Отчёт №4

Сравнение качества оценок

Дунидин Дмитрий М3239
weeping_samael@niuitmo.ru

Романенко Демьян М3238
romanenko@niuitmo.ru

Гречишкина Дарья М3238
darya.grechishkina@gmail.com

24.03.2020

1 Задачи

Для заданных распредленений построить график плотности распределения и гистограмму по выборке из этого распределения. H_0 — тип распределения. На основе критерия хи-квадрат оценить ошибку первого рода α. Для нормального рапределения исправить интервалы так, чтобы $\forall j < m : n_j \geq 6$ и провести повторную проверку критерия. В конце испортить выборочные параметры и провести оценку ошибок второго рода β.

Входные данные

- Для построения гистограммы:
  - $m = 10^3$ — количество выборок,
  - $n = 10^6$ — число элементов в выборке;
- Для проведения тестов:
  - $k = 10^3$ — количество тестов,
  - $n = 10^4$ — число элементов в выборке;
- $\delta = 0.005$, $\Delta = 0.5$ — дельты сдвига,
- $U(a = 2, b = 4)$ — равномерное распределение,
- $U(a - \delta, b - \delta)$ — слабо испорченное равномерное распределение,
- $U(a - \Delta, b - \Delta)$ — сильно испорченное равномерное распределение,
- $N(\mu = 1, \sigma = 4)$ — нормальное распределение,
- $N(\mu - \delta, \sigma - \delta)$ — слабо испорченное нормальное распределение,
- $N(\mu - \Delta, \sigma - \Delta)$ — сильное испорченное нормальное распределение,
- $\gamma = 0.95$ — оценка.

2 Исходный код

2.1 Равномерное распределение

Listing 1: sol_for_unif.m

```
pkg load statistics

clc
clear

% U(2, 4)

function ans = test(a, b, n, m, k, gamma, delta, type)
  ans = 0;
  for i = 1:k
    unif = unifrnd(a, b, n, 1);
```

```
12      [hist_y, hist_x] = hist(unif, m);
13
14      cur_a = min(unif);
15      cur_b = max(unif);
16
17      cur_delta = (cur_b - cur_a) / m;
18      pos = cur_a;
19
20      cur_a -= delta;
21      cur_b += delta;
22
23      chi2 = 0;
24      for q = 1:m
25          pj0 = unifcdf(pos + cur_delta, cur_a, cur_b) - unifcdf(pos, cur_a,
                cur_b);
26          chi2 += (hist_y(q) - n * pj0)^2 / (n * pj0);
27          pos += cur_delta;
28      endfor
29      if (type == 1)
30        ans += (chi2 >= chi2inv(gamma, m - 3));
31      else
32        ans += (chi2 < chi2inv(gamma, m - 3));
33      endif
34    endfor
35    ans /= k;
36  endfunction
37
38  % Print Plot
39  a = 2;
40  b = 4;
41  n = 10^6;
42  m = 10^2;
43
44  unif = unifrnd(a, b, n, 1);
45  [hist_y, hist_x] = hist(unif, m);
46  [stairs_x, stairs_y] = stairs(hist_x, hist_y / n * m / (max(unif) - min(
        unif)));
47  interval = (a - 1):0.01:(b + 1);
48  plot(interval, unifpdf(interval, a, b), stairs_x, stairs_y);
49
50  % Calc Errors
51  n = 10^4;
52  m = round(n ^ (1/3)); % m ~ 22
```

```
k = 10^3;
gamma = 0.95;

delta0 = 0;
delta1 = 0.005;
delta2 = 0.5;

printf("Expected alpha: %f\n", 1 - gamma);

error = test(a, b, n, m, k, gamma, delta0, 1);
printf("Type I error alpha: %f\n", error);

error = test(a, b, n, m, k, gamma, delta1, 2);
printf("Type II error beta with delta = %f: %f\n", delta1, error);

error = test(a, b, n, m, k, gamma, delta2, 2);
printf("Type II error beta with delta = %f: %f\n", delta2, error);
```

2.2 Нормальное распределение

Listing 2: sol_for_norm.m

```
pkg load statistics

clc
clear

% N(1, 4)

function ans = calc_for_type_error(chi2, gamma, sigma, type)
  if (type == 1)
    ans = (chi2 >= chi2inv(gamma, sigma));
  else
    ans = (chi2 < chi2inv(gamma, sigma));
  endif
endfunction

function ans = test(mu, sigma, n, m, k, gamma, delta, mod, type)
  result = 0;
  for i = 1:k
    norm = normrnd(mu, sigma, n, 1);

    left = min(norm);
    curr_delta = (max(norm) - left) / m;
```

