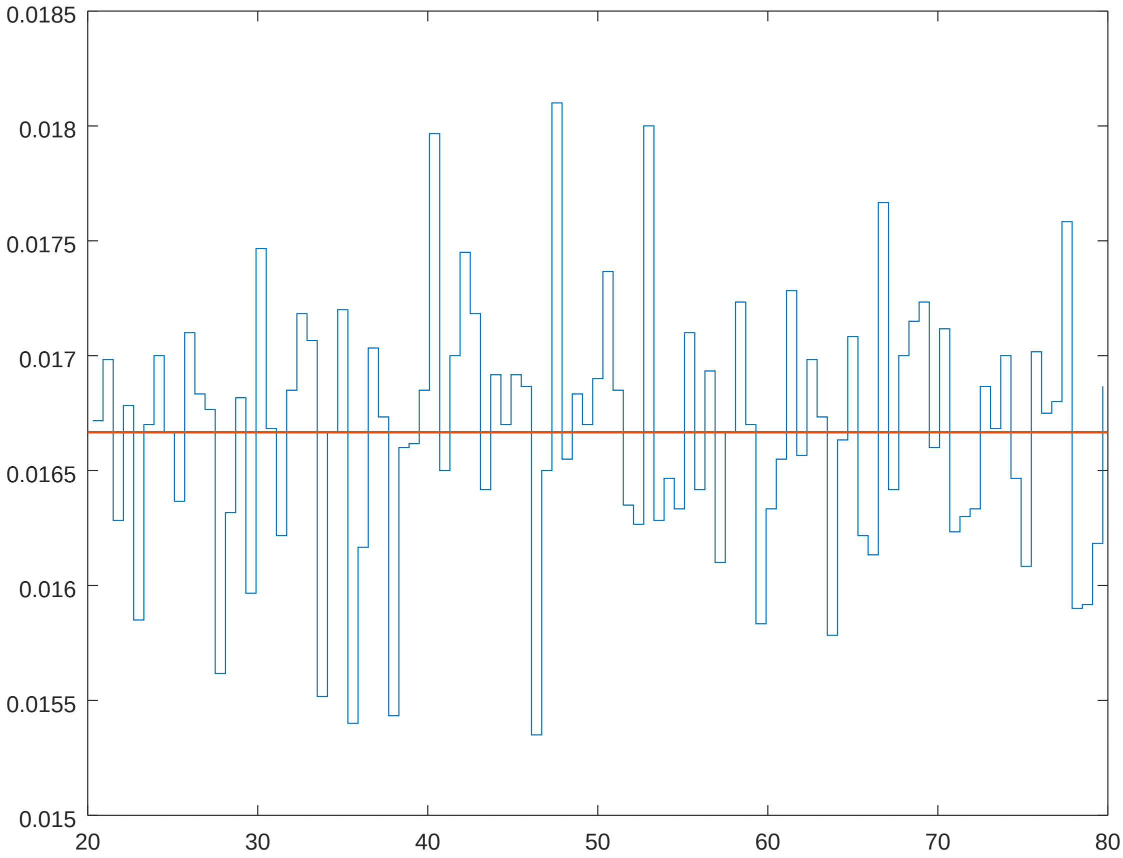
Возьмём данные из равномерного распределения и вероятности из нормального
Тогда для $\alpha = 0.05$, вероятность ошибки второго рода получается - 0

Вывод

1. Как видно по рисунку, график гистограммы хорошо приближает график плотности распределения.

2. Как и нужно, вероятность ошибки первого рода теста по критерию хи-квадрат для $\alpha = 0.05$, сходится к 0.05. Вероятность ошибки второго рода для испорченной выборки сходится к 0.


0.4
0.3
0.2
0.1
0
-4
-2
0
2
4
6
8



0.0185
0.018
0.0175
0.017
0.0165
0.016
0.0155
0.015
20
30
40
50
60
70
80