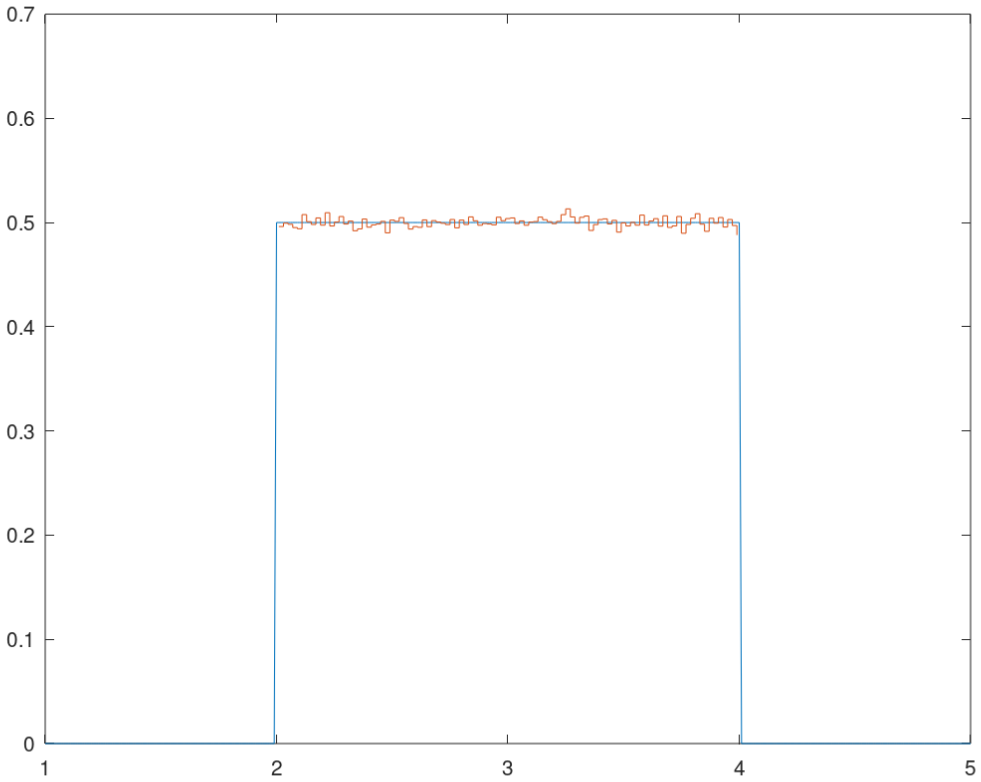
```
23      [hist_y, hist_x] = hist(norm, m);
24      curr_mu = mean(norm);
25      curr_sigma = std(norm);
26
27      if (mod == 0) % simple
28        chi2 = 0;
29        for q = 1:m
30          pj0 = normcdf(left + curr_delta, curr_mu, curr_sigma) - normcdf(
              left, curr_mu, curr_sigma);
31          chi2 += (hist_y(q) - n * pj0)^2 / (n * pj0);
32          left += curr_delta;
33        endfor
34        result += calc_for_type_error(chi2, gamma, m - 3, type);
35      else % corrected
36        curr_mu += delta;
37        parts = 0;
38        j = 1;
39        chi2 = 0;
40        while (j <= m)
41          nj = hist_y(j);
42          rt = left + curr_delta;
43          while (j < m && nj < 6)
44            j++;
45            nj += hist_y(j);
46            rt += curr_delta;
47          endwhile
48          pj0 = normcdf(rt, curr_mu, curr_sigma) - normcdf(left, curr_mu,
              curr_sigma);
49          chi2 += (nj - n * pj0)^2 / (n * pj0);
50          left = rt;
51          parts++;
52          j++;
53        endwhile
54        result += calc_for_type_error(chi2, gamma, parts - 3, type);
55      endif
56    endfor
57    ans = result / k;
58  endfunction
59
60
61  % Print Plot
62  mu = 1;
63  sigma = 4;
```

```
n = 10^6;
m = 10^2;

norm = normrnd(mu, sigma, n, 1);
[hist_y, hist_x] = hist(norm, m);
[stairs_x, stairs_y] = stairs(hist_x, hist_y / n * m / (max(norm) - min(
    norm)));
interval = min(norm):0.01:max(norm);
plot(interval, normpdf(interval, mu, sigma), stairs_x, stairs_y);

% Calc Errors
n = 10^4;
m = round(n ^ (1/3)); % m ~ 22
k = 10^3;
gamma = 0.95;

delta0 = 0;
delta1 = 0.005;
delta2 = 0.5;

printf("Expected alpha: %f\n", 1 - gamma);

error = test(mu, sigma, n, m, k, gamma, delta0, 0, 1);
printf("Type I error alpha: %f\n", error);

error = test(mu, sigma, n, m, k, gamma, delta0, 1, 1);
printf("Type I error alpha (corrected): %f\n", error);

error = test(mu, sigma, n, m, k, gamma, delta1, 1, 2);
printf("Type II error beta with delta = %f: %f\n", delta1, error);

error = test(mu, sigma, n, m, k, gamma, delta2, 1, 2);
printf("Type II error beta with delta = %f: %f\n", delta2, error);
```

3 Результат работы программ: ошибки и график

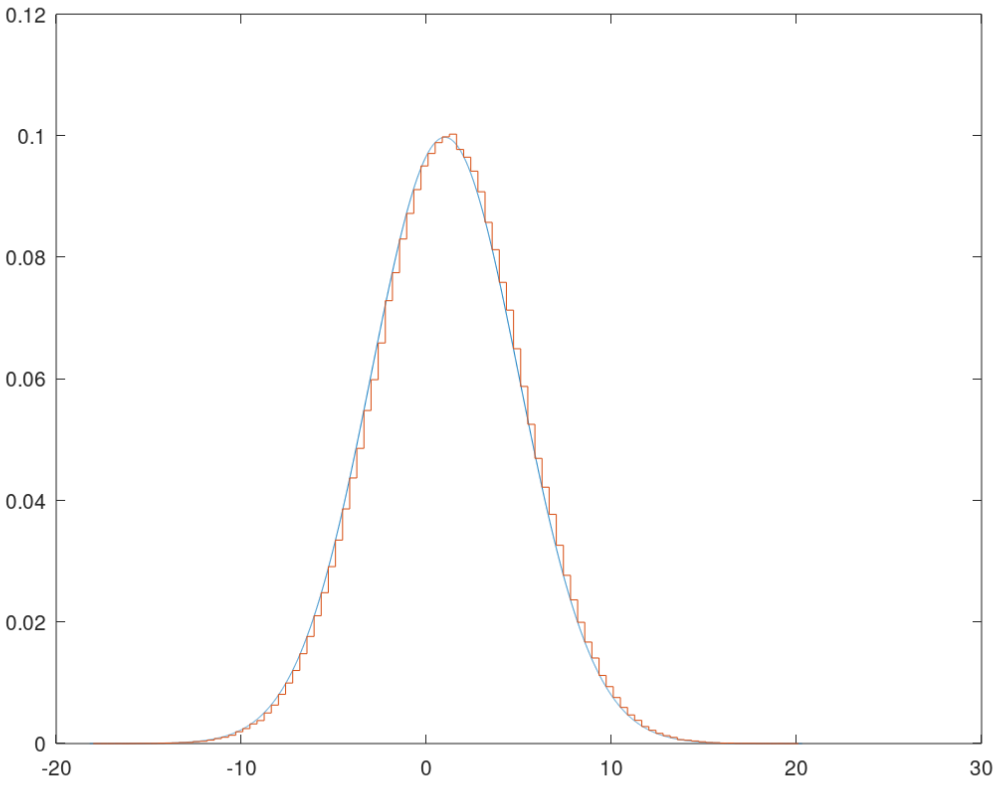
3.1 Равномерное распределение

```
Expected alpha: 0.050000
Type I error alpha: 0.074000
Type II error beta with delta = 0.005000: 0.898000
Type II error beta with delta = 0.500000: 0.000000
```



3.2 Нормальное распределение

```
Expected alpha: 0.050000
Type I error alpha: 0.135000
Type I error alpha (corrected): 0.069000
Type II error beta with delta = 0.005000: 0.913000
Type II error beta with delta = 0.500000: 0.000000
```



4 Вывод

Проверяя гипотезы по критерию хи-квадрат, установил, что значение оценки вероятности ошибки первого рода α отличает от теоретического на допустимую погрешность. По оценкам ошибки второго рода β видно, что критерий хи-квадрат оказался чувствителен только к сильному изменению ($\delta = 0.5$), почти "не реагируя" на слабое ($\delta = 0.005$).